Inventory management with dynamic Bayesian network software systems

Mark Taylor[1] and Charles Fox[2]

[1] Management School
[2] Adaptive Behaviour Research Group
University of Sheffield, UK
charles.fox@sheffield.ac.uk

Abstract. Inventory management at a single or multiple levels of a supply chain is usually performed with computations such as Economic Order Quantity or Markov Decision Processes. The former makes many unrealistic assumptions and the later requires specialist Operations Research knowledge to implement. Dynamic Bayesian networks provide an alternative framework which is accessible to non-specialist managers through off-the-shelf graphical software systems. We show how such systems may be deployed to model a simple inventory problem, and learn an improved solution over EOQ. We discuss how these systems can allow managers to model additional risk factors throughout a supply chain through intuitive, incremental extensions to the Bayesian networks.

Keywords: inventory, supply chain, risk management, Bayesian networks, economic order quantity, Markov decision process

1 Introduction

Supply chain risk management (SCRM, [6]) involves modelling and optimising the flow of goods between suppliers, customers and warehouses. Broadly construed, the scope of SCRM includes factors such as transportation risk, supplier failure and customer order models, at all levels of a supply chain. Traditionally, inventory theory has been applied at the level of the single organisation, but can now be viewed as a component of larger multi-echelon supply chain models.

Economic Order Quantity (EOQ, [3]) is still a common approach to practical inventory management, determining the order quantity of stock for a particular item [12, p. 374], and is used to determine optimal ordering quantity given a large number of assumptions [8, p. 275]. It is these assumptions that give EOQ its limitations. EOQ's limitations have previously been addressed using mathematical formulations of Markov Decision Processes, known as Stochastic Inventory Theory [11, 4, 2, 15]. These approaches, while optimal, require complex modelling and dynamic programming mathematics specific to each particular case and are not available to non-specialist managers.

The present paper provides a new approach to practical inventory and supply chain risk management using off-the-shelf Bayesian network software. This approach is highly general and may relax most of the standard assumptions made by EOQ theory. It is highly extensible, as Bayesian networks allow and encourage incremental qualitative and quantitative modelling of details about the world in an intuitive and graphical manner by non-specialist managers. Our long-term goal is to model entire supply chains and detailed risk factors with these tools, however this paper presents our initial proof-of-concept results on a standard inventory problem, then discusses extensions to larger supply chain models. We show here that Bayesian networks can first replicate the standard EOQ case which is a familiar reference point for many mangers, then show how they can extend EOQ by relaxing most of its assumptions and finding dynamic policies using reinforcement learning. These policies solve the Markov Decision Process of Stochastic Inventory Control but – unlike in previous work – the learning is wrapped inside a software package leaving the manager free to specify increasingly detailed world models with intuitive visual modelling and historical or subjective probability data. Our aim is not to introduce new mathematics but to illustrate the beginnings of a software-based methodology for highly general, integrated inventory and supply chain risk management by non-specialists.

1.1 Standard EOQ

We make a distinction between the 'EOQ world' and the 'EOQ policy'. The *EOQ world* is the set of assumptions made in EOQ theory, and various inventory policies may be tested in simulations of this world. The *EOQ policy* is the particular policy found by EOQ theory. We will also introduce a *relaxed world* which relaxes most of the assumptions of the EOQ world. (As well as the EOQ policy, we will use a reinforcement learning software system to find optimal policies in the EOQ world and in the relaxed world.)

The *EOQ world* assumes [8, p. 275]: (1) Demand is certain, constant and continuous over time; (2) The quantity ordered is constant over ordering times; (3) Ordering cost is constant and independent of quantity ordered; (4) Lead time is fixed; (5) The cost of holding a unit of stock is independent from the quantity in stock; (6) The purchase price of the item is constant; (7) There are no limits on order size.

The *optimal EOQ policy* is defined algebraically by [8, p. 277]

$$EOQ = \sqrt{\frac{2ca}{h}}, \tag{1}$$

where c is the acquisition cost of each order (i.e. the administrative and overhead costs of the order, not the price of the goods themselves); a is the annual usage in units; and h is the holding cost per unit per year.

The following is an example application of EOQ which will be referenced throughout the rest of this study. Assume a demand of $a = 6000$ items per year; potential ordering points at the start of each month; a monthly holding

cost of $h = \$0.10$ per item and an ordering cost of $c = \$15$ per order. Using the EOQ formula above, the EOQ-optimal policy is to order 1341 items, with $4.47 = a/1341$ orders per year, which gives an order every $2.7 = 12/4.47$ months.

1.2 Bayesian networks

Bayesian networks provide a formalism for reasoning about partial beliefs under conditions of uncertainty. These parameters are combined and manipulated according to the rules of probability theory [10]. Let us consider n discrete random variables $x_1, x_2, ...x_n$, a directed acyclic graph with n nodes, and suppose the jth node of the graph is associated to the x_j variable. Then the graph is a Bayesian network, representing the variables $x_1, x_2, ...x_n$, if

$$P(x_1, x_2, ...x_n) = \prod_j P(x_j|parents(x_j)),$$

where $parents(x_j)$ denotes the set of all variables x_i such that there is an arc from node x_i to x_j in the graph. The probability terms in the product are described by Conditional Probability Tables (CPTs) which may be set by hand or learned from data. Standard algorithms such as junction trees [1] exist to perform inference on Bayesian networks.

Dynamic Bayesian Networks (DBNs) allow the modelling of entities in a changing environment where the values of variables change over time [1, 9]. Functionally, DBNs capture the process of variable values changing over time by representing multiple copies of network nodes with one copy for each time step [1]. Visually, they may be displayed using two copies of each recurrent node representing the current, t, and previous, $t-1$ states.

Decision Networks (DNs) [14] – also known as 'influence diagrams' [5] – are Bayesian Networks with the addition of two types of nodes: Decision Nodes and Utility Nodes. Inferences can be made about the best actions to take to maximise utility.

Utility nodes u_i – conventionally represented as diamonds in pictures of Bayesian networks – may have multiple parents and no children. Here they have finite, integer-valued state space, and are associated with Utility Tables specifying their values as a deterministic functions of the state of their parents. The utility of a whole network is given by the sum of all utility nodes, $\sum_i u_i$. When there is uncertainty in the network, the expected value of the total utility is used, $\langle \sum_i u_i \rangle$. When used in DBNs, utility nodes are discounted by the time value of utility γ to give an expected discounted present utility objective,

$$\langle U \rangle = \sum_{t=1}^{\infty} \gamma^t (\sum_{i=1}^{N} \langle u_i \rangle).$$

Action nodes – conventionally represented as squares in pictures of Bayesian networks – are here taken to be finite discrete variables, whose values are selected

dynamically by a policy, as a function of the states of their parent nodes[3]. The goal of policy optimisation is to find the policy that maximises the expected discounted present utility.

The networks in fig. 1 and 2 are dynamic Bayesian decision networks, including recurrent connections, action and utility nodes. There are many off-the-shelf software systems that allow Bayesian networks to be constructed graphically by end-users, for example BayesiaLab (www.bayesia.com), Netica (www.norsys.com) and Hugin (www.hugin.com). The examples in this paper are produced using BayesiaLab and show a novel application to inventory and supply chain risk management.

1.3 Reinforcement learning policy

To find optimal policies in the relaxed world – as opposed to EOQ policies – our software system uses internally a form of reinforcement learning [13]. It maintains a Q-value for each (action,state) pair, where the state ranges over the combined states of the parents of the action node. These values become estimates,

$$\langle U|s_t, a_t\rangle \approx Q(s_t, a_t) \approx \langle (\sum_i u_i)_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})\rangle, \tag{2}$$

when updated by

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha((\sum_i u_i)_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})). \tag{3}$$

If Q is known exactly, or a best estimate finalised from learning, then the optimal policy is to select actions,

$$a_t = \arg_{a_t} \max Q(s_t, a_t). \tag{4}$$

Learning can be performed by drawing actions from annealed distributions based on successive estimates of Q,

$$P(a_t) = \frac{1}{Z} Q(s_t, a_t)^{1/T}. \tag{5}$$

If T is reduced sufficiently slowly over many Monte Carlo Markov Chain samples from the DBN [9], then Q converges to a locally optimal policy, with the probability of reaching a global optimal increasing with the slowness. MCMC sampling and reinforcement learning are standard DBN algorithms which may be performed with off the shelf software systems and knowledge of their details is not required by the end-user.

We emphasise that in contrast to EOQ, which computes all order sizes and times in advance, DBNs can learn dynamic policies, where the ordering action at each time is a function of the state s_t of the network at that time. Such policies can for example increase the number of orders when a backlog state is large.

[3] The present paper considers only cases where the parents of action nodes are observed. For unobserved parents the actions must be function of the *distribution* of belief rather than the state and the task is known as a Partially Observable MDP.

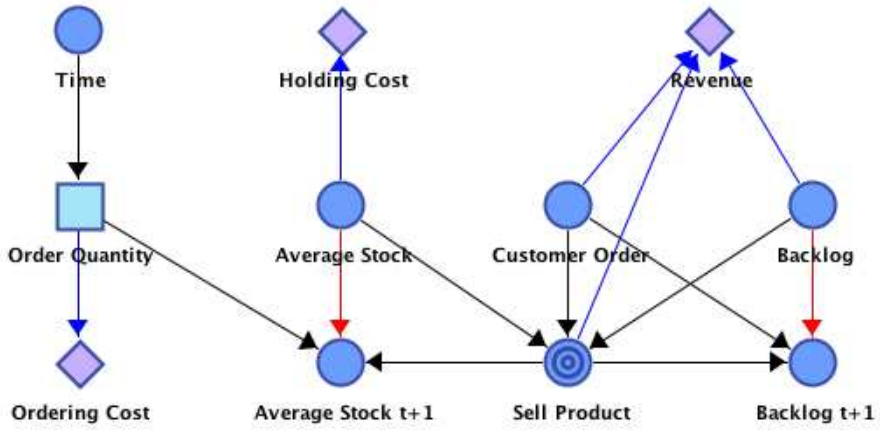

Fig. 1. Dynamic Bayesian network implementation of the EOQ world

2 Methods

We work with two Bayesian network models and two policy types. The first model is of the EOQ world, where customer demand is constant over time. The second is a relaxed world in which customer demand is a random variable conditioned on time of year. The first policy type is the EOQ policy described above. The second policy is the reinforcement learning policy, found automatically as described below. All models are created in BayesiaLab using a graphical interface to design the network, then deterministic equations or probabilities to specify the CPTs. Importantly, specialist knowledge of Bayesian inference and reinforcement learning is not required by the user, who needs only specify the world model using a graphical interface.

2.1 DBN model of the EOQ world

We first aim to demonstrate how a DBN can model the standard EOQ world. This will serve as a basis for the more complex relaxed model later. Fig. 1 shows the DBN structure. We use discrete time steps, of one month in length. Time-dependent CPTs are discretised into months, and stock quantity CPTs are discretised into integers between 1 and 20 'units' of stock, where one unit comprises 500 items.

As the demand is constant and known (6000 items per year), the *CustomerOrder* node is deterministic and constant, giving the number of customer orders, in units, at each time step to be

$$CustomerOrders = (6000/500)/12 = 1$$

which can be entered in the software as a specification of the CPT.

The number of units ordered for our own inventory each month is modelled by the *OrderQuantity*, an action node whose policy is to be learned.

There are two utility nodes modelling the ordering and holding costs, where the ordering cost is the administrative cost, \$15, of actually placing an order rather than the price of the units ordered, and it costs $-\$0.1 * 500 = -\$50$ to hold each unit,

$$OrderCost = -15$$

$$HoldingCost = -50 * AverageStock$$

The *Revenue* node models the reward for completing unit sales, being the difference between the purchase and sale price, which we set to be \$200 (this quantity is not required in standard EOQ), multiplied by the number of unit sales in the month,

$$Revenue = 200 * SellProduct$$

We model the size of the inventory and/or order backlog over time by introducing recurrent deterministic nodes,

$$AverageStock_{t+1} = AverageStock_t + OrderQuanity_{t+1}$$

$$Backlog_{t+1} = Backlog_t + CustomerOrder_{t+1}.$$

The quantity sold each month is the minimum of the stock level and the number of orders,

$$SellProduct = min(AverageStock, Backlog + CustomerOrders).$$

2.2 DBN model of a relaxed world

The relaxed world model in fig. 2 violates most of the EOQ assumptions. In particular, (1) the demand (*CustomerOrders*) are no longer constant each month but now depend stochastically on an underlying demand level which is a function of the time of year. This leads to an inherently dynamic model as this uncertainty propagates through the DBN; (2) Non-constant order quantities are now allowed, which may be functions of the backlog size; (3) the order cost utilities may now depend on order size; (4) Dynamic ordering policy allows non-fixed lead times to be modelled: if a component is not received when ordered, the backlog increases which may allow a re-order at the next step; (5) the holding cost utility may now be a function of the stock size (this is common in practice, for example filling a warehouse and needing to build a new one introduces a large non-linearity).[4] The variable costs used to illustrate (3) and (5) are listed in tables 1a and 1b.

[4] In the present network we retain (6) fixed purchase prices, although their dependency on time or random variability could easily be added to the DBN model. A new limitation introduced by the DBN formalism is that (7) we must use some upper limit on available order size, although in practice this can be made arbitrarily large, at the expense of computation time, until it becomes unimportant.

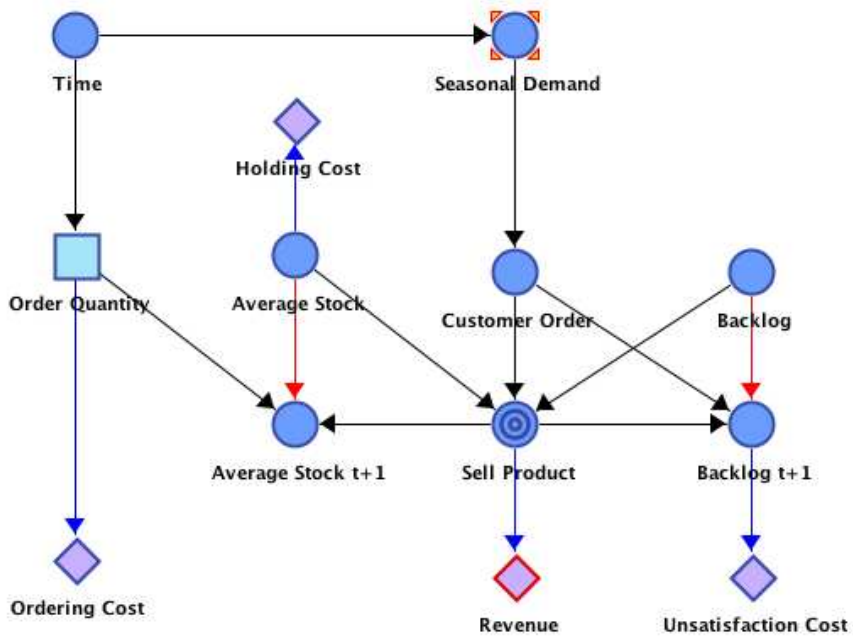

Fig. 2. Dynamic Bayesian network implementation of the relaxed world

Table 1. (a) Non-constant and dependent ordering cost; (b) Quantity dependent stock cost

Order quantity	Ordering cost
10	-15
9	-16
8	-17
7	-18
6	-19
5	-19
4	-19
3	-21
2	-22
1	-23
0	0

Average stock	Holding cost
10	-420
9	-320
8	-290
7	-260
6	-230
5	-200
4	-120
3	-90
2	-60
1	-30
0	0

3 Results

Having introduced time and backlogs into model, we first found that it was necessary to add a negative utility to model customer unsatisfaction if an order is late – this concept does not appear in the original EOQ world, and we found that it can lead to computational instabilities such as infinite backlogs if not included in the model. The *UnsatisfactionCost* node in figure 2 adds a $10 penalty for each unit-month of delay which stabilises the model.

3.1 Bayes nets can model EOQ world

To show that DBNs have the power to model the EOQ world, and to check for DBN model assumptions effects on solution optimality, we first run the standard EOQ policy on the EOQ DBN world model. If it produces similar results to the standard EOQ model then evidence is obtained that the model is good. In particular, the DBN introduces dynamic state over time, and discretizes time steps, both of which could lead to small deviations from the EOQ solution. Table 2 shows the results of the standard EOQ model against the DBN model of the EOQ world and EOQ policy. It can be seen that there is a small difference in the profit. To test whether this was due to the discretisation of time (which could be improved by using smaller time intervals in the DBN) or to other assumptions of the DBN, we computed similar results for a discretised version of standard EOQ, rounding the EOQ policy to integer months and the same order size bins as used in the DBN model. This result was very close to the original EOQ, suggesting that the rest of the discrepancy in the DBN model is due to other assumptions which do not appear in the EOQ model.

Table 2. Expected Average Profit

	Order freq.	size	cost	Hold. cost	Goods cost	Revenue	Profit
Standard EOQ model	4.47	1341.6	67.04	67.08	1200	2400	1065.88
DBN model of EOQ world	4	1500	60	100	1200	2400	1040
Discretised EOQ model	4	1500	60	75	1200	2400	1065

3.2 RL policy beats EOQ in relaxed world

Table 3 shows results from running the EOQ policy in the relaxed world, which includes stochastic customer demand, and variable holding and order costs. It also shows the results of a policy learned by reinforcement learning – built into our software system and usable by non-specialist modellers. Although the overall customer demand level is the same as the EOQ world, its fluctuations over time give rise to backlogs and unsatisfaction costs. The RL policy is able to adapt to

these fluctuations and make more profit than the rigid EOQ policy. (The overall profit is lower than in the EOQ *world* because the demand fluctuations make inventory management into a harder problem in general.)

Table 3. EOQ vs RL policy, in Relaxed world

	Holding cost	Unsatisfaction	Revenue	Profit
EOQ policy, on relaxed world	120.5	959.87	2399.63	61.28
RL policy, on relaxed world	116.15	825.60	2200.33	112.76

4 Discussion

The first set of results in table 2 show that the DBN formalism – recently made accessible to non-specialist managers through off-the-shelf software systems – has the power to model the standard EOQ world. The results show that the DBN simulation gives similar results to the discretised version of the EOQ policy, which is turn gives a close match to the performance of the continuous EOQ policy. In practice discretisation is often a real-world requirement, with stock being ordered in large standardised unit sizes. The discrepancy between the discrete EOQ math model and the DBN simulation is small, but should be explained.

We believe the discrepancy is due to latency factors relating to ordering of node updates. For example, demonstrating EOQ policy over a series of 12 time steps causes a cost discrepancy where a unit of stock is held until it can be sold fulfilling a customer order. The EOQ policy requires the ordering 3 units of stock four times per 12 time steps. Ordering of stock starts at time period 1. At time period 1 there are only two customer orders available (one customer order at time period 0, and one customer order at time period 1). This requires the third unit of stock to be held until another constant customer order is received at time period 2, at which point it is sold, and there are no units of stock left. This pattern therefore repeats every third time step and incurs additional holding cost over the EOQ model at each repetition. Latency factors could be made arbitrarily small by using smaller time steps in the model, at the expense of computation time.

Table 3 shows that EOQ policy is poor in the more realistic, relaxed world. Sticking to a predetermined, deterministic policy gives poor results when the customer orders and backlog are dynamic. In contrast, the profit is increased in this world when a dynamic, reinforcement learning policy is used, which can alter its ordering behaviour as a function of these variables.

Together, our results give a proof-of-concept example that shows how Dynamic Bayesian Network software systems can improve over standard EOQ inventory methods. Unlike previous Markov Decision Process approaches, such

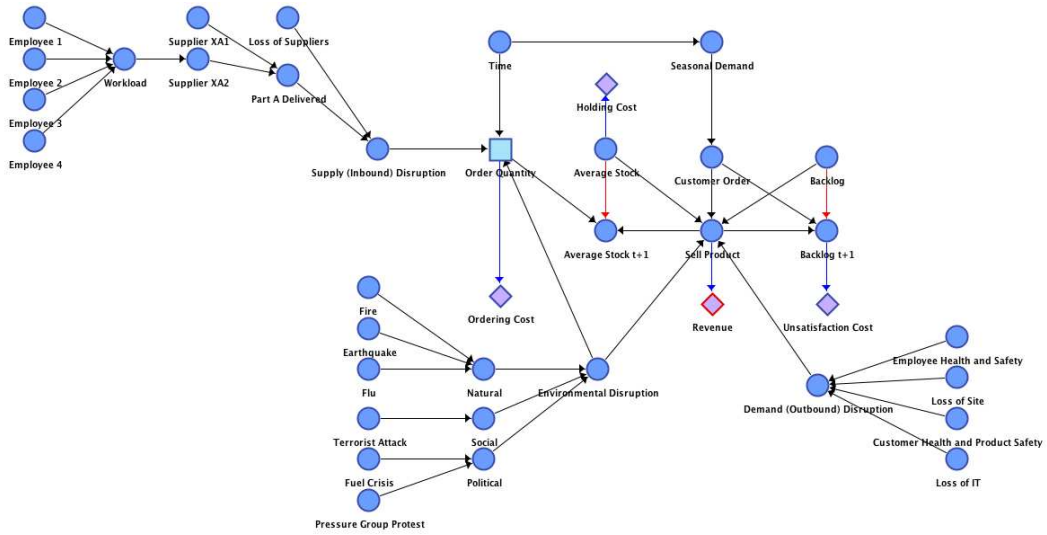

Fig. 3. Example of an extended SCRM Dynamic Bayesian network

software systems do not require the application of advanced math by end-users, rather they allow graphical representations of the Bayesian networks to be constructed by managers in an intuitive 'point and click' manner. This allows the manager to focus on constructing a realistic world model rather than on the details of inference and training algorithms. In contrast to recent fuzzy inventory approaches which solve a similar task, [7], DBNs have precise probabilistic semantics; in contrast to (s,S) inventory systems, DBNs are more general, allowing the policy to depend on factors other than the current stock level such as time of year.

We found that a problem with practical Bayesian network reinforcement learning is that it requires large amount of computation time to find good policies. The simple inventory policy used in table 3 took several hours to learn on an Apple iBook laptop, and the time typically grows exponentially with the number of parent nodes of the action nodes and the number of states in those parents. This is because the number of possible policies scales with these values and any learning algorithm needs to search amongst these policies for the best one. However for economically important decisions it is possible to buy supercomputer or cloud computing time to reduce the search time by arbitrary amounts.

A further advantage of dynamic Bayesian network software systems over existing MDP methods is the potential for extensibility of the models, both within the manager's organisation and beyond it into its supply chain partners. Bayesian networks allow incremental model refinement by replacing generic prior nodes with more detailed world models. As an example of this process, fig 3 shows an extended, supply chain risk version of the inventory model. Three sources of uncertainty have been modelled in detail: inbound, outbound, and environmental

factors. Inbound disruption can be caused by one or more suppliers failing to deliver one or more parts required to manufacture. In this example, supplier XA2 is a collaborating supply chain partner who has provided internal data about its individual employees workloads. By including this information into the Bayesian network we can made more detailed inferences about the supply side, and find better policies depending on them, such as distributing orders across multiple suppliers as a function of the dynamic internal states of those suppliers. On the outbound side, we can model potential problems with our client's state such as loss of site or IT facilities which would affect their ability to take our deliveries. Internal to our own manufacturing processes, we can model the effects of potential disruptions such as earthquakes, fuel crises or epidemics. Each of these nodes contains a CPT table, whose values can be estimated from historical data (e.g. the number of earthquakes in the location of our factory) or by experts (e.g. political analysts forecasts of terrorist risks). Unlike EOQ and standard MDP approaches to inventory, the Bayesian network formalism, as illustrated here, allows seamless integration of such detailed risk models with inventory policy, both within the organisation and across the supply chain partners. Bayesian network software systems thus provide a common language through which to integrate supply chain management, enterprise management and inventory policy, which can be spoken and used by non-specialist inventory managers through intuitive graphical tools.

Bibliography

[1] M. A. Arbib, editor. *The handbook of brain theory and neural networks*, chapter Bayesian Networks. MIT Press, 2003.

[2] S. Buffet. A Markov model for inventory level optimisation in supply chain management. *Eighteenth Canadian Conference on Artificial Intelligence*, pages 133–144, 2005.

[3] D. Erlenkotter. Ford Whittman Harris and the economic order quantity model. *Operations Research*, 38(6):937–946, 1990.

[4] I. Giannoccaro and P. Pontrandolfo. Inventory management in supply chains: a reinforcement learning approach. *International Journal of Production Economics*, 78:153–161, 2002.

[5] R. A. Howard and J. E. Matheson. Influence diagrams. *Decision Analysis*, 2(3):127–143, 2005.

[6] D. Kayne. *Managing Risk and Resilience in the Supply Chain.* BSI British Standards Institutio, 2008.

[7] D. Kofjac, M. Kljajic, A. Skraba, and B. Rodic. Adaptive fuzzy inventory control algorithm for replenishment process optimization in an uncertain environment. In *Business Information Systems 10th International Conference, BIS 2007*, pages 536–548, 2007.

[8] K. Lysons and M. Gillingham. *Purchasing and Supply Chain Management.* Prentice Hall, 6th edition, 2003.

[9] K. P. Murphy. Dynamic Bayesian networks: Representation, inference and learning. Technical report, University of California, Berkeley, 2002.

[10] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann Publishers, 2nd edition, 1988.

[11] E. L. Porteus. *Foundations of stochastic inventory theory.* Standford University Press, 2002.

[12] N. Slack, S. Chambers, and R. Johnston. *Operations Management.* Prentice Hall, Financial Times, Harlow, Essex England, 5th edition, 2007.

[13] R. Sutton and A. Barto. *Reinforcement Learning.* MIT Press, 1998.

[14] N. L. Zhang, R. Qi, and D. Poole. A computational theory of decision networks. *International Journal of Approximate Reasoning*, 11:83–158, 1994.

[15] Q. Zhao, S. Chen, S. Leung, and K. Lai. Integration of inventory and transportation decisions in a logistics system. *Transportation Research Part E*, 46:913–925, 2010.