Computers Play the Beer Game: Can Artificial Agents Manage Supply Chains?[1]

Steven O. Kimbrough
The Wharton School, University of Pennsylvania, Philadelphia, PA 19104, USA
Sok@grace.wharton.upenn.edu
D.J. Wu, Fang Zhong
LeBow College of Business, Drexel University, Philadelphia, PA 19104, USA
{Wudj, fz23}@drexel.edu

[1] Thanks to David H. Wood for many stimulating conversations and to the participants of the FMEC 2000 workshop for their comments. An earlier version of this paper appeared in Proceedings of the Thirty-fourth Annual Hawaii International Conference on System Sciences, IEEE Computer Society Press, Los Alamitos, California, 2001. Thanks for two anonymous referees of HICSS-34 for their helpful comments. This material is based upon work supported by, or in part by, DARPA contract DASW01 97 K 0007. File: DSS-KWZ.doc. Corresponding author is D.J. Wu, his current address is: 101 North 33rd Street, Academic Building, Philadelphia, PA 19104. Email: wudj@drexel.edu.

Abstract

In this study, we model an electronic supply chain that is managed by artificial agents. We investigate whether artificial agents do better than humans when playing the MIT Beer Game. Can the artificial agents mitigate the Bullwhip effect or discover good and effective business strategies? In particular, we study the following questions: Can agents learn reasonably good policies in the face of deterministic demand with fixed lead-time? Can agents cope reasonably well in the face of stochastic demand with stochastic lead-time? Can agents learn and adapt in various contexts to play the game? Can agents cooperate across the supply chain?

1. Introduction

We consider the case in which an automated enterprise has an electronic supply chain consisting of various artificial agents. In particular, we view such a virtual enterprise as a MAS (multi-agent system). Our example, for this study, of supply chain management is the MIT Beer Game [9], which has attracted much attention from both the supply chain management practitioners as well as academic researchers. There are four types of agents in this chain, sequentially arranged: Retailer, Wholesaler, Distributor, and Manufacturer. In the MIT Beer Game, as played with human subjects, each self-interested agent tries to achieve its own goal of minimizing its inventory costs in making orders to its supplier. Each agent makes its own prediction of future demand of its customer based on its own observations. We distinguish two versions of the Beer Game in the literature by labeling them as the “MIT Beer Game” and the “Columbia Beer Game” [1] or “Stationary Beer Game” [1]. The latter is a modified version of the former. These two versions differ in whether the system is stationary or non-stationary; whether demand is deterministic or stochastic; whether information delay is fixed for all players or varies among players or even stochastic; whether all players incur penalty costs or only the Retailer incurs such a cost; whether all players incur the same holding cost or the downstream player should incur higher holding cost; and whether the players know the demand distribution. Table 1 summarizes the major differences between these two versions of the Beer Game. In this paper, we focus on the MIT Beer Game first and then extend our results to the Columbia Beer Game.

The observed performance of human beings managing supply chains, whether in field or laboratory settings, is usually far from optimal from a system-wide point of view [4, 9]. This may be due to lack of incentives for information sharing, bounded rationality, or possibly the consequent of individually rational behavior that works against the interests of the group. It would thus be interesting to see if we can gain insights into the operations and dynamics of such supply chains, by using artificial agents instead of humans. This paper makes a contribution in this direction. We differ from current research on supply chains in the OM/OR area, in that the approach taken here is an agent-based, information-processing model, in an automated marketplace. While in the OM/OR literature, the focus is usually on finding optimal solutions assuming fixed environment: such as fixed linear supply chain structure, known demand distribution to all players and fixed lead time in information delay as well physical delay, it is generally very difficult to derive and to generalize such optimal policies when the environment changes.

Table 1: Comparison of the MIT and Columbia Beer Game

	MIT Beer Game	Columbia Beer Game
Customer Demand	Deterministic	Stochastic
Information Delay	Same for all players	Vary among players
Physical Delay	Fixed positive number	Fixed positive number
Penalty Cost	All players incur same penalty cost	Only retailer incurs penalty cost
Holding Cost	All players incur same holding cost	Players have decreasing costs upstream the supply chain
Demand Information	No player has knowledge about demand distribution	Demand distribution is common knowledge

We have organized the paper as follows. Section 2 provides a brief literature review. Section 3 describes our methodology and implementation of the Beer Game using artificial agents approach. Section 4 reports results from our various agent-based experiments on the MIT Beer Game. Section 5

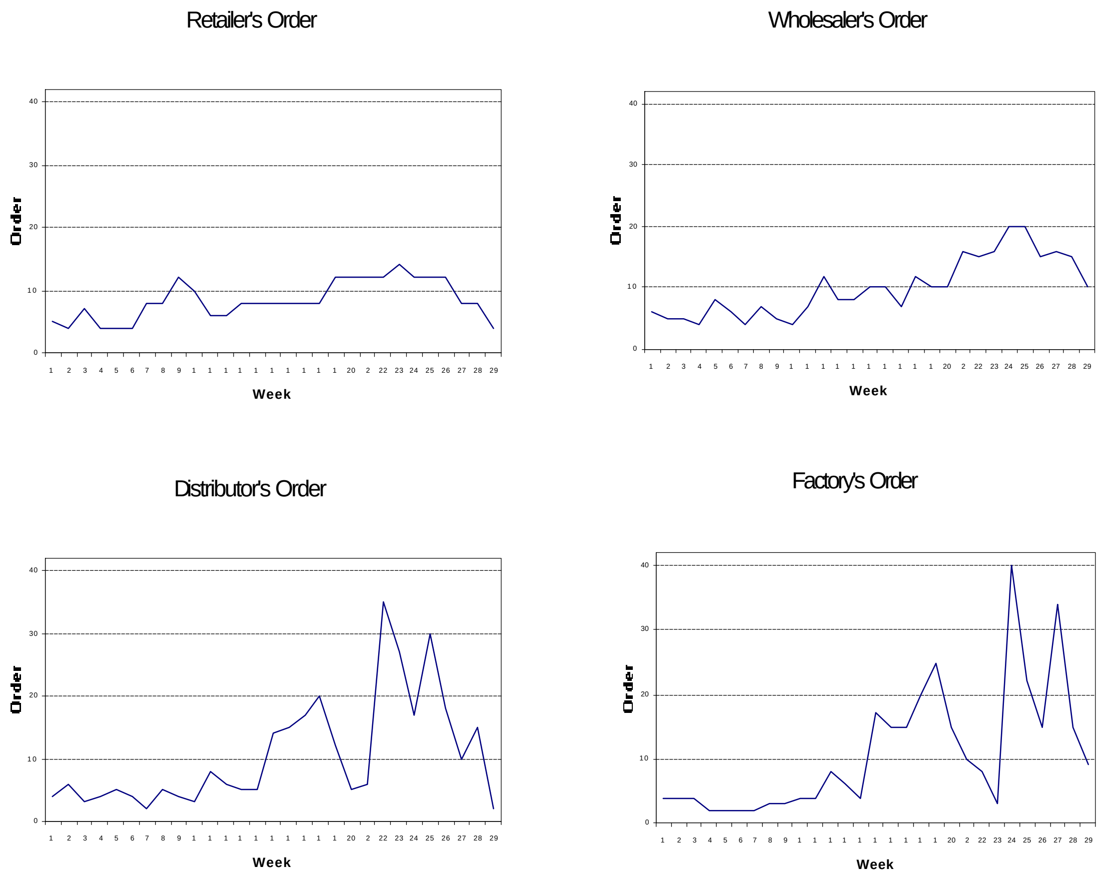


Figure 1: Observed Bullwhip effect from undergraduates game playing.

extends our experiments to the Columbia Beer Game. Section 6 summarizes our findings, offers some comments, and points to future research

2. Literature Review

A well-known phenomenon, both in industrial practice and when humans play the Beer Game, is the bullwhip or whiplash effect – the variance of orders amplifies upstream in the supply chain [3, 4]. Figure 1 illustrates the bullwhip effect using real data from a group of undergraduates playing the MIT Beer Game (date played May 8, 2000). These results are typical and have been much replicated. As a consequence, much interest has been generated among researchers regarding how to eliminate or minimize the bullwhip effect in the supply chain. In the management science literature, Lee and Whang [4] identify four sources of the bullwhip effect and offer counter actions for firms. Chen [1] shows that the bullwhip effect can be eliminated under base-stock installation policy under the assumption that all divisions of the supply chain work as a team [6]. Chen shows that base-stock is optimal when facing stochastic demand with fixed information and physical lead time. As a special case of Chen's result, we can show that when facing deterministic demand, with penalty costs for every player (The MIT Beer Game), the optimal order for every player is the so-called "Pass Order," or "One for one" (1-1) policy - order whatever is ordered from your customer.

In the MIS literature, the idea of combining multi-agent systems (MAS) and supply chain management has been proposed by several researchers [5, 7, 8, 10, 11], however, the work has mostly been limited to the conceptual level. Our approach differs from these researchers in that we focus on quantitative and agent computation methodologies, rather than purely conceptual methodological, with agents learning via genetic algorithms. We have developed DragonChain, a general platform for modeling multi-agent supply chain management, and we have implemented the MIT Beer Game in it.

In the next section, we describe in detail our methodology and implementation of the MIT Beer Game

3. Methodology and Implementation

Our main idea is to replace human players with artificial agents by playing the Beer Game and to see if artificial agents can learn to mitigate the bullwhip effect by discovering good and efficient order policies.

The basic setup and temporal ordering of events in the MIT Beer Game [9] are as follows: New shipments arrive from the upstream player (or, in the case of the Manufacturer, from the factory floor); orders are received from the downstream player (or for the Retailer, exogenously, from the end customer); the orders just received plus any backlog orders are filled; the agent decides how much to order to replenish stock; inventory costs are calculated.

Cost Function

In the MIT Beer Game, each player incurs both inventory holding costs, as well as a penalty cost if the player has a backlog. We now derive the total inventory cost function of the whole supply chain.

We begin with all needed notation definitions in this paper. N = number of players, i = 1…N. In the MIT Beer Game, N = 4; IP_i = Inventory Position of player i; C_i = Cost of the current week of player i; H_i = Inventory holding cost of player i; P_i = Penalty cost of player i; IC_i = On hand inventory at the beginning of each week; S_i = New shipment player i received in the current week; D_i = Demand received from the downstream player (for Retailer, the demand from customers).

According to the temporal ordering of the MIT Beer Game, each player's cost for a given time period, e.g., a week, can be calculated as following:

If $IP_i >= 0$ Then $C_i = IP_i * H_i$ Else $C_i = IP_i * P_{i,}$

where $IP_i = IC_i + S_i – D_i$ and S_i is a function of both information lead time and physical lead time. The total cost for the whole team after M weeks will be:

$$TC = \sum_{i=1}^{N} \sum_{j=1}^{M} C_{ij}$$

We implemented a multi-agent system, DragonChain, for the Beer Game. DragonChain is then used as a vehicle to perform a series of investigations that aim to test the performance of our agents.

The following Algorithm BEER defines the procedure for our multi-agent system to search for the optimal order policy in the Beer Game, and is further depicted in Figure 2 using a flowchart.

Algorithm BEER

1) Initialization. A certain number of rules are randomly generated to form generation 0.
2) Pick the first rule from the current generation.
3) Agents play the beer game according to their current rules.
4) Repeat step 3, until the game period (say 35 weeks) is finished.
5) Calculate the total average cost for the whole team and assign fitness value to the current rule.
6) Pick the next rule from the current generation and repeat step 2, 3 and 4 until the performance of all the rules in the current generation have been evaluated.
7) Use genetic algorithm to generate a new generation of rules and repeat steps 2 to 6 until the maximum number of generation is reached.

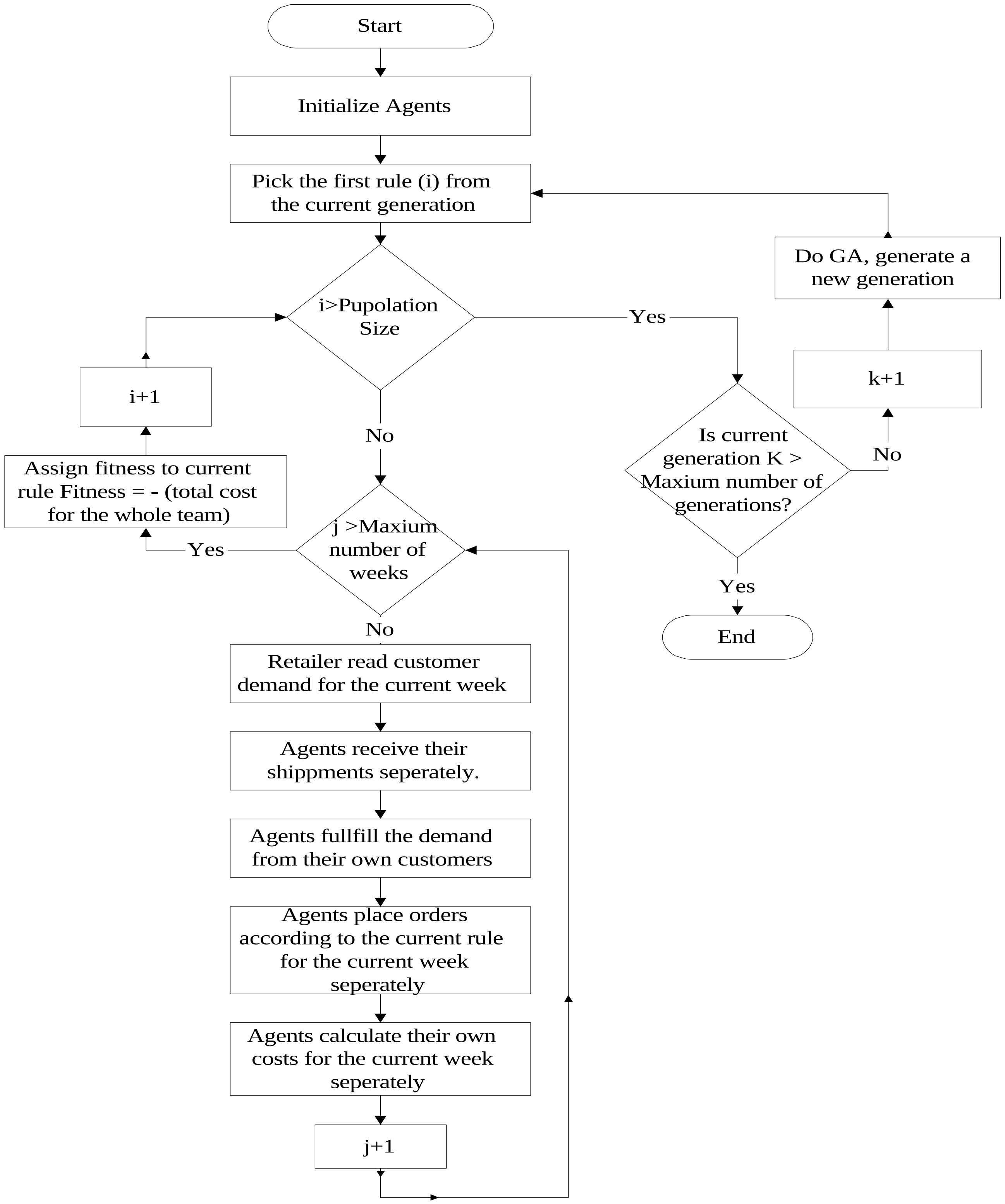
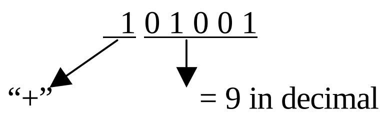

Start
Initialize Agents
Pick the first rule (i) from the current generation
i>Pupolation Size
Yes
No
Do GA, generate a new generation
k+1
No
Is current generation K > Maxium number of generations?
Yes
End
i+1
Assign fitness to current rule Fitness = - (total cost for the whole team)
Yes
j >Maxium number of weeks
No
Retailer read customer demand for the current week
Agents receive their shippments seperately.
Agents fullfill the demand from their own customers
Agents place orders according to the current rule for the current week seperately
Agents calculate their own costs for the current week seperately
j+1

Figure 2: The flowchart of the implementation of agent-based beer game.

Rule Representation

Now we describe in detail our coding strategy for the rules. In this regard, each agent's rule is represented with a 6-bit binary string. The leftmost bit represents the sign (“+” or “ – “) and the next five bits represent (in base 2) how much to order. For example, rule 101001 can be interpreted as *“x+9”:*



That is, if demand is *x* then order *x+9*. Technically, our agents are searching in a simple function space. Notice that if a rule suggests a negative order, then it is truncated to *0* since negative orders are infeasible. For example, the rule *x-15* means that if demand is *x* then order *Max[0, x-15]*. Thus the size of the search space is 2^6 for each agent.

When all four agents are allowed to make their own decisions simultaneously, we will have a “Über” rule for the team, and the length of binary string becomes 6 * 4 = 24. The size of search space is considerably enlarged, but still fairly small at 2^{24}. Below is an example of such an Über rule:

110001 001001 101011 000000

where 110001 is the rule for Retailer;

001001 is the rule for Wholesaler;

101011 is the rule for Distributor;

000000 is the rule for Factory.

Rule Learning

Agents learn rules via a Genetic Algorithm (GA, see, e.g., [2]). The following describes our specific implementation of the genetic algorithm.

1) Absolute Fitness function: the negative of total cost under (Über) rule *r*, *F = - TC(r)*.

2) Selection mechanism: proportional to fitness.

3) Genetic operators: standard crossover and mutation operators.

The Execution Cycle

The following is the GA execution cycle:

Step 1: A relative fitness is assigned for each rule in the current generation *t* according to its absolute fitness by the following equation

$$RF_i = \frac{AF_i - \min\{AF_1,\ AF_2 ... AF_n\}}{\max\{AF_1,\ AF_2 ... AF_n\} - \min\{AF_1,\ AF_2 ... AF_n\}}$$

where RF_i is the relative fitness of the *i*th rule in the generation *t*, and AF_i is the absolute fitness of the *i*th rule in the generation *t*.

Step 2: Using this probability distribution to select *n* pairs of strings, *2n* = population size of each generation, and make copies of them.

Step 3: Apply crossover to each copied pair according to the crossover rate. And replace the old strings with the new strings.

Step 4: Apply mutation to the current population of rules.

Step 5: Set *t* to *t + 1* and return to step 1.

4. Experimental Results

In this section, we report multiple rounds of experiments conducted using DragonChain. In the first round of our experiments, we tested agent performance under deterministic demand as set up in the classroom MIT Beer Game. In the second round, we explored the case of stochastic demand where demand is randomly generated from a known distribution, e.g., uniformly distributed between [0, 15].

In the third round of experiments, we changed lead-time from fixed 2 weeks to uniformly distributed between 0 to 4. Finally, we conducted two more experiments with 8 players under both deterministic and stochastic demands. We now briefly describe each round of experiment and our findings.

Experiment 1. In the first round of our experiments, we tested agent performance under deterministic demand as set up in the classroom MIT Beer Game. The customer demands 4 cases of beer in the first 4 weeks and then demands 8 cases of beer per week starting from week 5 and continuing until the end of the game (35 weeks). We fixed the policies of three players (e.g., the Wholesaler, Distributor and Manufacturer) to be 1-1 rules, i.e., if current demand is *x* then order *x* from my supplier, while letting the Retailer agent to adapt and learn. The goal of this experiment is to see whether the Retailer agent can match the other players' strategy and learn the 1-1 policy.

Our initial results show that DragonChain can learn the 1-1 policy consistently. With a population size of 20, the Retailer agent finds and converges to the 1-1 policy very quickly.

We then allowed all four agents make their own decisions simultaneously (but jointly, in keeping with the literature's focus on teams). In the evaluation function, each agent places orders according to its own rule for the 35 weeks and current cost is calculated for each agent separately. But for Wholesaler, Distributor and Manufacturer, their current demand will be the order from its downstream partner, which are Retailer, Wholesaler and Distributor. The four agents act as a team and their goal is to minimize their total cost over a period of 35 weeks. The result is that each agent in DragonChain can find and converge to the 1-1 rule separately, as shown in Figure 3. This is certainly much better than human agents (MBA and undergraduate students, however, undergrads tend to do better than MBAs), as shown in Figure 4.

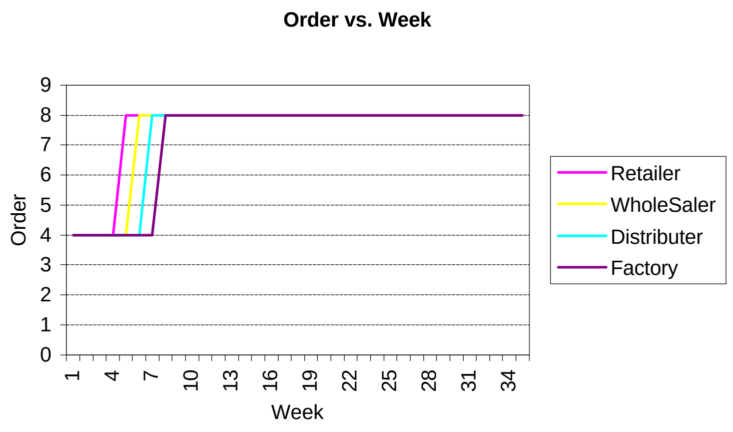


Figure 3: Artificial agents in DragonChain are able to find the optimal "pass order" policy when playing the MIT Beer Game.

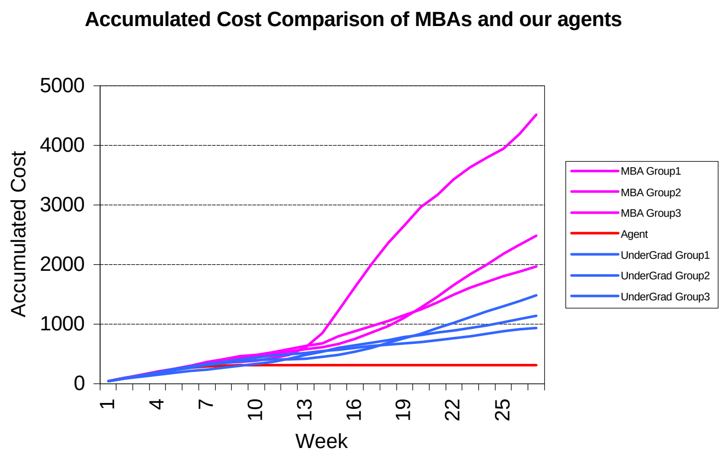


Figure 4: Artificial agents in DragonChain perform much better than MBA and undergraduate students in the MIT Beer Game.

We conducted further experiments to test one for one is a Nash equilibrium. The result was that each agent finds one for one and it is stable, therefore it apparently constitutes allowing agents to search a much large space. We introduced more agents into the supply chain. In these experiments, all the agents discover the optimal one for one policy, but it takes longer to find it when the number of agents increases. The following table summarizes the findings.

Table 2: Convergence speed when having different numbers of agents

No. of Agents	Four	Five	Six
Convergence generation	4	11	21

<u>Experiment 2.</u> In the second round of experiments, we tested the case of stochastic demand where demand is randomly generated from a known distribution, e.g., uniformly distributed between [0, 15]. The goal of this experiment was to find whether artificial agents can track the trend of demand, whether artificial agents can discover good dynamic order policies. Again we used 1-1 as a heuristic to benchmark. Our initial experiments show that DragonChain can discover dynamic order strategies (for example, each agent uses the *x + 1* rule) that outperform 1-1, as shown in Figure 5. Furthermore, as shown in Figure 6, notice an interesting and surprising discovery of our experiments is that, when using artificial agents to play the Beer Game, in all test cases, there is no bullwhip effect, even under stochastic customer demand.

As in experiment 1, we conducted a few more experiments to test the robustness of the results. First, the Nash property remains under stochastic demand. Second, we increased the game period from 35 weeks to 100 weeks, to see if agents can beat one for one over a longer time horizon. The customer demand remained uniformly distributed between [0, 15]. This time, agents found new strategies (*x+1, x, x + 1, x*) that outperform the one for one policy. This is so, because the agents seem to be smart enough to adopt this new strategy to take advantage of the end effect of the game (more on this is the concluding section).

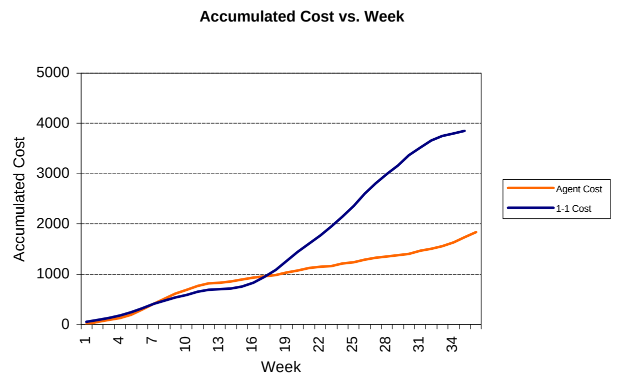


Figure 5:When facing stochastic demand and penalty costs for all players as in the MIT Beer Game, artificial agents seem to discover a dynamic order policy that outperforms “1-1’.

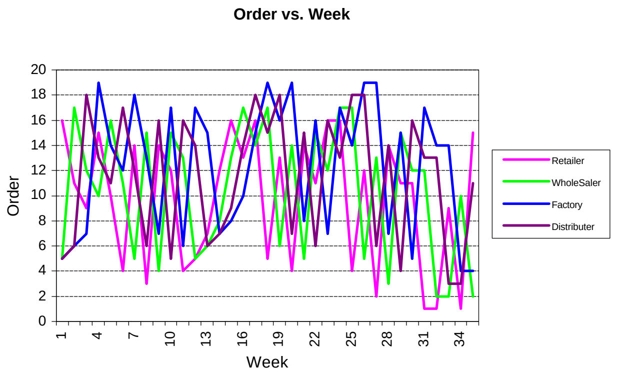


Figure 6: The bullwhip effect is eliminated when using artificial agents to replace human beings in playing the MIT Beer Game.

Table 3: Variance comparison for experiment 2.

	Retailer	Wholesaler	Distributor	Manufacturer
Var.	26.08	26.13	25.26	26.87

Third, we tested for statistical validity by rerunning the experiment with different random number seeds. The results were robust.

<u>Experiment 3.</u> In the third round of experiments, we changed lead-time from fixed 2 weeks to uniformly distributed from 0 to 4. So now we are facing both stochastic demand and stochastic lead-time.

Following the steps we described in the first experiment, the agents find strategies better than 1-1 within 30 generations. The best rule agents found so far is *(x + 1, x + 1, x + 2, x + 1)*, with a total cost for 35 weeks of 2555. This is much less than 7463 obtained using the 1-1 policy. Table 4 shows the evolution of effective adaptive supply chain management policies when facing stochastic customer demand and stochastic lead-time in the context of our experiments. We further tested the stability of these strategies discovered by agents, and found that they are stable, thus constitute an apparent Nash equilibrium.

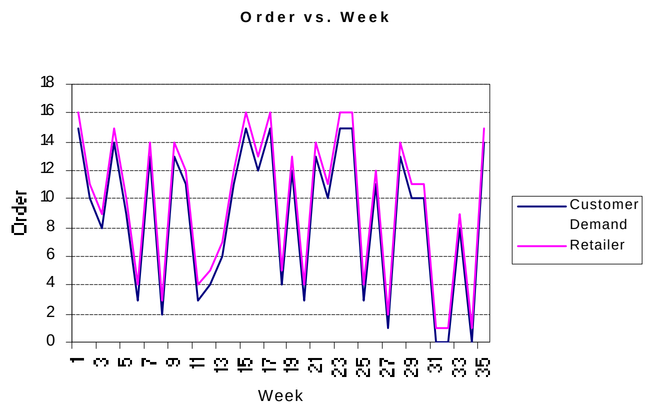


Figure 7: Artificial Retailer agent is able to track customer demand reasonably well when facing stochastic customer demand.

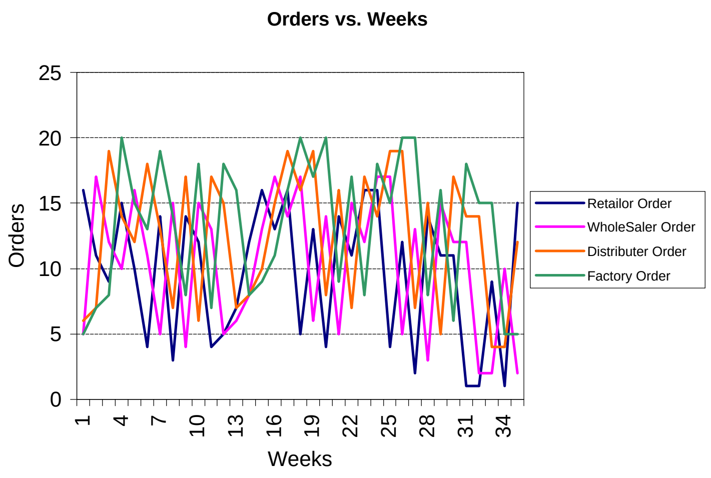


Figure 8: Artificial agents mitigate the bullwhip effect when facing stochastic demand and stochastic lead time.

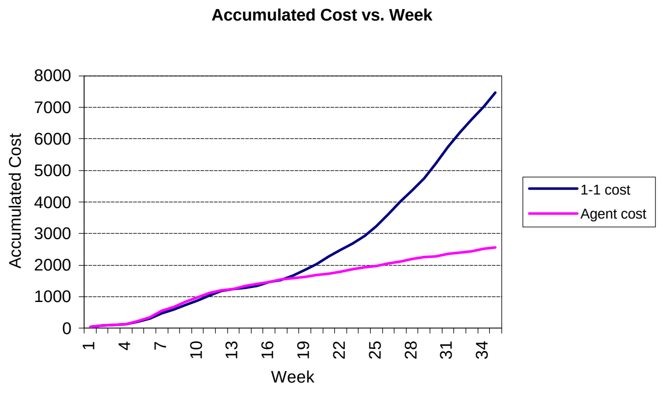


Figure 9: Artificial agents seem to be able to discover dynamic ordering policies that outperform the 1-1 policy when facing stochastic demand and stochastic lead time.

Table 4: The evolution of effective adaptive supply chain management policies when facing stochastic customer demand and stochastic lead-time.

Generation	Strategies of Agents				Total Cost
	I	II	III	IV	
0	x – 0	x – 1	x + 4	x + 2	7380
1	x + 3	x – 2	x + 2	x + 5	7856
2	x – 0	x + 5	x + 6	x + 3	6987
3	x – 1	x + 5	x + 2	x + 3	6137

4	x + 0	x + 5	x – 0	x – 2	6129
5	x + 3	x + 1	x+ 2	x + 3	3886
6	x – 0	x + 1	x + 2	x + 0	3071
7	x + 2	x + 1	x + 2	x+ 1	2694
8	x + 1	x + 1	x + 2	x + 1	2555
9	x + 1	x + 1	x + 2	x + 1	2555
10	x + 1	x + 1	x + 2	x + 1	2555
11	x + 1	x + 1	x + 2	x + 1	2555
12	x + 1	x + 1	x + 2	x + 1	2555
13	x + 1	x + 1	x + 2	x + 1	2555
14	x + 1	x + 1	x + 2	x + 1	2555
15	x + 1	x + 1	x + 2	x + 1	2555
16	x + 1	x + 1	x + 2	x + 1	2555
17	x + 1	x + 1	x + 2	x + 1	2555
18	x + 1	x + 1	x + 2	x + 1	2555
19	x + 1	x + 1	x + 2	x + 1	2555
20	x + 1	x + 1	x + 2	x + 1	2555

Further Experiments: The main challenge to computational investigation is the complexity of the search spaces when we use more realistic distribution networks. In most of above experiments, we have a small search space: 2^{24}. A simple extension to 8 players, however, quickly produces an apparently nontrivial search space of 2^{48}. So we conducted two further experiments with 8 players under both deterministic and stochastic demands. For the deterministic demand, all the agents find the known optimal policy of 1-1 at around the 80th generation with a population size of 2000. And for the stochastic demand, agents found a better policy (Total cost under this policy is 4494) than 1-1 (Total cost under 1-1 is 12225) at around the 190th generation, with a population size of 3500.

5. The Columbia Beer Game

We experimented with the Columbia Beer Game along the same settings as in Chen [1], except that none of our agents has knowledge about the distribution of customer demand. As an example, we set information lead time to be (2, 2, 2, 0), physical lead time to be (2, 2, 2, 3) and customer demand as normally distributed with mean of 50 and standard deviation of 10.

Since different players might have different information delay in the Columbia Beer Game, we modified our string representation to accommodate this addition. To do so, we added two bits into the bit-string. For example, if the Retailer has a rule of 10001101, it means “if demand is *x* in week *t-1*, then order *x+3*”, where *t* is the current week:

$$\underbrace{1}_{\text{“+”}}\;\underbrace{0\,0\,0\,1\,1}_{3}\;\underbrace{0\,1}_{\text{“ } t-1\text{”}}$$

Notice that the master rule for the whole team now has the length of 8 * 4 = 32. Under the same learning mechanism as described in the previous sections, agents found the optimal policy of Chen (1999): order whatever amount is ordered with time shift, i.e., $Q_1 = D\ (t\text{-}1)$, $Q_i = Q_{i-1}\ (t - l)$.

6. Summary and Further Research

This study makes the following contributions. First and foremost, we have explored the concept of an intelligent supply chain managed by artificial agents. We have found that agents are capable of playing the Beer Game effectively. They are able to track demand, eliminate the Bullwhip effect, discover the optimal policies (where they are known), and find good policies under complex scenarios where analytical solutions are not available. Second, we have observed that such an automated supply chain is adaptable to an ever-changing business environment. The principal results we obtained, in support of these contributions, are as follows:

1. In the cases studied for which optimal order policies are known analytically (deterministic or stochastic demand, deterministic lead times), the artificial agents rather quickly found the optimal rules.

2. In the cases for which optimal order policies are known analytically (deterministic or stochastic demand, deterministic lead times), we performed additional experiments and found that the optimal solutions are also (apparently) Nash equilibria. That is, no single agent could find an alternative order policy for itself that was better for itself than the policy it had in the team optimum, provided the other team members also used the team optimum policies. This is a new result, one that extends what was known analytically. It has intriguing consequences for applications; development of that point, however, is beyond the scope of this paper.

3. In certain cases for which optimal order policies are not known analytically (stochastic demand and lead times, with penalty costs for every players), the artificial agents rather quickly found rules different from rules known to be optimal in the simpler cases (deterministic lead times).

4. In these cases (for which optimal order policies are not known analytically), we performed additional experiments and found that the apparently optimal solutions found by the agents are also apparently Nash equilibria. That is, no single agent could find an alternative order policy for itself that was better for itself than the policy it had in the team apparent optimum, provided the other team members also used the team apparent optimum policies. This is a new result, one that extends what was known analytically; again, there are intriguing consequences for practice.

5. In repeated simulations with stochastic demands and stochastic lead times, we paired the apparently optimal rules found by the artificial agents with the 1-1 rules (known to be optimal

in the simpler situations). We consistently found that the agents performed better than the 1-1 rules. Thus, we have shown that analytic optimal results do not generalize and that in the generalization the agents can find good (if not optimal) policies.

A few remarks to put our results in their proper perspective. Our agents are searching in rather simple function spaces. The functions in the spaces available to our agents are functions only of the demands seen by the agents (either in the present period or in some previous period). Consequently, the agents are not allowed to choose rules that are, for example, sensitive to its inventory position. Clearly, in these circumstances and in the long run, only some version of a 1-1 rule can be optimal; otherwise the agents will either build up inventory without bound or will essentially eliminate it, thereby incurring excessive penalties. How long is long? Under the assumptions of the Beer Game, we have seen that 36 weeks is not long enough. Extending the play to 100 weeks is, our simulations indicate, also not long enough. For practical purposes, it may well be that, say because of inevitable changes in the business environment, only the short run characteristics of the system are of real interest. If so, then real agents may well find the discoveries of artificial agents such as ours most interesting. One way of interpreting why, in the short run, our agents deviate from a 1-1 policy is this. The agents are searching for rules that yield, over the life of the simulation, the best levels of inventory. Non-1-1 rules emerge when the starting inventory levels are non-optimal and the time horizon is limited. As theoreticians, we leave it to others to determine whether businesses ever encounter such situations: inventory on hand that is not known to be at an optimal level and a less than infinitely long planning horizon confronts us.

We are actively continuing these investigations. Several items are high on our agenda. We want to scale up the systems studied to include supply networks (rather than linear supply chains). We plan to continue using an agent-based approach, and hope to use standard computers (“silicon”) as well as

DNA computing. Once we consider realistically large supply networks and function spaces, the computational complexities will explode. One reason we are looking to DNA computing to be possibly of help here is the prospect of having truly huge generations participate in “genetic” learning. In general, the tradeoff in genetic learning between population size and number of generations is not well understood. Will we learn more with a population size of 1000 running for 1000 generations, or with a population size of 100000 running for 10 generations? The number of fitness evaluations (typically the main computational cost in standard "silicon" computing) is the dominant source of cost, and this is constant for the example just given. DNA, and more generally, molecular computing offers the possibility of economically having huge population sizes, but comparatively small numbers of generations.

We also aim to investigate: the optimal structure of the supply chain; the value of information sharing in supply chains (and networks); the emergence of trust in supply chains; mechanisms to induce coordination, cooperation and information sharing among agents in an electronic supply chain when facing automated electronic markets; and the study of alternative agent learning mechanisms such as classifier systems.

The work reported here is just a start, but it does provide some basis for optimism.

6. References

1. Chen, F., “Decentralized supply chains subject to information delays,” *Management Science*, 45, 8 (August 1999), 1076-1090.
2. Holland, John. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The MIT Press, Cambridge, 1992.
3. Lee, H., Padmanabhan, P., and Whang, S. “Information distortion in a supply chain: The bullwhip effect”, *Management Science*, 43, 546-558, 1997.
4. Lee, H., and Whang, S. “Decentralized multi-echelon supply chains: Incentives and information”, *Management Science*, 45, 633-640, 1999.
5. Lin, F., Tan, G., and Shaw, M. “Multi-agent enterprise modeling”, forthcoming in, *Journal of Organizational Computing and Electronic Commerce*.
6. Marschak, J., R. Radner. 1972. *Economic Theory of Teams*. Yale University Press, New Haven, CT.
7. Nissen, M. “Supply chain process and agent design for e-commerce”, in: R.H. Sprague, Jr., (Ed.), *Proceedings of The 33rd Annual Hawaii International Conference on System Sciences*, IEEE Computer Society Press, Los Alamitos, California, 2000.
8. Sikora, R., and Shaw, M. “A multi-agent framework for the coordination and integration of information systems”, *Management Science*, 40, 11 (November 1998), S65-S78.
9. Sterman, J. “Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment”, *Management Science*, 35, 321-339, 1989.
10. Strader, T., Lin, F., and Shaw, M. “Simulation of order fulfillment in divergent assembly supply chains”, *Journal of Artificial Societies and Social Simulations*, 1, 2, 1998.
11. Yung, S., and Yang, C. “A new approach to solve supply chain management problem by integrating multi-agent technology and constraint network”, in: R.H. Sprague, Jr., (Ed.), *Proceedings of The 32nd Annual Hawaii International Conference on System Sciences*, IEEE Computer Society Press, Los Alamitos, California, 1999.