Appendix A :

Supplement to Accompany the Article

Simpler is (Sometimes) Better

A Comparison of Cost Reducing Agent Architectures in a Simulated Behaviorally-Driven Multi-Echelon Supply Chain

Table of Contents

Table of Figures

Table of Tables

Model and Code Availability

All code used to produce the results in the main article are available at:

https://github.com/jpain3/Bullwhip-Policy-Architectures

This includes the code, in Python, used to train the DQN structures described in the original article, along with code, in R, used to apply those trained TensorFlow objects. Note that the DQN was originally trained using TensorFlow (Abadi et al., 2016) version 2.3.0. The DQN model objects *must* be used in an environment that similarly uses TensorFlow version 2.3.x or is backwards compatible with objects trained in that environment. The Python scripts also includes the custom OpenAI gym environment (Brockman et al., 2016) used to train the DQN agent. To train the agent, the package Keras-RL2 (McNally, 2019) was used as a front-end api manager for TensorFlow. Keras-RL2 is an extension of the Keras package (Chollet, 2015) and was used specifically because it better implements the dueling reward function structure central here (Wang et al., 2016).

To support use of these objects trained objects and to replicate the training process, the code repository includes yaml objects as well, which contain snapshots of the supporting packages and similar supporting infrastructure used in Python while training these DQN agents. Note that while these yaml objects do include all necessary packages for recreating the training environment, they may contain superfluous packages as well. The author has attempted to trim down these unnecessary packages but makes not guarantees.

The R scripts include self-contained functions to simulate the Beer Game over a given time horizon with variable values of information delays, shipping delays, costing, and order input types. All R scripts include code at the beginning to install any needed packages that are beyond the vanilla installation of R. Note that these scripts were primarily developed and run using RStudio as the IDE (RStudio Team, 2020), and may contain references to graphical objects (such as progress bars or window status values) that may not be pertinent in all environments, especially headless clusters or similar decentralized computing systems.

The code is intended to act as a flexible framework for future research and allows for multiple models of human ordering based on prior literature to be substituted into each position in the supply chain. As of this publication, the framework supports the following ordering schemes:

- Base-Stock replenishment – This does not calculate the optimal base stock policy like that seen in (Clark & Scarf, 1960) but rather follows a fixed replenishment policy based on a given bast-stock value for the position in the supply chain
- (Sterman, 1989) – This is the mechanism used in the main article, and follows a four-parameter ordering scheme with anchoring and adjustment of expectations of future ordering. This ordering scheme is also derived most directly from the context in which the real-world runs of the Beer Game on which this paper is built were derived.
- (Sterman & Dogan, 2015) – This paper was based on stationary and known orders, and introduces a more complex rule built on other similar research (Croson et al., 2014) and allows for the desired supply line to shift over time.
- (Oliva et al., 2022) – All four variants of the model utilized in this paper are available as options in the framework here, but models 3 and 4 notably vary from the (Sterman, 1989) model described above in that they allow the response to differ when agents are in a backlog state.

For all the models described above, the framework allows for entity-level parameters to be supplied (like those fitted to real world ordering behavior for the Sterman ’89 rule used in the main article). If no parameters are supplied, the code utilizes the ‘average’ or ‘baseline’ or ‘best-fit’ values reported in the corresponding original paper.

Order Data Availability

For the 49 simulated teams used throughout the main article as a testing bed for the agents, 11 come directly from the original presentation of the four parameter ordering rule used throughout the analyses here, and 1 additional team modeled on ‘average’ performance of those other 11 (Sterman, 1989), for 12 historic teams. These teams do not have specific order traces, and instead were presented as estimated models using the four-parameter ordering rule in Sterman ’89. Order traces for real runs of the game were obtained from online runs of the Beer Game at MIT as part of various executive and graduate-level classes. These runs occurred in twice in August of 2021, with 12 teams in one run and 22 teams in another run, and in June of 2022 for an additional 3 teams.

For these more recent runs, order traces are available with the team names and exact dates of the games obfuscated for individual privacy. Additionally, these teams have been fit to the Sterman ’89 ordering rule to provide the total 49 teams used in the main paper. The code used to perform this fit is also available at the repository linked above.

Applicability to Alternative Ordering Rules

The framework allows for the cost-reducing agent to be placed in the supply chain in any of the positions, and for its ordering rules to be defined separately from those used by the other entities. Thus, a DQN agent can be placed in a supply chain run by Sterman ’89 behaving agents (as in the original article), or base-stock agents, or any other of the available ordering schemes. For the model-predictive learning agent, this is taken a step further, and the assumption of the agent about its surroundings can be further defined. For simplicity of exposition, the in main article the core architecture of the agent and its assumptions about its environment were kept the same, with a base-stock responding agent assuming base-stock responses from the other agents. This follows from the idea that for an agent to assume that a base-stock response would be near optimal it must also assume the other agents around it are behaving similarly. Conversely, if the agent itself is using a behavioral response model this presupposes that the agent is assuming behavioral responses from its peers in the supply chain.

However, while this matching of architecture and assumptions makes intuitive sense, it can be relaxed in the framework developed here, and the agent can assume any one of the above listed ordering rules are being used by other entities in the supply chain, and in turn use any of those rules (plus the DQN structures) to respond. Furthermore, the underlying reality that the agent is placed in, e.g. the *actual* rules being followed by the other entities can take on any of the above forms and does not have to match the assumptions the agent is making.

While not the focus on the main article, this framework also allows for some additional observations, namely that the model-predictive learning method can perform well even when making fundamentally flawed assumptions about its environment. Table S1 shows the results from placing the model-predictive learning agent (with a calibration memory of 10 time steps and a forward horizon of 30 timesteps as in the main article) that assumes the other entities int the supply chain are following the Sterman ’89 ordering heuristic. In reality, they are following Model 3 from Oliva et. .al ’22, a related but still fundamentally different ordering rule.

Table S1. Model-Predictive Learning Agent Assuming Sterman '89 but in Oliva et al '22 Average Model 3 Environment subject to Step Input

		Model-Predictive Learning Agent
	Baseline Costs	26257
Agent Position	1 (Retailer)	869 (-96.7%)
	2 (Wholesaler)	1040 (-96.0%)
	3 (Distributor)	2859 (-89.1%)
	4 (Factory)	12274 (-53.3%)

Model-Predictive Learning Agent Hyperparameters

The model-predictive learning agent introduced in the main article has several hyperparameter assumptions that are implied by the pseudocode used in the second that introduces that agent. In addition to the assumptions used to model the environment in which it resides, which is discussed in this Appendix in the sections above, this agent also has a calibration memory over which to fit an estimate of that model, and an optimization horizon over which to plan based on that calibrated model.

Calibration Memory and Optimization Horizon

In the main paper, all results for the model-predictive learning agent are based on a calibration memory of 10 units of time and a forward optimization horizon of 30 units of time. These numbers are somewhat heuristically chosen based on multiple trials of the agent during development, and also somewhat from support in related literature (notably the memory of 10 units used in the semi-optimal baseline developed in (Moritz et al., 2021)). However, some of the trade offs from choosing differing values of these hyperparameters can be directly explored. As discussed above, the choice of position 2 (wholesaler) in the supply chain was chosen for this analysis in an attempt to isolate the main feature choices of the agents versus other idiosyncrasies. The hyperparameter choices are largely muted at this position in the supply chain as well.

However, when a model-predictive learning agent is placed at the beginning of the supply chain, in position 1 (retailer) and exposed to non-stationary order inputs then the

influence of these hyperparameters, especially the forward optimization horizon, can be more significant. Figure S1 shows the total team costs incurred in the simulated four entity supply chain, all using the average ordering rule reported in Sterman ’89, with a model-predictive learning agent placed at position 1, the retailer, and exposed to either deterministically linearly increasing orders, or to noisily increasing orders. Figure S2 zooms in on the specific case of the optimization horizon equaling 35 in the noisy non-stationary case to illustrate how the orders being placed by the agent in position 1 influence the overall inventory positions taken during the simulation.

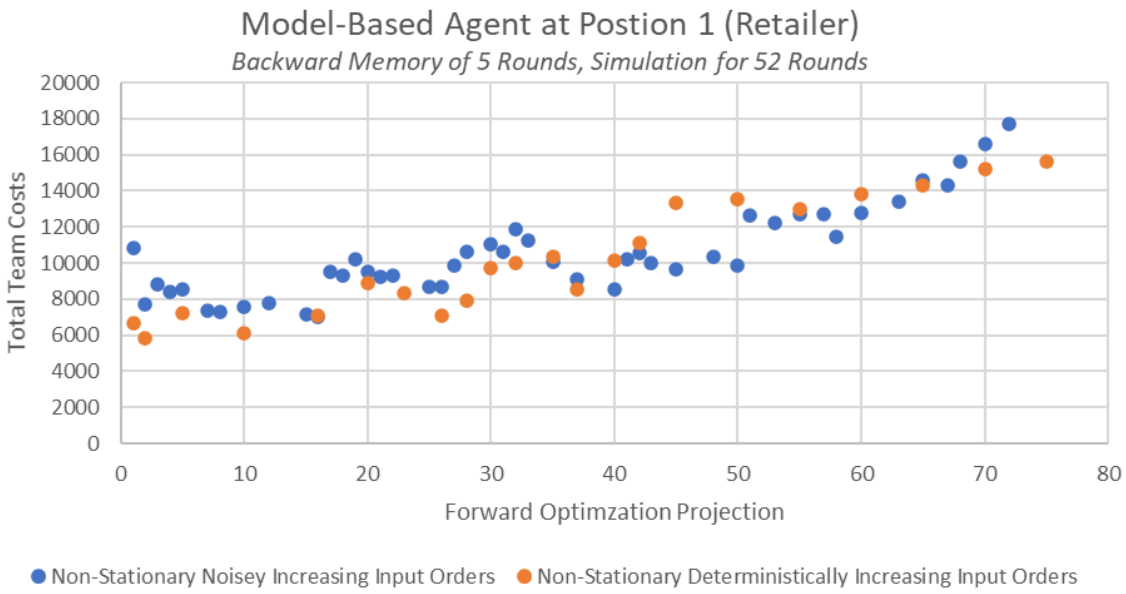

Figure S1: Total Team Costs With MP Agent at Position 1 with Non-Stationary Increasing Orders

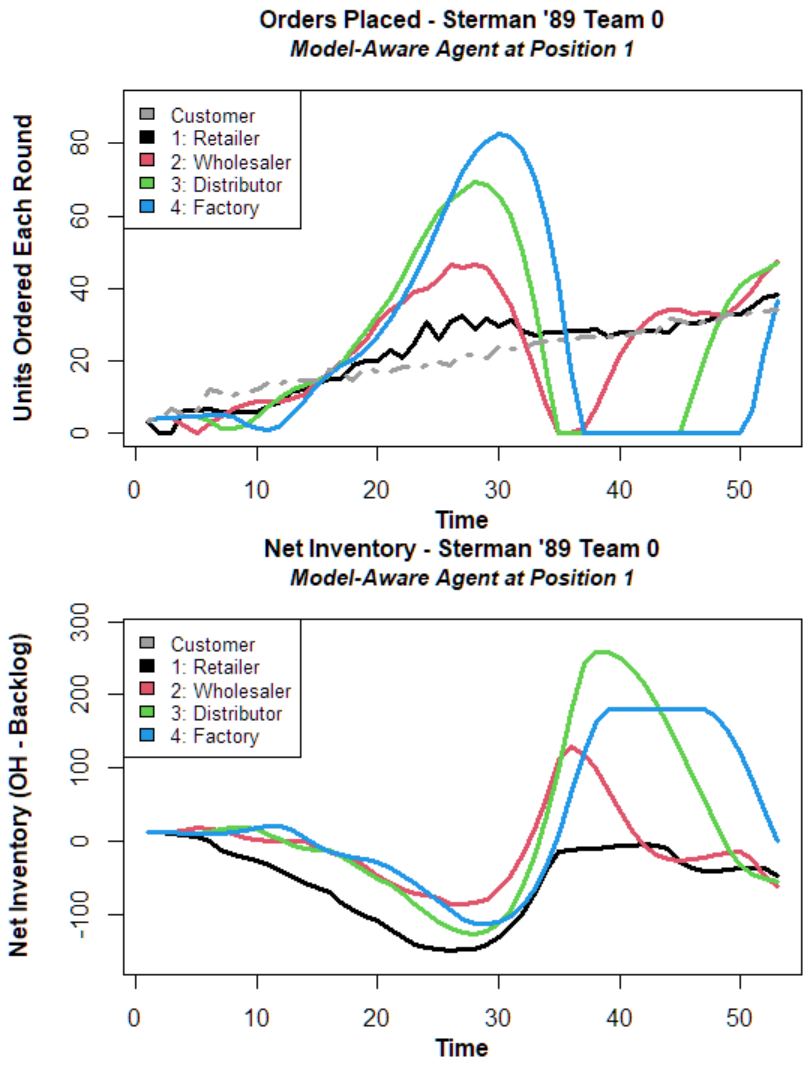

Figure S2: Orders and Inventory with MP Agent at Position 1 and Optimization Horizon of 35

Figure S1 would seem to imply that, generally, a smaller optimization horizon results in lower costs for the team. In other words, a greedier agent in position 1, one that only considers the immediate future, reduces overall team costs. However, consider Figure S3 which zooms in on the case with optimization horizon equal to 10. While the average team costs over the 52 week simulation are objectively lower than in the case of the longer optimization horizon above, the agents behavior is arguably much worse. By attempting to minimize the costs that are incurred by having a destabilized supply chain, the agent effective ignores the increasing orders from the end customer, and eventually gets into a position later in the simulation where matching customer orders and incurring temporary disruptions in the supply chain are too costly over the very near horizon.

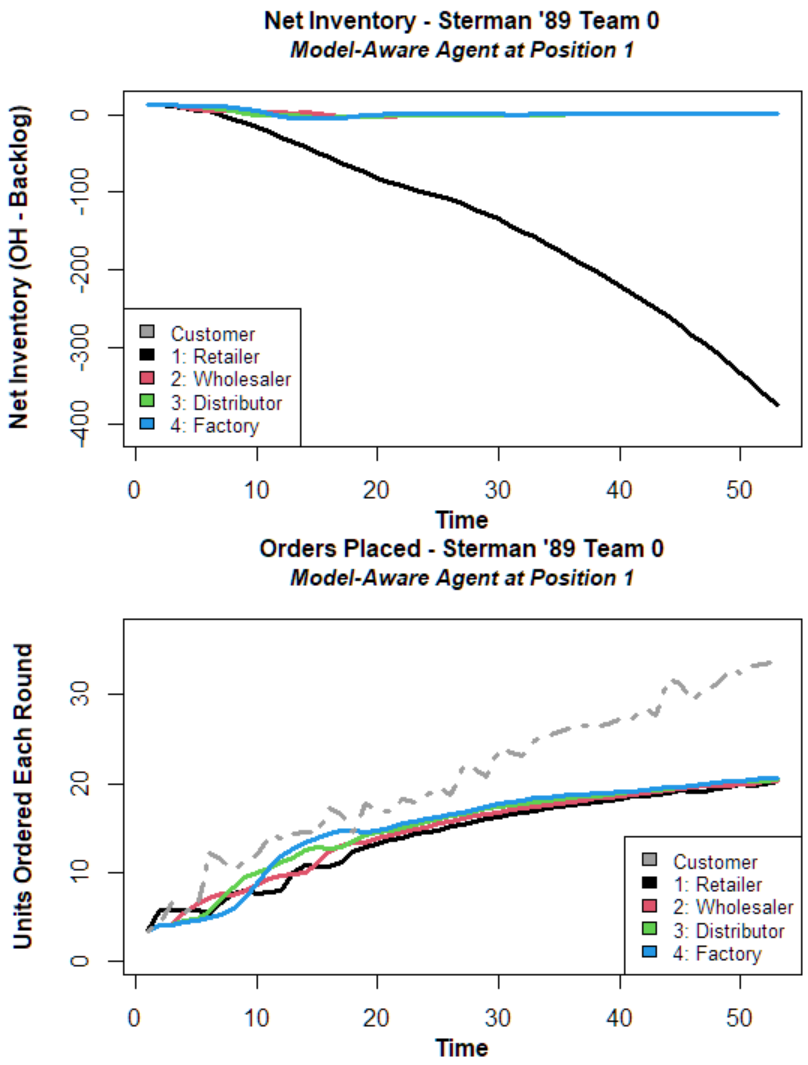

Figure S3: Orders and Inventory with MP Agent at Position 1 and Optimization Horizon of 10

Such pernicious outcomes as function of hyperparameter choices are interesting, and of concern for specific scenarios but ultimately secondary to the central points of the main article and thus left for this Appendix.

Matched vs Mismatched Environmental Assumption and Agent Response

As discussed elsewhere in the main article and in this appendix, the most straightforward structural assumption of the model-predictive learning agent is to match the assumption of the environmental ordering rules with the agent's ordering rules. If the agent assumes the other entities in its environment are rational base-stock actors, then the best course of action for that agent is to also respond in a base-stock manner. Similarly, if the agent assumes other entities are not necessarily rational and following some other ordering heuristic, then using a behavioral

response itself at minimum grants the agent more degrees of freedom to form its own ordering policy.

Figure S4 shows box-and-whisker plots for the cost reduction for an agent placed at various spots the supply chain versus the baseline of no agent present for the same 49 teams used elsewhere in these analyses (the original 11 Sterman ’89 teams plus the average Sterman ’89 team plus 37 additional teams fitted from real order data from three separate runs of the Beer Game in 2021 and 2022), subject to the step input signal from the original Sterman 89 paper. Note that the truth’ of the environment in which the agent is acting is behaviorally-driven (all other agents are using the Sterman ’89 ordering rule). Note that this implies that the primary benefit comes from matching the assumption of the environment to the truth, independent of the agent architecture. However, this is less clear in the middle positions of the supply chain, and even reversed in some middle positions. The relationship when matching the ordering rule of the agent to the assumed environment is however consistent among all four positions.

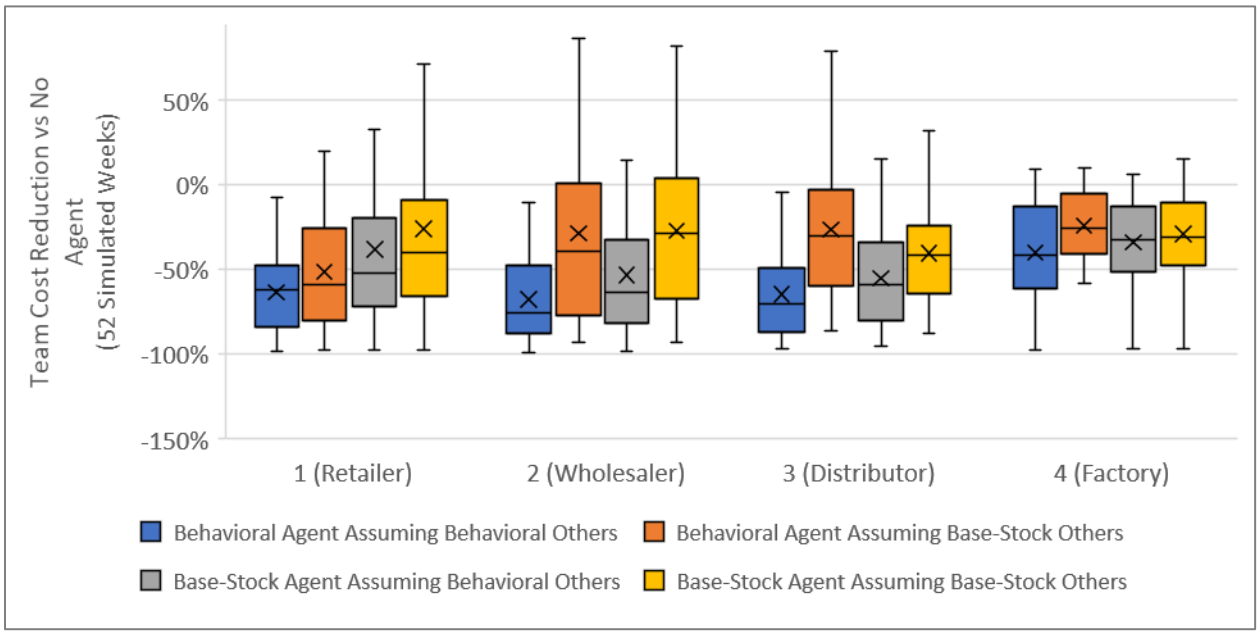

Figure S4: Orders and Inventory with MP Agent at Position 1 and Optimization Horizon of 10

Full Interaction Table for Model-Predictive Learning Agents

The main article presented, in Table 4, a regression emphasizing how features of the model-predictive learning agent affect, on average, the cost reducing performance of the agents. Table S2 presents a more complete series of regression models, including interaction terms, that build on the results in Table 4 of the main article. In that section of the main article, no interaction terms in the models were presented because, as seen below, they are not of significance. While insignificant terms were presented in the main article it was that no terms related to information availability were significant it the first place that was a main observation of the article.

Table S2. Feature Performance in a Learning Agent with Interactions at Position 2

	Dependent variable:					
	Performance Improvement over Baseline					
	(5)	(5.1)	(6)	(6.1)	(7)	(7.1)
Behavioral Agent	-0.084***	-0.067**	-0.085***	-0.066**	-0.084***	-0.053
	(0.018)	(0.026)	(0.018)	(0.026)	(0.018)	(0.053)
Non-Myopic Agent	-0.028	-0.010		-0.010	-0.028	-0.026
	(0.018)	(0.026)		(0.026)	(0.018)	(0.053)
Behavioral x Non-Myopic		-0.035		-0.036		-0.017
		(0.037)		(0.037)		(0.074)
Behavioral x Low Info						-0.003
						(0.075)
Behavioral x Standard Info						-0.032
						(0.074)
Behavioral x High Info						-0.015
						(0.074)
Non-Myopic x Low Info						0.014
						(0.075)
Non-Myopic x Standard Info						0.017
						(0.075)
Non-Myopic x High Info						0.035
						(0.075)
Behavioral x Non-Myopic x Low Info						-0.025
						(0.105)
Behavioral x Non-Myopic x Standard Info						-0.048
						(0.104)
Behavioral x Non-Myopic x High Info						-0.004
						(0.104)
Low Information			-0.016	-0.016	-0.016	-0.014
			(0.026)	(0.026)	(0.026)	(0.053)
Standard Information			-0.018	-0.019	-0.019	0.002
			(0.026)	(0.026)	(0.026)	(0.052)
High Information			-0.017	-0.017	-0.017	-0.026
			(0.026)	(0.026)	(0.026)	(0.052)
Constant	-0.508***	-0.517***	-0.509***	-0.504***	-0.495***	-0.507***
	(0.016)	(0.018)	(0.021)	(0.024)	(0.023)	(0.037)
Observations	744	744	744	744	744	744
R^2	0.031	0.032	0.029	0.033	0.032	0.036
Adjusted R^2	0.028	0.028	0.024	0.025	0.025	0.016
Residual Std. Error	0.250 (df = 741)	0.250 (df = 740)	0.251 (df = 739)	0.250 (df = 737)	0.250 (df = 738)	0.252 (df = 728)
F Statistic	11.870*** (df = 2; 741)	8.223*** (df = 3; 740)	5.481*** (df = 4; 739)	4.211*** (df = 6; 737)	4.865*** (df = 5; 738)	1.816** (df = 15; 728)
Note:						*p<0.1; **p<0.05; ***p<0.01

Alternative Analysis for Step-Inputs

The order input into the simulated supply chain discussed in the main article was gaussian normally drawn orders with mean 10 and with standard deviation of 4 units, similar to that in prior literature (Chen & Samroengraja, 2009) and later used again in similar simulated supply chain settings (Oroojlooyjadid et al., 2021). This stochastic input was chosen in part to isolate the effects of different agent policy features from the effects of specific order strings (by avoiding, for example, a single-shot policy or the DQN policies from inadvertently 'memorizing' the order input during the training or pre-optimization processes).

However, the original Beer Game runs on which the real-world ordering data was derived did not follow this gaussian order input but rather followed the original step-input orders (from 4 to 8 units after four rounds) as used in (Sterman, 1989). Table S3 shows the same analysis done in the original article, but for this classical step input in orders and can be compared with Table 3 in the original article. The values of each parameter in this model are extremely similar to those developed for the gaussian input, and the same parameters are significant in both settings. Of note, in this classical step input setting, the influence of the feature of 'Behavioral Agent' in models 2 and 3 is *higher* than in the original gaussian setting (though it is still significant there).

Table S3. Feature Influence: Behavioral and Learning Agent for Step Input at Position 2

Base Features influence on Agent Performance with Interactions

	Dependent variable:			
	Performance Improvement over Baseline			
	(1)	(2)	(3)	(4)
Behavioral Agent	-0.055***		-0.055***	0.038
	(0.020)		(0.020)	(0.064)
Learning Agent		-0.071**	-0.069**	-0.019
		(0.034)	(0.034)	(0.047)
Behavioral x Learning				-0.102
				(0.067)
Constant	-0.555***	-0.519***	-0.493***	-0.538***
	(0.014)	(0.032)	(0.033)	(0.045)
Observations	800	800	800	800
R^2	0.009	0.005	0.015	0.018
Adjusted R^2	0.008	0.004	0.012	0.014
Residual Std. Error	0.283 (df = 798)	0.284 (df = 798)	0.283 (df = 797)	0.282 (df = 796)
F Statistic	7.622*** (df = 1; 798)	4.383** (df = 1; 798)	5.940*** (df = 2; 797)	4.737*** (df = 3; 796)
Note:				*p<0.1; **p<0.05; ***p<0.01

Similarly, Table S4 repeats the same analysis performed and summarized in Table 4 in main article, but with the classical step input order string instead of the gaussian orders. As with the above comparison, using a step input to the simulated system produces qualitatively very similar results as using the gaussian normal input. Degrees of influence of specific features shift slightly (notably the influence of a behavioral agent versus a simple base-stock agent is reduced marginally in the step-input case), but the key and surprising observation that the degree of information availability in a model-predictive learning agent is insignificant holds.

What does change in the step-input setting is that the influence of having a non-myopic agent becomes significant. While in the gaussian setting, the sign of that feature parameter was negative, it was not significant, implying that while directionally having a more global view of the

supply chain was influential (and cost reducing) on outcomes it was not significantly so. For the simpler step-input case, the sign is still negative but now of statistical significance. While this influence is less than the other features identified in the main article (namely having a behavioral agent and having a stable policy to begin with), it is nevertheless present. This does not diminish the original discussion in the main article as the sign is still directionally consistent, and the overall results still support the observation that having a locally optimizing agent can still be beneficial, or at least that having a globally optimizing agent is not a necessary feature of a cost reducing agent.

Table S4. Feature and Information State Influence within a Learning Agent for Step Input at Position 2

	Dependent variable:		
	Performance Improvement over Baseline		
	(1)	(2)	(3)
Behavioral Agent	-0.063***	-0.065***	-0.063***
	(0.021)	(0.021)	(0.021)
Non-Myopic Agent	-0.058***		-0.058***
	(0.021)		(0.021)
Low Information		0.005	0.005
		(0.030)	(0.030)
Standard Information		-0.010	-0.010
		(0.030)	(0.029)
High Information		-0.013	-0.013
		(0.030)	(0.030)
Constant	-0.528***	-0.552***	-0.524***
	(0.018)	(0.024)	(0.026)
Observations	722	722	722
R^2	0.024	0.014	0.024
Adjusted R^2	0.021	0.008	0.017
Residual Std. Error	0.280 (df = 719)	0.282 (df = 717)	0.281 (df = 716)
F Statistic	8.663*** (df = 2; 719)	2.487** (df = 4; 717)	3.549*** (df = 5; 716)
Note:			*p<0.1; **p<0.05; ***p<0.01

DQN Agent Architecture and Hyperparameters

The DQN agent introduced in the main article serves two purposes: 1) to provide a minor methodological contribution by providing another viable DQN approach towards managing ordering decisions in multi-echelon supply chain, specifically in the beer game, and specifically utilizing the dueling reward function architecture (Z. Wang et al., 2016), and intends to extend directly from other recent work notably on architectures that use transfer learning (Oroojlooyjadid et al., 2021). 2) Provide a 'high complexity' point for comparison of other, often significantly less complex, agent policy architectures.

As the DQN itself is secondary to the central argument of this article, and recent prior literature by Oroojlooyjadid et al 2021 has provided a recent very detailed assessment of the DQN architecture in this environment in general, only a curiously overview of the agent is provided in the main article. In the supporting material that accompanies this Appendix is all of the code used to train both the model-free and model-aware versions of the DQN. To restate from the main article both DQN agents have the following general architecture: 1) An 'order-plus' action space (Oroojlooyjadid et al., 2021) which both allows for unbounded ordering in absolute terms and follows from observations in the model-based approach above, 2) a dual DQN network (Z. Wang et al., 2016) that separately maintains a value function estimation for both the current overarching combined state of the system and separately for each action, 3) an observation space defined over a window of prior state observations corresponding to the signal delay in the system, 4) a combination of epsilon-greedy and Boltzmann exploration policies (Wiering, 1999), and finally 5) three sequential dense layers with ReLu activations of 256, 128, and 64 free parameters respectively for a total of 448 free parameters.

With respect to the hyperparameters of the system, the one of most interest is the amount of training steps employed and how this affects the spread in performance between the model-free version and the model-aware version of the agent. Figure S5 shows the average cost improvement as a function of training steps when the agent is placed in position 1 (the retailer) for both of these versions the DQN. Note that this is inclusive of *all* 49 different teams, including destabilizing outcomes, and subject to the classical step input from Sterman '89. While there is *some* improvement from the Model-Aware agent versus the Model-Free agent (which is also seen in the main article), this improvement is relatively minimal once the sufficient training steps occur. Indeed the primary value of the model aware agent is under smaller training iterations. This supports similar observations made in the main article above the model-

predictive learning agent as well, namely that additional information about the environment is less useful than expected. Here we hypothesize, given enough training data, the model-free agent can still determine a sufficient estimate of the state of the entire system from its own state variables without needing to be told an explicit representation of that system.

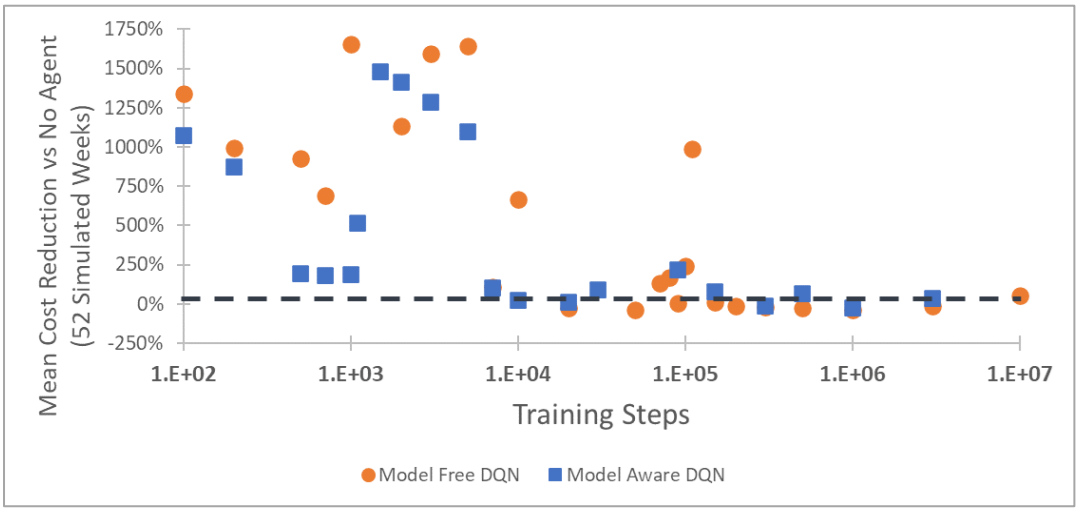

Figure S5: Overall DQN Performance at Position 1 (Retailer) versus Training Steps

References to the Appendix

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). TensorFlow: A System for Large-Scale Machine Learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 265–283. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *OpenAI Gym*. http://arxiv.org/abs/1606.01540

Chen, F., & Samroengraja, R. (2009). The Stationary Beer Game. *Production and Operations Management*, *9*(1), 19–30. https://doi.org/10.1111/j.1937-5956.2000.tb00320.x

Chollet, F. (2015). *Keras*. GitHub. https://github.com/fchollet/keras

Clark, A. J., & Scarf, H. (1960). Optimal Policies for a Multi-Echelon Inventory Problem. *Management Science*, *6*(4), 475–490. https://doi.org/10.1287/mnsc.6.4.475

Croson, R., Donohue, K., Katok, E., & Sterman, J. (2014). Order stability in supply chains: Coordination risk and the role of coordination stock. *Production and Operations Management*, *23*(2), 176–196. https://doi.org/10.1111/j.1937-5956.2012.01422.x

McNally, T. (2019). *Keras-rl2*. GitHub. https://github.com/taylormcnally/keras-rl2

Moritz, B. B., Narayanan, A., & Parker, C. (2021). Unraveling Behavioral Ordering: Relative Costs and the Bullwhip Effect. *Manufacturing & Service Operations Management*, *November*. https://doi.org/10.1287/msom.2021.1030

Oliva, R., Abdulla, H., & Gonçalves, P. (2022). Do Managers Overreact When in Backlog? Evidence of Scope Neglect from a Supply Chain Experiment. *Manufacturing & Service Operations Management*. https://doi.org/10.1287/msom.2021.1072

Oroojlooyjadid, A., Nazari, M., Snyder, L. V., & Takáč, M. (2021). A Deep Q-Network for the Beer Game: Deep Reinforcement Learning for Inventory Optimization. *Manufacturing & Service Operations Management*, msom.2020.0939. https://doi.org/10.1287/msom.2020.0939

RStudio Team. (2020). *RStudio: Integrated Development Environment for R*. RStudio, PBC. http://www.rstudio.com/

Sterman, J. (1989). Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision Making Experiment. *Management Science*, *35*(3), 321–339. https://doi.org/10.1287/mnsc.35.3.321

Sterman, J., & Dogan, G. (2015). “I’m not hoarding, I’m just stocking up before the hoarders get here.” Behavioral causes of phantom ordering in supply chains. *Journal of Operations Management*, *39–40*(1), 6–22. https://doi.org/10.1016/j.jom.2015.07.002

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Frcitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning. *33rd International Conference on Machine Learning, ICML 2016*, *4*(9), 2939–2947.