Simpler is (Sometimes) Better

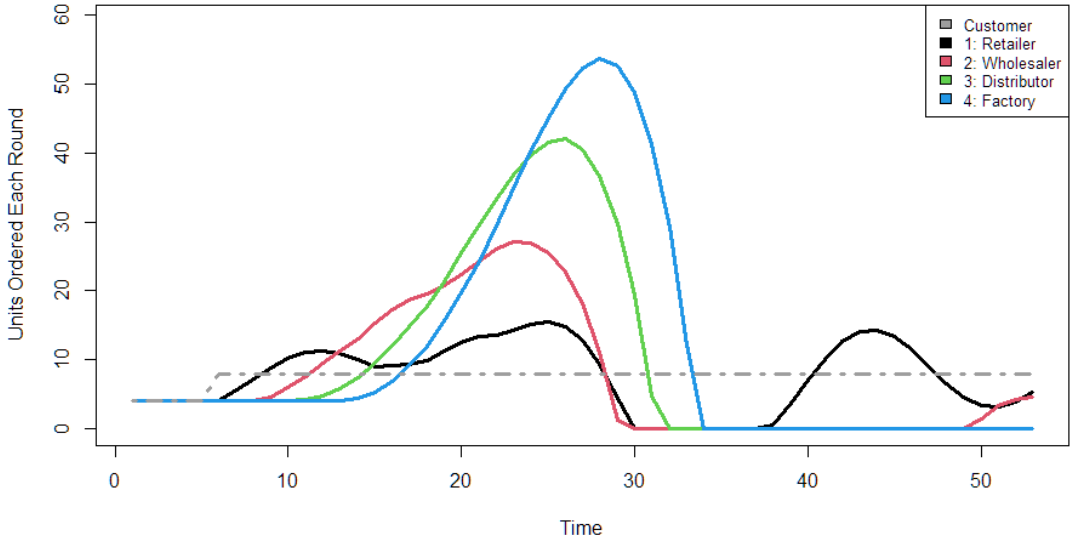
A Comparison of Cost Reducing Agent Architectures in a Simulated Behaviorally Driven Multi-Echelon Supply Chain

MIT MANAGEMENT SLOAN SCHOOL

2022 INFORMS Annual Meeting
SD34 Supply Chains
October 16, 2022

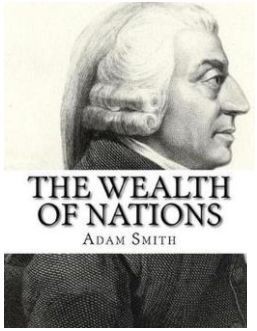
James Paine
jpaine@mit.edu
http://jpaine.mit.edu

Bullwhip

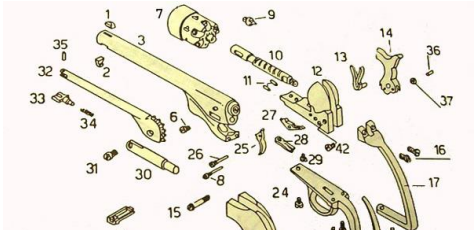


Bullwhip is *structurally induced* but *behaviorally amplified* in this environment

Operations Management



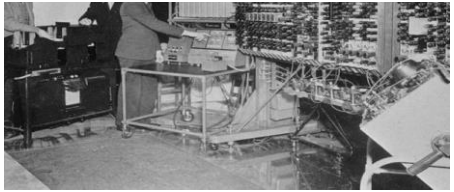
18th Century – Division o



th Century – Taylor's
ientific Management'

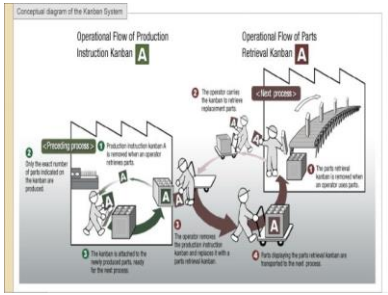
Operations Management focus on understanding and improving *real world* problems

20th Century – Ford's Assembly Line



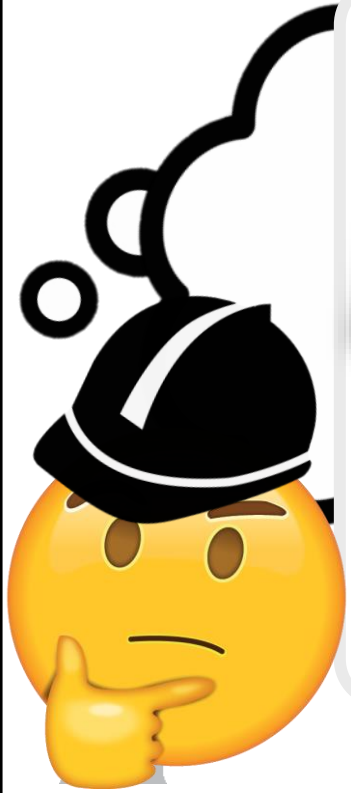
20th Century – Computing Power

20th Century – WWII



20th Century – TPS / Lean

What to do?

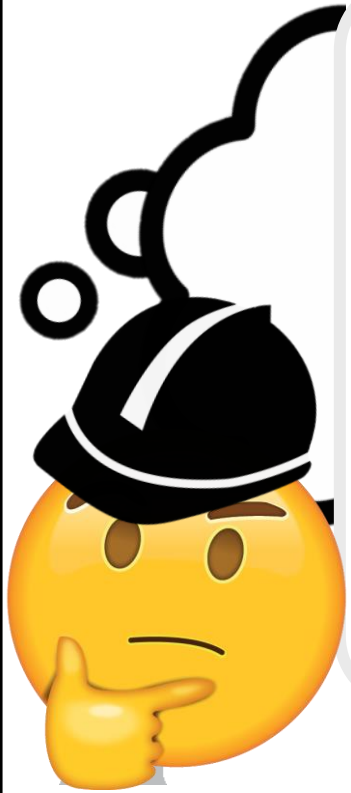


Rational Policies?

Behavioral Setting?

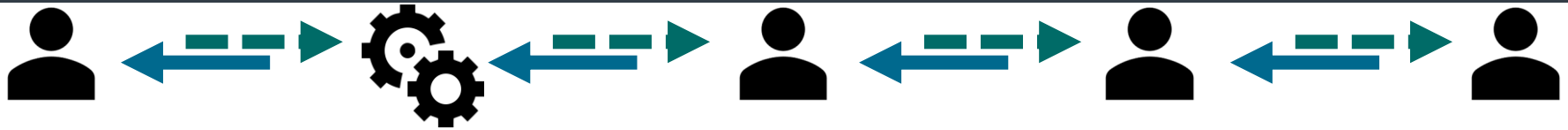
Information Availability?

What to do?

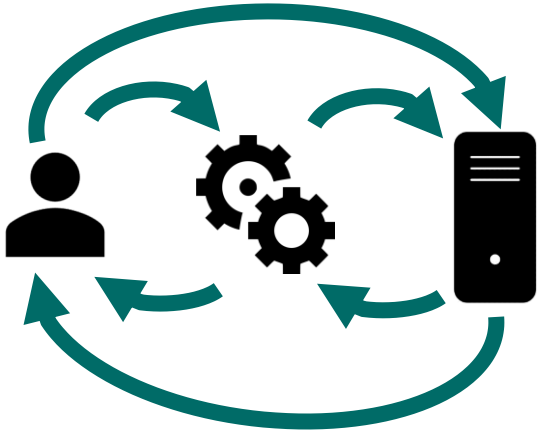


Blackbox !?

This Work



- **Assess contribution of different approaches**
  - **Design agent to follow policies in the context of a well-established simulated supply chain**
  - **Compare performance outcomes from different policy features**



DESIGN OF EXPERIMENT

Design of Experiment

(1) Rule Complexity	(2) Degree of Adaptability	(3) Incentive Structure	(4) Information Availability
(.1) Base Stock Ordering	(.1) Single-Shot	(.1) Self (Myopic)	(.1) Minimal
(.2) Behavioral Ordering	(.2) Model-Predictive Learning	(.2) Team (Non-Myopic)	(.2) Low
(.3) Deep-Q Network			(.3) Standard
			(.4) High

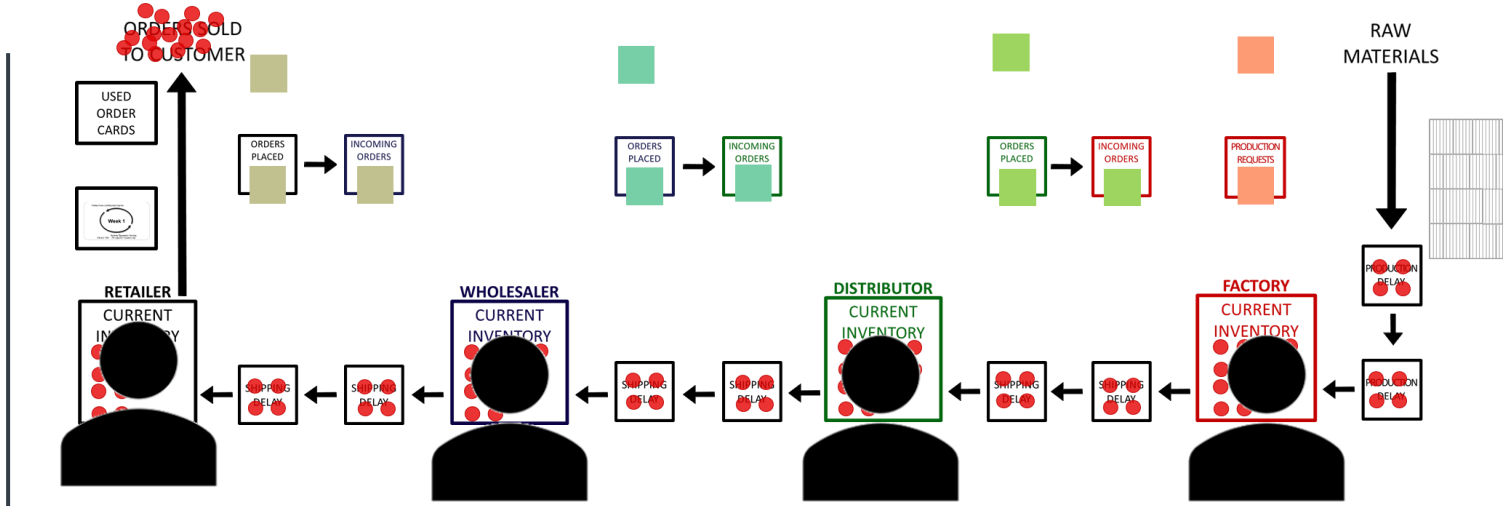
Full Conditions

Run Code	Agent Feature Description
1.1 - 2.1	Base Stock Ordering, Single-Shot
1.2 - 2.1	Behavioral Ordering, Single-Shot
1.3 - 2.1	Deep-Q Network, Single-Shot
1.3 - 2.2	Deep-Q Network, Model-Predictive Learning
1.1 - 2.2 - 3.1 - 4.1	Base Stock Ordering, Model-Predictive Learning, Self (Myopic), Minimal
1.1 - 2.2 - 3.1 - 4.2	Base Stock Ordering, Model-Predictive Learning, Self (Myopic), Low
1.1 - 2.2 - 3.1 - 4.3	Base Stock Ordering, Model-Predictive Learning, Self (Myopic), Standard
1.1 - 2.2 - 3.1 - 4.4	Base Stock Ordering, Model-Predictive Learning, Self (Myopic), High
1.1 - 2.2 - 3.2 - 4.1	Base Stock Ordering, Model-Predictive Learning, Team (Non-Myopic), Minimal
1.1 - 2.2 - 3.2 - 4.2	Base Stock Ordering, Model-Predictive Learning, Team (Non-Myopic), Low

Run Code	Agent Feature Description
1.1 - 2.2 - 3.2 - 4.3	Base Stock Ordering, Model-Predictive Learning, Team (Non-Myopic), Standard
1.1 - 2.2 - 3.2 - 4.4	Base Stock Ordering, Model-Predictive Learning, Team (Non-Myopic), High
1.2 - 2.2 - 3.1 - 4.1	Behavioral Ordering, Model-Predictive Learning, Self (Myopic), Minimal
1.2 - 2.2 - 3.1 - 4.2	Behavioral Ordering, Model-Predictive Learning, Self (Myopic), Low
1.2 - 2.2 - 3.1 - 4.3	Behavioral Ordering, Model-Predictive Learning, Self (Myopic), Standard
1.2 - 2.2 - 3.1 - 4.4	Behavioral Ordering, Model-Predictive Learning, Self (Myopic), High
1.2 - 2.2 - 3.2 - 4.1	Behavioral Ordering, Model-Predictive Learning, Team (Non-Myopic), Minimal
1.2 - 2.2 - 3.2 - 4.2	Behavioral Ordering, Model-Predictive Learning, Team (Non-Myopic), Low
1.2 - 2.2 - 3.2 - 4.3	Behavioral Ordering, Model-Predictive Learning, Team (Non-Myopic), Standard
1.2 - 2.2 - 3.2 - 4.4	Behavioral Ordering, Model-Predictive Learning, Team (Non-Myopic), High

- **Full factorial design on Behavioral x Learning x Information**
- **Plus two non-learning conditions**
- **Plus two DQN conditions**

Specific Simulation Framework: The Beer Game



Simulation Framework

Modular Simulation Model of the Beer Game

- Discrete time simulation
- Modularized to allow different ordering rules from prior literature and to allow embedding in popular AI and simulation tools (AI Gym, Keras, PyTorch, etc)
- E.g. Sterman 89 Ordering rule based on four parameters:

$$O_t = MAX\left(0, \hat{L_t} + \alpha_S(S' - S_t - \beta\, SL_t) + \varepsilon_t\right)$$

$$where\ \hat{L_t} = \theta L_t + (1-\theta)\hat{L}_{t-1}$$

O = order placed at time t

L^ = smoothed interpolation of the expected outflow of inventory

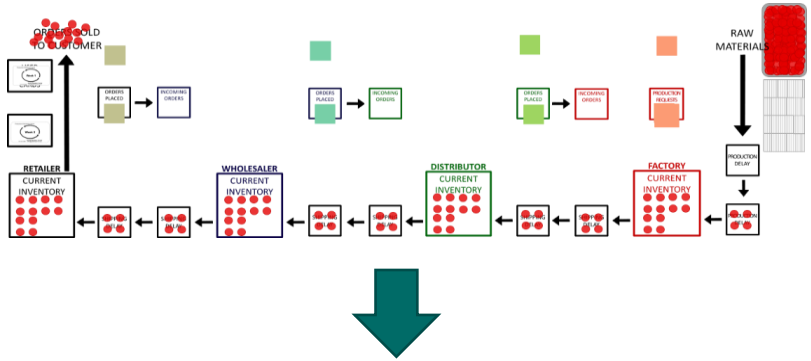
Θ = smoothing parameter

SL = total inbound supply line of inventory

S = current on-hand inventory (or stock)

S' = analogous to the desired or goal on-hand inventory of the player

β = weight of supply line

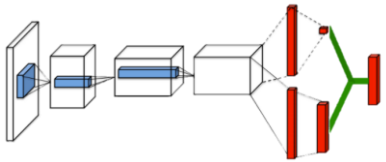



```
#debug
Entity_Index = 1

for (i in 1:4) {

 Entity_Index = i

 #Observe supply line
 SL = sum(Shipping_flows[Entity_Index,,t]) #Total supply line inbound

 #L[Entity_Index,t] = Incoming_Order[Entity_Index]
 L[Entity_Index,t] = max(min(Incoming_Order[Entity_Index],OH_Inventory[Entity_Index,t]),Order_flows[Entity_Index,1,t-1]
 L[Entity_Index,t] = Order_flows[Entity_Index,1,t-1]

 if (t==1) {
   L_hat[Entity_Index,t] = Order_flows[Entity_Index,1,t]
 } else if (t==2) {
   L_hat[Entity_Index,t-1] = Order_flows[Entity_Index,1,t-1]
   L[Entity_Index,t-1] = L_hat[Entity_Index,t-1]
 }

 if (t==20) {
   t = t
 }

 L_hat[Entity_Index,t] = theta[Entity_Index]*L[Entity_Index,t] + (1-theta[Entity_Index])*L_hat[Entity_Index,t-1]

 S = OH_Inventory[Entity_Index,t] #Stock as of current time
```

Preview of Results

- **Incorporating *behavioral* elements significantly improves performance**

- **Very complex methods, while interesting, gain performance at the cost of orders of magnitude more complexity**



- **Agents can be myopic in their optimization goal and have limited information about their environment and *still* be cost reducing**

METHODS: AGENT POLICY ARCHITECTURES

Single-Shot Policy – Base-Stock Agent

Agent must assume that others are fully rational with same information

- Assuming features of order input are known, costs are increasing along supply chain, and delays are fixed, can follow Clark, A. J., & Scarf, H. (1960)
- This game does not *exactly* meet that criteria
- Can follow example of Oroojlooyjadid et al '21 *(start from C&S then grid search, taking average best values from 50 draws)*

Single-Shot Policy – Behavioral Agent

$Exogenous\ O_{t,c}$

$$O_{t,1} = MAX\left[0, f(\theta_1) + \alpha_1\left(S'_1 - S_{t,1} - \beta_1\, SL_{t,1}\right)\right]$$

$$O_{t,2} = MAX\left[0, f(\theta_2) + \alpha_2\left(S'_2 - S_{t,2} - \beta_3\, SL_{t,2}\right)\right]$$

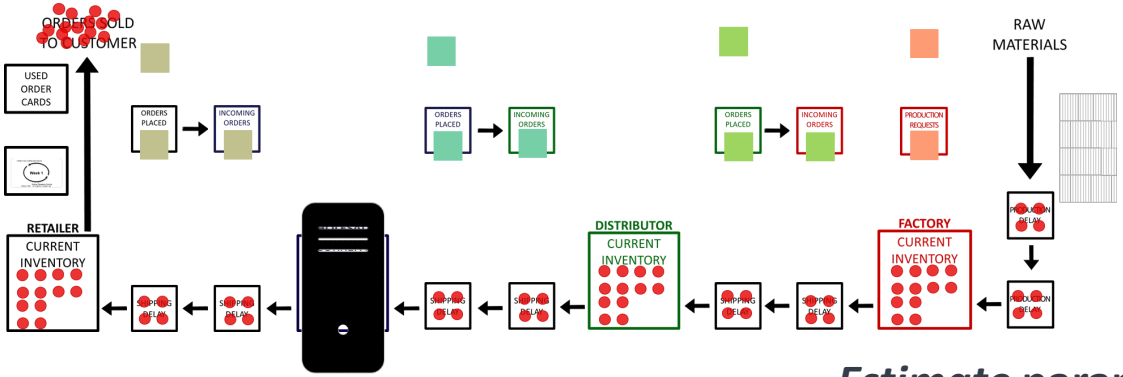
$$O_{t,3} = MAX\left[0, f(\theta_3) + \alpha_3\left(S'_3 - S_{t,3} - \beta\_3\, SL_{t,3}\right)\right]$$

$$O_{t,4} = MAX\left[0, f(\theta_4) + \alpha_4\left(S'_4 - S_{t,4} - \beta_4\, SL_{t,4}\right)\right]$$

For some entity index i

- Fix $\theta_{n\neq i}$, $\alpha_{n\neq i}, S'_{n\neq i}, \beta_{n\neq i}$
- Default average performance of Sterman '89 as starting point
- Vary θ_i , α_i, S'_i, β_i to minimize cost function

Model-Predictive Learning Policy



$Exogenous\ O_{t,c}$

$$\widehat{O_{t,1}} = MAX\left[0, f\left(\widehat{\theta_1}\right) + \widehat{\alpha_1}\left(\widehat{S_1'} - \widehat{S_{t,1}} - \widehat{\beta_1}\widehat{SL_{t,1}}\right)\right]$$

$$\widehat{O_{t,2}} = MAX\left[0, f\left(\widehat{\theta_2}\right) + \widehat{\alpha_2}\left(\widehat{S_2'} - \widehat{S_{t,2}} - \widehat{\beta_2}\widehat{SL_{t,2}}\right)\right]$$

$$\widehat{O_{t,3}} = MAX\left[0, f\left(\widehat{\theta_3}\right) + \widehat{\alpha_3}\left(\widehat{S_3'} - \widehat{S_{t,3}} - \widehat{\beta_3}\widehat{SL_{t,3}}\right)\right]$$

$$\widehat{O_{t,4}} = MAX\left[0, f\left(\widehat{\theta_4}\right) + \widehat{\alpha_4}\left(\widehat{S_4'} - \widehat{S_{t,4}} - \widehat{\beta_4}\widehat{SL_{t,4}}\right)\right]$$

Estimate **parameters that best fit observations to an** ***Assumed*** **structural model**

Model-Predictive Learning Policy Pseudo Algorithm

Assume system model structure

Calibrate to **observed history**

Optimize based on **assumed and estimated model**

t = 0

Assume Structural and Dynamic Model of System
Define Agent position in System model
Define observable space for Agent
Populate initial assumption of parameterization and initializations
Define backward calibration memory and forward optimization horizon

for t in 1:horizon

Calibrate System Model given history

ArgMin{System Parameter Estimate}

Error (Observed space of simulation of System Model, Actual Observed space)

Return estimated parameters of System Model

Optimize forward given System Model estimate

ArgMax{Agent Decision Rule}

Over t:(t+opt horizon): Reward from t:horizon given System Model estimate

Model-Predictive Learning Policy Information States

Minimal
(Narrow Self)

- On-hand inventory position of Agent only
- Most recent order placed by Agent only

Low
(Self)

- On-hand inventory position of Agent only
- Inbound shipments to Agent only
- Order history of the Agent only

Standard
(Self + Visible Others)

- On-hand inventory position all positions
- Inbound shipments to all positions
- Order history of the Agent only

High
(Near Omniscient)

- On-hand inventory position all positions (all visible state variables)
- Backorder positions of all positions (all censored state variables)
- Inbound shipments to all positions (all material flows)
- Order history of all positions (all information flows)

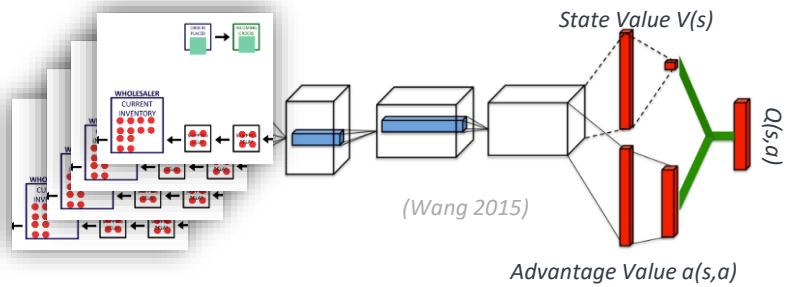
Deep Q-Network Policy

Framework

- OpenAI Gym custom environment based on same functional-form of Beer Game developed in Model-based optimization
- Environment consisting of:
  - All supply chain positions in parallel (versus transfer learning in sequence)
  - Randomly drawn models of real teams (from Sterman '89)
  - Random but bounded simulation horizons
  - Noisy realizations of order decisions

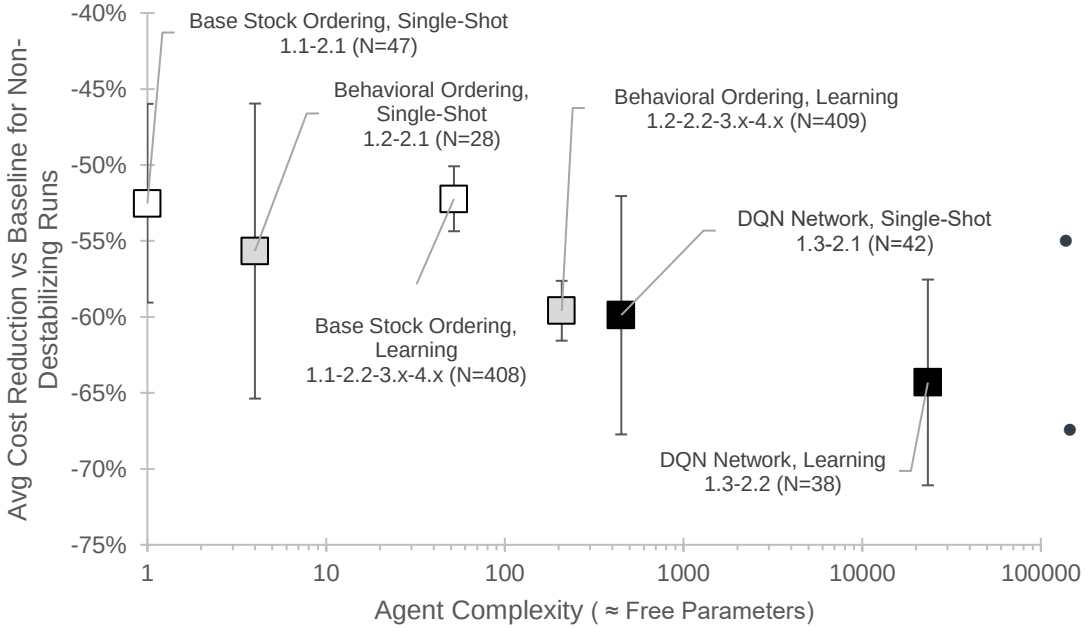
DQN Architecture

- Dual DQN network (split Action / State Q values)
- Three sequential dense layers with ReLu activation
- Order-plus action space guided by prior model-optimization
- Combination of epsilon-greedy and Boltzmann policy (Wiering 1999)
- Observation space limited to data available in Beer Game (x4 window for sequential memory)



RESULTS

High-Level Results



- Increasing complexity yield lower average costs (generally) but with decreasing returns
- 'Single-Shot' vs 'Learning' vs 'DQN' offset from each other
- DQN policies *much* more complex with similar performance

*Average cost reduction at position 2 (wholesaler) vs baseline of no agent with 90% CI bars
**Only cost-reducing results reported
***Information state-varying architectures aggregated

Regressing Predicted Agent Improvement on Policy Features

$$Model\ (1): \frac{Cost_{Agent_{Team_i}} - Cost_{Baseline\ Team_i}}{Cost_{Baseline\ Team_i}} = \beta_0 + \beta_1 f_{Behavioral} + \epsilon_{Team_i}$$

Does having a *behavioral* policy matter (single-shot or model-predictive)

$$Model\ (2): \frac{Cost_{Agent_{Team_i}} - Cost_{Baseline\ Team_i}}{Cost_{Baseline\ Team_i}} = \beta_0 + \beta_1 f_{learning} + \epsilon_{Team_i}$$

Does having a *learning* policy matter (single-shot vs model-predictive)

$$Model\ (3): \frac{Cost_{Agent_{Team_i}} - Cost_{Baseline\ Team_i}}{Cost_{Baseline\ Team_i}} = \beta_0 + \beta_1 f_{Behavioral} + \beta_2 f_{learning} + \epsilon_{Team_i}$$

Does having a *behavioral and learning* policy matter

$$Model\ (4): \frac{Cost_{Agent_{Team_i}} - Cost_{Baseline\ Team_i}}{Cost_{Baseline\ Team_i}} = \beta_0 + \beta_1 f_{Behavioral} + \beta_2 f_{learning} + \beta_3 (f_{Behavioral} * f_{learning}) + \epsilon_{Team_i}$$

Results

$$C_i^{M4} = \beta_0 + \beta_1 f_{Behavioral} + \beta_2 f_{learning} + \beta_3 (f_{Behavioral} * f_{learning}) + \epsilon_{Team_i}$$

	Dependent variable:			
	Performance Improvement over Baseline			
	(1)	(2)	(3)	(4)
Behavioral Agent	-0.074***		-0.071***	0.088
Learning Agent		-0.072**	-0.060*	0.003
Behavioral x Learning				-0.173***
Constant	-0.522***	-0.494***	-0.469***	-0.525***
Observations	817	817	817	817
Log Likelihood	-32.62	-38.62	-30.71	-27.06
McFadden R^2	0.210	0.064	0.256	0.324
Residual Std. Error	0.252 (df = 815)	0.254 (df = 815)	0.252 (df = 814)	0.251 (df = 813)
F Statistic	17.436*** (df = 1; 815)	5.291** (df = 1; 815)	10.651*** (df = 2; 814)	9.588*** (df = 3; 813)

Note: *p<0.1; **p<0.05; ***p<0.01

Agent at position 2 (wholesaler)
Only cost-reducing results reported

- Base agent (Single-Shot Base-Stock Agent) still surprisingly cost reducing
- Adding Behavioral assumptions is statistically significantly improving
- Adding learning also improves performance
  - Even m*ore* helpful when agent is making rationality assumption

Zooming In: Regressing Predicted Agent Improvement on Information Space in Learning Agents

Does having a *Non-Myopic* (team-level) cost reduction goal affect matter?

$$Model\ (5): \left.\frac{Cost_{Agent_{Team_i}} - Cost_{Baseline_{Team_i}}}{Cost_{Baseline_{Team_i}}}\right|_{Learning} = \beta_0 + \beta_1 f_{Behavioral} + \beta_2 f_{NonMyopic} + \epsilon_{Team_i}$$

Does having *different levels of information* about the environment matter?

$$Model\ (6): \left.\frac{Cost_{Agent_{Team_i}} - Cost_{Baseline_{Team_i}}}{Cost_{Baseline_{Team_i}}}\right|_{Learning} = \beta_0 + \beta_1 f_{Behavioral} + \beta_2 f_{LowInfo} + \beta_2 f_{StandardInfo} + \beta_2 f_{HighInfo} + \epsilon_{Team_i}$$

Does having different levels of information about the environment while being Non-Myopic matter?

$$Model\ (7): \left.\frac{Cost_{Agent_{Team_i}} - Cost_{Baseline_{Team_i}}}{Cost_{Baseline_{Team_i}}}\right|_{Learning} = \beta_0 + \beta_1 f_{Behavioral} + \beta_2 f_{NonMyopic} + \beta_3 f_{LowInfo} + \beta_4 f_{StandardInfo} + \beta_5 f_{HighInfo} + \epsilon_{Team_i}$$

Results

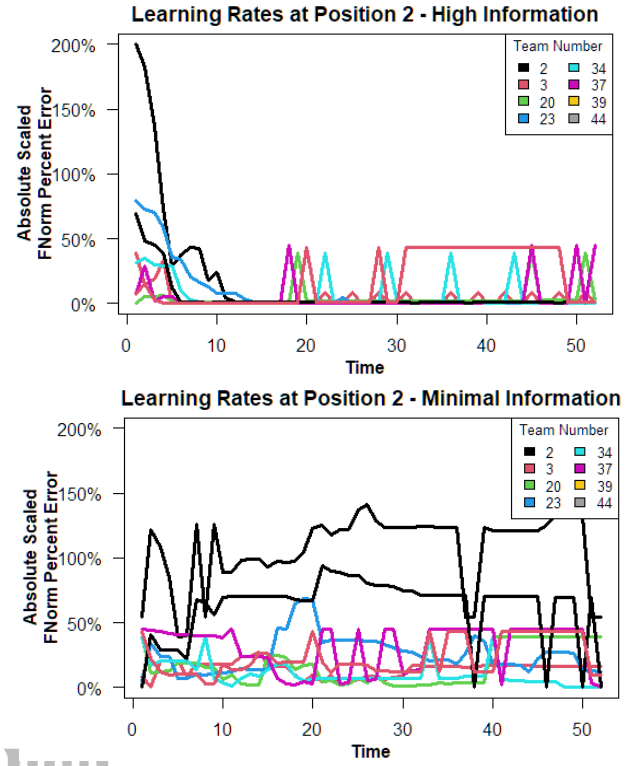
$$C_i^{M7} = \beta_0 + \beta_1 f_{Behavioral} + \beta_2 f_{NonMyopic} + \beta_3 f_{LowInfo} + \beta_4 f_{StandardInfo} + \beta_5 f_{HighInfo} + \epsilon_{Team_i}$$

	Dependent variable:		
	Performance Improvement over Baseline, given Learning Agent		
	(5)	(6)	(7)
Behavioral Agent	-0.084***	-0.085***	-0.084***
Non-Myopic Agent	-0.028		-0.028
Low Information		-0.016	-0.016
Standard Information		-0.018	-0.019
High Information		-0.017	-0.017
Constant	-0.508***	-0.509***	-0.495***
Observations	744	744	744
Log Likelihood	-23.04	-23.91	-22.72
McFadden R^2	0.344	0.313	0.345
Residual Std. Error	0.250 (df = 741)	0.251 (df = 739)	0.250 (df = 738)
F Statistic	11.870*** (df = 2; 741)	5.481*** (df = 4; 739)	4.865*** (df = 5; 738)
Note:			*p<0.1; **p<0.05; ***p<0.01

Agent at position 2 (wholesaler), Only cost-reducing results reported

- Again Base learning agent (Base-Stock Learning Myopic Agent) still surprisingly cost reducing
- Adding Behavioral assumptions is statistically significantly improving
- Information availability and team-focus *not significant*
  - Having *Minimal* information comparable to having *Omniscient* information
  - Myopic agents do similar to non-myopic
- Note: Results qualitatively similar in most other positions in supply chain. Interaction terms omitted due to lack of significance (see paper)

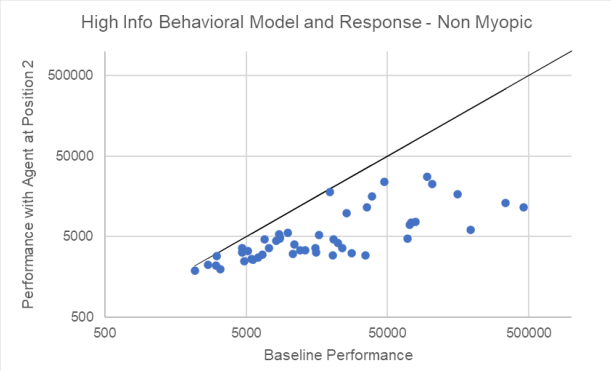
Trade-offs from Low Information

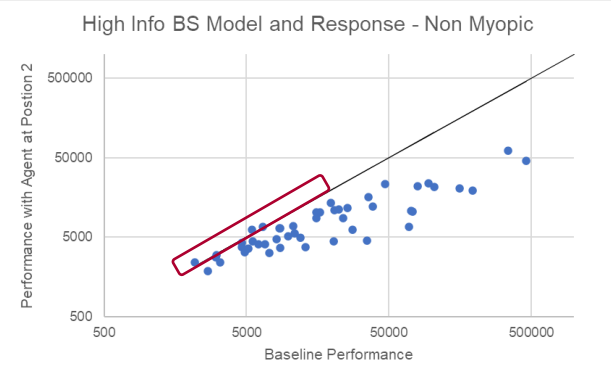


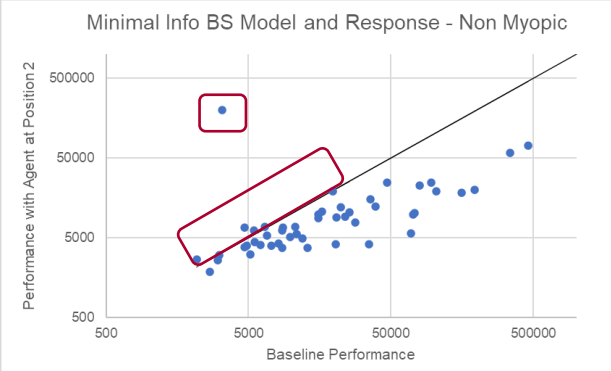
Information Availability Value is Limited

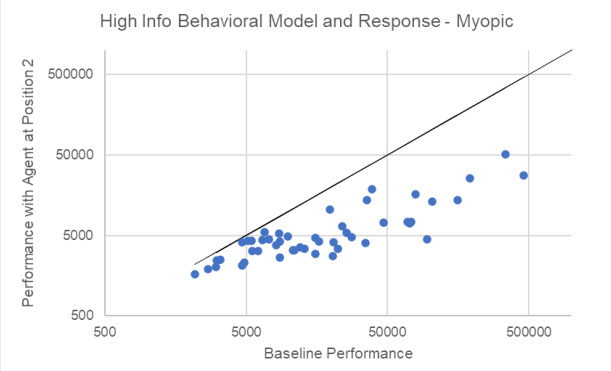
- Key benefit is having a model to fit in the first place
- Dynamically respond to poor initial choices (vs single-shot)
- Near perfect information = more likely to learn ‘true’ model of environment more quickly
- Minimal information = good enough when entire state can be estimated from limited info

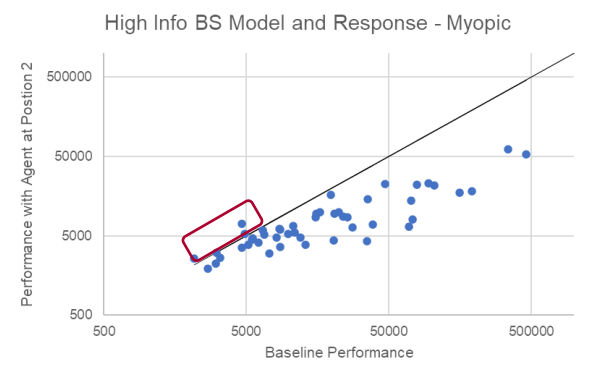
Trade-offs from Low Information

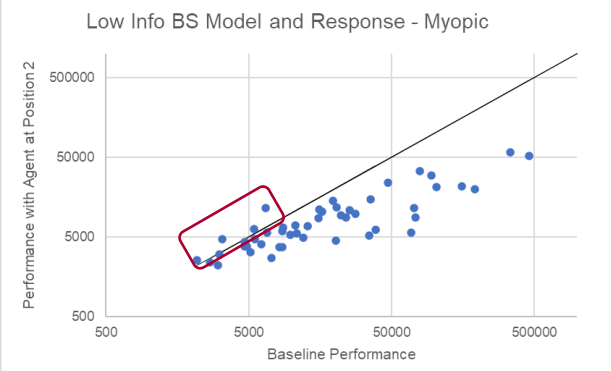












Trade-offs from Low Information

Fraction of Experiments with Destabilizing Outcomes with Model-Predictive Learning Agent in Position 2

Agent Type	Information State: *Minimal*	*Low*	*Standard*	*High*
Base-Stock (Myopic Agent)	6.1%	8.2%	6.1%	6.1%
Behavioral (Myopic Agent)	8.2%	10.2%	0.0%	0.0%

Agent Type	Information State: *Minimal*	*Low*	*Standard*	*High*
Base-Stock (Non-Myopic Agent)	10.2%	10.2%	10.2%	6.1%
Behavioral (Non-Myopic Agent)	0.0%	0.0%	0.0%	0.0%

Note: $N = 49$ *for all conditions*

Destabilizing is defined here as $\frac{Cost_{Agent_{Team_i}} - Cost_{Baseline_{Team_i}}}{Cost_{Baseline_{Team_i}}} > 0$

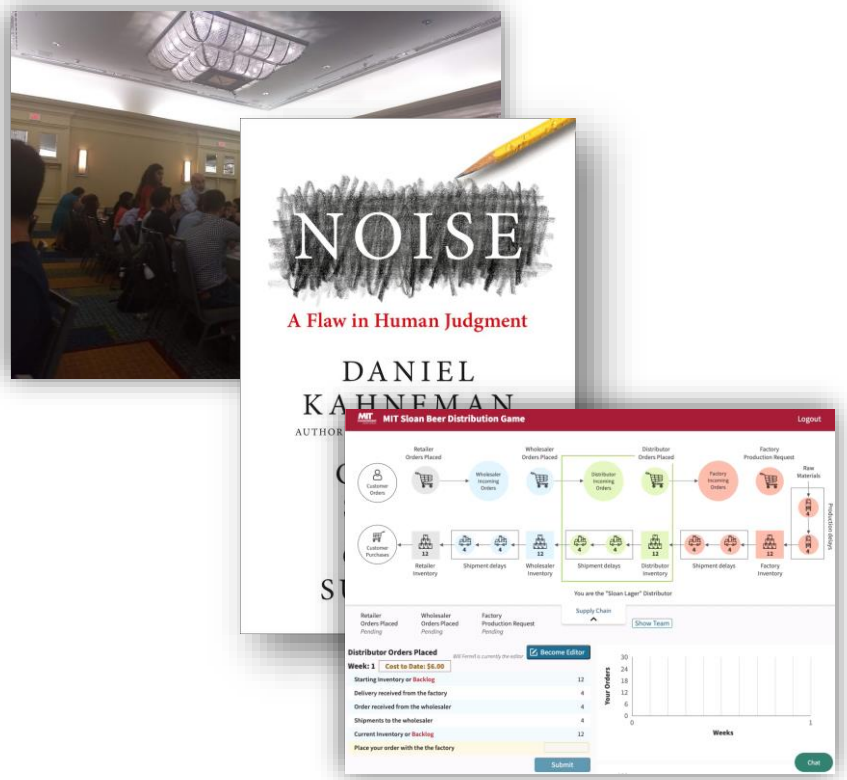
- More likely in base-stock agents, myopic agents, and with low information
- *Did not occur* for Behavioral Non-Myopic Agents

DISCUSSION, LIMITATIONS, AND TAKE-AWAYS

Limitations and Notes

Notable Limitations

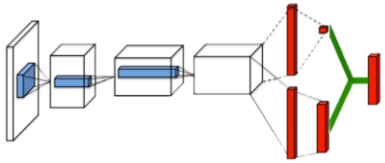
- This is an empirically grounded simulation, but not an empirically verified one (yet)
- Simulation of ordering rules from human players, while ‘behavioral-based’ are still *stable*
  - Next step is to add noise to ordering realizations
  - Prior experiments imply results should hold, but need to actually test this
- Ordering data from real games implies subtle behavioral differences between ‘in-person’ and online/hybrid runs.
  - Possibly captured in fitted parameters
  - Future research includes teasing these apart



Discussion and Summary

Minor (Methodological) Contributions in this Work

- Developed methods to apply Model-Predictive Learning methods to supply chain ordering policies
- Application of dueling DQN architecture to expand on recent work on DQN in multi-echelon supply chain settings



Main Contributions and Observations of this Work

- Direct comparison of features of policies largely identified separately in prior literature, to reduce cost in behaviorally-driven operational systems
- Incorporating *behavioral* elements significantly improves performance
- Very complex methods, while interesting, gain performance at the cost of orders of magnitude more complexity
- Agents can be myopic in their optimization goal and have limited information about their environment and *still* be cost reducing
    - So long as they are *learning*, and assuming behavioral responses low risk of destabilizing

Final Thoughts

Behavioral OM is about Operating Systems that consist of, and exist for, people

- This work shows
  - Cost reducing polices that exist in behaviorally-driven settings
  - How recognizing that people are making decisions is beneficial to these policies
- For managers, this work shows policies that can be locally focused and still achieve globally-preferred outcomes

Thank You!

Please send questions and comment to:

James Paine

jpaine@mit.edu

https://jpaine.info

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

https://ssrn.com/abstract=4244188