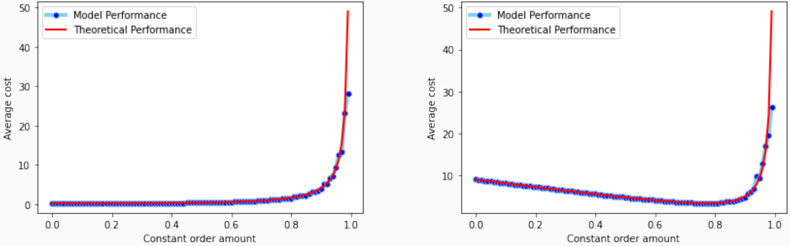
Inventory Control with Lost Sales and Lead Times

Zhi Liu & Wenchang Zhu

ORIE 6590 Spring 2021

Implementation and Validation

Implemented the environment using `gym` and validated using constant-order policy. The evaluation is done with 20 episodes of length 50000 each.



(a) $p = 0.25, L = 30$ **(b)** $p = 9, L = 10$

Figure 1: Performance of constant-order policy in our environment versus theoretical results.

After validation we used PPO from Stable Baselines 3 [Hill et al., 2018] package to train the policy.

PPO Parameterization: Network

- → Feature extraction: directly used the state space as input without any extraction.
- → Policy network: set the weights of output layer to 0 and the bias to a moderately sized amount, e.g. 0.4 when $p = 9$ and 0.5 when $p = 39$, then added Gaussian noise and pass through squashing function. This mimics a near-constant-order policy.
- → Value network: used the default parameterization.

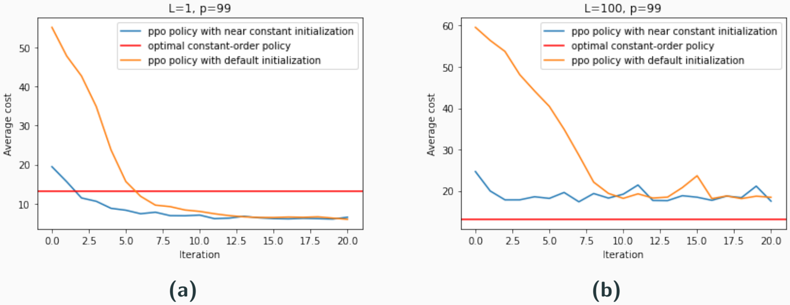


Figure 2: Comparison of default and custom initialization.

PPO Parameterization: Advantage Estimation

For small L, the default value `gae_lambda`$= 0.95$ gives good performance; when state space becomes larger, larger values yield better performance, so we chose `gae_lambda`$= 0.99$ when $L \in \{30, 50, 70, 100\}$.

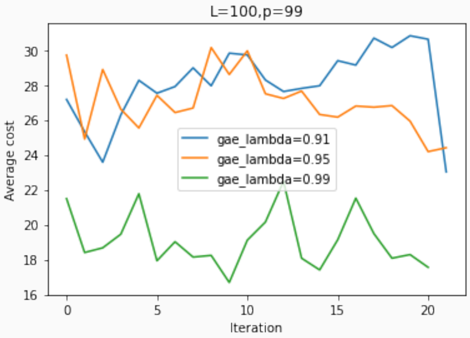


Figure 3: PPO performance with different `gae_lambda`

PPO Parameterization: Other Parameters and Training

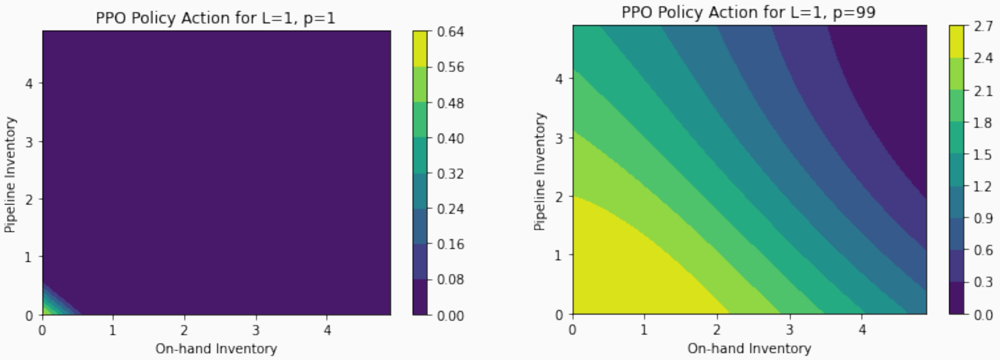
- $\rightarrow$ Maximum action allowed: $\mathcal{A} = [0, 20]$.
- $\rightarrow$ `n_envs`: 4.
- $\rightarrow$ `learning_rate`: 0.0003.
- $\rightarrow$ `n_steps`: rollout length 2048.
- $\rightarrow$ `batch_size`: 64.
- $\rightarrow$ `n_epochs`: 10.

PPO Parameterization: Other Parameters and Training

- → Maximum action allowed: $\mathcal{A} = [0, 20]$.
- → `n_envs`: 4.
- → `learning_rate`: 0.0003.
- → `n_steps`: rollout length 2048.
- → `batch_size`: 64.
- → `n_epochs`: 10.

- → Trained using PPO for 20 iterations to balance performance and runtime.
- → Evaluated policy after each iteration, and chose the policy with the best performance in terms of average cost.
- → Repeated for each combination of p and L to obtain the results.

Actions from Resulting Policies



(a) **(b)**

Figure 4: Action taken by trained policies under different model parameters.

Results: Compared with Default PPO Parameterization

C(const_order)/C(PPO)	L=1	L=4	L=10	L=20	L=30	L=50	L=70	L=100
p=0.25	0.92651	0.93007	0.92678	0.93586	**0.93465**	**0.91137**	**0.90932**	**0.91685**
p=1	1.00970	0.97611	0.95648	0.89441	**0.82767**	0.82281	**0.80145**	**0.78845**
p=4	1.14217	1.03615	1.00781	0.93620	0.79672	0.74644	0.71561	0.70126
p=9	1.26952	**1.13438**	1.02028	0.94697	0.82723	0.73339	0.71735	0.65570
p=39	1.83206	1.46433	1.22145	1.02358	0.80260	0.70389	0.57001	0.34879
p=99	**2.49023**	1.95434	1.43221	1.10200	0.79454	0.66899	0.39838	0.42888

(a) Results from default method

C(const_order)/C(PPO)	L=1	L=4	L=10	L=20	L=30	L=50	L=70	L=100
p=0.25	1.00368	1.00124	0.99923	0.98822	0.90195	0.88395	0.83419	0.85435
p=1	1.00970	1.00112	1.00050	0.99501	0.82632	0.83170	0.79193	0.77658
p=4	1.09048	1.04100	1.00175	0.97143	0.81546	0.81599	0.77624	0.79345
p=9	1.23925	1.13170	1.06793	1.00547	0.82239	0.79777	0.77362	0.77989
p=39	1.83981	1.46979	1.22424	1.17589	0.84864	0.78660	0.77462	0.76134
p=99	2.48444	2.30920	1.45911	1.13117	0.91039	0.81862	0.75929	0.78665

(b) Results from modified method

Figure 5: Comparison of results with default PPO parameterization.

Results

C(const_order)/C(PPO)	L=1	L=4	L=10	L=20	L=30	L=50	L=70	L=100
p=0.25	**1.00368**	**1.00124**	**0.99923**	**0.98822**	0.90195	0.88395	0.83419	0.85435
p=1	1.00970	**1.00112**	**1.00050**	0.99501	0.82632	**0.83170**	0.79193	0.77658
p=4	1.09048	**1.04100**	1.00175	**0.97143**	0.81546	0.81599	0.77624	0.79345
p=9	1.23925	**1.13170**	**1.06793**	**1.00547**	0.82239	0.79777	0.77362	0.77989
p=39	**1.83981**	**1.46979**	**1.22424**	**1.17589**	0.84864	0.78660	0.77462	0.76134
p=99	**2.48444**	**2.30920**	1.45911	1.13117	0.91039	0.81862	0.75929	0.78665

(a) Results from our method

	L = 1	L = 4	L = 10	L = 20	L = 30	L = 50	L = 70	L = 100
p = 1/4	1.000732	0.998664	0.979373	0.957875	0.967347	0.906137	0.909529	0.910072
p = 1	1.012899	0.995453	0.996600	0.999953	1.001167	0.814216	0.976300	0.904669
p = 4	1.110758	1.026892	1.003398	0.970293	0.993897	0.997304	0.992948	0.992391
p = 9	1.258596	1.079733	1.027929	0.950493	0.966457	0.990411	0.968652	0.944635
p = 39	1.777229	1.407220	1.205311	1.064118	0.870776	0.942677	0.955548	0.909879
p = 99	2.392048	1.823024	1.471593	1.271550	1.139031	0.992178	0.927584	0.936422

(b) Results from the other group's method

Figure 6: Comparison of results with the other group.

Conclusions and Guidelines

When lead time is not large, we can use PPO to decide a policy. However, when lead time becomes larger, it might be better to apply constant-order policies, which require less computation yet give lower costs than PPO.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. (2018).
Stable baselines.
https://github.com/hill-a/stable-baselines.