A Appendix

A.1 Proof of Proposition 3.1

Proposition A.1 *Consider epoch j of the PARL algorithm with a RELU-network value-to-go estimate $\hat{V}_\theta^{\pi_{j-1}}(s)$ for some fixed policy π_{j-1}. Suppose $\pi_j, \hat{\pi}_j^\eta$ are the optimal policies as described in Problem (1) and its corresponding SAA approximation respectively. Then, $\forall\ s$,*

$$\lim_{\eta\to\infty} \hat{\pi}_j^\eta(s) = \pi_j(s).$$

where $\pi_j(s)$ is as described in Problem (1) and $\hat{\pi}_j^\eta(s)$ is the corresponding SAA approximation.

Proof Consider any state s and let $g(s,a,d) = R(s,a,d) + \gamma \hat{V}_\theta^{\pi_{j-1}}(T(s,a,d))$. We start by showing that $g^\eta(s,a,d)$ uniformly converges to $\mathbb{E}[g(s,a,D)]$ with probability 1. We prove this result by proving two main properties of $g(s,a,d)$: (i) $g(s,a,d)$ is *continuous* in a for almost every $d \in D$, and (ii) $g(s,a,d)$ is dominated by an *integrable function*. To prove (ii), we show that $g(s,a,d) \le C < \infty$ w.p. 1 $\forall a \in \mathcal{A}(s)$.

First, notice that $g(s,a,d)$ is an affine function of the immediate reward $R(s,a,d)$ and NN approximation of the value-to-go function. By assumption, the immediate reward follows these properties. Hence, to show these properties for $g(s,a,d)$, we only need to illustrate that the value-to-go estimation also follows these properties.

Consider the value-to-go approximation, simply denoted as $\hat{V}_\theta(T(s,a,d))$ with $\theta = (c, \{(W_k, b_k)\}_{k=1}^{K-1})$ denoting the parameters of the K-layer RELU-network. As $T(s,a,d)$ is continuous and $\hat{V}_\theta(s)$ is continuous, $\hat{V}_\theta(T(s,a,d))$ is continuous. Note that $T(s,a,d)$ lies in a bounded space for any realization of the uncertainty d. Furthermore, since the parameters of the NN θ are bounded, the outcome of each hidden layer, and subsequently the outcome of the NN are also bounded. This proves that the NN is uniformly dominated by an integrable function. Then, following Proposition 8 of [40], we have uniform convergence of $g^\eta(s,a,d)$ to $\mathbb{E}[g(s,a,D)]$ w.p. 1. Finally, convergence of the optimal solution follows from a direct application of Theorem 5.3 of [41], where we have used the fact that for all s the set of feasible actions is a bounded polyhedron $\mathcal{A}(s)$ and that for any η, the set of optimal actions $\hat{\pi}^\eta(s)$ is non-empty. This proves the final result. □

A.2 Math-Programming Actor in PARL for Inventory Management

Below we show mixed-integer linear reformulation of the inventory management MDP described in § 4 using PARL for the value-to-go terms. This formulation can be solved using commercially available standard optimization software such as CPLEX and Gurobi.

$$V(\mathbf{I}) = \max_{x_{l'l} \in Z^+, \mathbf{U^L} \le \mathbf{x} \le \mathbf{U^H}} \sum_{i=1}^{n} w_i \left[R(\mathbf{I}, \mathbf{x}, \mathbf{d}_i) + \gamma c^T \mathbf{z}_{Ki} \right] \tag{11}$$

$$\text{where} \quad R(\mathbf{I}, \mathbf{x}, \mathbf{d}_i) = \sum_{l \in \Lambda} R_l(\mathbf{I}_l, \mathbf{x}_l, \mathbf{d}_{li}) \quad \forall i, \tag{12}$$

$$R_l(\mathbf{I}_l, \mathbf{x}_l, \mathbf{d}_{li}) = p_l s_{li} - \sum_{l' \in O_l} \left[K_{l'l} w_{l'l} + C_{l'l} x_{l'l} \right] - h_l {I'_{li}}^0 - \delta B_{li}, \quad \forall\, l \in \Lambda, i \tag{13}$$

$$s_{li} \le d_{li}^d \quad \forall\, l \in \Lambda, i, \tag{14}$$

$$s_{li} \le \tilde{I}_{li}^0 \quad \forall\, l \in \Lambda, i, \tag{15}$$

$$w_{ll'} \le x_{ll'} \quad \forall\, l \in O_{l'}, l' \in \Lambda, \tag{16}$$

$$x_{ll'} \le U_{ll'}^H w_{ll'} \quad \forall\, l \in O_{l'}, l' \in \Lambda, \tag{17}$$

$$\tilde{I}_{li}^0 = I_l^0 + I_l^1 + d_{li}^p + \sum_{l' \in O_l} x_{l'l} \mathbb{1}_{L_{l'l}=0} - \sum_{\{l' \in \Lambda | l \in O_{l'}\}} x_{ll'}, \quad \forall\, l \in \Lambda, i, \tag{18}$$

$${I'_{li}}^0 = \tilde{I}_{li}^0 - s_{li} - B_{li}^0, \quad \forall\, l \in \Lambda, i, \tag{19}$$

$${I'_{li}}^j = I_l^{j+1} + \sum_{l' \in O_l} x_{l'l} \mathbb{1}_{L_{l'l}=j} - B_{li}^j, \quad \forall\, 1 \le j \le \max_{l' \in O_l} L_{l'l}, l \in \Lambda, i, \tag{20}$$

$$B^j_{li} \geq 0, \quad \forall\, j = 0, \cdots, \max_{l' \in O_l} L_{l'l}, l \in \Lambda, i, \tag{21}$$

$$(\mathbf{I}'_{-i}, z_{2qi}, y_{2qi}) \in P(W_{1q}, b_{1q}, \mathbf{0}, \bar{\mathbf{U}}) \quad \forall q \in N_1, \tag{22}$$

$$(z_{k,i}, z_{k+1,qi}, y_{k+1,qi}) \in P(W_{kq}, b_{kq}, l_k, u_k) \quad \forall q \in N_k, k = 2, \cdots, K-1. \tag{23}$$

We describe the extra notation that use in formulating the PARL actor as a MIP. Let $\mathbf{I}'_{-i}$ be the vector of $\mathbf{I}'_{li}$ across all the locations l which is an input to the DNN for every uncertainty sample i. We denote N_k as the number of neurons indexed by in layer k of the RELU-network and it is indexed by q. We let l_k and u_k are pre-computed lower and upper bounds of the inputs to every layer of the NN given the fixed bounds $[\mathbf{0}, \bar{\mathbf{U}}]$ of the first later and computed as described in § 3. We introduce a binary variable $w_{ll'}$ which is 1 if $x_{ll'} \geq 0$ and 0 otherwise. This is enforced with Eqs. (16-17). A sales variable s_{li} is modeled via Eqs. (14-15) to be less than demand d^d_{li} and the auxiliary inventory variable $\tilde{I}^0_{li}$. Since the sales is multiplied by price p_l, a positive constant, sales will be exactly the minimum of the demand and inventory. B^j_{li} is decision variable that captures the inventory spilled over which is positive if the state update variables I'_{li} exceeds $\bar{U}_l$ and 0 otherwise. This condition is enforced using a small linear penalty term δB_{li} in the objective.

A.3 Hyperparameters and parameter tuning

In this section, we discuss the different parameters selected for PARL and the other benchmark methods and the corresponding tuning procedure.

A.3.1 Fixed hyper parameters

Here we report the fixed set of hyper parameters used by all methods. These were determined based on two factors: (1) the commonly used settings across the RL literature (for example 64x64 architecture and batch size 64 is most commonly used across many different problems and methods), and by sampling random combinations from a large grid of hyper parameters and comparing results trends to narrow down the set of hyper parameters to consider to consistently well-performing values and reasonable ranges.

This was an iterative process where we tried a range of hyper parameters, then refined. We focused mostly on the PPO method at first as it was the first one we had implemented and started testing in this supply chain setting, but it gave us a general sense of what kind of hyper parameters had a chance at working well for these problems and environments. In particular, we tried larger network architectures, including 128x128, 512x512, 1024x1024, 128x128x128, 512x256, 1024x512, 1024x512x256, 128x32, 512x128, 512x256x64, but generally did not see significant improvement across environments, especially when using the continuous action and state spaces (perhaps also because the limited size of our observation and action spaces) - so decided to fix everything to the standard 64x64 for fair comparison, and improved computationally efficiency. We tried 32, 64, and 128 batch size, but also did not see significant difference in what gave the best results, so set this to the most commonly used 64. We also consistently saw ReLU activation performing as good or better than tanh (overall it gave close but slightly better results) - so fixed the activation to ReLU across methods for fair comparison, and since ReLU is known to enable more efficient optimization [42, 43]. These initial experiments also revealed that the standard gamma value of 0.99 consistently gave much poorer results than smaller gamma values for most environments (trying 0.99, 0.9, 0.85, 0.8 and 0.75) - so we included smaller gamma values in our hyper-parameter grids but still kept the option of the traditionally used 0.99 gamma for the benchmark RL methods in case they were able to factor in longer-term impact better. Furthermore, we originally tried different observation and action space representations, including discrete, multi-discrete and continuous for each decision variable (normalized to be within -1 to 1). Interestingly, and somewhat surprisingly to us, we found both continuous action representation and continuous state representation to work as good or better than multi-discrete (and much better than discrete) while offering significant computational speedup, even enabling significantly higher mean reward in some cases, so we fixed the state and action representations to continuous (i.e., the box space - continuous for each action variable and normalized to be within -1 to 1). This may be because of the much fewer model parameters required, and the main focus in the majority of RL research on continuous spaces, potentially making the existing set of algorithms most suited for such spaces. Additionally for number of internal training iterations per collected buffer (PPO-specific setting) we tried 10, 20, and 40, and 70 found best results with 10-20, so fixed the amount to 20. Finally, for the number of steps per epoch / update (PPO and A2C-specific

setting) we tried 512, 1024 and 2048 and found the larger number to give better results generally so fixed this to 2048. For PPO we also found early stopping the policy update per epoch, based on KL-divergence threshold of default 0.15, to consistently provide better results than not using this.

These initial evaluations using PPO were also used to set the suitable range of hyper-parameters for our PARL algorithm. In particular we fixed all the hyper-parameters as specified in Table 3 (as with the other methods), and also the 20 internal train iterations as with PPO. Additionally we tried a few values of gamma before fixing it to 0.75 (included in the range of gammas for the benchmark RL algorithms) in order to speed up experiments for PARL by having a reduced grid of hyper parameters.

The final set of fixed hyper-parameters used across all experiments, models, runs, and environments are given in Table 3.

Table 3: Fixed set of hyper-parameters used for all methods

Batch size	Net arch. (hidden layers per net)	Activation	State representation	Action representation	Epoch length
64	64x64	ReLU	continuous (normalized)	continuous (normalized)	2048

A.3.2 Tuning hyper parameters

Here we show the set of tuning hyper parameters that were applied to each method and all environments, from which the best set of hyper parameters were selected per environment for each method. PPO and A2C tuning hyper parameters are given in Table 4 and SAC and TD3 in Table 5. This best set of hyper parameters was then used for the evaluation of the RL model - by retraining 10 different times with different random seeds using those best hyper paramaters for each method, and reporting statistics on the 20-episode evaluations of the best epoch model across the 10 runs.

Besides initial experimentation to set some of the hyper-parameters as mentioned in the previous section, and specific hyper parameters that were tuned here, all other hyper parameters were set at their default values in the Stable Baselines 3 implementation (so please refer to the API[1] for other settings not listed here). Note, default optimizers are used, which is ADAM for all except A2C which uses RMSprop by default.

Table 4: Tuning hyper parameters and additional fixed hyper parameters for PPO and A2C - we vary gamma, learning rate, and value function coefficient - resulting in 36 hyper parameter combinations

Hyper Parameters for PPO and A2C	**Value(s)**
Discount Factor - Gamma *(G)*	0.99, 0.9, 0.80, 0.75
Learning rate *(LR)*	0.01, 0.003, 0.0003
Value function coefficient (in loss) *(VFC)*	0.5, 1.0, 3.0
Number of steps to run per update (epoch length)	2048
Max gradient norm (for clipping)	0.5
GAE lambda (trade-off bias vs. variance for Generalized Advantage Estimator)	0.95 (PPO) and 1.0 (A2C) (defaults)
Number of epochs to optimize surrogate loss (internal train iterations per update - PPO only)	20
KL divergence threshold for policy update early stopping per epoch (PPO only)	0.15 (“target_kl”=0.1)
Clip range (PPO only)	0.2
RMSprop epsilon (A2C only)	1e-05

[1] https://stable-baselines3.readthedocs.io/en/master/

Table 5: Tuning hyper parameters and additional fixed hyper parameters for SAC and TD3 - we vary gamma, learning rate, and exploration options - resulting in 32 hyper parameter combinations

Hyper Parameters for SAC and TD3	**Value(s)**
Discount Factor - Gamma *(G)*	0.99, 0.9, 0.80, 0.75
Learning rate *(LR)*	0.01, 0.003, 0.0003, 0.00003
Use generalized State Dependent Exploration vs. Action Noise Exploration (SAC) or Action Noise vs. not (TD3) *(EO)*	True, False
Tau (soft update coefficient)	0.005
Replay buffer size	10^5
Entropy regularization coefficient (SAC only)	auto

Table 6: Tuning hyper parameters for PARL

Hyper Parameters for PARL	**Value(s)**
Discount Factor - Gamma *(G)*	0.99, 0.9, 0.8, 0.75
Learning rate *(LR)*	0.01, 0.003, 0.001
Sample approximation averaging (SAA) approach used to generate demand samples	quantile, random
SAA samples per step	2, 3

A.3.3 Evaluation hyper parameters

Here we show the selected set of best hyper parameters used for each benchmark RL method and environment, in Table 7. These were selected based on what gave the best average reward (maximum over the training epochs), averaged across 10 different model runs for that hyper-parameter combination.

Table 7: Best hyper parameters selected for each environment and method, for the benchmark RL methods. See Tables 4 and 5 for hyper parameter abbreviations.

method setting	SAC	TD3	PPO	A2C
1S-3R-High	G=0.9 LR=0.01 EO=True	G=0.9 LR=0.0003 EO=False	G=0.9 LR=0.003 VFC=1.0	G=0.8 LR=0.003 VFC=0.5
1S-3R	G=0.75 LR=0.003 EO=True	G=0.8 LR=0.0003 EO=False	G=0.8 LR=0.003 VFC=1.0	G=0.8 LR=0.003 VFC=0.5
1S-10R	G=0.8 LR=0.003 EO=True	G=0.9 LR=0.0003 EO=True	G=0.8 LR=0.003 VFC=1.0	G=0.9 LR=0.003 VFC=1.0
1S-2W-3R	G=0.8 LR=0.003 EO=False	G=0.9 LR=0.0003 EO=False	G=0.8 LR=0.003 VFC=1.0	G=0.8 LR=0.003 VFC=3.0
1S-2W-3R (DS)	G=0.9 LR=0.0003 EO=True	G=0.9 LR=0.0003 EO=True	G=0.75 LR=0.003 VFC=3.0	G=0.8 LR=0.003 VFC=0.5

For PARL, a common set of hyper-parameters were used across all five settings. The discount factor was set to 0.75, learning-rate was set to 0.001, and the sample-averaging approach used was quantile sampling with 3 demand-samples per step.

A.4 Supply chain environment problem parameters

In the table below we provide the details of the environment parameters in a concise format for the 5 different supply chain networks that we study and we describe it below.

We assume deterministic production constant per-period production and that the variability is only in the demand. The parameters are provided node type - Retailer (R), Supplier (S) and Warehouse (W) - and then by links between them. Whenever they are provided in a list format, they correspond to the retailers and warehouses in a chronological order (i.e., R1, R2, R3 or W1, W2). Also, when there are more nodes or links than the parameters (few elements in the list specified in the table), it means the parameters list in repeated in a cyclic fashion. For example the lead time (S or W to R) for the environment 1S-10R is given by [1,2,3] and this means the lead time for links [S-R1, S-R2,....,S-R10] is (1,2,3,1,2,3,1,2,3,1). The notation for the distributions used are $N(\mu, \sigma)$ for a normal distribution with mean μ and standard deviation σ and $U(a, b)$ discrete uniform between a and b. Note that because demand is discrete and positive, when we use a normal distribution, we round and take the positive parts of the realizations. The lead-time list has a tuple representation in the dual-sourcing setting to represent the lead time of a retailer from the two different warehouses. For example (1,5) in the list represents the lead time for W1-R1 and W2-R2. Lastly, the environment only imposes spillage cost at the node if the on-hand exceeds the holding capacity, while PARL MIP actor imposes it on the pipeline inventory actor to ensure it is not over-ordering. As the maximum order is less or equal to the holding capacity on various links, such a constraint in PARL actor helps to avoid over-ordering in dual sourcing settings. The ability of PARL to gainfully incorporate such constraints exactly when they are known, possibly provides it an edge over vanilla RL methods.

Parameters	1S-3R-High	1S-3R	1S-10R	1S-2W-3R	1S-2W-3R (DS)
Retailer demand distribution	[N(2,10)]	[N(2,10)]	[N(2,10)]	[N(2,10)]	[N(2,10)]
Retailer revenue per item	[50]	[50]	[50]	[50]	[50]
Retailer holding cost	[1,2,4]	[1,2,4]	[1,2,4,8]	[1,2,4]	[1,2,4]
Retailer holding Capacity	[50]	[50]	[50]	[50]	[50]
Supplier production per step	15	10	25	10	10
Supplier holding capacity	100	100	150	100	100
Warehouse holding cost	-	-	-	[0.5]	[0.5, 0.1]
Warehouse holding capacity	-	-	-	[150]	[150]
Spillage cost at S,W,R	[10]	[10]	[10]	[10]	[10]
Lead time (S or W to R)	[1,2,3]	[1,2,3]	[1,2,3]	[1,2,3]	[(1,5),(2,6),(3,7)]
Lead time (S to W)	-	-	-	[2]	[2]
Fixed order cost (S or W to R)	[50]	[50]	[50]	[50]	[50]
Fixed order cost (S to W)	-	-	-	[0]	[0]
Variable order cost (any link)	[0]	[0]	[0]	[0]	[0]
Maximum order (any link)	[50]	[50]	[50]	[50]	[50]
Initial inventory distribution (node or link)	[U(0,4)]	[U(0,4)]	[U(0,4)]	[U(0,4)]	[U(0,4)]

Table 8: Environment parameters for different supply chains studied

Then for all methods, given a selected best hyper-parameter combination per method and environment, we trained 10 different models for each (i.e., with different random seeds). Each training run used an episode (trajectory) length of 256, an epoch length of 2048 and 500 epochs. Finally, for each trained model, we took the best epoch model according to the observed reward during training, and then evaluated each of the 10 trained models with 20 test episodes, initialized with random seeds, and using trajectory length 256, to get our final reported mean and standard deviation per method and environment.

A.5 Comparison of quantile and random sampling in PARL

Here we compare the use of quantile sampling and random sampling to generate realizations of the uncertainty in (3). For the five settings under consideration, we compare the per-step reward and per-step training time across the two sampling approaches.

As can be seen in Table 9, random sampling yields per-step rewards which are close to those obtained via quantile sampling. In terms of training time, random sampling is slower in certain settings (e.g. 1S-3R-High and 1S-10R), with higher per-step train-time average and variance.

Setting	PARL-quantile per-step reward	PARL-random per-step reward	PARL-quantile per-step train-time (s)	PARL-random per-step train-time (s)
1S-3R-High	514.8 ± 5.3 514.3	505.3 ± 11.0 505.1	0.178 ± 0.06	0.457 ± 0.26
1S-3R	400.3 ± 3.3 400.8	399.5 ± 2.8 400.8	0.051 ± 0.01	0.053 ± 0.01
1S-10R	1006.3 ± 29.5 1015.7	1005.4 ± 21.1 1007.3	0.089 ± 0.03	0.12 ± 0.07
1S-2W-3R	398.3 ± 2.5 399.7	395.3 ± 3.1 395.9	0.051 ± 0.01	0.050 ± 0.01
1S-2W-3R (DS)	405.4 ± 2.0 405.9	398.9 ± 9.7 402.0	0.044 ± 0.01	0.043 ± 0.01

Table 9: Comparison of quantile and random sampling in PARL.