Reinforcement Learning Provides a Flexible Approach for Realistic Supply Chain Safety Stock Optimisation 论文复现

摘要

安全库存就是在最小化供应链中所有层次的总安全库存成本的同时确保防止缺货，且该问题已被研究多年，但多数研究重点放在推导具有可证明最优性的算法上，通常以简化现实为代价。而公司喜欢使用能够满足在实际复杂环境中的解决方案。强化学习 (RL) 算法能够在安全库存优化中动态处理复杂环境上发挥作用，同时优化安全库存水平和订单数量规则。

关键词：供应链；安全库存；强化学习

1 引言

不断增加的供应链复杂性和经济不确定性同时影响着供需。供应链管理者需要改进库存管理策略，以应对新出现的不确定性 [3]。本文 [6] 旨在比较现有基于分析模型和提出的基于强化学习模型所得到的见解。为了获得可比较的结果，研究采用了一个标准的串行供应链问题，其分析解已经被深入研究，最优解可以被推导和证明。通过将安全库存问题作为 Q-Learning [10]、时间差异优势行动者-批评 (A2C) 和多智能体时间差异优势行动者-批评 (MAA2C) 强化学习方法的背景来检验强化学习的使用。建立了基线分析模型和测试问题 [2]，以便对结果进行交叉比较。研究结果表明，虽然与分析得出的基线结果相比，RL 略显次优，但它能够优化库存策略的安全库存水平 [9] 和订单数量参数。但是，强化学习在处理安全库存优化问题时面临高计算复杂性的挑战，因此从强化学习中获得的好处必须与获得解决方案所需的时间相平衡。[8]

本文组织如下。首先，进行问题建模，将库存优化问题转为数学表达。并针对具体情况进行数学规划 [2]。其次，RL 被认为是解决这些问题的一种潜在方法。故本文详细介绍了用于解决该问题的三种 RL 算法设计。对 RL 算法的在设计的具体情况上的实验结果进行了分析和讨论，并对未来的研究方向提出了建议。

2 相关工作

在本节中将会探讨安全库存方面的相关知识，对其进行数学化表示。

2.1 问题建模

安全库存配置问题的最简单表述如下: 最小化供应链中所有层次的总安全库存成本，同时确保防止缺货 [4]。本文基于保证服务（GSM [1] [7]）模型进行研究。最简单的 GSM 将特定代理 j 的库存表述为需求的平均值 μ_j 和标准差 σ_j（建模为正态分布）、供应商的交货时间 SI_j、处理时间 T_j、服务时间 S_j、服务水平 z_j 和安全库存函数 $\mathrm{SS}_j()$ 的函数，如公式 1所示。

$$I_j = \mu_j \times (SI_j + T_j - S_j) + SS_j(z_j, \sigma_j, SI_j, T_j, S_j) \tag{1}$$

由于平均需求是一个固定值，因此只考虑方程的安全库存部分进行优化。假设 h_j 代表代理 j 的库存保持成本，优化问题可以被表述为供应链中所有 N 个代理的最小化问题，如公式2所示。

$$\sum_{j \in N} h_j \times SS_j(z_j, \sigma_j, SI_j, T_j, S_j) \tag{2}$$

安全库存函数 SS_j 的定义表示如公式3所示:

$$SS_j(z_j, \sigma_j, SI_j, T_j, S_j) = z_j \times \sigma_j \times \sqrt{SI_j + T_j - S_j} \tag{3}$$

式中 z_j 为服务水平，σ_j 表示节点 j 处需求标准差的乘数，影响覆盖多少需求界限，SI_j 为节点 j 的供应商的服务时间，T_j 则是加工时间，S_j 为向客户保证的服务时间。因此，安全库存被定义为覆盖提前期 (SI_j+T_j-S_j) 的库存。

2.2 数学规划

在三级供应链中有三个代理商: 零售商、仓库和工厂。零售商采用 (Q, r_p) 库存策略，Q 零售商的订单数量为正态分布，当其库存位置达到再订货点 r_p 时，将其放入仓库。作为一个基于拉动的供应链，工厂和仓库只会在看到零售商的需求时采购或生产库存。工厂和仓库可以选择订购或生产超过需求的产品，这样任何额外的产品都将在下一个周期中作为库存保存。有了更多的库存，工厂和仓库可以为客户提供更短的服务时间。任何无法满足的需求 (例如，当实际履行超过保证的服务时间) 将被视为缺货。问题的目标是最小化系统中的安全库存，同时确保没有缺货。本文假设一个合作博弈设置，其中代理一起工作以优化一个共同的目标函数。使用强化学习方法，需要定义三组变量: 代理状态、代理行为和环境模型。针对不同的成本分配进行两组实验。将安全库存问题表述为有界多面体上的凹极小化问题 [7]。根据公式，GSM 问题表示为：

最小化:

$$h_{\text{factory}} \times z_{\text{factory}} \times \sigma \times \sqrt{0 + 1 + S_{\text{factory}}} + h_{\text{factory}} \times z_{\text{factory}} \times \sigma \times \sqrt{S_{\text{factory}}} + 3 - S_{\text{warehouse}} \tag{4}$$

其中:

$$S_{\text{factory}} \leq 0 + 1 \tag{5}$$

$$S_{\text{warehouse}} \leq S_{\text{factory}} + 3 \tag{6}$$

$$S_{\text{warehouse}} \leq 3 \tag{7}$$

其中 z_j 为服务水平 =3，σ 为需求的标准差 =1。因此，工厂的安全库存量为 $3 \times 1 \times \sqrt{1 - S_{\text{factory}}}$ 仓库的安全库存量为 $3 \times 1 \times \sqrt{S_{\text{factory}} + 3 - S_{\text{warehouse}}}$。两代理的总库存可以写成平均需求

和安全库存之和。其中 $I_{\text{factory}} = \mu \times (1 - S_{\text{factory}}) + 3 \times 1 \times \sqrt{1 - S_{\text{factory}}}$ 和 $I_{\text{warehouse}} = \mu \times (S_{\text{factory}} + 3 - S_{\text{warehouse}}) + 3 \times 1 \times \sqrt{S_{\text{factory}} + 3 - S_{\text{warehouse}}}$ 本文分为两种具体情况：

情况 1：$\text{S}_{\text{factory}} = 1$; $\text{S}_{\text{warehouse}} = 3$, 或 $\text{I}_{\text{factory}} = 0$; $\text{I}_{\text{warehouse}} = 13$;$\text{r}_p = 6$

表 1. 用枚举解决情况 1

Service Level	Inventory Level	Total Safety Stock Cost
$\text{S}_{\text{factory}} = 0$, $\text{S}_{\text{warehouse}} = 0$	$\text{I}_{\text{factory}} = 13$, $\text{I}_{\text{warehouse}} = 30 + 3\sqrt{3}$	$3000 + 15\sqrt{3}$
$\text{S}_{\text{factory}} = 0$, $\text{S}_{\text{warehouse}} = 3$	$\text{I}_{\text{factory}} = 13$, $\text{I}_{\text{warehouse}} = 0$	3000
$\text{S}_{\text{factory}} = 1$, $\text{S}_{\text{warehouse}} = 0$	$\text{I}_{\text{factory}} = 0$, $\text{I}_{\text{warehouse}} = 46$	30
$\text{S}_{\text{factory}} = 1$, $\text{S}_{\text{warehouse}} = 3$	$\text{I}_{\text{factory}} = 0$, $\text{I}_{\text{warehouse}} = 13$	15

情况 2：$\text{S}_{\text{factory}} = 0$; $\text{S}_{\text{warehouse}} = 3$, 或 $\text{I}_{\text{factory}} = 13$; $\text{I}_{\text{warehouse}} = 0$;$\text{r}_p = 6$

表 2. 用枚举解决情况 2

Service Level	Inventory Level	Total Safety Stock Cost
$\text{S}_{\text{factory}} = 0$, $\text{S}_{\text{warehouse}} = 0$	$\text{I}_{\text{factory}} = 13$, $\text{I}_{\text{warehouse}} = 30 + 3\sqrt{3}$	$15 + 3000\sqrt{3}$
$\text{S}_{\text{factory}} = 0$, $\text{S}_{\text{warehouse}} = 3$	$\text{I}_{\text{factory}} = 13$, $\text{I}_{\text{warehouse}} = 0$	15
$\text{S}_{\text{factory}} = 1$, $\text{S}_{\text{warehouse}} = 0$	$\text{I}_{\text{factory}} = 0$, $\text{I}_{\text{warehouse}} = 46$	6000
$\text{S}_{\text{factory}} = 1$, $\text{S}_{\text{warehouse}} = 3$	$\text{I}_{\text{factory}} = 0$, $\text{I}_{\text{warehouse}} = 13$	3000

3 本文方法

在本节中将会介绍三种强化学习算法在以上两个安全库存问题上的实现，与上述基于分析的方法得到的见解进行并置。对问题进行模拟，该模拟包括代理商根据客户的需求运送或生产物品的周期。对于每一个周期，在满足需求后，向所有代理提供如下的联合奖励。库存和缺货量越高，奖励就越负。因此，目标是使奖励最大化。

$$\text{reward} = -h_{\text{factory}} \times I_{\text{factory}} + h_{\text{warehouse}} \times I_{\text{warehouse}} + \eta \times \text{stockouts} \tag{8}$$

3.1 Q-learning 算法

Planneragent 看到一个状态向量 $\text{s}_t = [\text{I}_{\text{factory}}\ \text{I}_{\text{warehouse}}\ r_p^t]$，并据此决定一个动作向量 $\text{a}_\text{t} = [\text{Q}_{\text{factory}}\ \text{Q}_{\text{warehouse}}\ r_p^{t+1}]$。代理使用 Q-learning 算法1进行训练，其中 Q 函数中的参数为折扣因子 γ=0.2，学习率 α=0.8。

3.2 Temporal Difference Advantage Actor-Critic(A2C) 算法

虽然 Q-Learning 有效，但它不能很好地扩展动作空间的大小，因为它假设动作是离散的。此外，它不能泛化状态和动作，因为它只使用表格表示。在本节中将会探索另一种称为策略梯度的方法，它可以在连续空间上操作并进行推广。训练使用算法2(见附录)。，Actor 和 Critic 代理都用神经网络表示。使用由 tensorflowkeras 软件包规定的默认超参数。在训练过程中，Actor 和 Critic 都是迭代训练的。然而，在测试期间，只使用 Actor。

3.3 Multi-agent Temporal Difference Advantage Actor-Critic (MAA2C) 算法

尽管集中式方法有效，但它们会随着代理的数量呈指数级增长。在本节中将探讨分散的方法，其中每个代理只能看到自己的库存和需求。因此，该模型将随代理的数量线性扩展。正如算法3(见附录) 中提到的，这种方法需要一个只在训练期间出现的集中批评。同时，在测试中，Actor 都是分散的。Critic 看到的是联合状态 $s_t = [I_{factory}\ I_{warehouse}\ r_p^t]$，而 Actor 看到的是局部状态，即工厂的状态 $s_t^{factory} = [I_{factory}\ Q_{warehouse}\ ]$ 和仓库的状态 $s_t^{warehouse} = [I_{warehouse}\ Q_{retailer}\ ]$。

4 复现细节

4.1 与已有开源代码对比

本实验的 Q-learning 算法和 A2C 算法参考原论文的公开代码做出了部分改进。MAA2C 算法在实现过程中没有单独重写一个多智能体环境，只为每个 agent 生成一个单独的单智能体环境，同时都使用自己的策略网络来进行决策。

分别使用三种不同的算法对两种不同成本的情况分别进行实现

（1）Q-learning

构造 Q 表: 二元组（状态-三级供应链的库存状态，动作- 服务时间，订单数量，再订货点）训练：agent 使用 Q-learning 算法与环境进行交互，运行多个 episode, 在每个 episode 中，通过 -greedy 策略选择动作观察奖励, 更新 Q 表中的存储的状态和动作元组学习策略

测试:禁用 agent 的探索行为,根据训练过程中学习到的策略在环境中运行一整个 episode，输出在测试中的部分性能指标，即两个节点的库存、再订货点及奖励

改进：调整缺货成本的权重及训练次数，使重新订货点和库存接近最优

（2）A2C

构造策略网络和价值网络：多层神经网络，用于估计状态值和动作值训练：使用了多个随机初始条件来训练 agent，并记录了每个训练周期的平均库存水平

测试：使用训练好的策略网络来预测最优的安全库存水平，并与基准模型进行比较。使用 95%置信区间来评估模型性能

改进：将模型中隐藏层改为由 1000 改成包含更少神经元的结构，同时将模型中的采样方式改为使用手动采样，加入剪切限制范围为 [0,30]（最大库存容量）

（3）MAA2C

构造策略网络和价值网络：每个智能体有自己的策略网络，用于输出动作值，由于所有智能体的目标均为使总成本最小，故所有智能体有相同的 reward，公用一个相同的价值网络

训练：随机初始化多个智能体，使用 TD 算法训练价值网络，每个智能体计算当前和下一状态的预测值，使用均方误差损失函数和 Adam 优化器进行参数优化，通过调整状态和目标值更新价值网络参数。策略网络通过最大化期望累积回报的策略梯度来训练，最大化在每个状态下的期望回报, 更新策略网络参数。

测试：每个代理都使用自己的策略网络进行决策，输出的动作向量作为代理的决策，即再订货点与订单数量

4.2 实验环境搭建

基本环境：Ubuntu18.04, Python 3.8, CUDA 11.3, cuDNN 8, NVCC, Pytorch 1.11.0, torchvision 0.12.0, torchaudio 0.11.0

项目环境：Tensorflow 2.4.0, tensorflow-probability 0.12.0, tqdm4.65.0, keras 2.13.1

使用一个标准的串行供应链问题作为仿真环境 [5]。

4.3 创新点

本文的创新点是将强化学习应用于供应链安全库存优化问题。传统的安全库存优化方法通常依赖于分析模型和规则的推导，但这些方法难以应对复杂的供应链环境和实时需求的波动。本文通过引入强化学习算法，特别是 Q-Learning、Temporal Difference Advantage Actor-Critic (A2C) 和 Multi-agent A2C，通过训练智能代理来自主学习最优的库存管理策略。

这种基于强化学习的方法具有以下优势：

(1) 能够适应不确定性和动态变化的供应链环境。通过与环境的交互，智能代理可以自主学习并适应不同的需求和库存情况。

(2) 能够学习复杂的库存策略。与传统的基于规则的方法相比，强化学习方法能够学习更复杂、个性化的库存管理策略，并能够根据实际情况做出实时调整。

(3) 能够处理连续动作空间。与传统的 Q-Learning 方法不同，本文还引入了 A2C 和 Multi-agent A2C 算法，这些算法能够处理连续动作空间，提高了模型在实际问题中的可扩展性和泛化性能。

5 实验结果分析

本文的实验结果分析主要涉及三种算法：Q-Learning、A2C 和 MAA2C。

Q-Learning 算法的实验结果显示，代理的行为虽然不是最优的，但与解析推导的结果非常接近。这意味着 Q-Learning 算法能够在标准串行供应链问题中优化安全库存水平和订货量规则。

表 3. Q-Learning 情况 1 和情况 2 95 % 置信区间平均库存水平与论文中实验结果对比

	Case 1 RL	Case 1 RL(my)	Case 1 Analytical	Case 2 RL	Case 2 RL(my)	Case 2 Analytical
r_p	[1.02,1.12]	[3.85,3.91]	6	[3.58,3.86]	[3.29,4.17]	6
$I_{warehouse}$	[11.22,13.34]	[10.34,12.10]	13	[1.29,1.7]	[1.01,2.21]	0
$I_{factory}$	[0.41,0.98]	[0.98,1.30]	0	[11.91,13.18]	[11.13,14.79]	13

A2C 算法是一种优化安全库存水平和订货量规则的策略梯度算法。实验结果表明，A2C 算法在训练过程中能够逐步提高奖励，最终得到接近最优的结果。然而，由于 A2C 算法使用的是离散动作空间，不适合处理大型动作空间的问题。

为解决离散动作空间的限制，MAA2C 算法被引入。该算法中的每个代理都独立地训练自己的策略网络。实验结果表明，该算法能够在标准串行供应链问题中获得接近最优的结果。相比于 A2C 算法，该算法具有较好的可扩展性，适用于处理多智能体系统。

表 4. A2C 情况 1 和情况 2 95 % 置信区间平均库存水平与论文中实验结果对比

	Case 1 RL	Case 1 RL(my)	Case 1 Analytical	Case 2 RL	Case 2 RL(my)	Case 2 Analytical
r_p	[2.11,2.84]	[3.98,4.46]	6	[3.18,3.77]	[4.55,4.86]	6
$I_{warehouse}$	[12.58,15.11]	[12.02,13.89]	13	[0.37,0.85]	[0.16,1.10]	0
$I_{factory}$	[1.66,2.87]	[0.99,1.78]	0	[8.97,11.24]	[10.48,12.88]	13

表 5. MAA2C 情况 1 和情况 2 95 % 置信区间平均库存水平与论文中实验结果对比

	Case 1 RL	Case 1 RL(my)	Case 1 Analytical	Case 2 RL	Case 2 RL(my)	Case 2 Analytical
r_p	[2.73,3.09]	[3.55,4.67]	6	[2.94,3.46]	[4.53,4.70]	6
$I_{warehouse}$	[13.98,19.46]	[12.60,14.12]	13	[1.97,2.96]	[1.25,4.56]	0
$I_{factory}$	[0.58,1.64]	[0.23,2.46]	0	[8.64,10.98]	[12.80,13.72]	13

实验结果表明，RL 算法在供应链安全库存优化问题中具有潜力。虽然与解析模型相比略有不足，但 RL 算法能够生成更复杂的库存策略。此外，实验还表明，MAA2C 算法在解决多智能体系统中的安全库存优化问题时具有良好的性能和可扩展性。

6 总结与展望

本文提出了一种基于强化学习的方法，用于解决供应链安全库存优化问题。通过训练智能代理来自主学习最优的库存管理策略，这种方法能够适应不确定性和动态变化的供应链环境，学习复杂的库存策略，并处理连续动作空间。实验结果表明，强化学习方法在实际应用中具有良好的效果和灵活性。

尽管本文已经展示了强化学习在供应链安全库存优化中的潜力，但仍然存在一些改进和未来的研究方向。首先，本文采用了三种常用的强化学习算法，未来可以进一步探索其他强化学习算法的应用，比如深度强化学习方法，以提高模型的性能和泛化能力。其次，本文在实验中使用了标准的串行供应链问题，但实际供应链通常更为复杂。未来的研究可以考虑更真实的供应链环境和约束条件，如多阶段供应链、多产品和多渠道供应链等，并结合实际数据对模型进行验证。

参考文献

[1] Andrew J. Clark and Herbert Scarf. Optimal policies for a multi-echelon inventory problem. *Management Science*, 6(4):475–490, 1960.

[2] Paul Glasserman and Sridhar Tayur. Sensitivity analysis for base-stock levels in multiechelon production-inventory systems. *Management Science*, 41(2):263–281, 1995.

[3] Salal Humair et al. Incorporating stochastic lead times into the guaranteed service model of safety stock optimization. *Interfaces*, 43(5):421–434, 2013.

[4] Karl Inderfurth. Safety stock optimization in multi-stage inventory systems. *International Journal of Production Economics*, 24(1-2):103–113, 1991.

[5] Andreas Klemmt et al. Simulation-based optimization vs. mathematical programming: A hybrid approach for optimizing scheduling problems. *Robotics and Computer-Integrated Manufacturing*, 25(6):917–925, 2009.

[6] Edward Elson Kosasih and Alexandra Brintrup. Reinforcement learning provides a flexible approach for realistic supply chain safety stock optimisation. *IFAC-PapersOnLine*, 55(10):1539–1544, 2022.

[7] Kenneth F Simpson Jr. In-process inventories. *Operations Research*, 6(6):863–873, 1958.

[8] Carles Sitompul and et al. Safety stock placement problem in capacitated supply chains. *International Journal of Production Research*, 46(17):4709–4727, 2008.

[9] Amit Surana et al. Supply-chain networks: a complex adaptive systems perspective. *International Journal of Production Research*, 43(20):4235–4265, 2005.

[10] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Algorithm 1 Q-Learning

Result: $Q(s, a)$ for $s \in$ States, $a \in$ Actions(s)
Hyperparameters: Learning rate $\alpha \in (0, 1]$, discount factor $\gamma \in (0, 1]$, epsilon-greedy factor $\varepsilon \in (0, 1]$
Initialize $Q(s, a)$ for $s \in$ States, $a \in$ Actions(s) arbitrarily
for each episode **do**
    Initialize s_{factory}
    for t in $[0, T]$ **do**
        if random:random() $< \varepsilon$ **then**
            $a_t =$ randomly selected from Actions(s_t)
        else
            $a_t = \arg\max_a(Q(s_t, a))$
        end if
        Take action a_t, observe r_{t+1} and s_{t+1}
        Find off-policy action $a_{0,t+1} = \arg\max_a(Q(s_{t+1}, a))$
        $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{0,t+1}) - Q(s_t, a_t)]$
    end for
end for

Algorithm 2 TD Advantage Actor-Critic (A2C)

Result: $V(s)$ and $\pi(a|s)$ for $s \in$ States and $a \in$ Actions(s)
Hyperparameters: Learning rate $\alpha_\theta, \alpha_w \in (0, 1]$, discount factor $\gamma \in (0, 1]$, standard deviation σ
Initialize policy network weights θ and value network weights w arbitrarily
for each episode **do**
    Initialize s_{factory}
    for t in $[0, T]$ **do**
        $a_t \sim \pi(\cdot|s_t; \theta)$
        Take action a_t, observe r_{t+1} and s_{t+1}
        $\delta = r_{t+1} + \gamma V(s_{t+1}|w) - V(s_t|w)$
        Update value parameter w: $w \leftarrow w + \alpha_w \delta \nabla V(s_t|w)$
        Update policy parameter θ: $\theta \leftarrow \theta + \alpha_\theta \delta \nabla \ln \pi(a_t|s_t; \theta)$
    end for
end for

Algorithm 3 Multi-Agent TD Advantage Actor-Critic (MAA2C)

Result: $V([s_1; \ldots; s_N])$ and $\pi_i(a_i|s_i)$ for $s_i \in$ States and $a_i \in$ Actions(s_i) for all agents $i \in \{1, \ldots, N\}$
Hyperparameters: Learning rate $\alpha_\theta = \alpha_{\theta_1} = \ldots = \alpha_{\theta_N}$, $\alpha_w \in (0, 1]$, discount factor $\gamma \in (0, 1]$, standard deviation σ
Initialize policy network weights $\theta_1, \ldots, \theta_N$ for each agent $i \in \{1, \ldots, N\}$ and value network weights w arbitrarily
for each episode **do**
    Initialize $S_{\text{factory}} = [s_1^0; \ldots; s_N^0]$
    for t in $[0, T]$ **do**
        for each agent $i \in \{1, \ldots, N\}$ **do**
            $a_{i,t} \sim \pi_i(\cdot|s_{i,t}; \theta_i)$
        end for
        Take action $a_t = [a_{1,t}; \ldots; a_{N,t}]$, observe joint reward r_{t+1} and joint state $s_{t+1} = [s_1^t; \ldots; s_N^t]$
        Joint $\delta = r_{t+1} + \gamma V(s_{t+1}|w) - V(s_t|w)$
        Update value parameter w: $w \leftarrow w + \alpha_w \delta \nabla V([s_1^t; \ldots; s_N^t]|w)$
        for each agent $i \in \{1, \ldots, N\}$ **do**
            Update policy parameter θ_i: $\theta_i \leftarrow \theta_i + \alpha_\theta \delta \nabla \ln \pi_i(a_{i,t}|s_{i,t}; \theta_i)$
        end for
    end for
end for