基于循环神经网络架构的大规模供应链网络的仿真和优化

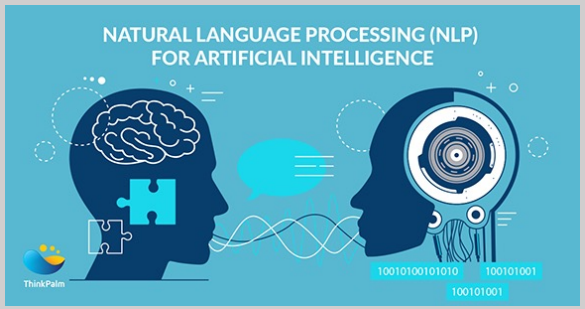
复旦大学管理学院 & 大数据学院
洪 流 教授

论文链接：https://arxiv.org/abs/2201.05868

近十年来人工智能技术不断取得突破性进展

推荐系统

YouTube, Netflix, Amazon，字节跳动



自然语言处理

Google Translate, 舆情监测，自动问答

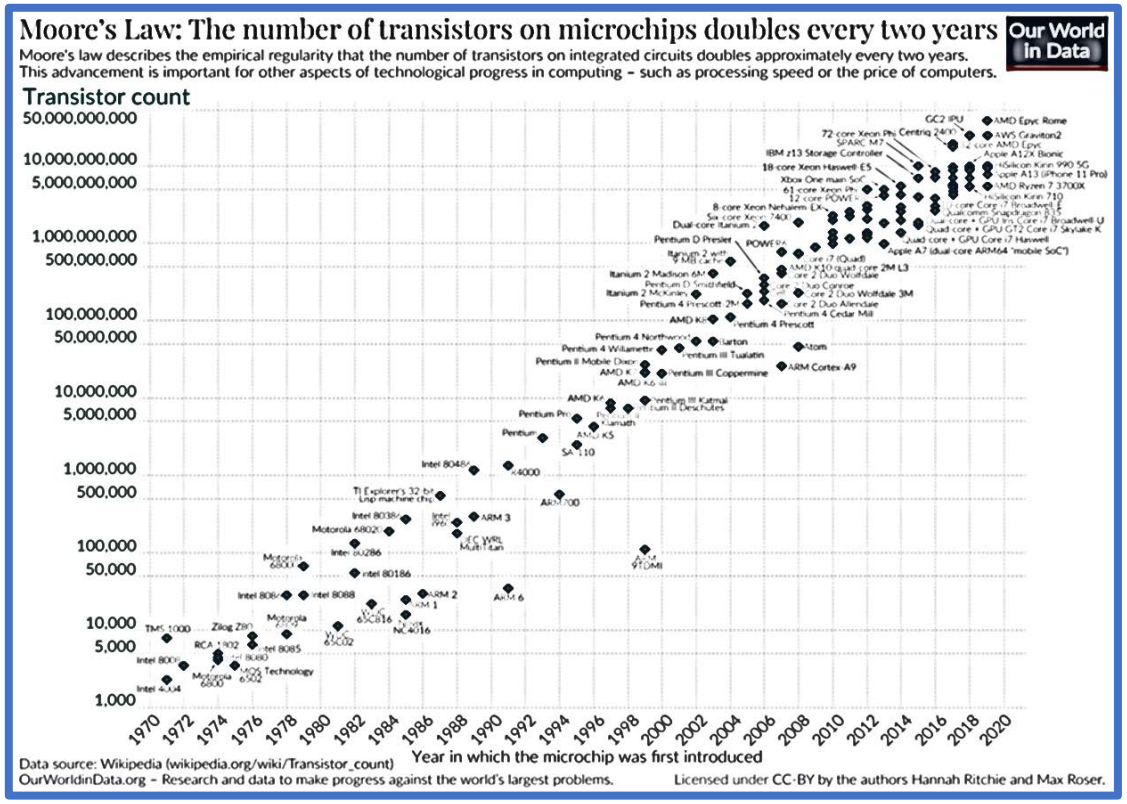
图像识别

图像识别，人脸识别，指纹识别，医学诊断

强化学习

AlphaGo，自动驾驶、机器人

算力提高是人工智能技术发展的原动力



摩尔定律

集成电路上可容纳的晶体管数目，约每隔两年便会增加一倍

$$2^{25} = 33,554,432$$

如果汽车按照这个速度发展，1970年的一万美元的甲壳虫汽车今天只需要1美分，而且时速可以达到5000公里/小时。

工业革命带来了机械力的提升
（突破了人的体力限制）

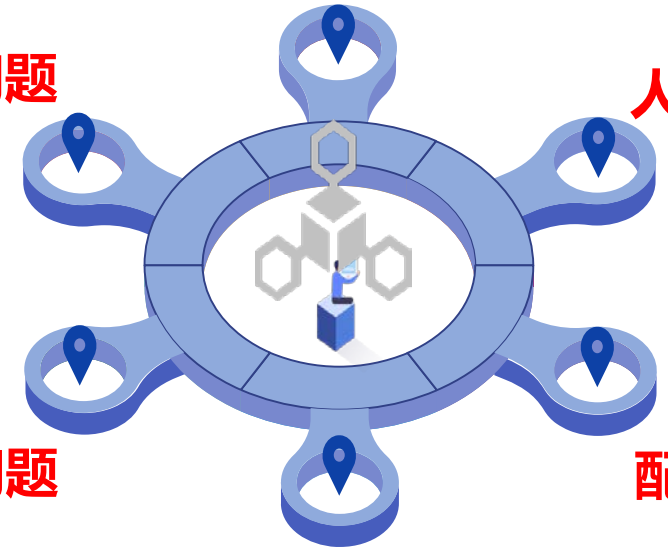
摩尔定律带来了计算力的提升
（突破了人的脑力限制）

供应链建模与优化

物流网络规划

需求预测问题

人员排班问题



生产计划问题

配送路径优化

库存管理问题

$$\max \sum_{i=1}^{n} \sum_{t=1}^{T} [p_{it} R_{i,t-L_i} - h_{it} I_{it}],$$

subject to

$$\sum_{ij \in S} a_{ij} R_{i,t-L_j} \leq \sum_{k \in S} C_{kt}, \quad \text{for all generated sets } S \text{ and periods } t,$$

$$\sum_{\tau=1}^{t} R_{i,t-L_i} - I_{it} + B_{it} = D_{it}, \quad \text{for all } i; \text{for all } t \leq T-1,$$

$$\sum_{\tau=1}^{t} R_{i,\tau-L_i} + B_{iT} = D_{iT}, \quad \text{for all } i,$$

$$\sum_{\tau=1}^{T} R_{i,\tau-L_i} + B_{iT} = D_{iT}, \quad \text{for all } i,$$

$$R_{it}, B_{it}, I_{it} \geq 0.$$

线性规划/混合整数规划模型

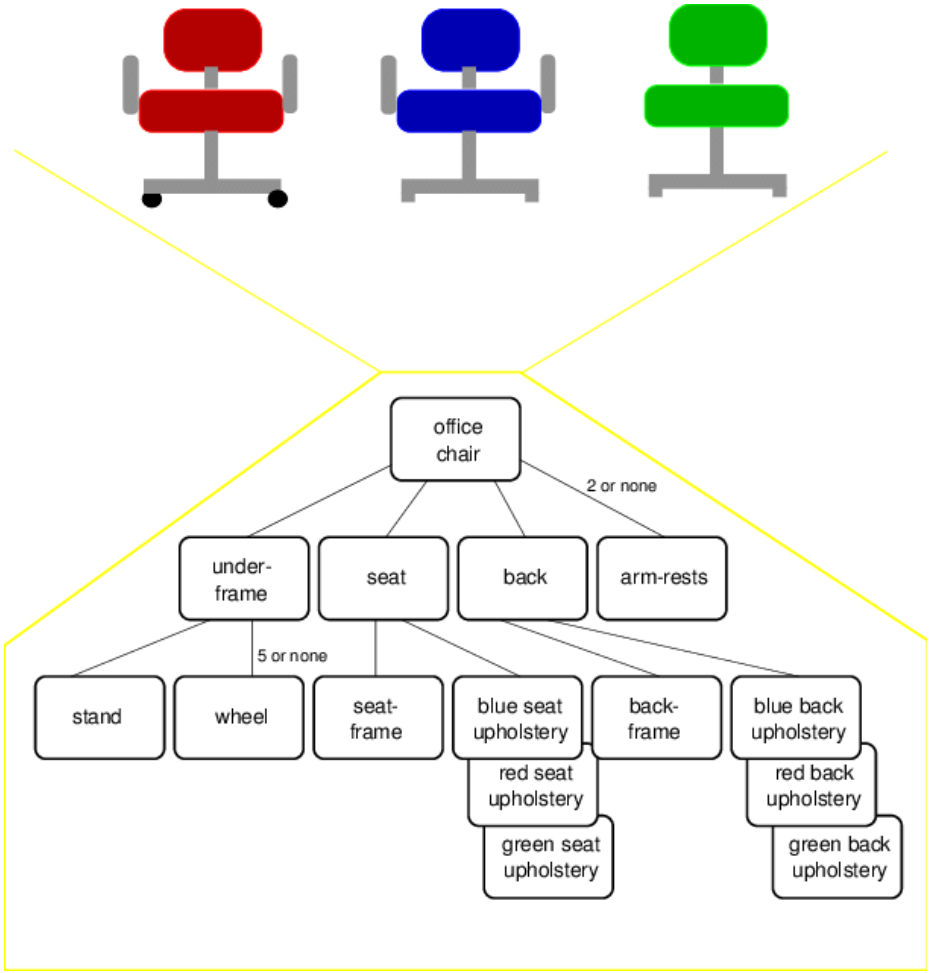
传统的供应链建模和优化方法（例如线性规划、混合整数规划、离散事件仿真等）无法充分利用并行计算的大规模算力，例如最有名的优化问题求解器Gurobi和CPLEX只能支持最多32核。

基于人工智能方法的供应链建模与优化

- 目前已有一些将人工智能技术应用于供应链管理的尝试，例如利用机器学习模型进行需求预测等，主要是模型或算法的直接使用。

- 我们团队致力于将人工智能技术的底层逻辑和方法应用于供应链管理之中，解决当前供应链管理领域遇到问题和挑战。

案例：大规模库存优化

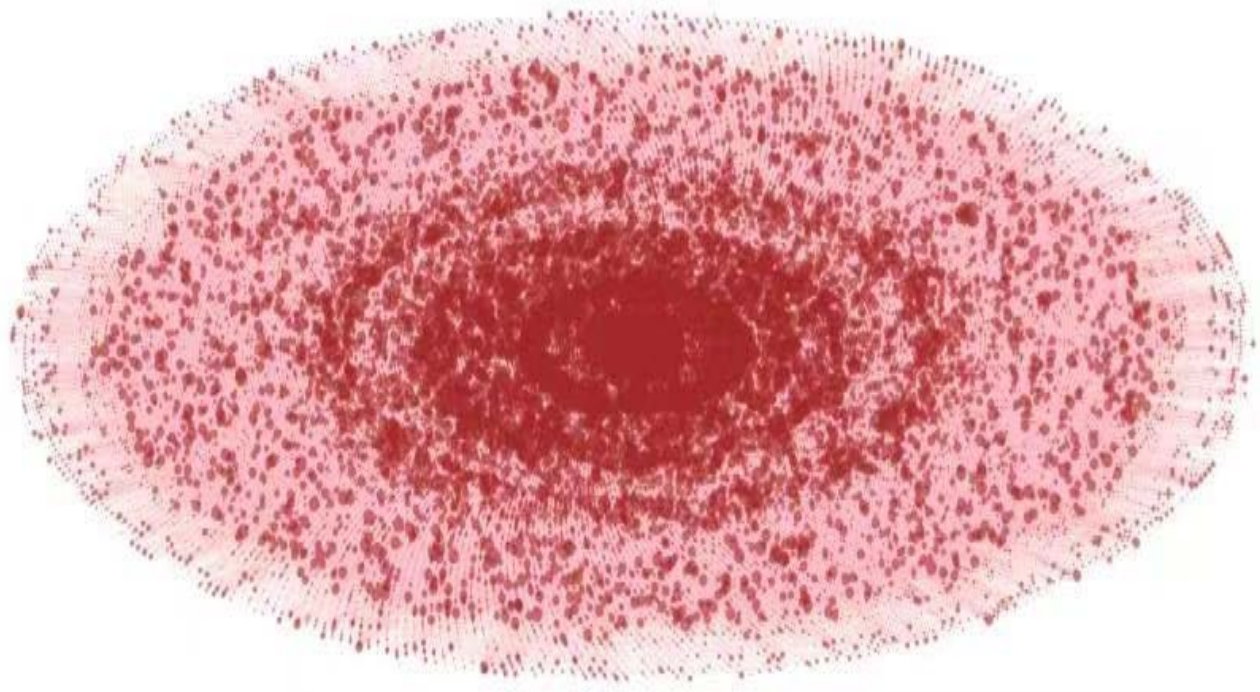


- 物料清单（BOM，bill of materials）用于描述生产最终产品所需的原材料、半成品的构成关系以及所需数量
- 我们的库存优化问题试图确定BOM网络中每个节点的安全库存水平（base-stock level）来最小化成本

大规模库存优化

国内某知名头部制造商BOM

- 节点超过50万个，链接超过400万条；
- 活跃节点超过5万个，链接超过50万条；
- 极为复杂的网络结构；
- 库存价值约为￥200亿。



库存策略优化问题：

- 安全库存应该存放在那些节点上？
- 安全库存量应该如何设定？

这是一个超大规模优化问题，主要的挑战来自于计算能力和计算速度

现有方法和挑战

随机服务模型（Stochastic service models, Clark & Scarf 1960, Rosling 1989, Chen et al. 2014）

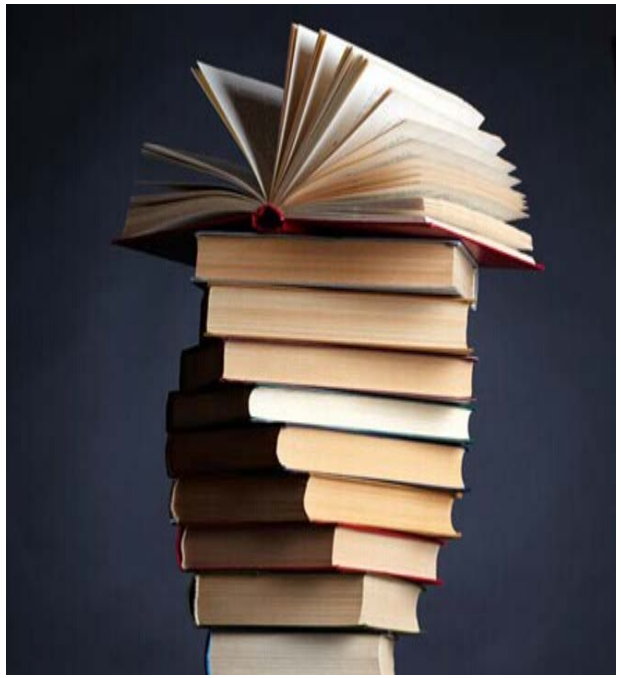
- 针对串行系统/装配系统，无法处理复杂BOM网络结构

保证服务模型（Guaranteed service models，Simpson 1958, Graves & Willems 2000）

- 主要针对生成树结构的网络
- 需要将随机问题转换为确定性问题

仿真方法（Simulation approach，Glasserman & Tayur 1995）

- 允许更一般的网络结构
- 运行时间通常很长



问题定义

假设库存系统根据 base stock policy 运行，周期性（每天）检查需求和库存。

仿真过程：

- 计算inventory positions，产生orders；

$$IP_{t,i} = IP_{t-1,i} + O_{t-1,i} - D^{in}_{t-1,i} - D^{out}_{t-1,i}$$

$$O_{t,i} = -\min\left\{0, IP_{t,i} - D^{out}_{t,i} - D^{in}_{t,i} - S_i\right\}$$

- 接收完成的orders，满足外部需求，更新on-hand inventories 和 backlogs

$$I^0_{t,i} = \max\left\{0, I_{t-1,i} + P_{t,i} - B^{out}_{t-1,i} - D^{out}_{t,i}\right\}$$

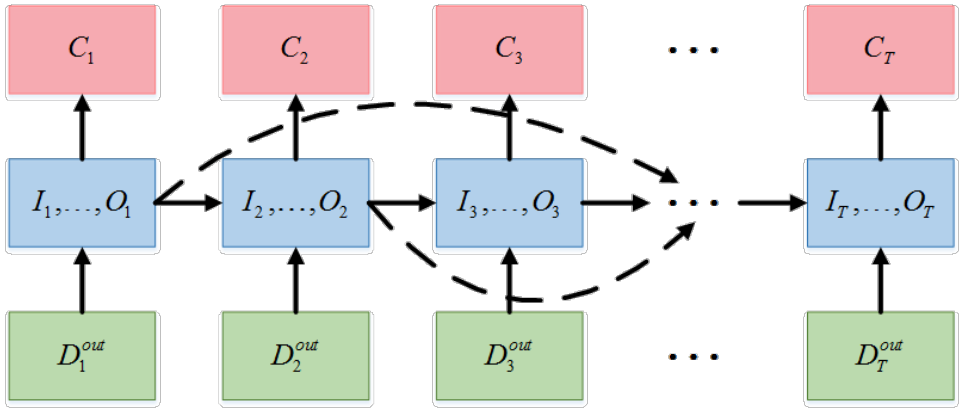
$$B^{out}_{t,i} = -\min\left\{0, I_{t-1,i} + P_{t,i} - B^{out}_{t-1,i} - D^{out}_{t,i}\right\}$$

- 根据orders和可用的原材料决定生产情况

• • •

- 计算成本

$$C_t = \sum_i \left(h_i I_{t,i} + p_i B^{out}_{t,i}\right)$$



变量： 库存位置、库存量

优化目标： 期望总成本最小、安全库存位置稀疏

$$\min_S E\left[\sum_{t=1}^{T} C_t(S)\right]$$

基于仿真的优化方法

仿真优化框架
(Glasserman & Tayur 1995)：

- 建立仿真模型
- 梯度估计（IPA）
- 随机梯度下降（SGD）

传统库存仿真方法的复杂度：

仿真模型	梯度计算
$O(Tn^2)$	$O(Tn^3)$

其中T为仿真的时段数，n为BOM网络节点个数

传统库存仿真方法的耗时（单次采样）：

节点个数	仿真模型	梯度计算
5000	1小时	多个小时

我们的目标是将运行时间降到可以接受的水平，从而至少可以实现50,000个节点的库存优化

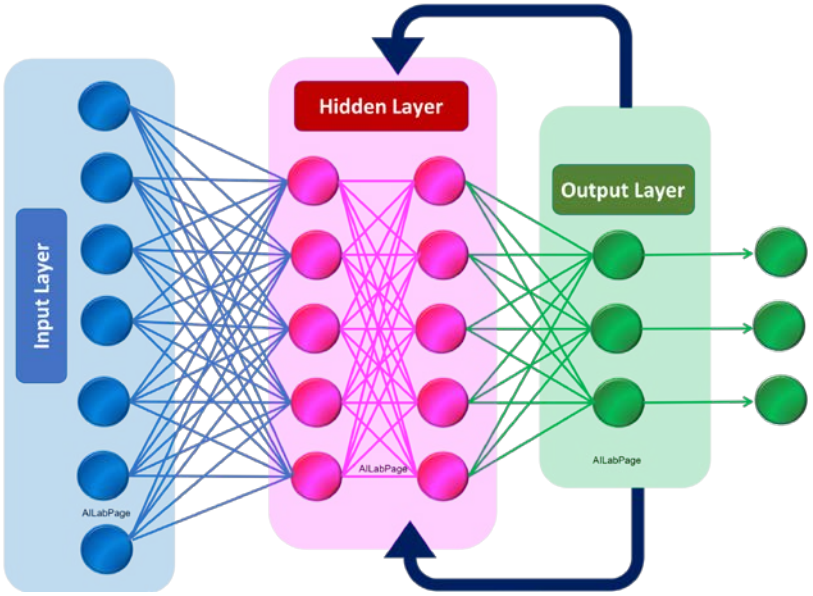
循环神经网络

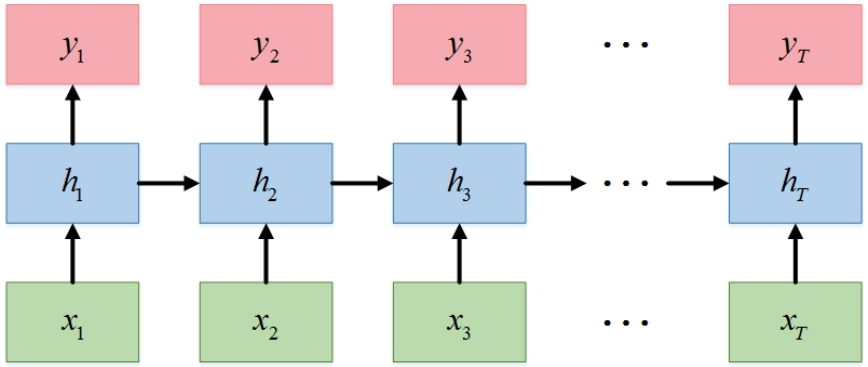
- 循环神经网络是一类所有节点（循环单元）按链式连接的神经网络
- 它在语音识别、语言建模、机器翻译等任务上表现出色
- 目前，循环神经网络模型的参数能达到千万量级
- 更新公式可表示为

$$h_t = f\left(Wh_{t-1} + Ux_t + b\right)$$

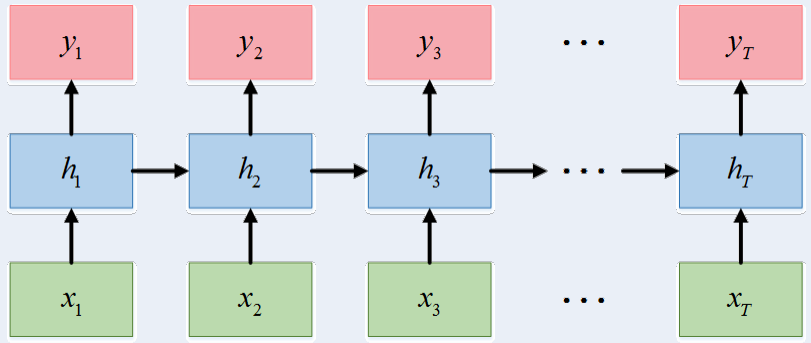
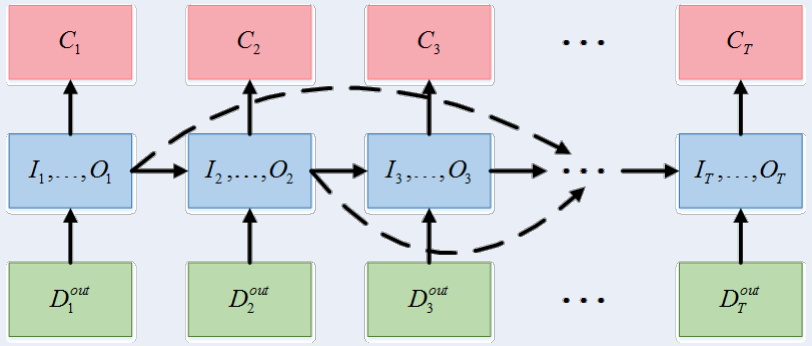
$$y_t = g\left(Vh_t + c\right)$$

- 训练中利用反向传播（BP）和SGD算法计算梯度





循环神经网络训练和库存优化的类比

	RNN训练	库存优化
结构		
更新公式	$h_t = f\left(Wh_{t-1} + Ux_t + b\right)$	$I_{t,i}^0 = \max\left\{0, I_{t-1,i} + P_{t,i} - B_{t-1,i}^{out} - D_{t,i}^{out}\right\}$
目标函数	$\min \frac{1}{N}\sum_{n=1}^{N} L_T^n$ $L_T^n = \sum_{t=1}^{T} L\left(y_t^n, \hat{y}_t^n\right)$	$\min_S \ E\left[\sum_{t=1}^{T} C_t(S)\right]$ $C_t = \sum_i \left(h_i I_{t,i} + p_i B_{t,i}^{out}\right)$
优化变量	神经网络权重	base-stock levels
优化算法	随机梯度下降	随机梯度下降

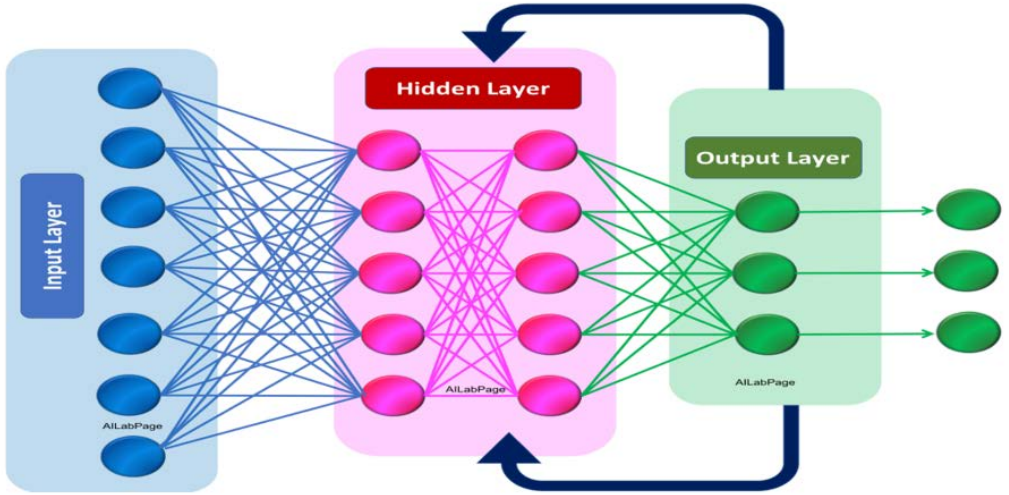
注意：库存优化只是用循环神经网络的计算架构，没有训练任何东西！

循环神经网络训练和库存优化的类比

关键问题：为什么RNN可以解决大规模问题?

对解决大规模库存优化问题有什么启发?

- **张量化：**通过对仿真过程的矩阵化和张量化表示来实现并行计算；
- **梯度计算：**通过构建计算图和利用反向传播算法计算sample-path梯度；
- **L1-正则化：**通过L1-正则化结合随机梯度下降算法来实现优化结果的稀疏性。



Demands/
Lead time

BOM
Network

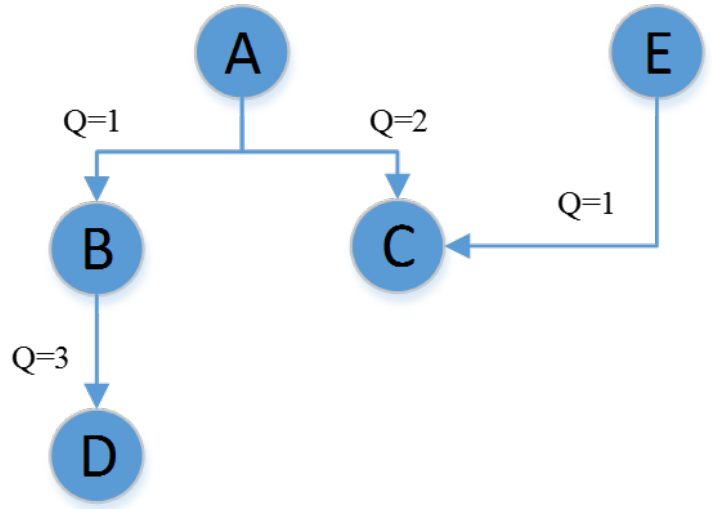
Inventory/
Backlog

张量化：BOM的邻接矩阵

- 利用BOM网络的邻接矩阵，我们可以把整个仿真过程转化为矩阵/向量运算
- 利用TensorFlow等机器学习工具，矩阵运算可以轻松地并行化，甚至可以在GPU上运行

```
os.environ['CUDA_VISIBLE_DEVICES'] = '1,0'
```

- n个节点的BOM网络邻接矩阵为n x n的方阵
- 对于大规模的网络，邻接矩阵的存储将会占用大量内存空间



	A	B	C	D	E
A	0	1	2	0	0
B	0	0	0	3	0
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	1	0	0

张量化：邻接矩阵的稀疏性

- 实际上BOM网络的邻接矩阵通常非常稀疏（例如，50万节点400万条边）
- 可以利用稀疏矩阵技术减少内存消耗
- 一个有n个节点m条边的有向图的密度定义为：

$$\rho = \frac{m}{n(n-1)}$$

- 可以用平均出度<k>来表示网络的稀疏度：

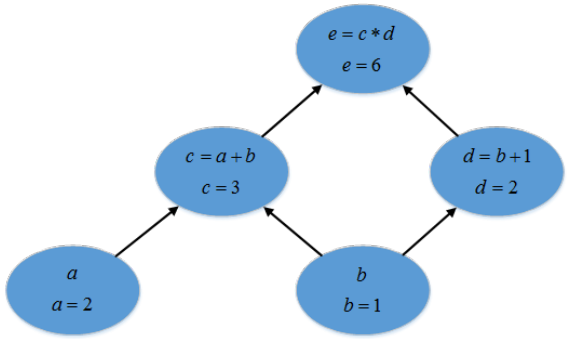
$$\langle k \rangle = \frac{m}{n} = (n-1)\rho \approx n\rho$$

- BOM网络的平均出度代表组成一个下游成品/半成品所需的上游部件的平均个数，通常来讲，它不会随着网络规模的增加而变大，即

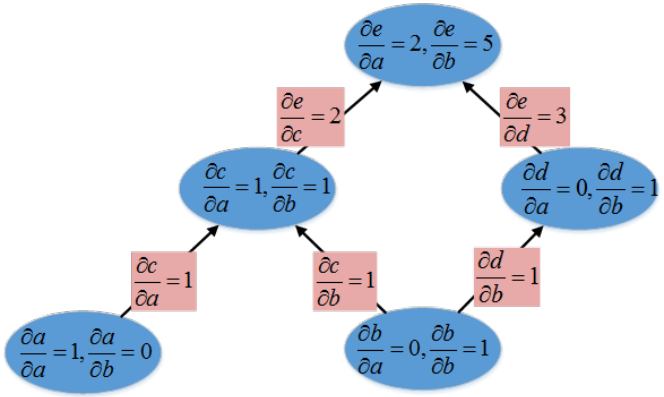
$$n\rho \sim \text{constant}$$

- 基于BOM网络平均出度的性质以及稀疏矩阵技术的使用，仿真及梯度计算的复杂度能够大大降低

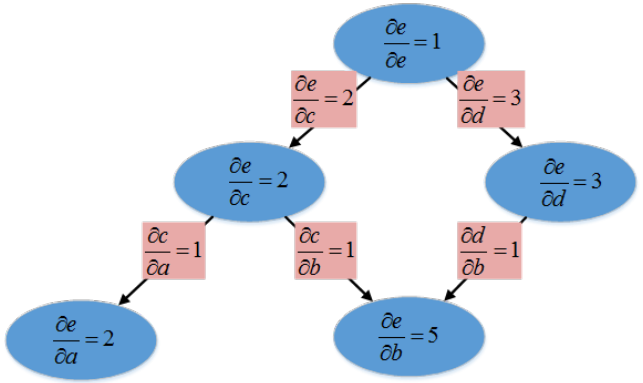
梯度计算：IPA和反向传播



计算图



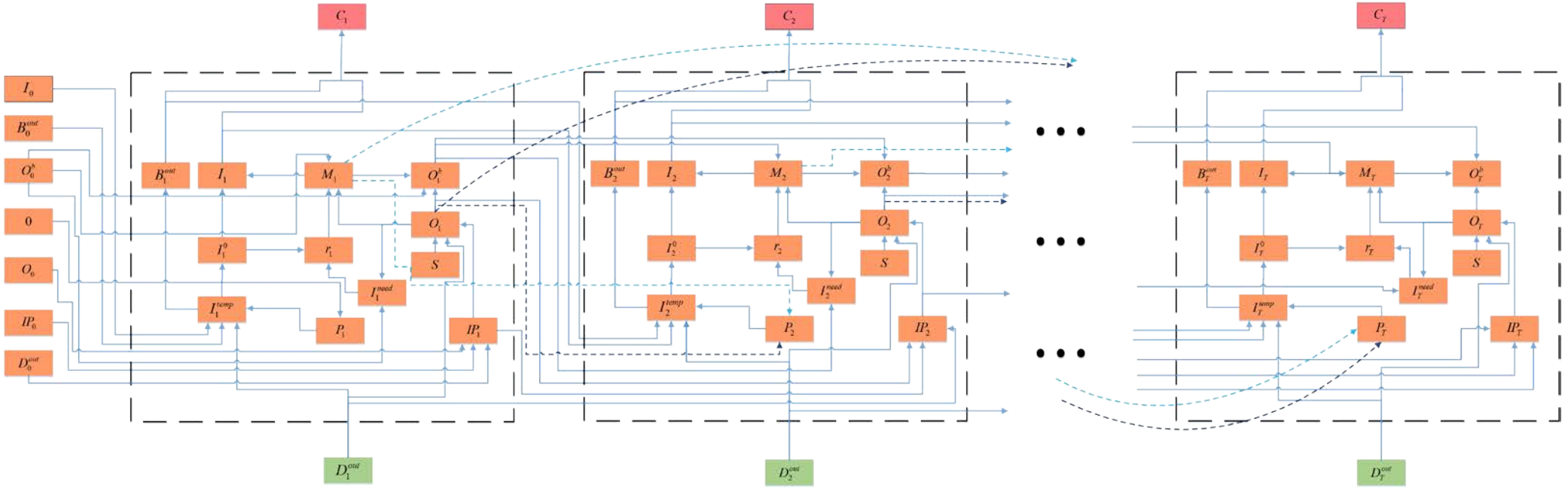
正向模式



反向模式

- IPA和BP产生相同的sample-path梯度
- IPA采用正向计算模式
- BP采用反向计算模式
- 如果利用TensorFlow等工具进行仿真代码编写，则可采用其内置的BP算法自动计算梯度

梯度计算：计算图和反向传播



我们构建了库存仿真过程的计算图，并且基于计算图构建了针对我们问题的
定制化反向传播算法

L1-正则化：库存位置稀疏化

受高维回归领域relaxed Lasso算法的启发，我们提出了一个两阶段的优化算法：

- 首先求解带L1-正则化项的优化问题来选择库存放置的位置：

$$S_1^* = \arg\min\left\{E\left[\sum_{t=1}^{T} C_t(S)\right] + \lambda\|S\|_1\right\}$$

- 接着基于第一步的结果，固定S向量中特定的位置为0，求解新的优化问题，进一步优化库存值：

$$\min E\left[\sum_{t=1}^{T} C_t\left(\tilde{S}\right)\right] \quad where \ \tilde{S} = S \square 1_{\{S_1^*>0\}}$$

- 在第一步中，采用随机版的快速迭代阈值收缩算法（FISTA, fast iterative shrinkage-thresholding algorithm）进行求解
- 第二步采用随机梯度下降（SGD）算法进行求解

数值实验结果：仿真

运行一次仿真所需时间：

~8600倍 ⟶

节点个数	传统方法	我们算法-稠密	我们算法-稀疏
1000	2.21min	0.28s	0.19s
5000	56.00min	6.28s	0.73s
10000	**212.18min**	25.12s	**1.48s**
50000	-	632.21s	6.92s
100000	-	2535.56s	13.90s
500000	-	/	85.70s

时间复杂度

传统方法	我们算法
$O(Tn^2)$	$O(Tn)$

节点个数	TensorFlow-CPU-dense	TensorFlow-GPU-dense	TensorFlow-CPU-sparse	TensorFlow-GPU-sparse
1000	1.66s	4.60s	**1.69s**	2.98s
5000	13.08s	12.76s	**7.08s**	11.61s
10000	40.19s	24.45s	**14.38s**	23.14s
50000	735.64s	129.72s	**71.30s**	112.70s
100000	2912.50s	/	**145.31s**	231.39s
500000	/	/	**775.07s**	1144.78s

硬件条件：
2 Intel Xeon Gold 6248R CPUs（每个24核），
1 NVIDIA GeForce RTX 3090 GPU（10496核），
256GB RAM

数值实验结果：梯度计算

计算一次梯度所需时间：（包括仿真运行时间）

~2300倍

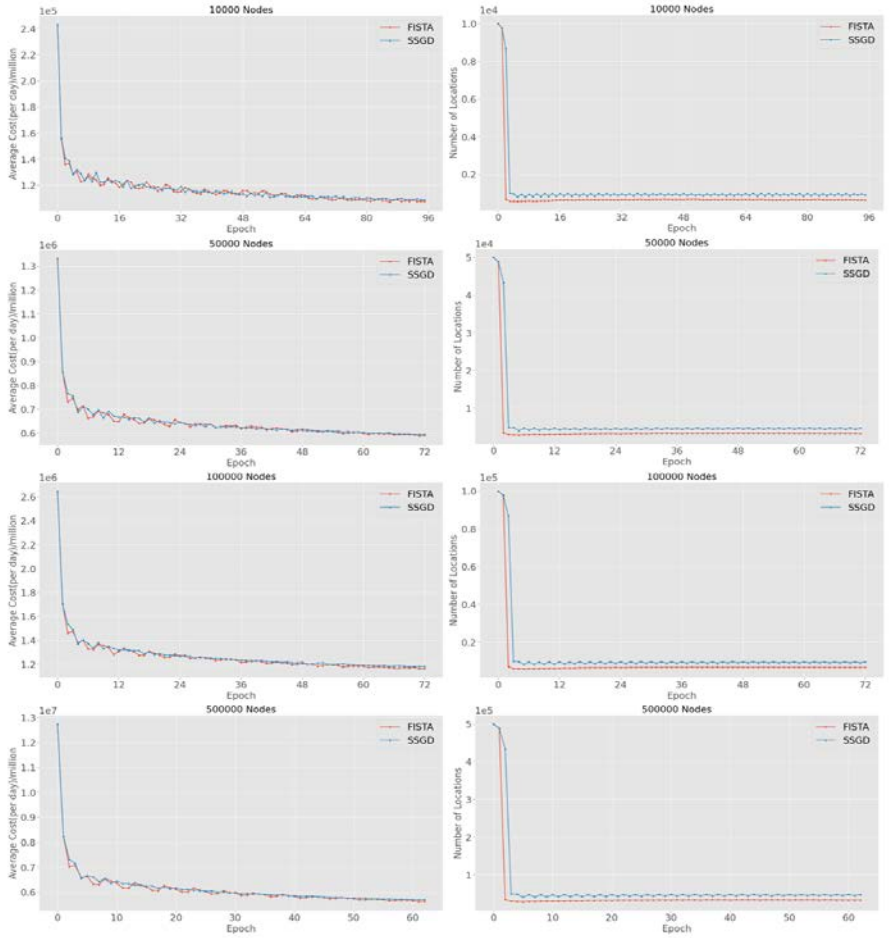
节点个数	IPA-dense	IPA-sparse	BP-dense	BP-sparse
1000	0.55min	0.34min	1.00s	0.81s
5000	29.36min	3.01min	14.36s	2.95s
10000	**219.73min**	11.55min	53.35s	**5.68s**
50000	-	654.67min	1277.84s	26.88s
100000	-	/	5152.54s	54.97s
500000	/	/	/	305.64s

节点个数	TensorFlow-CPU-dense	TensorFlow-GPU-dense	TensorFlow-CPU-sparse	TensorFlow-GPU-sparse
1000	9.79s	12.33s	**10.51s**	13.37s
5000	59.94s	54.59s	**48.06s**	55.02s
10000	152.76s	108.37s	**100.99s**	110.11s
50000	2453.98s	/	**609.78s**	547.05s
100000	11112.54s	/	**1874.00s**	/
500000	/	/	**20337.55s**	/

时间复杂度

算法	复杂度
IPA-dense	$O(Tn^3)$
IPA-sparse	$O(Tn^2)$
BP-dense	$O(Tn^2)$
BP-sparse	$O(Tn)$

数值实验结果：库存优化



平均成本　　库存位置

Stage1 优化效果

Stage2 vs. Stage 1 优化效果

节点数	10000	50000	100000	500000
成本降幅	3.38%	2.07%	3.32%	2.17%

运行时间

节点数	10000	50000	100000	500000
Stage1	25.14min	**92.40min**	198.10min	**19.36hr**
Stage2	14.37min	**26.97min**	59.41min	**4.98hr**
Total	39.51min	**119.37min**	257.51min	**24.34hr**

总 结

- 通过对库存优化问题和RNN类比，使得我们可以采用RNN的底层技术来解决大规模库存优化问题。
- 提出的框架使得我们可以使用TensorFlow等高性能计算工具，从而降低了建模、优化和大规模并行化的障碍。
- 在数值实验中，我们的方法在仿真与梯度计算的运行速度上较传统方法各自提升了数千倍，5万个节点的测试问题可以在2个小时内解决。
- 该方法可以进一步拓展到其它大规模排队网络的建模和优化中（包括交通网络、服务网络等）。

非常感谢您的观看