题目：A

参赛队员：

队员 1：

多级库存模型的到货率与库存水平的关系及优化讨论

摘要

多级库存模型是用于描述供应链中从生产商开始向少数大规模总公司供货，再由总公司向下属分公司供货，再由分公司向下属子公司供货，由此层层传递至终端消费者的供应形式。该模型现多应用于规模较大，自有供应商且公司层次分明的企业，具有分工明确，便于管理，降低成本等特点。在此供货过程中，随着货物流经的仓库数量越多，流经的仓库规模越小，上下级仓库间距越短，在途时间越短，需要考虑的订货提前期越短。现阶段的多级库存管理多存在着降低库存水平与提高到货率之间的均衡问题，即为降低库存成本，企业期望各级公司仓库均处于低库存水平，而同时希望拥有较高的到货率以提高服务水平，故如何平衡两者成为多级库存模型管理的重点问题。

本文在模型建立上，利用供应链各级之间的网状供货关系，通过分析已知的供货量数据得出一定时间段内各级节点间的货物需求与供应情况，并根据现有的供应链网络结构建立多级库存模型并进行优化与求解。在技术上，对两张表格数据进行统计分析，提取出公司等级、供应量等有用的数据信息，来作为建模与分析的依据。读取表格中的数据并进行标注从而将产品流转网状关系参考表中提及的公司建成一张网络，并将产品流转数据表中的数据按公司名称对应存储到该网络图中。

在讨论库存与到货量之间的关系的问题中，采用数据分析、matlab 拟合方法得出库存量I与到货率φ之间的函数关系。利用产品流转网络中的数据，通过统计分析，用已有的供货量数据求出每家公司对应的库存以及针对下级公司的到货率（库存代表每个仓库的平均库存，到货率用每个公司节点的供货量和下级需求的比值表示）。通过 matlab 曲线拟合得出每种产品库存与到货率之间的函数关系。

在 matlab 曲线拟合的多个方案中，分析认为指数函数最切合实际数据间的变化关系，具体形式为：$\varphi(\mathrm{I}) = \mathrm{a}\varphi^{b}$，例如，第四种货物为$\varphi_4(I) = 0.3585I^{0.1012}$ 。

在固定到货率为 90%优化库存量的问题中，根据需求确定的允许缺货的多级库存模型，参考 EOQ 模型定义出库存总和以及级库存的概念，绘制库存-时间图分析图，得出满足到货率 90%的最小库存量总和的约束条件与计算方法，通过 Lingo 计算处理，得到最终结果。

定义库存总和=$\sum$平均库存=$\sum\frac{1}{2}\mathrm{q}\varphi$，即得相关的目标函数为 min z=45%×$q_1$，设定相关约束条件，用 lingo 求解得出最优结果。

在提高到货率至 95%求解最优库存的问题中，结合问题 1 中得出的库存与到货率之间的关系以及问题 2 中与到货率相关的参数，绘图分析得出各参数随到货率改变而变化的函数关系式，后将 90%到货率改为 95%，即可得出相关参数值，代回上一问题公式中，即可得此条件下的最优库存。

关键词：多级库存模型、整体库存、matlab 曲线拟合、lingo 模型求解

1 问题重述

(同济大学校内数模竞赛 2016 年 A 题) 某行业货物供应商（Supplier）通过各公司（Fac），进而向下级子公司（Fac）直至零售商发行某种专业商品，货物的流通过程如下图。一般地，某个发货商有可能同时在其它订单中也作为收货商，所以该图只是显示了某批货物可能的运输销售流程，但不足以表示发货商与收获商的上下级关系，通常它们会形成一个网状结构，如附件。 另一附件数据集“产品流转数据”是一份销售链行为记录表。数据集的每条观测记录了一次订单情况，即某个上级发货商到下级收货商的某货物批次的情况，包括某上级供应商的某批次产品在某天流通到某个下级收货商的具体信息。在平日里，各公司都有一个初始库存，假设公司的库存量一旦小于某个 值就会立即向其某个上级下订单补货，订单量为常数 。而上级供应商要向多个下级供货，因此下级发来的订单请求未必能得到满足，记下级收货商实际收到的货量占其需求量的百分比的值为到货率。目前该商品较为紧俏，末端收货商（实际使用部门）需求旺盛，到货率也仅有 90%。 该行业供应商关心如下问题，邀请你们小组在尚未提高生产能力之前提升到货率，降低流通库存。

（1）库存与到货率之间究竟有什么关系？

（2）求若要满足目前到货率 90%不变，并且使所有分销商的库存量总和最小，和的值应该为多少？库存总和需要自己定义。

（3）若生产能力提高，估算能使末端收货商的到货率提高至 95%，请重新估算供应商的最优库存。

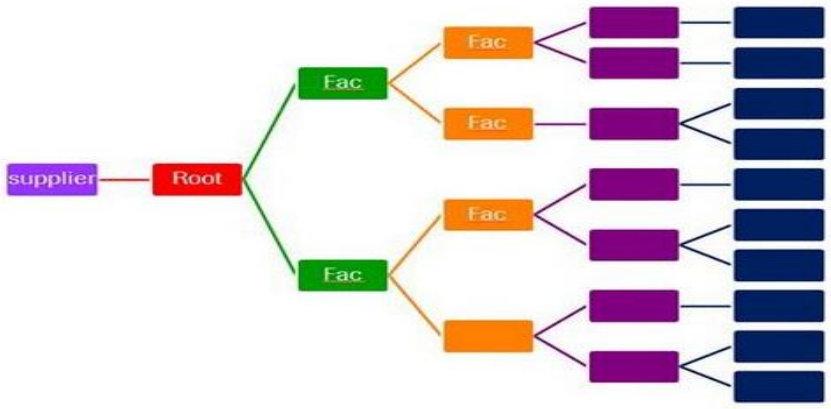


图 1.1 产品流转网状关系参考图

2 模型假设

1. 多级供应链节点级别设为 k，k=1,2,…,7，物流方向为(k+1)级到 k 级节点，不存在逆向流动；

2. 已知的产品流转数据表的供应量可以代表企业一定时间段内的全部供货状况；

3. 固定的两节点之间的最大货物供应数量等于该下游节点的恒定需求量；

4. 某一节点在一段时间内的收到上游节点供应量的总和等于该节点的最大库存量；

5. 系统中同级节点流向下级节点的路程量在一定的区域内使用均值表示（如同省范围内货物距离相等取均值）；

6. 系统采用信息集中型库存控制策略，即各节点并不自己决定订货，由核心成员或专门成立的库存调度中心做出决策，且各级节点均了解最终需求的信息；

7. 客户需求以确定的速度 D（单位时间的需求数量）在节点 1 连续发生；

8. 不计 1~k 级节点之间的相关的订货费用；

9. 库存控制策略为（s，Q），同一级的各节点具有相同的订货提前期，即保有相同的安全库存；

10. 各级节点之间的运输时间t_k为定值；

11. 同级的各节点仓库具有相同的规模，故具有相同的订货点r_k和订货量q_k。

3 符号说明

符号	假设
t_2	一个周期内，库存出现缺货状况的总时长
D	一个周期内，需求方的总需求量
φ	到货率
d	外界对系统的需求速度，即单位时间得需求数量，在节点 1 处连续均匀发生
T_k	节点 k 的订货周期
$I_k'(t)$	所有 k 级节点在 t 时刻的点库存量
$I_k(t)$	所有 k 级节点在 t 时刻的级库存量（相关定义见下文注释）
q_k'	所有 k 级节点的总点订货批量
q_k	所有 k 级节点的总级订货批量
r_k	所有 k 级节点的总级订货点
r_k'	所有 k 级节点的总点订货点
t_k	批产品有 k 级节点至下级节点的在途时间
n_k	k 级节点的个数
m	原始库存
q	最大库存量，即订货批量
q'	一个周期内，最大缺货量
d	下级需求方的需求速度
t_1	一个周期内，从上一次补货后，库存未出现缺货状况的总时长
qk	表示 k 级节点处一个仓库的订货批量

rk	表示 k 级节点一个仓库的订货点

注：定义某节点的级为包括节点本身和下游所有节点，某节点的级库存是指该节点现有点库存和转移到或正在转移给其后续节点的所有库存之和。

由级库存的定义有：$I_k(t) = \sum_{k=1}^{7} I'_k(t)$

4 模型建立与求解

4.1 第一问：求库存与到货率的关系

4.1.1 各参数定义与参数之间的关系

供货量：供应商实际发给分销商的货物量；

需求量：即订单数量，为分销商向供应商申请的进购量；

下级总计需求量 = 请求数*每单需求量；

到货率：到货率=实际到货量（供货量）÷所有下级总计需求量；

库存：暂时闲置将要销售的货物量，令此处库存为平均库存，平均库存为期初与期末货量的平均值，即平均库存=原始库存+（收获量-发货量）/2。

4.1.2 如何从所给表格中获取所需数据

供货量：统计表格《产品流转数据》中某个节点某种货物的供货量之和；

收货量：统计表格《产品流转数据》中某个节点某种货物的收货量之和；

需求量：由条件知，每个节点每种产品的需求量为一常数，要获得某一节点的某种 物品需求量，统计该种物品的每单所收值的最大值，即为该节点该产品的需求量。

请求数：统计该节的每个子节点对于该父节点的请求数（区分货物种类）

库存：库存是一个动态变化的值，一个节点的库存必然与它的收货量密切相关，所以在下面的步骤中暂且用节点的收货量来代替库存，参与关系的推导。

4.1.3 数据点的选择

由上面获得数据的方法，可以获得每一个节点的供货量、收货量、需求量和请求数，但由于不同节点下级对其的总需求量存在巨大差异，经由 matlab 统计，我们根据需求量的密集程度，选取对于每种货物下级需求量在 1000——5000 之间的点作为有效数据。

通过公式

总需求量=每次需求量*次数

货率=收货量/总需求量（下家）

可计算出收货量与到货率，记收获量为变量 x，到货率为变量 y，将每组数据作为平面直角坐标系的坐标值，可得到 5 种商品的大量离散

4.1.4 选取拟合函数类型

根据题意与生活经验分析，库存与到货率应为正比例函数关系，即到货率随库存的增大而增大并逐渐趋于一个稳定值，且此值必小于 1，根据初步筛选，选出 4 种大体呈递增状态的拟合函数图像，以第四种商品为例，分析如下

（1）一次函数

拟合图形

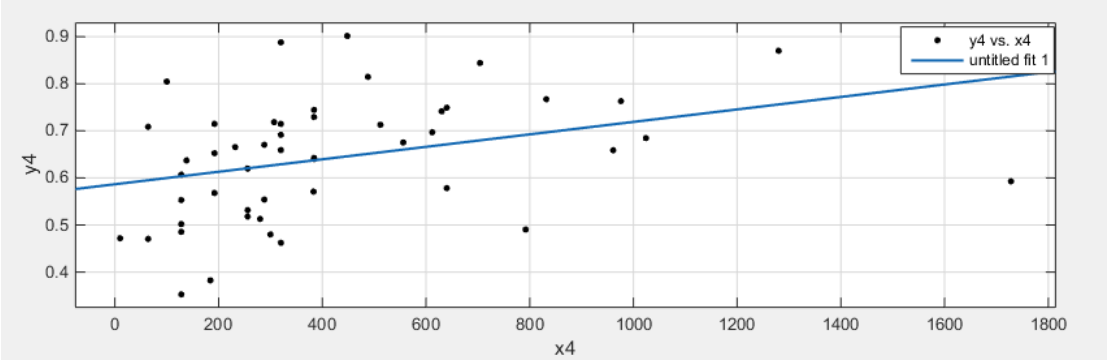


图 4.1.4.1

函数表达式

f(x) = p1*x + p2

参数值

p1 = 0.0001324　(2.786e-05, 0.000237)

p2 = 0.5865　(0.5303, 0.6427)

由于一次函数只是单纯增大并未趋于一个稳定值，所以不符合拟合函数要求。

（2）指数函数

拟合图形

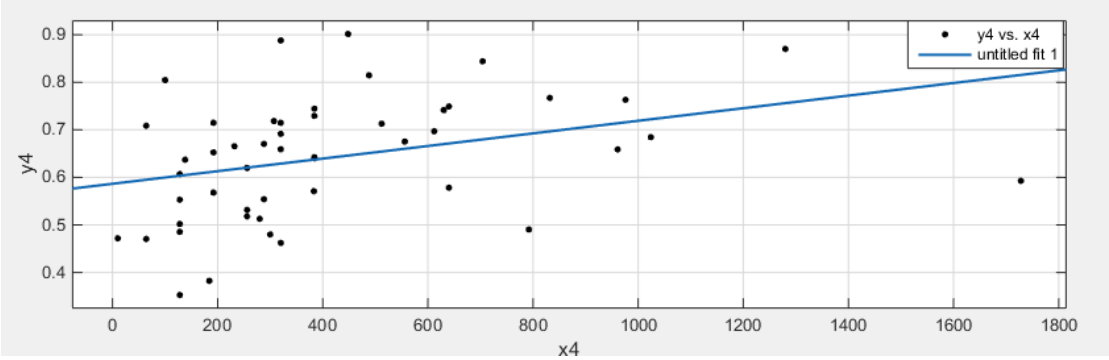


图 4.1.4.2

函数表达式

f(x) = a*exp(b*x)

参数值

a = 0.595　(0.5428, 0.6472)

b = 0.0001776　(3.024e-05, 0.000325)

由于指数函数只是单纯增大并未趋于一个稳定值，所以不符合拟合函

数要求。

（3）线性拟合

拟合图形

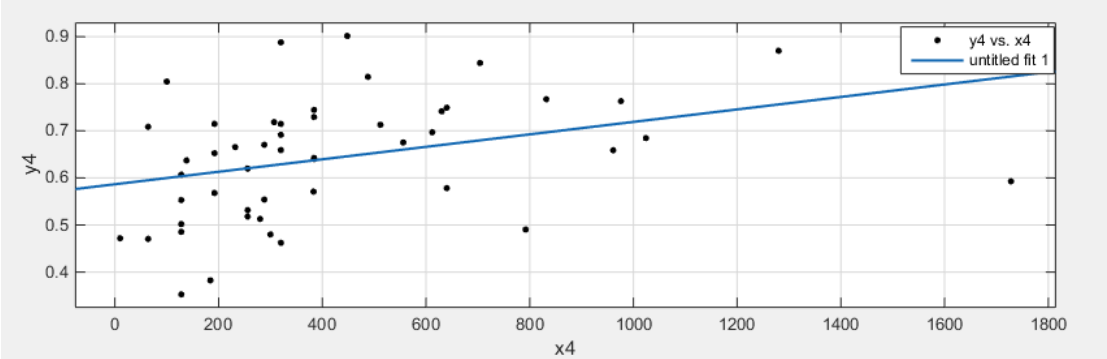


图 4.1.4.3

函数表达式

f(x) = a*(sin(x-pi)) + b*((x-10)^2) + c

参数值

a = 0.009932　(-0.04552, 0.06538)

b = 5.359e-08　(-2.015e-08, 1.273e-07)

c = 0.6267　(0.5843, 0.6691)

由于线性拟合虽大体呈上升状，而局部呈波动状，所以不符合拟合函数要求。

（4）幂函数

拟合图形

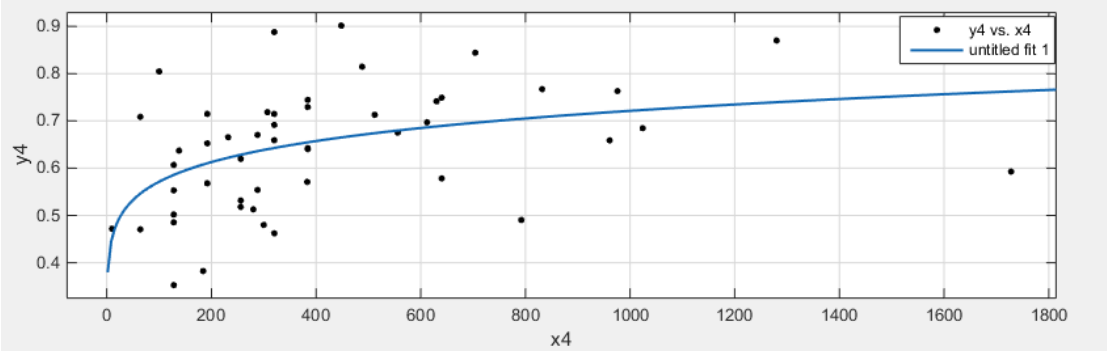


图 4.1.4.4

函数表达式

f(x) = a*x^b

参数值

a = 0.3585　(0.2227, 0.4944)

b = 0.1012　(0.03722, 0.1651)

符合到货率随库存的增大而增大并逐渐趋于一个稳定值，且此值必小于 1 的拟合条件，故库存与到货率函数形式为幂函数。

4.1.5 获得拟合结果

1. 第一种商品：

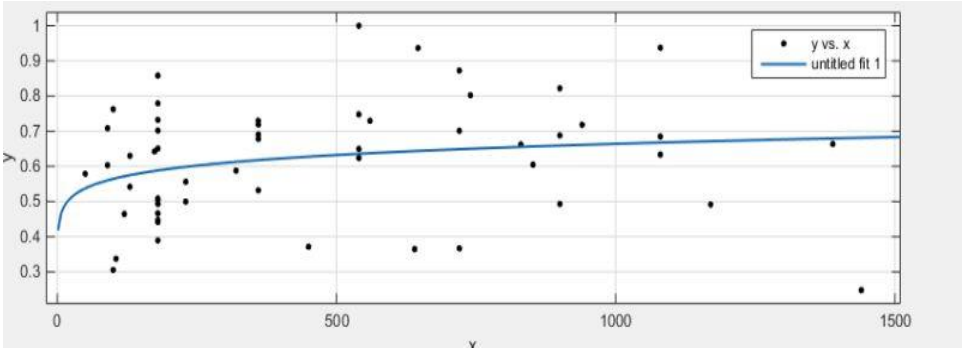


图 4.1.5.1

函数模型： f(x) = a*x^b

系数(95% 的置信区间):

a = 0.4078 (0.202, 0.6136)

b = 0.07059 (-0.01379, 0.155)

2. 第二种商品：

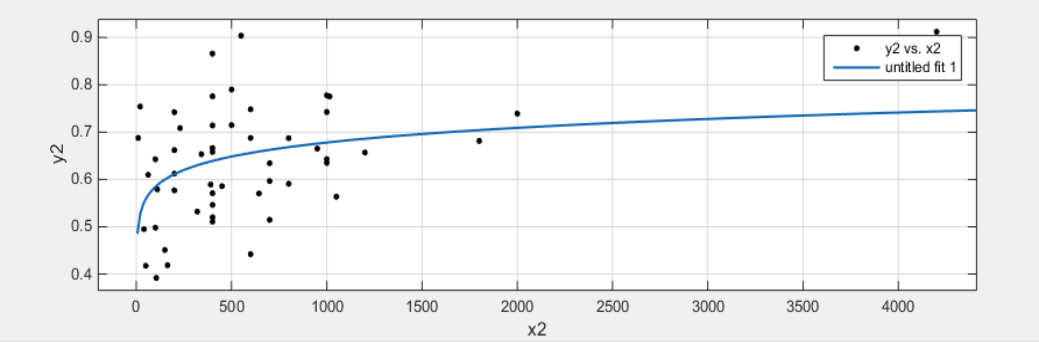


图 4.1.5.2

函数模型：

f(x) = a*x^b

系数(95% 的置信区间):

a =0.4342 (0.3108, 0.5576)

b = 0.06453 (0.01823, 0.1108)

3. 第三种商品：

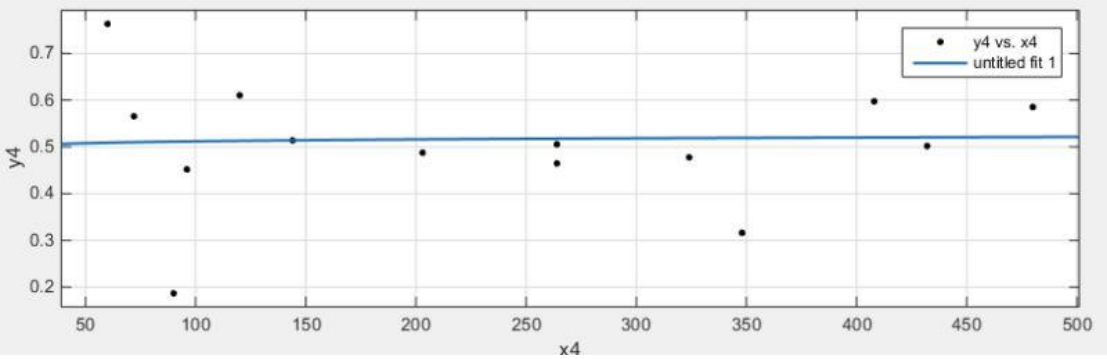


图 4.1.5.3

函数模型：

f(x) = a*x^b

系数(95% 的置信区间):

a = 0.4856 (-0.1105, 1.082)

b = 0.0115 (-0.2172, 0.2402)

4. 第四种商品：

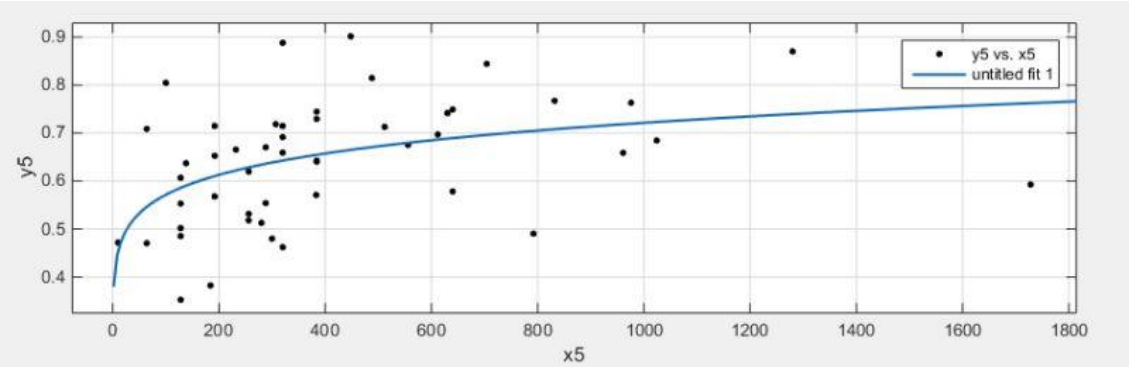


图 4.1.5.4

函数模型：

f(x) = a*x^b

系数(95% 的置信区间):

a =0.3585 (0.2227, 0.4944)

b =0.1012 (0.03722, 0.1651)

5. 第五种商品：

第五种产品数据较为特殊，产品的发送和接收经过数据处理发现主要集中几家公司，可用数据较少，无法拟合出对应关系。

4.1.6 计算与结论

在上图中，x 为某一公司的收货总量，y 为该公司对下级的供货率，若要得到库存和供货率的关系，仍需对 x 做进一步处理。

设库存为 q，原始库存为 m，总需求量为 b（单次需求量*次数），x 为收货总量，y 为供货率

规定 q=m+(x-by)/2；

带入 5.1.2 中各商品拟合函数，得

q=m+ $\frac{1}{2}(\frac{a}{y})^b - by$;

其中，m 为初始库存，为常数，q 为库存，$\boldsymbol{y}$为到货率，a,b 为参数，

前四种商品的函数参数如下（第五组数据较少为求解）：

$$\begin{cases} a1 = 0.4078 \\ b1 = 0.07059 \end{cases}, \begin{cases} a2 = 0.4342 \\ b2 = 0.06453 \end{cases}, \begin{cases} a3 = 0.4856 \\ b3 = 0.0115 \end{cases}, \begin{cases} a4 = 0.3585 \\ b4 = 0.1012 \end{cases};$$

4.2 第二问：优化库存量总和

4.2.1 库存总和的定义

根据图 1.1.2 以及图 1.1.3 上周期T_k内假定需求量单位时间恒定，库存未出现缺货状况的时间与总时间的比值即等于系统的到货率，详细过程如下：

$$q = t_1 \times d$$

$$q' = d \times t_2$$

$$D = q + q' = (t_1 + t_2) \times d$$

$$\varphi = \frac{q}{D} \times 100\% = \frac{t_1}{t_1 + t_2} \times 100\%$$

其中：

q—最大库存量，即订货批量

q'—一个周期内，最大缺货量

d—下级需求方的需求速度

t_1—一个周期内，从上一次补货后，库存未出现缺货状况的总时长

t_2—一个周期内，库存出现缺货状况的总时长

D—一个周期内，需求方的总需求量

φ—到货率

可将库存总和表示为各仓库平均库存之和，平均库存=$\frac{1}{2}q\times\frac{t_1}{t_1+t_2}=\frac{1}{2}q\varphi$

故，库存总和=$\sum$平均库存=$\sum\frac{1}{2}q\varphi$.

4.2.2 符号及变量说明

首先对模型中所用到的符号进行说明：

d—外界对系统的需求速度，即单位时间得需求数量，在节点 1 处连续均匀发生

T_k—节点 k 的订货周期

$I'_k(t)$—所有 k 级节点在 t 时刻的点库存量

$I_k(t)$—所有 k 级节点在 t 时刻的级库存量（相关定义见下文注释）

q'_k—所有 k 级节点的总点订货批量

q_k—所有 k 级节点的总级订货批量

r'_k—所有 k 级节点的总点订货点

r_k—所有 k 级节点的总级订货点

t_k—一批产品有 k 级节点至下级节点的在途时间

n_k—k 级节点的个数

注：定义某节点的级为包括节点本身和下游所有节点，某节点的级库存是指该节点现有点库存和转移到或正在转移给其后续节点的所有库存之和。

由级库存的定义有：$I_k(t)=\sum_{k=1}^{7}I'_k(t)$

4.2.3 多级库存模型建立

由假设条件知，由于需求速度为一固定常数 d，则各级节点的级库存水平具有相同的形状，如图 4.2.3.1 及图 4.2.3.2：

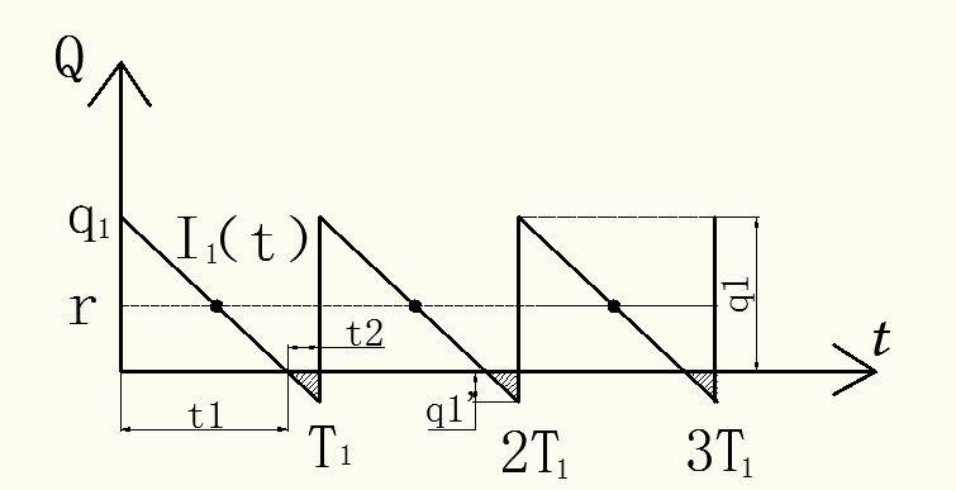


图 4.2.3.1

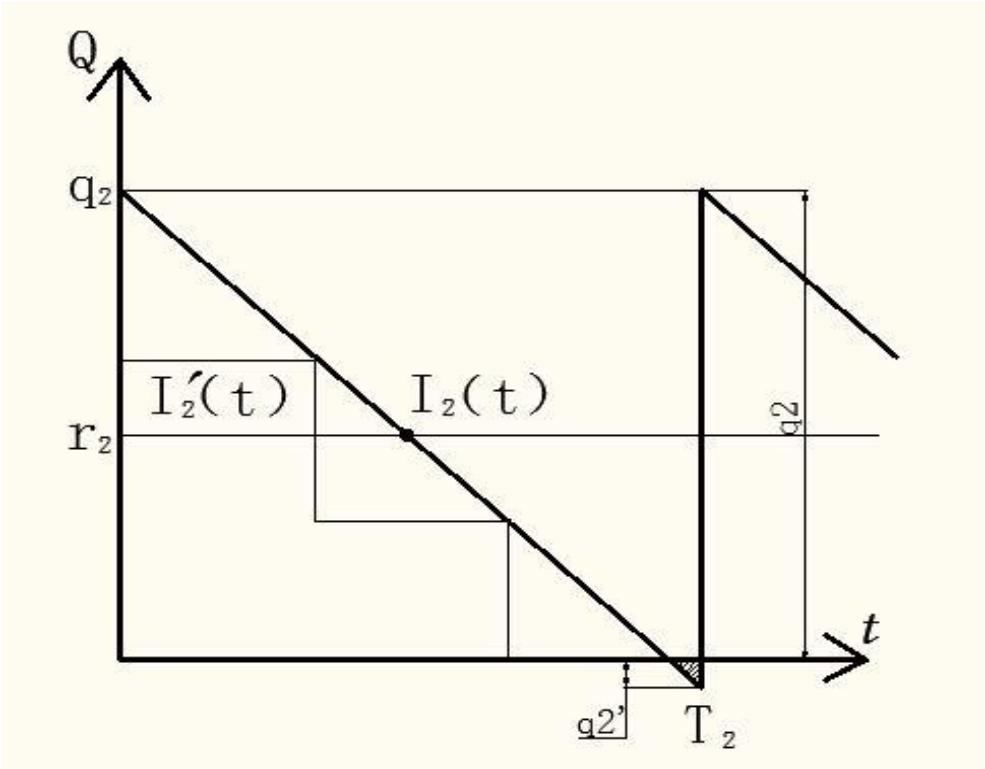


图 4.2.3.2

图 4.2.3.1 描述了一级节点的库存量变化图，其中$I_1(t)$是节点 1 的级库存量，同时也是节点 1 的点库存量。同一周期中，t_1表示从上一次补货后，库存未出现缺货状况的总时长；t_2表示库存出现缺货状况的总时长；q_1表示一级节点的订货批量,同时也是最大库存量；q_1'表示一级节点的一周期的总缺货量。r 表示考虑订货提前期（即货物运输需要消耗一定时间）时的订货点。

图 4.2.3.2 描述了二级节点的库存量变化图，其中$I_2'(t)$表示节点 2 的点库存量，呈阶梯状，$I_2(t)$表示节点 2 的级库存量，它包括节点 1 和节点 2 的点库存量，其图形与节点 1 的级库存相似，从而各级的级库存量变化情况均有相似的图形，但订货周期满足：

$$T_k = m_k \times T_{k-1} \quad (m_k\text{是大于或等于 1 的整数}) \ k=2,3,\cdots,7$$

q_1表示一级节点的订货批量,同时也是最大库存量；q_1'表示一级节点的一周期的总缺货量。r 表示考虑订货提前期（即货物运输需要消耗一定时间）时的订货点。

相关参数的计算公式如下：

根据上文库存综合的定义可得库存总和为：

$$\frac{1}{2} \times q_1 \times \varphi = \frac{1}{2} \times q_1 \times 90\% = 45\% \times q_1$$

一个订货周期内，总需求量为：$T_k \times d$

一个订货周期内，总缺货量为：$(1-\varphi) \times T_k \times d = (1-90\%) \times T_k \times d$

K 级节点的级订货点为：$r_k = t \times d - (1-90\%) \times T_k \times d$

K 级节点的点订货量为：$r_k' = r_k - r_{k-1}$

第七级

故系统的数学模型为：

$$\min z = 45\% \times q_1$$

$$\text{s.t.} \quad r_k = t_k \times d - (1-90\%) \times T_k \times d, \ k=1,2,\cdots,7$$

$$r_k' = r_k - r_{k-1}, \ k=2,3,\cdots,7$$

$$r_1' = r_1$$

$$q_k = 90\% \times T_k \times d, \ k=1,2,\cdots,7$$

$$q_k' = q_k - q_{k-1}, \ k=2,3,\cdots,7$$

$$q_1' = q_1$$

$$T_k = m_k \times T_{k-1} \quad (m_k\text{是大于 1 的整数}) \ k=2,3,\cdots,7$$

$$T_k \geq t_k,\ k=1,2,\cdots,7$$

4.2.4 多级库存模型求解

采用 lingo 对以上模型进行求解，库存总和应该为含有参数 d 的表达式，为方便 lingo 求解，令约束条件中的 d 为 1，lingo 求解后，在结果后乘以 d，即为最终答案。

根据假设条件中，各级节点之间的运输时间t_k为定值，通过改变t_k的取值，来探寻运输时间 t 与库存总和之间的关系。

当t_k=(1,2,3,4,5,6,7)时，相关的 lingo 编程语言见附录 2.5，

求解得：

最小库存综合为 57.15d

T_1=1, T_2=2 T_3=4, T_4=8, T_5=16, T_6=32, T_7=64;

q_1=d, q_2=3d, q_3=7d, q_4=15d, q_5=31d, q_6=63d, q_7=127d;

q_1'=d, q_2'=2d, q_3'=4d, q_4'=8d, q_5'=16d, q_6'=32d, q_7'=64d;

r_1=0.9d, r_2=3.8d, r_3=8.6d, r_4=15.2d, r_5=23.4d, r_6=32.8d, r_7=42.6d;

r_1'=0.9d, r_2'=2.9d, r_3'=4.8d, r_4'=6.6d,r_5'=8.2d,r_6'=9.4d,r_7'=9.8d;

m=(2,2,2,2,2,2,2);

通过分析已知条件中给出的货物流通表格，将第 1 级节点的所有固定时间段内的收货量求和取平均近似代表第 1 级节点的库存总和，即五种货物分别为（387,239,18,111,31），由上述 lingo 结果可知q_1=d，故五种货物需求量 d 分别为（387,239,18,111,31），可得最终结果：

货物 1 的各级仓库的 qk,rk 值：

相关计算公式：$qk=\frac{q_k'}{n_k}$，$rk=\frac{r_k'}{n_k}$

所以：

q1=0.9699,q2=0.2470,q3=0.4495,q4=2.3419,q5=19.4106,q6=387,q7=9.0658；

r1=0.8729,r2=0.3582,r3=0.5394,r4=1.9321,r5=9.9480,r6=113.6813,r7=1.3882；

货物 2 的各级仓库的 qk,rk 值：

相关计算公式：$qk=\frac{q_k'}{n_k}$，$rk=\frac{r_k'}{n_k}$

所以：

q1=0.5990,q2=0.1526,q3=0.2776,q4=1.4463,q5=11.9875,q6=239,q7=5.5988；

r1=0.5391,r2=0.2212,r3=0.3331,r4=1.1932,r5=6.1436,r6=70.

2063,r7=0.8573,

货物 4 的各级仓库的 qk,rk 值：

相关计算公式：$qk=\frac{q_k'}{n_k}$，$rk=\frac{r_k'}{n_k}$

所以：
q1=0.2782,q2=0.0709,q3=0.1289,q4=0.6717,q5=57.2903,q6=111,q7=2.0603；
r1=0.2504,r2=0.1027,r3=0.1547,r4=0.5542,r5=2.8533,r6=32.6063,r7=0.3982；,

由于货物 3 和 5 数据记录缺失或不全面，致使需求量过少，故不进行求解。

4.3 第三问:提高生产率估算最优库存

4.3.1 相关系数分析

q_k, r_k与φ值均相关，采用控制变量法进行分析。

当q_k不变时，$r_k=t_k\times d-(1-\varphi)\times T_k\times d$

当r_k不变时，$q_k=\varphi\times T_k\times d$

4.3.2 数学模型分析

修改后的数学模型为：

$$\min z=47.5\%\times q_1$$

$$\text{s.t.}\quad r_k=t_k\times d-(1-95\%)\times T_k\times d,\ k=1,2,\cdots,7$$

$$r_k'=r_k-r_{k-1},\ k=2,3,\cdots,7$$

$$r_1'=r_1$$

$$q_k=95\%\times T_k\times d,\ k=1,2,\cdots,7$$

$$q_k'=q_k-q_{k-1},\ k=2,3,\cdots,7$$

$$q_1'=q_1$$

$$T_k=m_k\times T_{k-1}\quad (m_k\text{是大于 1 的整数}) k=2,3,\cdots,7$$

$T_k \geq t_k$，k=1,2,…, 7

4.3.3 模型求解

采用 lingo 对以上模型进行求解，库存总和应该为含有参数 d 的表达式，为方便 lingo 求解，令约束条件中的 d 为 1，lingo 求解后，在结果后乘以 d，即为最终答案。

当t_k=(1,2,3,4,5,6,7)时，相关的 lingo 编程语言见附录 2.5，

求解得：

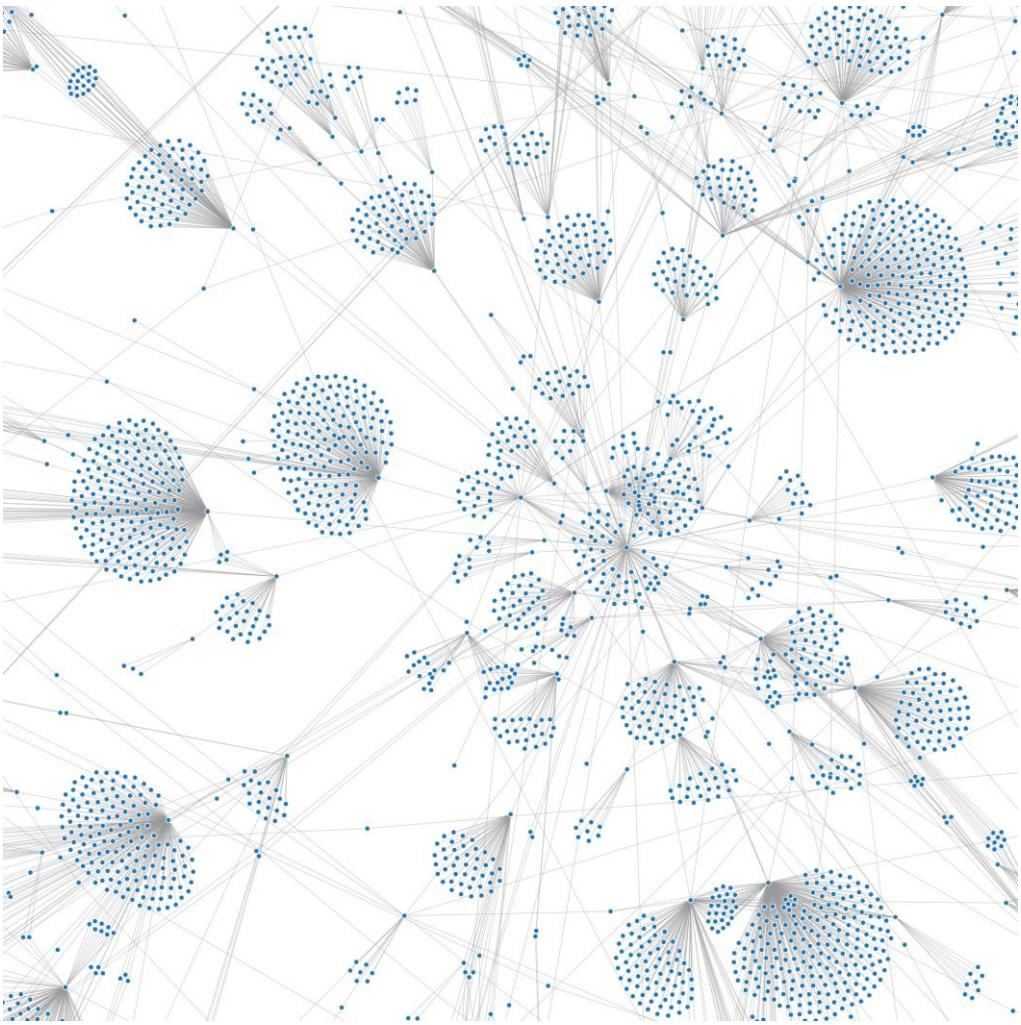
最小库存总和为 60.325d

所以五种货物库存总和分别为（23345.78,14417.68,1085.85,66960.08,1870.08）

5 模型评价

5.1 计算机模型

对于第一问模型，先根据表格《产品流传网状关系图》建立公司之间相互联系的网络，然后再读取《产品流转数据》，获得每家公司发送和收到的每种货物的数量，以及在网络中所处的最大最小层级、公司所处地点、公司上下级有哪些公司以及每种货物各对其发货多少次。基本可以获得求第一问所能获得的全部信息，但是丢失了每单货物的发送时间，无法精确估计终端货物的销售速度，为下面两问的求解带了不便。



5.1.1 物流网络部分截图

5.2 多级库存模型

模型分析

优点：

1.利用多级库存模型以及 EOQ 模型定义出级库存量、级订货批量，使从终端向生产商一端倒推可以看做一个整体，简化了计算过程，将原本的多元高次函数转化为线性函数。

2.可以利用 lingo 软件直观地编程，检验方便，求解快速。

缺点：

1.运输时间的设定简单，不能很好地概括同级间不同节点的运输时间的变化与波动，不能很好地反映现实情况。

2.只设定了时间上的约束条件，由于已知资料未提供各种成本的设定，所以没有设置一次订货成本、库存成本以及缺货费用三者之间平衡使得总成本最低的约束条件，使得模型的现实应用能力不强。

3.题中分五种货物，而五种货物的数据量相差较大，所以分析出的精确度也差别较大，所以无法通过最优库存量总结得出比较适普的表达形式。

6 模型改进

本次建立的模型还过于理想化，只考虑了不同层级公司之间的的距离，而实际情况中低级公司之间的地理距离可能远大于高级公司，细化仓库之间的距离可以大大提高模型精度，贴近实际情况。

而且物流过程中会有大量的成本，如次订货的成本、货物的仓储成本、仓库的存储上限、以及缺货对下级的违约成本，考虑这些成本才可能更好的贴近实际情况。

根据图 4.2.3.1 和图 4.2.3.2，运用 EOQ 模型对每个节点的库存成本进行如下计算：

C_B——节点 l 的缺货费率也即系统的缺货费率

t_k——节点 k 在一个订货周期T_k内未缺货的时间长度

S_k——节点 k 最高库存保有量

由定义知：$D*t_k = S_k$，$D*T_k = Q_k$

节点 K 周期内平均定货费为：$\frac{C_{O,k}}{T_k}$

节点 k 周期内的平均级库存量为：$\frac{D*T_k}{2}$

节点 k 周期内的级存储费为：$\frac{D*t_k*h_k}{2}$ *Tk

节点 l 周期内平均级存储费为：$\frac{D*t_k*h_k}{2}$

节点 l 周期内平均缺货量：$\frac{(T_l - t_l)}{2}*D$

节点 l 周期内平均缺货费为：$\frac{(T_l - t_l)^2}{2}*D*C_B$

节点 k 周期内总运输成本为：$C_{T,k} + D*T_k*C'_{T,k}$

节点 k 周期内平均运输成本为：$\frac{C_{T,k}}{T_k}+D*C_{T,k}^{'}$

7 结论

通过建模分析，我们得出了以下结论：
(1) 库存与到货率的关系为幂函数关系，其函数表达式为：

$$q=m+\frac{1}{2}\left(\frac{a}{y}\right)^{b}-by;$$

其中，m 为初始库存，为常数，q 为库存，y为到货率，a,b 为参数，前四种商品的函数参数如下（第五组数据较少为求解）：

$$\begin{cases}a1=0.4078\\b1=0.07059\end{cases}\begin{cases}a2=0.4342\\b2=0.06453\end{cases}\begin{cases}a3=0.4856\\b3=0.0115\end{cases}\begin{cases}a4=0.3585\\b4=0.1012\end{cases};$$

(2)第二问：

当$\varphi=90\%$时，假定t_k=(1,2,3,4,5,6,7)时，求解得：

最小库存综合为 57.15d

T_1=1, T_2=2 T_3=4, T_4=8, T_5=16, T_6=32, T_7=64;

q_1=d, q_2=3d, q_3=7d, q_4=15d, q_5=31d, q_6=63d, q_7=127d;

q_1'=d, q_2'=2d, q_3'=4d, q_4'=8d, q_5'=16d, q_6'=32d, q_7'=64d;

r_1=0.9d, r_2=3.8d, r_3=8.6d, r_4=15.2d, r_5=23.4d, r_6=32.8d, r_7=42.6d;

r_1'=0.9d, r_2'=2.9d, r_3'=4.8d, r_4'=6.6d,r_5'=8.2d,r_6'=9.4d,r_7'=9.8d;

m=(2,2,2,2,2,2,2);
故通过公式以及分析出的数值得：
货物 1 的各级仓库的 qk,rk 值：

相关计算公式：$qk=\frac{q_k'}{n_k}$，$rk=\frac{r_k'}{n_k}$

所以：
q1=0.9699,q2=0.2470,q3=0.4495,q4=2.3419,q5=19.4106,q6=387,q7=9.0658；
r1=0.8729,r2=0.3582,r3=0.5394,r4=1.9321,r5=9.9480,r6=113.6813,r7=1.3882；

货物 2 的各级仓库的 qk,rk 值：

相关计算公式：$qk=\frac{q_k'}{n_k}$，$rk=\frac{r_k'}{n_k}$

所以：

q1=0.5990,q2=0.1526,q3=0.2776,q4=1.4463,q5=11.9875,q6=239,q7=5.5988；

r1=0.5391,r2=0.2212,r3=0.3331,r4=1.1932,r5=6.1436,r6=70.2063,r7=0.8573,

货物 4 的各级仓库的 qk,rk 值：

相关计算公式：$qk=\frac{q_k'}{n_k}$，$rk=\frac{r_k'}{n_k}$

所以：

q1=0.2782,q2=0.0709,q3=0.1289,q4=0.6717,q5=57.2903,q6=111,q7=2.0603；

r1=0.2504,r2=0.1027,r3=0.1547,r4=0.5542,r5=2.8533,r6=32.6063,r7=0.3982；，

由于货物 3 和 5 数据记录缺失或不全面，致使需求量过少，故不进行求解。

(3)第三问：

修改后的数学模型为：

$$\min z=47.5\% \times q_1$$

$$\text{s.t.}\quad r_k=t_k \times d-(1-95\%) \times T_k \times d,\ k=1,2,\cdots,7$$

$$r_k'=r_k-r_{k-1},\ k=2,3,\cdots,7$$

$$r_1'=r_1$$

$$q_k=95\% \times T_k \times d,\ k=1,2,\cdots,7$$

$$q_k'=q_k-q_{k-1},\ k=2,3,\cdots,7$$

$$q_1'=q_1$$

$$T_k=m_k \times T_{k-1}\quad (m_k\text{是大于 1 的整数}) \ k=2,3,\cdots,7$$

$$T_k \geq t_k,\ k=1,2,\cdots,7$$

求解得：

最小库存总和为 60.325d

所以五种货物库存总和分别为

（23345.78,14417.68,1085.85,66960.08,1870.08）

参考文献

[1]李璧君. EOQ 模型修正及其应用.[D].暨南大学,2008.第 15 页
[2]郭晓霞,闫学锋. 供应链环境下库存管理 EOQ 模型分析及应用[A]. 中国运筹学会不确定系统分会、南京理工大学.第三届不确定系统年会论文集[C].中国运筹学会不确定系统分会、南京理工大学:,2005:7. 第 1 页
[3]李怡娜,徐学军,叶飞. 允许缺货的可控提前期供应链库存优化与协调[J]. 工业工程与管理,2008,No.6801:41-46 第 3 页
[4]来源. 基于供应链分销网络的多级库存优化研究[D].武汉理工大学,2008.第 13~16 页

附录

附录 1：数据分析代码（C#）

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;

namespace ConsoleApplicationMath
{
    class Program
    {

        static void Main(string[] args)
        {
            Graph graph =new Graph("C:\\Users\\xuan\\Documents\\tjjs2016A2.csv");
            Road road = new Road("C:\\Users\\xuan\\Documents\\roadTime.csv");
            Information infor = new
Information("C:\\Users\\xuan\\Documents\\tjjs2016A1.csv", graph);
            road.Output("C:\\Users\\xuan\\Documents\\roadtimedata.csv");
            graph.Output("C:\\Users\\xuan\\Documents\\informa 不排除未知省
份.csv");
            graph.CaluP("C:\\Users\\xuan\\Documents\\re2.csv");
            graph.ProOne("C:\\Users\\xuan\\Documents\\re 未修饰");
            graph.protwo("C:\\Users\\xuan\\Documents\\require");
            graph.prothree("C:\\Users\\xuan\\Documents\\final");
            //graph.Short("C:\\Users\\xuan\\Documents\\short.csv");
            graph.Terminal("C:\\Users\\xuan\\Documents\\terminal.csv");
            graph.CreateJson("C:\\Users\\xuan\\Documents\\node.json");
        }
    }
}
```

Information.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplicationMath
{
    class Information
    {

        Dictionary<string, int> productInt;
        string[,] mapData;
        int length = 0;
        Graph graph;
        int test = 0, test1 = 0;
        int test2 = 0;
        int test3 = 0;
        public Information(string path, Graph g)
        {
            productInt = new Dictionary<string, int>();
            mapData = new string[50000, 19];
            graph = g;
            ReadFile(path);
        }
        public bool ReadFile(string path)
        {
            string strline;
            string[] aryline;
            int num = 0;
            int ta = 0, tb = 0, tc = 0;
            System.IO.StreamReader mysr = new System.IO.StreamReader(path,
System.Text.Encoding.Default);
            strline = mysr.ReadLine();
            while ((strline = mysr.ReadLine()) != null)
            {
                aryline = strline.Split(',');
                test1++;
                if (!graph.compmIndex.ContainsKey(aryline[7])
|| !graph.compmIndex.ContainsKey(aryline[12]))
                {
```

```
                continue;
            }
            //if (aryline[5] == "NULL"||aryline[6]=="NULL") continue;
            if (!productInt.ContainsKey(aryline[0]))
            {
                productInt.Add(aryline[0], num);
                num++;
            }
            int goodType = productInt[aryline[0]];
            int sendCom = graph.compmIndex[aryline[7]];
            int receiveCom = graph.compmIndex[aryline[12]];
            // Convert.
            test2++;
            if (aryline[8] != "未知" && graph.nodes[sendCom].privonce == null)
            {
                graph.nodes[sendCom].privonce = aryline[8]; ;
                //Console.WriteLine(receiveCom + aryline[8]);
                test3++;
            }

            if (aryline[13] != "未知" && graph.nodes[receiveCom].privonce ==
null)
            {
                graph.nodes[receiveCom].privonce = aryline[13];
                //Console.WriteLine(receiveCom + aryline[13]);
                test++;
            }

            if (aryline[11] == "零售终端")
            {
                graph.nodes[sendCom].isFinal[goodType] = true;
                ta++;
            }
            if (aryline[11] == "生产企业")
            {
                graph.nodes[sendCom].isCreate[goodType] = true;
                tb++;
            }
            if (aryline[16] == "终端 A" || aryline[16] == "终端 B" || aryline[16]
== "终端 C")
            {
                graph.nodes[receiveCom].isCreate[goodType] = true;
                tc++;
            }
```

```
                graph.nodes[sendCom].sendGoods[goodType] +=
int.Parse(aryline[18]);
                graph.nodes[sendCom].numSendTimes[goodType]++;
                graph.nodes[sendCom].maxSendGood[goodType] =
Math.Max(graph.nodes[sendCom].maxSendGood[goodType], int.Parse(aryline[18]));

                graph.nodes[receiveCom].receiveGoods[goodType] +=
int.Parse(aryline[18]);
                graph.nodes[receiveCom].numReceiveTimes[goodType]++;
                graph.nodes[receiveCom].maxReceiveGood[goodType] =
Math.Max(graph.nodes[receiveCom].maxReceiveGood[goodType],
int.Parse(aryline[18]));

                if (graph.nodes[sendCom].dic.ContainsKey(receiveCom))
                    graph.nodes[sendCom].dic[receiveCom][goodType]++;
                else
                {
                    List<int> list = new List<int>();
                    for(int i=0;i<5;i++)
                    {
                        list.Add(0);
                    }
                    list[goodType]++;
                    graph.nodes[sendCom].dic.Add(receiveCom, list);
                }

                length++;
            }
            return true;
        }
    }
}
```

Road.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
namespace ConsoleApplicationMath
{
    class Road
    {
        int[,] roadTime;
        Dictionary<string, int> provinceInt;
        Dictionary<int, string> intProvince;

        public Road(string path)
        {
            roadTime = new int[40, 40];
            provinceInt = new Dictionary<string, int>();
            intProvince = new Dictionary<int, string>();
            ReadFile(path);
        }

        public bool Output(string pathFile)
        {
            string strLine = "";
            StreamWriter sw;
            try
            {
                sw = new StreamWriter(pathFile, false,
System.Text.Encoding.Default); //覆盖
                strLine += ",";
//表头
                for (int i = 0; i < provinceInt.Count; i++)
                {
                    if (i > 0)
                        strLine += ",";
                    strLine += intProvince[i];
                }
                strLine.Remove(strLine.Length - 1);
                sw.WriteLine(strLine);
                strLine = "";
                //表的内容
```

```
                for (int j = 0; j < provinceInt.Count; j++)
                {
                    strLine = "";
                    strLine += (intProvince[j] + ",");
                    for(int i=0;i< provinceInt.Count;i++)
                    {
                        if (i != 0) strLine += ",";
                        strLine += roadTime[j, i];

                    }
                    sw.WriteLine(strLine);
                }
                sw.Close();
                string msg = "数据被成功导出到：" + pathFile;
                Console.WriteLine(msg);
            }
            catch (Exception ex)
            {
                string msg = "导出错误：" + pathFile;
                Console.WriteLine(msg);
                return false;
            }
            return true;
        }

        public bool ReadFile(string path)
        {
            string strline;
            string[] aryline;
            int num = 0;
            System.IO.StreamReader mysr = new System.IO.StreamReader(path,
System.Text.Encoding.Default);
            //strline = mysr.ReadLine();
            while ((strline = mysr.ReadLine()) != null)
            {
                aryline = strline.Split(',');
                if (!provinceInt.ContainsKey(aryline[8]))
                {
                    provinceInt.Add(aryline[8], num);
                    intProvince.Add(num, aryline[8]);
                    num++;
                }
                if (!provinceInt.ContainsKey(aryline[13]))
                {
```

```
                    provinceInt.Add(aryline[13], num);
                    intProvince.Add(num, aryline[8]);
                    num++;
                }
                DateTime uploadTime = DateTime.Parse(aryline[20]);
                DateTime sendTime = DateTime.Parse(aryline[17]);
                TimeSpan day = uploadTime - sendTime;
                roadTime[provinceInt[aryline[8]], provinceInt[aryline[13]]] =
day.Days;
            }
            return true;
        }
    }
}
```

Graph.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace ConsoleApplicationMath
{

    class Node
    {
        public bool[] isFinal;
        public bool[] isCreate;
        public HashSet<Node> afterNode;
        public HashSet<Node> beforeNode;
        public int order;
        public int[] receiveGoods;
        public int[] maxReceiveGood;
        public int[] numReceiveTimes;
        public int[] sendGoods;
        public int[] maxSendGood;
        public int[] numSendTimes;
        public int[] nodeClass;
        public int maxClass = 10;
        public int minClass = -1;
        public string privonce;
        public int maxReceive = 0;
        public int[] require;
        public int[] lowRequire;
        public Dictionary<int, List<int>> dic;
        public float[] supplyPercent;
        public int[] minLayerNum;
        public int[] minRoad;
        public double[] protwoan;

        public Node()
        {
            protwoan = new double[5];
            minRoad = new int[5];
            for (int i = 0; i < 5; i++)
                minRoad[i] = -1;
```

```
            minLayerNum = new int[5];
            for (int i = 0; i < 5; i++)
                minLayerNum[i] = 10;
            isCreate = new bool[5];
            isFinal = new bool[5];
            lowRequire = new int[5];
            afterNode = new HashSet<Node>();
            beforeNode = new HashSet<Node>();
            receiveGoods = new int[5];
            sendGoods = new int[5];
            numReceiveTimes = new int[5];
            maxReceiveGood = new int[5];
            numSendTimes = new int[5];
            maxSendGood = new int[5];
            nodeClass = new int[7];
            require = new int[5];
            supplyPercent = new float[5];
            dic = new Dictionary<int, List<int>>();
        }


    }

    class Graph
    {
        public string[,] mapData;
        public int length = 0;
        public Dictionary<string, int> compmIndex;
        //public bool[,] edge;
        //public int num = 0;
        public Node[] nodes;

        public double[] p;

        public Graph(string path)
        {
            p = new double[5];
            nodes = new Node[12000];
            for(int i=0;i<12000;i++)
            {
                nodes[i] = new Node();
            }
            mapData = new string[60000, 7];
            compmIndex = new Dictionary<string, int>();
```

```
            //edge = new bool[12000, 12000];
            ReadFile(path);
            CreateNet();
        }

        void CreateNet()
        {
            string[] temps = new string[7];
            for(int i =0;i<length;i++)
            {
                for(int j=0;j<7;j++)
                {
                    if(mapData[i,j]!="")
                    {
                        if (temps[j] == null)
                        {
                            temps[j] = mapData[i, j];
                        }
                        else if(j!=0)
                        {
                            int a = compmIndex[temps[j - 1]];
                            int b = compmIndex[mapData[i, j]];
                            if (a != b)
                            {
                                nodes[a].afterNode.Add(nodes[b]);
                                nodes[a].order = a;

                                nodes[b].beforeNode.Add(nodes[a]);
                                nodes[b].order = b;

                            }
                        }
                        temps[j] = mapData[i, j];
                        //nodes[compmIndex[mapData[i, j]]].nodeClass[j] = 1;
                        //nodes[compmIndex[mapData[i, j]]].maxClass =
Math.Min(nodes[compmIndex[mapData[i, j]]].maxClass, j);
                    }

                }
            }
            for(int i=0;i<length;i++)
            {
                for(int j=0;j<7;j++)
                {
```

```
                    if (mapData[i, j] != "")
                    {
                        nodes[compmIndex[mapData[i, j]]].nodeClass[j] = 1;
                        nodes[compmIndex[mapData[i, j]]].maxClass =
Math.Min(nodes[compmIndex[mapData[i, j]]].maxClass, j);
                    }
                }
            }
        }

        public void Output(string pathFile)
        {
            string strLine = "";
            StreamWriter sw;
            sw = new StreamWriter(pathFile, false, System.Text.Encoding.Default);
//覆盖
            strLine = "企业序号,省份,所在最大层级,总计发货,总计收货,父节点,子节点";
            strLine.Remove(strLine.Length - 1);
            sw.WriteLine(strLine);
            strLine = "";
            //表的内容
            for (int j = 0; j < compmIndex.Count; j++)
            {
                strLine = "";
                strLine += (j + ","+nodes[j].privonce+","+nodes[j].maxClass +
"," );
                string s1 = "";
                string s2 = "";

                for (int i=0;i<5;i++)
                {
                    if (i > 0)
                    {
                        s1 += ".";
                        s2 += ".";
                    }
                    s1 += nodes[j].sendGoods[i];
                    s2 += nodes[j].receiveGoods[i];
                }
                strLine += (s1 + "," + s2 + ",");
                if (nodes[j].beforeNode.Count == 0)
                    strLine += "null";
                else
                {
```

```
            int i = 0;
            foreach(var tem in nodes[j].beforeNode)
            {
                if (i > 0) strLine += ".";
                i++;
                strLine += tem.order;
            }
        }
        strLine += ",";
        if (nodes[j].afterNode.Count == 0)
            strLine += "null";
        else
        {
            int i = 0;
            foreach (var tem in nodes[j].afterNode)
            {
                if (i > 0) strLine += ".";
                i++;
                strLine += tem.order;
            }
        }
        strLine += ",";
        for (int i=0;i<7;i++)
        {
            if (i > 0) strLine += ".";
            strLine += nodes[j].nodeClass[i];
        }
        strLine += ",";
        for (int i = 0; i < 5; i++)
        {
            if (i > 0) strLine += ".";
            strLine += nodes[j].maxReceiveGood[i];
        }
        strLine += ",";
        for (int i = 0; i < 5; i++)
        {
            if (i > 0) strLine += ".";
            strLine += nodes[j].numReceiveTimes[i];
        }
        sw.WriteLine(strLine);
    }
    sw.Close();
    string msg = "数据被成功导出到：" + pathFile;
    Console.WriteLine(msg);
```

```
        }

        public bool ReadFile(string path)
        {
            string strline;
            string[] aryline;

            System.IO.StreamReader mysr = new System.IO.StreamReader(path,
System.Text.Encoding.Default);
            strline = mysr.ReadLine();
            int num = 0;
            while ((strline = mysr.ReadLine()) != null)
            {
                aryline = strline.Split(',');
                for (int i = 0; i < aryline.Length; i++)
                {
                    mapData[length, i] = aryline[i];
                    if (aryline[i] != "")
                    {
                        if (!compmIndex.ContainsKey(aryline[i]))
                        {
                            compmIndex.Add(aryline[i], num);
                            num++;
                        }
                    }
                }
                length++;
            }
            return true;
        }

        public void CaluP(string pathFile)
        {
            int[] all = new int[5];
            int[] require = new int[5];

            for (int i=0;i< compmIndex.Count; i++)
            {
                for(int j = 0; j < 5; j++)
                {
                    all[j] += nodes[i].receiveGoods[j];
                    require[j] += (nodes[i].numReceiveTimes[j] *
nodes[i].maxReceiveGood[j]);
```

```
                    nodes[i].require[j] = nodes[i].maxReceiveGood[j];
                    /*if (nodes[i].receiveGoods[j] * 10 / 9 >
nodes[i].numReceiveTimes[j] * nodes[i].maxReceiveGood[j])
                    {
                        nodes[i].require[j] = nodes[i].maxReceiveGood[j];
                    }
                    else
                    {
                        if(nodes[i].numReceiveTimes[j]!=0)
                            nodes[i].require[j] = nodes[i].receiveGoods[j] * 10 /
(9 * nodes[i].numReceiveTimes[j]);
                    }*/
                    /*if(nodes[i].receiveGoods[j]*10/9 > nodes[i].numTimes[j] *
nodes[i].maxGood[j])
                         require[j] += (nodes[i].numTimes[j] *
nodes[i].maxGood[j]);
                     else
                        require[j] +=(int)(nodes[i].receiveGoods[j]*10/9);*/

                }
            }
            StreamWriter sw;
            sw = new StreamWriter(pathFile, false, System.Text.Encoding.Default);
//覆盖
            int dn = 0;
            /*for (int i = 0; i < 10000; i+=100)
            {
                dn = i;

                string result = dn.ToString();
                for (int j = 0; j < 5; j++)
                {
                    p[j] = ((double)all[j] + dn) / (double)require[j];
                    if (p[j] > 1) p[j] = 1;
                    result += ("," + p[j]);
                    Console.WriteLine(all[j] + " " + require[j] + " " + p[j]);

                }
                sw.WriteLine(result);
            }*/
        }

        public void ProOne(string path)
        {
```

```
            StreamWriter sw;
            string[] spath = { "1", "2", "3", "4", "5" };
            sw = new StreamWriter(path, false, System.Text.Encoding.Default);
            for (int j=0;j<5;j++)
            {
                string pathx = path + spath[j] + ".txt";
                sw = new StreamWriter(pathx, false, System.Text.Encoding.Default);
                for (int i = 0; i < compmIndex.Count; i++)
                {
                    int allRequire = 0;
                    foreach(var t in nodes[i].dic)
                    {
                        if(t.Value!=null)
                        {
                            allRequire += nodes[t.Key].require[j] * t.Value[j];
                        }

                    }
                    nodes[i].lowRequire[j] = allRequire;
                    if(allRequire!=0)
                    {
                        nodes[i].supplyPercent[j] = (float)nodes[i].sendGoods[j]
/ (float)allRequire;
                    }
                }

                List<int> ltemp = new List<int>();
                string strLine = "[";
                /*for (int i = 0; i < compmIndex.Count; i++)
                {
                    if(nodes[i].receiveGoods[j]!=0)
                    {
                        ltemp.Add(i);
                        strLine += nodes[i].receiveGoods[j];
                        strLine += " ";
                    }

                }
                strLine += "]\n";
                sw.WriteLine(strLine);

                strLine = "[";
               foreach (var tt in ltemp)
```

```
                {
                    strLine += nodes[tt].supplyPercent[j];
                    strLine += " ";
                }
                strLine += "]\n";
                sw.WriteLine(strLine);*/
                int[] ynum = new int[55];
                double[] yp = new double[55];
                for (int i = 0; i < compmIndex.Count; i++)
                {
                    if (nodes[i].supplyPercent[j] !=
0&&nodes[i].receiveGoods[j]!=0&& nodes[i].receiveGoods[j]<250)
                    {
                        ltemp.Add(i);
                        //strLine += nodes[i].supplyPercent[j];
                        //strLine += " ";
                    }

                }

                //strLine += "]";
                //sw.WriteLine(strLine);

                //strLine = "[";
                foreach (var tt in ltemp)
                {
                    int ttt = nodes[tt].receiveGoods[j] / 5;
                    ynum[ttt]++;
                    yp[ttt] += nodes[tt].supplyPercent[j];
                    //strLine += nodes[tt].receiveGoods[j];
                    //strLine += " ";
                }
                //strLine += "]\n";
                string xs="[", ys = "[";
                double re = 2.5;
                for(int i=0;i<50;i++)
                {
                    if (ynum[i] == 0) continue;
                    xs += (re + " ");
                    re += 5;
                    ys += (yp[i] / ynum[i])+" ";
                }
                xs += "]";
                ys += "]";
```

```
            sw.WriteLine(xs);
            sw.WriteLine(ys);

            sw.Close();
        }

    }

    public void protwo(string path)
    {
        StreamWriter sw;
        string[] spath = { "1", "2", "3", "4", "5" };

        for (int j = 0; j < 5; j++)
        {
            string pathx = path + spath[j] + ".txt";
            string s = "[";
            sw = new StreamWriter(pathx, false, System.Text.Encoding.Default);
            for (int i = 0; i < compmIndex.Count; i++)
            {
                if (nodes[i].lowRequire[j] != 0)
                {
                    s += (nodes[i].lowRequire[j] + " ");
                }
            }
            s += "]";
            sw.WriteLine(s);
            sw.Close();
        }
    }

    public void prothree(string path)
    {
        StreamWriter sw;
        string[] spath = { "1", "2", "3", "4", "5" };

        for (int j = 0; j < 5; j++)
        {
            string pathx = path + spath[j] + ".txt";
            string s1 = "[";
            string s2 = "[";

            sw = new StreamWriter(pathx, false, System.Text.Encoding.Default);
```

```
                int[] numy = new int[51];
                double[] py = new double[51];
                for (int i = 0; i < compmIndex.Count; i++)
                {
                    if (nodes[i].lowRequire[j] >=1000&&nodes[i].lowRequire[j] <=
5000&& nodes[i].receiveGoods[j]!=0)
                    {
                        s2 +=
((double)nodes[i].sendGoods[j]/(double)nodes[i].lowRequire[j] + " ");
                        s1 += (nodes[i].receiveGoods[j] + " ");
                        //int n = (nodes[i].lowRequire[j] - 1000) / 100;
                        //numy[n]++;
                        //py[n] += nodes[i].supplyPercent[j];
                    }
                }
                //int start = 1050;
                /*for(int i=0;i<40;i++)
                {
                    if (numy[i] == 0) continue;
                    s1 += (start + " ");
                    start += 100;
                    s2 += py[i] / numy[i]+" ";
                }*/
                s1 += "]";
                s2 += "]";

                sw.WriteLine(s1);
                sw.WriteLine(s2);

                sw.Close();
            }
        }

        public void Short(string path)
        {
            for (int i = 0; i < compmIndex.Count; i++)
            {
                int j = 6;
                while (nodes[i].nodeClass[j] != 1 && j >= 0)
                {
                    j--;
                }
                nodes[i].minClass = j;
            }
```

```
            int ta = 0, tb = 0, tc = 0;
            int times = 10;
            for (int i = 0; i < compmIndex.Count; i++)
            {
                for (int j = 0; j < 5; j++)
                {
                    if (nodes[i].maxClass==0)
                    {
                        foreach (var a in nodes[i].afterNode)
                        {
                            ta++;
                                nodes[i].minRoad[j] = a.order;
                                nodes[i].minLayerNum[j] = a.minLayerNum[j] + 1;

                        }
                    }
                }
            }
            while (times!=0)
            {
                times--;
                for (int i = 0; i < compmIndex.Count; i++)
                {
                    for (int j = 0; j < 1; j++)
                    {
                        if (nodes[i].maxClass!=0&&nodes[i].minLayerNum[j]==10)
                        {
                            foreach (var a in nodes[i].beforeNode)
                            {

                                        //
if(a.minLayerNum[j]+1<nodes[i].minLayerNum[j])
                                         //{
                                             nodes[i].minLayerNum[j] =
a.minLayerNum[j] + 1;
                                             nodes[i].minRoad[j] = a.order;
                                        // }

                            }
                        }
                    }
                }
            }
```

```
            for(int i=0;i<5;i++)
            {
                int j;
                for(j=0;j<compmIndex.Count;j++)
                {
                    if (nodes[j].isFinal[i])
                    {
                        int temp = j;
                        double pernum = (double)nodes[j].receiveGoods[i] / 120;
                        int ttime = 0;
                        while(temp!=-1)
                        {
                            if (ttime > 7) break;
                            nodes[temp].protwoan[i] += pernum;
                            tb++;
                            if (temp == nodes[temp].minRoad[i]) break;
                            temp = nodes[temp].minRoad[i];
                            ttime++;
                        }
                    }
                    Console.WriteLine(j.ToString());
                }

            }
            System.IO.StreamWriter sw = new StreamWriter(path, false,
System.Text.Encoding.Default);
            string s = "";
            for (int i=0;i<compmIndex.Count;i++)
            {
                s += i+",";
                for(int j=0;j<5;j++)
                {
                    if (j != 0) s += ",";
                    s += nodes[i].protwoan[j];
                }
                sw.WriteLine(s);
                s = "";
                Console.WriteLine(i.ToString());
            }
            sw.Close();
        }

        public void Terminal(string path)
```

```
        {
            for (int i = 0; i < compmIndex.Count; i++)
            {
                int j = 6;
                while (nodes[i].nodeClass[j] != 1 && j >= 0)
                {
                    j--;
                }
                nodes[i].minClass = j;
            }
            System.IO.StreamWriter sw = new StreamWriter(path, false,
System.Text.Encoding.Default);
            int[] avr = new int[5];
            int num = 0;
            string s = ",";
            for (int i = 0; i < compmIndex.Count; i++)
            {
                if (nodes[i].minClass == 6)
                {
                    for (int j = 0; j < 5; j++)
                    {
                        avr[j] += nodes[i].receiveGoods[j];
                    }
                    num++;
                }
            }
            for (int j = 0; j < 5; j++)
            {
                if (j != 0) s += ",";
                s += (avr[j]/num);
            }
            sw.WriteLine(s);
            s = "";
            for (int i = 0; i < compmIndex.Count; i++)
            {
                if (nodes[i].minClass == 6)
                {
                    s += i + ",";
                    for (int j = 0; j < 5; j++)
                    {
                        // if (j != 0) s += ",";
                        //s += nodes[i].receiveGoods[j];

                        {
```

```
                        if (j != 0) s += ",";
                        s += nodes[i].receiveGoods[j];
                    }
                }
                sw.WriteLine(s);
                s = "";
                Console.WriteLine(i.ToString());
            }
        }
        sw.Close();
    }


    public void CreateJson(string path)
    {
        StreamWriter sw;
        string s;

        sw = new StreamWriter(path, false, System.Text.Encoding.Default);
        sw.WriteLine("{");
        s = "";
        for (int i = 0; i < compmIndex.Count; i++)
        {
            s += i;
            s += ": [";
            int j = 0;
            foreach(var a in nodes[i].afterNode)
            {
                if (j != 0) s += ",";
                s += a.order;
                j++;
            }
            s += "]";
            if (i != compmIndex.Count - 1)
                s += ",";
            sw.WriteLine(s);
            s = "";
        }
        sw.WriteLine("}");
    }
}

}
```

附录二：Lingo 代码

1. t=(1,2,3,4,5,6,7)时

```
model:
sets:
levels/1,2,3,4,5,6,7/:q,qd,r,rd,m,T,ta;
endsets
data:
ta=1,2,3,4,5,6,7;
enddata
min=0.45*q(1);
@for(levels(i):@gin(m(i)));
@for(levels(i)|i#GT#1:T(i)=m(i)*T(i-1));
@for(levels(i)|i#GT#1:T(i)>T(i-1));
T(1)>=1;
qd(1)=T(1);
q(1)=qd(1);
@for(levels(i)|i#GT#1:qd(i)=qd(i-1)*m(i));
@for(levels(i)|i#GT#1:q(i)=q(i-1)+qd(i));
@for(levels(i):r(i)=i*ta(i)-0.1*T(i));
rd(1)=r(1);
@for(levels(i)|i#GT#1:rd(i)=r(i)-r(i-1));
end
```

```
model:
sets:
levels/1,2,3,4,5,6,7/:q,qd,r,rd,m,T,ta;
endsets
data:
ta=1,2,3,4,5,6,7;
enddata
min=0.475*q(1);
@for(levels(i):@gin(m(i)));
@for(levels(i)|i#GT#1:T(i)=m(i)*T(i-1));
@for(levels(i)|i#GT#1:T(i)>T(i-1));
T(1)>=1;
qd(1)=T(1);
q(1)=qd(1);
```

```
@for(levels(i)|i#GT#1:qd(i)=qd(i-1)*m(i));
@for(levels(i)|i#GT#1:q(i)=q(i-1)+qd(i));
@for(levels(i):r(i)=i*ta(i)-0.05*T(i));
rd(1)=r(1);
@for(levels(i)|i#GT#1:rd(i)=r(i)-r(i-1));
end
```