1. 作业车间

1.1 介绍

生产系统的基本结构之一是作业车间。作业车间的原则是，生产资源是根据属性和容量来分类的。在这些生产资源类别中，将会进行同类型的进程。

这种结构的结果就是不同类型的产品的线路会有所不同。由于大量的不同线路的存在，基本结构的管理是非常复杂的。作业车间的优点是高度的灵活性和相对较低的投资，而缺点则是高缓冲以及长期的吞吐和传输时间。作业车间最适合短期和中期的生产运作。

1.2 情况说明

作业车间包括 5 个机器组，每个机器组执行不同的进程（见图 1-1）。作业车间有 4 种产品，每种都以独立的平均 90 分钟的指数分布间隔到达。每个产品都通过一系列的进程（4 到 6 个）在机器上进行加工生产。作业车间工作时间是从 9 点到 17 点，每天都是延续前一天的工作。

产品在进入进程前会被放置在一个队列中。在这个阶段，没有优先的规则，这意味着所有的产品都是先进先出的。

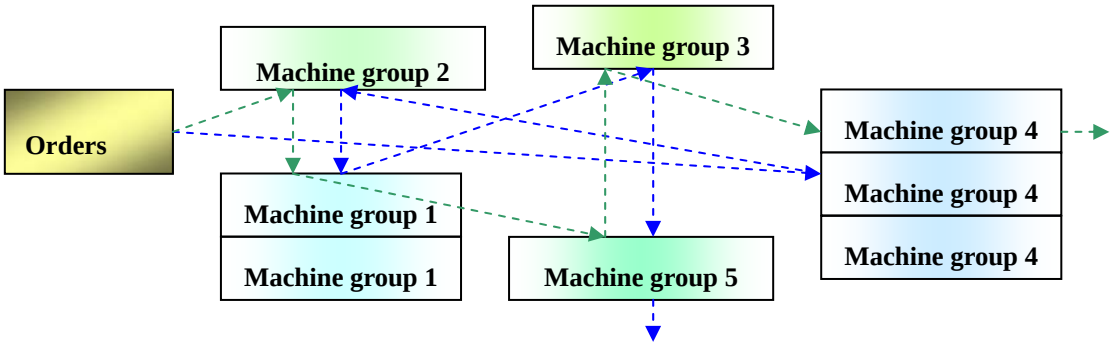

图 1-1 作业车间

表 1-1 展示了产品在机器中的运行线路，以及在每个机器上的循环时间分布。

Job order	*product 1*	*product 2*	*product 3*	*product 4*
1	4 / N(40,5)	1 / Exp(25)	4 / N(100,15)	2 / Uni(20,30)
2	2 / N(10,2)	4 / N(35,10)	2 / Uni(10,20)	1 / Exp(23)
3	1 / Exp(70)	5 / Exp(5)	5 / N(20,5)	5 / N(40,4)
4	3 / Uni(6,14)	2 / Uni(25,45)	1 / N(35,10)	3 / N(25,5)
5	5 / Uni(5,15)	4 / Uni(30,40)		4 / Uni(50,60)
6		3 / Exp(10)		

表 1-1 路线以及运行时间

通过这个表，我们能看到产品 1 通过了 5 个进程：首先在机器组 4 伴随着一个平均为 40 分钟标准差为 5 分钟的正态分布。然后在机器组 2 中，标准正态分布平均值为 10 分钟，标准差为 2 分钟。接下来在机器组 1 中，平均为 70 分钟的指数分布。然后在机器组 3 中，为 6 到 14 分钟的均匀分布，最后在机器组 5 上的 5 到 15 分钟的均匀分布。

管理团队希望大致了解到产品的平均生产时间以及在不同机器组前的等待时间以及队列特征。（模型 1）

看起来在两个机器组中间形成了一个瓶颈。管理团队希望通过在这两个机器组中适用不同的优先级来减少输出时间他们打算通过不同的循环时间来区分在队列中的产品。具有最短循环时间的产品将会被放在前面以来降低平均等待时间（模型 2）.

1.3 要求

1，记录下每个产品的平均循环时间和每个机器的利用率。根据这个决定了两个瓶颈。你是否可以估算出每个产品的输出时间？
2，制作一个模型来研究每台机器前的等待时间以及每个产品的输出时间。（jobshop1.mod）
3，检验管理团队的建议是否能在输出时间上有所改善。（jobshop2.mod）
4，a．用图显示这两种情况下产品输出的时间。（jobshop3.mod & jobshop4.mod）
b．在 Excel 中用图显示输出时间的改善。（jobshop5.mod）

2.ED建模建议

建模方法有若干种，在本章节，我们给出一种针对问题的解决方法，但是，其他的方法也可以是正确的。

框架很容易建造，为一个机器组生成一个 Queue，用 Servers 代替机器，Sink 来移除产品。

路线和 cycletime 导致了实际的问题。一个简单的避免复杂系统的方法是用 Table 中的表。用两个表，一个代表产品路线，一个代表 cycletime。通过这样，来实现图 1-1 的结构：行代表产品，列代表进程步骤。在 cycletime 表中，分布必须重新在 4DScript 中定义。在路线表中，在每个产品中增加一列 0 来表示下一步是系统出口（其他方式也可行）。
接下来，在 Source 和机器组的队列间增加个缓冲区将会非常有用。这个缓冲区的作用是区分不同的产品，并将其引导至相应的机器组的队列。正确的值从 cycletime 表中读出，并转换成机器组的型号。

可以通过标签的形式来准确搜索在表格中的值。例如，定义一个 step 标签和一个 product 标签来读出表格的值：step x 和 product y 指的是表中的（x，y）位置。要注意，在每次进程后 step 都要递增。可以使用 4D 语句 inc（）来使标签中的值递增。参见 4DScript 语言指令获得更多细节。

服务器处的执行时间在 cycletime 表中进行定义。然而，在指向一个表的某个单元格时，ED 将会以文本的形式进行阅读而不是指令。为此，语句 ExecString（）必须使用，这样，括号中的内容就是正确的单元格。

3.模型讲解

1.1 场景一

产品由 4 个 Source 生成然后被送至一个名为 routing 的队列中。这个缓冲区作为一个分配装置，通过队列将产品发送至不同的机器组。每个机器组都有一个队列，及机组中的机器通过队列来选择和处理产品。当处理完成后，将产品送回至 routing 中，以选择下一步的目的地。当所有的进程都完成以后，产品将会被送到一个队列中。每一种类型的产品通过各自的缓冲区以计算每种产品平均时间。之后，产品通过 Sink 原子离开系统。

1）表可以用来定义循环时间以及产品的路线。实验向导是通常的结果测量。初始化原子的作用是一次循环完成后，重置输出时间。

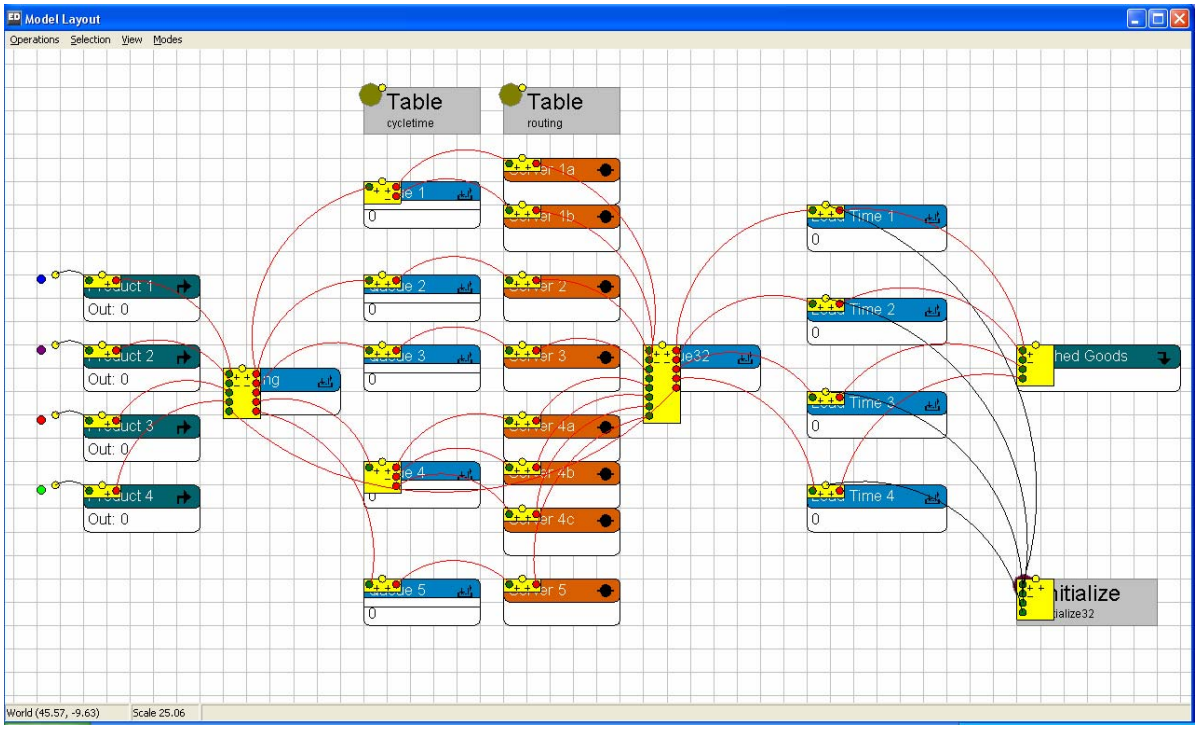

图 4-1 模型图

下面这两个表提供有关的路线和循环时间的信息。

	Product 1	Product 2	Product 3	Product 4
Step 1	4	1	4	2
Step 2	2	4	2	1
Step 3	1	5	5	5
Step 4	3	2	1	3
Step 5	5	4	0	4
Step 6	0	3		0
Step 7		0		

图 4-2 routing

2）Routing 图中，每列表示每个产品通过的机器组。0 表示进程的结束。一个路径

的末端都要注明，因为每个产品通过的进程步骤是不同的。例如，产品 3 先后通过机器组 4,2,5 和 1.

3）循环时间表显示每一步的时间的概率分布。因此，当两个表定义后，你可以设定每一个产品在某个机器组的处理时间是多少。例如，产品 1 的第四步是在机器组 3 上进行，并且持续时间是 6 到 14 分钟的均匀分布。

	Product 1	Product 2	Product 3	Product 4
Step 1	max(0,normal(40,5))	negexp(25)	max(0,normal(100,15))	uniform(20,30)
Step 2	max(0,normal(10,2))	max(0,normal(35,10))	uniform(10,20)	negexp(23)
Step 3	negexp(70)	negexp(5)	max(0,normal(20,5))	max(0,normal(40,4))
Step 4	uniform(6,14)	uniform(25,45)	max(0,normal(35,10))	max(0,normal(25,5))
Step 5	uniform(5,15)	uniform(30,40)		uniform(50,60)
Step 6		negexp(10)		

图 4-3 循环时间

Product 1 to Product 4：到达产品的生成器

退出触发器：这里生成两个标签，一个为 product，另一个是 step。产品名称（1、2、3、4）在 product 中设定，进程步骤在 step 中设定。开始的时候，步骤得到的值为 1.

Inter-arrival time: `negexp(mins(90))`

Send to: 16: Lookup table: Send to the channel specified in row `Label([step], First(c))` column `Label([product], First(c))` of global table named `routing`

Server 1a to Server 5:机器组

- Cycletime:

```
mins(ExecString(
  cycletime(Label([step],First(c)), Label([product],First(c)))
))
```

`Cycletime`（… …）函数用来调用在 `cycletime` 中的正确分布。

`ExecString( )` 用来把从表中单元格的内容当做指令而非文本的格式进行阅读。

`mins( )` 将表中的时间单位从分变成秒。

Trigger on exit: `inc(Label([step], i))`

`Inc(a,b)`意思是把 `a` 进行增大，幅度为 `b`。如果 `b` 没有定义，那么默认为 `1.`

Queue32:

- Send to:

```
if(
 routing(Label([step], First(c)), Label([product], First(c))) = 0,
 Label([product], First(c)),
 5
)
```

Lead Time 1 to Lead Time 4:产品 1 到 4 的输出时间的测量地点

Trigger on entry: `inc(Label([leadtime],c), Age(i))`

`Age(i)`代表相关原子从生成到测量之间经过的时间。

Initialize32:重置 leadtime 表

重置代码: `Repeat(NrIC(c), Label([leadtime], in(Count,c)) := 0)`

实验向导，PFM1 到 PFM4：产品 1 到产品 4 的平均输出时间。

实验向导，PFM5 到 PFM9：机器组 1 到 5 的平均等待时间。

1.2 场景 2

在场景 2 中，我们从第一个模型的描述入手，并添加一些功能。第三部分的输出结果显示，计算机组 2 和 4 是最大的瓶颈。在计算机组前面的队列中的产品会按照最短的循环时间进行排队分类。在物流学中，这种现象称为 SPT 规则（最短处理时间）。

Product 1 to Product 4:

Trigger on exit: `do(`

```
Label([step],1,i) := 1,
Label([product],1,i) := 1,
Label([server1],70,i) := 70,
Label([server2],10,i) := 10,
Label([server3],10,i) := 10,
Label([server4],40,i) := 40,
Label([server5],10,i) := 10
)
```

Queue 2 (also Queue4):

Queue discipline: 4. Sort by Label Ascending: The atoms with the lowest value of the label named `server2` are put in front of the queue

为了解决第四个问题，与输出时间改进相关的图表将会与两个模型结合。Leadtime 队列原子将被一个新原子所代替，名为 Data Recorder。一个图表原子同样需要。添加了这些原子后，布局类似于图 4-4 所示。Data Recorder 的作用是记录数据，然后由图表原子将数据转换成图。

因为这些模型没有运行试验 ，所以初始化原子可以忽略掉。

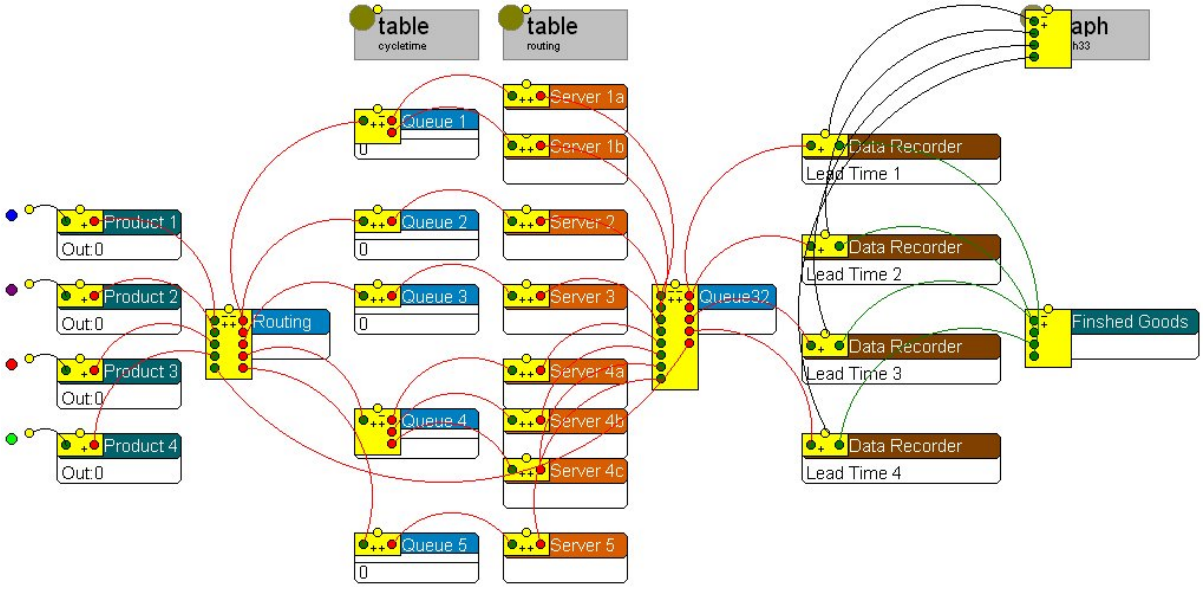

图 4-4 布局

Graph33:每个产品平均生产时间的图。

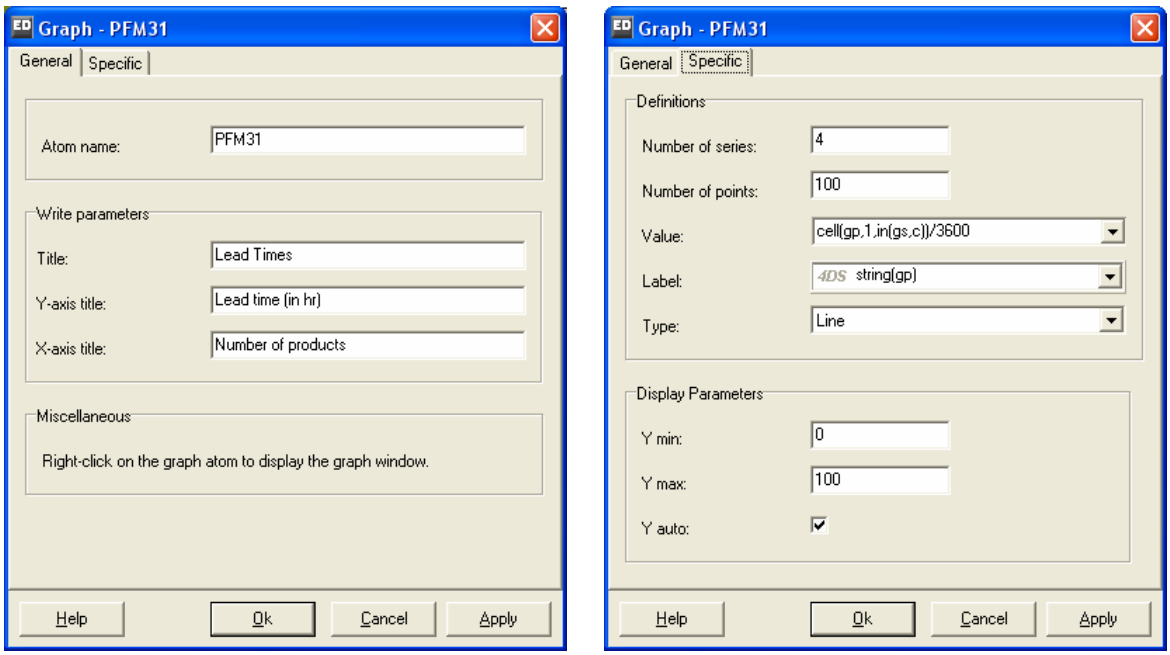

图 4-5 图原子的设置

- No of series: 4 四种产品，所以有 4 个系列。
- No of points: 100 需要绘制成图的产品数量。
- Value: `cell(gp,1,in(gs,c))/3600`

- Step 3 of 3: 不具有特殊性。