XJTU-IE, 2007.9-2008.9, M: xuchen.xuchen@163.com(徐琛), lucifer_tcl@126.com(李慰祖)

第十五章 生产排程

Let all things be done decently and in order。

——I Corinthians

注：*schedule* 在这里往往译作“排程”，有时也作“计划”；也有资料讲“调度”。

15.1 生产排程的目标

所有的制造经理都希望能准时交货，最小化在制品，缩短客户提前期，以及最大化资源利用率。不幸的是，这些目标相互冲突。资源利用率很低时，更容易准时完成作业；持有巨量库存时，客户提前期可以达到零；诸如此类。生产排程的目标即是在这些冲突的目标中达到一个有利可图的平衡。

在这一章中，我们将讨论解决排程问题的各种方法。我们始于考察排程的标准量度，并纵览传统的排程方法；然后讨论为何排程问题如此难以解决，及其对现实世界的启示；接下来开发实用的排程方法，先用于瓶颈资源再用于整间工厂；最后讨论如何连接排程——在概念上属于推——与 CONWIP 这样的拉式环境。

15.1.1 满足交货日期

生产排程的一个基本目标是满足交期。这些交期有两种典型来源：直接来自客户，或者以物料需求的形式来自其他制造流程。

在接单生产环境中，客户交期推动着所有的其他交期。如在第三章所见，一系列的客户需求可以依据相关的物料清单展开，生成对所有低层级工件和组件的需求。

在备货生产环境中不存在客户交期，因为所有的客户订单都要在提出后立即满足。然而，在某些点处，下降的库存触发对制造系统的需求。以这种方式生成的需求与实际的客户订单同等真实，因为如果它们得不到满足客户需求最终也难以完成。（488|489）这些补足存货的需求展开后生成对低层级组件的需求，与客户需求的方式相同。

有几种量度可用于判断交期绩效，包括以下的这些：

服务水平（Service level）（或简称作**服务**），一般用于接单生产系统，是在交期到来之时或之前满足的订单的比例。等价地说，它是周期时间等于或小于计划提前期的加工任务的比例。

补给率（Fill rate）等同于备货生产中的服务水平，定义为由库存满足的需求的比例，也就是说无延迟（without backorder）。

延迟（Lateness）是订单交期与实际完成日期之间的差值。记加工任务 j 的交期为 d_j，实际完成日期为 c_j，则其延迟就是 $L_j = c_j - d_j$。注意到延迟可正（意味着加工任务落后了）可负（意味着加工任务超前了）。因此，平均延迟较小没什么意义。它可能指所有的加工任务都在交期附近完成，这是好事；也可能指每个很晚完成的加工任务都有一个很早完成的与

之对应，这是坏事。对于延迟这个有用的量度，我们在均值之外更要考虑其方差。均值和方差都很小的延迟，才意味着大多数加工任务按时或接近于交期完成。

拖延（Tardiness）定义为加工任务准时或落后时的延迟。超前的加工任务的拖延为零。因此，平均拖延是客户交期绩效的一个有意义的量度。

这些量度提出了几个目标，可用于阐述排程问题。已成为经典的一个就是“最小化平均拖延”。当然了，它的经典性仅仅在生产排程文献中，而非在产业界。可以想象的是，“最小化延迟方差”在工业界中也很少用到。

服务水平和补给率确实用于工业界。这可能是由于拖延难以追踪，以及平均拖延和延迟方差的量度不够直观。按时完成的加工任务的比例很容易陈述，而类似于“落后的平均天数，若超前则计为零”或“加工任务交期与完成日期差值的标准差”则不易说明。可是，服务水平和补给率有着明显的问题。一旦有加工任务落后，不管落后了多少它都被认为对服务不利。天真的方法就会导致荒谬的排程，提倡永远不去完成已落后的加工任务或者对客户撒谎。我们在 15.3.2 节中提出一种交期提报的程序，来避免这些难点。

15.1.2 最大化利用率

在工业界，成本会计鼓励高的机器利用率。较高的资产设备利用率意味着较高的投资回报率，其前提当然是这个设备用于增加收益（即，创造有需要的产品）。不然的话，好的利用率增加的只是库存，而非利润。生产日用品并存储时，高的利用率最有效。

在周期时间、质量和服务水平不过分降低的情况下，工厂物理学也促进高的利用率。然而，回忆起产能定律指出 100%的利用率是不可能的。产线以多大程度接近满载并仍有合理的 WIP 和周期时间，取决于其中变动性的水平。产线的变动性越高，利用率就越低以作补偿。另外，如第七章中经验最差情形（practical worst case）揭示的，平衡的产线比不平衡的有更多拥堵，尤其在变动性很高时。（489|490）这意味着最好不要使产线中所有资源的利用率接近 100%。

与利用率密切相关的一个量度是**生产期（makespan）**，定义为完成固定数量的加工任务所需的时间。对于这组加工任务，生产速率就是加工任务数量除以生产期，利用率就是生产速率除以产能。生产期在产业中的应用并不广泛，但它常见于排程的理论研究。

利用率水平的决策属于厂内计划层级中的的战略问题（第十三章）。由于高层决策没有低层的频繁，不能通过调整利用率来促进生产排程。类似地，产线变动性水平也是高层决策（如，产能和制程设计决策）的结果，这些决策也没有排程决策频繁。因此，出于排程的目的，我们可以假设利用率目标和变动性水平都是给定的。在大多数情况下，瓶颈资源的目标利用率都很高。一种重要的例外情况是，高度变动和定制化的需求需要极其迅速的响应（如，救护车和消防车）。这类系统往往有着非常低的利用率，并且不适合排程。我们通篇强调的都是要求相当高的瓶颈利用率的系统。

15.1.3 削减 WIP 和周期时间

如我们在第二篇中的讨论，保持短的周期时间有一些动机，包括如下的：

1. 更好地响应客户。如果制造产品的时间更短，则客户提前期将会缩短。
2. 保持柔性。改变计划要投放的工件的清单，比改变已经在作业的加工任务带来的破坏性小。较短的周期时间有利于较晚的投放，它们促进了这种类型的柔性。
3. 提高质量。长的周期时间往往意味着系统中长的队列，进而意味着缺陷产生与缺陷检测之间长的延迟。出于这个原因，短的周期时间支持好的质量。

4. 更少地依赖预测。如果周期时间比客户愿意等待的时间长，那么生产的依据将是对需求的预期，而不是对它的响应。由于大多数需求预测都缺乏精确性，那么极端重要的就是在可能的情况下保持周期时间短于提报的提前期。

5. 做出更好的预测。周期时间超出客户提前期越多，预测就要向前扩展地越远。因此，即使周期时间不能被削减到消除对预测的依赖性的点处，周期时间削减仍能缩短预测展望期。这样可以显著地降低预测误差。

里特定律（$CT = WIP / TH$）指出，在产出保持恒定的情况下，削减周期时间等效于削减 WIP。然而，变动性缓冲定量指出，不削减变动性就降低 WIP 将导致产出下降。所以变动性削减常常是 WIP 和周期时间削减计划的一个重要组成部分。

WIP 和周期时间从削减策略的立场上看可能是等效的，但它们在量度上有所区别。（490|491）WIP 易于测度，因为可以对加工任务计数；而周期时间则需要对加工任务进出系统进行计时。在装配作业中，周期时间更加难以测度。例如，考虑一辆汽车，周期时间始于对火花塞和钢铁等组件的订单，还是组件进入装配线（when the classis starts down the assembly line）？对于这些情形，较实际的办法是通过里特定律得到周期时间的间接量度，测度所考察系统的 WIP（以美元计）再除以产出（以美元/天计）。

15.2 排程研究纵览

作为一项实践，排程如同制造业本身一样古老。排程作为一项研究主题，要回溯至二十世纪早起的科学管理运动。但对排程问题的严谨分析直到二十世纪五六十年代计算机出现后才开始。本节中，我们回顾来自排程理论的重要结果。

15.2.1 MRP、MRPⅡ与 ERP

如第三章中讨论所述，MRP 是计算机在排程上的最早应用。然而，MRP 过于简单的模型侵害了它的有效性。其原因可见于第五章，引述如下：

1. MRP 假定提前期是工件的属性，而与车间的状态无关。从本质上来说，MRP 假定产能是无限的。

2. MRP 仅使用一个提前期进行偏移（offsetting）。落后的加工任务常常比过量的库存槽糕，这样就会产生膨胀系统提前期的强烈动机。这将导致更早的投料，更长的队列，以及因而产生的更长的周期时间。

如我们在第二篇中的讨论，这些问题促使一些排程研究者和从业者转向 MRP 的加强版 MRPⅡ，以及最近的 ERP。其他的人否决了 MRP 而转向完全赞同 JIT。然而，大部分排程研究者关注于运筹学领域中的数学表达式，如我们以下的讨论。

15.2.2 经典排程

我们称本节的一组问题为经典排程问题，因为它们是运筹学文献中的传统研究目标。这些问题在大多数部分已经高度简化和抽象化，限制了它们在实际情形中的直接应用。然而，除了从实用的角度来看不够经典，它们还是可以提供一些有益的见解。

大多数经典排程问题涉及一台、两台或可能是三台机器。其他共同的简化假设有：

1. 在问题之始，所有的加工任务都可得（即，没有在作业开始之后到达的加工任务）。
2. 加工时间是确定的。
3. 加工时间与排程无关（即，无换模时间）。
4. 机器永不出故障。
5. 无占先（即，加工任务一旦开始作业，就必须完成）。
6. 加工任务不可取消。（491|492）

这些假设在某些情况下可以将排程问题简化为可管理的问题。原因之一在于，它使我们将注意力限制在简化的排程上，称为排序。一般来说，**排程（schedule）**给出各个加工任务在各个资源处的预期开始时间，而**排序（sequence）**仅给出加工任务的作业顺序。在某情况下，如作业开始时加工任务可得的单机问题，简单的排序就够了；对于较复杂的问题，可能就需要在不同资源处分别排序；而某些问题，则需要完整的排程来引入必要的系统指南（a full-blown schedule is necessary to impart the needed instructions to the system）。所要寻找的排程的形式越复杂，其难度就越大，这点并不奇怪。

在运筹学文献提出的假设的背景下进行的一些流传久远（best-known）的研究问题如下。

最小化单机平均周期时间。首先，注意到对于单机问题，完成所有加工任务的总时间与顺序无关——它就是各个加工任务加工时间之和。所以需要另一个标准。一种选择是平均周期时间（生产排程文献中称为**流程时间（flow time）**），它可以按照加工时间的顺序，将最短的放在最前最长的放在最后，从而取得最小值。这被称为**最短加工时间（shortest process time，SPT）**排序准则。这个结果的主要启示是，加工时间短的加工任务能较快地通过车间，因此趋于减少拥堵。

最小化单机最大延迟。另一个可能的标准是任何加工任务的最大延迟，它可以按照交期的顺序，将最早的放在最前最晚的放在最后，从而取得最小值。这被称为**最早交期（earliest due date，EDD）**排序准则。这种方法背后的直觉（intuition）是，如果可能准时完成所有的加工任务，EDD 排序将做到这一点。

最小化单机最大拖延。单机问题的第三个标准是平均拖延。（注意到它等效于总拖延，因为平均拖延即使总值除以加工任务数量。）不幸的是，并没有一种一定可以使其最小化的方法。EDD 常常是一种很好的探索式方法，但不能保证效果，正如本章末尾的一道练习题显示的那样。同样地，也没有最小化延迟方差的排序准则。我们将要讨论为何这个问题以及许多其他类似问题难以解决。

最小化双机生产期。当生产过程由双机组成，完成所有加工任务的总时间，也即生产期，不再是确定的了。原因在于，某些排序可能会使第二台机器等待第一台完成一个加工任务时发生空闲时间（idle time）。Johnson（1954）提出最小化这种问题生产期的直觉算法，陈述如下：将加工任务分为 A、B 两组，在第一台机器处的加工时间小于或等于第二台处的加工任务进入组 A，余下的进入组 B；组 A 中的加工任务先行，顺序是加工时间由短到长；然后是组 B 中的加工任务，顺序是加工时间由长到短；结果就是最小化双机生产期的序列。

Johnson 算法背后的原理是，由于第一个加工任务在第一台机器完成之前第二台机器空闲，我们希望短的加工任务在首位。（492|493）类似地，由于第二台机器加工最后一个加工任务时第一台机器空闲，我们希望短的加工任务在末位。因此，这个算法暗示小的加工任务有利于削减周期时间和提高利用率。

最小化加工车间生产期。最小化 n 个加工任务以一般路线通过 m 台机器的时间（满足先前讨论的所有假设）是运筹学文献中有名的难题。其困难之处在于，要考虑的可能的排程数目是巨量的。即使对于中度规模的 10 加工任务 10 机器问题，可能的排程就接近于 4×

10^65 种（比地球上的原子还多）。正是由于这一点，10×10 问题直到 1988 年才由一台巨型机经过五个小时的计算得出最优解。

这种问题的一种标准解法称为**分枝定界（branch and bound）**。

15.2.3 分派

从理论（如我们将要看到的）和实际来讲，排程都是困难的。将所有加工任务安排到所有机器的传统方法是在它们到达机器时进行简单的**分派（dispatch）**——依据特定规则分类拣选。最简单的分派准则（也是对客户最公平的）是**先进先出（first-in，first-out，FIFO）**。FIFO 准则简单地按照加工任务到达机器的顺序执行作业。然而，仿真研究显示这条准则在复杂的加工车间效果不佳。效果较好的有前面讨论的 SPT 或 EDD 准则。事实上，如我们第三章对 ERP 车间作业控制的讨论，这些准则都在实际常常用到。研究者和从业者还提出了数以百计的分派准则（见 Blackstone 等 1982 的纵览）。

然而，所有的分派准则在本质上都是短视的。它们的定义即指出，它们仅考虑局部和当前的情况。某台机器现在加工什么的最优决策取决于其他机器以及未来的加工任务，所以我们无法期待指派准则始终运行良好；并且，事实上，它们也不能。但由于对现实系统排程的方法仍然有限，指派在产业中继续发挥着广泛作用。

15.2.4 排程为何困难

我们已经数次指出排程问题很困难。数学的一枝，称为**计算复杂度分析（conputational complexity analysis）**给出了评估它们有多复杂的正式方法。（493|494）尽管计算数学的知识在研究范围之外，我们还是给出定性介绍来正确评价为什么有些排程问题无法寻求最优解。在这些情形中，我们被迫从寻找最优解到获得可行解。

问题类别（Problem Classes）。依据其复杂度，数学问题可分为两类：

1. **P 类问题（Class P problems）**指可通过计算时间随问题规模呈多项式增长的算法求解的问题。

2. **NP 难问题（NP-hard problems）**指没有已知的多项式算法的问题，故求解时间随问题规模呈指数增长（即，比多项式函数快得多）。没人最终证明 NP 难问题的求解没有灵巧的多项式算法，但许多有名的数学家都尝试并且失败了。当前，占优势的证据显示不能为这些问题找到高效率的（多项式）算法。

概略地讲，P 类问题很容易，NP 难问题很困难。此外，有些 NP 难问题比其他的更难些。对于其中一些，高效率算法可以经验地产生良好的近似解。其他的 NP 难问题，包括许多排程问题，甚至用高效率的近似算法也难以求解。

为了对**多项式（polynomial）**与**指数（exponential）**的技术术语有直观了解，让我们考察三加工任务的单机排序问题。对三个加工任务排序有多少种方法呢？首位可以是三个中的任何一个，第二位就有两个候选，末位就只剩下一个。因此，序列数目或称排列（*permutation*）就是 3×2××1 = 6，写作 3!，叫作“3 的阶乘”。如果要在某个目标函数下寻求这个问题的最优排序，我们将不得不考虑（明确地或模糊地）六种选项。由于阶乘函数呈指数增长，我们必须遍历的选项也呈指数增长，因此寻找最优解所需的时间也随问题规模呈指数增长。

这点很重要；原因就是任何指数函数将最终占优于任何多项式函数。例如，10,000n^10 是个很大的多项式值，而 e^n/10,000 看起来很小。事实上，对于较小的 n 值，多项式函数大于指数函数，但在 n = 60 的临界值处指数函数开始占优，而在 n = 80 时指数函数比多项式

大五千万倍。

回到三加工任务的单机问题，我们注意到 3！并不很大。然而，看看这个方程膨胀地多快：3！= 6，4！= 24，5！= 120，6！= 720，等等。随着要排序的加工任务数目变大，可能序列的数目也变得不可想象：10！= 3,628,800，13！= 6,227,020,800，以及

25！= 15,511,210,043,330,985,984,000,000

为了对这个数字是多大有个概念，我们将它与（美国）国债相比，它在本书写作时还未达到五万亿美元。虽然如此，假设它就是五万亿，并且我们用美分来偿还。五百万亿枚美分硬币几乎可以覆盖四分之一个德克萨斯州。与之相比较，25！枚美分硬币可以覆盖整个德州——且厚度超过 *6,000* 英里！现在就看出它的大了。（也许这就是数学家用感叹号来表示阶乘函数的原因。）

现在让我们将这些大数和计算时间联系起来。假设我们有个以 1,000,000 个/秒的速度检视序列的“慢”计算机，并希望建立一个响应时间不超过一分钟的排程系统。若必须检视每个可能的序列来寻找最优解，我们最多能对多少个加工任务进行排序？表 15.1 给出各种数目的加工任务的计算时间，并指出 11 个加工任务是能在短于一分钟的时间内进行排序的最大值。

表 15.1 加工任务排序在慢计算机上所需的计算时间

任务数目	计算时间
5	0.12毫秒
6	0.72毫秒
7	5.04毫秒
8	40.32毫秒
9	0.36秒
10	3.63秒
11	39.92秒
12	7.98分钟
13	1.73小时
14	24.22小时
15	15.14天
…	…
20	77,147年

又假设我们采购了一台比原有“慢”计算机快 1,000 倍的计算机（即，检视速度为十亿个/秒）。现在一秒内最多能对多少个加工任务进行排序？从表 15.2 可以看到问题规模的最大值仅上升到 13 个（如果最长时间增加到 1.5 分钟，就是 14 个）。计算机速度提升 1,000 倍，仅带来在特定时间内求解的问题的最大规模提升 18%。基本结论就是更多地提升计算机速度，也不能显著提高我们求解非多项式问题的能力。

出于比较，现在考察非指数增长的问题。它们被称为多项式问题，原因是求解时间被限制在问题规模的多项式函数（如，*n*^2、*n*^3 等等，其中 *n* 是问题规模的量度）内。

作为一个特例，考虑 15.2.3 节中的加工任务指派问题，并假设采用SPT准则。它要求我们在工站前依据各个加工任务的加工时间对其进行拣选（sort）。[1]对一列元素进行拣选的知

[1] 实际上，我们将以分类后的顺序保持队列，从而不必在每个加工任务到达后重新分类。这样甚至比我们在这里讨论的问题还简单。

名算法的计算时间（即，步骤的数目）与*n*log*n*成正比，其中*n*是要拣选的元素数目。（495|496）这个函数显然小于*n*^2。因此，分派有着多项式复杂度。

假定只出于比较的目的，在慢计算机上用检视 10！个序列的时间（即，3.6 秒）来对 10 个加工任务进行拣选。表 15.3 显示了加工任务超过 10 个时的拣选时间。注意到在 85 个加工任务时仍低于一分钟（对比于排序问题的 11 个）。

表 15.2 加工任务排序在 *1,000* 倍速的计算机上所需的计算时间

任务数目	计算时间
5	0.12微秒
6	0.72微秒
7	5.04微秒
8	40.32微秒
9	362.88微秒
10	3.63毫秒
11	39.92毫秒
12	479.00毫秒
13	6.23秒
14	87.18秒
15	21.79分钟
…	…
20	77,147年

表 15.3 加工任务拣选在慢计算机上所需的计算时间

任务数目	计算时间
10	3.6秒
11	4.1秒
12	4.7秒
…	…
20	9.4秒
30	16.1秒
…	…
80	55.2秒
85	59.5秒
90	63.8秒
…	…
100	72.6秒
200	167.0秒

购买 1,000 倍速计算机后，更有意思的事情发生了。表 15.4 显示了计算时间，并指出可以从慢计算机的 85 个提高到快计算机的大约 36,000 个。它代表着 400%的提升，而排序问题中仅可见到 18%的提升。显然，较快的计算机对“简单的”（多项式）拣选问题帮助很大，而对“困难的”（指数）排序问题益处了了。

表 15.4 加工任务拣选在 *1,000* 倍速的计算机上所需的计算时间

任务数目	计算时间
1,000	1.1秒
2,000	2.4秒
3,000	3.8秒
…	…
10,000	14.5秒
20,000	31.2秒
30,000	48.7秒
35,000	57.7秒
36,000	59.5秒
…	…
50,000	85.3秒
100,000	181.4秒
200,000	384.7秒

对实际问题的启示。大多数现实世界的问题属于NP难类型，规模也很大（如，几百个加工任务和几十台机器），上述结果对制造的实际意义重大。相当确切地说（Quite literally），它们意味着对许多实际规模的排程问题求最优解是不可能的。[2]

幸运的是，实际结果并非如此严重。找不到最优解，并不代表找不到优良解。在某些方面，问题的非多项式本质甚至也有益处，因为它暗示可能会有许多优良解。（496|497）再次考虑 25 个加工任务的排序问题。如果“优良”解极端地少到概率是万亿分之一的程度，那么它仍有 15 万亿个优良解。我们可以使用一种近似算法，称为**探索式方法（heuristic）**，以多项式绩效来搜寻一个优良解。探索式方法有多种类型，如定向搜索（*beam search*）、禁忌搜索（*tabu search*）、模拟退火（*simulated annealing*）以及遗传算法（*genetic algorithms*）这些趣味命名的技术。在讨论瓶颈排程时，我们将详细描述其中的一种方法（禁忌搜索）。

15.2.5 好消息与坏消息

回顾排程研究后，我们得到对设计排程系统有益的一些启示。

坏消息。我们从负面的开始。首先，很不幸，大多数现实世界的问题违背经典排程理论文献中的假设，至少有以下几个方面：

1. 机器数总是多于两台。所以 Johnson 的最小化生产期算法及其诸多变种不能直接应用。

2. 加工时间不确定。在第二篇中我们了解到，随机性和变动性在很大程度上引起了制造系统中可见的拥堵。排程理论忽视了这一点，就违反了一些基本原则。

3. 并非所有的加工任务在开始时都就绪。新的加工任务确实会前来，并还在工厂的整个寿命期内都会源源不断地到达。假装这不会发生或假设在新作业开始之前“清空”工厂，否认了工厂行为的基本准则。

[2] 内存如宇宙间质子的数目，以光速运行的计算机，付出宇宙的寿命，都不足以解决它们中的某些问题。所以，不可能并非夸大之辞。

4. 加工时间常常依赖于序列。换模发生的次数依赖于加工任务的序列。同样的或者类似的加工任务可以共享换模，不相似的则不行。在对瓶颈作业排程时，这将是一个重要的因素。

其次，现实世界的排程问题很困难（从 NP 难的角度看），这意味着

1. 不能期望对许多实际规模的排程问题求最优解。
2. 非多项式方法，如分派等，可能无效。

好消息。值得庆幸的是还有正面消息，尤其当我们意识到许多排程研究都犯了第三类错误：正在求解错误的问题。运筹学文献中形式化了的排程问题都是模型，而非实际。这些模型假定的约束条件对现实世界并不必要，因为在某种程度上我们可以通过控制环境来控制问题。这正是日本人做到的，他们通过缩短换模时间使困难的排程大规模简单化。沿着这些线索思考，排程研究文献的成功与失败能带给我们有益的见解，列示如下。（497|498）

交期（Due dates）：我们对交期确实有一些控制，毕竟公司里有人设置交期或对其谈判。尽管有些公司和大多数排程公式假设它们是给定的，我们也可以不必这样。15.3.2 节给出一种制定既可行又有竞争力的交期的程序。

加工任务分离（Job Splitting）：单机问题的 SPT 准则指出，短时加工任务比长时清空地快。类似地，Johnson 算法的机制也要求在开始和末尾放置短时加工任务。因此，短时加工任务一般能提高平均周期时间和机器利用率的绩效。可是我们在第二篇中也看到小批次增加了换模的次数，因此导致产能损失。故而，如果我们在某种程度上使用大的加工（*process*）批次（即，一次换模后加工许多单位）和小的转运（*move*）批次（即，转运到下道制程之前累积的数量），就可以获得短的周期时间和高的产出。这个在第九章中讲到的批次分离概念，有益于降低系统对排程误差的敏感性。

可行的排程（Feasible Schedule）：最优排程仅仅在数学模型中有意义，实际中我们需要的是优良、可行的方案。这就使得排程问题简单的多了，因为优良解比最优解多得多。事实上，如最近的研究所示，多种探索式方法可以相当有效地生成合理的进度表。

关注瓶颈（Foucu on bottleneck）：瓶颈资源可能支配制造系统的行为，所以对这些资源进行良好的排程就最为重要。单独对瓶颈排程然后将进度表传递到非瓶颈资源，可以将大规模复杂排程问题分解为较简单的部分。还有，通过关注瓶颈，我们可以应用从单机排程文献中得到的见解。

产能（Capacity）：如同交期那样，我们也对产能有所控制。我们可以用某些产能控制措施（如，加班）来影响那些用于计划生产的时间帧。其他的（如，设备或劳动力变更）则需要较长的计划展望期。取决于如何使用加班，排程的程序可以通过提供更多解决不可行性的方案来简化。再者，如果制定长期产能决策时考虑了对排程的影响，这些也会使排程简化。

记住了这些见解，现在我们来更详细地检视一些基本的排程情形。我们提供的并非现成方案——排程环境的范围太广泛，不允许有这样的事情——而是解决实际问题的合理方案的构建模块。

15.2.6 实际有限产能排程（Practical Finite-Capacity Scheduling）

本节中我们讨论一些有代表性的排程方法，商业软件称为**先进计划系统（advanced planning systems）**或**有限产能排程（finite-capacity scheduling）**。由于解决的问题是大规模

和 NP 难的，所有这些方法都是用探索式方法并且不产生最优排程（不管营销材料是如何宣传的）。还有，这些排程应用程序一般都已经添加到 ERP 框架中的 MRP 模块。（498|499）通过这样，他们试图接收 MRP 的计划投入量（planned order release）并安排它们通过车间，从而满足交期，减少换模次数，提高利用率，降低 WIP 等等。不幸的是，如果 MRP 生成的计划投入量代表着一个不可行的计划，则再多的排程也不能使它可行。这是此种“闩住（bolt-up）”方法的主要缺点。

有限产能排程系统一般有两类：基于仿真的和基于优化的。可是，许多基于优化的方法也使用仿真。

基于仿真的排程（Simulation-Based Scheduling）。避免 NP 难的一种方法就是直接忽视它，可以通过开发一个详细的、确定性的（即，加工时间没有不可预测的变动，没有计划的断供，等等）整体系统仿真模型来实现。该模型然后与 ERP 的 WIP 追踪系统联系，从而允许下载活动任务的当前状态信息。需求信息从 ERP 的主生产计划模块或其他来源获取。提前运行模型，记录下各工站处到达和离开的加工任务，从而生成进度表。可以在各工站处应用各种**分派准则（dispatching rules）**，来生成不同的排程计划。再依据经过挑选的绩效指标，寻找“最好的”排程。

仿真方法的好处之一是它比大多数基于优化的方法容易解释。仿真模型以直观的方式模拟实际系统的行为，计划者和执行者能理解它的逻辑。另一个好处是它可以通过简单地更改分派准则快速生成一系列不同的排程，然后向使用者报告机器利用率和拖延的加工任务数目这样的统计数据。使用者从中选取最适合他或她需求的方案。例如，定制化的加工车间可能对准时配送感兴趣，而使用极其昂贵的设备来生产日用品的生产系统则对保持高的利用率更感兴趣。

可是，它也有坏处。首先，仿真需要长时期收集、维护的巨量数据。其次，模型没有考虑变动性，可能导致预测行为与实际之间的巨大差异。但事实上所有的有限产能排程都忽略变动性，所以这个问题不限于仿真方法。结果就是，为了防止误差累积并在最后使排程完全失效，很重要的事情就是频繁更新排程。

第三个问题是，由于对给定的分派准则何时运行良好并无共识，寻找有效排程是个试错的过程。同时，由于分派准则固有的近视性，也可能没有一个分派准则能生成优良的排程。

最后，仿真方法和优化方法，通常用作 MRP 的附件。在一个基于仿真的排程器中，MRP 投料时间被用于定义作为模型输入的作业。可是，若 MRP 投料计划本质上不可行，仅有分派也不能使它变得可行。其他的——产能或需求——必须做出改变。基于仿真的排程方法不适合指出使计划可行的途径。出于这个原因，我们需要一种完全不同的程序，讨论见 15.5 节。

基于优化的排程（Optimization-Based Scheduling）。不同于经典优化问题，基于优化的排程技术对绩效不明的情形使用探索式方法。基于优化的排程技术与基于仿真的排程技术之间的区别在于，前者使用某种算法来积极地搜索优良的计划。（499|500）我们将简短地回顾这些技术，并推荐感兴趣的读者阅读 Morton 和 Pentico（1993）年的著作来获取更多细节。

有许多途径可将复杂的排程问题简化以适用可控的探索式方法。途径之一是使用仿真模型，类似于前面讨论过的基于仿真的方法，并驱使系统寻找能最大化某个目标函数的参数（如，分派准则）。但是，由于仅仅搜索所有策略（如，分派准则所代表的）的一个子集，它并不是一种真正的优化方法。

真正使用优化的一种途径是通过聚焦瓶颈而将产线或车间的排程问题简化为单机排程问题。我们参考了如“类似于 OPT 的”等探索式方法，这个“最优生产技术（OPT）”软件

包由 Eliyahu Goldratt 在二十世纪八十年代开发，并由他人普及开来。尽管 OPT 没有给出求解方法的细节而被当作“黑箱”来销售，它包含了四个基本步骤：

1. 确定车间的瓶颈。
2. 使用有时间缓冲的固定提前期，将交期需求从线尾传递到瓶颈。
3. 最有效地对瓶颈排程。
4. 使用固定提前期，将物料需求从瓶颈反向传递到线首来制定投料计划。

Simons 和 Simpson（1997）更详尽地刻画了这个程序，将其延伸到多瓶颈、工件经过瓶颈不止一次的情形。他们使用的目标函数是交期绩效与利用率的加权平均，故而类似于 OPT 的方法可依据权重的调整生成不同类型的排程。

一种完全不同的基于优化的探索式方法是**定向搜索**（**beam search**），它由前述的分支界定技术派生而来。然而，它并不是逐支检查，而是关注依据某种“机智的”准则挑选出的相对几个分支。结果就是，它比分支界定快得多，但不能保证得到最优解。

局部搜索技术（Local search techniques）也是一整类基于优化的探索式方法。它始于一个给定的排程，然后在其“邻域”搜索一个更好的。它证明了总选择邻近最优解的“贪婪”方法效果不佳，因为许多排程从整体来看并不好但在微小的局部却是最好的。简单的贪婪方法通常终止于这类解中的一个，然后退出计算。

人们已经提出几种方法来避免这个问题。其中之一称为禁忌搜索（tabu search），它将最当前的排程设置为考虑的“禁忌”，从而防止搜索停滞于局部优但整体劣的解。结果是，搜索从局部良好的解跳出，有时却得到一个更差的解。防止局部最佳解的另一种方法是**基因算法（generic algorithms）**的使用，它考虑若干“亲代”排程的特性来生成新的排程，然后仅允许优良的“子代”存活并“再生”新的排程。还有一种是**模拟退火（simulated annealing）**，它以类似于金属逐渐冷却来最小化应力的方式拣出备选的排程。在模拟退火过程的早期，多种随机变化都可能发生，有的优化了排程有的劣化了排程。可是，随着时间的流逝，排程变得稳定起来（即，被“冷却了”），该方法也越来越贪婪。（500|501）当然了，在找不到更有解的时候，所有的局部搜索方法也都“记得”曾在任意点处找到的最优解。我们将在 15.4 节中比较这些技术中的一种（禁忌搜索）和贪婪算法在瓶颈排程中的应用。

基于优化的探索式方法可以以不同方式应用到众多的排程问题中。工厂中最常见的问题形式是（1）最小化拖延（tardiness）的某种量度，（2）最大化资源利用率，以及（3）上述两者的某种组合。我们已经看到，拖延问题（tardiness problem）即使对于单机也是极其复杂的。利用率（如，生产期）问题简单一些，但当机器多于两台时也变得难以处理。所以，开发有效的探索式方法并不简单。Pinedo 和 Chao（1999）给出哪种方法在各种设定下运行良好以及如何有效执行的细节。

基于优化的排程的一个问题是许多现实的排程根本不是优化问题，而是*求满意解的*（*satisficing*）问题。大多数排程专家不会认为有着几个延迟的加工任务的排程是最优的。这是因为一些约束条件，如交期和产能，不是“紧张（*hard*）”约束而更多地是个“祈愿单”。尽管制定者不倾向于增加产能，但若有需求要求时仍要这样做。一个可执行的排程，优于一个优化简明的目标函数却不可执行的排程。

与基于仿真的排程相比，基于优化的排程尽管有其缺点，仍获得了更广泛的应用。一些企业已经成功地将这类软件（有些是内部开发的）与 MRP Ⅱ 系统相连来辅助计划者。Arguello（1994）提供了一项关于半导体产业中有限产能排程软件（基于优化以及基于仿真的）的调查。由于其中大多数软件也已用于其他产业，这项调查对非半导体产业的实践者也有参考意义。

15.3 连接计划和排程

在企业资源计划系统内，MRP 模块基于固定的提前期和其他简化的假设生成计划投入量。如前面的讨论，这常常导致不可行的排程。同样地，由于有限产能排程远非一项成熟的技术，见于 ERP 的许多先进计划系统都是复杂和笨重的。生成产能可行的排程所需的时间使其不大可能通过常规方法实现。

这些问题引起以时间、软件和人力的形式区别对待物料计划（如，MRP）、产能计划（如，产能需求计划（CRP））和生产执行（如，投料和分派）。例如，物料需求计划不考虑产能从而决定需要哪些物料并提供一个原初的计划。然后产能计划职能核查是否具备所需的产能。如果不具备，或者是使用者（如，通过重复 CRP）或者是系统（如，通过使用某种先进计划系统）将试图重排投料。但由于设定物料需求时未考虑产能，产能计划问题可能被不必要地复杂化了（事实上，这种情况不可能发生）。（501|502）这个问题，又被一个部门（如，生产控制）生成生产计划（物料以及产能的）并传给另一个部门（制造）来执行的一般做法加重了。

计划/执行分离问题的一项重要的矫正办法是周期时间压缩。如果周期时间很短（如，变动性削减和/或使用某种拉式系统的结果），短期生产计划职能（即，满足需求）能提供生产计划。[3]然而，在那之前，生产计划和排程问题必须从受限于产能和需求约束的优化（*optimization*）形式，转变为确定采取何种措施来产生可执行的生产计划的可行性分析（*feasibility analysis*）的形式。这就需要一种同时考虑物料和产能需求的程序。在理论上，它可以通过大规模的数学规划模型实现。可是，这类求解往往很慢，并因而阻止了情况变化时要经常做出的可行性检查。我们在 15.5.2 小节给出一种快速进行可行性检查的实用探索式方法。

本章的余下部分聚焦于开发实用的排程程序的中心议题。在本节的余下部分，我们考虑简化排程问题的技术，即有效成批（effective batching）和交期提报（due date quoting）。15.4 节在 CONWIP 产线的背景下处理瓶颈排程的问题。对于更一般的情形，我们在 15.5 节中提供一种同时考虑物料和产能的方法。在 15.6 节中，我们展示了如何在拉式环境中使用排程（其天性是属于“推”的）。

15.3.1 最优批次

在第九章中我们已看到加工批量对周期时间有着极大的影响。因此，批次同样会严重影响排程。通过明智地选择批量从而保持加工时间较短，排程就更容易满足交期。我们现在开发确定批次以使周期时间最小化的方法。

最优串联批次（Optimal Serial Batch）。图 15.1 显示了平均周期时间和串联批次之间的关系。使用第九章中推导的公式，我们可以为某工站处某一工件绘制出总周期时间曲线，并找出最优的批量。然而，它很不方便，在多种工件互相影响的情况下也不实用。所以我们转向推导一种简单的程序，首先寻找（近似地）工站的最优利用率，然后用它来计算串联批量。我们首先计算单一工件情形，继而将其推广到多产品系统。

[3] 长期生产计划，也称作集结计划，用于设定产能水平，计划劳动力变更，等。（见第十六章）。

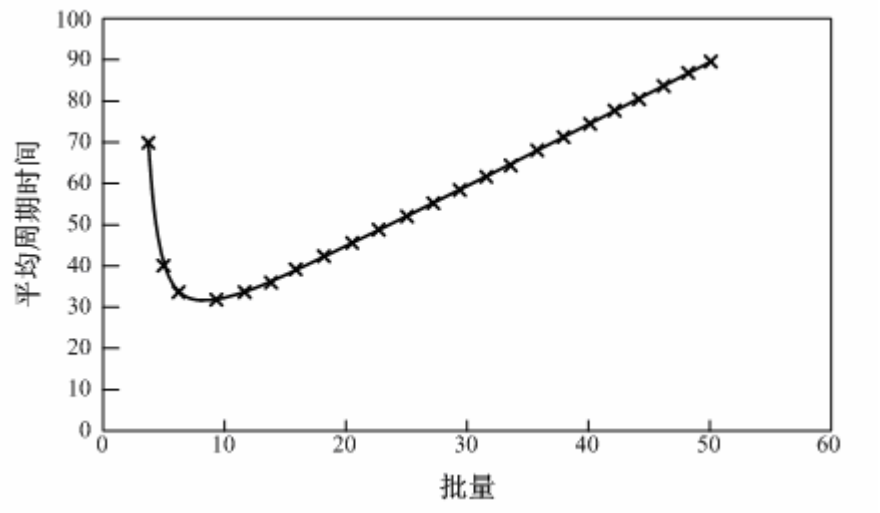

图 15.1 平均周期时间与串联批量

技术性注释：最优串联加工批量

我们首先考虑产品族同质，加工时间、换模时间和到达都服从泊松分布的情形。该问题是在单一工站处寻找能最小化总周期时间的串联批量。如果仅有一个工站有着显著的换模并趋于是瓶颈时，这个批次对产线来说应当是优良的。

继续使用第九章中的记号，批次的有效加工时间是 $t_e = s + kt$，利用率是（502|503）

$$u = \frac{r_a}{k}(s + kt)$$

现在定义“无换模的利用率”$u_0 = r_a t$。微小的代数变换给出有效加工时间的另一种形式为

$$t_e = \frac{su}{u - u_0}$$

由于我们假设的是泊送到达（产品来自于多个源头时是个贴切的假设），到达的平方变异系数（SCV）为 $c_a^2 = 1$，平均周期时间为

$$CT = \left(\frac{1 + c_e^2}{2}\right)\left(\frac{u}{1-u}\right)\frac{su}{u - u_0} + \frac{su}{u - u_0} \qquad (15.1)$$

写成这种方式后，周期时间就仅仅是 u，而非 k 和 u，的函数。故而最小化周期时间归结为寻找最优工站利用率。在（15.1）式中对 u 求导，令其等于零，解出

$$u^* = \frac{\alpha u_0 + \sqrt{\alpha^2 u_0^2 + [\alpha(1 + u_0) + 1]u_0}}{\alpha(1 + u_0) + 1} \qquad (15.2)$$

其中 $\alpha = (1 + c_e^2)/2 - 1$。注意到对于 $c_e^2 = 1$ 的特殊情形，$\alpha = 0$，且有

$$u^* = \sqrt{u_0} \qquad (15.3)$$

但即使c_e^2不等于一时，u^*的值一般仍接近$\sqrt{u_0}$。例如，当u_0= 0.5、c_e^2= 15 时，差别低于百分之五。还有，u_0越接近一（即，无换模系统的利用率越高），对于所有c_e^2，u^*和$\sqrt{u_0}$的差别就越小（见 Spearman 和 Krockel 1999）。

为了得到批量，回忆起

$$u^* = \frac{r_a}{k}\left(s + k^* t\right) = \frac{r_a s}{k} + u_0$$

就可以求出k^*。

上述分析给出某一工站处最小化周期时间的串联批量的优良近似值

$$k^* = \frac{r_a s}{u^* - u_0} \approx \frac{r_a s}{\sqrt{u_0} - u_0} \qquad (15.4)$$

其中$u_0 = r_a t$。我们通过下来的例子来说明这个公式。（503|504）

例子：最优串联批次（单产品）

考虑 9.4 节和图 15.1 给出的串联批次例子。不考虑换模时的利用率u_0为

$$u_0 = r_a t = (0.4 \text{ 件/小时})(1 \text{ 小时}) = 0.4$$

因此，由（15.3）式，最优利用率近似为

$$u^* = \sqrt{u_0} = \sqrt{0.4} = 0.6325$$

并由（15.4）式，最优批量为

$$k^* = \frac{r_a s}{u^* - u_0} = \frac{0.4(5)}{0.6325 - 0.4} = 8.6 \approx 9$$

从图 15.1 中可以看到，这个值确实很接近实际最优值八。周期时间的差别低于百分之一。

最优工站利用率非常接近无换模利用率的平方根的这种关系极其稳健。这使它成为在更一般的多产品族系统中设定串联批量程序的基础。在下一个技术性注释中，我们开发出这样的方法。

技术性注释：多产品最优串联批次

为了给多产品情形建模，我们有如下的定义：

n = 产品的数目

i = 产品的标号，$i = 1, ..., n$

r_a^i = 对产品 i 的需求（件/小时）

t_i = 产品 i 的平均加工时间（小时）

c_{ti}^2 = 产品 i 平均加工时间的 SCV

s_i = 切换到产品 i 的平均换模时间（小时）

c_{si}^2 = 切换到产品 i 的平均换模时间的 SCV

t_e = 所有产品的平均有效加工时间（小时）

c_e^2 = 所有产品平均有效加工时间的 SCV

u_0 = $\sum_i r_{ai}t_i$ = 工站的无换模利用率

u = 工站利用率

k_i = 产品 i 的批次

使用 *VUT* 方程计算工站周期时间为

$$CT = \left(\frac{Vu}{1-u}+1\right)t_e \qquad (15.5)$$

其中 $V=\left(1+c_e^2\right)/2$。为了利用上述公式，还必须通过单一工件的数据计算出 u 、t_e 和 c_e。利用率为

$$u = \sum_{i=1}^{n}\frac{r_{ai}}{k_i}\left(s_i+k_i\right)$$

有效加工时间，在某种意义上等同于“均值的均值”。换句话说，若产品 i 的成批平均加工时间为 $s_i+k_it_i$，并且该批次为产品 i 的概率为 π_i，则有效加工时间为（504|505）

$$t_e = \sum_{i=1}^{n}\pi_i\left(s_i+k_it_i\right) \qquad (15.6)$$

某个批次属于某种给定产品类型的概率是该类型到达速率与总到达速率的比值

$$\pi_i = \frac{r_{ai}/k_i}{\sum_{j=1}^{n}\left(d_j/k_j\right)} \qquad (15.7)$$

通过标准随机性分析，我们计算出有效运行时间的方差 σ_e^2 为

$$\sigma_e^2 = \sum_{i=1}^{n}\pi_i\left(k_ic_{ti}^2t_i^2+c_{si}^2s_i^2\right)+\left[\sum_{i=1}^{n}\pi_i\left(s_i+k_it_i\right)^2-t_e^2\right] \qquad (15.8)$$

因而有效 SCV 就是 $c_e^2=\sigma_e^2/t_e^2$。

现在，假设如单产品情形中$u^* = \sqrt{u_0}$是最优利用率的良好近似值，批量问题简化成寻找一组k_i值，达到u^*并保持c_e^2和t_e较小。从（15.5）式中可以明显看出，这将引起较小的周期时间。注意到如果$s_i + k_i t_i$的值，即所有的平均运行时长，都相等的话，（15.8）式中方括号中的项将等于零。因而，保持c_e^2和t_e都较小的一种方法是最小化平均运行时长以及使所有的运行时长相等。我们可将其表述为如下的优化问题。

最小化　　L

受限于：　　$s_i + k_i t_i \le L$，　$i = 1, ..., n$

$$\sum_{i=1}^{n} \frac{r_{ai}}{k_i}\left(s_i + k_i\right) = u^*$$

求解如

$$s_i + k_i t_i = L$$

$$k_i = \frac{L - s_i}{t_i} \qquad (15.9)$$

然后求L，使用约束

$$\sum_{i=1}^{n} \frac{r_{ai}}{k_i}\left(s_i + k_i\right) = u^*$$

$$\sum_{i=1}^{n} \frac{r_{ai}}{k_i} = u^* - u_0$$

$$\sum_{i=1}^{n} \frac{r_{ai} s_i t_i}{L - s_i} = u^* - u_0$$

如果换模时间都接近平均换模时间，标记为$\bar{s}$，则可解出如下形式的L。

$$L = \frac{\sum_{i=1}^{n} r_{ai} s_i t_i}{u^* - u_0} + \bar{s} \qquad (15.10)$$

将其带入（15.9）式即产生最优批量的近似值。

上述分析给出多产品、有换模时某一工站处最小化周期时间的产品 *i* 的串联批量为

$$k_i^* = \frac{L - s_i}{t_i} \qquad (15.11)$$

其中 L 由（15.10）式算出。（505|506）

例子：最优串联批次（多产品）

考虑一个通过搅拌机生产三种不同产品的工艺。对每种产品的需求是 15 单位/月，由使用常量批次的 MRP 系统控制。当搅拌机从一种产品切换到另一种产品时，都需要一次清洗。产品 A 和 B 需要四小时的混合和八小时的清洗。产品 C 需要八小时的混合和 12 小时的清洗。所有的加工和换模时间的变异系数为 1.5。搅拌机每天两班运行，每周五天。每班损失一小时，每月损失 52/12 周，这项损失的均值为 303.33 小时/月。

按照常规思维（如，EOQ 模型），切换所需的时间越长，批量就应越大。该企业现在对产品 A 和 B 使用 20 单位的批量，对产品 C 使用 30 单位的批量。该工艺的平均周期时间现在约是 32 个工作日。但能做得更好吗？

将三种产品的需求按小时转化有 r_{ai} = 15/303.33 = 0.0495 单位/小时。无换模的利用率因此是

$$u_0 = 0.0495(4 + 4 + 8) = 0.7912$$

故而，最优利用率为 $u^* = \sqrt{u_0} = \sqrt{0.7912} = 0.8895$。

平均换模时间是 $\bar{s}$ =（8+8+12）/3 = 9.33 小时，所以（15.10）式所需的和就是

$$\sum_{i=1}^{3} r_{ai} s_i t_i = 0.0495[8(4) + 8(4) + 12(8)] = 7.912$$

并因而有

$$L = \frac{7.912}{0.8895 - 0.7912} + 9.33 = 89.92$$

通过它就可以求出最有批量的近似值如下

$$k_A = k_B = \frac{L - s_A}{t_A} = \frac{89.82 - 8}{4} = 20.46 \approx 20$$

$$k_C = \frac{L - s_C}{t_C} = \frac{89.82 - 12}{8} = 9.73 \approx 10$$

使用这些批量后的平均周期时间为 20.28 天，降幅超过 36%。对所有可能的批量进行的遍历结果显示，这确实是最优解。

注意到 C 的批次小于 A 和 B。在独立产品假设下的 EOQ 逻辑认为 C 应当有个较大的批量，因为它的换模时间较长。但为了保持各种产品的运行时长相等，我们需要降低 C 的批量。

最优并联批次（Optimal Parallel Batches）。有着并联批次的机器才是真正的批处理机器，如机加车间的热处理炉或电路板工厂的镀铜装置。在这些情况下，不管一次加工多少工件（批量）加工时间都是相等的。

在并联批次情形，基本的权衡是有效产能利用率，由它我们期望较大的批次，以及最小的等待成批时间，由它我们期待较小的批次。若机器是瓶颈，一般的最佳做法就是使用尽可

能大的批次。在非瓶颈处，可能的最佳做法（考虑到周期时间）是加工串联批次。（506|507）以下的技术性注释介绍了一种在单机处确定最优并联批次的程序。

技术性注释：最优并联批次

为了寻找在并联批处理时最小化周期时间的批量，有一条捷径是先确定最优利用率，然后将其转换成批量，如我们在串联加工批次中所做的那样。

为了实现这个过程，我们使用如下的标记：

r_a = 到达速率（件/小时）

c_a = 到达时间间隔的变异系数（CV）

t = 每批所需的加工时间（小时）

c_e = 每批加工时间的有效 CV

B = 最大批量（可装入制程的工件数量）

u_m = $r_a t$ = 批量为一时的利用率

u = 工站利用率

k = 并联批次

注意到利用率 $u = r_a/(k/t)$，为使工站稳定而必须小于一。因为 $u = u_m/k$，我们可以用 $u_m = r_a t$ 对其进行改写，这也意味着批量 $k = u_m/u$。

回忆起第九章中，并联批作业的总时间包括等待成批时间（WTBT）、排队时间以及作业本身的时间，即

$$CT = WTBT + CT_q + t$$

$$= \frac{k-1}{2r_a} + \left(\frac{c_a^2/k + c_e^2}{2}\right)\left(\frac{u}{1-u}\right)t + t$$

$$= \frac{k-1}{2ku}t + \left(\frac{c_a^2/k + c_e^2}{2}\right)\left(\frac{u}{1-u}\right)t + t \qquad (15.12)$$

其中最后一个等号依据 $r_a = uk/t$。

代入 $k = u_m/u$，改写（15.12）式，有

$$CT = \left(\frac{u_m/u - 1}{2u_m} + \frac{c_a^2 u/u_m + c_e^2}{2}\frac{u}{1-u} + 1\right)t \qquad (15.13)$$

不幸的是，从利用率的角度最小化 CT 不能得出一个简单的表达式。所以为了近似估计，我

们令 $\beta = c_a^2 u / u_m$ 并假定它可以被当作常数项。我们这样做的理由是，当 k 很大时，u / u_m 将很小，β 就微不足道了。周期时间的表达式因而简化成

$$CT \approx \left(\frac{1}{2u} - \frac{1}{2u_m} + \frac{\beta + c_e^2}{2} \frac{u}{1-u} + 1 \right) t$$

$$= \left(\frac{y(u)}{2} - \frac{1}{2u_m} + 1 \right) t \qquad (15.14)$$

其中

$$y(u) = \frac{1}{u} + \frac{\beta + c_e^2 u}{1-u}$$

最小化（15.14）式等效于对 u 最小化 $y(u)$，而这是相当容易的。将 $y(u)$ 对 u 求导，令其为零，可以解出（507|508）

$$u = \frac{1}{1 + \sqrt{\beta + c_e^2}} \qquad (15.15)$$

若我们的设想成立，β 接近零，则最优利用率就简化为

$$u^* \approx \frac{1}{1 + c_e} \qquad (15.16)$$

当 c_e^2 不太小时，去掉 $\beta = c_a^2 u / u_m$ 项影响不大，（15.16）式给出一个相当好的近似值。然而，当 c_e^2 很小时，去掉这一项将显著改变问题的实质。实际上，当 $c_e^2 = 0$ 时，（15.16）式意味着最优利用率等于一！当然了，我们都知道这是不合理的，只要到达过程中出现了一点变动性，其队列就会爆炸般延长。

故而，回到先前并再次引入 β 项，将（15.16）式的 u^* 近似表达式代入 $c_a^2 u / u_m$，有

$$\beta = \frac{c_a^2}{u_m (1 + c_e)}$$

且有

$$u^* = \frac{1}{1 + \sqrt{c_a^2 / [u_m (1 + c_e)] + c_e^2}} \qquad (15.17)$$

一旦我们知道了最优利用率 u^*，就可以轻易通过 $k = u_m / u$ 得到最优批量 k^*。

故而，并联批次工站处最小化周期时间的加工批量为

$$k^* = \frac{u_m}{u^*} \qquad (15.16)$$

其中$u_m = r_a t$，u^*用（15.17）式计算得出。为了获得一个整数的批量值，我们可以依据习惯对（15.18）式的值进行向上圆整。这种做法将趋于部分抵消技术性注释中由近似估计带来的偏差。

除了作为计算工具，（15.17）和（15.18）式还给出一些定性的见解。它们指出，工站处的变动性越大，所能承担的利用率就越小。特别地，随着c_e或c_a上升，系统的利用率将下降。这是一个必然结果，工厂物理学对于变动性和利用率的结论就显示这两个因素联合起来降低了绩效。因此，当优化绩效时，我们必须通过较低的利用率来抵消较高的变动性。

我们通过以下的例子说明并联批次公式的应用。

例子：最优并联批次

再考虑 9.4 节中讨论的烧结作业。不管一次投入多少，所有的单位都要在温度受控的房间停留 24 小时。烧结室一次可容纳 100 件，到达速率是 1 件/小时（24 件/天）。图 9.6 绘出这个例子中周期时间与批量的关系，并显示周期时间在批量为 32 时最小，最小值为 42.88 小时。

现在使用上面的最优批量公式解决这个问题。到达速率r_a=1 件/小时，到达过程服从泊松分布，故c_a = 1。（508|509）加工时间t = 24 小时，有着c_e = 0.25 的变动性。所以出于稳定性的考虑我们需要一个$k > u_m = r_a t$ = 24 件的批量，这就意味着最小批量是 25。

然而，如果用 25 件作批量，就有

$$u = \frac{r_a}{k/t} = \frac{1}{25/24} = 0.96$$

$$WTBT = \frac{k-1}{2r_a} = \frac{25-1}{2(1)} = 12 \text{ 小时}$$

$$CT_q = \left(\frac{c_a^2/k + c_e^2}{2}\right)\left(\frac{u}{1-u}\right)t$$

$$= \left(\frac{1/25 + 0.25^2}{2}\right)\left(\frac{0.96}{1-0.96}\right)24 = 29.52 \text{ 小时}$$

因此，经过热处理作业的平均周期时间将是

$$CT = WTBT + CT_q + t = 12 + 29.52 + 24 = 65.52 \text{ 小时}$$

现在考虑另一个极端。令k = 100，即烧结室的容量。

$$u = \frac{r_a}{k/t} = \frac{1}{100/24} = 0.24$$

$$WTBT = \frac{k-1}{2r_a} = \frac{100-1}{2(1)} = 49.5 \text{ 小时}$$

$$CT_q = \left(\frac{c_a^2/k + c_e^2}{2}\right)\left(\frac{u}{1-u}\right)t$$

$$= \left(\frac{1/100 + 0.25^2}{2}\right)\left(\frac{0.24}{1-0.24}\right)24 = 0.27 \text{ 小时}$$

故经过热处理作业的平均周期时间将是

$$CT = WTBT + CT_q + t = 49.5 + 0.27 + 24 = 73.77 \text{ 小时}$$

现在来寻找最优批量，首先计算最优利用率

$$u^* = \frac{1}{1+\sqrt{c_a^2/[u_m(1+c_e)]+c_e^2}}$$

$$= \frac{1}{1+\sqrt{1/[24(1+0.25)]+0.25^2}}$$

再使用（15.18）式来计算

$$k^* = \frac{u_m}{u^*} = \frac{24}{0.7636} = 31.43 \approx 32$$

注意到它正是我们在图 9.6 中观察到的最有批量。还有，最小批量产生的周期时间比最优值高出 53%，最大批量产生的周期时间比最优值高出 72%。显然，在并联批作业中，成批可能对周期时间有着显著的影响。（509|510）

15.3.2 交期提报（Due Date Quoting）

变动性削减（第九章）、拉式生产（第十章）以及经济批量（前面刻画的）都使得生产系统易于计划。简化排程的另一项技术是交期提报。包含了交期的排程问题都极其困难，而交期设定问题相对容易些，所以似乎值得从此入手。当然了，在现实世界，任何实施都远比数学计算复杂。开发一个交期提报系统将包含一个更为困难的问题——使制造和销售人员彼此沟通。

除了人员问题，交期提报系统的困难性还取决于制造环境。为了给出合理的交期，我们必须能够依据已有的投料计划预知加工任务何时完成。如果环境很复杂以致于那难以做到，则交期提报也将很困难。然而，若我们简化环境使其更容易预知，则交期提报将变得简单易行。

CONWIP 产线的交期提报。最易预知的制造系统之一就是 CONWIP 产线。如先前的讨论，CONWIP 行为可以通过传送带模型来刻画。这使我们能为提报交期开发一中简易的程序。

考虑这样一条CONWIP产线，其中保持ω标准单位[4]的WIP，每期（如，班、天）产出有着稳定的均值μ和标准差σ。假设某客户下了一张代表c标准单位加工任务的订单，并且我们能自由地指定交期。为了平衡响应性和可靠性，我们希望提报能保证服务水平（按时送达的

[4] 一个标准单位指的是在产线瓶颈处需要特定数量的时间的 WIP。故而，以瓶颈处的时间来量度，CONWIP 在产线中保持了恒定负载。

概率）为*s*的最早交期。当然了，实现这个目标的交期取决于在这个新订单之前还有多少加工任务。这又进而取决于客户订单是如何排序的。一种可能情形是按照先到先服务的顺序处理，这时我们令*b*代表当前的积压（即，已接收但尚未投入产线的标准加工任务的数量）。作为一种选择，可以通过在较低优先序的加工任务交期之前设置类似“占位符”的形式，为高优先序的加工任务保持“紧急通道”（见图 15.2），这时，我们定义*b*代表在下一个紧急通道之前的作业数量。

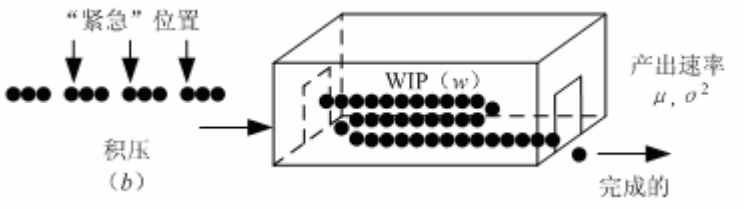

图 15.2 提报提前期方法的简图

对于上述两种情形，客户订单都将在产线产出 $m = \omega + b + c$ 标准单位加工任务后得到满足。因此寻找一个能确保服务水平为 s 的最早交期，就等效于寻找一个能以 s 的的概率完成 m 标准单位加工任务的时间。我们在以下的技术性注释中推导求这个时间的表达式。(510|511)

技术性注释：CONWIP 产线的交期提报

令 X_t 为表示第期加工量（以标准单位计）的随机变量。假设 X_t，$t = 1, 2, ...,$ 相互独立并都服从均值为 μ 方差为 σ^2 的正态分布。为了确保时间 l 之内完成的概率为 s，下式必须成立：

$$p\left\{\sum_{t=1}^{l} X_t \le m\right\} = 1 - s$$

注意到由于独立随机变量的均值和方差这些添加条件，时间 l 之内完成加工任务数量就是

$$\sum_{t=1}^{l} X_t \sim N\left(l\mu, l\sigma^2\right)$$

这就表示，它服从均值为 μ 方差为 σ^2 的正态分布。因此，

$$p\left\{Z \le \frac{m - l\mu}{\sqrt{l}\sigma}\right\} = 1 - s$$

其中 Z 是个标准 0-1 正态随机变量。

因此，

$$\frac{m - l\mu}{\sqrt{l}\sigma} = z_{1-s} \qquad (15.19)$$

其中 z_{1-s} 从标准正态分布表中查得。

我们可以将（15.19）式写成

$$l^2\mu^2-\left(2\mu m+z_{1-s}^2\sigma^2\right)l+m^2=0 \qquad (15.20)$$

就可以通过二次方程求解。它有两个根：只要 $s\geqslant 0.5$，就总取较大的那个。它就是（15.21）式。

在一条 WIP 水平为 ω 的 CONWIP 产线中，排在 b 标准单位之后的包含 c 标准单位的新加工任务若要确保服务水平为 s，最小能提报的提前期是

$$l=\frac{m}{\mu}+\frac{z_{1-s}^2\sigma^2\left[1+\sqrt{4\mu m/\left(z_{1-s}^2\sigma^2\right)+1}\right]}{2\mu^2} \qquad (15.21)$$

其中 $m=\omega+b+c$。

对上述方法的一种可能批评是它以服务水平为前提。因此，滞后一天的加工任务被认为与滞后一年的加工任务同样糟糕。一种能从客户角度较好地追踪绩效的量度是拖延（tardiness）。幸好，结果显示，以相同的服务水平为每个加工任务提报，也产生平均拖延约束下的最小预期提报提前期（见 Spearman 和 Zhang 1999）。

还有，为了简化实施从而减少绩效损失，（15.21）式可以被下式替代

$$l=\frac{m}{\mu}+\text{ 计划的库存时间} \qquad (15.22)$$

其中计划的库存时间可以通过试错法调整到可接受的服务水平（见 Hopp 和 Roof 1998）。（511|512）

例子：交期提报

假设我们有这样一条 CONWIP 产线，其中保持 320 标准单位的 WIP，每天产出的均值是 80 件，标准差是 15 件。产线接收了一个 20 标准单位的高优先序订单，且积压单上首个可用的紧急通道在产线启动后的 100 件之时。我们希望以 99%的服务水平提报一个交期。

为了使用（15.21）式，我们注意到 $\mu=80$，$\sigma^2=225$，$\omega=320$，$b=100$，$c=20$，故 $m=440$。通过标准正态分布表可以查得 $z_{1-s}=z_{0.01}=-2.33$。因此

$$l=\frac{m}{\mu}+\frac{z_{1-s}^2\sigma^2\left[1+\sqrt{4\mu m/\left(z_{1-s}^2\sigma^2\right)+1}\right]}{2\mu^2}$$

$$=\frac{440}{80}+\frac{(-2.33)^2(225)\left[1+\sqrt{4(80)(440)/\left[(-2.33)^2(225)\right]+1}\right]}{2(80^2)}$$

$$=6.62$$

所以我们向客户提报七天。

注意到完成订单的平均时间是 $m/\mu=5.5$ 天。附加的一天半代表**安全提前期（safety lead time）**，用作抵消生产过程中变动性的缓冲。

图 15.3 显示了提前期提报作为总积压 m 的函数的关系。虚线表示平均完成时间 m/μ，是在生产速率恒定不变时的提报量。实线与虚线之间的差值就是安全提前期，可以注意到它随积压水平升高而延长。原因在于满足订单所需完成的工作越多，完成时间的变动性就越大，故而所需的安全提前期就越长。

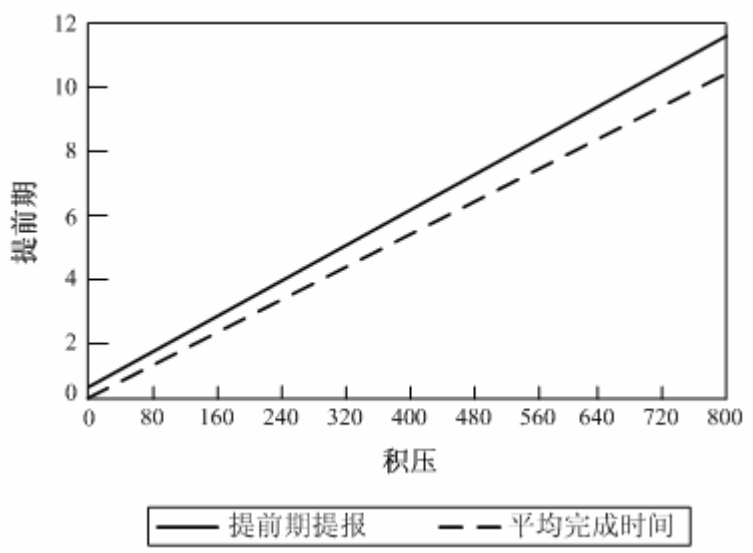

图 15.3 提报的提前期与积压

在有着多条 CONWIP 路线的环境中，可以对各条路线进行类似的一系列计算。所需的数据仅仅是该路线生产速率的前两阶条件（即，均值和方差），当前 WIP 水平（在 CONWIP 下是常数），以及积压清单的当前状态。这些数据应保持在一个向销售和制造人员开放的中心位置。销售人员需要该信息来提报交期，制造人员则用它来决定接下来生产什么。制造人员也可以按销售人员建立的积压单来追踪生产（如，第十四章描述的统计产出控制）。总的结果就是，交期有竞争力，可实现，并且与相关的制造参数保持一致。（512|513）

15.4 瓶颈处的排程

排程研究文献的一个主要结论就是：排程问题，尤其是现实规模的，其求解非常困难。所以人们常常用分解的方法将其简化。措施之一就是将瓶颈制程按其自身排程，再将这个排程传递到其他非瓶颈工站。这种方法对于简单的流水线非常有效。然而瓶颈排程在复杂得多的排程情形中也可能是重要的组成部分。

将注意力限制于瓶颈就能简化排程的一个主要原因是，它可以将多机问题简化为单机问题。从过去的讨论中可以回忆起来，与详尽排程相对的简单排序就常常足以解决单机问题。由于排程（*schedule*）给出各个加工任务何时在各台机器上作业的信息，而排序（*sequence*）仅给出加工任务的作业顺序，排序的计算就更简单。还有，排程随着时间的流逝而愈发不准确，排序则更为稳健。

若制造环境由 CONWIP 产线构成，排程问题还可以被进一步简化。如我们所知（第十三章），CONWIP 产线可以用速率为 r_b^p（实际生产速率）、通过时间为 T_0^p（最小实际提前期）的传送带刻画。由于参数 r_b^p 和 T_0^p 已调整到包括了变动性引起的效果如失效、变动的加

工时间和换模等，并且安全产能（加班）也用于确保产线在各期（天、周或其他时长）达到它的目标速率，确定性的传送带模型就成为随机的生产系统的良好近似估计。故而，在 CONWIP 产线中通过聚焦于瓶颈，我们可以有效地将非常困难的多工站随机性排程问题简化为简单得多的单工站确定性排程问题。再者，由于在各个工站处使用先入系统先出（FISFO）的分派准则，将瓶颈序列传递到其他工站就是小事一桩了——只需在所有的工站使用同样的序列。这个序列就是我们在先前的章节中提到的 **CONWIP 积压单（CONWIP backlog）**。在这一节中，我们来讨论如何生成这份清单。

15.4.1 无换模的 CONWIP 产线

我们始于 CONWIP 产线最简单的情形——排程不考虑换模。这可能是因为任何工件之间的切换不需要明显的换模；或者因为换模是周期性的（如，为了清洁或维护），与作业顺序无关。为一条无换模的 CONWIP 产线排序，其做法就像前述的按交期为单机排程，因此可以按最早交期（earliest due date，EDD）准则进行。排程理论显示，只要可能，EDD 排序就将按时完成所有的加工任务。当然了，这个结论实际上是说加工任务将在计划的（*planned*）排程内按时完成。但由于可能出现随机事件，我们不能预知它是否真的能实现。尽管如此，始于可行的计划比始于不可行的计划，能给我们在实践中达到优良绩效更多机会。

一种稍微复杂的情形是两条或更多 CONWIP 产线共享一个或更多工站。图 15.4 显示了此类（1）两条 CONWIP 产线共享一台碰巧是瓶颈的机器以及（2）产线为组装作业生产组件的情形。（513|514）我们要讨论这个例子是因为它清楚地图示了相关议题。但这里的排程从根本上来说，仍与产线补给独立制成品库存（FGI）的系统中的排程相同。

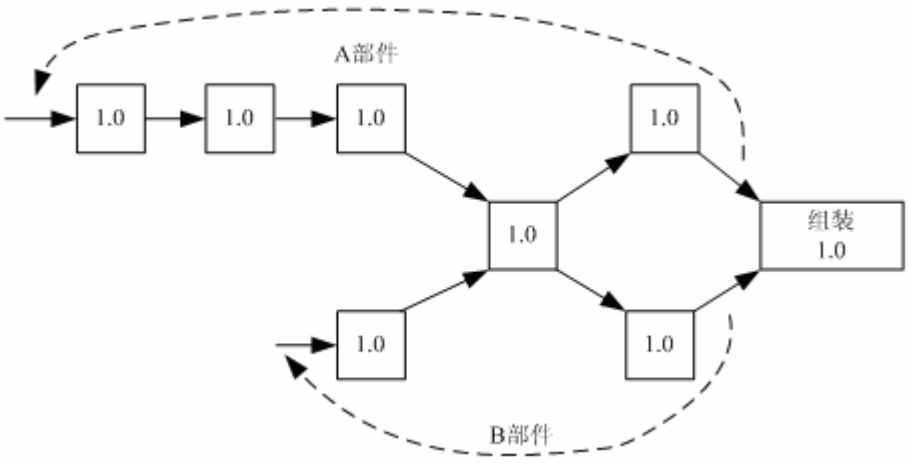

图 15.4 *共享一个加工中心的两条 CONWIP 产线*

对于两种情形，我们都应当依据 EDD 准则对单个产线的投料进行排序，并在所有的非共享工站应用这个排序，如我们对独立的 CONWIP 产线所做的那样。这时问题出现了，共享的工站该使用什么序列？

人们出于直觉可能会认为先进先出（FIFO）能运行良好。可是，如果加工时间有变动性，那么，例如就可能逐渐有一串 A 在与之匹配的 B 之前到达共享资源。这时应用 FIFO 将因此仅仅在组装作业之前创建一个未匹配工件的队列。更有极端，这么多的 WIP 停滞在组装处，将真正导致瓶颈因缺乏工件而饥饿。

较好的选择是在共享资源处应用先入系统先出（first-in-system-first-out，FISFO）分派

准则。在这种准则之下，加工任务依照进入系统的时间（即，它们的 CONWIP 卡片授权自身投放的时间）排序。由于 CONWIP 卡片授权在组装作业处相匹配的工件（即，一个 A 和一个 B）同时投放，这条准则有益于按组装序列对共享机器进行排序。因此它有益于协调工件尽可能紧密地到达组装作业。当然了，当共享的机器处没有 B 可用时（可能是由于上游不同寻常的漫长加工时间）它将仅加工 A。但一旦接收到 B，它将启动组装。

15.4.2 有换模的单 CONWIP 产线

当考虑瓶颈处有换模的 CONWIP 产线时，情况就复杂起来了。事实上，即使想确定满足所有交期的排序是否存在，就是在回答一个 NP 难问题了。

为了了解这个问题的难度并提供一种解法，我们考虑表 15.5 所示的 16 个加工任务的组合。每个加工任务需要一小时来完成，不包括换模。换模发生于从一个族切换到另一个族时，每次花费四小时。表 15.5 中的加工任务按 EDD 显示。我们可以看到，EDD 在这里并不是很有效，因为它引发了 10 次换模和 12 个拖延的加工任务，平均拖延为 10.4 小时。要寻找一个更好的解，我们显然不会考察每一种可能性，因为有 16！= 2×10^13 种可能的排序。我们转而寻找一种能给出优良解的探索式方法。（514|515）

表 15.5 *EDD* 排序

任务号	族号	交期	完成时间	延迟
1	1	5	5	0
2	1	6	6	0
3	1	10	7	-3
4	2	13	12	-1
5	1	15	17	2
6	2	15	22	7
7	1	22	27	5
8	2	22	32	10
9	1	23	37	14
10	3	29	42	13
11	2	30	47	17
12	2	31	48	17
13	3	32	53	21
14	3	32	54	22
15	3	33	55	22
16	3	40	56	16

一种可能的方法是**贪婪算法（greedy algorithm）**。贪婪算法的每一步都考虑所有的简单选项（即，序列中的加工任务两两互换）并选择对排程的改进最多的那个。它因此被称作贪婪。可能的交换组合的数量（这里是 120 个）远小于排序的总数量，所以这种方法能很快找到一个解。问题当然就是，这个解的质量如何？我们接下来对此进行讨论。

检查交换任意两个加工任务后的总拖延可以发现，将加工任务 4 放在加工任务 5 之后带来的降幅最大。如表 15.6 所示，它消除了两次换模（从族 1 到 2，再从族 2 到族 1）。现在有 8 次换模，平均拖延为 5.0。

表 15.6 贪婪算法第一次交换之后的排序

任务号	族号	交期	完成时间	延迟
1	1	5	5	0
2	1	6	6	0
3	1	10	7	-3
5	1	15	8	-7
4	2	13	13	0
6	2	15	14	-1
7	1	22	19	-3
8	2	22	24	2
9	1	23	29	6
10	3	29	34	5
11	2	30	39	9
12	2	31	40	9
13	3	32	45	13
14	3	32	46	14
15	3	33	47	14
16	3	40	48	8

在该算法的第二步，我们重复这种程序。这一次，总拖延的最大降幅发生于加工任务 7 移至加工任务 8 之后。再一次地，它通过归并同一族而消除了两次换模。现在有 6 次换模，平均拖延降低到 1.2 小时。第三步将 10 移至 12 之后，消除一次换模，将平均拖延降到 1.5 小时。作为结果的序列如表 15.7 所示。

表 15.7 贪婪算法的最终解答

任务号	族号	交期	完成时间	延迟
1	1	5	5	0
2	1	6	6	0
3	1	10	7	-3
5	1	15	8	-7
4	2	13	13	0
6	2	15	14	-1
8	2	22	15	-7
7	1	22	20	-2
9	1	23	21	-2
11	2	30	26	-4
12	2	31	27	-4
10	3	29	32	3
13	3	32	33	1
14	3	32	34	2
15	3	33	35	2
16	3	40	36	-4

到了这时，再没有简单的互换可以降低总拖延。故而贪婪算法终止，给出一个有着三个加工任务拖延的序列。现在的问题是，我们是否能做到更好？

回答是肯定的，证据就如给出了一个可行序列的表 15.8。但我们是否必须评估所有 16! 种可能序列来找到它？从数学的角度讲，必须遍历。然而，从实际的角度讲，我们常常可以用一种比单纯的贪婪算法稍微聪明些的方法，找到一个更优（且可行）的序列。

表 15.8 一个可行序列

任务号	族号	交期	完成时间	延迟
1	1	5	5	0
2	1	6	6	0
3	1	10	7	-3
5	1	15	8	-7
4	2	13	13	0
6	2	15	14	-1
8	2	22	15	-7
11	2	30	16	-14
12	2	31	17	-14
7	1	22	22	0
9	1	23	23	0
10	3	29	28	-1
13	3	32	29	-3
14	3	32	30	-2
15	3	33	31	-2
16	3	40	32	-8

为了开发这样的程序，我们注意到贪婪算法的问题在于它们能迅速到达**局部最优（local optimum）**——一个比任何邻近值都优的解，却不一定比非邻近值优。由于贪婪算法只考虑邻近移动（两两互换），它很容易停滞于某处局部最优。（515|516）这种情况很可能发生，因为像本例这样的 NP 难问题有许多局部优解。因此，我们需要的是一种机制，能迫使算法离开局部最优从而考察是否在远处还有更好的序列。（516|517）

一种措施是阻止（制造“禁忌”）某些近期考虑过的互换移动。这种方法称作**禁忌搜索（tabu search）**（见 Glover 1990），近期（现在被禁止）移动的清单称作**禁忌列表（tabu list）**。在实际中，刻画互换移动的方式很多。一种明显（尽管低效）的选择就是整个序列。在这种情况下，某些序列一旦被评估后就变成禁忌。但由于序列太多，禁忌列表得非常长才能有效。另一种效率更高却有失精确的选择是加工任务在序列中的位置。这时，将加工任务 4 移至 5 之后的移动（如演示程序的第一步）在执行一次后即变成禁忌。但由于我们仅需暂时地阻止这个移动从而避免算法落入局部最优，禁忌列表的长度也是受限的。一旦某个禁忌移动在列表中驻留了足够长的时间，它就被解禁并可以重新加入考虑。

禁忌搜索还可以通过不考虑那些明知无益的移动来进一步精炼。例如，在上例中我们知道在族内（即，换模之间）做任何违反 EDD 的改变都会使结果变糟。比如说，我们不会考虑将加工任务 1 和 2 对调，因为它们同属一个族，且 1 的交期早于 2。这种类型的规定减少了必须考虑的移动数量，并因此加速了算法。

禁忌搜索原理简单，但实现起来可能也很复杂（详尽的讨论见 Woodruff 和 Spearman 1992）。同样地，还有其他许多种探索式方法可用于排序和排程问题。研究者一直在发展新方法，并评估哪一个对于给定的问题效果最好。关于探寻式排程方法，参见 Morton 和 Pentico（1994）以及 Pinedo（1995）的工作。（517|518）

15.4.3 瓶颈排程的结论

本节的一个重要结论是，排程问题不必像排程理论对问题复杂性的狭隘解读那样无望。通过简化环境（如，使用 CONWIP 产线）以及使用精心选择的探索式方法，管理人员可以找到足够有效的排成程序。

对于 CONWIP 产线这样的拉式系统，简单的排序就够了，因为投料的时机由系统进程控制。如果无换模，一个 EDD 序列就是单条 CONWIP 产线的最好选择。只要没有显著的换模及在共享资源处应用 FISFO 准则，它也适用于有共享资源的多条 CONWIP 产线。如果有显著的换模，则一个简单序列对于 CONWIP 产线仍然足够，但它却不是 EDD 的。如禁忌搜索之类的探索式方法，可为这种情形寻找优良的解答。

15.5 排程诊断

很不幸，并非所有的排程情形都可以归入简单的瓶颈排序。在某些系统中，由于产品组合变化或使用浮动劳动力带来产能经常变化，瓶颈的位置会漂移。在某些工厂中，极端复杂的路线使得 CONWIP 或其他拉式系统无法使用。还有一些情况中，系统中的 WIP 被指定给不同的客户以响应不断变化的需求。

一个对待这些情形的近似诡辩的建议就是消除它们。对于那些具备可行性的系统，它可能是最佳的行动纲领。然而，对于其他的系统，它其实在物理上或经济上不可行。这时，大多数企业转而求助 MRP 的某个变种。从概念上讲，MRP 可用于几乎任何制造环境。但如我们在第三章和第五章中所讲，基本的 MRP 模型在潜在假设方面有重大缺陷，尤其是无限产能的假设。作为回应，生产研究者和软件提供商将越来越多的注意力放在有限产能排程工具上。再次如前面所讲的，这个方法依赖 MRP 投料计划作为输入，所以太不足道，太迟了。本节的目标就是在保持 ERP 层级的同时移除 MRP 排程模型的缺陷。

在现实世界中，有效排程远非仅为数学问题寻找优良的解。以下是两个要点：

1. *模型依赖于数据，而数据是估计的。*许多排程模型都要用到的一个公共参数是拖延造成的成本（tardiness cost），它用于决定客户服务水平和库存成本的权衡。可是，我们在产业界几乎没有遇到一个人乐意在见到它对排程的影响之前设定其值。

2. *许多无形事项都无法通过模型说明。*客户的特殊考虑、车间条件的变更、与供应商和分包商关系的发展等等都使得完全自动化的排程不可能实现。结果就是，我们与之有过交谈的大多数排程专家都认为有效的排程必须要允许人的介入。为了有效地利用人的智能，这类系统应当评估给定排程的可行性（而非最优性），如果可行就提出变更建议。建议可能包括通过加班、临时工或分包来增加产能，向后推某些加工任务的交期，以及分拆大的加工任务。（518|519）人的判断力要用于在它们之间做出明智选择，从而解决诸如此类的问题，哪些客户能容忍延迟或部分送达，哪些工件现在可以分包出去，哪些群组的工人可以或不可以被要求加班？

基于优化的或基于仿真的方法都不是很适合于评估备选排程及给出改进建议。可能正是

由于这一点，一项对排程软件的调查发现所有系统的诊断能力都很弱（no systems with more than trivial diagnostic capability）（Arguello 1994）。

与之相反，ERP 范式旨在开发和评估（*evaluate*）生产排程。主生产计划（MPS）提供需求；物料需求计划（MRP）计算净需求（net demand），决定物料需求，并通过偏移生成投料计划；产能需求计划（CRP）检查这个计划的可行性。作为一个计划体系，它完美地适合于现实世界的生产控制。但是如我们在前面讨论过的，MRP 中的基本模型太简单而无法准确表示工厂的实况。与之类似，CRP 对 MRP 的检查也不准确，因为 CRP 也受到同样的建模缺陷（固定的提前期）的损害。即使 CRP 能准确检查计划的可行性，它也没有为如何改造不可行性提供有用的诊断意见。

故而，我们的目标是提供这样一种排程程序，它保留适用的 ERP 体系却消除 MRP 的建模缺陷。在本节，我们要讨论不可行如何以及为何出现，并提供检测和纠正的措施。

15.5.1 排程不可行的类型

排程不可行有两种基本类型。**WIP 不可行（WIP infeasibility）**由 WIP 的不合理位置引起。如果系统没有足够的 WIP 来满足最近的交期（to facilitate fulfillment of near term due dates），不管产能如何该排程都将不可行。**产能不可行（capacity infeasibility）**由产能不足引起。产能不可行可以通过推后需求（push out demand）或增加产能来补救。

例子：

我们通过考察这样一条产线说明排程不可行的类型和效果。其验证的产能为 r_b^P = 100 件/天，实际最短加工时间 T_0^P = 3 天。所以，根据里特定律，平均 WIP 为 300 件。当前，有 95 件预期在第一天末完成，90 件应当在第二天末完成，还有 115 件刚开始加工。在最后的 115 件中，100 件将在第三天末完成，余下的 15 件由于产能约束将在第四天完成。需求在表 15.9 中给出，开始很低后来超过了产能。

表 15.9 诊断器材例子中的需求

开始后的时间	到期数量
1	90
2	100
3	90
4	80
5	70
6	130
7	120
8	110
9	110
10	110
11	100
12	90
13	90
14	90
15	90

首先注意到前三天的总需求为 280 件，而 WIP 和产能（每件加工任务为一个单位）为 300 件。接下来 12 天的需求为 1,190 件，而有 1,200 件的产能以及满足前三天还余下的 20 件 WIP。所以，从整体迅速一撇，满足需求似乎是可行的。

然而，看得更仔细些就会发现一个问题。第一天末产线将输出 95 件来满足 90 件的需求，余下 5 件制成品库存（FGI）。第二天末产线再输出 90 件，但该日需求为 100 件。（519|520）即使用上第一天剩下的 5 件 FGI，仍有 5 件的缺口。在第三天末输出 100 件来满足 90 件的需求，有 10 件的过剩。它可用于补救第二天的缺口，但仅仅发生在我们愿意延迟一天送达的情况下。

第二天出现赤字的原因是系统内在两天内完成的 WIP 不足以满足前两天的需求。第一天和第二天的总需求为 90 + 100 = 190 件，而到第二天末仅有 95 + 90 = 185 件 WIP 能完成并输出。因此，不管产能有多大，总有 5 件的赤字。注意到由于它没有涉及产能，MRP 可以检查到这类不可行性。

再看第三天之后的需求，我们发现还有其他问题。图 15.5 显示了产线最大累计产量与累计需求。只要当最大累计产量低于累计需求时，排程就不可行。刻度在右侧的过剩线，就是最大累计产量与累计需求的差值。负值就意味着不可行。该曲线在第二天首先变成负的——产线 WIP 不足引起的不可行。在那之后，产线的输出大于需求，过剩线变成正的。它在第八天再次变成负的，原因是需求开始超过产能，并且保持到产能逐渐跟上的第十四天。

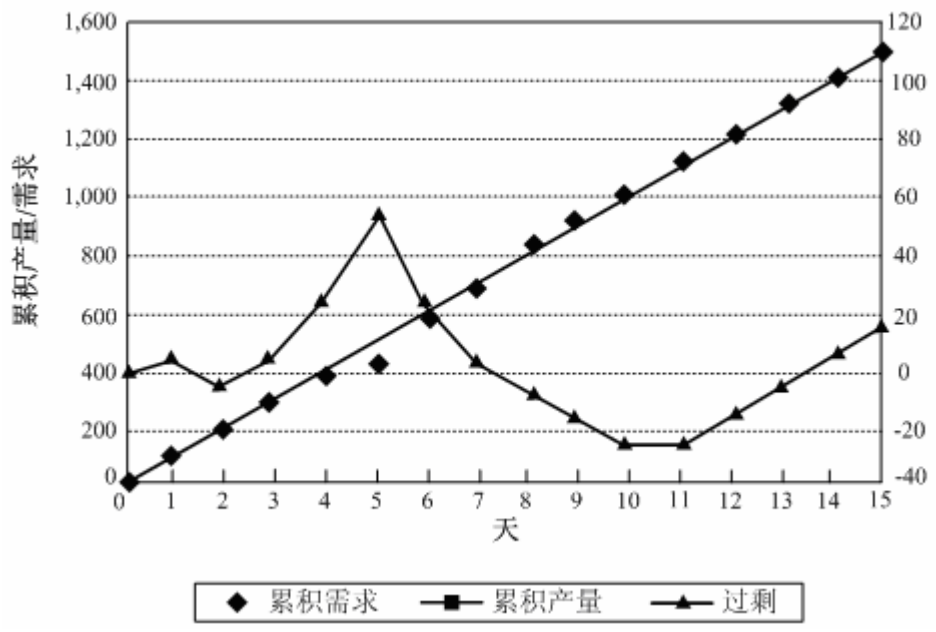

图 15.5 需求与可得产量和 *WIP* 之间的关系

第八天的不可行不同于第二天，因为它是产能的函数。第二天没有额外的产能可以使产线满足需求，而第八天之前的额外 25 件可以使它满足需求。因此发生于第八天的不可行是个产能不可行的实例。由于 MRP 和 CRP 都基于无限产能模型，它们不能检测到这类不可行。（520|521）

两类不同的不可行需要不同的补救措施。增加产能无益于 WIP 不可行，所以唯一的办法是推后交期。例如，若原定于第二天的 100 件中能有 5 件推后到第三天，该部分的排程就变得可行了。

产能不可能有两种补救措施：增加产能或推后交期。例如，若在第八天利用加班生产 25 件输出，排程就可行了。可是，这样做也会在计划展望期的末尾增加过剩（见 图 15.6）。还有另一种选择，如果第六天的 130 件中有 30 件移至第十二天、第十三天和第十四天（各 10 件），排程也变得可行（见 图 15.7）。这样做之后计划展望期末尾的过剩会比前述加班的少，因为没有增加产能。当然了，在实际的排程情形中我们也还要纠正这些过剩，具体途径在下节给出。（521|522）

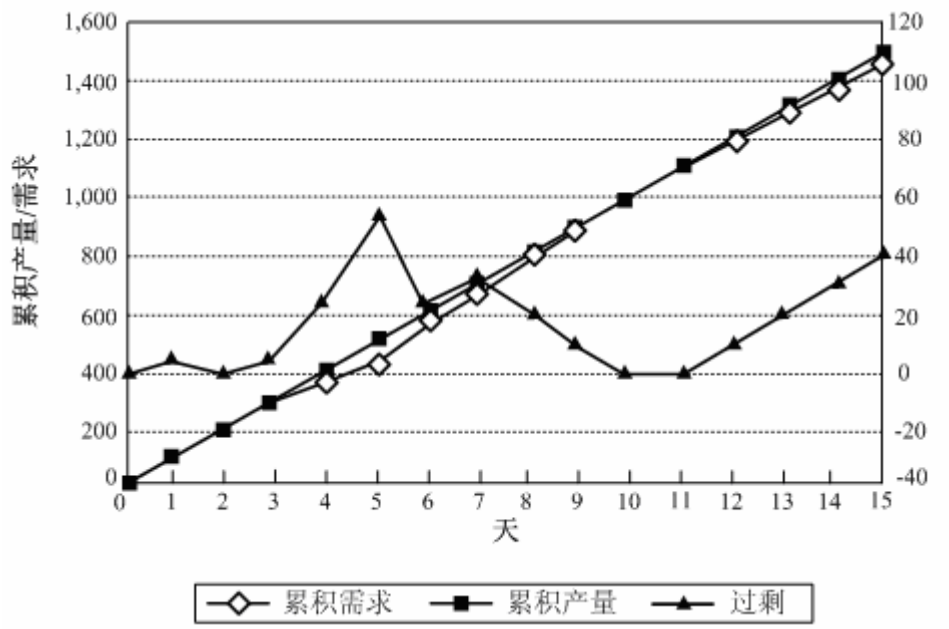

图 15.6 产能提升后需求与可得产量和 *WIP* 之间的关系

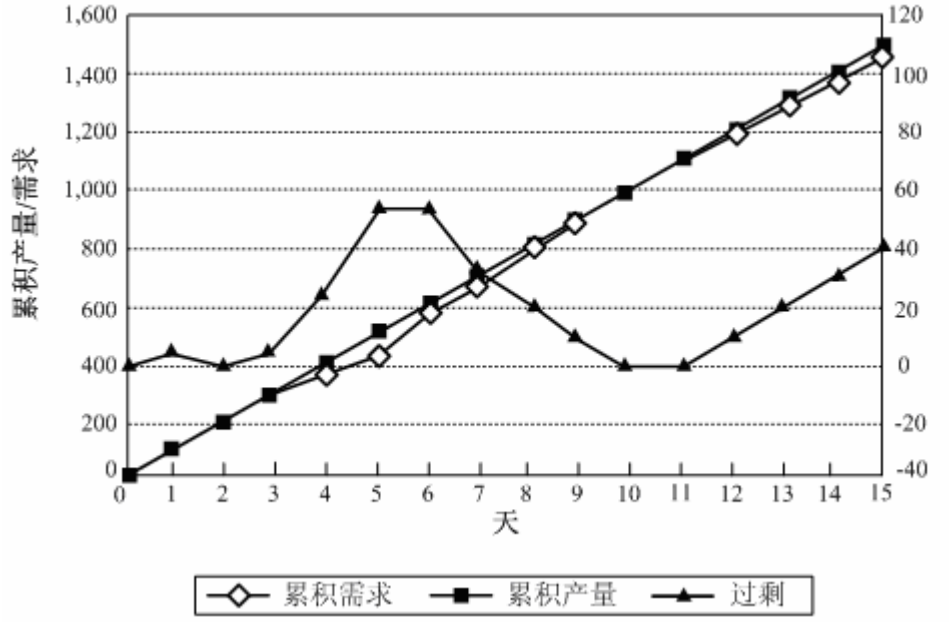

图 15.7 推后需求后需求与可得产量和 *WIP* 之间的关系

15.5.2 产能限制的物料需求计划--MRP-C

一种设计来检测和弥补排程不可行的程序是**产能限制的物料需求计划（capacitated material requirements planning，MRP-C）**（细节参见 Tafdif 1995）。MRP-C 类似 MRP，不同之处在于它明确地考虑产能。这样，它就在 MRPⅡ计划层级中替代了 MRP。

MRP-C 的基本结构取自生产计划的层级本性。如我们在第十三章所见，高层问题的决策变量常常是低层问题的约束条件。例如，集结计划可能将产能变量（如，加班）视作变量，需求管理可能将客户交期视作变量（若使用了交期提报），但排程就常常将产能和交期作为约束条件。如我们所见，这些约束引发了生产研究文献中某些最难的问题（如，最小化加工车间生产期问题）。

为了使排程问题易于处理，MRP-C 始于寻找一个满足所有交期而不超越产能，在需用之前建立尽可能少库存的低层排程。我们将**预建库存（build-ahead）**定义为在实际需求之前制造出来的库存。MRP-C 的目标即是寻找一个最小化预建库存的可行排程。如果找不到，MRP-C 就要强调不可行的原因，使计划者能依据具体情况对交期、产能或以上二者做出变更。

MRP-C 使用的算法基于刻画系统中每个制程（取决于所需细节的层级而是机器、线段或产线）行为的传送带模型。需要估计以下两个参数：

1. **最小实际提前期（Mimimum practical lead time）**标记为 T_0^P，表示无排队时通过产线的时间。它应当包括任何一般性的延迟，如等待移动、微小调整等，并因而常常大于第七章中的原始加工时间（raw process time）T_0。

2. **实际生产速率（practical production rate）**标记为 $r_b^P(t)$，表示产线在时间段 t 内的实际产能。如果 $r_b^P(t)$ 对于所有的 t 恒定，则由于瓶颈利用率必须小于 100%，$r_b^P(t)$ 必须小于第七章中的瓶颈速率 r_b。然而，如果 $r_b^P(t)$ 可变（如，依赖于计划的加班），则 $r_b^P(t)$ 可能在某些 t 时会超过 r_b。即使这样，在均值上它也必须小于长期运行时的瓶颈速率。（522|523）

仅使用这两个数值，MRP-C 就抓住了全部制程中 WIP 和周期时间的基本关系，而不必依赖全面开花的仿真或每个工站处的详细产能知识。

MRP-C 分为两个阶段。首先比较近期需求和产线已有的 WIP 以及可用的产能，看是否存在不可行性。一旦所有的不可行都被解决，第二阶段进行时间回溯从而确定满足需求所需的最少投料。我们在以下的技术性注释中描述 MRP-C 阶段Ⅰ的机制。

技术性注释：MRP-C（阶段Ⅰ）

该程序将时间划分为时期，依所需拆分的层次可以是班次、天或周。使用以下的记号：

T = 问题的计划展望期（最后一期）

t = 时期的标示，$t = 0, 1, ..., T$

T_0^P = 所考虑制程的最小实际提前期

l = 用于获取原材料的提前期

$r_b^P(t)$ = 第 *t* 期的生产速率（产能）

$D(t)$ = 第 *t* 期的需求，即主生产计划

$a(t)$ = 第 *t* 期原材料的计划接收量（到达）

$\omega(t)$ = 产线中离完成还有 *t* 期的“定时可用”的 WIP（timed-available WIP，TAWIP），*t* = 0, 1, ..., T_P。注意到$\omega(0)$表示已完成的 WIP，现在成为 FGI 或其他制程的原材料。

$\hat{\omega}(t)$ = 按产能调整的定时可用的 WIP（CATAWIP），它考虑了时期 *t* 的可用产能量

$c(t)$ = 第 *t* 期的传出（carryover）WIP

$I(t)$ = 第 *t* 期的计划 FGI 量

$N(t)$ = 第 *t* 期的净 FGI 需求量

第一阶段计算净 WIP 需求量$N(t)$如下：

1. 确定 TAWIP，其形式可能是产线现存的 WIP 或计划接收量。我们设定

$$\omega(t)=\begin{cases}\text{现存的}WIP & 1\le t\le T_0^P\\ a\left(t-T_0^P\right) & T_0^P<t\le T_0^P+l\\ \infty & t>T_0^P+l\end{cases}$$

在实际最小加工时间之内的时期，$\omega(t)$等于产线中现存的 WIP。在前面的T_0^P = 3 天的例子中，$\omega(1)$已在产线中两天了，离完成只有一天；$\omega(2)$已在产线中一天，因此还需要两天来完成；以此类推。*t* 值大于T_0^P而小于用于取得原材料的时间时，定时可用的 WIP 等于时期之前接收的原材料。对于超过原材料提前期（l）加上加工时间（T_0^P）的时期，该值设定为无限因为物料可在它们的提前期内获取。

2. 计算 CATAWIP。我们始于$c(0)$= 0 并计算（523|524）

$$\hat{\omega}(t)=\min\left\{r_b^P(t),\omega(t)+c(t-1)\right\}$$

$$c(t)=\omega(t)+c(t-1)-\hat{\omega}(t)$$

t = 0, 1, ..., T。这一步解释了由于产能约束，第 *t* 期定时可用的 WIP 最终在第 *t* 期完成的将少于r_b^P。所以$\hat{\omega}(t)$表示各期以全部产能运行时的产量。如果第 *t* 期的可用 WIP 大于产能，

它就作为$c(t)$传出并在第 *t*+1 期可用。

3. 计算预期 FGI。我们将$I(0)$定为原始的 FGI，并计算

$$I(t)=I(t-1)+\hat{\omega}(t)-D(t)$$

t = 0, 1, ..., T。这一步使用最大量产能/原材料，计算出各期末的净 FGI。如果这个值一旦变成负的，就意味着没有足以满足需求的 WIP 和/或产能。

4. 计算净需求。计算

$$N(t)=\max\{0,\min\{-I(t),D(t)\}\}$$

t = 0, 1, ..., T。若$I(t)$为正，则没有净需求，因为有足够的库存来满足毛需求；若$I(t)$为负而$I(t-1)\geq 0$，则净需求等于$I(t)$的绝对值；若$I(t)$和$I(t-1)$都为负，则$N(t)$就等于当期需求。注意到这非常像常规 MRP 中的净值计算。

如果$N(t)>0$且$c(t)<N(t)$，这个排程 WIP 不可行，唯一的补救措施是移出$N(t)-c(t)$单位的需求。如果$N(t)>0$且$c(t)\geq N(t)$，则问题就是产能不可行，补救措施可以是移出需求或增加产能。

5. 采取任何变更（如，推后交期）后，所有的值都必须重新计算。

以上的 MRP-C 程序貌似很复杂，实际上它在电子表格中非常直观。以下的例子给出了具体说明。

例子：

将 MRP-C 程序应用于前述例子中可以得到如表 15.10 所示的结果。第二期 5 个单位的 WIP 不可行性由$N(2)$= 5 揭示出来。唯一的解决办法是将该期需求从 100 削减到 95，并通过传出使第三期需求从 90 增加到 95。$N(t)$在第十、十一期达到 25 的现象，指出产能赤字为 25。解决办法是在第八期加班来多产 25 件（表 15.11）；或者在没有可用的额外产能时，通过推后交期将 25 件的产量放在排程较后的位置。计划库存量指出哪些时期还有多余的产能和/或 WIP，还可以接收额外的需求。最终排程如表 15.11 所示。

这时，我们知道可行的排程是存在的。可是所生成的主生产计划并不太好，因为它有些时期的需求超过了产能。因此，必须有预建库存。它从排程的最后一期算起，并进行时间回溯。以下的技术性注释给出细节。（524|525）（525|526）

表 15.10 可行性计算

时期 t	需求 $D(t)$	TAWIP $w(t)$	产能 $rbP(t)$	CATAWIP $\hat{w}(t)$	传出WIP $c(t)$	计划库存量 $I(t)$	净需求 $N(t)$
0					0	0	
1	90	95	100	95	0	5	0
2	100	90	100	90	0	-5	5
3	90	115	100	100	15	5	0
4	80	∞	100	100	∞	25	0
5	70	∞	100	100	∞	55	0
6	130	∞	100	100	∞	25	0
7	120	∞	100	100	∞	5	0
8	110	∞	100	100	∞	-5	5
9	110	∞	100	100	∞	-15	15
10	110	∞	100	100	∞	-25	25
11	100	∞	100	100	∞	-25	25
12	90	∞	100	100	∞	-15	15
13	90	∞	100	100	∞	-5	5
14	90	∞	100	100	∞	5	0
15	90	∞	100	100	∞	15	0

表 15.11 最终的可行主生产计划

时期 t	需求 $D(t)$	TAWIP $w(t)$	产能 $rbP(t)$	CATAWIP $\hat{w}(t)$	传出WIP $c(t)$	计划库存量 $I(t)$	净需求 $N(t)$
0					0	0	
1	90	95	100	95	0	5	0
2	95	90	100	90	0	0	0
3	90	115	100	100	15	10	0
4	85	∞	100	100	∞	25	0
5	70	∞	100	100	∞	55	0
6	130	∞	100	100	∞	25	0
7	120	∞	100	100	∞	5	0
8	110	∞	125	125	∞	20	0
9	110	∞	100	100	∞	10	0
10	110	∞	100	100	∞	0	0
11	100	∞	100	100	∞	0	0
12	90	∞	100	100	∞	10	0
13	90	∞	100	100	∞	20	0
14	90	∞	100	100	∞	30	0
15	90	∞	100	100	∞	40	0

技术性注释：MRP-C（阶段Ⅱ）

为了刻画将（可行的）需求计划转化成投料（起始）计划的 MRP-C 程序，我们使用以下记号：

$D(t)$ = 第 *t* 期的需求，即主生产计划

$I(t)$ = 第 *t* 期的计划 FGI 量

$N(t)$ = 第 *t* 期的净 FGI 需求量

$\hat{\omega}(t)$ = 第 *t* 期的 CATAWIP

$X(t)$ = 第 *t* 期的产量

$Y(t)$ = 第 *t* 期的预建库存，即产于第 *t* 期而用于满足第 *t* 期之后需求的数量

$S(t)$ = 第 *t* 期的投料（“起始”）数量

基本程序是首先减去 FGI 得到净需求，其方法和 MRP 差不多；然后各期的可用产量由按产能调整的定时可用的 WIP（CATAWIP）给出。由于它包括产线中的 WIP，我们不必像 MRP 那样将其净去。就这样计算各期的产量、预建库存以及投料数量。

具体步骤如下：

1. *净值计算*（*netting*）。首先以标准（MRP）方式计算净需求。

a. 变量初始化：

$$I(0) = \text{原初的 FGI}$$

$$N(0) = 0$$

b. 从第一期到第 T 期，计算计划可用库存量和净需求：

$$I(t) = I(t-1) + \hat{\omega}(t) - D(t)$$

$$N(t) = \max\{0, \min\{-I(t), D(t)\}\}$$

2. *排程*（*scheduling*）。排程程序始于最后一期（第 T 期），再进行时间回溯。

a. 变量初始化：

$$D(T+1) = 0$$

$$X(T+1) = 0$$

$$Y(T+1) = \text{期望的终期 FGI 水平}$$

b. 从第T期开始回溯到第一期，计算

$$Y(t)=Y(t+1)+D(t+1)-X(t+1)$$

$$X(t)=\min\{\hat{\omega}(t),D(t)+Y(t)\}$$

c. 表达式$Y(0)=Y(1)+D(1)-X(1)$给出一种简易的产能检查方法。若阶段Ⅰ解决了所有的不可行性，这个值应当为零。如果不为零，该排程就不可行，阶段Ⅰ要重做。

d. 假设不再有不可行性，我们按最小实际提前期偏移产量来制定生产计划：（526|527）

$$S(t)=X\left(t+T_0^P\right),\qquad t=0,1,\ldots,\ T-T_0^P$$

MRP-C 排程从计划期的末端向前计算预建库存的数量。第T期的预建库存水平就是期望的计划期末库存水平。人们一般将其设为零，除非有特殊原因而愿意以过量库存结束计划期。在每一期，输出数量或者等于产能或者等于总需求（净需求加预建库存），并总是二者之中较小的那个。这很直观，因为产量不可能超过产线的最大速率，也不应该超过需求（包括预建库存）。

如果第零期的预建库存为正，则排程不可行。第零期的预建库存量就是在 t = 0 时为使排程可行而必需的额外 FGI。但是如果阶段Ⅰ解决了所有的产能和 WIP 不可行问题，$Y(0)$将等于零。事实上，这正是阶段Ⅰ的整个要点。

MRP-C 程序的最终输出是一个在产能和物料约束内满足所有（可能经过修订）的交期并产生最小预建库存的生产起始清单。

例子：

我们继续阶段Ⅰ中的例子，并用上 MRP-C 的第二阶段。生成的结果如表 15.12 所示。注意到该排程要求产量尽可能地高，因为前两期受 WIP 限制，然后受产能限制，直到第十二期才平缓下来。在第十二期，产量降到 90 件，这个数字低于 CATAWIP 但仍足以跟上需求。

注意到 MRP-C 做了找出不可行性并给出可能的补救措施的粗活，但它把涉及提高产能（是否要，如何，在哪里）和延迟加工任务（哪些，程度如何）的敏锐判断留给了读者。MRP-C 通过这样做，鼓励人和计算机在排程过程中发挥各自的天分。（527|528）

15.5.3 将 MRP-C 延伸到更一般的环境

前面的内容描述了如何使用 MRP-C 对通过传送带模型表示的一个制程（工站、产线或产线的一段）进行排程。MRP-C 的真正威力在于，它可以延伸到多级、多产品的系统。

对于串联产线，MRP-C 很容易推广。将进入下游工站的生产起始（投料）也意味着对其上游工站的需求（The production starts into a downstream station represent the demands upon the upstream station that feeds it）。所以，我们可以很容易地应用 MRP-C，始于最后一个工站，一直回溯到线首。类似地，定时可用的 WIP（time-adjusted WIP，TAWIP）通过上游制程的产量生成。

如果有组装工站，则生产起始必须转化为对其补给的各工站的需求。这非常像 MRP 的 BOM 展开，不同之处仅是它应用于各条路线。MRP-C 程序的其他方面保持不变。

表 15.12 最终的生产排程

时期 t	需求 $D(t)$	计划可用量 $I(t)$	净需求 $N(t)$	CATAWIP $\hat{w}(t)$	预建库存 $Y(t)$	当期产量 $X(t)$	投料 $S(t)$
0		0	5		0		
1	90	-90	90	95	5	95	100
2	100	-95	95	90	0	90	100
3	90	-95	95	100	5	100	100
4	80	-80	80	100	25	100	100
5	70	-70	70	100	55	100	125
6	130	-130	130	100	25	100	100
7	120	-120	120	100	5	100	100
8	110	-11	110	125	20	125	100
9	110	-110	110	100	10	100	90
10	110	-110	110	100	0	100	90
11	100	-100	100	100	0	100	90
12	90	-90	90	100	0	90	90
13	90	-90	90	100	0	90	
14	90	-90	90	100	0	90	
15	90	-90	90	100	0	90	

对于多条路线（即，生产不同的产品）通过一个工站的系统，我们必须合并独立需求（即，下游工站的生产起始）来形成一个总需求。又由于不同的产品在共享工站处可能有着不同的加工时间，MRP-C 应以时间为单位展开计算。也就是说，产能、需求、WIP 等都应以小时计量。这在实质上类似于我们在第十四章中讨论的观点，即在多产品的 CONWIP 产线中维持恒定数量的作业时间而非恒定数量的工件。

在多产品的系统中，事情就复杂一些了，因为当不止一种产品在同一时期需要使用预建库存时我们必须选择一种办法来拆开它们的关联。错误的选择可能很少或根本没有为某种产品的早期生产计划可用的 WIP，却为另一种产品计划充沛的 WIP。这样可能会导致下一阶段的计划出现 WIP 不可行。Tardif（1995）给出几种拆分关联的灵巧方法，他还阐述了实际执行中的其他问题。

15.5.4 实践中的议题

MRP-C 比起 MRP 有两个优势：（1）它使用了一个明确考虑产能故而更加精确的模型，（2）它为使用者提供了有用的诊断措施。但是，它也有问题。

首先，MRP-C 依赖探索式方法，所以不能保证一定找到存在的可行排程。（然而，如果它找到一个可行排程，则这个排程一定确实可行。）尽管某些类型的 MRP-C 可以使用确定性算法，但其速度大打折扣（见 Tardif 1995）。从本质上来说，前面一直讨论的方法为了速度而牺牲精确性。但考虑到它要用于反复进行的、“决策支持”的模式，速度的增加可能比精确性的微小牺牲更值得。此外，MRP-C 产生的任何偏差都会使排程更为保守。即，MRP-C 可能以多于最低限度的调整来达到可行性。故而，排程可能比它们所需要的“更加可行”并有着更高的成功执行的机会。

其次，就像所有的排程方法那样，MRP-C 意味着一种推式哲学（即，它设定投料时间）。

如我们在第十章的讨论，这使它遭受推式系统所有缺点的侵害。幸而，我们可以将 MRP-C（事实上任何推式系统都可以，包括 MRP）整合进入拉式环境，并获取与拉相关的效率、可预测性及稳健性的好处。我们在接下来的节中描述如何做到这一点。（528|529）

15.6 拉式环境中的生产排程

现在回忆推式与拉式生产控制的定义。推式系统基于交期计划向产线的投料，而拉式系统基于运营状态授权向产线的投料。推式系统控制投料速率（因而也有产出）并测量 WIP 来看速率过大还是过小。拉式系统的做法正好相反，它们控制 WIP 并测量产出来看产量是否足够。由于 WIP 控制比投料控制较不敏感，拉式系统面对偏差就比推式系统稳健。还有，由于拉式系统直接控制 WIP，它们能避免常见于拉式系统的 WIP 爆炸以及与之相关的超时的罪恶循环。最后，拉式系统具备短期超前作业的能力，使这些时期超常生产。

出于这些原因，我们想尽可能多地保留拉式系统的好处。现在的问题是，如何在一个需要详尽排程的环境中做到这一点？本节中我们将讨论排程和拉式生产之间的关联。

15.6.1 排程计划，拉式执行

即使最好的排程也只是对将发生的事的预测，而不能保证它们一定发生。出于必要性，排程相对于车间作业来说更新频率较低；尽管物流、机器失效等等随时在发生，排程可能一周只更新一次。因此，它们不可避免会过期，有时还会很快就过期了。所以我们应当将排程视为对向系统投料的顺序和时机的一组建议，而不是要求。

拉式系统是一种连接投料与实时状态信息的理想机制。当产线已被 WIP 阻塞时，更多的投料只会增加拥堵而无益于加工任务更早完成，而拉式系统则将阻止投料。当产线的运行速度快于预期，拉式系统将使其减缓。幸运的是，同时使用拉式系统和排程一点都不困难。

为了说明其运行机理，假设我们有个各条路线都是 CONWIP 的系统，并且使用 MRP-C 对整个系统生成排程。注意到 MRP-C 和 CONWIP 之间有个重要的关联——传送带模型。所以，如果所用的参数正确，MRP-C 生成的一组投料时间将非常接近于 CONWIP 系统生成的投料授权指令（拉的信号）。当然了，变动性常常使这种相合不那么精确，但一般来说实际绩效将与计划的排程一致。

当生产落后于排程时，如果有可用的产能缓冲（如，每班或每天末尾的补充（makeup）时间）我们就可以追赶上去。如果没有此类缓冲，我们必须在下次更新时调整排程。当生产赶过排程时，我们可以通过允许产线拉入多于计划的物料而允许它超前作业。一条简单的准则，即比较当前的日期和时间与下次投料的日期和时间，可以是 CONWIP 产线不致于超前太远。通过这种方式，CONWIP 系统可以利用“顺利”生产的时间而又不太超前于排程。（529|530）

当无法依赖产能缓冲来赶上生产延迟（如，已经在尽可能快地运行产线）时，我们可以用第十三章中讲到的统计产出控制（STC）来补足 CONWIP 控制系统。它提供了一种检测生产何时相对于排程失控的方法。若发生了失控现象，该系统与 MRP-C 二者之一需要做出调整。究竟是哪一个，则是一个重要的管理决策。削减 MRP-C 的产能参数等效于承认企业的目标实现不了。但是提高产能又需要对设备、人员投资，或者增加外包成本，或者是咨询费。

15.6.2 使用 CONWIP 和 MRP

先前关于 CONWIP 与排程相连的讨论并没有绝对要求排程必须由 MRP-C 生成。当然

了，由于 MRP-C 使用的是与 CONWIP 内在机理相同的传送带模型来考虑产能，可以预期它能运行良好。但是我们当然可以将任何排程系统与 CONWIP 联用，比如说 MRP。我们使用 MRP 生成的按路线分拣的计划投入量列表，作为各条 CONWIP 产线的作业积压单。然后 CONWIP 产线决定加工任务何时被拉入系统。

对于 MRP-C，我们可以实施产能缓冲，超前作业，以及追踪排程。基本区别在于 MRP 和 CONWIP 的内在模型不一致。结果就是，MRP 比 MRP-C 更可能生成不一致的计划投入量。不过这也可以在某种程度上得到缓解，如通过良好的主生产计划技术或使用自底向上的再计划来调试该过程。

15.7 结论

生产问题的困难已经臭名昭著，一是因为它们包含了许多相互冲突的目标，二是因为它们的内在数学关系可能会非常复杂。许多排程研究成果建立了排程问题复杂性的规范量度，也得出了一些良好的见解。可是，实际排程情形还没有良好的解。

排程很困难，所以我们讨论的一个重要结论就是，往往可能通过求解另一个问题而避免本身难解的问题。一个实例就是使用系统化的提报方法外生这些交期。另一个实例则是将保持短的周期时间的问题（要使用较小的加工任务）与保持高的产能的问题（要归组类似的加工任务来减少换模）分离。一旦问题得到合适的规范化描述，就会有可用于寻找可行（并非最优）排程的良好探索式方法。

当前关于用排程研究与软件开发的一个重要趋势是有限产能排程。通过克服 MRP 的基本缺陷，这些模型可能使 MRP Ⅱ层级在实践中更为有效。然而，为了提供对无形关系做出表述的柔性，执行有限产能排程的一个有效途径是针对系统评估排程的可行性并生成对不可行性的诊断意见。为此设计的一种程序就是产能限制的物料需求计划——MRP-C。

最后，排程尽管从本质上讲属于推式哲学，它也可以与拉式系统联用。基本观点是用排程来计划物料投放，用拉式系统来执行。这样做就提供了排程系统在计划方面的益处，以及拉式系统在环境方面的益处。