Vol. 1 No. 1 运 筹 学 杂 志 第1卷 第1期
Oct., 1982 CHINESE JOURNAL OF OPERATIONS REASEARCH 1982年10月

组 合 时 间 表 理 论

R. L. Graham 著 高 彻 译

在Acme自行车厂的装配工段，事情进行得很不顺利. 在过去的六个月里，这个工段总是完成不了定额，头头们开始来回打转。假若你是新任命的这一工段的一位段长，你将会得到指示：要设法改变这种惨状。因而你在上任的第一天，你将卷起袖子去弄清楚这一工段中所进行的一切。

你所知道的第一件事是这样的，一辆自行车的装配过程通常分成几道特定的工作来进行：

FP——装大架，包括安装前叉和挡泥板，
FW——安装和校正前轮，
BW——安装和校正后轮，
DE——在大架上装换速叉，
GC——安装齿轮串，
CW——将键轮装于曲棍，
CR——将键轮及曲棍安装于大架，
RP——装右脚蹬及趾卡，
LP——装左脚蹬及趾卡，
FA——最后安装，包括装手把、座位、刹车等。

你也知道，你的刚离开的前任对于一个熟练工人需要多少时间才能完成上述的工作，曾经搜集了大量数据。这些数据可以简要地列成下表：

工　　作	FP	FW	BW	DE	GC	CW	CR	RP	LP	FA
时间(分钟)	7	7	7	2	3	2	2	8	8	18

由于车间中空间与设备的限制，该工段中的20个装配工通常分成十组，每组2人同时装配一辆自行车。经过简单计算，装配一辆自行车总共需要64分钟。因此每组应设法在32分钟内安装一辆自行车。假若每天工作八小时，每组可以安装15辆，因而整个工段每天可以完成所给的150辆的定额，你可以尝到你下次被提升的味道。

但当你认识到自行车是不能把零件按任意次序凑合在一起装配时，你的热情就会大大缩减。某些工作必须在某些别的工作之前先做，比如说，假若你已经先将把手装到前叉，那末当你将前叉装到大架时就会感到特别麻烦。同样，曲棍在安装脚蹬之前必须装到大架上。经过与几位有经验的装配工的长时间的讨论，你可以得出下列的卡片，它说明在安装时哪些工作应在另外哪些工作之前进行。

本文1980年1月7日收到。

这件工作	必后于	这些工作
FA		FP, FW, BW, GC, DE
BW		GC, DE
GC, CW		DE
LP, RP		CR, CW, GC
CR		CW

在工作时间表中除了这些机械上的限制之外，还有两条上级领导要求在工作中必须遵守的规则：

规则 1　工人在有工作可做时不能闲着；

规则 2　装配工在开始干一件工作后，他必须继续将该件工作完成为止。

在 Acme 自行车厂习惯采用的安装顺序如下表所示：

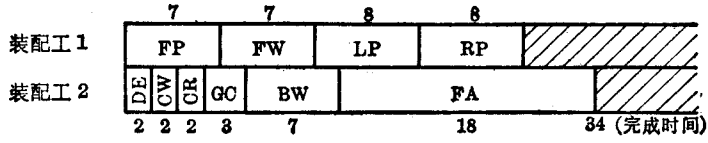


图 1　标准的装配时间表

从上表可以看出，要全部装好一辆自行车至少需要 34 分钟。虽然这一时间表遵从上述所要求的装配次序的限制，但是每组每天只能完成 14 辆多一点。因而该工段的总产量是每天比 140 辆正好多一点，大大低于 150 辆的定额。

在浪费了大量纸张来寻求种种不同的时间表而又未得到成功之后，你就匆忙决定去租用一些全电动工具供给装配工人使用。这些工具使每道工作正好减少了一分钟，也就是说安装一辆自行车总需时 54 分。你带着多少幸运的心情希望有可能将产量提高到每组每天能出产 18 辆。但是在使用租来的工具一星期之后，你却看到：产量下降到每天不到 14 辆。这看起来很难理解，于是你就编制出一些当采用已经缩减了的新的工作时间时可能用得上的时间表。令人惊奇地是，你所能得出的最好的时间表是下图所示：

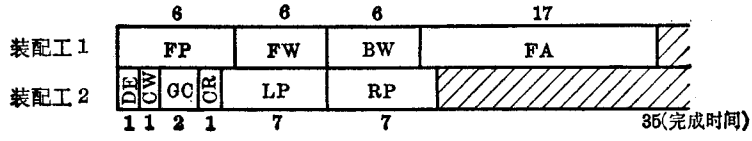


图 2　缩减工时后的最优时间表

所有遵从规则 1 和 2 的时间表至少得用 35 分钟才能装配一辆自行车。

于是你把租来的工具全部退了回去。在绝望之中你决定来一次蛮干：你雇用了 10 名临时的装配工，并命令，从今以后，每个装配组由三人组成，共同装配这些令人沮丧的自行车。你明白，你已经增加开销百分之五十的劳动费用，但你决心要完成规定的定额，否则你……。

这一回，只消两天你就会看出：事情非常不妙。每三人组的产量已降到每天不到 13 辆！

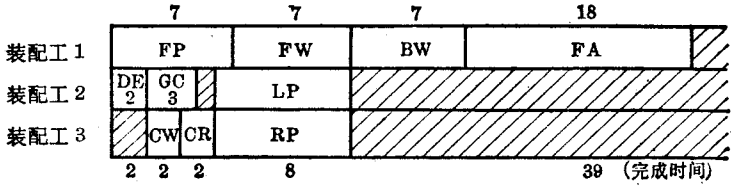


图3 三人组的最优时间表

你只得再写出一些可供使用的时间表。上图所示是其中典型的一种:

你会发现,很奇怪,对于三人组来说,所有服从规则1和2的可行时间表,至少需要39分钟才能装配一辆自行车。你花了几天时间在屋里转来转去,口里嘀咕着:"我到底错在哪儿?"你的解雇通知单在周末的时候已经送到手里了。

对于许多实际安排时间表的问题来说,上述的这一喻言确实具有十分严重的含义。实际上,早期研究时间表的这种类型的反常性的动机,就是起源于要为反弹道导弹防卫系统设计一个计算机工作程序。当时(幸而通过模拟)发现了:减少工作时间和增加计算机皆可能导致需要用更多的时间来完成全部工作[在F. Brooks所著"The Mythical Man-Month"一书中,曾对这一著名的"多了厨师坏了汤"效应(有时称为"Brooks定律")作了很有趣的讨论]。而且还发现,对于给定的系统增加更多的计算能力并不能保证按时完成所规定的工作。就一个导弹防卫系统来说,这显然是一件多少令人关心的事情。

人们会问,在Acme自行车厂中,我们假想的那位工段长究竟错在什么地方。我们将指出,他是规则1和2(以及一点坏运气)的牺牲品。他们那种目光短浅的迫切要求,其结果(正如常常出现的那样)就是给整个系统的效果带来全面的失败。无论在哪种情况,虽然出现了更加迫切的工作,但装配工(根据规则1)却不得不开始去干他在中途不能中断的工作(规则2)。

一般说来,时间表问题在几乎所有的人类活动领域中皆会出现。从典型问题看,这类问题可出现于从高度复杂的计划中人力资源的分配(例如海盗式火星探察计划,它要求在两万以上的人力和一个现代化的电子通迅网同时进行活动时能够相互协调)到一些简单的事情,比如准备一份七道菜的法国膳食之类。在过去几年中,不少时间表模型已经得到相当彻底的研究,以期了解:为什么会出现一些象我们上述的自行车例子里的出人预料的现象,其后果将会坏到什么程度,如何避免以及如何使之尽量减少等等。在本文中,我们将要说明对于其中一部分问题最近取得的一些进展,并谈一下一些研究工作者开始追求的某些令人鼓舞的新方向。一般,我们将从几个不同的角度来考察一个特定的问题,以期指出:一些不同的途径是怎样为我们提供有力的工具。

这一课题的一个非常重要的方面是使用例子,它一方面是为了懂得某些特殊类型的安排时间表的方法,另一方面则是为了真正发现什么样是对的,什么样是错的。实际上,本文的许多部分是谈一些例子,它们常常可以生动地阐明在这一领域中出现的某些没有预料到的微妙之处。通过这一讨论,我们希望:读者将不仅能洞察时间表理论本身,而且在数学(在当前情况是组合数学),计算机科学(在当前情况是算法理论)以及实际生活之间可能(而且常常)大量出现的相互作用方面也得到深入的了解。

一个数学模型

为了进一步确切地讨论我们的时间表问题，我们需要将自行车例子中出现的一些基本概念加以提炼。我们通过描述一个抽象的时间表问题模型来达到这一点。这一模型是由 m 台同样的机床(在前例中是装配工) $P_1, P_2, \cdots, P_m$ 及一组需要由这些机床依据某种规则来完成的工作 A, B, C, …, T, …(在前例中是 FP, BW, …)所组成。任意一台机床皆同样可以用来完成任何一件工作。但一台机床不能同时进行两件或两件以上的工作。每一工作 T 伴随着一个正数 $t(T)$，称为它的加工时间(在前例中是每道工序的装配时间)。一台机床一旦开始进行(或执行)某一工作 T，就要求它连续作下去，直到 T 完成为止，共使用 $t(T)$ 个单位时间(此即前述的规则 2)。

在每两件工作，例如 A 与 B 之间，可以有某种先后的限制(在前例中即为装配次序的限制)。我们写 A→B，或说 A 在 B 之前或 B 在 A 之后，来表示在任何时间表中，事件 B 总是在事件 A 完成之后才开始。当然，我们必须假定在先后限制上不存在循环(例如 A→B, B→C, C→A 之类)。因为循环的出现就显然使得这些工作不能做完(甚至不能开始)。若一台机床在某一时刻发现没有一件可以使它去干的工作，则该机床即可闲着。但若存在一件它可以去干的工作时，该机床就不允许闲着(此即前述的规则 1)。

根本的时间表问题就是要去决定：完工时间是如何依赖于加工时间、工作的先后限制、机床的台数以及用来构造时间表的策略。特别，我们想找出一种可以用来构造具有最短的完工时间的时间表的方法。

成效保证

关于如何测量我们用以编制时间表的方法的效果，我们要集中讨论所谓"最坏情况分析"这一途径。在这一途径中，我们试图去决定的是：对于任意一组满足某些特定限制的工作，可能发生的最坏情况是什么。作为这条途径的一个简单的例子是：我们想要知道，当单个工作的加工时间减少时，完工时间真正会增加多少。答案是由下列的结果给出，此结果是作者(Graham)在 1966 年得到的，在文献中，它是这一类型的结果中最早者之一。

设有 m 台机床。当加工时间为 $t(A), t(B), \cdots$ 时某一时间表的完工时间记为 f，当加工时间为 $t'(A), t'(B), \cdots$ 时某一时间表的完工时间记为 f'。若对于每一工作 T，皆有 $t'(T) \leqslant t(T)$，则

$$\frac{f'}{f} \leqslant 2-\frac{1}{m}。$$

比如说，若只有两台机床，$m=2$，则 $\frac{f'}{f} \leqslant \frac{3}{2}$。这表示：若加工时间下降，则在完工时间方面至多有 50% 的增加。

这一界限 $\left(2-\frac{1}{m}\right)$ 乃是一项成效保证。它说明：不管这一组工作的先后限制和加工时间是如何的复杂和异乎寻常，也不管在选取时间表时是如何聪明或愚笨，只要对于所有的工作

T，有 $t'(T)\leqslant t(T)$（容许对于所有的 T 有 $t'(T)=t(T)$ 这种可能性），则完工时间之比 f'/f 不大于 $2-\frac{1}{m}$ 这一结论总成立，而且 $2-\frac{1}{m}$ 这一界限是不能再改进的。这即是说：我们总可找到一些例子，使得 $f'/f=2-\frac{1}{m}$。我们现对 $m=6$ 给出一个例子。在这个例子中，时间 $t(T)$ 如下：

工作T	A	B	C	D	E	F	G	H	I	J	K	L	M
时间$t(T)$	7	4	5	6	5	3	7	6	5	5	4	3	12

且没有先后限制，而且对于所有工作 T，皆有 $t(T)=t'(T)$。在下两个图中，我们指出最坏的和最好的时间表。他们相应的完工时间分别为 $f'=22$ 和 $f=12$。当 $m=6$ 时，比值 $f'/f=22/12$ 正好等于 $2-\frac{1}{m}$。

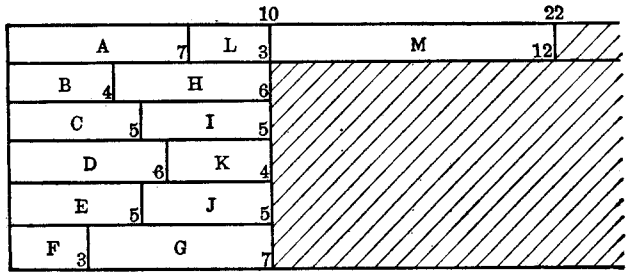


图4 最坏的时间表

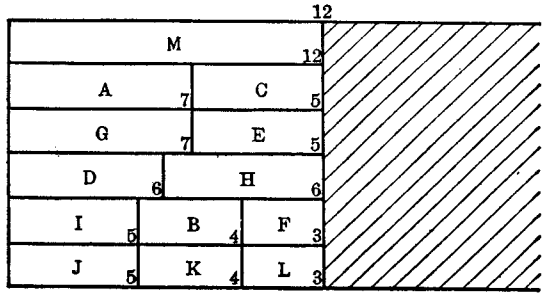


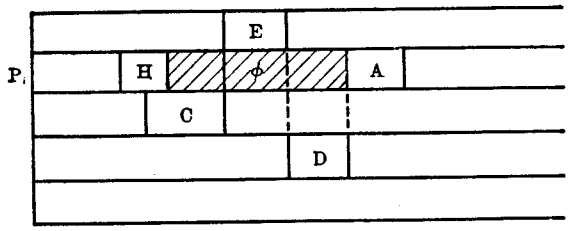
图5 最优的时间表

下面，我们将简短地来说明：这种类型的结果是如何得到的。有了这样的一个结果，我们就能预先知道 f 可能有的变化范围，因而可以设置一个适当的安全因子（比如 2），以容许在完工时间上可能出现的增加。在诸如前面提到的 ABM 防御系统的应用上，这一点可能具有非常重要的意义。

成效保证：一个证明

为了使爱好数学的读者能了解到比值 f'/f 的上界 $2-\frac{1}{m}$ 是如何得到的。我们现在简

略地来叙述一个证明。现在来考虑一个具有加工时间 $t'(T)\leqslant t(T)$ 及完工时间 f' 的时间表：



特别，我们现来察看一段其中有某一台机床，比如 P_i 空闲着的时间；在上图中，这段时间记作 ϕ。从图中可以看出，P_i 在完成工作 H 之后一直空着，直到工作 A 开始之后为止。A 之所以不能在 H 完成之后立即开始被加工的唯一原因就是：A 的某些优先者（即必须先于 A 加工的工作），比如 D，在当时尚未完成。或者是 D 在 H 完成之时已在被加工，或者是当时 D 尚未开始，两者必居其一。若当时尚未开始，则必是由于 D 之某一优先者尚未完成所引起。继续此种论证，我们就可断言：在 P_i 空闲着的全部时间里必有一串工作$\cdots$C$\rightarrow$E$\rightarrow$D$\rightarrow$A$\rightarrow\cdots$ 正被加工。利用同样的想法可知：只要有一台机床空闲时，必存在一串工作 $\mathscr{C}$，使得其中某一工作当时正被加工。若以 $t'(\mathscr{C})$ 记作 $\mathscr{C}$ 中所有工作的加工时间之和，以 $t'(\phi)$ 记作时间表中所有空闲时间之和，则从刚才所说，即得到

$$t'(\phi)\leqslant(m-1)t'(\mathscr{C}), \tag{1}$$

这是因为，一台机床只当 $\mathscr{C}$ 中有某一工作正在加工时才能空闲着，而此时，我们至多有 $m-1$ 台机床真正闲着。

显然有

$$f\geqslant t(\mathscr{C}), \tag{2}$$

这是因为 $\mathscr{C}$ 中的工作形成一个串，因此对于任何时间表，在一个时候必须加工其中一个。于是，若用 t^* 表示所有的已经减少了的加工时间之和，用 $\bar{t}$ 表示所有原来的加工时间之和，则有

$$\begin{aligned}
f'&=\frac{1}{m}(t^*+t'(\phi)) \quad \text{（因在时刻 0 与 } f' \text{ 之间，每一台机床或者忙着，或者闲着）}\\
&\leqslant\frac{1}{m}(t^*+(m-1)t'(\mathscr{C})) \quad \text{（由 (1)）}\\
&\leqslant\frac{1}{m}(t^*+(m-1)t(\mathscr{C})) \quad \text{（因为对于所有之 T，} t'(T)\leqslant t(T)\text{）}\\
&\leqslant\frac{1}{m}(t^*+(m-1)f) \quad \text{（由 (2)）}\\
&\leqslant\frac{1}{m}(\bar{t}+(m-1)f) \quad \text{（因 } t^*\leqslant\bar{t}\text{）}\\
&\leqslant\frac{1}{m}(mf+(m-1)f) \quad \text{（因 } \bar{t}\leqslant mf\text{）}\\
&=\left(2-\frac{1}{m}\right)f。
\end{aligned}$$

换言之，有

$$\frac{f'}{f}\leqslant 2-\frac{1}{m},$$

此即所欲证者。

关键路时间表

用来构造时间表的最普通的方法中，有一个就是以"关键路"时间表著称的方法。这一方法是构造所谓 PERT 网络的基础，PERT 在计划管理中得到广泛的应用。其基本概念是：当一台机床作完一件工作之后，它就应选择"最迫切的"工作来作为它下一步开始干的工作。所谓"最迫切"，我们是指在未处理的工作串中具有最大的加工时间之和的那种串里领头的工作。我们称这一"最长的"串为关键路。因在完成全部工作时，关键路中的工作最象是一个瓶口。在关键路(CP)时间表中，当前的关键路的领头工作总是被选来作为下一步处理的工作。

作为一个例子，现在重新来看看我们的自行车装配工作。我们可以把有关工作时间与先后限制的信息表示如图 6。

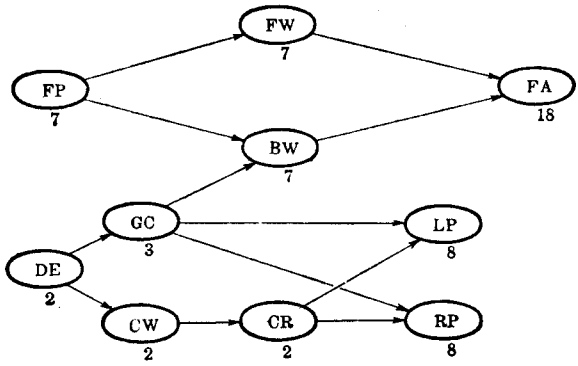


图 6 在 Acme 自行车厂中自行车装配时间与先后关系

在开始时，关键路为 FP→FW→FA 和 FP→BW→FA，两者的长皆为 32。在关键路时间表中先从此 FP 开始。当 FP 开始后，新的关键路即为 DE→GC→BW→FA，其长为 30。于是，在关键路时间表中另一台机床(装配工)即从 DE 开始。若我们继续这一时间表策略，最后我们即得到为图 7 所示的时间表。注意这一时间表之完工时间为 32。它显然是我们所能希望得到的最好的时间表。要是我们假想的那位工段长知道 CP 时间表该多好啊！

当然，这一例子实在太小，不足以显示出 CP 时间表的力量。关于这一方法存在的大量文献说明它在各种各样的应用中得到了广泛的使用。但是，从我们的最坏情况分析的观点

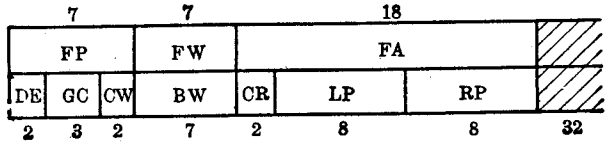
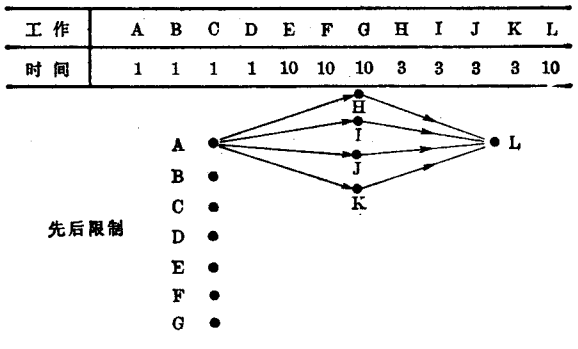


图 7 自行车装配的一个关键路时间表

来看，应当指出，在某些例子中，CP 时间表却可以表现得很差。事实上，不仅不能保证 CP 时间表接近于最优解，而且还可能出现：这一方法可能导致最坏的时间表。

下面是一个四台机床的例子。工作，加工时间以及先后限制如图 8 所示，图中还有一个CP 时间表，其完工时间 $f_{CP}=23$，及一个最优时间表，其完工时间 $f_{OPt}=14$。注意 $\frac{f_{CP}}{f_{OPt}}=\frac{23}{14}$，它与 $2-\frac{1}{4}$ 相差甚微。对于四台机床来说，按照我们的成效保证结果，$2-\frac{1}{4}$ 是任何两种完工时间之比的可能最坏的值。



A	H	I	J	K	L
E				B	
F				C	
G				D	

关键路法时间表　　$f_{OP}=23$

A	H	E
B	I	F
C	J	G
D	K	L

最优时间表　　$f_{OPT}=14$

图 8　关键路时间表并不总是产生最好的结果。在本例中对四台机床。关键路时间表所需时间差不多是最优时间表的两倍

给相互独立的工作安排时间表

对于构造时间表来说，关键路时间表给出了一个合理的，而且在实际中有用的方法。但我们已经看到，CP 时间表可能表现得非常坏。实际上，我们无法保证 CP 时间表不会产生可能最坏的时间表：它可以给出完工时间之比（与可能最短的完工时间之比）为 $2-\frac{1}{m}$。但在某些特殊情况，CP 时间表却决不会表现得如此坏。其中最普通的情况，就是在诸工作中没有先后限制的情况，现在我们来处理之。

假定你是某大公司中的一个打字室的主管人。每天早晨你要做的一件事就是将头天晚上送上来的各种文件分给室中的打字员来打字。由于各打字员之间以及文中引用页码等等

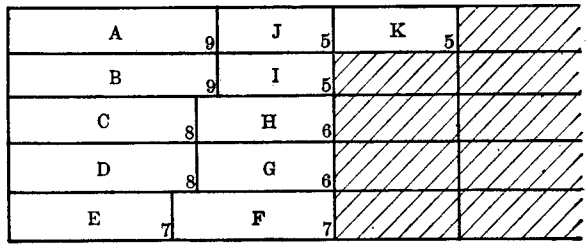
的不同，每一份文件必须由一单独的打字员来打字。你应如何分配文件使得需要打完全部文件的时间为最短？

正如我们在图 8 中所看出，一个坏的 CP 时间表可以比一个好的时间表花费将近两倍的时间。但若 CP 时间表是用于象打字室这样的情况，则下列的不等式常成立：

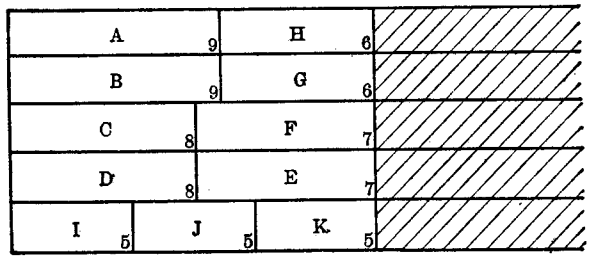
$$\frac{f_{CP}}{f_{OPt}} \leqslant \frac{4}{3} - \frac{1}{3m}。$$

故对于相互独立的(即无先后限制的)工作来说，使用 CP 时间表总可以在不超过最优完工时间的百分之 $33\frac{1}{3}$ 之内将全部工作完成。另一方面，正如讨论界限 $2-\frac{1}{m}$ 的情况那样，我们不难找出例子，它真正达到 $\frac{4}{3}-\frac{1}{3m}$ 这一界限。这说明这一界限是不可能改进的。在图 9 中，我们对 $m=5$ 给出了这样的一个例子。通过简单计算可以看出，比值 $f_{CP}/f_{OPt}=\frac{19}{15}=\frac{4}{3}-\frac{1}{15}$，当 $m=5$ 时，恰好是 $\frac{4}{3}-\frac{1}{3m}$。

工 作	A	B	C	D	E	F	G	H	I	J	K
时 间	9	9	8	8	7	7	6	6	5	5	5



关键路(CP)时间表 $f_{CP}=19$



最优时间表 $f_{OPT}=15$

图 9 关于五台机床的关键路时间表与最优时间表。诸如打字室那样，无先后限制的关键路时间表可以得出不比最优时间表的$\frac{4}{3}$更坏的结果

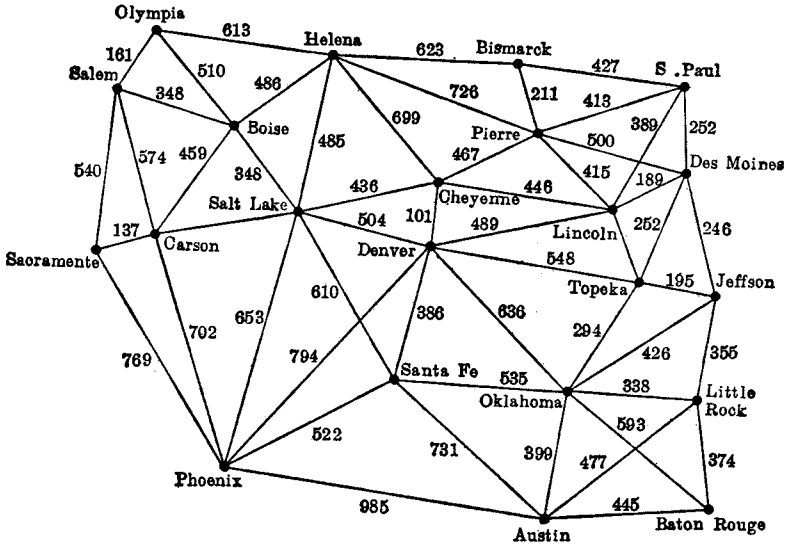
NP 完备问题

到了这一步，人们或许会问："为什么我们竟然满足于一个并非最好的时间表？为什么不使用一种总是产生一个具有可能最短完工时间的时间表的方法？"确实，这是一个值得称

赞的雄心壮志。毕竟，对于任一组特别给定的工作，你总是面对着一个有限组，因此你总可以去考察所有可能的时间表，然后去选取其中最好的一个。

这种使用蛮劲的方法所带来的麻烦是：当工作的件数增长时，可能有的时间表的个数的增长是爆炸性的，以至于当工作的件数相当大时，即使要将其中一小部分加以考察也是不可能的。假如我们出发时是 n 件工作，两台机器，则不同的时间表的数目为 2 自乘 n 次，或 2^n，即使 n 不大，2^n 也是一个非常大的数目。比如说，若 $n=70$，即使我们每秒能核算 100 万个时间表，也须要 30 万个世纪才能把所有 2^{70} 个可能有的时间表核算完毕。我们所真正需要的是一种方法(或者象计算机科学家所称的，一种算法)，它在工作件数增加时不会如此猛烈地增长。

不幸的是，这一目的看来非常可能永远达不到。这一阴暗的展望乃是 Toronto 大学的 Stephen Cook 的一篇带有基本性的著作的一个结果。他在 1972 年引入了"NP 完备"问题这一概念。现在已知，这一类问题包含有几百个不同的问题，它们以其在计算上难予对付而闻名。NP 完备问题，它们出现于象计算机科学、数学、运筹学、经济学等领域之中，具有两个重要性质：其一是，若 NP 完备问题中有一个具有有效解法，则其余各个皆然；其二是，对于任一 NP 完备问题，现阶段已知的求通解的方法总是类似于 2^n 所显示的那样以指数级增长。数学家们强烈地认为(但并未证明)：人们不可能找出一个有效算法这一事实是 NP 完备问题的一个固有性质。他们相信，不会有有效算法存在。在下面的方框中，我们举出了 PN 完备问题中一个比较著名的例子，即所谓货郎担问题。



正如读者现在可能猜想的那样，总的说来，时间表问题都是 NP 完备的。事实上，即使所用的是两台机器，而且没有先后限制，安排时间表仍是一个 NP 完备问题。这就是在本节开头我们采取悲观看法的根据。其后果之一就是把早先试图寻求一些好的方法以求问题的确切解的努力，大部分转到能够容易地求出近似解这一更有效果的方向。下一步我们所要做的，就是要从这一观点来看看两台机器系统的时间表问题。

货郎担问题

假设给定了一组"城市"，寻求从其中某一城市出发，把该组中所有的城市皆访问过，然

后回到出发点的最短路径，传统上称为货郎担问题。这类问题以种种不同的内容出现，例如：到各自动电话间去收款；周期性地到一组散布于各处的售货机去服务；到一工厂中某些点作安全检查；在一城市中将某种产品分送到不同的货站，等等。上图所示，乃是密西西比西部各州首府及其间的近似英里数。黑线的总长为 8119 英里。但最短路程为 8117 英里，你能求出来吗？

逐步接近最优时间表

对于没有先后限制的工作来说，正如我们已经看到的，关键路求时间表法总可以保证这些工作能在最短完工时间的 $\frac{4}{3}$ 的范围内完成。但要定出一个关键路时间表，我们必须将此 n 件工作按加工时间从大到小排列。此种安排可以在比如 $n\log_2 n$ 这样的时间之内完成，这只比依线性增长稍多一些。

我们现在假定要处理的是两台机器的问题，而且我们愿多做一些工作，但我们希望得到一个与最好的解答非常接近的答案。要做到这一点，一条途径就是：对某一给定的整数 k，将加工时间最长的 $2k$ 件工作挑出，再对于这 $2k$ 件工作构造出一个最好的时间表；然后将其余的工作随便排一个次序，假若我们将这样的一个时间表的完工时间记作 f_k，则有

$$\frac{f_k}{f_{\text{opt}}}\leqslant 1+\frac{1}{2k+2}。$$

对于包含 n 件工作的一组工作来说，全部手续至多在 $n\log_2 n+2^{2k}$ 次运算即可完成，于此 $n\log_2 n$ 这一项来自于要将此 $2k$ 个加工时间最长的工作挑选出来，2^{2k} 这一项来自于检验各种可能的(二台机器的)时间表。因 k 是一固定数，故当 n 增大时，这一函数的增长率仍是可容忍的。

例如取 $k=3$，我们可以保证 $f_3/f_{\text{opt}}\leqslant\frac{9}{8}$，而工作量至多与 $n\log_2 n+64$ 成比例。一般说来，只要愿意付出代价，我们总可保证得到所要的精确度。不幸的是，我们所要付出的代价增长得太快了。例如要保证 f_k 的值不超过最优值的百分之二，而运算时间可能与 $n\log_2 n+2^{49}$ 成比例。这可能把一点少量的计算机预算花光了还不够。对于这样一种情况，也不必过于感到诧异。假若在求最优解时看来需要指数量的时间，那么我们可以预料到求近似解时的花费，当所要保证的精确度增大时，也会类似地变化。令人惊奇的是：这种在时间上指数量级的增长确是可以避免的。Minnesota 大学的 Oscar Ibarra 和 Maryland 大学的 Chul Kim 最近发现了一个求时间表的算法，在保证完工时间 f_k 满足关系

$$\frac{f_k}{f_{\text{opt}}}\leqslant 1+\frac{1}{k}$$

时，至多需要 kn^2 步即可将所要的时间表作出。函数 kn^2 是 k 和 n 的"多项式"的一个例子，当 k 和 n 增大时，kn^2 的值比指数函数 2^k 小得多。得出这一算法的基本思想是：将动态规划与"rounding"巧妙地相结合。它不在本文讨论范围之内。但是，这种类型的近似解可以通过使用合理的计算机时间保证得到与最优解非常相近的解答。