文章编号：1000－8152(2001)01－0119－08

MRCPSP 的一种精确算法*

毛 宁 陈庆新 陈 新
(广东工业大学机电学院·广州，510090)

摘要：着眼于多模式资源受限项目调度方法，其特色在于，针对项目中每个任务的工期不仅取决于自身的执行模式，而且取决于该任务实际开工时间的一般情形，同时考虑每个任务对可更新(再生)资源需求呈任意分布、可更新(再生)资源的最大供给量随时间而变化的一般情况．作为对前人研究成果的进一步推广，本文在经典单模式 DH 分枝定界算法的基础上，利用事件驱动的时间增量方式，成功地获得了这种最一般的项目调度问题的最优解．

关键词：多模式；资源受限；项目调度；不可中断；分枝定界
文献标识码：A

An Extension to the DH-Branch-and-Bound Algorithm for MRCPSP

MAO Ning, CHEN Qingxin and CHEN Xin
(School of Mechanical & Electronic Engineering, Guangdong University of Technology·Guangzhou, 510090, P. R. China)

Abstract: This paper deals with the multi-mode multiple resource-constrained project scheduling problem (MRCPSP). The duration of each activity in the project is not only dependent upon its executive mode, but also related to its actual start-time in a schedule. Besides, differently from the problem invesigated by other researchers, the problem handled is with variable resource requirement and variable renewable resource availability constraints. As an extension, the branch-and-bound algorithm put forward can solve this kind of most general project scheduling problems with optimality.

Key words: multi-mode; resource-constrained; project scheduling; nonpreemptive; branch-and-bound

1 引言(Introduction)

多种模式资源受限项目调度(multi-mode multiple resource-constrained project scheduling problem, MRCPSP)是一类非常广义的调度优化问题．它要求在满足项目紧前约束与资源约束的前提下，确定项目中每个任务的执行模式，以及相应的开工期和完工期，以便最小化项目的总工期．其中人们熟知的经典单一模式资源受限项目调度问题(multiple resource-constrained project scheduling problem, RCPSP)，Job Shop, Flow Shop, 单机与并行多机等等各类调度问题，都是它的特殊情形．它在敏捷制造的伙伴选择、异地制造的项目规划、新产品的开发、单件全订货型(engineering to order)生产的规划与调度管理、基建项目施工调度等领域有着十分广泛的应用．在敏捷制造环境下，任务的多模式也就意味着人们可以从多个制造厂商中选择合作伙伴；在传统制造业中，任务的多模式也就意味着人们可以对多种完成某一任务的工艺路径进行选择；而在基建业中，任务的多模式也就意味着人们可以对多种施工方式进行选择．

但是这一问题十分难于求解．针对一个具体的项目调度问题，为每个任务选配执行模式将毫无疑问地扩大解空间．已经证明，在至少有两种不可更新(再生)资源的约束条件下，每个任务又至少存在两种可供选配的执行模式，寻找基本 MRCPSP 的一个可行解是 NP-完备问题，目前所有的启发式算法都无法确保在任何有解的情况下找到一个可行解[1]．虽然在目前可以接受的计算资源和计算时间条件下，精确算法只能求解小规模问题，但是它至少能够在任何有解的情况下确保找到一个可行解甚至是最优解，或者确证不存在可行解．除此而外，任何精确算法的研究进展，往往会给以后的研究者们带来十分有益的启示，使他们能够提出更为有效的启发式求解策略和算法．因此，研究 MRCPSP 的精确求解算法有着重要的理论意义和现实的应用前景．

与前人的着眼点有所不同，本文针对项目中每

* 基金项目：国家 863/CIMS 跟踪项目(863－511－9843－008 与 863－511－9944－008)与广东省自然科学基金(970380)联合资助项目．
收稿日期：1998－09－28；收修改稿日期：2000－02－18．

个任务的工期不仅取决于自身的执行模式，而且取决于该任务的实际开工时间的一般情形.除此之外，我们还考虑了项目中每个任务对可更新(再生)资源需求呈任意分布、以及可更新(再生)资源的最大供给量随时间而变化的情况.这些假设是根据实际应用场合的需要而提出的，是完全符合工程实际情况的.接着，我们在经典单模式 DH 分枝定界算法的基础上，利用事件驱动的时间增量方式，成功地获得了这种最一般的项目调度问题的最优解法.

2 模型(Model)

为了方便地进行模型描述，我们首先需要解释一些常用符号的含义.项目中的任务总个数为 J；可更新(再生)资源集为 R；不可更新(再生)资源集为 N；资源 r 在基本时间段 $(t-1,t]$ 的有限能力为 $K_{rt}\geqslant 0$；任务 j 可供选配的模式个数为 M_j；模式 m 与任务 j 的实际开工期 s_{jm} 决定了其工期 $d_{jm}(s_{jm})$，并满足如下条件：$s_{jm1}+d_{jm}(s_{jm1})\leqslant s_{jm2}+d_{jm}(s_{jm2})\Leftrightarrow s_{jm1}\leqslant s_{jm2}$；我们相应地定义每个任务对不可更新(再生)资源的消耗量为 $k_{jmr}\geqslant 0$；同时采用 $w_{jmrt}\geqslant 0$ 来表示任务 j 在基本时间段 $(\tau-1,\tau)$ 上对可更新(再生)资源 r 的需求分布，其中 $\tau=s_{jm},\cdots,s_{jm}+d_{jm}(s_{jm})-1$，而在其它时间段上 $w_{jmrt}=0$.任务 j 的紧前任务集为 P_j，按照拓扑排序规则，针对 $\forall i\in P_j$，满足条件 $i<j$；对于具有多种可选配模式的任务，我们以其工期的非降次序排列模型；针对项目总工期的一个上界 T，我们可以按照以下方式来计算每个任务 j 的最早紧前可行完工期 EFT_j 和最迟紧前可行完工期 LFT_j.首先我们给每个任务配置相应工期最短的模式，再利用传统的前向递归计算每个任务的 EFT_i；然后，我们设置 $LFT_J=T$ 并利用传统的反向递归计算每个任务的 LFT_j.接着，我们引入二位式决策变量 x_{jmt}，$1\leqslant j\leqslant J$，$1\leqslant m\leqslant M_j$，如果任务 j 在模式 m 下于基本时间段 $(t-1,t]$ 结束时完成，我们就定义它等于 1，而在其它情况下，我们都定义它等于0.最后对于网络源点1，我们假设 $d_{1m}=0$，$k_{1mr}=0$，$M_1=1$，$P_1=\varnothing$，$1\leqslant m\leqslant M_1$，$r\in R\cup N$，而对于网络汇点 J，我们也假设 $d_{Jm}=0$，$k_{Jmr}=0$，$M_J=1$，$1\leqslant m\leqslant M_1$，$r\in R\cup N$，因此，在文[3~5]中模型的基础上，并且在项目中每个任务的工期不仅取决于其执行模式，而且取决于该任务的实际开工时间的一般情形下，我们建立了具有任务资源需求分布与资源时变约束的一般 MRCPSP 0-1 整数规划模型如下：

$$\sum_{m=1}^{M_j}\sum_{t=EFT_j}^{LFT_j}x_{jmt}=1,\ j=1,\cdots,J, \tag{1}$$

$$\begin{cases}\sum_{m=1}^{M_i}\sum_{t=EFT_i}^{LFT_i}tx_{imt}\leqslant\sum_{m=1}^{M_j}\sum_{t=EFT_j}^{LFT_j}(t-d_{jm}(s_{jm}))x_{jmt},\\ j=2,\cdots,J,\ i\in P_j,\end{cases} \tag{2}$$

$$\sum_{j=2}^{J-1}\sum_{m=1}^{M_j}w_{jmrt}\sum_{\tau=t}^{t+d_{jm}-1}x_{jm\tau}\leqslant K_{rt},\ r\in\mathbb{R},\ t=1,\cdots,T, \tag{3}$$

$$\sum_{j=2}^{J-1}\sum_{m=1}^{M_j}k_{jmr}\sum_{t=EFT_j}^{LFT_j}x_{jmt}\leqslant K_r,\ r\in\mathbb{N}, \tag{4}$$

$$x_{jmt}\in\{0,1\},\ j=1,\cdots,J,\ m=1,\cdots,M_j. \tag{5}$$

约束集(1)确保了每个任务 $j=1,\cdots,J$ 总是在时间窗 $[EFT_j,LFT_j]$ 中以其多个可选模式之一被执行.约束集(2)表示了项目中各个任务之间的紧前关系.约束集(3)则在每个基本时间段 $(t-1,t]$ 中限制了所有正执行任务对每种可更新(再生)资源的最大供给量(容量).约束集(4)限制了项目中所有任务对每种不可更新(再生)资源的总消耗量不超过相应不可更新(再生)资源的拥有量.最后，约束集(5)定义了所有的二位式决策变量.针对约束集(1)~(5)的常用目标函数是最小化项目总工期：

$$\min\phi(M,S)\triangleq\sum_{t=EFT_J}^{LFT_J}tx_{JM_JT}. \tag{6}$$

此外，在实际应用中，往往还会出现这样的情况，即针对某些任务 j，要求其必须在时间区间 $[g_j,h_j]$ 内完成，其中 g_j 和 h_j 分别是任务 j 的准备就绪时间和必须完工时间.这时，还需要算法就这种针对任务的时间约束，对产生的调度方案进行附加的可行性验证.

问题(1)~(6)是最为一般和最为困难的(项目)调度问题之一.作为众所周知的 Job Shop 调度问题的一种推广，它属于一类 NP 困难的问题类[6].

3 算法(Algorithm)

首先，把每个部分调度方案对应于分枝定界搜索树的每个节点，并给该方案中的每项任务暂定完工期和执行模式.如果该部分调度方案既满足紧前约束又满足资源约束，那它就是可行方案.然后在以下四个量中取最小值，即可更新(再生)资源约束发生下一个变化的时刻、目前正进行任务的可更新(再生)资源总需求发生下一个变化的时刻、除部分延迟开工的任务之外所有正进行任务的最早完成时间、不因部分延迟开工任务而构成紧前约束的所有未调

度任务在所有可行模式下的最早可能完成时间，作为下一步调度的决策时间点．在一个调度方案的执行期间，假设 K_{rt} 和 $w_{jmrt} \geqslant 0$ 是已知的变量．针对优化目标(6)，已经证明通过半活动时间列表(semi-active time-tabling)原则来构造部分调度方案是充分的[7]．通俗地说，就是在满足项目紧前约束和资源约束的前提下，应当尽可能早地执行每个任务．我们称调度决策是"暂时"的，是指那些已经暂定了完工期和执行模式的任务，可能会因以后的决策而延期．在时刻 t，我们定义那些尚未安排开工期和执行模式的任务为未调度任务，而对应的部分调度方案 PS_t 将包括那些已经完工的和正在进行中的任务．前者的完工时间小于等于 t，我们将它们放入在时刻 t 已完工的任务集 F_t．而后者属于在时刻 t 正在进行中的任务集 S_t．在时刻 t，未完工的任务集 U_t 则包含了所有不属于 F_t 的任务．

我们从基本时间段(0,1]开始就建立部分调度方案，并在每个后续的决策时间点上，增加任务子集(包括空集)或选择任务组合的执行模式，直到获得一个完整可行的调度方案．在这个意义下，我们将一个部分调度方案一直延续下去，最后就一定能够得到一个完整的调度方案．

在现有剩余的不可更新(再生)资源的约束条件下，我们再扣除所有未调度任务中非目前需调度任务对不可更新(再生)资源的最低需求，接着针对目前需调度且尚未选配执行模式的所有任务，产生所有可行的任务组合模式配置方案．这样的任务组合可行模式配置方案将在分枝定界求解树中产生新的分枝．这些分枝描述了任务组合不同的可行执行模式配置，即在这些任务组合中，由哪个任务配置对应的哪种执行模式，所构成的不同模式组合决策．

在每个时刻 t，我们定义合格集 E_t 作为不属于部分调度方案而其紧前任务已经完成的那些任务的集合．如果资源约束不受到破坏，这些合格的任务能够在时刻 t 开工．

如果在时刻 t，由于资源约束而不可能调度所有的合格任务，我们就说产生了一个可更新(再生)资源冲突．这样的冲突也将在分枝定界求解树中产生新的分枝．这些分枝描述了解决资源冲突的方式，即关于哪些任务组合将要延迟执行的决策．我们定义延滞集 $D(p)$ 为由所有的任务子集 D_q 所组成的集合，其中 D_q 中的任务要么正在执行，要么是合格的，这些任务的延迟将在搜索树的第 p 层解决目前的可更新(再生)资源冲突．如果一个延迟选择 D_q 不包含其它的延迟选择 $D_v \in D(p)$ 作为子集，则称 D_q 是最小化的延迟选择．

定理 1　为了解决可更新(再生)资源冲突，仅仅考虑最小化的延迟选择是充分的．

证　设 $(i_{1m(1)},\cdots,i_{pm(p)},\cdots,i_{km(k)})$ 是导致一个最优调度方案 G 求解树的节点链．如果在求解树第 p 层上，$\mathrm{flag}(p)=1$，则每个这样的节点 $i_{pm(p)}$ 对应着一个延迟选择 D_p 和一个紧前约束集 P_p．针对所有其他节点 $i_{pm(p)}$，集合 P_p 可以从集合 $P_{p-1},\cdots,P_{p-n}$ 与相应的延迟决策中推出．设 $f^*_{jm(j)}$ 是最优调度方案 G 中任务 j 的完工期．而节点 i_q 为节点链中第一个针对非最小延迟选择 D_q 的节点．如果不存在这样的节点 i_q，定理的结论就成立．

否则，与 D_q 相对应的所有已调度任务的完工期为 $f_{jm(j)}$ 和所有未调度任务的开工期为 $s_{im(i)}$，针对所有 $i \in D_q$ 和 $y \in P_q$，$s_{im(i)} \geqslant f_{ym(y)}$，并且针对所有未调度任务，其最早可能开工期不小于各自所有紧前任务的最大完工期．根据最小延迟选择的定义，至少存在一个非延迟任务 x，设其完工期为 $f^t_{xm(x)}$ 可以从 D_q 中除去，即 $\exists x, \ni D^t_q = D_q \setminus \{x\}$，而 D^t_q 是搜索树第 q 层上发生在节点 i^t_q 上的一个有效延迟选择．而针对所有 $i \in D^t_q$ 和 $y \in P^t_q$，$s^t_{im(i)} \geqslant f^t_{ym(y)}$ 现在，我们只须讨论以下两种可能性：第一，如果 $f_{ym(y)} \leqslant f^t_{xm(x)}$ 则针对所有的 $i \in D^t_q$，有 $f^t_{ym(y)} = f_{ym(y)}$．因为 $D^t_q = D_q \setminus \{x\}$，则 $P^t_q \subseteq P_q$．又因为 G 相对于 Pq 是可行的，那么 G 相对于 P^t_q 也是可行的，因而能够通过继续从节点 i^t_q 处产生分枝找到调度方案 G．故我们能够探测到节点 i_q．第二，如果 $f_{ym(y)} > f^t_{xm(x)}$，则针对所有的 $i \in D^t_q$，我们不妨更新开工期：$s^t_{im(i)} = f^t_{xm(x)} \leqslant f_{ym(y)}$，并且所有未调度任务 j 的开工期取决于它们各自的紧前任务的完工期．因为 P^t_q 中任务的最大完工期不大于 P_q 中任务的最大完工期，故有 $s^t_{jm(j)} \leqslant s_{jm(j)}$．当 $t = f^t_{xm(x)}$ 时，我们有 $F^t_t = F_t \cup \{x\}$ 和 $U^t_t = U_t \setminus \{x\}$．我们能够做出以下推论：针对所有的 $i \in F_t$，$f^t_{im(i)} = f_{im(i)}$；针对所有的 $j \in U^t_t$，$s^t_{jm(j)} \leqslant s_{jm(j)}$；并且 $f^t_{xm(x)}$ 小于任务 x 在 P_q 中的任何可行完工期．现在我们可以利用任务的完工期 $a_{jm(j)}$ 来构造一个新的调度方案 G 如下：针对所有的 $i \in F^t_t$，$a_{im(i)} = f^t_{im(i)}$，并且针对所有的 $j \in U^t_t$，$a_{jm(j)} = f^*_{jm(j)}$．那么 G' 相对于 P^t_q 也是可行的，因而能够通过继续从节点 i^t_q 处产生分枝找到调度方案 G'．因为 G' 与 G 具有完全相同的总工期，所以调

度方案 G' 也是最优调度方案,故我们能够探测到节点 i_q.

现在,我们已经证明了:如果从比第一个在求解树节点 i_q 处的非最小延迟选择少一个任务的延迟选择出发进行分枝,能够找到一个最优解.我们能够不断重复从这个非最小延迟选择中除去一个任务的过程,直到在节点 i_q^u 处获得一个最小延迟选择 D_q^u.如果我们继续从节点 i_q^u 处进行分枝,沿着节点链 $(i_1,\cdots,i_{q-1},i_q^u,i_{q+1}^u,\cdots,i_{\mu}^u)$,一定能够找到一个新的最优解.沿着这条节点链进行构造,在节点 $i_1,\cdots,i_q^u$ 处的所有延迟选择都是最小化的,而在节点 $i_{(q+1)}^u,\cdots,i_{\mu}^u$ 处的延迟选择可能是非最小化的.我们能够重复不断地从一个非最小化的延迟选择中除去一个任务直到获得一个最小化的延迟选择,这一过程还能够在沿着这一新的节点链上产生第一个非最小化延迟选择的那一层就开始执行.然后我们能够不断地重复这一步骤直到所有的延迟选择都是最小化的.至此,我们完全证明了仅仅通过考虑最小化延迟选择就能够获得最优调度方案. 证毕.

通过计算关键序列下界 L_q,我们将对每个延迟选择 D_q 进行评价.这一考虑紧前约束的下界,是按照以下的步骤计算的.针对每个我们希望确定关键序列下界的延迟选择 D_q,基于所谓的延迟点 w_q,确定相对应的部分调度方案 $PS = PS_t \cup E_t \setminus D_q$,再给所有未配置执行模式的任务选配对应工期最短的模式,然后针对网络进行基于紧前约束的关键路径计算.这样,就可以从对应着具有最小 L_q^* 的延迟选择 D_q^* 的节点出发,不断地进行分枝搜索过程.

以前人们普遍认为,在深度优先的搜索过程中,不可能利用调度方案的控制支配原则改进搜索效率.与这一观点相反,我们利用两种控制支配规则对搜索树进行部分删除以提高搜索效率.无论何时,只要算法辨别出,在不破坏紧前约束和资源约束的条件下,能够在现有部分调度方案中左移一个任务,就应用第一项控制支配规则(定理 2).这一左移控制支配规则在原理上与前人所使用的完全一致.本文中这一规则是按照以下具体步骤执行的.定义集合 $DS \triangleq |j \in D_q^* \mid f_{jm} < t + d_{jm}(s_{jm})|$ 作为在早于时刻 t 的某个时刻就已经决定开工,但当我们选定了延迟选择 D_q^* 而不得不延迟的那些任务所组成的集合.如果 DS 不是空集,搜索树以前层上作出的延迟决策迫使任务 j 在时刻 t 变为合格任务,如果当前的决策就是在时刻 t 开工这项任务,并且如果在不产生资源冲突的条件下,延迟任务集 DS 允许任务 j 左移,那么相对应的部分调度方案就受到了控制支配.

定理 2 在搜索树的第 p 层上考虑部分调度方案 PS_t,其中任务 j 在时刻 t 开工.如果在搜索树以前层上确定的延迟选择迫使这一任务在时刻 t 才成为合格任务,并且如果这一任务能够在不导致紧前和资源冲突的情况下左移(因为在第 p 层确定的延迟选择是正进行中的任务),则部分调度方案就受到了控制支配.

证 根据定理的假设条件,我们不妨设任务 j 能够在不导致紧前和资源冲突的情况下左移至时刻 t' 开工,其中 $t' < t$.这是因为现在(在时刻 t)对在以前时刻决定开工的部分任务决定进行延迟从而释放了部分项目资源所致.根据最小化延迟选择原则(定理 1)和半活动时间列表原则[7],此时对在以前时刻决定开工而现在又决定延迟的那部分任务所构成的延迟选择集合一定是最小化的.但是根据时间增量原则和最小化延迟选择原则,这一部分调度方案所对应的节点一定是在时刻 t' 所对应的搜索树上的分枝层中一个节点的后续节点.故此,该部分调度方案就受到了支配. 证毕.

第二项控制支配规则(定理 3)是基于割集的概念.在每个时刻 t,我们定义一个割集 C_t 为所有那些未调度任务组成的集合,而它们的所有紧前任务又都属于部分调度方案 PS_t.所谓相同的可更新(再生)资源时间区间,就是指在该时间区间内,可更新(再生)资源的供应量与割集中所有任务对可更新(再生)资源的总需求都为常数.

定理 3 在时刻 t 考虑一个包含着与割集 C_k 同样任务及其执行模式的割集 C_t,而割集 C_k 是以前在搜索树上另一条路径的搜索期间暂时存储的割集,并且 C_k 与 C_t 在相同的可更新(再生)资源时间区间内.如果时刻 k 不大于时刻 t,并且如果在时刻 k,所有正在进行中的任务不迟于时刻 t' 完成,其中 t' 定义为 t 与 PS_t 中相对应任务的完成时间之间的最大值,并且如果 C_k 中每个任务的最早可能开工期小于等于 C_t 中相对应任务的最早可能开工期,则现行的部分调度方案 PS_t 就受到了控制支配.

证 假设 G 是一个最优调度方案,其中每个任务的完工期为 $f_{im(i)}^*$,相应的执行模式为 $m(i)$,它是按照部分调度方案 PS_t 延续而获得的,而 PS_t 中的割集 C_t 满足定理的假设条件.根据定理的假设条件,

存在一个以前在搜索树不同的路径上产生的割集 C_k，它对应着部分调度方案 $PS_k = PS_t$、更进一步，$k \leqslant t$—— 它们都属于相同的资源区间内，并且针对所有的 $j \in PS_k$，有 $f_{jm(j)k} \leqslant \max\{t, f_{jm(j)t}\}$，其中 $f_{jm(j)k}$ 和 $f_{jm(j)t}$ 分别指 PS_k 和 PS_t 中的任务 j 在执行模式 $m(j)$ 下的完工期. 根据定理假设，针对所有的 $i \in C_k = C_t$，我们有：$s_{im(i)k} \leqslant s_{im(i)t}$，其中 $s_{im(i)k}$ 和 $s_{im(i)t}$ 分别指 C_k 和 C_t 中的每个任务 i 在执行模式 $m(i)$ 下的开工期. 因为 G 是按照部分调度方案 PS_t 延续而获得的，那么针对所有的任务 $j \in PS_t$，我们可以得到：$f_{jm(j)t} \leqslant f^*_{jm(j)}$.

我们定义 J_t 为满足条件 $f^*_{jm(j)} - d_{jm(j)}(S_{jm(j)}) > t$ 的所有任务 j 所构成的集合. 按照以下的步骤构造一个新的调度方案 G'：针对所有的任务 $j \in J_t$，我们设置 $f^l_{jm(j)} = f^*_{jm(j)}$；而针对所有的任务 $i \in \{J \setminus J_t\}$，我们设置 $f^l_{im(i)} = f_{im(i)k}$. 根据定理假设，针对任务 $j \in C_t$ 的所有紧前任务 i，$f^l_{im(i)} f_{im(i)k} \leqslant \max\{t, f_{im(i)t}\} \leqslant \max\{t, f^*_{im(i)}\}$ 成立，并且针对所有的割集任务 $j \in C_t$，根据条件：$s_{jm1} + d_{jm}(s_{jm1}) \leqslant s_{jm2} + d_{jm}(s_{jm2}) \Leftrightarrow s_{jm1} \leqslant s_{jm2}$，可以得到：$s_{jm(j)k} \leqslant s_{jm(j)t} \leqslant f^*_{jm(j)} - d_{jm(j)}(s_{jm(j)})$，所以 G' 没有破坏任何项目任务之间的紧前约束；又因为 PS_k 与 G 是可行的，针对所有任务 $j \in PS_k$，$f_{jm(j)k} \leqslant \max\{t, f_{jm(j)t}\} \leqslant \max\{t, f^*_{jm(j)}\}$ 成立，而调度方案 G' 中的所有任务 $i \in J_t$ 在时刻 t 与任何一个紧前任务 j 的完工期 $f^*_{jm(j)}$ 之后才进行调度，所以调度方案 G' 没有破坏任何项目的资源约束；因此 G' 是可行的调度方案. 但是 G' 又与 G 具有完全相同的总工期，故根据构造 PS_k 的节点进行分枝而产生的调度方案 G' 也是最优调度方案，因此 PS_t 就受到了支配.

证毕.

当完成了一个调度方案或利用下界计算与(或)控制支配规则探查了一个分枝时，算法就要开始回溯. 在回溯过程中，我们首先判断这层的分枝是由任务组合不同的执行模式配置还是由可更新(再生)资源的冲突产生的. 如果是前者，就探查任务组合的另一个可行执行模式配置，如果这一层中没有遗留尚未探查的任务组合执行模式配置，我们就回溯到前一层；如果是后者，我们就删除刚才选择的延迟集元素 $D^*_q \in D(p)$，并且在同一层中针对下一个可选的延迟选择进行评价. 如果在这一层中没有遗留尚未探查的延迟选择 D^*_q，我们就回溯到前一层. 当到达了搜索树的第零层，我们就完成了整个搜索过程，也就已经找到并确证了调度问题的最优解.

本文描述的搜索过程是深度优先的. 我们首先构造了割集中尚未选配执行模式任务组合的所有可行执行模式配置集 $MA(p)$，采用当前面临调度的任务组在每种执行模式配置方案的情况下的剩余关键序列长短来评价这些可行配置方案的优劣，并作为下界来决定探查次序的先后. 我们必须详尽无遗地产生所有可行的任务执行模式配置组合以便于确保最优性. 接着我们构造包含了所有合格或进行中任务组合的延滞集 $D(p)$，这些任务满足以下条件：i) 它们的延迟能够释放足够的可更新(再生)资源以解决资源冲突；ii) 这些组合满足最小化条件，即任意一个组合不可能包含另一个组合作为子集. 同样，我们也必须详尽无遗地产生这一最小化延迟选择集合以便于确保最优性. 在我们的方法中通过计算关键序列下界对每个分枝选择进行评价. 在这个意义下，我们的深度优先步骤在开始做得更艰难，以便得到一个较完善的可行初始方案. 利用两个支配规则，我们能够在回溯过程中探查数量比较多的节点.

下面针对 MRCPSP 描述分枝定界算法的详细计算步骤. 为了简化符号表示，我们省略了集合 PS、S、E 和 C 的下角标 t. 应当区分第二步中执行的暂存操作与第五步中执行的存储操作. 暂存操作暂存需要的割集信息以便应用割集控制支配规则. 存储操作存储回溯期间需要加以恢复的信息.

第一步　初始化.

· 设置 $T = 9999$ 作为项目总工期的一个上界；设置分枝定界搜索树的层数 $p = 0$；初始化 $t = 0$. 设置 $t' = 0$(t' 是一个特定的决策点，以便于检验左移支配规则的可应用性)；针对所有指定了准备就绪时间 g_j 或完工期 h_j 的任务 j，输入 g_j 或 h_j；调度初始任务：$f_{11} = 0$，$PS = \{1\}$ 以及 $S = \{1\}$；针对每个任务，配置相应工期最短的模式，计算第 0 层的下界：$LB(0) = q_1$；将所有以任务 1 为唯一紧前的任务放入割集：$C = \{x \mid x$ 具有唯一的紧前任务 $1\}$，同时设置割集任务 x 的最早开工期 $s_x = g_x$.

· 计算并存储不可更新(再生)资源的约束范围，并转向第二步.

第二步　模式选择与配置.

· 如果割集内没有尚未配置执行模式的任务，转向第三步.

· 更新搜索树的分枝层：$p = p + 1$. 同时设置层类型标志 $\mathrm{flag}(p) = 0$；在目前不可更新(再生)资源的约束范围内，扣除不属于 C 的未调度任务对不可更新(再生)资源的最小所需量，再针对割集 C 中尚未配置执行模式的所有任务，确定所有可行的模式配置，构成割集任务的可行模式配置集 $MA(p)$. 其中每个元素对应着割集 C 中尚未配置执行模式的所有任务，在目前剩余的不可更新(再生)资源约束条件下，各自选配一种相应的执行模式，总体构成的一种模式选配方案；如果割集任务的可行模式配置集 $MA(p) = \emptyset$，则转向第七步(回溯).

· 针对割集任务组合的每个可行执行模式配置方案，确定任务组合相应的剩余关键路径，选择具有最小关键路径的任务组合可行执行模式配置方案作为下界 $LB(p)$，如果 $LB(p) \geqslant T$，转向第七步(回溯).

· 给任务组合配置具有最小关键路径的可行执行模式方案，并将选中的配置模式从可行配置集 $MA(p)$ 中除去；存储以前的不可更新(再生)资源的约束范围，再计入当采用这一目前的配置方案时对不可更新(再生)资源的消耗，更新目前的不可更新(再生)资源的约束范围，并转向第三步.

第三步　时间的增加.

· 如果最后结束的任务 J 属于部分调度方案 PS，就对每个任务 j 以及配置的相应模式 m，验证是否满足开工期与完工期约束，即是否 $s_{jm} \geqslant g_j$，$s_{jm} + d_{jm}(s_{jm}) \leqslant h_j$，如果是，则调度过程圆满完成；否则转向第七步回溯.

· 如果这是一个改进了的解，更新调度方案总长 $T = f_J$. 如果 $T = LB(0)$，则已经获得了最优解，停止. 否则转向第七步：回溯.

· 在目前割集任务所配置的模式条件下，取以下三个量中的最小值，即所有割集任务的最早开工时间、可更新(再生)资源约束发生变化的下一个时刻、以及目前正进行任务对可更新(再生)资源的总需求发生变化的下一个时刻，作为下一个决策点 t；针对所有任务 $j \in S$，并且 $f_{jm} \leqslant t$，更新正进行的任务集：$S = S \setminus \{j\}$；检验割集支配性. 如果目前的割集受到支配，则转向第七步(回溯)，否则暂存目前的割集 C、其中任务的最早开工时间与相应的执行模式、决策点 t，以及正进行的任务和它们的完成时间与相应的执行模式. 构造合格任务集：$E = \{x \in C \mid s_x = t, s_x \geqslant g_x\}$，并转向第四步.

第四步　任务开工期或完工期的调度.

· 如果此时 $E = \emptyset$，则转向第二步.

· 将所有的合格任务放入正进行的任务集：针对所有的 $j \in E$，设置 $f_{jm} = t + d_{jm}(s_{jm})$，$PS = PS \cup \{j\}$，以及 $S = S \cup \{j\}$；更新割集：$C = C \setminus E \cup \{x \mid x$ 是任务 $j \in E$ 的一个紧后任务，而 x 的所有紧前任务都属于 $PS\}$，同时设置这些新的割集任务的最早可能开工期为：$s_x = \max\{\max\{f_{ym} \mid (y,x) \in P_x\}, g_x\}$；针对每种可更新(再生)资源类型 r，在基本时间段 $(\tau - 1, \tau]$ 上检验是否 $\sum_{j \in s} w_{jm r\tau} \leqslant K_{rt}, \tau = t + 1, \cdots, \zeta, \forall r \in \mathrm{R}$. 其中 $\zeta = \max_{j \in s}\{f_{jm}\}$. 如果至少存在一种可更新(再生)资源 r，在其中某个基本时间段 $(\tau - 1, \tau]$ 上，正进行的所有任务对其的需求总和超过了资源最大供给量，我们就面临了一个资源冲突：转向第五步，否则转向第二步.

第五步　可更新(再生)资源冲突(导致可更新资源冲突的类型、严重程度、以及可加以选择的解决方案).

· 更新搜索树的分枝层：$p = p + 1$. 同时设置层类型标志 $\mathrm{flag}(p) = 1$；针对每种可更新(再生)资源类型 r，确定为解决资源冲突而必须释放多少单位的资源，即针对每种资源类型 r，以及有关的时间区间 $(\tau - 1, \tau]$，计算 $C_{r\tau} = \sum_{j \in s} w_{jmr\tau} - K_{rt}$，其中 $t + 1 \leqslant \tau \leqslant \zeta, \forall, \in \mathrm{R}, \zeta = \max_{j \in s}\{f_{jm}\}$；定义延滞集 $D(p) = \{D_q \mid D_q$ 是 S 的一个子集，针对所有可更新(再生)资源类型 $r, \forall, \in \mathrm{R}, \sum_{j \in D_q} w_{jmr\tau} \geqslant c_{r\tau}\}$，其中 $t + 1 \leqslant \tau \leqslant \zeta, \zeta = \max_{j \in s}\{f_{jm}\}$. D_q 又不包含另一个 $D_v \in D(p)$ 作为子集$\}$；针对每个 $D_q \in D(p)$，从产生可更新(再生)资源冲突的第一个 τ 起，确定以下四个量中的最小值，即可更新(再生)资源约束发生下一个变化的时刻、目前集合 $S \setminus D_q$ 中任务的可更新(再生)资源总需求发生下一个变化的时刻、如果 D_q 延迟而所有正进行任务的最早完成时间、以及紧前约束不属于 D_q 的所有未调度任务在所有可行模式下的最早可能完成时间，并将其赋予所谓的延迟点 w_q；针对每个延迟选择 D_q，按照 w_q 与每个任务 $j \in D_q$ 的所有剩余关键路径 q_j 中最大值的和，来计算基于紧前约束的下界 L_q；选择具有最小 L_b 的 $D_b \in D(p)$：下标 b 指明了在第 p 层中剩余的最好延迟选择；更新延滞集：$D(p) = D(p) \setminus D_b$；计算 $LB(p) = L_b$. 如果 $LB(p) \geqslant T$，减少分枝层数 $p = p - 1$ 并且转向第七

步.

· 存储任务完成时间 f_{jm} 以及相应的执行模式，部分调度方案 PS，正进行的任务集 S，割集 C 及其任务的最早可能开工期 S_x 以及相应的执行模式，决策点 t 和 t'、并转向第六步.

第六步　延迟(执行延迟选择).

· 在分枝上搜索一个新的节点.定义 DS 作为在时刻 t' 之前已经决定开工但目前决定必须延迟的任务集：$DS = \{j \in D_b \mid f_{jm} - d_{jm}(s_{jm}) < t'\}$；执行延迟选择：$PS = PS \setminus D_b$ 以及 $S = S \setminus D_b$；更新割集：$C = C \cup D_b \setminus \{y \mid x \in D_b$ 并且 $(x, y) \in P_y\}$；针对所有的 $j \in D_b$，更新最早可能开工期 $s_{jm} = w_b$；针对每个任务 $j \in D_b$，在对应的模式下确定了其开工期 s_{jm}，如果存在一个任务 $j \in D_b$、其对应的 $s_{jm} + d_{jm}(s_{jm}) \geqslant h_j$、则转向第七步(回溯).

· 如果 DS 非空，援引左移支配规则如下.如果延迟选择中的一个任务 j 在搜索树中以前的某一层上被选中，从而迫使其在时刻 t' 才成为合格任务，如果任务 j 在时刻 t' 就开工，同时如果延迟执行任务集合 DS 可在不导致资源冲突的情况下容许任务 j 左移，则这一调度方案就受到支配，转向第七步(回溯).

· 更新 $t' = w_b$，并转向第二步.

第七步　回溯.

· 如果分枝层数 $p = 0$，则停止.

· 如果在这一层上没有遗留的选择(延迟选择或模式配置选择)，即如果当 $\text{flag}(p) = 1$ 时，$D(p) = \varnothing$ 或当 $\text{flag}(p) = 0$ 时，$MA(p) = \varnothing$；设置 $p = p - 1$ 并且重复第七步.

· 如果当 $\text{flag}(p) = 1$ 时，在这一层上的遗留延迟选择中，选择具有最小下界 L_b 的延迟 $D_b \in D(p)$，并更新延滞集 $D(p) = D(p) \setminus D_b$；计算 $LB(p) = L_b$.如果 $LB(p) \geqslant T$，减少分枝层数 $p = p - 1$ 并且重复第七步.

· 恢复任务完成时间 f_{jm} 以及相应的执行模式，部分调度方案 PS，正进行的任务集 S，割集 C 及其任务的最早可能开工期 s_x 以及相应的执行模式，决策点 t 和 t'.转向第六步.

· 当 $\text{flag}(p) = 0$ 时，以割集任务总工期最短或最长原则排列遗留的可供选配的执行模式，并将选中的配置模式从可行的执行模式配置集 $MA(p)$ 中除去；恢复任务完成时间 f_{jm} 以及相应的模式，部分调度方案 PS，正进行的任务集 S，割集 C 及其任务的最早可能开工期 s_x，决策点 t 和 t'，不可更新(再生)资源的约束范围.转向第二步.

4　示例分析(Example)

考虑企业动态联盟形成过程.假设所有联盟企业拟共同完成的项目网络图如图 1 所示，其中以节点表示每个任务.其中任务 2,4 和 6 各分别有两个企业竞标，有关这些任务的可选配执行模式(潜在的可选择的合作伙伴)，根据投标企业各自的投标书，列出每种模式所对应的任务工期、资源使用与消耗量，如表 1 所示.每种资源的约束情况也在表 1 的附加说明和表 2 中给出.这一节将简要说明如何采用本文提出的算法计算求解这个 MRCPSP 示例的主要过程.

表 1　示例的有关数据

Table 1　Relevant data of the example

j	1	2		3	4	5		6		7
m	1	1	2	1	1	2	1	1	2	1
d_{jm}	0	4	6	2	3	5	2	2	4	0
w_{jm1i}	0	2,2	1,1,2	1,1	2,2	1,1	1,1	3,2	1,1	0
		2,3	2,1,1		2	1,1,1			2,2	
k_{jm2}	0	3	1	0	4	2	0	3	2	0

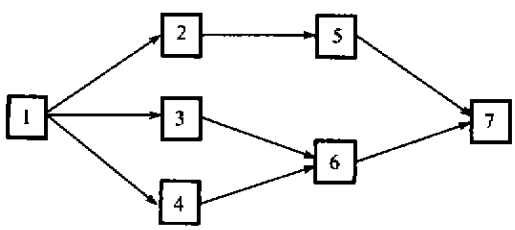

图 1　示例的网络图

Fig. 1　Network of the example

表 2　示例中可更新资源的最大供应量与时间的关系

Table 2　Dynamical availability of the renewable resource in the example

$(t-1,t)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
K_{1t}	4	4	3	3	4	4	3	3	4	4	3	3	4	4	3

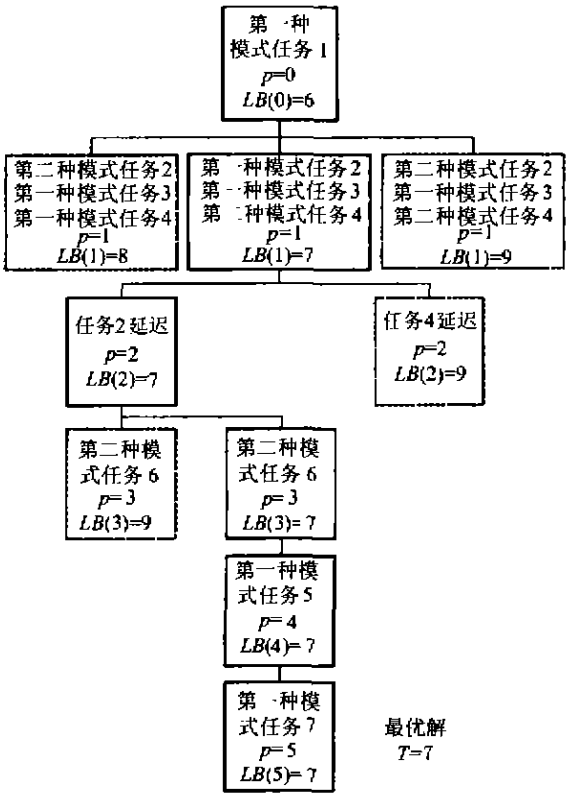

图2 分枝定界搜索树

Fig. 2 The search tree of the branch-and-bound algorithm

其中 $R = |1|, N = |2|, K_2 = 8, K_{1t}$ 如表2所示.利用本文所提的算法,得到并确证了最优调度方案.它的项目总工期是 $T = 7$. 算法执行过程的分枝定界搜索树如图2所示.毫无疑问,这一算法对虚拟企业的形成具有明确的决策支持作用.

5 结论(Conclusion)

本文与前人研究的问题有所不同,作者们结合工程应用的实际需要,探讨了项目中每个任务的工期不仅取决于自身的执行模式,而且取决于该任务实际开工时间的一般情形.此外,我们还考虑了项目中每个任务对可更新(再生)资源需求呈任意分布而非经典的均匀分布,以及可更新(再生)资源的最大供给量随时间而变化而非经典的时不变之一般情况.作为针对前人已有研究结果的进一步推广,我们在经典的单模式DH分枝定界算法的基础上,利用事件驱动的时间增量方式,成功地获得了这种最一般项目调度问题的最优解.

参考文献(References)

[1] Kolisch R and Drexl A. Local search for nonpreemptive multi-mode resource-constrained project scheduling [J]. IIE Transactions, 1997, 29(9): 987 - 999

[2] Kolisch R, Sprecher A and Drexl A. Characterization and generation of a general class of resource-constrained project scheduling problems [J]. Management Science, 1995, 41(10): 1693 - 1703

[3] Demeulemeester E and Herroelen W S. A branch-and-bound procedure for the multiple resource-constrained scheduling problem [J]. Management Science, 1992, 38(12): 1803 - 1818

[4] Demeulemeester E and Herroelen W S. New benchmark results for the resource-constrained project scheduling problem [J]. Management Science, 1997, 43(11): 1485 - 1492

[5] Demeulemeester E and Herroelen W S. A branch-and-bound procedure for the generalized multiple resource-constrained scheduling problem [J]. Operations Research, 1997, 45(2): 201 - 212

[6] Blazewicz J, Lenstra J K and Rinnooy Kan A H G. Scheduling subject to resource constraints: classification and complexity [J]. Discrete Applied Mathematics, 1983, 5(1): 11 - 24

[7] Sprecher A, Kolisch R and Drexl A. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem [J]. European Journal of Operational Research, 1995, 80(1): 94 - 102

本文作者简介

毛 宁 1962年生.1985年、1988年毕业于西安交通大学机械工程系,获工学学士和硕士学位,现为广东工业大学机电学院副教授,硕士生导师.感兴趣的研究方向是:敏捷制造,并行工程.发表论文30余篇,获国家教委科技进步二等奖一项.

陈庆新 1963年生.1985年、1988年毕业于西安交通大学机械工程系,分别获工学学士和硕士学位,1991年毕业于西安交通大学系统工程专业,获工学博士学位,现为广东工业大学机电学院教授,硕士生导师.感兴趣的研究方向是:敏捷制造,并行工程.发表论文40余篇,获国家教委科技进步二等奖一项.

陈 新 1960年生.1982年长沙铁道学院机械制造专业毕业,获工学学士学位,1988年哈尔滨工业大学机械制造专业毕业,获工学硕士学位,1995年2月华中理工大学CAD中心毕业,获工学博士学位,现为广东工业大学机电学院教授,博士生导师.感兴趣的研究方向是:敏捷制造,并行工程.发表论文60余篇,获国家教委科技进步二等奖一项.