考虑资源转移时间的资源受限项目调度问题的算法

陆志强 [1] 刘欣仪 [1]

摘 要 现有项目调度问题的研究一般假设资源在任务间转移不需要时间, 但这一假设与很多实际情况不相符, 本文在资源受限项目调度问题 (Resource-constrained project scheduling problem, RCPSP) 中引入资源转移时间, 以最小化项目工期为目标, 建立了考虑资源转移时间的资源受限项目调度问题的数学模型. 为改善遗传算法在局部搜索能力方面的不足, 提出将分支定界法与遗传算法相结合, 构造了一种内嵌分支定界寻优搜索的遗传算法, 在保证算法全局搜索能力的前提下提升局部精确搜索能力. 同时, 对于遗传算法, 为了适应算法结构提出了一种基于任务绝对顺序的编码策略. 数据实验表明, 对于小规模问题可获得近似精确解, 对于大规模问题相较现有文献所提算法, 在算法求解精度上可提升 10 %.

关键词 项目调度, 资源受限, 资源转移时间, 内嵌分支定界的遗传算法

引用格式 陆志强, 刘欣仪. 考虑资源转移时间的资源受限项目调度问题的算法. 自动化学报, 2018, **44**(6): 1028−1036

DOI 10.16383/j.aas.2017.c160834

Algorithm for Resource-constrained Project Scheduling Problem With Resource Transfer Time

LU Zhi-Qiang[1] LIU Xin-Yi[1]

Abstract Most studies in the literature so far assume that resources can be transferred from one job to the other without any expense of time. However, in many practical cases, it takes time for resources to transfer from one job to another. In this paper, we introduce the resource-constrained project scheduling problem (RCPSP) considering resource transfer time, which aims at minimizing the makespan of the project. A linear model is established and a branch-and-bound embedded genetic algorithm is proposed. A new precedence-based coding method which adapts to the structure of the problem is also proposed. Comparative computational results reveal that the branch-and-bound embedded genetic algorithm improves about 10 % in terms of solution accuracy compared with the algorithm proposed in the literature.

Key words Project scheduling, resource constraint, resource transfer time, branch-and-bound embedded genetic algorithm

Citation Lu Zhi-Qiang, Liu Xin-Yi. Algorithm for resource-constrained project scheduling problem with resource transfer time. *Acta Automatica Sinica*, 2018, **44**(6): 1028−1036

经典资源受限项目调度问题 (Resource-constrained project scheduling problem, RCPSP) 假设任务在其紧前任务结束后即可开始, 但是这一假设与很多实际情形不符. 在许多工程项目尤其是大型项目 (例如飞机装配线生产) 中, 因为各工位所处的位置不同, 当一些稀缺资源 (例如高级技工或关键设备) 在不同工位间转移时需要消耗一定的转移时间. 本文以飞机装配线生产为研究背景, 将其抽象为资源受限项目调度问题, 同时考虑到飞机装配线生产问题中稀缺资源在不同的工位间转移需要消耗时间, 为了能够使理论模型处理这一实际情况, 本文研究带资源转移时间的资源受限项目调度问题, 分析转移时间对项目调度决策造成的影响, 并研究该类问题的建模与有效求解算法.

对于资源受限项目调度问题, 目前的求解算法主要有精确算法、构造型启发式算法和元启发式算法. 精确算法主要以分支定界法为主, 例如 Brucker 等[1] 和 Dorndorf 等[2]. 虽然精确算法能够求得问题的精确解, 但是计算时间较长, 无法适用于大规模问题, 而构造型启发式算法和元启发式算法能在可接受时间内求得问题的相对较优解. 构造型启发式算法以基于规则的启发式算法为主, 通常根据不同的任务优先规则确定任务优先级, 使用串行或并行调度生成机制获得调度结果, 例如 Klein[3] 和 Buddhakulsomsiri 等[4]. 元启发式算法方面, 多使用遗传算法和模拟退火算法等搜索算法, 采用任务优先级编码或任务列表编码, 通过搜索方法获取较优编码, 并使用串行或并行调度生成机制解码获取较优调度结果, 例如 Valls 等[5], Peteghem 等[6],

收稿日期 2016-12-23 录用日期 2017-04-07
Manuscript received December 23, 2016; accepted April 7, 2017
国家自然科学基金 (61473211, 71171130) 资助
Supported by National Natural Science Foundation of China (61473211, 71171130)
本文责任编委 王红卫
Recommended by Associate Editor WANG Hong-Wei
1. 同济大学机械与能源工程学院 上海 201804
1. School of Mechanical and Energy Engineering, Tongji University, Shanghai 201804

Cho[7] 等. 对于带资源转移时间的资源受限项目调度问题, 目前有极少文献进行了研究, Krüger 等[8] 最先提出了带资源转移时间的资源受限项目调度问题, 假设资源在任务间及多项目间转移都需要资源转移时间, 问题的求解中采用了基于规则的启发式方法. Krüger 等[9] 对资源传递过程中的资源进行了定义和分类, 分为被传递的资源和协助资源传递的辅助资源, 建立了带资源转移时间的资源受限项目调度问题的扩展模型. 宗砚等[10] 建立了一种考虑资源传递时间并以多项目总工期及各个项目工期的加权和最小为目标的多项目调度模型, 并提出一种基于三级启发式规则解码的改进遗传算法求解. 在带资源转移时间的项目调度问题中, 资源转移时间与任务顺序两者互相耦合, 转移时间会随任务顺序的不同发生变化, 而这种变化又会影响任务执行顺序的决策, 因此在决策任务顺序的同时应进一步考虑资源转移时间造成的影响. 虽然上述文献对带资源转移时间的问题进行了一些研究, 但是在求解问题时, 都只提出了简单的启发式思路, 未能对资源转移时间与作业顺序调度决策的相互影响做进一步分析, 仍然存在可以改进的空间.

综上所述, 目前的求解算法中, 精确算法无法求解大规模问题, 构造型启发式算法虽然求解效率快, 但是单一的规则无法适用于不同的问题结构, 相对精确算法和元启发式算法而言, 这一类算法往往无法求得较优的任务顺序. 而元启发式算法虽然通过搜索方法可以获得较优编码 (任务顺序), 但是在解码时多使用启发式方法进行解码, 例如经典的串、并行调度生成机制, 在求解的精度上仍有欠缺. 针对以上问题, 本文提出一种内嵌分支定界的遗传算法, 将遗传算法与分支定界法相结合, 使用遗传算法框架保证算法的广度搜索能力并避免枚举所有分支, 同时使用基于深度优先的分支定界法进行深度搜索, 有效提高解的质量. 同时, 本文分析了资源转移时间对于问题解的影响, 并由此提出了相应的支配规则并应用于分支定界算法, 从而提升分支搜索的效率.

1　问题描述及数学模型

带资源转移时间的资源受限项目调度问题可描述如下: 项目包含 J 项任务, 记任务集合为 $J=\{1,2,\times s,n\}$, 其中任务 1 和任务 n 为虚拟资源, 任意任务 $j\in J$ 需要使用 K 种不同的资源, 记资源种类集合为 $K=\{1,2,\times s,K\}$, 第 $k\in K$ 种资源的供应量为 R_k. 时间离散化, 记时间集合为 $T=\{0,1,\times s,T\}$, $t\in T$ 为离散时间点. 假定 0 时刻, 所有资源都存放在虚拟任务 1 处, 项目完成时资源存放至虚拟任务 n 处. 任务间具有时序约束关系, 资源 k 从任务 i 转移到任务 j 需要一定的资源转移时间 Δ_{ijk}. 设另一任务为 h, 假设资源转移时间满足三角形原则 $\Delta_{ijk}\le\Delta_{ihk}+\Delta_{hjk}$, 即资源从任务 i 直接转移到任务 j 的时间最短. 项目的目标是最小化项目总工期.

数学模型具体如下:

$$\text{目标函数:}\ \min F_n \tag{1}$$

约束条件:

$$\sum_{t=0}^{T-d_j} f_{jt}=1,\ \forall j\in J \tag{2}$$

$$F_j=\sum_{t=0}^{T-d_j}(t+d_j)\times f_{jt},\ \forall j\in J \tag{3}$$

$$F_j-F_i\ge d_j,\ \forall j\in J,\forall i\in JP_j \tag{4}$$

$$F_i+\Delta_{ijk}+d_j\le F_j+M\times(1-z_{ijk}),\quad \forall j\in J,\forall i\in JP_j,\forall k\in K \tag{5}$$

$$x_{ijk}\le z_{ijk}\times\min\{r_{ik},r_{jk}\},\quad \forall j\in J,\forall i\in JP_j,\forall k\in K \tag{6}$$

$$z_{ijk}\le x_{ijk},\ \forall j\in J,\forall i\in JP_j,\forall k\in K \tag{7}$$

$$\sum_{i\in JP_j}x_{ijk}=r_{jk},\ \forall j\in J,\forall k\in K \tag{8}$$

$$\sum_{j\in JS_i}x_{ijk}=r_{ik},\ \forall i\in J,\forall k\in K \tag{9}$$

$$\sum_{j=1}^{n}\sum_{\tau=t}^{t+d_j}r_{jk}\times f_{j\tau}\le R_k,\ \forall t\in T,\forall k\in K \tag{10}$$

$$f_{jt}\in\{0,1\},\ z_{ijk}\in\{0,1\},\ x_{ijk}\ge 0\quad \forall t\in T,\forall k\in K,\forall j\in J,\forall i\in J \tag{11}$$

式 (1) ∼ (11) 中的符号及变量定义如下: J 是任务集合, K 是资源集合, R_k 是资源 k 的资源供应量, T 是时间集合, P_j 是任务 j 的直接紧前任务集合, S_j^* 是任务 j 的直接与间接紧后任务集合, A_j^* 是任务 j 的直接与间接紧前任务集合, JP_j 是资源可供任务 j 使用的任务集合, $JP_j=J-\{j\}-S_j^*$, JS_j 是需求任务 j 的资源的任务集合, $JS_j=J-\{j\}-A_j^*$, Δ_{ijk} 是资源 k 从任务 i 转移到任务 j 的转移时间, F_j 是任务 j 的完成时间, d_j 是任务 j 的作业时间, r_{jk} 是任务 j 所需资源 k 的数量, M 是无穷大的参数.

f_{jt}, z_{ijk}, x_{ijk} 是决策变量, 如果任务 j 在时刻 t 开始, f_{jt} 为 1, 否则为 0; 如果资源 k 从任务 i 转移至任务 j, z_{ijk} 为 1, 否则为 0; x_{ijk} 为从任务 i 调配至任务 j 的资源 k 的数量.

式 (1) 为目标函数, 表示项目工期最小化; 式

(2) 表示任务只能在一个时间点开始; 式 (3) 表示任务的完成时间与决策变量 f_{jt} 的关系; 式 (4) 表示任务间的紧前关系, 在任意任务完成前, 其紧后任务不可开始; 式 (5) 表示任务开始时间与资源转移时间的关系; 式 (6) 表示资源转移数量不得超出任务所需数量; 式 (7) 约束决策变量 x_{ijk} 与 z_{ijk} 之间的关系; 式 (8) 表示所有转入的资源总量等于任务所需资源数量; 式 (9) 表示所有转出的资源量等于任务所拥有的资源量; 式 (10) 表示任意时刻所有任务所使用的资源量不得超出资源的总供应量; 式 (11) 定义了决策变量的可行域.

2 算法设计

本文设计了一种内嵌分支定界的遗传算法求解问题, 将遗传算法与基于深度优先的分支定界算法相结合, 图 1 给出了算法的具体流程图. 分支定界算法可以求得问题的最优解, 但搜索时间过长, 无法在规定的时间内求解大规模问题. 为解决该问题, 本文将分支定界法嵌入遗传算法框架, 因为遗传算法可以跳出局部搜索, 拥有很强的广度搜索能力. 本文使用遗传算法进行全局搜索, 再使用分支定界算法进行深度搜索, 保证了算法的广度及深度双重搜索能力. 在遗传算法编码方面, 本文提出了一种基于任务绝对执行顺序的编码方式, 不仅可用于遗传算法框架, 也可转化为分支定界法中的搜索树. 该编码与传统任务列表编码的不同在于, 在任务列表编码中, 任务的顺序前后依次排列, 而本文提出的编码允许任务并行排列, 该编码方式可以使得编码结构更为紧凑, 从而提升算法搜索的效率. 在此基础上, 进一步设计了相应的交叉和变异方法, 构建了启发式调度生成方法进行解码. 同时, 在分支定界算法方面, 本文分析了搜索树各分支之间的结构关系, 提出与证明了相应的支配规则, 用于剪除多余分支, 提高算法的搜索效率.

2.1 遗传算法

本文使用遗传算法框架对问题进行初步求解, 遗传算法具有很强的全局搜索能力, 可以保证求解结果的相对质量. 本节主要阐述一种适应本文算法结构的编码方式以及相应的交叉、变异方法和启发式解码方式.

2.1.1 基于任务绝对顺序的编码方式

本文设计了一种基于任务绝对顺序的编码方式, 将编码划分为多层, 编码需满足以下条件: 1) 后一层中不得包含前一层任务的直接或间接紧前任务; 2) 同一层中的任务不能有紧前关系; 3) 同一层中所有任务的资源总使用量必须小于资源的总供应量. 一个编码一旦完成, 其顺序代表了任务的绝对执行顺序, 只有前一层中的任务执行完后, 后一层中的任务才可开始. 事实上, 这种编码可由项目网络图转化而来, 图 2 给出了一个编码生成的例子.

从图 2 可以看出, 此编码第 1 层选择任务 1, 此时第 2 层可选择的任务为 2, 3, 4. 假定任务所需资源量为 $r_2 = 4$, $r_3 = 3$, $r_4 = 3$, 资源总上限 $R = 6$, 根据编码性质, 每层中的任务资源使用总和不得超过资源总上限, 因此此处可选择编码为 $\{2\}$, $\{3\}$, $\{4\}$ 或 $\{3, 4\}$. 通过随机方式选择第 2 层编码为 $\{3, 4\}$, 因此任务 2 延后且必须在任务 3, 4 之后执行, 需要为其添加虚拟约束. 此时第 3 层可选择的任务为 $\{2\}$, $\{6\}$ 或 $\{2, 6\}$, 随机选择第 3 层编码为 $\{2, 6\}$. 剩余编码根据上述方法继续分层, 直至完成整体编码.

图 3 是对应上述项目网络图及编码的一个实例, 包括具体的任务所需资源量、作业时间及资源转移时间. 为了便于理解, 此处假设该实例只有一种资源, 甘特图中灰色部分表示资源转移时间. 同时, 由于

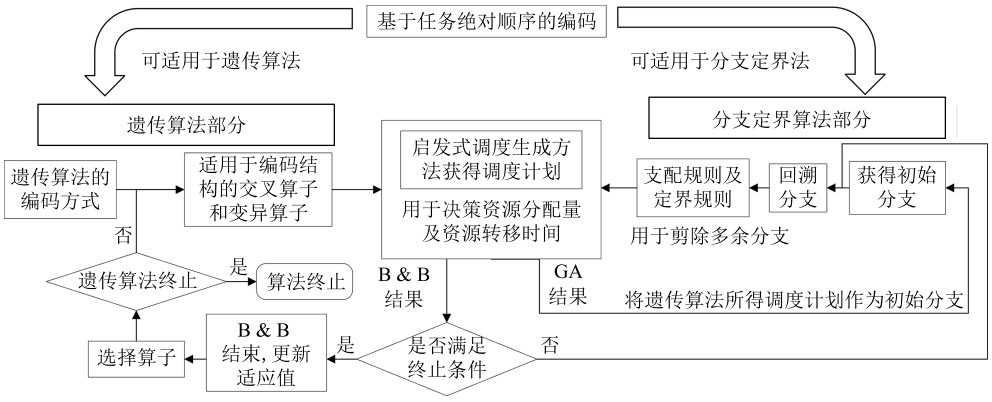


图 1 算法流程图

Fig. 1 Flowchart of the algorithm

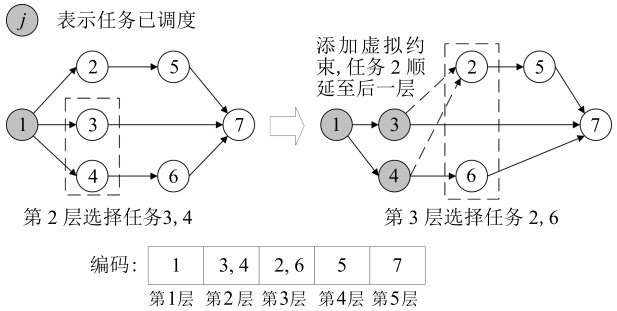


图 2　编码的生成方式

Fig. 2　The generation of the coding

转移时间	2	3	4	5	6
2	0	4	5	2	5
3	2	0	5	4	6
4	5	3	0	5	4
5		4	5	0	6
6	3	3		2	0

作业	资源量	工期	资源供应量
2	4	12	6
3	3	14	
4	3	15	
5	4	13	
6	2	14	

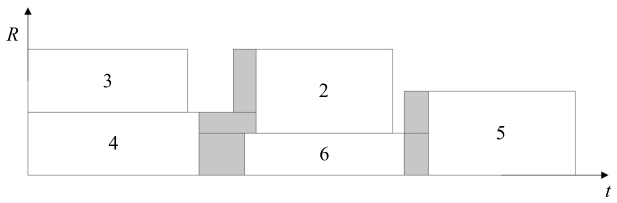


图 3　带资源转移时间的资源受限项目调度问题实例

Fig. 3　An example of the RCPSPTT

任务 2, 5 及任务 4, 6 之间存在紧前约束关系, 因此资源无法从任务 5 转移至任务 2, 也无法从任务 6 转移至任务 4, 所以不存在资源转移时间 Δ_{52} 和 Δ_{64}.

2.1.2　交叉与变异

为保证交叉后编码的可行性, 本文设计了一种单点交叉方法. 随机选取交叉点, 子代中交叉点前的编码顺序遵循父代, 从母代中将已选取的任务剔除, 交叉点后的编码顺序遵循剔除任务后的母代, 图 4 给出了一个单点交叉的例子. 通过此交叉方式得到的编码仍然满足时序约束及资源约束.

变异方法使用对换变异, 如果编码的前后两层之间的所有任务都无紧前约束, 则对换两层编码. 图 5 给出了一个变异的例子, 通过此变异方法得到的编码仍然满足时序约束及资源约束.

2.1.3　启发式调度生成方法

本文使用启发式调度生成方法对相应的染色体 (编码) 进行解码, 以获得对应该染色体的初始解. 通过决策具体的资源分配方案, 确定任务所需的资源转移时间. 具体决策方法是: 按照层级从前往后选择任务, 对于每层编码中的任务, 按照任务编号从小到大选择, 按尽早开始原则决策资源从哪项任务转移而来, 并确定相应的转移量和资源转移时间. 按此方法逐步确定每层中任务的开始时间, 获得完整的调度计划. 具体的算法步骤如下:

步骤 1. 令 $g = 1$.

步骤 2. 读取第 g 层编码中的任务集合 A_g.

步骤 3. 按照任务编号从小到大选择任务 $j \in A_g$, 更新 $A_g = A_g - \{j\}$.

步骤 4. 按尽早开始原则为任务 j 分配资源, 计算开始时间, 更新分配后的资源占有量.

步骤 5. 如果 $A_g \neq \emptyset$, 转步骤 3.

步骤 6. 如果调度计划未完成, $g = g + 1$, 转步骤 2; 否则结束算法.

2.2　分支定界算法

本文设计了一种分支定界算法, 以遗传算法所得调度计划为起始, 将它视为在搜索树上进行深度搜索的一个初始分支, 通过回溯方法搜索其邻域解, 优化项目调度结果. 以下主要阐述分支方法以及支配规则和定界规则.

2.2.1　分支方法

分支定界算法以深度优先, 搜索树的结构与遗传算法的编码结构相同, 只是将编码中的层级关系转化为了搜索树中的树节点关系. 为控制分支定界算法的搜索时间和搜索精度, 本文设定了搜索终止条件, 经过数据实验确定终止条件为: 对遗传算法每一条染色体的回溯层数达到 5 层或搜索时间达到 0.5 秒时, 停止算法. 通过设定分支定界搜索终止条件, 间接设定了搜索的邻域, 即初始分支回溯 5 层范围内的节点. 具体分支步骤如下:

步骤 1. 令 $g = 1$.

步骤 2. 确定节点 g 的满足资源约束的候选任务组合集合 C_g.

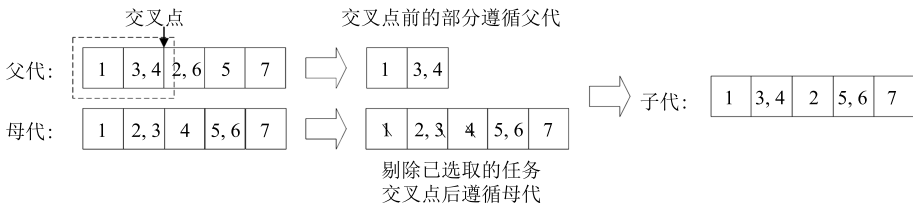


图 4　交叉方式

Fig. 4　An example of crossover

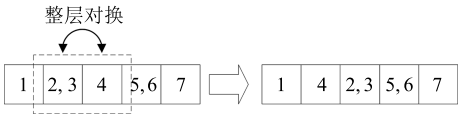


图 5 变异方式

Fig. 5 An example of mutation

步骤 3. 从 C_g 中随机选择任务集合 A_g, 更新 $C_g = C_g - A_g$.

步骤 4. 如果 $n \notin A_g$, 则继续分支, $g = g + 1$. 否则回溯搜索树, 如果找到某节点 g 的 $C_g \neq \emptyset$, 则从该节点开始分支, $g = g + 1$.

步骤 5. 如果所有节点的 $C_g = \emptyset$ 或回溯层数达到 5 层或搜索时间达到 0.5 秒, 则分支结束.

图 6 给出了一个搜索树的例子, 包括任务的具体资源量与资源总上限对应的项目网络图及编码如图 2 所示.

2.2.2 支配规则及定界规则

本文提出三种不同的支配规则, 用以剪除多余分支, 提升分支定界算法的效率.

定义 1. 给定当前节点 g, 如果节点 g 从节点 g' 分出, 则节点 g' 称为节点 g 的直接根节点, 节点 g 称为节点 g' 的子节点.

记当前节点为 g, 其直接根节点为 g', 直接根节点调度的任务集合为 $A_{g'}$, 当前节点调度的任务集合为 A_g, 所有任务 $i \in A_{g'}$ 的资源 k 的使用量之和为 $R_k^{g'}$, 所有任务 $j \in A_g$ 的资源 k 的使用量之和为 R_k^g, 资源 k 的总上限为 R_k.

规则 1. 如果当前节点的所有任务均可以合并入其直接根节点, 且合并之后的调度计划与合并前相同, 则当前节点视为重复情况, 不再分支.

为了提升算法效率, 避免计算完整调度计划, 本文对规则 1 所述的重复情况进行了分析并得出结论: 当满足以下任意条件时, 上述重复情况不会发生, 即只要不满足以下所有条件, 则当前节点不再分支.

条件 1. 对于任意资源 k, 如果满足条件 $R_k^{g'} + R_k^g > R_k$, 则当前节点不为重复情况.

证明. 如果满足 $R_k^{g'} + R_k^g > R_k$, 则表明两节点中所有任务的资源使用总量大于资源的总供应量, 当前节点中的任务无法全部合并入其直接根节点, 因此当前节点不为重复情况. □

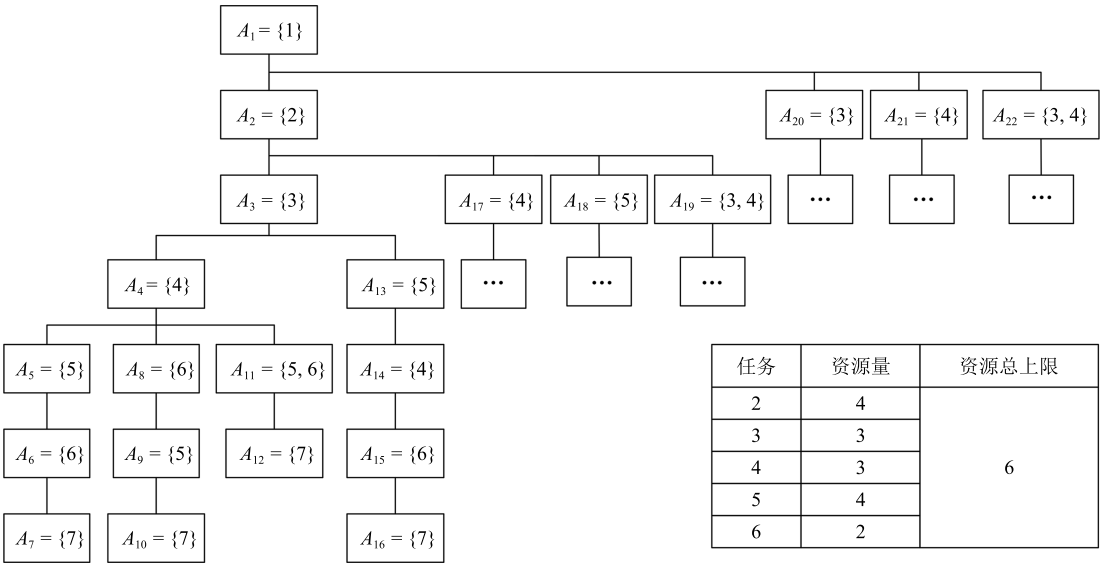
条件 2. 如果任务 $j \in A_g$ 与任意任务 $i \in A_{g'}$ 存在时序约束关系, 则当前节点不为重复情况.

证明. 如果两节点中的任务间存在时序约束关系, 则当前节点调度的任务无法全部合并入其根节点, 因此当前节点不为重复情况. □

条件 3. 在不满足条件 1 和条件 2 的情况下, 如果对于任意任务 $i \in A_{g'}$ 和 $j \in A_g$ 满足 $i > j$, 则当前节点不为重复情况.

证明. 根据分支结构, 任务不合并时应该先为任务 i 分配资源; 任务合并后, 如果 $i > j$, 根据启发式调度生成方法, 按照任务编号从小到大依次分配资源, 那么应该先为任务 j 分配资源, 因此两种情况的调度结果不同, 当前节点不为重复情况, 具体可参考图 7. □

图 7 中, 任务 5 和 6 之间的关系不满足条件 1 和 2, 即两者可放入同一节点执行. 在最左侧的分支中, 先执行任务 5 后执行任务 6, 资源分配顺序与最右侧的分支相同, 因此最终调度结果相同, 为重复节点; 而在中间的分支中, 按照分支的顺序, 先执行任务 6 后执行任务 5, 因此与左侧和右侧的分支均不相



任务	资源量	资源总上限
2	4	6
3	3	
4	3	
5	4	
6	2	

图 6 搜索树示例

Fig. 6 An example of the search tree

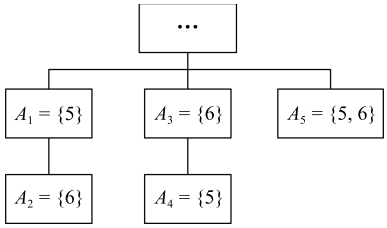


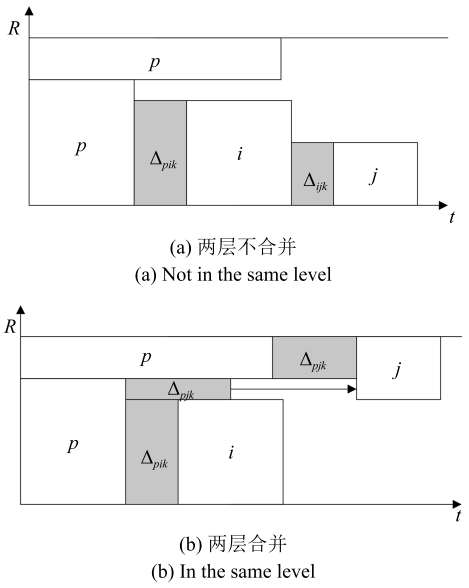
图 7　条件 3 示例图

Fig. 7　An example of condition 3

同, 所以不为重复情况.

条件 4. 如果不满足条件 1 ∼ 3, 计算任务 $j \in A_g$ 的最早开始时间 EST_j. 如果对于任意任务 $i \in A_{g'}$ 和任意资源 k, 满足 $F_i + \Delta_{ijk} \le EST_j$, 则当前节点不为重复情况.

证明. 如果不满足条件 1 和条件 2, 则表明当前节点的所有任务都可合并入其根节点. 任务不合并时, 任务 j 可使用任务 i 的资源, 如果对于任意任务 $i \in A_{g'}$ 和任意资源 k, 满足条件 $F_i + \Delta_{ijk} \le EST_j$, 则表明当任务 j 使用任务 i 的资源时, 调度结果最好. 如果将两节点的任务合并, 任务 i 与任务 j 会变为并行关系, 任务 j 不使用任务 i 的资源, 其开始时间必将大于最早开始时间, 所以合并前和合并后的调度计划必不相同, 因此不为重复情况, 具体可参考图 8. □



(a) 两层不合并

(a) Not in the same level

(b) 两层合并

(b) In the same level

图 8　条件 4 示例图

Fig. 8　An example of condition 4

图 8 给出了一个条件 4 的示例, 图中任务 $i \in A_{g'}$, 任务 $j \in A_g$, 两者无紧前关系且满足 $R_k^{g'} + R_k^g \le R_k$, 图 8 (a) 表示任务未合并时的情形, 此时任务 j 的最早开始时间 $EST_j = F_i + \Delta_{ijk}$, 满足 $F_i + \Delta_{ijk} \le EST_j$. 图 8 (b) 表示任务合并后的情景, 此时任务 i 和任务 j 属于同一节点, 任务 j 无法使用任务 i 的资源, 任务所需的资源需从任务 p 调配, 此时 $ST_j > EST_j$, 合并前和合并后的调度计划不相同, 不为重复情况.

规则 2. 如果当前节点的任务的完成时间超过目前所得最好解, 则该节点不再分支.

记目前所得的项目最短工期为 UB. 如果任务 $j \in A_g$ 的完成时间 $F_j > UB$, 则节点 g 不再分支. 新的完整调度计划完成时, 如果当前 $F_n < UB$, 则更新 $UB = F_n$.

规则 3. 如果当前节点的下界 LB_g 大于目前所得最好解, 则当前节点不再分支, 下界 LB_g 的计算可使用式 (12) ∼ (14).

本文提出一种考虑时序约束和资源转移时间的动态下界, 以提升分支定界算法的效率. 在计算下界时, 需要将未调度任务区分为当前节点可开始的候选任务及剩余未调度任务, 并分别计算其最早开始时间, 图 9 给出了一个具体示例.

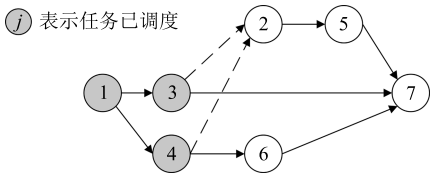


图 9　任务区分示例

Fig. 9　An example of the classifications of the activities

在图 9 中, 任务 1, 3, 4 都已调度, 需要将未调度任务进行区分. 根据图中的紧前约束可得, 当前节点可开始的候选任务为任务 2, 6, 剩余未调度任务为任务 5, 7. 定义当前可提供资源的任务集合为 A, 任务 j 的最早完成时间 $EFT_j = EST_j + d_j$, 当前节点的下界为

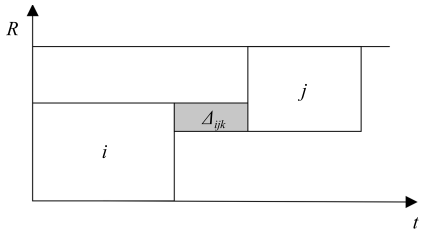
$$LB_g = EST_n \tag{12}$$

对于当前节点可开始的候选任务, 在计算其最早开始时间时考虑具体资源分配, 计算方式为

$$EST_j = \min\{\max\{F_i + \Delta_{ijk} | k \in K\} | i \in A \text{ 且 } \sum_i x_{ijk} = r_{jk}\} \tag{13}$$

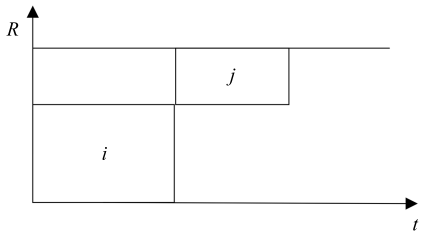
对于剩余未调度任务, 如果任务 j 及其直接紧前任务 i 满足条件 $r_{ik} + r_{jk} > R_k$, 则表示任务 j 必须使用任务 i 的资源 k, 资源转移时间 Δ_{ijk} 必然存在, 具体可参考图 10.

定义集合 P_j^* 为任务 j 必须使用的任务集合, 集合 K^* 为必须使用的资源集合. 任务 j 的最早开始时间计算方式为



(a) 任务 j 必须使用 i 的资源

(a) Resource must be transferred from i to j



(b) 任务 j 不必使用 i 的资源

(b) Resource do not have to be transferred from i to j

图 10 任务关系示例

Fig. 10 An example of the relationship of the activities

$$EST_j=\begin{cases}\max\{EFT_i+\Delta_{ijk}|i\in P_j^*\text{且}k\in K^*\},\\ \qquad\qquad r_{ik}+r_{jk}>R_k\\ \max\{EFT_i|i\in P_j\},\\ \qquad\qquad r_{ik}+r_{jk}\le R_k\end{cases}$$

2.2.3 分支定界算法步骤

分支定界算法的具体步骤如下:

步骤 1. 初始分支为遗传算法所得调度计划.

步骤 2. 回溯搜索树, 继续分支, $g=g+1$, 并更新已调度任务集合 S_g.

步骤 3. 根据支配规则和定界规则判断当前节点是否能够剪除, 如果可以则转步骤 5.

步骤 4. 调用启发式调度生成方法生成项目调度结果.

步骤 5. 如果 $n\notin S_g$, 转步骤 2; 如果 $n\in S_g$, 比较任务完成时间, 如果 $F_n<UB$, 更新 $UB=F_n$, $g=g+1$.

步骤 6. 满足终止条件则算法结束.

3 数据试验

本文使用 Microsoft Visual Studio 2013 进行编程, 通过实验的方式对上述算法的有效性进行检验.

3.1 遗传算法参数选取

本文算法测试的基础算例随机选自 PSPLIB 算例库[11], 资源转移时间通过 C# 随机生成, 存储在算例文件中, 取值范围在 1 ∼ 15 之间, 符合三角形规则. 为了在遗传算法中选取合适的参数, 本文在不同参数设置情况下对不同规模的算例进行求解. 分别选取交叉概率 pc 为 0.8, 0.6 及变异概率 pm 为 0.2, 0.1 进行数据测试. 表 1 给出了不同规模下的实验结果对比, 其中每组包含 30 个算例, 平均值偏差表示的是每组算例所得结果的均值与本组最好均值之间的差值百分比.

由表 1 可知, 在参数设置为 $pc=0.8$, $pm=0.1$ 的情况下, 其平均偏差与最好解相差最小, 且在不同算例情况下与最好解的差值都相差较小, 因此本文遗传算法最终选取交叉概率为 0.8, 变异概率为 0.1, 以下部分数据对比均使用本组参数设置.

3.2 与 CPLEX 的对比

为了验证本文所提内嵌分支定界的遗传算法的有效性, 将算法的求解结果与 CPLEX 所求精确解以及无内嵌分支定界的遗传算法所得结果进行对比. 表 2 给出不同算法对于小规模算例的求解结果, 每组均包含 10 个算例. 表 2 中 GAP 栏表示算法所得解与 CPLEX 所得上界间的差值百分比, 设定 CPLEX 的最长计算时间为 7 200 s.

由表 2 可知, 内嵌分支定界的遗传算法可以在可接受时间内求得较优结果, 相较无内嵌分支定界的遗传算法, 内嵌分支定界的遗传算法在求解所有小规模算例时均可优化 2 % 左右. 对于 J10 和 J12 规模下的算例, CPLEX 可以求得最优解, 内嵌分支

表 1 不同参数设置下的实验结果对比

Table 1 Comparison of experimental results under different parameter settings

	平均值偏差 (%)			
算例规模	$pc=0.8$ $pm=0.2$	$pc=0.6$ $pm=0.2$	$pc=0.6$ $pm=0.1$	$pc=0.8$ $pm=0.1$
J30	0.89	0.99	1.44	0.00
J60	1.11	0.00	0.08	0.13
J90	0.66	0.00	0.13	0.24
平均	0.87	0.33	0.55	0.12

表 2　不同算法求解小规模算例的实验结果对比

Table 2　Comparison of different algorithms in small-to-medium size problems

算例规模	组别	CPLEX			无分支定界的遗传算法			内嵌分支定界的遗传算法		
		上界均值	下界均值	时间均值 (s)	均值	时间均值 (s)	GAP (%)	均值	时间均值 (s)	GAP (%)
J10	1	34.9	34.9	86.00	35.8	1.44	2.52	35.2	17.85	0.74
	2	23.4	23.4	19.01	24.8	1.30	5.98	23.6	25.97	0.85
	3	37.9	37.9	102.35	38.4	1.31	1.32	38.1	16.14	0.53
	平均			69.12		1.35	3.27		19.99	0.71
J12	1	62.1	60.6	4 035.60	64.1	1.72	3.22	62.7	20.46	0.97
	2	41.3	39.3	3 233.42	42.3	1.74	2.42	41.7	19.10	0.97
	3	27.1	27.1	30.69	28.1	1.68	3.69	27.3	24.25	0.74
	平均			2 429.90		1.71	3.11		21.27	0.89
J14	1	62.8	49.0	5 898.80	64.2	2.17	2.23	61.7	35.70	−1.75
	2	32.1	30.2	2 335.54	33.4	2.21	4.05	32.5	39.12	1.25
	3	47.1	46.2	950.73	48.6	2.67	3.18	47.7	39.73	1.27
	平均			3 061.69		2.35	3.15		38.18	0.26
J16	1	78.7	53.6	7 200.00	77.7	2.33	−1.27	75.7	44.38	−3.81
	2	39.8	30.1	6 484.84	43.6	3.08	9.55	42.4	41.76	6.53
	3	59.9	46.1	6 509.60	61.5	2.70	2.67	59.8	56.31	−0.17
	平均			6 731.48		2.70	3.65		47.48	0.85
J20	1	49.4	28.8	7 200.00	48.3	3.12	−2.23	46.9	59.68	−5.06
	2	64.4	41.0	7 200.00	63.0	3.23	−2.17	61.1	59.80	−5.12
	3	68.1	46.0	7 200.00	67.8	3.09	−0.44	65.6	72.30	−3.67
	平均			7 200.00		3.15	−1.61		63.93	−4.62

定界的遗传算法所得解与最优解之间的 GAP 差值均在 1 % 以下. 对于 J14 ∼ J20 规模下的算例, 由于问题较为复杂, CPLEX 有时无法求得最优解, 此处取 CPLEX 运行 2 小时所得上界为对比对象, 将本文算法所得结果与 CPLEX 所得最好解进行对比. 对于 J14 和 J16 规模下的算例, 通过内嵌分支定界可以将遗传算法与最好解间的 GAP 均值从 3 % 左右缩减至小于 1 %, 甚至在有些个别算例中可以将 GAP 从 5.98 % 缩减至 0.85 %; 对于 J20 规模下的算例, 相比上界, 内嵌分支定界的遗传算法能够使得解的质量提高 5 % 左右.

3.3　与现有文献的对比

由于 CPLEX 无法求解大规模问题, 所以本文选取 Krüger 等[8] 所提的基于规则的改进并行调度算法为对比对象, 验证本文所提算法的有效性. 经数据实验分析, Krüger 等[8] 认为 LFT 规则和 SLACK 规则可以求得较优解, 因此此处选取运用 LFT 规则与 SLACK 规则所得的结果作为对比对象. 表 3 给出不同算法对于大规模问题的实验结果对比, 其中每组包括 10 个算例, 表 3 中的 GAP 栏表示不同算法所得解与本组最好解间的差值.

由表 3 可知, 相较无内嵌分支定界的遗传算法, 内嵌分支定界的遗传算法在求解大规模算例时均可优化 2 % 左右. 遗传算法本身具有通过交叉迭代不断优化结果的特性, 因此其求解质量本身就已经可以达到一定精度, 虽然内嵌分支定界的遗传算法耗费时间相对较长, 但 2 % 的改进仍具有意义. 算法求解大规模算例时所用平均求解时间在 30 分钟左右, 在可接受的范围之内. 同时, Krüger 等[8] 所提算法的求解结果与内嵌分支定界的遗传算法所得结果的偏差值为 10 % 左右, 证明了本文所提算法的优越性.

4　结论

本文提出一种内嵌分支定界的遗传算法求解带有资源转移时间的资源受限项目调度问题, 算法使用遗传算法保证全局搜索能力, 同时使用基于深度优先的分支定界算法优化局部邻域搜索能力, 基于对问题结构的研究提出并证明了相应的支配规则, 并提出了问题的动态下界, 提升了分支定界算法的搜索效率. 通过数据实验验证了本文所提内嵌分支

表 3 不同算法求解大规模算例的实验结果对比

Table 3 Comparison of different algorithms in large size problems

算例规模	组别	启发式算法				无分支定界的遗传算法			内嵌分支定界的遗传算法		
		LFT 均值	GAP (%)	SLACK 均值	GAP (%)	均值	时间均值 (s)	GAP (%)	均值	时间均值 (s)	GAP (%)
J30	1	159.0	14.80	156.9	13.29	140.7	5.69	1.59	138.5	63.65	0.00
	2	157.6	16.65	161.9	19.84	136.7	6.27	1.18	135.1	77.20	0.00
	3	120.1	12.77	119.3	12.02	109.1	6.22	2.44	106.5	117.81	0.00
	平均		14.74		15.05		6.06	1.74		86.22	0.00
J60	1	217.0	12.55	216.0	12.03	196.7	24.14	2.02	192.8	744.97	0.00
	2	152.4	11.00	151.0	9.98	140.4	26.58	2.26	137.3	777.18	0.00
	3	115.8	8.22	114.9	7.38	109.2	25.77	2.06	107.0	798.22	0.00
	平均		10.59		9.80		25.50	2.11		773.46	0.00
J90	1	383.8	11.05	386.3	11.78	351.4	68.89	1.68	345.6	1 191.98	0.00
	2	199.2	9.15	198.8	8.93	186.8	68.18	2.36	182.5	1 952.75	0.00
	3	520.9	14.38	515.9	13.29	462.6	71.57	1.58	455.4	1 755.12	0.00
	平均		11.53		11.33		69.55	1.87		1 633.28	0.00

定界的遗传算法的有效性. 在求解小规模问题时, 算法所得解与最优解间的差值均在 1 % 以下; 在求解大规模问题时, 相较现有文献的启发式算法, 内嵌分支定界的遗传算法可以提高解的精度近 10 %.

References

1 Brucker P, Knust S, Schoo A, Thiele O. A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 1998, **107**(2): 272−288

2 Dorndorf U, Pesch E, Phan-Huy T. A branch-and-bound algorithm for the resource-constrained project scheduling problem. *Mathematical Methods of Operations Research*, 2000, **52**(3): 413−439

3 Klein R. Bidirectional planning: improving priority rule-based heuristics for scheduling resource-constrained projects. *European Journal of Operational Research*, 2000, **127**(3): 619−638

4 Buddhakulsomsiri J, Kim D S. Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *European Journal of Operational Research*, 2007, **178**(2): 374−390

5 Valls V, Ballestín F, Quintanilla S. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 2008, **185**(2): 495−508

6 Peteghem V, Vanhoucke M. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 2010, **201**(2): 409−418

7 Cho J, Kim Y. A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of the Operational Research Society*, 1997, **48**(7): 736−744

8 Krüger D, Scholl A. A heuristic solution framework for the resource constrained (multi-)project scheduling problem with sequence-dependent transfer times. *European Journal of Operational Research*, 2009, **197**(2): 492−508

9 Krüger D, Scholl A. Managing and modelling general resource transfers in (multi-)project scheduling. *OR Spectrum*, 2010, **32**(2): 369−394

10 Zong Yan, Liu Qiong, Zhang Chao-Yong, Zhu Hai-Ping. Multi-project scheduling problem with resource transfer time. *Computer Integrated Manufacturing Systems*, 2011, **17**(9): 1921−1928
(宗砚, 刘琼, 张超勇, 朱海平. 考虑资源传递时间的多项目调度问题. 计算机集成制造系统, 2011, **17**(9): 1921−1928)

11 Kolisch R, Sprecher A. PSPLIB — A project scheduling problem library: OR Software-ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research*, 1996, **96**(1): 205−216

陆志强 同济大学教授. 2003 年获得法国南特大学博士学位. 主要研究方向为物流与供应链管理. 本文通信作者.
E-mail: zhiqianglu@tongji.edu.cn
(**LU Zhi-Qiang** Professor at Tongji University. He received his Ph. D. degree from Universite De Nanets, France in 2003. His research interest covers logistics and supply chain management. Corresponding author of this paper.)

刘欣仪 同济大学硕士研究生. 2015 年获得同济大学学士学位. 主要研究方向为项目调度.
E-mail: liuxinyi@tongji.edu.cn
(**LIU Xin-Yi** Master student at Tongji University. She received her bachelor degree from Tongji University in 2015. Her main research interest is project scheduling.)