计算机科学
COMPUTER SCIENCE

求解资源受限项目调度问题的分支定价算法

张宇哲，董兴业，周正

引用本文

张宇哲，董兴业，周正.求解资源受限项目调度问题的分支定价算法[J]．计算机科学, 2022, 49(12): 274-282.

ZHANG Yu-zhe, DONG Xing-ye, ZHOU Zheng. Branch & Price Algorithm for Resource-constrained Project Scheduling Problem [J]. Computer Science, 2022, 49(12): 274-282.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211100276

求解资源受限项目调度问题的分支定价算法

张宇哲[1]　董兴业[1]　周　正[2]
1 北京交通大学计算机与信息技术学院交通数据分析与挖掘北京市重点实验室　北京 100044
2 中国船舶工业系统工程研究院　北京 100094
(19120448@bjtu.edu.cn)

摘　要　资源受限项目调度问题是最具代表性的一类项目调度问题,是很多实际调度问题的抽象表示,属于 NP-hard 问题,对于大规模问题难以求得全局最优解。文中提出了问题的整数规划模型,通过将模型分解为约束主问题和子问题设计了求解线性松弛模型的列生成方法,然后通过分支定价寻找问题的整数解。在求解过程中引入松弛变量解决模型伪不可解的问题,设计了剪支策略和分支策略,并根据不同情况提出了两种缩小解空间的方法。在 PSPLIB 基准数据集上,对于有 30 个工序的问题,所提算法在 10m 内能够求出 480 个问题中的 301 个问题的当前最优解;对于有 60 个工序的问题,在 20m 内能够求出 480 个问题中的 269 个问题的当前最优解;对于有 90 个工序的问题,在 30m 内能够求出 480 个问题中的 263 个问题的当前最优解。同时,算法使用缩小解空间策略后,超时算例的个数明显减少,优化初始解的性能得到明显提升。以上实验结果表明,所提算法具有较好的求解能力。

关键词:资源受限项目调度;整数规划;列生成;分支定价;伪不可解
中图法分类号　TP181

Branch & Price Algorithm for Resource-constrained Project Scheduling Problem

ZHANG Yu-zhe[1],DONG Xing-ye[1] and ZHOU Zheng[2]
1 Beijing Key Lab of Traffic Data Aanlysis and Mining,School of Computer and IT,Beijing Jiaotong University,Beijing 100044,China
2 Systems Engineering Research Institute,Beijing 100094,China

Abstract　Resource-constrained project scheduling problem(RCPSP) is the most representative scheduling problem,which is an abstract representation of many practical scheduling problems. It belongs to the NP-hard problem and is difficult to obtain the global optimal solution for large-scale problems. In this paper,an integer programming model is proposed. By decomposing the model into a constrained master problem and some subproblems,a column generation method is designed for the solving of the linear relaxation model,and then the integer solution of the problem is found through branch & price. In the process of solving,relaxation variables are introduced to solve the pseudo infeasibility of the model. Furthermore,pruning strategy,branch strategy,and two methods for reducing the solution space according to different situations are designed. On the PSPLIB data set,for the problems with 30 processes,the current optimal solutions can be obtained in 10 minutes for 301 out of 480 instances. For the problems with 60 processes,the current optimal solutions can be obtained in 20 minutes for 269 out of 480 instances. For the problem with 90 processes,the current optimal solutions can be obtained in 30 minutes for 263 out of 480 instances. At the same time,by using the strategies of reducing the solution space,the number of timeout instances decreases significantly,and the performance of the optimized initial solution is significantly improved. Experimental results show that the proposed algorithm is effective.

Keywords　RCPSP,Integer programming,Column generation,Branch & Price,Pseudo infeasibility

1　引言

资源受限项目调度问题(Resource-Constrained Project Scheduling Problem,RCPSP)是最具代表性的一类项目调度问题,已被证明是 NP-hard 问题[1]。RCPSP 是很多实际调度问题的抽象表示[2],其调度目标通常是最小化总完工时间。

目前国内外求解 RCPSP 的思路主要有启发式算法、元启发式算法和精确算法 3 种。

启发式算法指为问题设计一系列的优先规则并根据规则进行调度,可分为串行调度与并行调度两种机制。常用的优先规则有最多紧后工序(Max Total Successors,MST)、最迟开始时间(Latest Start Time,LST)和最迟完成时间(Latest Finish Time,LFT)等[3]。Kolisch 等[4]介绍了串行进度生成机制与并行进度生成机制,并比较了在串行进度生成机制下

到稿日期:2021-11-29　返修日期:2022-04-30
通信作者:董兴业(xydong@bjtu.edu.cn)

几种最经典的优先规则的求解效果。Browing 等[5]以资源的冲突和分布程度、项目的复杂程度、项目的延迟程度为目标，在 12 320 个测试用例上测试了 20 种优先规则的效果；Türkakn 等[6]在 RCPSP 经典数据集 PSPLIB 上测试了 17 种优先规则在串行与并行两种机制下的求解效果，为了更好地探究不同优先规则的求解能力，根据工序总数、资源总数、资源供应情况等组合生成了不同规模的数据集并测试。启发式算法能够在较短的时间内得到一个解，但解的质量通常较差。

元启发式算法是在启发式算法的基础上结合仿生学的知识进行群体寻优，常见的有蚁群算法[7-8]、遗传算法[9]、模拟退火算法[10-11]、禁忌搜索算法[12]等，通常对工序序列进行编码。Tofighian 等[7]发现蚁群算法在求解 RCPSP 中性能普遍较优。Savitri 等[8]总结了蚁群算法求解 RCPSP 的流程，并分析了算法参数对求解效果的影响。Jia 等[9]使用串行进度生成机制将序列转化为调度方案，通过遗传算法不断优化工序序列。Boctor[10]首先提出使用模拟退火算法求解 RCPSP，从不同的初始解出发搜索不同方向，根据接受概率判断当前解是否被接受。Pan 等[11]使用优先规则 MINSLK 生成初始解，结合模拟退火算法和邻域搜索求解 RCPSP。Omer[12]使用并行进度生成机制将序列转化为调度方案，通过交换序列中两个工序的位置获得新解，建立禁忌表来禁止某些工序的交换。不同的元启发式算法也可以结合使用，Pellerin 等[13]在 PSPLIB 上测试了不同结合方法的性能并归纳了表现较优的结合策略。Golab 等[14]归纳了元启发式算法在 RCPSP 领域的研究，并比较了 5 种算法的求解性能。元启发式算法进行群体寻优，因此在寻找较优解的过程中更具导向性，求解效果更好。但由于过程较复杂，其求解时间远多于启发式算法。

启发式算法与元启发式算法都无法确保得到最优解。为了保证求得全局最优解，必须采用精确算法。精确算法指为问题建立数学模型，直接求解该数学模型，其中分支定界(Branch&Bound)、动态规划、整数规划是求解 RCPSP 的经典方法。Patterson[15]提出了基于紧前关系的分支定界法，Rostami 等[16]提出通过构造上界改进剪支策略。精确算法虽然能够求得全局最优解，但由于计算复杂度过高，仅适用于小规模问题，无法求解大规模问题。

列生成方法是求解大规模 RCPSP 问题的有效方法。目前该方法针对 RCPSP 的研究较少，且在现有工作中，为了方便对 RCPSP 建模，通常会放松部分约束，导致实际上无法保证取得全局最优解。其中，Mingozzi 等[17]放松了不可抢占约束与部分偏序约束，给出了 RCPSP 的线性规划形式。Brucker 等[18]在此基础上对线性规划模型做了改动，使其能够使用列生成求解，并结合约束传播构造总完工时间 T 的下界，虽然其通过确定每个工序的时间窗进一步加强了偏序约束，但该下界仍有可能不满足不可抢占约束与偏序约束。Damay 等[19]在放松不可抢占约束与偏序约束的条件下重构了线性规划模型，使用列生成求解模型，期望找出能够并发执行的工序的最优组合，之后再对得到的解进行修正，使其满足所有约束。Wang 等[20]使用列生成求解需要在处理机上执行工序的 RCPSP，在子问题中求解出单个处理机的方案，在主问题中组合这些方案以获得最优总方案，其采用了启发式方法将决策变量由连续转为整数。RCPSP 的总体研究现状如图 1 所示。

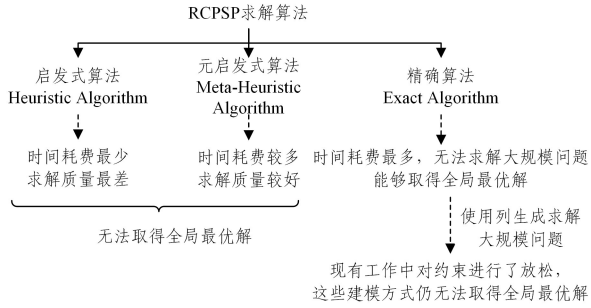


图 1 RCPSP 的研究现状

Fig. 1 Research status of RCPSP

本文提出了一种基于分支定价的精确算法，为 RCPSP 建立列生成主问题与子问题模型，并解决了求解过程中出现的伪不可解问题。在求解过程中不断缩小解空间并使用剪支、分支等策略，最终能够在满足资源约束、偏序约束和不可中断约束的条件下求得全局最优解。

2 RCPSP 定义

可更新资源集合 $R=\{k\mid k=1,\cdots,K\}$，定义 R_k 表示第 k 种资源的总量。

工序集合 $J=\{i\mid i=1,\cdots,N\}$，定义 s_i 表示工序 i 的开始时间，c_i 表示工序 i 的结束时间，p_i 表示执行工序 i 所需要的时间，r_{ik} 表示工序 i 对第 k 种资源的需求量。P_i 为工序 i 的前序工序集，工序 i 必须等待其前序工序集中的所有工序完成之后才能开始。本问题中的工序皆为不可中断工序，即工序一经执行，在其完成之前不允许被中断。

记时间标号 $t\in[0,T)$。设 t 时刻正在执行的工序集为 $A(t)$，则工序 i 对第 k 种资源的需求量 $r_{ik}(t)$ 的取值如式(1)所示；t 时刻所有工序对第 k 种资源的需求总量 $r_k(t)$ 如式(2)所示，则 RCPSP 的数学模型如式(3)－式(6)所示：

$$r_{ik}(t)=\begin{cases} r_{ik}, & i\in A(t) \\ 0, & i\notin A(t) \end{cases} \tag{1}$$

$$r_k(t)=\sum_{i=1}^{N} r_{ik}(t) \tag{2}$$

$$\min \max\{c_i,\forall i\in J\} \tag{3}$$

$$\text{s.t. } s_i+p_i=c_i,\forall i\in J \tag{4}$$

$$c_j\leqslant s_i,\forall i\in J,\forall j\in P_i \tag{5}$$

$$r_k(t)\leqslant R_k,\forall t\in[0,T) \tag{6}$$

其中，式(3)表示目标函数为最小化总完工时间；式(4)表示需要为工序分配足够的执行时间，且执行不能被中断；式(5)表示工序的偏序约束；式(6)表示工序的资源约束。

图 2 所示为一个简单的调度算例，每个节点表示一个工序，节点间的箭头表示偏序约束。

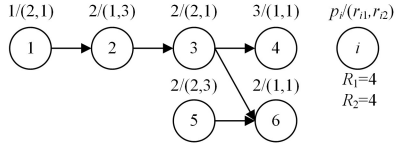


图 2 资源受限项目调度问题示例

Fig. 2 Example of RCPSP

最优调度方案下资源 1 和 2 的消耗情况分别如图 3 和图 4 所示。可以看出，在满足资源约束与偏序约束的条件下，总完工时间为 8。

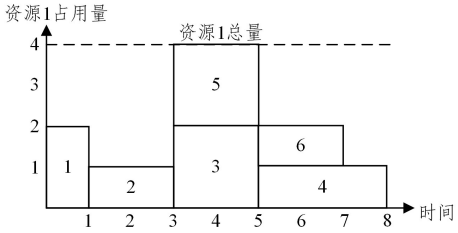


图 3 1 类型资源消耗情况

Fig. 3 Type 1 resource consumption

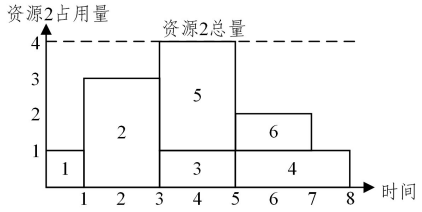


图 4 2 类型资源消耗情况

Fig. 4 Type 2 resource consumption

3 列生成求解

直接对大规模问题建立完整的整数规划模型时，模型中的决策变量会非常多，导致难以求解。实际上只有小部分决策变量会出现在最优解中，大部分决策变量是“无用”的，不会出现在最优解中，也不必出现在模型中。因此可以先建立一个可行的、不完整的模型，称之为“限制主问题”，然后通过求解子问题寻找“有用”的决策变量并加入主问题中，直至主问题达到最优。相比完整模型，通过列生成方法得到的最终模型的复杂度会大幅降低，因此求解时间也会大大缩短。

3.1 列生成问题建模

问题目标函数为最小化总完工时间。为了更好地表示该目标，在原 RCPSP 定义的基础上加上首尾虚工序。虚工序执行所需时间为 0 且不消耗任何资源，仅作为开始和结束的标志。整个调度方案的开始时间为开始虚工序的开始时间，结束时间为结束虚工序的完成时间。为图 2 所示算例添加首尾虚工序后的问题示例如图 5 所示。

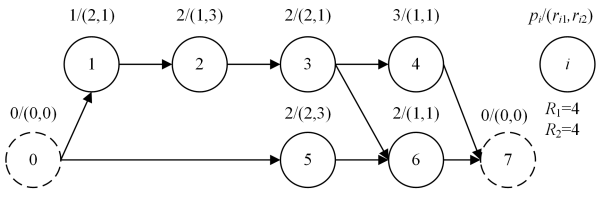


图 5 为图 2 所示算例增加虚工序后的问题示例

Fig. 5 Instance after adding dummy processes to the instance in Fig. 2

在上述符号的基础上，主问题模型中所使用的符号如表 1 所列。

表 1 主问题符号

Table 1 Symbols of the master problem

参数名称	含义
x_{ij}	决策变量，工序 i 的第 j 个方案是否被选中，1 表示被选中，0 表示未被选中
a_{ij}^t	工序 i 的第 j 个方案在 t 时刻是否在执行，1 表示在执行，0 表示未执行
s_{ij}	工序 i 的第 j 个方案的开始时间
c_{ij}	工序 i 的第 j 个方案的结束时间
T	时间轴最大值，初始化为初始解的总完工时间
P	偏序对集合，(i,j) 表示 i 是 j 的前序工序

建立主问题模型如下：

$$\min \sum_j c_{(N+1)j} x_{(N+1)j} \tag{7}$$

$$\text{s.t.} \sum_j x_{ij} = 1, \forall i \in J \tag{8}$$

$$\sum_i \sum_j a_{ij}^t r_{ik} x_{ij} \leqslant R_k, \forall t \in [0,T], \forall k \in R \tag{9}$$

$$\sum_{i_1} \sum_j c_{i_1 j} x_{i_1 j} - \sum_{i_2} \sum_j s_{i_2 j} x_{i_2 j} \leqslant 0, \forall (i_1, i_2) \in P \tag{10}$$

$$x_{ij} \in \{0,1\} \tag{11}$$

式(7)表示目标函数为最小化总完工时间，即结束虚工序的完成时间；式(8)表示对于任一工序只能选择一个执行方案，该约束有 $N+1$ 行(含两个虚工序)；式(9)表示在任一时刻正在执行的所有工序对每种资源的需求总量不能超过资源总量，该约束有 $K \times T$ 行；式(10)表示偏序约束，处于偏序对中前序位置的工序的完成时间应不大于处于后序位置的工序的开始时间，该约束有 $|P|$ 行；式(11)表示决策变量 x_{ij} 为 0～1 的变量，若选中工序 i 的第 j 个决策方案则值为 1，否则为 0，模型为整数规划。

列生成算法需要获得每行约束的对偶变量，为了求得这些对偶变量，将 x_{ij} 松弛为连续变量，模型变为线性规划，称为“松弛限制主问题”，即式(11)变为式(12)。

$$0 \leqslant x_{ij} \leqslant 1 \tag{12}$$

对于工序 i 的一个执行方案 j，其约束系数在式(8)所示行中第 i 行为 1，其余为 0；在式(9)所示行中，由于工序不可中断，因此从 s_{ij} 到 c_{ij} 为连续的 r_{ik}，其余为 0，重复 K 次；在式(10)所示行中处于前序位置时为 c_{ij}，处于后序位置时为 $-s_{ij}$。决策变量 x_{ij} 的约束系数在约束矩阵中所对应的一列如图 6 所示。

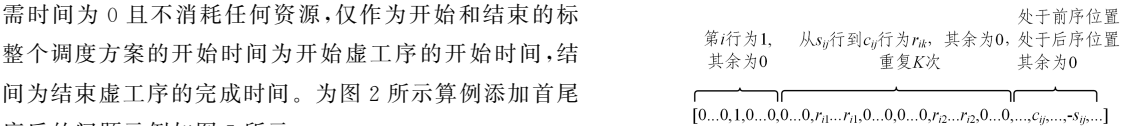


图 6 x_{ij} 的约束系数在约束矩阵中所对应的一列

Fig. 6 Column corresponding to constraint coefficient of x_{ij} in constraint matrix

在上述符号的基础上，子问题模型使用到的符号如表 2 所列。根据最优化理论，对于工序 i 的一个开始时间 y_i，其子问题目标函数如式(13)所示：

$$\max \alpha_i + \sum_{k=1}^{K} \sum_{t=y_i}^{y_i+p_i} r_{ik} \beta_{kt} + \sum_{p=0}^{|P|} w_p \gamma_p \tag{13}$$

其中，若 i 在第 p 个偏序对中为前序，则 w_p 为 $y_i + p_i$；若 i 在第 p 个偏序对中为后序，则 w_p 为 $-y_i$；否则为 0。求解时遍历 $[0,T)$，找到能最大化子问题目标函数的开始时间 $y_i \in [0,$

T)。所求得的方案自动满足工序不可中断约束，而偏序约束和资源约束则在主问题中满足。若目标函数最大值小于或等于0，那么工序 i 无论从什么时间开始都不能改进当前主问题，否则得到工序 i 的一个新的开始时间 y_i，即工序 i 的一个新的执行方案。

表2 子问题符号

Table 2 Symbols of the sub-problem

参数名称	含义
y_i	决策变量，表示第 i 个工序执行的开始时间
α_i	式(8)的对偶变量，$i=1,\cdots,J$
β_{kt}	式(9)的对偶变量，对应于 t 时刻关于资源 k 的约束
γ_p	式(10)的对偶变量，对应于偏序约束 $p\in P$
w_p	γ_p 在目标函数中的系数，即式(10)中的约束系数

初始时使用3.2节所述策略构造初始解，使用初始解构建松弛限制主问题。每次求解当前主问题，得到每行约束的对偶变量，构建每个工序的子问题。

求解每个工序的子问题，若所有工序子问题的最优值均小于或等于0，那么当前主问题已是最优松弛主问题，停止算法；否则根据子问题最优值最大的解构造一个新工序的执行方案并添加到主问题中（添加一个决策变量与对应的一列约束系数）。该方案有望使主问题的目标函数值下降最多从而改进主问题。算法流程如图7所示。

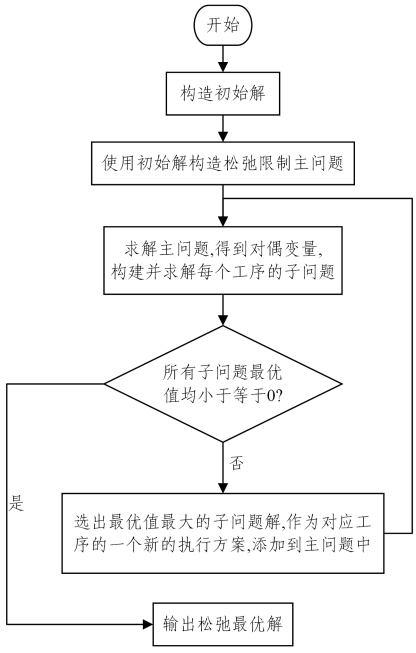


图7 列生成求解 RCPSP 流程图

Fig. 7 Flow chart of solving RCPSP with column generation algorithm

3.2 构建初始解

初始解采用随机生成方式与启发式算法中的串行进度生成机制生成，启发式规则为最早开始时间（EST）与最多紧后工序（MTS）[3]。

调度环境中的工序分为已调度工序集 FS、所有前序工序已经处于 FS 的工序集 CS 以及未调度工序集 NS。工序 $i\in CS$ 的最早可开始时间为 $\max\{\max_{j\in P_i} c_j$，满足资源约束的最早开始时间$\}$。

每次从 CS 中选出一个工序，计算其最早开始时间，并令其从此时刻开始。之后更新资源使用情况与3个集合。重复选工序的过程，直至选出所有工序。对于 EST，每次从 CS 中选择具有最早可开始时间的工序；对于 MTS，每次从 CS 中选择紧后工序数最多的工序；对于随机生成方式，每次从 CS 中随机选择一个工序。

生成初始解时，按随机方式运行5次，EST 和 MTS 各运行1次，取总完工时间最小的结果作为初始解。

4 分支定价求解

4.1 分支定价主问题重建模

由于对决策变量进行了松弛，即经过第3节得到的是松弛最优主问题，因此求得的最优解中可能存在小数，这显然是不可行的。此时可以采用分支定价法逐步确定每个非整数变量的整数取值。

分支定界算法的思想是根据问题特点设计分支策略，分解原问题可行域，在各子域内求解松弛问题，通过逐渐缩小解的上下界来获得最优整数解。其与列生成结合被称为分支定价，每次分支后对当前节点重新进行列生成，优化主问题。

松弛最优解中的某个小数决策变量 x_{ij} 在最优整数解中要么等于1，要么等于0。因此从一个小数解可以分出两支：$x_{ij}=1$ 与 $x_{ij}=0$。此时对偶变量会随之改变，主问题可能已经不是最优，需要再使用列生成改进主问题，直至主问题重新到达最优。

进行分支定价时，按照3.2节生成初始解，由初始解构成松弛限制主问题作为根节点。从根节点出发，首先使用列生成求解本节点得到松弛最优解；其次按照4.2节所述分支策略从节点选择一个非整数决策变量 x_{ij} 进行分支，所得的两个新模型作为两个子节点，求解子节点并按照4.3节所述剪支策略进行剪支；然后按照4.4节所述遍历策略遍历节点，直至遍历完整个二叉树。分支定价搜索过程如图8所示，其中上界初始化为初始解的总完工时间。在不进行4.5节所述缩小解空间的情况下，分支定价算法流程如图9所示。

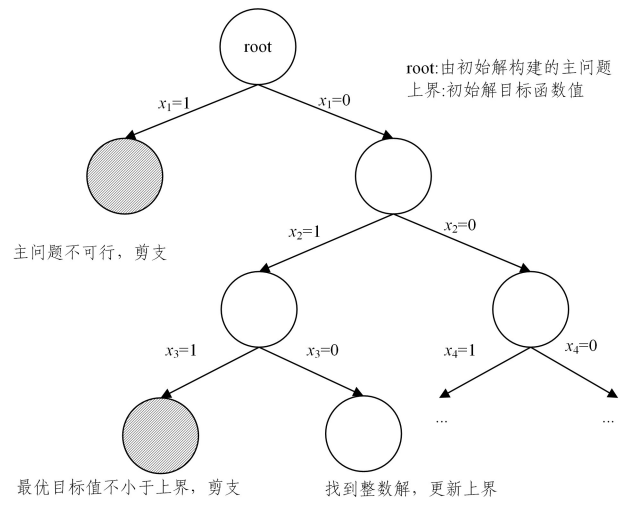


图8 分支定价搜索过程示例

Fig. 8 Example of search process of branch & price

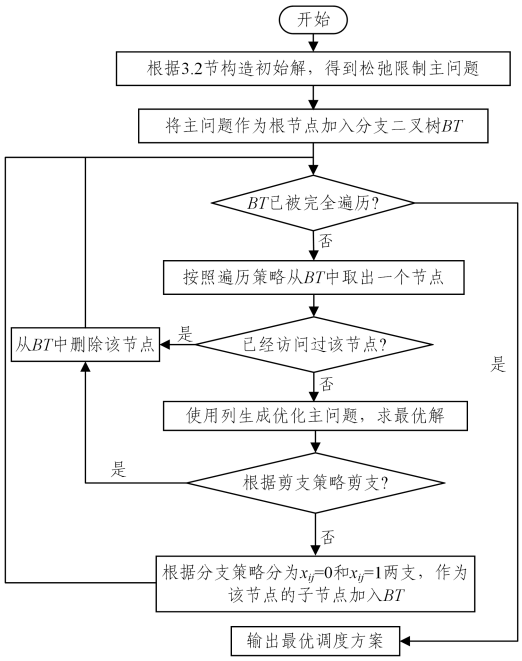


图 9 未缩小解空间的分支定价流程图

Fig. 9 Flow chart of branch & price without reducing the solution space

固定$x_{ij}=1$后主问题可能不可行，有两种情况：

(1) 真不可解。原因是该变量和其他固定为 1 的决策变量发生资源冲突。

(2) 伪不可解。将小数变为 1 后，由于约束(9)的存在会影响其他工序决策变量的取值，使其相加后不能等于 1，从而违反式(8)，也可能违反式(10)。但这并不能说明$x_{ij}=1$不可能存在于最优整数解中，因为当前主问题是不完整的。

例如，图 10 所示的算例在进行某步分支时，当前主问题中工序 0，1，2，3 相关变量的信息如表 3 所列，其中有些变量在之前的分支中已经确定了取值为 0 或 1，其余变量取值为主问题取得最优解时对应的小数值。

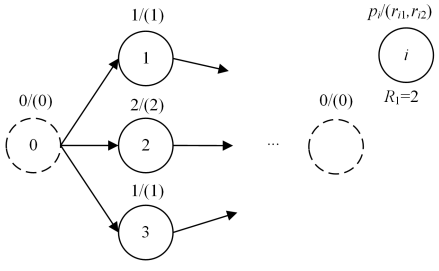


图 10 某调度问题示例

Fig. 10 Example of RCPSP

表 3 图 10 算例主问题中变量取值

Table 3 Variables in the master problem of instance in Fig. 10

x_{ij}	开始时间，结束时间	是否在之前的分支中确定了取值	取值	所需资源量
x_{01}	0，0	是	1.0	0
x_{11}	0，1	是	1.0	1
x_{21}	0，2	否	0.5	2
x_{22}	1，3	否	0.0	2
x_{23}	2，4	否	0.5	2
x_{31}	3，4	否	1.0	1

若该步分支将x_{21}确定为 1，则和之前分支时已经确定为 1 的x_{11}产生资源冲突，因为 0 时刻所需的资源量之和为 3，大于资源总量 2，因此$x_{21}=1$的分支不可行，即真不可解。

若该步分支将x_{23}确定为 1，由于其和x_{31}产生资源冲突，则x_{31}的取值应为 0，此时会违反式(8)中关于工序 3 的约束(每个工序必须选一个执行方案)，因此$x_{23}=1$的分支不可行。但因为主问题是不完整的，可能存在不在当前主问题中的工序 3 的另一执行方案x_{32}使得模型可行，这就是伪不可解的情况。

固定$x_{ij}=0$不会导致资源冲突，但同样有可能影响其他变量的取值，从而导致伪不可解。

为了解决伪不可解问题，向主问题模型中添加 N 个松弛变量，第 i 个松弛变量x_i'解决第 i 个工序的不可解问题。可行解中应不包含任何松弛变量，因此需要将其在目标函数中的系数设为一个很大的值 M。

对于式(8)，松弛变量x_i'在第 i 行的系数为 1，其余为 0；对于式(9)，系数均为 0；对于式(10)，若工序 i 处在前序，则系数为 0，若处在后序，则系数为$-T$。如果因确定了其他工序的决策变量为 1 而导致工序 i 原有的决策变量取值相加小于 1，那么x_i'会取小数值补上，且一定满足其余约束。

添加松弛变量是必要的，可以解决模型伪不可解的问题。我们在实验中发现求解中出现伪不可解的情况非常多，如果不添加，基本不会对初始解有任何优化。添加松弛变量后，主问题模型如式(14)－式(18)所示：

$$\min \sum_j c_{(N+1)j}x_{(N+1)j}+\sum_{i=1}^{N}Mx_i' \tag{14}$$

$$\text{s.t. } \sum_j x_{ij}+x_i'=1,\ \forall i\in J \tag{15}$$

$$\sum_i\sum_j a_{ij}^t r_{ik}x_{ij}\leqslant R_k,\ \forall t\in[0,T],\ \forall k\in R \tag{16}$$

$$\sum_{i_1}^{j} c_{i_1 j}x_{i_1 j}-\sum_{i_2}^{j} s_{i_2 j}x_{i_2 j}-Tx'_{i2}\leqslant 0,\ \forall (i_1,i_2)\in P \tag{17}$$

$$0\leqslant x_{ij}\leqslant 1,0\leqslant x_i'\leqslant 1 \tag{18}$$

在确定了 $x=1$ 后，除了真不可解的情况，其余情况下模型一定可解，就能使用列生成继续优化主问题。当主问题达到最优后，若模型实际上仍不可行，则最优解中松弛变量一定会取到大于 0 的值。因此，松弛变量在主问题最优解中的取值有以下两种情况。

(1)取值全部等于 0：最优解可行。

(2)存在取值大于 0：最优解不可行，但因为 M 很大，此时主问题目标函数值也会很大，会通过 4.3 节的剪支策略停止对该分支的搜索。

4.2 分支策略

节点最优解中非整数决策变量可能不止一个，分支策略决定了如何从这些非整数决策变量中选出一个进行分支。有以下几种分支方法：选取取值最接近 0.5 的变量分支；选取取值最接近 1 的变量分支；选取开始时间最早的变量分支；选取完成时间最晚的变量分支。选取开始时间最晚的变量分支能够将时间确定得最晚，从而留出较多时间给前面的工序安排执行，更容易找到一个新的可行解，但总完工时间不会减少太多，属于较为保守的做法。为提高求解性能，当非整数变量的数量大于 $0.2N$ 时，选取完成时间最晚的变量分支，否则选取

取值最接近 0.5 的变量分支。

4.3 剪支策略

和分支定界法相同，有些节点及其分支不可能产生最优解，可以直接剪去，以缩减二叉树的遍历过程。设上界 T 为当前已知的整数最优解目标值，初始化为初始解的目标值。由最优化理论可知，对于最小化问题，在其余条件不变的情况下增加约束会使得新问题的最优值大于或等于原问题最优值。因此，若节点最优值不小于上界，则该节点及其分支不可能取得更好的整数解，应当剪去。

值得注意的是，在算法求解过程中，即使节点的松弛最优解不是整数解，但该节点主问题模型中仍有可能包含一个较优的整数解，因此当最优解中的非整数变量的数量小于 $0.2N$ 时，可直接将问题转化为整数规划求整数最优解。若整数最优值小于上界，则更新上界。

若节点不可行（真不可解），则该节点及其分支都不可行，应当剪去。若节点最优解为整数解，则得到一个新的整数解，剪去该节点并更新上界。

综上所述，剪支策略如下：

（1）当节点最优值不小于上界（包含伪不可解中模型实际不可行的情况）时剪支；

（2）当节点不可行（真不可解）时剪支；

（3）当节点最优解非整数变量的数量小于 $0.2N$ 时，求整数解，若整数解最优值小于上界，更新上界，不剪支；

（4）当节点最优解为整数解时，更新上界，剪支。

4.4 遍历策略

遍历分支二叉树时可以采用 3 种策略，分别是深度优先（Depth First Search）、广度优先（Breadth First Search）和最优下界优先（Best Bound Search）。深度优先能够较快地找到新的上界，在后续遍历中能够根据新的界剪支，从而缩短求解时间，适用于树的深度不大的情况。与之对应，广度优先因需要横向遍历到最底层才能找到新的界，因此求解时间花费较大，不适用于分支定价算法。最优下界优先指每次从所有节点中选取松弛最优解目标值最小的节点，因为此时意味着该节点的可行空间中更有可能得到更好的整数解。但该策略需要计算当前所有节点的松弛最优值，时间和空间消耗都较大。本文采用深度优先作为遍历策略。

4.5 缩小解空间

为缩短求解时间，应尽量省去无意义的分支，尽可能缩小解空间。可从两个角度考虑：1）从当前模型中去除无效的决策变量；2）在子问题决策变量 y_i 的取值范围中去除无效的范围，避免为主问题添加新的无效决策变量。

4.5.1 静态缩小解空间

实际上，在各类资源充足的假设下，工序 i 的最早开始时间 r_i 不得早于从开始虚工序到该工序的最长路径，最晚开始时间 d_i 不得晚于 T-q_i，其中 q_i 表示从工序 i 到结束虚工序的最长路径，因此可将子问题中 y_i 的取值范围从 $[0,T]$ 缩小为 $[r_i,d_i]$。在图 5 所示的算例中，$r_6=5$，$d_6=T-2$，因此对于工序 6，y_6 的取值范围为 $[5,T-2]$。

4.5.2 动态缩小解空间

在求解过程中，确定了某个变量 $x_{ij}=1$，也就确定了调度方案中相应工序 i 的开始时间 s_i，由此可以更新两种信息。

（1）更新前序工序的最晚开始时间与后序工序的最早开始时间。前序工序最晚结束时间 $d_e=\min\{d_e,s_i-p_e\}$，后序工序最早开始时间 $r_e=\max\{r_e,c_i\}$。这种更新是递归进行的，包括间接关系。在图 5 所示的算例中，若确定了工序 2 的某个执行方案的决策变量为 1，那么 $d_1=\min\{d_1,s_2-p_1\}$，$r_3=\max\{r_3,c_2\}$，$r_4=\max\{r_4,r_3\}$，以此类推。如果 r_3 没有发生改变，那么无须计算其后序 r_4，r_6，r_7，前序同理。更新带来了两点影响：删去了开始时间不在 $[r_e,d_e]$ 中的决策变量；更新了工序 e 子问题中 y_e 的取值范围，其中 e 为与 i 存在间接关系的工序。

（2）更新资源使用情况，即某些时刻资源的总量减少。在图 5 所示的算例中，如果确定了工序 2 的某个执行方案的决策变量为 1，那么资源 2 在 $[s_2,c_2)$ 仅剩余 1 个单位，而工序 5 需要 3 个单位，因此工序 5 不能在 $[s_2,c_2)$ 内执行。更新带来了两点影响：删去了不满足资源约束的决策变量；从所有工序的子问题中 y_i 的取值范围中去除了不满足资源约束的时刻。

若在求解过程中得到了更短的总完工时间 T，则所有工序的可开始时间可能发生改变，即工序的 $d_i=\min\{d_i,T-q_i\}$。更新带来了两点影响：对于所有工序，删去了开始时间不在 $[r_i,d_i]$ 中的决策变量；更新了子问题中 y_i 的取值范围。

算法整体伪代码如算法 1 所示。

算法 1 分支定价算法

输入：算例数据，包括 R，J

输出：最优整数模型，最小总完工时间与对应的调度方案

```
1. 初始化松弛主问题：按照 3.2 节生成初始解，按照 4.1 节构造松弛限制主问题 root；
2. 初始化分支定价二叉树：上界 T 设为初始解的总完工时间，nodes=[root]，best_m=root；
3. while(true){
4.    c_node=nodes 的末尾元素；
5.    if(c_node 被搜索过)
6.       从 nodes 中删去 c_node，continue；
7.    标记 c_node 被搜索过；
8.    使用列生成按照图 7 求解 c_node；
9.    if(c_node 可解且最优目标值小于 T){
10.   if(最优解中没有小数变量)
11.      best_m=c_node，T=最优目标值；
12.   else{
13.      if(最优解中小数变量的数量≤0.2N){
14.         将 c_node 转为整数模型 int_m 求解；
15.         if(int_m 可解且最优目标值小于 T){
16.            T=最优目标值，按照 4.5.1 节缩小解空间；
17.         }
19.      }
20.      按照 4.2 节得到分支变量 x_ij；
21.      将 c_node 分支为 x_ij=0 的模型 node_0 和 x_ij=1 的模型 node_1；
22.      对 node_1 按照 4.5.2 节缩小解空间；
23.      将 node_0 和 node_1 加入到 nodes 尾部；
24.   } //end-else
25.} end-if
```

26. else //c_node 无解或最优目标值大于 T
27. 从 nodes 中删去 c_node;
28. }//end-while
29. 输出 best_m,T 与对应的调度方案。

对于图 2 所示的算例,使用分支定价算法后最优主问题模型中共有 24 个决策变量,其中松弛变量 8 个,取值全部为 0,剩余变量 16 个,取值如表 4 所列,$T=8$。如果不使用本文中的方法建模,则模型中决策变量数量为 $9\times7=63$ 个。使用本文提出的分支定价算法建模,能够使决策变量减少到 24 个,有效缩短了求解时间,因此能够求解大规模问题。

表 4 图 2 算例最优解中变量取值

Table 4 Variables in optimal solution of instance in Fig. 2

x_{ij}	开始时间,结束时间	是否在初始解中	取值
x_{01}	0,0	是	1.0
x_{11}	0,1	是	1.0
x_{12}	1,2	否	0.0
x_{21}	2,4	是	0.0
x_{22}	1,3	否	1.0
x_{31}	4,6	是	0.0
x_{32}	3,5	否	1.0
x_{41}	6,9	是	0.0
x_{42}	5,8	否	1.0
x_{51}	0,2	是	0.0
x_{52}	1,3	否	0.0
x_{53}	2,4	否	0.0
x_{54}	3,5	否	1.0
x_{61}	6,8	是	1.0
x_{71}	9,9	是	0.0
x_{72}	8,8	否	1.0

5 数值实验

5.1 整体求解能力

为了测试算法在不同规模数据集上的求解能力,随机生成了不同种类的算例,其中工序个数 N 可取 30,60,90,120,资源类别数 R 可取 1,2,3,4,共 16 种算例规模,每种规模生成 480 个算例。M 取 $1\,000N$。算例生成方式为:除虚工序外,工序执行时间的取值范围为[1,10]、对资源需求量的取值范围为[0,10]、资源总量的取值范围为[20,30]。初始随机阈值 $es\_threhold=0.9$,若随机值小于该阈值,则在$[i+1,i+5]$中随机选择一个工序作为 i 的后序,每选出一个后序工序,更新 $es\_threhold$ 为 $0.7es\_threhold$。

为实验设定最大求解时间,不同规模算例的最大求解时间为 $20N$(s)。算法在不同规模数据集上的求解结果如表 5—表 8 所列,其中 $init\_T$ 表示初始解,$best\_T$ 表示算法得到的最优解,优化初始解的个数表示 $best\_T$ 小于 $init\_T$ 的个数。

表 5 在工序规模为 30 的数据集上的求解结果

Table 5 Results on data set with 30 processes

K	超时个数	优化初始解的个数	平均求解时间/s	RPD/%
1	0	4	0.90	3.11
2	0	8	0.69	2.78
3	2	11	2.12	3.10
4	2	20	4.34	2.35

表 6 在工序规模为 60 的数据集上的求解结果

Table 6 Results on data set with 60 processes

K	超时个数	优化初始解个数	平均求解时间/s	RPD/%
1	7	18	1.54	1.38
2	13	34	9.15	1.07
3	16	43	12.37	1.40
4	18	73	19.17	1.44

表 7 在工序规模为 90 的数据集上的求解结果

Table 7 Results on data set with 90 processes

K	超时个数	优化初始解个数	平均求解时间/s	RPD/%
1	9	41	8.84	1.01
2	36	68	21.18	0.96
3	51	91	36.54	1.00
4	57	116	51.62	0.92

表 8 在工序规模为 120 的数据集上的求解结果

Table 8 Results on data set with 120 processes

K	超时个数	优化初始解个数	平均求解时间/s	RPD/%
1	20	69	24.81	0.90
2	47	108	32.40	0.67
3	95	146	83.59	0.71
4	105	171	104.88	0.70

剔除超时算例后计算平均求解时间和对初始解的优化程度 RPD,其中 $RPD=[(init\_T-best\_T)/init\_T]\times100\%$。从实验结果可以看出,超时个数、平均求解时间有随 N,R 的增加而增长的趋势,如图 11 和图 12 所示。

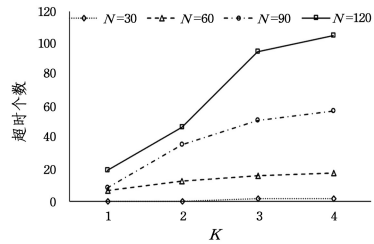


图 11 超时个数与工序数和资源类别数的关系

Fig. 11 Relationship between the number of timeouts and the number of operations and resource types

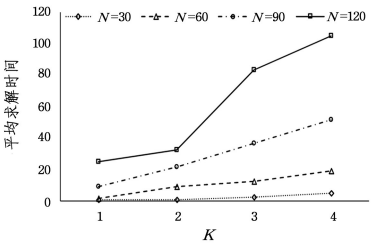


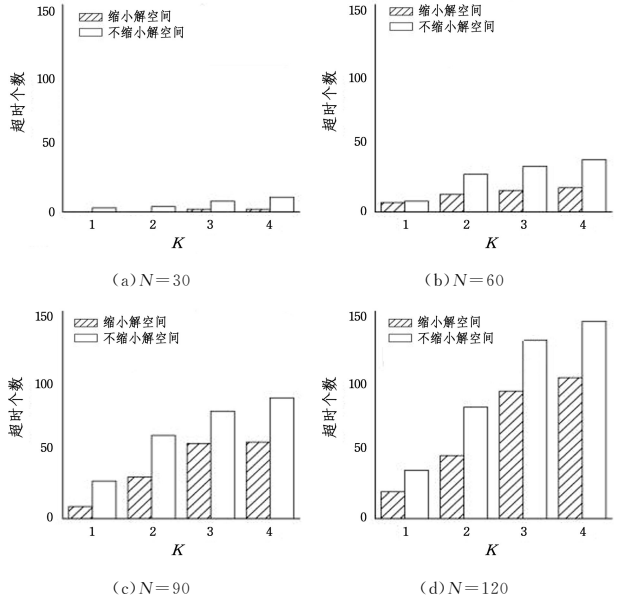
图 12 平均求解时间与工序数和资源类别数的关系

Fig. 12 Relationship of average solution time and the number of operations and resource types

5.2 缩小解空间的效果

为验证 4.5 节中缩小解空间对算法性能的影响,在上述数据集上进行不缩小解空间的实验,其在超时个数、优化初始解个数两个指标上的对比如图 13 和图 14 所示。可以看出,缩小解空间后,超时个数明显减少,优化初始解的能力明显

上升，说明所提出的缩小解空间方法是有效的，能够大幅提升算法求解性能。

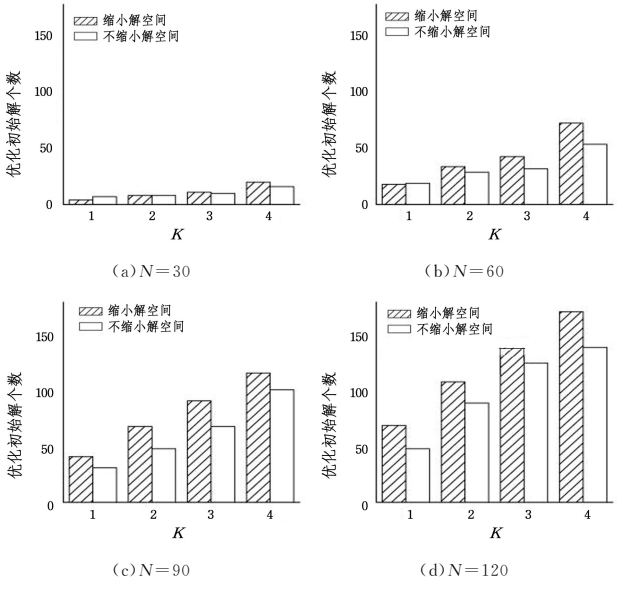


(a) $N=30$ (b) $N=60$

(c) $N=90$ (d) $N=120$

图 13 缩小解空间对超时个数的影响

Fig. 13 Influence of reducing solution space on the number of timeouts



(a) $N=30$ (b) $N=60$

(c) $N=90$ (d) $N=120$

图 14 缩小解空间对优化初始解数量的影响

Fig. 14 Influence of reducing solution space on the number of improved initial solutions

5.3 在公开数据集上和当前已知最优解对比

在公共数据集 PSPLIB 的 rcpsp 数据集[1)]上求解。该数据集工序个数包含 30，60，90，120，资源类别数固定为 4，每个规模有 480 个算例。

该数据集的约束相对于 5.1 节中生成的数据集更紧，因此更难求解。网站公开了使用不同算法求得的当前最优解。使用本文算法和当前已知最优解的对比如表 9 所列，其中，*optimal_T* 表示算例已知最优解。可以看到，随着工序个数的增加，超时个数和平均求解时间呈增长趋势，这和 5.1 节中得出的结论是相同的。*best_T*=*optimal_T* 表示算法得到的解和目前已知最优解相同。除 120 个工序规模的算例外，在规定的时间内，算法得到的最优解与大部分已知最优解相同，这表明本文算法具有良好的求解能力。

表 9 在 PSPLIB 数据集上的求解效果

Table 9 Solution effect on PSPLIB data set

	工序个数			
	30	60	90	120
超时个数	220	227	219	451
提高初始解个数	80	89	99	144
平均求解时间/s	20.33	24.29	21.79	35.35
RPD/%	3.27	2.70	2.04	2.72
best_T>*optimal_T*	179	211	217	449
best_T=*optimal_T*	301	269	263	31

结束语 本文提出了一种基于分支定价求解 RCPSP 的精确算法，通过添加松弛变量解决了分支过程中伪不可解的问题，并设计了分支、剪支策略与两种缩小解空间的方法以提高求解效率。实验表明所提算法能够求得 RCPSP 问题的全局最优解，且缩小解空间的方法有效。

但是所提算法仍有提升的空间。首先，许多研究表明分支定价算法的效率和初始解的关系明显。本文采用的初始解生成方法是较简单的启发式方法，如果能够在较好的初始解上开始优化，那么求解效率也会提高。其次，解空间仍可以进一步缩小。可以和约束传播等方法相结合，在求解过程中进一步缩小子问题解空间，减少模型中的列。在分支时需要分支的小数变量也可能会进一步减少，从而提高算法效率。

参 考 文 献

[1] BLAZEWICZ J, LENSTRA J K, RINNOOY KAN A H G. Scheduling Subject to Resource Constraints: Classification and Complexity[J]. Discrete Applied Mathematics, 1983, 5(1): 11-24.

[2] HARTMANN S, BRISKORN D. An Updated Survey of Variants and Extensions of the Resource-Constrained Project Scheduling Problem[J]. European Journal of Operational Research, 2022, 297(1): 1-14.

[3] HE Z W, JIA T, XU Y. A Survey of Heuristics for Resource-Constrained Project Scheduling Problems[J]. Operations Research and Management Science, 2007(3): 78-84.

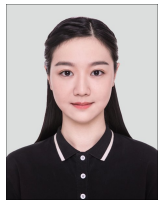
[4] KOLISCH R, HARTMANN S. Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis[M]// Project Scheduling: Recent Models, Algorithms and Applications. Boston, MA: Springer US, 1998: 148-178.

[5] BROWNING T R, YASSINE A A. Resource-Constrained Multi-Project Scheduling: Priority Rule Performance Revisited[J]. International Journal of Production Economics, 2010, 126(2): 212-228.

[6] TÜRKAKN O H, ARDITI D, MANISAL E. Comparison of Heuristic Priority Rules in the Solution of the Resource-Con-

1) http://www.om-db.wi.tum.de/psplib/getdata_sm.html

strained Project Scheduling Problem [J]. Sustainability, 2021, 13(7): 1-16.
[7] TOFIGHIAN A A, NADERI B. Modeling and Solving the Project Selection and Scheduling[J]. Computers & Industrial Engineering, 2015, 83(C): 30-38.
[8] SAVITRI N A, PUJAWAN I N, SANTOSA B. Resource-Constrained Project Scheduling with Ant Colony Optimization Algorithm[J]. Journal of Civil Engineering, 2020, 35(2): 34-38.
[9] JIA L, LIU Y S, SHI Y, et al. Solving Resource-Constrained Project Scheduling Problem via Genetic Algorithm[J]. Journal of Computing in Civil Engineering, 2020, 34(2): 04019055.
[10] BOCTOR F F. Resource-Constrained Project Scheduling by Simulated Annealing[J]. International Journal of Production Research, 1996, 34(8): 2335-2351.
[11] PAN N H, HSIANG T C, CHEN W T. Using Hybrid Simulated Annealing Algorithm in Resource Constrained Project Scheduling Problem[J]. International Journal of Organizational Innovation, 2020, 12(3): 25-46.
[12] OMER A. Tabu Search and An Exact Algorithm for the Solutions of Resource-constrained Project Scheduling Problems[J]. International Journal of Computational Intelligence Systems, 2011, 4(2): 255-267.
[13] PELLERIN R, PERRIER N, BERTHAUT F. A Survey of Hybrid Metaheuristics for the Resource-Constrained Project Scheduling Problem[J]. European Journal of Operational Research, 2020, 280(2): 395-416.
[14] GOLAB A, GOOYA E, FALOU A, et al. Review of Conventional Metaheuristic Techniques for Resource-Constrained Project Scheduling Problem[J]. Journal of Project Management, 2022, 7(2): 95-110.
[15] PATTERSON J H. A Comparison of Exact Approaches for Solving The Multiple Constrained Resource Project Scheduling Problem[J]. Management Science, 1984, 30(7): 854-867.
[16] ROSTAMI M, MORADINEZHAD D, SOUFIPOUR A. Improved and Competitive Algorithms for Large Scale Multiple Resource-Constrained Project-Scheduling Problems [J]. KSCE Journal of Civil Engineering, 2014, 18(5): 1261-1269.
[17] MINGOZZI A, MANIEZZO V, RICCIARDELLI S, et al. An Exact Algorithm for The Resource-Constrained Project Scheduling Problem Based on A New Mathematical Formulation[J]. Management Science. 1998, 44(5): 714-729.
[18] BRUCKER P, KNUST S. A Linear Programming and Constraint Propagation-Based Lower Bound for The RCPSP[J]. European Journal of Operational Research, 2000, 127(2): 355-362.
[19] DAMAY J, QUILLIOT A, SANLAVILLE E. Linear Programming Based Algorithms for Preemptive and Non-Preemptive RCPSP [J]. European Journal of Operational Research, 2007, 182(3): 1012-1022.
[20] WANG Q, LIU C C, ZHENG L. A Column-Generation-Based Algorithm for A Resource-Constrained Project Scheduling Problem with A Fractional Shared Resource[J]. Engineering Optimization, 2020, 52(5): 798-816.



ZHANG Yu-zhe, born in 1997, postgraduate. Her main research interests include RCPSP optimization, traditional methods and deep learning methods.

DONG Xing-ye, born in 1974, Ph.D, associate professor, is a member of China Computer Federation. His main research interests include scheduling theory, operations research and intelligent optimization algorithms.

(责任编辑:杨雪敏)