

# 带释放时间与固定周期维护约束的单机双代理组合优化模型与分支定界精确算法

刘志威, 杨 洋, 田丽君

重庆科技大学数理科学学院, 重庆

收稿日期: 2026年9月14日; 录用日期: 2026年10月12日; 发布日期: 2026年10月21日

## 摘 要

针对释放时间和固定周期维护约束下的单机双代理调度问题, 以最小化两代理最大完工时间的加权和  $\min Z = C_A + \theta C_B$  为目标, 建立混合整数规划模型。考虑到释放时间、维护停机和不可中断加工共同导致的搜索空间扩张, 设计一种分支定界算法。首先利用多种排序规则及局部邻域变换构造可行解, 从而获得搜索树的初始上界; 再通过放松不可抢占约束建立可抢占松弛下界, 并分别利用释放时间阈值下的有效加工容量约束和周期运行窗口的装箱可行性约束构造容量下界与装箱下界, 取三者最大值形成节点复合下界。在此基础上, 根据节点状态之间的偏序关系建立状态支配规则, 并利用同代理工件的可行插入关系构造插入支配规则, 在保持全局最优性的条件下消除重复状态和劣势分支。数值实验采用不同规模随机实例, 并与混合整数规划求解器CPLEX进行比较。在30组测试实例中, 所构造分支定界算法有18组获得更小的限时目标函数值, 其余12组与CPLEX所得结果相同; 所构造初始上下界的平均最优性间隙约为2.80%, 状态支配规则的总体剪枝命中率达到44.15%。实验结果表明, 基于整数规划、松弛下界与支配关系构造的分支定界方法能够有效压缩离散可行解空间, 并提高该类单机双代理组合优化问题的精确求解效率。

## 关键词

离散组合优化, 混合整数规划, 分支定界, 松弛下界, 支配关系, 单机多代理调度

# A Combinatorial Optimization Model and Exact Branch-and-Bound Algorithm for a Single-Machine Two-Agent Scheduling Problem with Release Dates and Fixed Periodic Maintenance

Zhiwei Liu, Yang Yang, Lijun Tian

文章引用: 刘志威, 杨洋, 田丽君. 带释放时间与固定周期维护约束的单机双代理组合优化模型与分支定界精确算法[J]. 运筹与模糊学, 2026, 16(5): 33-45. DOI: 10.12677/orf.2026.165032

## Abstract

For the single-machine dual-agent scheduling problem under release time and fixed periodic maintenance constraints, this study establishes a mixed-integer programming model with the objective of minimizing the weighted sum of the maximum completion times of the two agents, *i.e.*,  $\min Z = C_A + \theta C_B$ . To address the expansion of the search space caused by release times, maintenance downtime, and non-preemptive processing, a branch-and-bound algorithm is designed. First, feasible solutions are constructed using multiple sorting rules and local neighborhood transformations to obtain the initial upper bound of the search tree. Subsequently, a preemptive relaxation is applied by relaxing the non-preemptive constraint to derive a preemptive lower bound; additionally, capacity lower bounds and packing lower bounds are constructed by employing, respectively, the effective processing capacity constraint under a release time threshold and the packing feasibility constraint under a periodic operation window. The maximum of these three bounds is then taken to form the composite node lower bound. Building upon this foundation, a state domination rule is established based on the partial order between node states, and an insertion domination rule is constructed using the feasible insertion relationships among workpieces belonging to the same agent. This approach eliminates duplicate states and suboptimal branches while preserving global optimality. Numerical experiments were conducted using randomly generated instances of varying sizes, and the results were compared against those obtained by the mixed-integer programming solver CPLEX. Across 30 test instances, the proposed branch-and-bound algorithm achieved smaller time-limited objective function values in 18 instances, while in the remaining 12 instances, the results were identical to those of CPLEX; the average optimality gap of the constructed initial upper and lower bounds was approximately 2.80%, and the overall pruning hit rate of the state domination rule reached 44.15%. Experimental results demonstrate that the branch-and-bound method—constructed based on integer programming, relaxation lower bounds, and dominance relations—can effectively compress the discrete feasible solution space and improve the efficiency of solving this class of single-machine dual-agent combinatorial optimization problems exactly.

## Keywords

Discrete Combinatorial Optimization, Mixed-Integer Programming, Branch-and-Bound, Relaxation Lower Bounds, Dominance Relations, Single-Machine Multi-Agent Scheduling

Copyright © 2026 by author(s) and Hans Publishers Inc.

This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

## 1. 引言

单机调度问题是组合优化领域中的经典研究问题，由于工件排序决策具有排列组合特征，随着问题规模增加，其可行解空间通常呈指数增长，导致大量调度问题具有较高的计算复杂性。Lenstra 等[1]从计算复杂性角度对机器调度问题进行了系统分析，指出许多调度模型属于 NP-hard 问题。因此，如何建立有效的数学规划模型并设计高效精确算法一直是离散优化领域的重要研究方向。

双代理单机调度是在经典单机调度问题基础上发展形成的一类组合优化模型。Yin 等[2]研究了具有可指派交期约束的双代理单机调度问题,分析了不同目标组合下问题的复杂性,并设计了相应求解方法。Liu 等[3]考虑加工时间变化因素下的双代理调度问题,研究了代理目标与排序结构之间的关系。He 和 Leung [4]进一步研究了时变加工条件下的双代理调度问题,并分析了组合优化结构下的求解性质。Zhang 和 Wang [5]针对双代理调度中的加权延迟目标,研究了不同目标组合形式下的问题复杂性。Yu 等[6]进一步考虑释放时间约束,研究了以两个代理最大完工时间加权和为目标的单机双代理调度问题,并建立混合整数规划模型与近似算法。

经典双代理调度问题通常假设所有工件在初始时刻均可加工,而实际离散优化模型中,工件释放时间会进一步限制可行调度集合。Dyer 和 Wolsey [7]从整数规划角度研究了带释放时间单机排序问题,建立了相应混合整数规划模型,并通过松弛方法分析问题结构。Yu 等[6]针对带释放时间的双代理单机调度问题,研究了释放约束下两个代理最大完工时间之间的协调关系,并提出相应算法。除释放时间外,机器不可用约束也是影响单机调度可行结构的重要因素。Hsu 等[8]研究了带维护活动的单机调度问题,并建立相应整数规划模型。Zhao 和 Tang [9]针对维护约束下的单机调度问题,研究了机器维护活动与加工过程之间的数学关系。Hfaiedh 等[10]进一步研究了包含机器不可用区间的单机调度问题,并利用松弛方法构造有效下界,设计分支定界算法求解精确解。

对于具有复杂约束的离散组合优化问题,分支定界算法是一类重要的精确求解方法。Lee 等[11]针对双代理单机调度问题设计了分支定界算法,并利用界限条件提高求解效率。Luo 和 Chu [12]研究了具有复杂排序约束的单机调度问题,构造上下界信息并提出分支定界方法。

本文的主要贡献包括以下两个方面:(1) 以两个代理最大完工时间的加权和最小化为目标,建立带释放时间和固定周期维护的单机双代理调度混合整数规划模型。(2) 提出一种分支定界算法,有效解决了面向该问题的中小规模车间调度求解。

## 2. 问题描述与建模

### 2.1. 问题描述

设单台机器加工代理 A 和代理 B 的工件,工件集合为  $J = J_A \cup J_B$  且  $J_A \cap J_B = \emptyset$ 。工件  $i$  具有加工时间  $p_i$  和释放时间  $r_i$ ,只能在  $r_i$  之后开始。每个维护周期包含长度为  $T$  的运行窗口和长度为  $U$  的固定维护区间,周期长度  $H = T + U$ 。第  $k$  个运行窗口为  $[kH, kH + T)$ ,维护区间为  $[kH + T, (k+1)H)$ ,  $k \in \mathbb{Z}_+$ 。所有工件均不可抢占、不可跨维护区间,且任意时刻至多加工一个工件。目标是最小化两个代理最大完工时间的加权和。

分支定界节点以已排前缀及其最早可行解码状态表示。设节点  $v$  位于深度  $\tau$ ,前缀为  $\pi_\tau$ ,未排集合为  $R_\tau = J \setminus \pi_\tau$ ,前缀最后完工时刻为  $t_\tau$ ,当前运行窗口剩余容量为  $\omega_\tau$ ,前缀中代理 A、B 的最大完工时间分别为  $C_A^\tau$ 、 $C_B^\tau$ ,节点下界为  $LB_\tau$ ,则节点状态为:

$$v = (\tau, \pi_\tau, R_\tau, t_\tau, \omega_\tau, C_A^\tau, C_B^\tau, LB_\tau) \quad (1)$$

给定绝对时刻  $t_\tau$ ,  $\omega_\tau$  可由固定日历唯一确定。若  $t_\tau$  位于运行窗口内,则  $\omega_\tau$  为该窗口终点与  $t_\tau$  之差;若  $t_\tau$  位于维护区间,则状态先推进至下一运行窗口起点并令  $\omega_\tau = T$ 。扩展节点时,算法选择  $j \in R_\tau$ ,按  $r_j$  和固定运行窗口计算其最早可行开工与完工时刻,再更新  $t_\tau$ 、 $\omega_\tau$  及相应代理的前缀最大完工时间。

### 2.2. 问题建模

表 1 给出主要集合、参数和变量。

**Table 1.** Main notation of the mixed-integer programming model  
**表 1.** 混合整数规划模型主要符号

| 符号         | 含义                                           |
|------------|----------------------------------------------|
| $J$        | 全部工件集合, $J = J_A \cup J_B$                   |
| $J_A, J_B$ | 代理 A、B 的工件集合                                 |
| $n$        | 工件总数 $ J $                                   |
| $p_i$      | 工件 $i$ 的加工时间                                 |
| $r_i$      | 工件 $i$ 的释放时间                                 |
| $T$        | 每个周期的可加工时长                                   |
| $U$        | 固定维护时长                                       |
| $H$        | 周期长度, $H = T + U$                            |
| $\theta$   | 代理 B 最大完工时间的非负权重                             |
| $S_i$      | 工件 $i$ 的开工时刻(整数)                             |
| $q_i$      | 工件 $i$ 所属的固定运行周期索引(整数, $q_i \geq 0$ )        |
| $x_{ij}$   | $i < j$ 时的先后变量; $x_{ij} = 1$ 表示 $i$ 在 $j$ 之前 |
| $C_A, C_B$ | 代理 A、B 的最大完工时间                               |
| $Q_{\max}$ | 安全的最大周期索引                                    |
| $M$        | 两两互斥约束中的充分大常数                                |

目标函数为两个代理最大完工时间的加权和:

$$\min Z = C_A + \theta C_B \quad (2)$$

混合整数规划模型:

$$S_i \geq r_i, \forall i \in J \quad (3)$$

$$Hq_i \leq S_i, \forall i \in J \quad (4)$$

$$S_i + p_i \leq Hq_i + T, \forall i \in J \quad (5)$$

$$S_i + p_i \leq S_j + M(1 - x_{ij}), \forall i < j \quad (6)$$

$$S_j + p_j \leq S_i + Mx_{ij}, \forall i < j \quad (7)$$

$$C_A \geq S_i + p_i, \forall i \in J_A \quad (8)$$

$$C_B \geq S_i + p_i, \forall i \in J_B \quad (9)$$

$$0 \leq S_i \leq C^{UB}, S_i \in \mathbb{Z}^+, \forall i \in J \quad (10)$$

$$q_i^{\min} \leq q_i \leq Q_{\max}, q_i \in \mathbb{Z}^+, \forall i \in J \quad (11)$$

$$0 \leq C_A, C_B \leq C^{UB}, C_A, C_B \in \mathbb{Z}^+ \quad (12)$$

$$x_{ij} \in \{0, 1\}, \forall i < j \quad (13)$$

式(2)~(10)构成完整的混合整数规划模型。式(2)为目标函数,即最小化代理 A 与代理 B 最大完工时间的加权和;式(3)为工件释放时间约束,保证每个工件不得早于其释放时刻开始加工。式(4)~(5)为固定维护窗口约束,其中式(4)限定工件开工时刻不早于所属运行周期的起点,式(5)保证工件在该运行窗口结束前

完成,从而禁止工件在维护期间加工或跨越维护区间。式(6)~(7)为单机互斥约束,通过二元变量  $x_{ij}$  确定任意两个工件的加工先后关系,保证任意时刻机器至多加工一个工件;式(8)~(9)分别定义代理 A 和代理 B 的最大完工时间。式(10)~(13)给出各决策变量的取值范围以及运行周期索引的安全上下界。

### 3. 分支定界算法

分支定界算法是一种用于求解组合优化问题的精确式算法,它基于枚举的思想,通过系统搜索解空间寻找最优解。分支定界算法的性能主要取决于问题上界,搜索方式和剪枝策略。本节针对问题特征设计了有效的问题上下界和剪枝规则。

#### 3.1. 上界

优秀的上界能够有效提升算法效率,本文构造多策略上界生成方法。对于任一搜索节点  $v$ ,首先固定当前已排前缀  $\pi_r$ ,并针对剩余工件集合  $R_r$  分别采用五种启发式规则生成候选补全序列。再利用统一的固定维护解码器对各候选序列进行可行性解码,并按照目标函数  $Z = C_A + \theta C_B$  计算其真实目标值,五种启发式规则如下。

**(1) 目标平衡策略。**该策略综合考虑候选工件加入当前序列后对两个代理最大完工时间及剩余负荷的影响,优先选择有利于改善  $C_A + \theta C_B$  整体水平的工件,从而避免调度构造过程过度偏向单一代理,提高双代理目标之间的协调性。

**(2) 当前窗口最佳填充策略。**对满足释放时间约束的候选工件,根据其加工时间与当前运行窗口剩余容量  $\omega_r$  的匹配程度进行选择,优先安排能够充分利用当前可加工容量的工件,以减少维护前的剩余容量损失。若候选工件无法在当前窗口内完成,则转入下一运行窗口重新进行评价。

**(3) 代理优先策略。**分别以代理 A 和代理 B 为优先对象构造候选补全序列,以反映不同目标权重及代理负荷分布下可能出现的瓶颈差异。最终不直接依据预设代理优先级确定结果,而是通过统一解码后的实际目标函数值选取较优调度。

**(4) 最长加工时间优先策略。**优先选择加工时间较长的工件进行安排,以降低长工件在后续运行窗口中因剩余容量不足而被多次顺延的可能性,从而减弱固定维护窗口对长加工任务造成的累积影响。

**(5) 最早释放时间优先策略。**优先调度释放时间较早的工件,使已具备加工条件的任务能够尽早进入机器加工,并减少由释放时间限制引起的机器等待,从而形成较为稳定的基准补全序列。

在此基础上,对当前最佳完整序列进一步执行插入和交换局部操作,并仅接受能够严格改善目标函数值的邻域解,从而进一步提高上界质量。通过上述方法可在较小计算开销下获得较优的可行调度,为后续下界剪枝提供更有效的上界信息。具体过程如**算法 1**所示。

**Algorithm 1.** Multi-strategy upper bound construction

**算法 1.** 多策略上界构建算法

|                                                                                                                                                                                                                                                                  |                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| Input                                                                                                                                                                                                                                                            | J, T, UT, $\theta$ //工件集合、固定维护窗口参数及目标权重 |
| Output                                                                                                                                                                                                                                                           | UB                                      |
| $UB \leftarrow +\infty, \pi^* \leftarrow \emptyset$<br>//初始化当前最优上界与调度序列<br><b>for</b> $h \in H$ <b>do</b> // $H = \{\text{五种策略}\}$<br>$\pi_h \leftarrow \text{Generate}(h)$<br><b>end for</b><br>$F(\pi_h) \leftarrow \text{Decode}(\pi_h)$<br>//基于固定维护约束计算调度目标值 |                                         |

```

if  $F(\pi_h) < UB$  then
     $UB \leftarrow F(\pi_h), \pi^* \leftarrow \pi_h$ 
end if // 更新最优可行解

 $\pi^* \leftarrow \text{LocalSearch}(\pi^*)$ 
    LocalSearch = {Insertion, Swap}
// 局部操作改善解质量

return  $(\pi^*, UB)$ 
// 输出初始最优解及上界值
    
```

### 3.2. 下界

#### (1) 下界 1

下界 1 通过放宽工件不可中断加工的约束，允许尚未调度的工件被分段处理，并在后续时刻继续加工。由于该松弛问题扩大了原问题的可行域，其最优目标值不会高于原问题的最优目标值，因此可将松弛问题的最优值或其有效估计值作为原问题的下界，见图 1。定义下界 1 为  $LB_\tau^1 = L_A^1 + \theta L_B^1$ 。

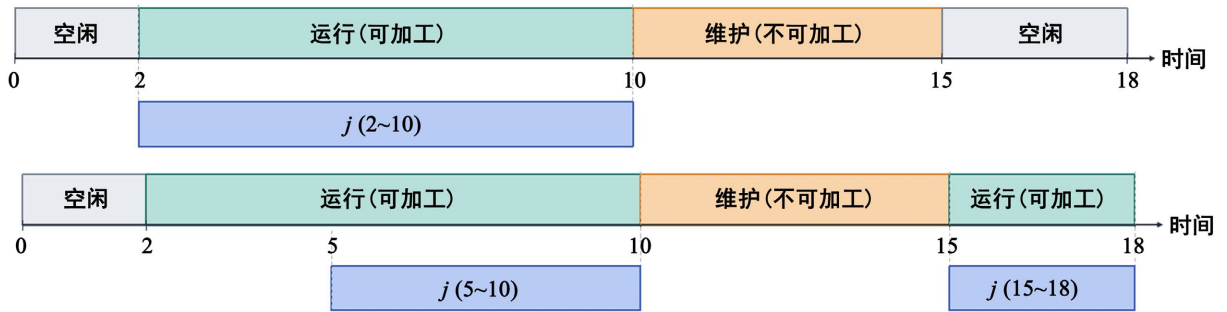


**Figure 1.** Example diagram of preemptive relaxation during processing  
**图 1.** 加工中中断松弛示例图

#### (2) 下界 2

可抢占松弛下界主要考虑剩余工件的总体加工量及固定维护造成的容量中断，但当多个工件集中在较晚时刻释放时，机器在前期存在的空闲容量不能用于加工这些工件，因此仍可能低估其实际完成时间。为进一步反映释放时间形成的后置加工压力，构造释放时间容量下界。

设  $Q \subseteq R_\tau$  为任意剩余工件集合。对于某一释放阈值  $\rho$ ，所有满足  $\max\{t_r, r_i\} \geq \rho$  的工件均不能在  $\rho$  之前完成其加工，因此这些工件的总加工量必须由  $\rho$  之后的可用机器容量承担。记  $\Gamma(s, P)$  为从时刻  $s$  开始，在固定维护日历下累计获得  $P$  个单位有效加工容量所对应的最早时刻，则集合  $Q$  的释放时间容量下界可表示为

$$F^2(Q, t_r) = \max_{\rho \in D(Q, t_r)} \Gamma\left(\rho, \sum_{\max\{t_r, r_i\} \geq \rho} p_i\right)$$

$$D(Q, t_r) = \{\max\{t_r, r_i\} : i \in Q\}$$

该式只考察剩余工件实际可能形成约束的释放时刻。其含义是：对于任一释放阈值  $\rho$ ，若一组工件只能在  $\rho$  之后加工，则其完成时刻至少应覆盖这部分工件所需的全部有效加工容量。下界 2 定义为  $LB_\tau^2 = L_A^2 + \theta L_B^2$ 。

#### (3) 下界 3

前两个下界主要从有效加工容量角度进行估计，但仍没有完全体现工件不可中断的特点。由于每个

工件必须完整安排在同一运行窗口内,即使截止时刻之前的总有效容量足够,也可能由于各运行窗口剩余容量过于分散而无法安排全部工件。因此,进一步构造运行段装箱下界。

对于给定截止时刻  $d$ ,将区间  $[t_r, d)$  内各运行窗口的可用部分视为若干容量有限的“箱体”,将工件加工时间  $p_i$  视为待装入箱体的“物品”。若集合  $Q$  能够在时刻  $d$  前完成,则其中每个工件至少必须能够完整放入某一个运行窗口,且同一窗口内所安排工件的总加工时间不能超过该窗口的有效容量。

据此,将集合  $Q$  的最早装箱可行时刻定义为

$$F^3(Q, t_r) = \min\{d \geq t_r : Q \text{ 能完整分配至 } [t_r, d) \text{ 内的运行窗口}\}.$$

在下界 3 中暂不考虑工件释放时间以及集合  $Q$  之外工件的机器竞争,因此该装箱问题是原调度问题的放松。若在某一时刻  $d$  前连上述放松后的装箱条件都无法满足,则原问题显然也不可能在  $d$  前完成集合  $Q$ 。

下界 3 定义为  $LB_r^3 = L_A^3 + \theta L_B^3$ 。

综合三个层次的估计,节点最终下界取为

$$LB_r = \max\{LB_r^1, LB_r^2, LB_r^3\}$$

其中,  $LB_r^1$  主要反映释放时间和固定维护条件下的可抢占完成能力,  $LB_r^2$  进一步考虑晚释放工件形成的后置加工压力,  $LB_r^3$  则补充工件必须完整落入运行窗口的不可中断结构。三个下界由松到紧逐层增加问题结构信息。

### 3.3. 剪枝

剪枝规则用于删除不可能产生更优目标值的搜索节点或分支。剪枝能够减少节点扩展数量,但其成立条件必须是原问题的必要条件或严格支配关系,否则可能破坏分支定界算法的完备性。本文仅使用下界剪枝、同剩余集合状态支配和同代理插入支配三类规则。

#### (1) 下界剪枝

下界剪枝是分支定界算法中最基本的剪枝方式。设当前已知最好可行调度的目标函数值为  $UB$ ,当前节点  $N$  的综合下界为  $LB(N)$ 。若满足  $LB(N) \geq UB$ ,则说明从节点  $N$  继续扩展得到的任意完整调度解都不可能优于当前最好可行解。因此,节点  $N$  可以被直接剪去。

#### (2) 同状态支配剪枝

固定维护日历完全由绝对时刻确定。对于任意工件  $i$ ,记  $E_i(t)$  为从时刻  $t$  开始安排该工件时的最早可行完工时刻。由于推迟可用起点不会使工件更早完成,因此,当  $t_1 \leq t_2$  时,有  $E_i(t_1) \leq E_i(t_2)$ 。

**规则 2 (状态支配):** 设节点  $v_1$  与节点  $v_2$  具有相同的剩余工件集合,即  $R_1 = R_2$ 。若  $t_1 \leq t_2$ ,  $C_{A,1} \leq C_{A,2}$ ,  $C_{B,1} \leq C_{B,2}$ ,则称节点  $v_1$  支配节点  $v_2$ ,可以删除节点  $v_2$ 。

**证明:** 任取节点  $v_2$  的一个完整后缀序列  $\sigma = (j_1, j_2, \dots, j_m)$ 。由于两个节点的剩余工件集合相同,序列  $\sigma$  同样可以用于补全节点  $v_1$ 。由  $t_1 \leq t_2$  以及最早可行完工算子  $E_i(t)$  关于  $t$  的单调性,在两个节点上依次按照相同顺序安排  $j_1, j_2, \dots, j_m$ ,可归纳得到节点  $v_1$  中每个后续工件的开工和完工时刻均不晚于节点  $v_2$  中对应工件的时刻。同时,保证节点  $v_1$  的两个代理前缀最大完工时间均不大于节点  $v_2$ 。因此,完成相同后缀后有

$$C_A(v_1, \sigma) \leq C_A(v_2, \sigma),$$

$$C_B(v_1, \sigma) \leq C_B(v_2, \sigma).$$

节点  $v_2$  的每一个完整补全方案,都能在节点  $v_1$  中找到目标值不大的对应方案。因此,节点  $v_2$  不可能产生节点  $v_1$  无法达到的更优目标值,删除该节点不会损失全局最优解。证毕。

### (3) 同代理插入支配剪枝

设节点前缀的最后一个工件为  $h$ ，其所属代理为  $x$ ，完工时刻为  $C_h$ ，完工后当前运行段的剩余容量为  $\omega_h$ 。考虑与  $h$  同属代理  $x$  的候选工件  $j$ ，记  $\alpha_j$  为不插入其他工件时， $j$  在  $h$  之后的最早可行开工时刻；记  $E_q(C_h)$  为另一剩余工件  $q$  从  $C_h$  开始安排时的最早可行完工时刻。

**规则 3 (同代理插入支配):** 若存在工件  $q \in R_t \setminus \{j\}$ ,  $a_q = a_j = a_h = x$ ，并满足

$$p_q \leq \omega_h, E_q(C_h) \leq \alpha_j,$$

则以工件  $j$  作为当前下一工件的分支可以删除。式中的第一项是运行段容量筛选条件，表示工件  $q$  的加工时间不超过工件  $h$  完工后的剩余容量；第二项为实际无延迟条件，它同时考虑  $q$  的释放时间和固定维护日历，保证将  $q$  插入  $h$  与  $j$  之间后不会推迟  $j$  的最早开工时刻。

**证明:** 任取以  $j$  为下一工件的分支中的一个完整补全方案。由于  $q$  尚未调度，它必然出现在该补全序列中  $j$  之后的某一位置。现将  $q$  从原位置移出，并将其插入  $h$  与  $j$  之间。

工件  $q$  可以从  $C_h$  开始按照最早可行方式安排，并且其完工时刻不晚于  $\alpha_j$ 。因此，插入  $q$  后，工件  $j$  仍可在原最早开工时刻  $\alpha_j$  开始，或者更早开始。删除  $q$  在原后缀中的位置后，位于其后的其他工件不会被推迟。于是，变换后的调度满足释放时间、固定维护、单机互斥和不可中断加工约束。

在该变换中，工件  $q$  的完工时刻提前，工件  $j$  及其他工件的完工时刻均不增加。因此，代理  $A$  和代理  $B$  的最大完工时间均不会增大，进而有

$$C^{A,(q)} \leq C^{A,(j)}, C^{B,(q)} \leq C^{B,(j)} \Rightarrow Z^{(q)} \leq Z^{(j)}$$

变换后的调度属于以  $q$  为下一工件的另一分支。这说明，以  $j$  为下一工件的任意完整方案，都存在一个目标值不大的  $q$  优先方案，故删除  $j$  分支不会剪去全局最优目标值。证毕。

### 3.4. 分支定界算法流程

① 构造根节点，利用多策略上界构建方法获得初始上界  $UB$ ，并计算节点综合下界  $LB$ ；② 按深度优先策略选择开放节点，若满足  $LB \geq UB$ ，则直接剪枝；③ 对当前节点进行状态支配检验，若被已有状态支配，则删除该节点；④ 对剩余工件依次生成子节点，并对满足条件的同代理候选工件执行插入支配剪枝；⑤ 对保留的子节点按照最早可行方式解码，更新节点状态并重新计算下界；⑥ 对具有改进潜力的节点调用上界构造方法，若获得更优完整调度，则更新  $UB$ 。重复上述过程，直至开放节点集合为空或达到设定时间限制。算法流程如下图 2。

## 4. 算法结果与分析

本章通过仿真实验对所提出算法的性能进行评估。所有仿真实验均在 Intel(R) Core(TM) i5-11800H 2.30 GHz、8 GB 内存的计算机上进行。本文使用 Python 3.8 编写算法的测试程序，在 ILOG CPLEX 12.9.0 求解器软件上建立数学模型进行求解。

由于本文研究的问题没有标准测试算例，因此下文使用的测试实例均随机生成。工件数量  $N$  分别取 10、20、30、40、50，对应的最大可连续加工时间  $T$  分别为 15、20、25、30、30，维护所需时间  $UT$  分别为 5、5、10、10、15。大工件加工时间  $p_j$  随机生成，并满足  $p_j \leq T$ ，保证每个工件均可在一个维护周期内完成加工。设所有工件加工时间之和为  $P$ ，工件释放时间  $r_j$  在区间  $[0, \alpha P]$  内随机生成，其中  $\alpha = 0.3, 0.5$ 。当  $\alpha = 0.3$  时，工件释放时间较为集中，当  $\alpha = 0.5$  时，工件释放时间分布更加分散。为分析代理权重变化对调度结果的影响，本文设置代理  $B$  的权重系数为  $\theta = 0.5, 1, 1.5$ 。其中， $\theta = 0.5$  表示代理  $A$  具有更高优先级， $\theta = 1$  表示两个代理具有相同的优先级， $\theta = 1.5$  表示代理  $B$  具有更高优先级。

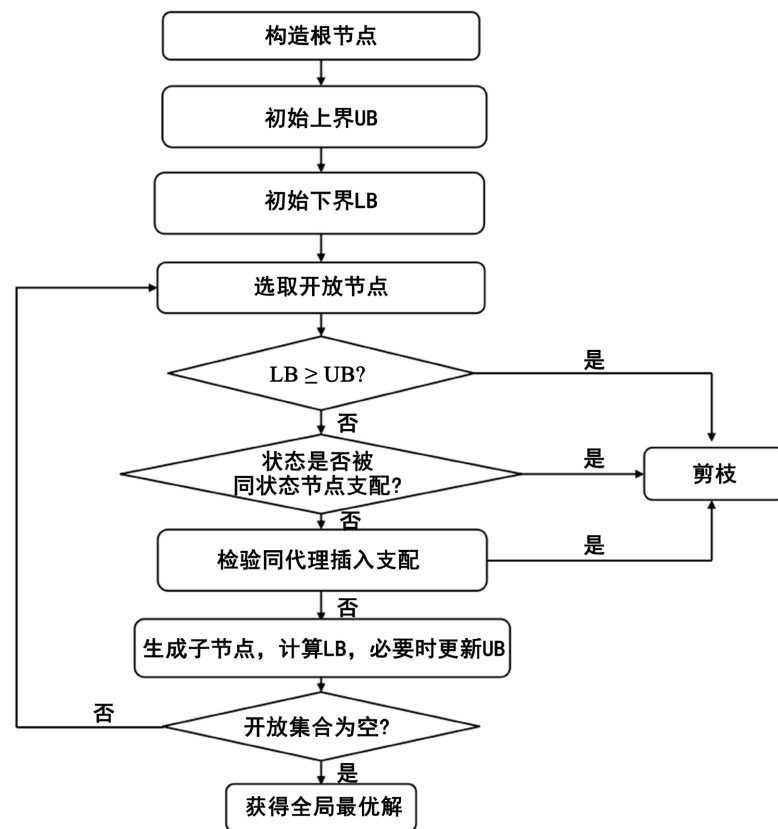


**Figure 2.** Flowchart of the branch-and-bound framework using the retained module  
**图 2.** 保留模块下的分支定界算法流程

#### 4.1. B&B 与 CPLEX 结果比较

表 2 汇总 30 组既有对比记录。

**Table 2.** Recorded objective values and running times of B&B and CPLEX

**表 2.** 分支定界与 CPLEX 的既有目标值和运行时间记录

| N  | $\alpha$ | T  | UT | $\theta = 0.5$ |        |              |       | $\theta = 1$ |       |              |       | $\theta = 1.5$ |        |              |       |
|----|----------|----|----|----------------|--------|--------------|-------|--------------|-------|--------------|-------|----------------|--------|--------------|-------|
|    |          |    |    | Obj            | Obj    | Time         | Time  | Obj          | Obj   | Time         | Time  | Obj            | Obj    | Time         | Time  |
|    |          |    |    | B&B            | CPLEX  | B&B          | CPLEX | B&B          | CPLEX | B&B          | CPLEX | B&B            | CPLEX  | B&B          | CPLEX |
| 10 | 0.3      | 15 | 5  | <b>131</b>     | 131    | <b>0.03</b>  | 0.17  | <b>196</b>   | 196   | <b>0.04</b>  | 0.16  | <b>229.5</b>   | 229.5  | <b>0.19</b>  | 0.12  |
|    | 0.5      |    |    | <b>138.5</b>   | 138.5  | <b>0.15</b>  | 0.05  | <b>202</b>   | 202   | <b>0.24</b>  | 0.07  | <b>237.5</b>   | 237.5  | <b>0.13</b>  | 0.07  |
| 20 | 0.3      | 20 | 5  | <b>277.5</b>   | 277.5  | <b>1.71</b>  | 25.28 | <b>411</b>   | 411   | <b>7.54</b>  | 600.0 | <b>481.5</b>   | 481.5  | <b>2.19</b>  | 600.0 |
|    | 0.5      |    |    | <b>283</b>     | 283    | <b>0.60</b>  | 18.05 | <b>425</b>   | 425   | <b>1.07</b>  | 176.1 | <b>531</b>     | 531    | <b>0.87</b>  | 218.1 |
| 30 | 0.3      | 25 | 10 | <b>419</b>     | 456    | <b>6.14</b>  | 600.0 | <b>624</b>   | 674   | <b>8.79</b>  | 600.0 | <b>748.5</b>   | 786    | <b>7.01</b>  | 600.0 |
|    | 0.5      |    |    | <b>419</b>     | 455.5  | <b>6.88</b>  | 600.0 | <b>624</b>   | 687   | <b>7.93</b>  | 600.0 | <b>762.5</b>   | 784.5  | <b>11.83</b> | 600.0 |
| 40 | 0.3      | 30 | 10 | <b>820</b>     | 886    | <b>164.5</b> | 600.0 | <b>1223</b>  | 1360  | <b>149.2</b> | 600.0 | <b>1445</b>    | 1553   | <b>154.9</b> | 600.0 |
|    | 0.5      |    |    | <b>827.5</b>   | 884    | <b>198.1</b> | 600.0 | <b>1232</b>  | 1352  | <b>156.8</b> | 600.0 | <b>1448.5</b>  | 1595.5 | <b>147.7</b> | 600.0 |
| 50 | 0.3      | 30 | 15 | <b>1013</b>    | 1289   | 600.0        | 600.0 | <b>1528</b>  | 1863  | 600.0        | 600.0 | <b>1869.5</b>  | 1994   | 600.0        | 600.0 |
|    | 0.5      |    |    | <b>1053</b>    | 1277.5 | 600.0        | 600.0 | <b>1579</b>  | 1781  | 600.0        | 600.0 | <b>1928</b>    | 2201   | 600.0        | 600.0 |

由表 2 可知, 在 30 组测试算例中, B&B 有 18 组获得了低于 CPLEX 的最好可行目标值, 12 组与 CPLEX 结果相同。对于较小规模算例, 两种方法所得结果基本一致, 但 B&B 整体具有较快的求解速度。随着工件数量增加, CPLEX 逐渐难以在 600 s 内获得高质量解, 而 B&B 仍能利用上下界和剪枝规则有效缩小搜索空间。特别是在  $N = 50$  时, B&B 在 6 组算例中所得目标值均低于 CPLEX, 表明所提出算法在中大规模问题上具有较好的限时求解能力。

#### 4.2. 算法性能统计

**Table 3.** Initial bounds obtained by the retained construction  
**表 3.** 保留模块构造的初始上下界

| $N$ | $\alpha$ | $\theta = 0.5$ |          |             | $\theta = 1$ |          |             | $\theta = 1.5$ |          |             |
|-----|----------|----------------|----------|-------------|--------------|----------|-------------|----------------|----------|-------------|
|     |          | 初始<br>上界       | 初始<br>下界 | Gap%<br>B&B | 初始<br>上界     | 初始<br>下界 | Gap%<br>B&B | 初始<br>上界       | 初始<br>下界 | Gap%<br>B&B |
| 10  | 0.3      | 131            | 131      | 0.00        | 196          | 196      | 0.00        | 261            | 250      | 4.21        |
|     | 0.5      | 138.5          | 136      | 1.81        | 204          | 197      | 3.43        | 237            | 232      | 2.11        |
| 20  | 0.3      | 277.5          | 275      | 0.90        | 412          | 395      | 4.13        | 483            | 461      | 4.55        |
|     | 0.5      | 283            | 275      | 2.83        | 425          | 410      | 3.53        | 531            | 501      | 5.65        |
| 30  | 0.3      | 426            | 419      | 1.64        | 639          | 624      | 2.35        | 756            | 734      | 2.91        |
|     | 0.5      | 428            | 419      | 2.10        | 641          | 624      | 2.65        | 772            | 747      | 3.24        |
| 40  | 0.3      | 830            | 810      | 2.41        | 1239         | 1204     | 2.82        | 1457           | 1411     | 3.16        |
|     | 0.5      | 852            | 826      | 3.05        | 1245         | 1227     | 1.45        | 1460           | 1437     | 1.58        |
| 50  | 0.3      | 1030           | 1009     | 2.04        | 1556         | 1520     | 2.31        | 1880           | 1836     | 2.34        |
|     | 0.5      | 1064           | 1014     | 4.70        | 1598         | 1525     | 4.57        | 1978           | 1889     | 4.50        |

**Table 4.** Initial bounds of the baseline construction  
**表 4.** 基线构造的初始上下界

| $N$ | $\alpha$ | $\theta = 0.5$ |          |             | $\theta = 1$ |          |             | $\theta = 1.5$ |          |             |
|-----|----------|----------------|----------|-------------|--------------|----------|-------------|----------------|----------|-------------|
|     |          | 初始<br>上界       | 初始<br>下界 | Gap%<br>B&B | 初始<br>上界     | 初始<br>下界 | Gap%<br>B&B | 初始<br>上界       | 初始<br>下界 | Gap%<br>B&B |
| 10  | 0.3      | 382.5          | 77       | 79.87       | 510          | 102      | 80.00       | 637.5          | 127      | 80.08       |
|     | 0.5      | 412.5          | 85       | 79.39       | 550          | 117      | 78.73       | 687.5          | 149      | 78.33       |
| 20  | 0.3      | 879.5          | 148.5    | 83.10       | 1177         | 202      | 82.83       | 1474.5         | 254      | 82.77       |
|     | 0.5      | 954.5          | 177      | 81.46       | 1277         | 235      | 81.60       | 1599.5         | 293      | 81.68       |
| 30  | 0.3      | 1757           | 172      | 90.21       | 2347         | 232      | 90.12       | 2937           | 292      | 90.06       |
|     | 0.5      | 1862           | 252.5    | 86.44       | 2487         | 348      | 86.01       | 3112           | 443.5    | 85.75       |
| 40  | 0.3      | 2732           | 356      | 86.97       | 3647         | 487      | 86.65       | 4562           | 598      | 86.89       |
|     | 0.5      | 2912           | 561      | 80.73       | 3887         | 732      | 81.17       | 4862           | 903      | 81.43       |
| 50  | 0.3      | 3392           | 379.5    | 88.81       | 4527         | 508      | 88.78       | 5662           | 635      | 88.78       |
|     | 0.5      | 3632           | 598      | 83.54       | 4847         | 810      | 83.29       | 6062           | 1022     | 83.14       |

由表 3 可知, 本文所采用的上下界构造方法能够获得较紧的初始界限, 30 组算例的平均初始 Gap 约

为 2.80%，最大值仅为 5.65%，部分小规模算例甚至在根节点即可达到上下界一致。说明多启发式上界、局部搜索以及考虑释放时间和固定维护结构的下界能够在搜索初期显著缩小可行解与理论下界之间的距离。与表 4 中的基线构造相比，其平均初始 Gap 约为 83.95%，明显高于本文方法。两者的差异说明，将释放时间、固定维护窗口等问题特征融入上下界构造后，可以显著提高界限质量，为后续剪枝和搜索提供更有效的基础。

**Table 5.** State-dominance statistics by job scale

**表 5.** 按工件规模汇总的状态支配统计

| 任务规模 $n$ | 支配检查    | 直接剪枝   | 剪枝率    |
|----------|---------|--------|--------|
| 10       | 71      | 7      | 9.86%  |
| 20       | 1063    | 284    | 26.72% |
| 30       | 322     | 80     | 24.84% |
| 40       | 42,718  | 19,399 | 45.41% |
| 50       | 110,936 | 48,710 | 43.91% |
| 合计       | 155,110 | 68,480 | 44.15% |

**Table 6.** State-dominance statistics by objective weight

**表 6.** 按目标权重汇总的状态支配统计

| 目标权重 $\theta$ | 支配检查    | 直接剪枝   | 剪枝率    |
|---------------|---------|--------|--------|
| 0.5           | 57,217  | 25,046 | 43.77% |
| 1.0           | 58,855  | 28,127 | 47.79% |
| 1.5           | 39,038  | 15,307 | 39.21% |
| 合计            | 155,110 | 68,480 | 44.15% |

由表 5 和表 6 可知，30 组实验共执行 155,110 次状态支配检查，其中 68,480 次节点被直接支配并剪除，总体命中率达到 44.15%，另有 27,885 条旧状态被更优状态反向支配并移除。因此，以相同剩余工件集合为基础，同时比较当前时刻及两代理前缀最大完工时间，能够有效识别重复或劣势状态，从而减少无效搜索节点。

### 4.3. 工件结果示例

表 7 给出了一个包含 20 个 A 类工件和 20 个 B 类工件的固定实例。该实例工件总加工时间为 582，最大释放时间为 148。图 3 给出了该实例对应的最好可行调度。各工件均满足释放时间和固定维护约束，验证了模型与固定维护解码方法的可行性。该示例在 114.9 s 剪去所有劣分支求得最优解为 1271。

**Table 7.** Job data of the recorded 40-job instance

**表 7.** 既有 40 工件固定实例数据

| A 工件 | $p$ | $r$ | B 工件 | $p$ | $r$ |
|------|-----|-----|------|-----|-----|
| J1   | 24  | 142 | J21  | 21  | 133 |
| J2   | 12  | 77  | J22  | 11  | 68  |
| J3   | 22  | 59  | J23  | 15  | 141 |
| J4   | 19  | 131 | J24  | 8   | 72  |

续表

|     |    |     |     |    |     |
|-----|----|-----|-----|----|-----|
| J5  | 4  | 120 | J25 | 17 | 139 |
| J6  | 25 | 148 | J26 | 3  | 148 |
| J7  | 10 | 66  | J27 | 14 | 126 |
| J8  | 16 | 104 | J28 | 12 | 64  |
| J9  | 7  | 57  | J29 | 23 | 107 |
| J10 | 20 | 137 | J30 | 9  | 132 |
| J11 | 13 | 45  | J31 | 15 | 71  |
| J12 | 18 | 124 | J32 | 5  | 144 |
| J13 | 6  | 63  | J33 | 18 | 35  |
| J14 | 23 | 100 | J34 | 22 | 124 |
| J15 | 9  | 129 | J35 | 12 | 146 |
| J16 | 14 | 52  | J36 | 20 | 91  |
| J17 | 17 | 88  | J37 | 7  | 48  |
| J18 | 5  | 37  | J38 | 25 | 118 |
| J19 | 21 | 143 | J39 | 13 | 81  |
| J20 | 11 | 9   | J40 | 16 | 103 |

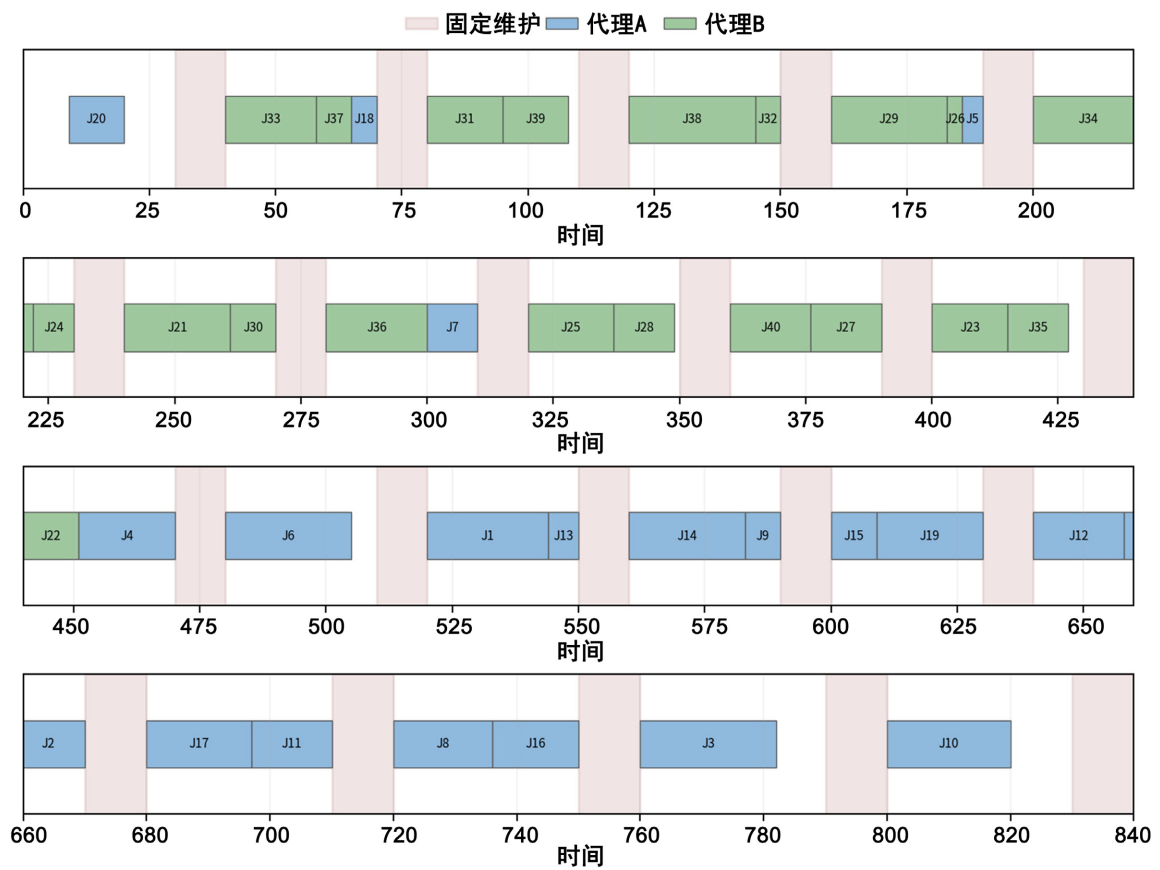


Figure 3. Best feasible schedule for the recorded 40-job instance  
图 3. 既有 40 工件固定实例的最好可行调度

## 5. 结论

本文研究了带释放时间和固定周期维护的单机双代理调度问题, 以最小化两个代理最大完工时间的加权和为目标, 建立了相应的混合整数规划模型, 并设计了针对该问题的分支定界算法。在算法设计方面, 构造了多策略上界和结合释放时间、固定维护及不可中断加工特征的下界, 并引入下界剪枝、状态支配和同代理插入支配规则, 以减少无效搜索节点和重复状态。数值实验表明, B&B 在多数算例中能够获得与 CPLEX 相同或更优的限时结果, 并在中等及较大规模问题上表现出更明显的求解优势。本文构造的初始上下界平均 Gap 约为 2.77%, 状态支配总体命中率达到 44.15%, 说明所提出的界限与剪枝机制能够有效提高搜索效率。本文所提出算法能够有效求解带释放时间和固定维护约束的单机双代理调度问题。未来可进一步研究更紧致的下界、高效的状态存储与并行搜索方法, 并将模型扩展至多代理、多机器及动态生产环境。

## 参考文献

- [1] Lenstra, J.K., Rinnooy Kan, A.H.G. and Brucker, P. (1977) Complexity of Machine Scheduling Problems. *Annals of Discrete Mathematics*, **1**, 343-362. [https://doi.org/10.1016/s0167-5060\(08\)70743-x](https://doi.org/10.1016/s0167-5060(08)70743-x)
- [2] Yin, Y., Cheng, S., Cheng, T.C.E., Wu, C. and Wu, W. (2012) Two-Agent Single-Machine Scheduling with Assignable Due Dates. *Applied Mathematics and Computation*, **219**, 1674-1685. <https://doi.org/10.1016/j.amc.2012.08.008>
- [3] Liu, P., Yi, N., Zhou, X. and Gong, H. (2013) Scheduling Two Agents with Sum-of-Processing-Times-Based Deterioration on a Single Machine. *Applied Mathematics and Computation*, **219**, 8848-8855. <https://doi.org/10.1016/j.amc.2013.03.040>
- [4] He, C. and Leung, J.Y.T. (2016) Two-Agent Scheduling of Time-Dependent Jobs. *Journal of Combinatorial Optimization*, **34**, 362-377. <https://doi.org/10.1007/s10878-016-9994-y>
- [5] Zhang, X.G. and Wang, Y. (2016) Two-Agent Scheduling Problems on a Single-Machine to Minimize the Total Weighted Late Work. *Journal of Combinatorial Optimization*, **33**, 945-955. <https://doi.org/10.1007/s10878-016-0017-9>
- [6] Yu, J., Liu, P. and Lu, X. (2023) Two-Agent Single-Machine Scheduling with Release Dates to Minimize the Makespan. *Optimization Letters*, **17**, 1915-1937. <https://doi.org/10.1007/s11590-022-01967-6>
- [7] Dyer, M.E. and Wolsey, L.A. (1990) Formulating the Single Machine Sequencing Problem with Release Dates as a Mixed Integer Program. *Discrete Applied Mathematics*, **26**, 255-270. [https://doi.org/10.1016/0166-218x\(90\)90104-k](https://doi.org/10.1016/0166-218x(90)90104-k)
- [8] Hsu, C., Low, C. and Su, C. (2010) A Single-Machine Scheduling Problem with Maintenance Activities to Minimize Makespan. *Applied Mathematics and Computation*, **215**, 3929-3935. <https://doi.org/10.1016/j.amc.2009.11.040>
- [9] Zhao, C. and Tang, H. (2010) Single Machine Scheduling with General Job-Dependent Aging Effect and Maintenance Activities to Minimize Makespan. *Applied Mathematical Modelling*, **34**, 837-841. <https://doi.org/10.1016/j.apm.2009.07.002>
- [10] Hfaiedh, W., Sadfi, C., Kacem, I. and Hadj-Alouane, A. (2015) A Branch-and-Bound Method for the Single-Machine Scheduling Problem under a Non-Availability Constraint for Maximum Delivery Time Minimization. *Applied Mathematics and Computation*, **252**, 496-502. <https://doi.org/10.1016/j.amc.2014.11.103>
- [11] Lee, W., Chung, Y. and Huang, Z. (2013) A Single-Machine Bi-Criterion Scheduling Problem with Two Agents. *Applied Mathematics and Computation*, **219**, 10831-10841. <https://doi.org/10.1016/j.amc.2013.05.025>
- [12] Luo, X. and Chu, F. (2006) A Branch and Bound Algorithm of the Single Machine Schedule with Sequence Dependent Setup Times for Minimizing Total Tardiness. *Applied Mathematics and Computation*, **183**, 575-588. <https://doi.org/10.1016/j.amc.2006.05.127>