中国科技期刊卓越行动计划项目入选期刊

控制与决策

CONTROL AND DECISION

启发式列生成算法求解带恶化效应的同构并行机调度问题

孙鑫伟, 钱斌, 胡蓉, 张森, 于乃康

引用本文:
孙鑫伟,钱斌,胡蓉,张森,于乃康. 启发式列生成算法求解带恶化效应的同构并行机调度问题[J]. 控制与决策, 2024, 39(5): 1636−1644.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2022.1615

启发式列生成算法求解带恶化效应的同构并行机调度问题

孙鑫伟[1], 钱 斌[1,2†], 胡 蓉[1,2], 张 森[1], 于乃康[1]

(1. 昆明理工大学 信息工程与自动化学院，昆明 650500；
2. 昆明理工大学 云南省人工智能重点实验室，昆明 650500)

摘 要: 针对实际生产中广泛存在的一类带恶化效应的同构并行机调度问题,以最小化最大完工时间为优化目标,构建该问题的整数规划模型,并提出一种启发式列生成算法 (HCGA) 进行求解. 在 HCGA 中,首先,利用 Dantzig-Wolfe 分解方法,将原问题分解为一个主问题 (MP) 和多个子问题;然后,设计启发式算法获得初始列,其中每列为一台机器上的一个调度方案,基于初始列构建限制主问题 (RMP) 模型;接着,设计快速有效的动态规划算法求解子问题,以得到需添加至 RMP 的列集,同时,考虑传统列生成算法收敛速度较慢,设计一系列方法来加速列生成过程;最后,基于所获取的 MP 线性松弛解,设计深潜启发式算法确定原问题的整数解. HCGA 与商用求解器 GUROBI 的对比实验结果表明,HCGA 可在较短时间内获得更优的解.

关键词: 并行机；恶化效应；最大完工时间；列生成；动态规划；深潜启发式

中图分类号: TP273 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2022.1615

引用格式: 孙鑫伟, 钱斌, 胡蓉, 等. 启发式列生成算法求解带恶化效应的同构并行机调度问题 [J]. 控制与决策, 2024, 39(5): 1636-1644.

Heuristic column generation algorithm for identical parallel machine scheduling problem with deterioration effect

SUN Xin-wei[1], *QIAN Bin*[1,2†], *HU Rong*[1,2], *ZHANG Sen*[1], *YU Nai-kang*[1]

(1. Faculty of Information Engineering and Automation，Kunming University of Science and Technology，Kunming 650500，China；2. Yunnan Key Laboratory of Artificial Intelligence，Kunming University of Science and Technology，Kunming 650500，China)

Abstract: An integer programming model is built to minimize the maximum completion time and a heuristic column generation algorithm (HCGA) is proposed for a class of identical parallel machine scheduling problem with deterioration effect that widely exist in practical production. In the HCGA, first, the original problem is decomposed into a master problem (MP) and multiple subproblems by using the Dantzig-Wolfe decomposition method. Secondly, a heuristic algorithm is designed to obtain initial columns, where each column represents a schedule on one machine. Based on the initial columns, a restricted MP (RMP) model is constructed. Then, a fast and effective dynamic programming algorithm is designed to solve each subproblem to obtain the column set to be added to the RMP. At the same time, considering that the convergence speed of traditional column generation algorithms is slow, a series of methods are designed to accelerate the column generation process. Finally, based on the obtained linear relaxation solution of the MP, a diving heuristic algorithm is designed to determine the integer solution of the original problem. According to the comparative experimental results of the HCGA and commercial solver GUROBI, the HCGA can obtain better solutions in a shorter time.

Keywords: parallel machine；deterioration effect；maximum completion time；column generation；dynamic programming；diving heuristic

0 引 言

目前,中国正处于由制造大国向制造强国转型的关键时期,而智能制造将是完成这一转型的主攻方向,高效的调度优化算法则是实现智能制造的重要手段. 并行机调度问题 (parallel machine scheduling problem, PMSP) 作为制造业中非常典型的一类调度

收稿日期: 2022-09-12；**录用日期:** 2023-02-18.
基金项目: 国家自然科学基金项目 (62173169, 61963022)；云南省基础研究重点项目 (202201AS070030).
责任编委: 刘民.
†通讯作者. E-mail: bin.qian@vip.163.com.

问题[1],一直是智能制造方向的重点研究领域.

在经典的生产调度问题中,通常假设工件的加工时间固定不变,如秦浩翔等[2]针对带阻塞约束的混合流水车间调度问题,以最小化makespan为目标,提出了一种双层变异迭代贪婪算法进行求解;轩华等[3]针对含不相关并行机的分布式柔性流水线调度问题,以最小化总加权提前/拖期惩罚为目标,提出了一种混合离散人工蜂群算法进行求解. 然而,在实际生产过程中,工件的加工时间往往会受到设备磨损、工件特性变化等因素的影响,导致工件实际加工时间会比预先设定的时间要长,这种现象称为恶化效应. 工件具有恶化效应的概念由Gupta等[4]提出,他们假设工件实际加工时间是其开始加工时间的线性递增函数,研究以最小化makespan为目标的单机调度问题,提出了两种启发式算法进行求解. 随后,学者们针对各种恶化效应展开了广泛研究. 在单机调度问题方面,Browne等[5]在Gupta等[4]的基础上,提出了该问题的最优调度方案;Zhao等[6]假设工件实际加工时间是其开始加工时间的线性递增函数,以最小化延迟工件的加权数量为目标,提出了一种伪多项式动态规划算法和一种完全多项式时间近似方案;Sun等[7]研究具有线性恶化效应和机器维护活动的单机调度问题,分别以最小化makespan和工件完工时间总和为目标,验证了该问题在多项式时间内可解,并给出了求解算法. 在PMSP方面,Guo等[8]假设工件具有阶梯恶化效应和加工顺序相关的设置时间,以最小化总拖期为目标,提出了一种混合离散布谷鸟搜索算法;Ding等[9]假设工件具有序列相关恶化效应,分别以最小化makespan和总加权完工时间为目标,提出了一种弹射链算法进行求解;Mir等[10]假设工件具有序列相关恶化效应,以最小化工件总完工时间为目标,建立了该问题的混合整数规划模型,并设计了启发式算法、遗传算法和蚁群算法进行求解.

本文研究带恶化效应的同构并行机调度问题(identical PMSP with deterioration effect, IPMSP-DE),其中恶化效应表现为工件实际加工时间是其开始加工时间的线性递增函数. 目前,已有部分学者针对带此种恶化效应的PMSP进行了研究. Kang等[11]以最小化makespan为目标,给出了该问题的一个完全多项式时间近似方案;Huang等[12]分别以最小化各工件间完工时间的总绝对差值和等待时间的总绝对差值为目标,验证了该问题在多项式时间内可解;Tigane等[13]以最小化makespan和总能耗为目标,提出了一种非支配排序遗传算法. 由文献调研可知,鲜有列生成算法求解带恶化效应的PMSP的研究.

列生成算法由Dantzig等[14]提出,用于求解大规模线性规划问题. 后来,列生成算法在组合优化领域得到了广泛研究,且已成功应用于下料问题[15]、车辆路径问题[16]、生产调度问题[17-20]等领域. 列生成算法获得的MP线性松弛解未必是整数解,而构造整数解的方法主要有两种:一种是分支定价法[17,20],另一种是将启发式算法与列生成算法相结合[19]. 分支定价法受制于子问题的求解效率[19],导致其虽然精度高,但是求解时间非常长,而将启发式算法与列生成算法相结合,能够在合理的时间内得到较好的解. 综合考虑,本文将深潜启发式算法(diving heuristic)与列生成算法相结合,快速构造整数近似最优解. 具体而言,本文以最小化makespan为目标,构建IPMSP-DE的整数规划模型,并提出一种HCGA进行求解. 首先,利用Dantzig-Wolfe分解方法将原问题分解为一个MP和多个子问题. 然后,采用启发式算法快速获得初始列,并基于初始列建立初始RMP模型. 接着,在子问题的求解方面,现有动态规划算法(如文献[17, 19-20])在求解工件加工时间为实数的单机子问题时时间复杂度非常高,针对此限制,本文设计快速有效的动态规划算法求解各子问题,并将检验数小于0的多个列添加至RMP. 传统列生成算法存在收敛速度缓慢的缺点,因此,本文设计一系列方法加速列生成算法. 最后,针对列生成算法产生的线性松弛解,与深潜启发式算法相结合得到整数解. 基于上述算法设计,所提出HCGA能够获得比商业求解器GUROBI更优的解,且求解时间更短.

1　IPMSP-DE问题描述及数学模型

1.1　IPMSP-DE介绍

以最小化makespan为目标的IPMSP-DE可描述如下:n个独立工件$N=\{1,2,\dots,n\}$在m台同构并行机$M=\{1,2,\dots,m\}$上加工处理. 每个工件$j\in N$必须选择唯一一台机器$i\in M$进行加工. 工件加工过程不允许中断. 每台机器同一时刻只能加工1个工件. 工件j在任意一台机器上的实际加工时间为$b_j+s_jv_j$. 其中:b_j为工件j的基础加工时间,s_j为工件j的开始加工时间,v_j为工件j的恶化系数. 问题的优化目标为最小化makespan.

1.2　整数规划模型建立

通过下式计算工件j的基础加工时间b_j与恶化系数v_j的比值rate_j:

$$\text{rate}_j=\frac{b_j}{v_j}. \tag{1}$$

Browne等[5]提出了基于“最小rate$_j$优先”(smallest rate$_j$ first, SRF)的调度策略,即在同一台机器上,rate$_j$越小的工件,其开始加工时间越早. 根据SRF调度策略得到$m=1$时以最小化makespan为目标的IPMSP-DE最优解. 因此,本文只考虑满足SRF顺序的调度. 本文将可行调度定义为满足SRF顺序的调度.

令$C_{\max}$为makespan,s_j为工件j的开始加工时间,C_j为工件j的完工时间. 若工件j在机器i上紧随工件k加工,则$x_{kj}^i=1$;否则,$x_{kj}^i=0$. 若工件j在机器i上第一个被加工,则$x_{0j}^i=1$;否则,$x_{0j}^i=0$. 若工件j在机器i上最后一个被加工,则$x_{j,n+1}^i=1$;否则,$x_{j,n+1}^i=0$.

IPMSP-DE整数规划模型(IP 1)如下式所示:

$$\min\ C_{\max}. \tag{2}$$

$$\text{s.t.}\sum_{i\in M}\sum_{k\in N\bigcup\{0\}|k\neq j}x_{kj}^i=1,\ \forall j\in N; \tag{3}$$

$$\sum_{j\in N}x_{0j}^i\leqslant 1,\ \forall i\in M; \tag{4}$$

$$\sum_{k\in N\bigcup\{0\}|k\neq j}x_{kj}^i=\sum_{k\in N\bigcup\{n+1\}|k\neq j}x_{jk}^i,\quad \forall i\in M,\forall j\in N; \tag{5}$$

$$(b_k/v_k)x_{kj}^i\leqslant(b_j/v_j)x_{kj}^i,\quad \forall i\in M,\forall j\in N,\forall k\in N,k\neq j; \tag{6}$$

$$s_j=\sum_{i\in M}\sum_{k\in N\bigcup\{0\}|k\neq j}C_kx_{kj}^i,\ \forall j\in N; \tag{7}$$

$$C_0=0; \tag{8}$$

$$C_j=b_j+s_j(1+v_j),\ \forall j\in N; \tag{9}$$

$$C_{\max}\geqslant C_j,\ \forall j\in N; \tag{10}$$

$$x_{kj}^i\in\{0,1\},\ \forall i\in M,\forall j\in N\bigcup\{n+1\},\quad \forall k\in N\bigcup\{0\}. \tag{11}$$

其中:目标函数(2)为最小化makespan,约束(3)保证每个工件只能被加工1次,约束(4)保证每台机器最多使用1次,约束(5)为保证工件正确排序的流守恒约束,约束(6)保证每台机器的调度符合SRF调度策略,约束(7)表示工件j的开始加工时间等于工件j紧前工件的完工时间,约束(8)为约束(7)和(9)提供初始值C_0,约束(9)定义了工件j的完工时间,约束(10)表示最大完工时间$C_{\max}$不小于任意工件的完工时间,约束(11)定义了决策变量的取值范围.

2 Dantzig-Wolfe分解

IP 1包含大量的变量,随着问题规模的不断增大,商业求解器求解能力将显著降低,因此,本节使用Dantzig-Wolfe分解方法将原问题转化为一个MP和m个子问题,然后采用列生成算法进一步求解.

2.1 主问题模型

令S^i为机器i上所有可行调度的集合,c_i^s为机器i上可行调度s的makespan. 若可行调度$s\in S^i$中包含工件j,则$a_{js}^i=1$;否则,$a_{js}^i=0$. 若机器i上的可行调度选为s,则$y_i^s=1$;否则,$y_i^s=0$.

针对上述参数和变量,得到如下MP:

$$\min\ C_{\max}. \tag{12}$$

$$\text{s.t.}\sum_{i\in M}\sum_{s\in S^i}a_{js}^iy_i^s=1,\ \forall j\in N; \tag{13}$$

$$\sum_{s\in S^i}y_i^s\leqslant 1,\ \forall i\in M; \tag{14}$$

$$\sum_{s\in S^i}c_i^sy_i^s\leqslant C_{\max},\ \forall i\in M; \tag{15}$$

$$y_i^s\in\{0,1\},\ \forall i\in M,\forall s\in S^i. \tag{16}$$

其中:目标函数(12)为最小化makespan,约束(13)保证每个工件必须分配给1台机器,约束(14)保证每台机器最多分配1个可行调度,约束(15)表示最大完工时间$C_{\max}$不小于任意机器的完工时间,约束(16)定义了决策变量y_i^s的取值范围.

随着问题规模的增大,每台机器中包含的可行调度数均呈指数增长,因此,列举出所有可行调度是非常困难的. 考虑到列生成算法只需要考虑部分可行调度,然后,通过求解子问题向RMP中添加检验数小于0的列,本文先对MP中的y_i^s线性松弛后得到线性主问题(linear master problem, LMP),通过启发式算法生成初始列,并基于初始列构建RMP,然后使用列生成算法和深潜启发式算法进一步求解.

2.2 子问题

每台机器均对应一个子问题. 令π_j、u_i和θ_i分别为约束(13)~约束(15)对应的对偶变量,r_i^s为机器i上可行调度s对应的RMP检验数,有

$$r_i^s=-\sum_{j\in N}a_{js}^i\pi_j-c_i^s\theta_i-u_i,\ \forall i\in M. \tag{17}$$

求解子问题的目的是寻找检验数小于0的列,并将其添加至RMP. 若找不到检验数小于0的列,则停止列生成.

3 算法设计

3.1 初始列的生成

生成RMP的初始列包括以下步骤.

step 1: 将工件按照rate$_j$非减顺序进行排列,并根据下式重新给工件定号,在这一设定下,序列$(1,2,\ldots,n)$即为SRF顺序,然后转至step 2,有

$$\text{rate}_1\leqslant\text{rate}_2\leqslant\ldots\leqslant\text{rate}_n. \tag{18}$$

step 2: 将工件按照SRF顺序依次随机地安排在机器上来产生一个RMP的可行解,越早空闲的机器被安排工件的可能性越大. 运行该算法λ次,保留其中最好的α个可行解,然后转至step 3.

step 3: 针对step 2产生的每个可行解,对其对应的每个工件j顺序执行如下操作: 1) 将工件j按照SRF排序策略依次插入至其他机器,选取产生的所有可行调度中的最优调度与原调度进行对比,若优于原调度则替换,否则保留原调度. 2) 将工件j与其他机器上的每个工件逐个进行交换,并将交换工件的两台机器上的工件序列按照SRF排序策略进行排序,选取产生的所有可行调度中的最优调度与原调度进行对比,若优于原调度则替换,否则保留原调度.

3.2　上　界

由于同构并行机不需要区分机器的不同,本节简化了式(13)～(16),由此构建如下整数规划模型(IP 2),通过求解后文所述的IP 2获得最优目标函数值上界$\overline{C}$. HCGA求得的近似最优调度方案中必定不包含makespan大于$\overline{C}$的列,为了加速HCGA,求解子问题时,只考虑完工时间小于$\overline{C}$的列.

令S为将RMP的列集去重后得到的列集. 其中: 去重表示列生成每次迭代求解m个子问题后,添加至RMP中的相同列只选择1列加入S. $a_j^{s'}$表示调度$s'\in S$中是否包含工件j,是则为1;否则为0. $C^{s'}$为可行调度s'的makespan. $y^{s'}$表示某台机器上的可行调度是否选为s',是则为1;否则为0.

针对上述参数和变量,IP 2可定义为

$$\min\ C_{\max}. \tag{19}$$

$$\text{s.t.}\ \sum_{s'\in S} a_j^{s'} y^{s'} = 1,\ \forall j\in N; \tag{20}$$

$$\sum_{s'\in S} y^{s'} \leqslant m; \tag{21}$$

$$C^{s'} y^{s'} \leqslant C_{\max},\ \forall s'\in S; \tag{22}$$

$$0\leqslant y^{s'} \leqslant 1,\ \forall s'\in S. \tag{23}$$

其中: 目标函数(19)为最小化makespan,约束(20)表示每个工件必须分配给1台机器,约束(21)表示最多有m台机器可用,约束(22)表示最大完工时间$C_{\max}$不小于任何一台机器的完工时间,约束(23)定义了决策变量$y^{s'}$的取值范围.

令$\overline{C}$为工件makespan的上界,min_solve为当前最优整数解. 令$\overline{C}=+\infty$,第1次运行初始化程序后和每次列生成结束后,对IP 2模型进行求解,得到目标函数值obj和其对应的调度min_solve_local. 若$\overline{C}>$obj,则令

$$\overline{C} = \text{obj}; \tag{24}$$

$$\text{min\_solve} = \text{min\_solve\_local}. \tag{25}$$

求解子问题时,只考虑完工时间小于$\overline{C}$的列. 为了减少IP 2的计算时间,每求解一次IP 2,删除IP 2中完工时间大于$\overline{C}$的列.

3.3　动态规划算法

首先由式(18)重新给工件定号,此时,序列$(1,2,\ldots,n)$即为SRF顺序. 基于这一设定,本文设计了3种动态规划算法求解子问题.

3.3.1　第1种动态规划算法

令$C(j)$为工件集合$\{1,2,\ldots,j\}$所有子集对应的可行调度完工时间的集合. 令$F(j,t)(\forall t\in C(j))$表示在0时刻开始加工,在$t$时刻完工的可行调度的最小目标函数值,其中该可行调度由工件集合$\{1,2,\ldots,j\}$的任意子集组成. 基于上述定义,第1种动态规划算法具体步骤如下.

step 1: 初始值定义为

$$F(0,0)=0,\ C(0)=\{0\}. \tag{26}$$

step 2: 求解$F(j,t)$. 算法1给出了$F(j,t)$求解的具体流程.

算法1　动态规划算法1.

输入: 机器编号i,工件基础加工时长$b_j(\forall j\in N)$,恶化系数$v_j(\forall j\in N)$;

输出: $F(j,t),\forall j\in N, t\in C(j)$.

```
1. 由式(26)设定初始值;
2. for j=1,2,...,n do
3.     C(j) ← C(j−1);
4.     for t∈C(j−1) do
5.         F(j,t) ← F(j−1,t);
6.     end for
7.     for t∈C(j−1) do
8.         if t+b_j+v_j t<C̄ then
9.             if t+b_j+v_j t∈C(j) then
10.                F(j,t+b_j+v_j t) ← min{F(j−1,t)−
                   π_j−(b_j+v_j t)θ_i, F(j,t+b_j+v_j t)};
11.            else
12.                C(j) ← C(j)⋃{t+b_j+v_j t};
13.                F(j,t+b_j+v_j t) ← F(j−1,t)−
                   π_j−(b_j+v_j t)θ_i;
14.            end if
15.        end if
16.    end for
17. end for
```

step 3: 计算最优目标函数值$F_{\min}$及其对应的完工时间$t_{\min}$,即

$$F_{\min}=\min_{t\in C(n)}F(n,t), \tag{27}$$

$$t_{\min}=\arg\min_{t\in C(n)}F(n,t). \tag{28}$$

step 4: 由$F(n,t_{\min})$反向推演动态规划过程计算出最优可行调度.

3.3.2 第2种动态规划算法

令$C(j)$为工件集合$\{1,2,\ldots,j\}$部分子集对应的调度完工时间的集合. 令$\lceil t\rceil$表示将t向上取整,如$\lceil 3.14\rceil=4$. 令$F(j,\lceil t\rceil)(\forall t\in C(j))$表示在0时刻开始加工,在时间区间$(\lceil t\rceil-1,\lceil t\rceil]$内完工的可行调度的近似最小目标函数值,其中该可行调度由工件集合$\{1,2,\ldots,j\}$的任意子集组成. 基于上述定义,第2种动态规划算法具体步骤如下.

step 1: 初始值定义为

$$\begin{aligned}&F(0,0)=0,\\&C(0)=\{0\}.\end{aligned} \tag{29}$$

step 2: 求解$F(j,\lceil t\rceil)$. 算法2给出了$F(j,\lceil t\rceil)$求解的具体流程.

算法2 动态规划算法2.

输入: 机器编号i,工件基础加工时长$b_j(\forall j\in N)$,恶化系数$v_j(\forall j\in N)$;

输出: $F(j,t),\forall j\in N,t\in C(j)$.

1. 由式(29)设定初始值;
2. for $j=1,2,\ldots,n$ do
3. $C(j)\leftarrow C(j-1)$;
4. for $t\in C(j-1)$ do
5. $F(j,\lceil t\rceil)\leftarrow F(j-1,\lceil t\rceil)$;
6. end for
7. for $t\in C(j-1)$ do
8. if $t+b_j+v_jt<\overline{C}$ then
9. if $\lceil t+b_j+v_jt\rceil$与$C(j)$中某元素向上取整后相等 then
10. $F(j,\lceil t+b_j+v_jt\rceil)\leftarrow\min\{F(j-1,\lceil t\rceil)-\pi_j-(b_j+v_jt)\theta_i,F(j,\lceil t+b_j+v_jt\rceil)\}$;
11. if $F(j,\lceil t+b_j+v_jt\rceil)==F(j-1,\lceil t\rceil)-\pi_j-(b_j+v_jt)\theta_i$ then
12. 删除$C(j)$中区间$(\lceil t+b_j+v_jt\rceil-1,\lceil t+b_j+v_jt\rceil]$内的完工时间,并令$C(j)\leftarrow C(j)\bigcup\{t+b_j+v_jt\}$;
13. end if
14. else
15. $C(j)\leftarrow C(j)\bigcup\{t+b_j+v_jt\}$;
16. $F(j,\lceil t+b_j+v_jt\rceil)\leftarrow F(j-1,\lceil t\rceil)-\pi_j-(b_j+v_jt)\theta_i$;
17. end if
18. end if
19. end for
20. end for

step 3: 计算近似最优目标函数值$F_{\min}$及其对应的完成时间$t_{\min}$,即

$$F_{\min}=\min_{t\in C(n)}F(n,\lceil t\rceil), \tag{30}$$

$$t_{\min}=\arg\min_{t\in C(n)}F(n,\lceil t\rceil). \tag{31}$$

step 4: 由$F(n,t_{\min})$反向推演动态规划过程计算出近似最优可行调度.

3.3.3 第3种动态规划算法

令工件实际加工时间向下保留ε位小数,然后采用第1种动态规划算法求解.

3.3.4 3种动态规划算法优劣分析

采用列生成算法求解的并行机调度方面的文献中,大多采用类似算法1的动态规划算法求解子问题(如文献[18, 20-21]),但是,算法1并不适合求解本文问题. 由于工件加工时间可取$(0,\overline{C}]$内的任何小数,算法1的$F(j,t)$中t最多具有2^j种不同的取值. 第2种动态规划算法的$F(j,t)$中t只能取区间$[0,\lceil\overline{C}\rceil]$内的整数,故$F(j,t)$中$t$最多具有$\min(2^j,\lceil\overline{C}\rceil+1)$种不同的取值. 第3种动态规划算法中$F(j,t)$中$t$只能在步长为$10^{-\varepsilon}$的区间$[0,\lceil\overline{C}\rceil]$内取值,故$F(j,t)$中$t$最多具有$\min(2^j,\lceil\overline{C}\rceil\times10^{\varepsilon}+1)$种不同的取值. 因此,相比于第1种动态规划算法,第2种和第3种动态规划算法均会大大缩短计算时长. 第3种动态规划算法可通过减小ε值来缩短计算时间,但是,这会使得列生成算法结束后得到的RMP的目标函数值与LMP的最优目标函数值差距增大,进而使其使用深潜启发式算法构造整数解的价值变低,而第2种动态规划算法不会改变每个可行调度的完工时间,且耗时最短. 因此,本文在HCGA中采用第2种动态规划算法求解子问题.

3.3.5 下 界

第2种动态规划算法求解出的为子问题近似最优解,因此,列生成阶段只能将部分检验数小于0的列添加至RMP,导致列生成算法第1次结束时获得的为LMP的近似最优目标函数值. 针对这种情况,为了评价HCGA得到的整数解的优劣,本文设计一种列生成算法(CG_LB)以获取LMP最优目标函数值的下界,即MP最优目标函数值的下界. 与HCGA中的列生成算法不同的是,CG_LB中工件加工时间向下保留ε位小数,其子问题通过第3种动态规划算法(见第3.3.3节)求解.

3.3.6 加速列生成的策略

1) 每求解一次子问题向RMP中添加$\theta(\theta \geqslant 1)$个检验数最小且小于0的列,而不是只添加一列检验数最小的列. 若检验数小于0的列不足θ列,则将其全部加入RMP. 这种策略通常会减少列生成迭代的次数[21].

2) 仅对部分子问题使用动态规划算法求解,具体如下: 依次对各机器的子问题使用动态规划算法求解. 检测产生的列是否能够使得未使用动态规划算法求解的机器对应的子问题检验数小于0,若能,则将其加入该机器的备选列集,且后续不对该机器的子问题使用动态规划算法求解. 上述步骤完成后,从未使用过动态规划算法求解子问题的机器的备选列集中选择要加入RMP的列集. 子问题的求解占据算法大部分求解时间,该策略可在极短的时间内为机器找出检验数小于0的列.

3.4 深潜启发式算法

令固定某台机器表示将该机器上其中一列的决策变量值赋值为1,固定某列表示将该列的决策变量值赋值为1. 列生成算法结束后,若得到的MP线性松弛解为非整数解,则从未固定的机器中选择决策变量最接近1的列所对应的机器i^{η}. 从机器i^{η}上选择决策变量取值在$(\delta, 1]$区间最接近1的n_c列,若决策变量取值在$(\delta, 1]$区间列的数量小于n_c,则将其全部选中. 对n_c列中每列分别计算将其固定且运行列生成算法后得到的RMP目标函数值,选择目标函数值最小的列进行固定,然后运行列生成算法. 重复上述步骤,直至未固定的机器不存在决策变量在$(\delta, 1]$区间的列或所有的机器均已固定为止,此时,min_solve和$\overline{C}$分别为HCGA得到的最终解和最终目标函数值. 深潜启发式算法固定列后可能会导致RMP不可行,因此,每固定一列,便采用初始化算法生成初始列,并添加至RMP.

4 实验设计与分析

4.1 实验设置

本文通过随机生成的实例进行算法和模型验证. 所提出算法采用C++编程实现,并在AMD R7-4 800H 2.90 GHz处理器和16.0 GB内存的个人计算机上运行. 线性规划和整数规划部分采用商业求解器GUROBI 9.5.1求解.

本文按照如下方式随机生成问题实例: 工件数量$n = \{10, 20, 30, 40, 50, 60, 70, 80\}$,机器数量$m = \{2, 4, 6, 8, 10, 12\}$,工件基础加工时间$b_j$为区间$[1, 100]$均匀分布的随机整数,工件的恶化系数$v_j$在步长为0.01的区间$[0, 0.4]$上随机产生.

基于初步参数调整实验,本文在HCGA中设置如下参数: 令$\lambda = n, \varepsilon = 1, \theta = 3, \delta = 0.15, n_c = 4$. 在第1次运行初始化算法时,令$\alpha = 10$;在深潜启发式算法中运行初始化算法时,由于RMP已具备足够多的列,不需要通过初始化算法添加太多的列,此时,令$\alpha = 2$.

4.2 列生成加速策略效果分析

为了验证第3.3.6节列生成加速策略的作用,本节实现4种列生成算法以求解LMP,每种列生成算法除加速策略,其余部分与HCGA中的第1次列生成算法结束前的步骤相同. 第1种列生成算法(CG 1)不使用第3.3.6节加速策略,此时,列生成算法结束时运行时间为t_1;第2种列生成算法(CG 2)只使用第3.3.6节第1种加速策略,此时,列生成算法结束时运行时间为t_2;第3种列生成算法(CG 3)只使用第3.3.6节第2种加速策略,此时,列生成算法结束时运行时间为t_3;第4种列生成算法(CG 4)使用第3.3.6节全部加速策略,此时,列生成算法结束时运行时间为t_4. 本节对每个不同规模的问题均随机生成1个实例进行实验,各算法在每个实例上均运行10次. 令$t_{lr}(\forall l = 2, 3, 4, \forall r = 1, 2, \ldots, 10)$为第$l$种列生成算法第$r$次求解某实例的运行时间,$t_{1r}$为第1种列生成算法第$r$次求解某实例的运行时间,$\text{bst}_l$、$\text{wst}_l$、$\text{avg}_l$分别为第$l$种列生成算法相对于CG 1在运行时间上提升率的最大值、最小值和平均值,如下式所示:

$$\text{bst}_l = \max_{r=1,2,\ldots,10} \frac{t_{1r} - t_{lr}}{t_{1r}}, \tag{32}$$

$$\text{wst}_l = \min_{r=1,2,\ldots,10} \frac{t_{1r} - t_{lr}}{t_{1r}}, \tag{33}$$

$$\text{avg}_l = \sum_{r=1,2,\ldots,10} \frac{t_{1r} - t_{lr}}{10t_{1r}}. \tag{34}$$

为了避免不同初始列对算法效率带来的影响,各算法在第r次求解同一实例时采用相同的初始列. 表1为4种列生成算法运行时间的统计结果,时间单位为s.

为了进一步验证所提出加速策略在统计意义下的性能,对表1中4种列生成算法独立运行10次的平均运行时间进行方差分析(ANOVA). 图1为4种列生成算法的平均运行时间在95%置信度下Tukey's HSD检验的置信区间.

由表1和图1可见,所提出加速策略具有明显加速作用,且同时使用两种加速策略的效果更优. 此外,加速策略对于大规模问题更为有效. 这是因为第3.3.6节的两种加速策略均是针对子问题设计的,而小规模问题需求解的子问题数量少,子问题总求解时间占列生成算法总时间的比例较小,故加速效果不明

表 1 列生成加速策略效果分析

规模	t_1/s	CG 2				CG 3				CG 4			
		t_2/s	bst_2	wst_2	avg_2	t_3/s	bst_3	wst_3	avg_3	t_4/s	bst_4	wst_4	avg_4
10 × 2	0.59	0.44	0.32	0.12	0.23	0.5	0.22	0.07	0.15	**0.4**	**0.38**	**0.17**	**0.31**
10 × 4	0.32	0.28	0.24	0.05	0.13	0.28	0.28	0.05	0.13	**0.25**	**0.32**	**0.11**	**0.22**
20 × 4	2.46	1.4	0.47	0.34	0.42	1.56	0.53	0.25	0.37	**1.05**	**0.65**	**0.49**	**0.57**
20 × 6	1.36	0.98	0.37	0.2	0.28	1	0.36	0.18	0.26	**0.81**	**0.47**	**0.32**	**0.4**
30 × 4	17.47	9.05	0.66	0.29	0.47	7.22	0.67	0.43	0.58	**4.17**	**0.82**	**0.71**	**0.76**
30 × 6	5.9	3.39	0.53	0.34	0.42	2.58	0.64	0.45	0.56	**1.71**	**0.76**	**0.66**	**0.71**
30 × 8	3.75	2.24	0.5	0.25	0.4	2.16	0.55	0.28	0.42	**1.41**	**0.68**	**0.53**	**0.62**
40 × 4	61.16	30.72	0.7	0.34	0.48	21.96	0.78	0.49	0.63	**13.42**	**0.86**	**0.7**	**0.77**
40 × 6	27.99	13.93	0.58	0.38	0.5	8.74	0.74	0.62	0.68	**5.22**	**0.85**	**0.76**	**0.81**
40 × 8	18.77	8.98	0.58	0.43	0.52	6.97	0.69	0.51	0.63	**4.18**	**0.82**	**0.67**	**0.77**
50 × 6	69.52	34.14	0.57	0.39	0.5	22.42	0.75	0.55	0.67	**13.85**	**0.85**	**0.75**	**0.8**
50 × 8	37.41	20.06	0.55	0.34	0.46	12.7	0.71	0.61	0.66	**8.83**	**0.8**	**0.7**	**0.76**
50 × 10	32.14	16.97	0.6	0.37	0.47	12.56	0.71	0.52	0.61	**7.82**	**0.81**	**0.7**	**0.75**
60 × 6	133.31	76.07	0.47	0.3	0.43	43.86	0.71	0.57	0.67	**24.45**	**0.84**	**0.79**	**0.82**
60 × 8	123.23	60.91	0.62	0.4	0.5	36.86	0.81	0.62	0.69	**20.92**	**0.86**	**0.79**	**0.83**
60 × 10	106.06	52.93	0.56	0.45	0.5	33.29	0.73	0.61	0.68	**19.52**	**0.86**	**0.77**	**0.81**
70 × 8	148.62	77.03	0.57	0.33	0.48	48.73	0.7	0.62	0.67	**25.68**	**0.85**	**0.8**	**0.83**
70 × 10	102.11	52.54	0.6	0.41	0.48	38	0.68	0.54	0.63	**19.75**	**0.83**	**0.79**	**0.81**
70 × 12	102.82	54.83	0.55	0.39	0.47	42.81	0.64	0.51	0.58	**22.25**	**0.81**	**0.75**	**0.78**
80 × 8	295.67	143.64	0.55	0.46	0.51	87.42	0.72	0.66	0.7	**45.44**	**0.86**	**0.81**	**0.85**
80 × 10	286.14	131.32	0.62	0.41	0.54	78.93	0.76	0.66	0.72	**42.1**	**0.88**	**0.84**	**0.85**
80 × 12	225.03	101.17	0.6	0.48	0.55	65.16	0.74	0.67	0.71	**32.25**	**0.87**	**0.84**	**0.86**
平均值	81.9	40.59	0.54	0.34	0.44	26.17	0.64	0.48	0.56	**14.34**	**0.76**	**0.66**	**0.71**

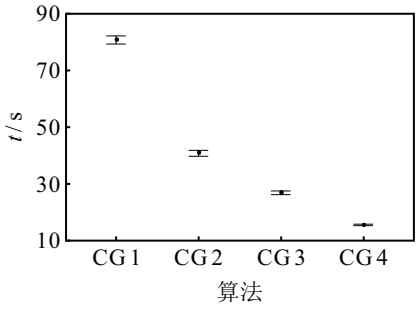

图 1 4种列生成算法置信区间对比

显. 随着问题规模的增大,被添加至RMP列的数量和每个子问题的求解时间均会大幅增加,这时加速策略的效果会明显增大.

4.3 算法对比

为了评价GUROBI和HCGA的性能,本节对GUROBI和HCGA每个规模的问题均随机生成1个实例进行实验,各算法在每个实例上均运行20次,实验结果取其平均值. GUROBI的求解时间上限为3 600 s.

首先,定义最优解的下界LB为

$$\mathrm{LB} = \max\{\mathrm{LB_{IP}}, \mathrm{LB_{DP_3}}\}. \tag{35}$$

其中: $\mathrm{LB_{IP}}$ 为商业求解器GUROBI求解IP 1时在给定时间内得到的下界, $\mathrm{LB_{DP_3}}$ 为第3.3.5节提出的CG_LB (每个实例只运行一次)求得的LMP最优目标函数值下界.

然后,定义如下指标: $\mathrm{BV_{IP}}$ 为GUROBI求解IP 1时在给定时间内求得的平均目标函数值; $\mathrm{Gap_{IP}}$ 为 $\mathrm{BV_{IP}}$ 与LB间的百分比差距; $\mathrm{LB_{DP_2}}$ 为HCGA第1次列生成结束后RMP的平均目标函数值; $\mathrm{BV_{HCGA}}$ 为HCGA求得的平均目标函数值; $\mathrm{Gap_{HCGA}}$ 为 $\mathrm{BV_{HCGA}}$ 与LB间的百分比差距; $\mathrm{Gap_{DP_{2\text{-}3}}}$ 为 $\mathrm{LB_{DP_2}}$ 与 $\mathrm{LB_{DP_3}}$ 间的百分比差距; T_{IP} 为GUROBI求解IP 1的平均求解时间,单位为s; T_{HCGA} 为HCGA的平均求解时间,单位为s. $\mathrm{Gap_{IP}}$、$\mathrm{Gap_{HCGA}}$ 以及 $\mathrm{Gap_{DP_{2\text{-}3}}}$ 的值分别由下式给出:

$$\mathrm{Gap_{IP}} = \frac{\mathrm{BV_{IP}} - \mathrm{LB}}{\mathrm{LB}} \times 100\,\%, \tag{36}$$

$$\mathrm{Gap_{HCGA}} = \frac{\mathrm{BV_{HCGA}} - \mathrm{LB}}{\mathrm{LB}} \times 100\,\%, \tag{37}$$

$$\mathrm{Gap_{DP_{2\text{-}3}}} = \frac{\mathrm{LB_{DP_2}} - \mathrm{LB_{DP_3}}}{\mathrm{LB_{DP_3}}} \times 100\,\%. \tag{38}$$

采用上述指标,GUROBI与HCGA的比较结果如表2所示. 由表2中的数据得到如下结论.

1) 当规模较小时,GUROBI在给定时间内能够获得较为不错的解,但是规模较大时,GUROBI求解能力显著降低,无法在给定时间获得高质量的解. HCGA在所有规模均能够在合理时间内得到高质量的解. GUROBI只有在20 × 6规模能够在给定时间内获得比HCGA更好的解,在其他规模得到的解质量均比HCGA得到的解质量差.

表 2　算法性能比较

问题规模	LB	LB_{DP_3}	GUROBI (IP 1)				HCGA				
			LB_{IP}	BV_{IP}	Gap_{IP} / %	T_{IP} / s	LB_{DP_2}	BV_{HCGA}	Gap_{HCGA} / %	$Gap_{DP_{2\text{-}3}}$ / %	T_{HCGA} / s
10 × 2	430.09	427.9	**430.09**	430.09	0	0.64	429.2	**430.09**	**0**	0.3	**0.58**
10 × 4	99.62	98.1	**99.62**	99.62	0	0.57	98.53	**99.62**	**0**	0.44	**0.55**
20 × 4	329.28	**329.28**	157.58	331.87	0.79	3 600	329.88	**330.96**	**0.51**	0.18	**2.83**
20 × 6	190.56	**190.56**	137.95	**193.38**	**1.48**	3 600	190.84	193.56	1.57	0.15	**3.44**
30 × 4	602.32	**602.32**	147.71	606.72	0.73	3 600	602.83	**604.88**	**0.43**	0.08	**9.1**
30 × 6	311.46	**311.46**	130.61	315.51	1.3	3 600	311.79	**314.13**	**0.86**	0.11	**8.11**
30 × 8	186.85	**186.85**	91	193.55	3.59	3 600	187.51	**190.51**	**1.96**	0.35	**8.35**
40 × 4	1 000.62	**1 000.62**	126.51	1 014.84	1.42	3 600	1 001.77	**1 003.8**	**0.32**	0.11	**25.39**
40 × 6	512.06	**512.06**	140.69	522.6	2.06	3 600	512.76	**515.29**	**0.63**	0.14	**24.01**
40 × 8	302.65	**302.65**	100	317.43	4.88	3 600	303	**305.83**	**1.05**	0.12	**21.46**
50 × 6	628.85	**628.85**	131.67	643.39	2.31	3 600	629.73	**633.07**	**0.67**	0.14	**28.81**
50 × 8	357.12	**357.12**	98	370.41	3.72	3 600	357.77	**360.29**	**0.89**	0.18	**38.97**
50 × 10	280.55	**280.55**	100	302.1	7.68	3 600	280.89	**283.8**	**1.16**	0.12	**51.1**
60 × 6	901.54	**901.54**	108.71	920.1	2.06	3 600	902.65	**904.44**	**0.32**	0.12	**53.61**
60 × 8	631.77	**631.77**	99	662.3	4.83	3 600	632.87	**636.32**	**0.72**	0.17	**92.39**
60 × 10	429.46	**429.46**	100	447.16	4.12	3 600	429.98	**433.81**	**1.01**	0.12	**89.87**
70 × 8	529.42	**529.42**	99	550.12	3.91	3 600	530.49	**532.93**	**0.66**	0.2	**134.13**
70 × 10	382.39	**382.39**	99	432.49	13.1	3 600	383.05	**386.16**	**0.99**	0.17	**138.36**
70 × 12	346.95	**346.95**	99	–	–	3 600	347.44	**350.74**	**1.09**	0.14	**137**
80 × 8	829.85	**829.85**	100	928.39	11.87	3 600	831.2	**834.01**	**0.5**	0.16	**157.84**
80 × 10	595.94	**595.94**	98	–	–	3 600	596.8	**600.87**	**0.83**	0.14	**100.22**
80 × 12	412.85	**412.85**	95	–	–	3 600	413.69	**417.37**	**1.09**	0.2	**163.77**

2) 所有规模的 $Gap_{DP_{2\text{-}3}}$ 均不超过 0.44 %, 表明 HCGA 中的列生成算法能够有效求解 LMP.

商业求解器 GUROBI 采用分支定界的通用搜索框架进行求解, 当问题规模增大时, IP 1 的决策变量增多, 会导致分支定界算法的搜索树规模急剧增大, 从而使其求解无效分支的时间大大增加, 因此, GUROBI 难以在限定时间内得到较好的解. 列生成算法每次仅需求解只含有少部分变量的 RMP, 可显著减小问题规模, 同时利用一系列加速方法来提升求解速度, 并通过只产生 makespan 小于 $\overline{C}$ 的列来提升松弛解中列的质量. 针对列生成算法产生的松弛解, 采用深潜启发式算法快速构造整数近优解. 因此, HCGA 可在较短时间内获取问题的较优解.

为了进一步验证 HCGA 与 GUROBI 间的统计性能差异, 对表 2 中两个算法的平均目标函数值 BV_{IP} 和 BV_{HCGA} (除 70 × 12、80 × 10 和 80 × 12 规模的实例) 进行方差分析 (ANOVA). 图 2 给出了 HCGA 与 GUROBI 在 95 % 置信度下 Tukey's HSD 检验的置信区间. 由图 2 可见, HCGA 在统计意义上明显优于 GUROBI.

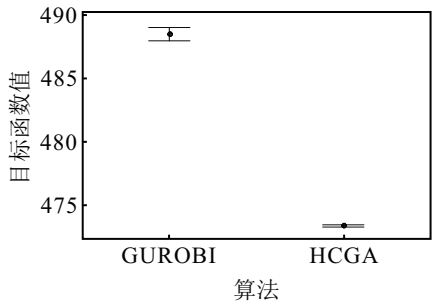

图 2　GUROBI 与 HCGA 置信区间对比

5　结　论

恶化效应广泛存在于实际生产中, 且目前针对本文问题的研究较少, 因此, 研究 IPMSP-DE 非常必要. 本文以最小化 makespan 为目标, 建立了 IPMSP-DE 的整数规划模型, 并提出了一种 HCGA 进行求解. 首先, 利用 Dantzig-Wolfe 分解方法将原问题分解为一个主问题和 m 个子问题, 并利用启发式算法获得初始列, 基于初始列构建 RMP; 然后, 设计了一种快速有效的动态规划算法求解各子问题, 以得到需添加至 RMP 的列集; 接着, 设计了一系列方法加速列生成算法; 最后, 与深潜启发式算法相结合确定原问题的整数解. 通过在不同测试规模问题上的仿真实验和算法对比, 验证了所提出 HCGA 具有比商业求解器 GUROBI 更优的求解性能, 可在更短时间内获得更优的解. 后续工作会将所设计的 HCGA 应用于其他带恶化效应的并行机调度问题.

参考文献 (References)

[1]　雷德明, 杨海. 求解多目标不相关并行机调度问题的多群体人工蜂群算法 [J]. 控制与决策, 2022, 37(5): 1174-1182.
(Lei D M, Yang H. Multi-colony artificial bee colony algorithm for multi-objective unrelated parallel machine scheduling problem[J]. Control and Decision, 2022, 37(5): 1174-1182.)

[2] 秦浩翔, 韩玉艳, 陈庆达, 等. 求解阻塞混合流水车间调度的双层变异迭代贪婪算法[J]. 控制与决策, 2022, 37(9): 2323-2332.
(Qin H X, Han Y Y, Chen Q D, et al. A double level mutation iterated greedy algorithm for blocking hybrid flow shop scheduling[J]. Control and Decision, 2022, 37(9): 2323-2332.)

[3] 轩华, 李文婷, 李冰. 混合离散人工蜂群算法求解含不相关并行机的分布式柔性流水线调度[J]. 控制与决策, 2023, 38(3): 779-789.
(Xuan H, Li W T, Li B. Hybrid discrete artificial bee colony algorithm for distributed flexible flowline scheduling with unrelated parallel machines[J]. Control and Decision, 2023, 38(3): 779-789.)

[4] Gupta J N D, Gupta S K. Single facility scheduling with nonlinear processing times[J]. Computers & Industrial Engineering, 1988, 14(4): 387-393.

[5] Browne S, Yechiali U. Scheduling deteriorating jobson a single processor[J]. Operations Research, 1990, 38(3): 495-498.

[6] Zhao C L, Hsu C J. Fully polynomial-time approximation scheme for single machine scheduling with proportional-linear deteriorating jobs[J]. Engineering Optimization, 2019, 51(11): 1938-1943.

[7] Sun X Y, Geng X N. Single-machine scheduling with deteriorating effects and machine maintenance[J]. International Journal of Production Research, 2019, 57(10): 3186-3199.

[8] Guo P, Cheng W M, Wang Y. Parallel machine scheduling with step-deteriorating jobs and setup times by a hybrid discrete cuckoo search algorithm[J]. Engineering Optimization, 2015, 47(11): 1564-1585.

[9] Ding J W, Shen L J, Lv Z P, et al. Parallel machine scheduling with completion-time-based criteria and sequence-dependent deterioration[J]. Computers & Operations Research, 2019, 103: 35-45.

[10] Mir M S S, Rezaeian J, Mohamadian H. Scheduling parallel machine problem under general effects of deterioration and learning with past-sequence-dependent setup time: Heuristic and meta-heuristic approaches[J]. Soft Computing, 2020, 24(2): 1335-1355.

[11] Kang L Y, Ng C T. A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs[J]. International Journal of Production Economics, 2007, 109(1/2): 180-184.

[12] Huang X, Wang M Z. Parallel identical machines scheduling with deteriorating jobs and total absolute differences penalties[J]. Applied Mathematical Modelling, 2011, 35(3): 1349-1353.

[13] Tigane M, Dahane M, Boudhar M. Multiobjective approach for deteriorating jobs scheduling for a sustainable manufacturing system[J]. The International Journal of Advanced Manufacturing Technology, 2019, 101(5/6/7/8): 1939-1957.

[14] Dantzig G B, Wolfe P. Decomposition principle for linear programs[J]. Operations Research, 1960, 8(1): 101-111.

[15] Lemos F K, Cherri A C, de Araujo S A. The cutting stock problem with multiple manufacturing modes applied to a construction industry[J]. International Journal of Production Research, 2021, 59(4): 1088-1106.

[16] Behnke M, Kirschstein T, Bierwirth C. A column generation approach for an emission-oriented vehicle routing problem on a multigraph[J]. European Journal of Operational Research, 2021, 288(3): 794-809.

[17] Chen Z L, Powell W B. Solving parallel machine scheduling problems by column generation[J]. Informs Journal on Computing, 1999, 11(1): 78-94.

[18] 汪恭书, 唐立新. 并行机实时调度问题的LR & CG算法[J]. 控制与决策, 2013, 28(6): 829-836.
(Wang G S, Tang L X. LR & CG algorithm for parallel machine real-time scheduling problem[J]. Control and Decision, 2013, 28(6): 829-836.)

[19] Ding J Y, Song S J, Zhang R, et al. Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches[J]. IEEE Transactions on Automation Science and Engineering, 2016, 13(2): 1138-1154.

[20] Xiong X Y, Zhou P, Yin Y Q, et al. An exact branch-and-price algorithm for multitasking scheduling on unrelated parallel machines[J]. Naval Research Logistics, 2019, 66(6): 502-516.

[21] Desaulniers G, Desrosiers J, Solomon M M. Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems[M]. Boston: Springer, 2002: 309-324.

作者简介

孙鑫伟 (1998−), 男, 硕士生, 从事智能优化调度的研究, E-mail: 1343457235@qq.com;

钱斌 (1976−), 男, 教授, 博士生导师, 从事智能优化调度理论与方法等研究, E-mail: bin.qian@vip.163.com;

胡蓉 (1974−), 女, 副教授, 硕士生导师, 从事优化方法及决策支持系统等研究, E-mail: ronghu@vip.163.com;

张森 (1993−), 男, 博士生, 从事智能算法与优化调度等研究, E-mail: 1615994229@qq.com;

于乃康 (1993−), 男, 博士生, 从事数学规划与优化调度等研究, E-mail: 240004982@qq.com.