DOI: 10.11992/tis.201804016
网络出版地址: http://kns.cnki.net/kcms/detail/23.1538.TP.20180704.1011.004.html

改进猫群算法求解置换流水车间调度问题

裴小兵, 于秀燕
(天津理工大学 管理学院, 天津 300384)

摘 要: 标准猫群算法 (CSO) 在求解最小化最大完工时间的置换流水车间调度问题 (PFSP) 时收敛速度较慢, 同时, 当问题规模变大时容易出现“维数灾难”。为加快寻优速度, 同时避免“维数灾难”, 提出了一种基于分布估计算法的改进猫群算法 (EDA-CSO)。以猫群算法为框架, 嵌入分布估计算法, 在搜寻模式下, 利用概率矩阵挖掘解序列中的优秀基因链组合区块, 使用猫群算法中的跟踪模式更新猫的速度和位置, 从而更新优秀解序列产生子群体。最后, 通过对 Carlier 和 Reeves 标准例题集的仿真测试和结果比较, 验证了该算法良好的鲁棒性和全局搜索能力。

关键词: 置换流水车间调度; 猫群算法; 分布估计算法; 搜寻模式; 概率矩阵; 组合区块; 跟踪模式; 优秀解序列
中图分类号: TP18 **文献标志码**: A **文章编号**: 1673−4785(2019)04−0769−10

中文引用格式: 裴小兵, 于秀燕. 改进猫群算法求解置换流水车间调度问题 [J]. 智能系统学报, 2019, 14(4): 769–778.
英文引用格式: PEI Xiaobing, YU Xiuyan. Improved cat swarm optimization for permutation flow shop scheduling problem[J]. CAAI transactions on intelligent systems, 2019, 14(4): 769–778.

Improved cat swarm optimization for permutation flow shop scheduling problem

PEI Xiaobing, YU Xiuyan
(School of Management, Tianjin University of Technology, Tianjin 300384, China)

Abstract: The standard cat swarm optimization (CSO) has a slow convergence rate in solving the permutation flow shop scheduling problem (PFSP) to minimize the maximum completion time. Meanwhile, the "dimension disaster" is prone to occur when the scale of the problem is large. To speed up the optimization and avoid the "dimension disaster," a CSO algorithm based on the estimation of distribution algorithms is proposed in this paper. Based on the cat swarm algorithm, the distribution estimation algorithm is embedded. In the search mode, the probability matrix is used to mine the excellent gene chain combination blocks in the solution sequence, and the tracking mode in the cat swarm algorithm is used to update the speed and position of the cat, thus updating the excellent solution sequence to generate a subpopulation. Finally, through the simulation test and comparison result of Carlier and Reeves standard example set, the good robustness and global searching ability of the algorithm are verified.

Keywords: permutation flow shop scheduling problem; cat swarm optimization; estimation of distribution algorithm; search mode; probability matrix; combination block; tracking mode; excellent solution sequence

置换流水车间调度问题 (permutation flow shop scheduling problem, PFSP) 是智能制造的核心问题之一, 具有很重要的工程应用价值。研究表明, 该问题属于 NP 难题[1](non-deterministic polynomial-time hard, NP)。常见的 PFSP 求解方法可分为: 精确算法、启发式算法[2] 等。启发式算法能够快速求得问题的可行解, 但这些算法结构比较复杂, 求解最优解比较困难。

在有限的资源条件下, 对 PFSP 的优化可有效增加企业收益, 相关人士一直致力于研究和开发

收稿日期: 2018−04−13. **网络出版日期**: 2018−07−04.
基金项目: 国家创新方法工作专项项目 (2017IM010800).
通信作者: 于秀燕. E-mail: Yu_xiuyan1026@163.com.

高效的优化技术，希望能够快速找到 PFSP 的最优解。受猫日常行为启发，Chu[3] 在 2006 年提出了猫群算法 (cat swarm optimization, CSO)。猫群算法将猫的行为分为搜寻模式和跟踪模式，这两种模式通过结合律 MR(mixture ratio) 进行交互转换。搜寻模式下，通过对个体进行扰动从而使得每个个体向局部最优个体靠近；跟踪模式下，猫通过一定速度来跟踪目标，从而使得个体向全局最优靠近。近年来，猫群算法被应用于很多领域，并取得较好效果。在识别领域，Ganapati[4] 将该算法运用到无线脉冲响应 (IIR) 系统识别；Pradhan[5] 将猫群算法应用于图像识别；Guo 等[6] 基于以上研究基础，运用猫群算法对光伏系统进行优化。在数学研究领域，Pappula 等[7]将猫群算法用于函数优化问题，但收敛速度较慢；为提高猫群算法的全局搜索速度，Tsai 等[8] 提出了猫群算法并行结构，同时验证了在种群规模较小及总迭代次数较少的情况下，该并行结构能有效提高收敛速度。在多目标生产调度方面，Pradhan[5] 运用猫群算法求解多目标优化问题；刘琼等[9] 将该算法运用到混流装配线排序问题。

尽管猫群算法在以上领域取得了一些研究成果，但在单目标置换流水车间调度领域的研究较少。Bouzidi 等[10] 针对置换流水车间调度问题利用标准的猫群算法来求解；此外，Bouzidi 等[11] 还将猫群算法与遗传算法中的相关解码操作结合起来对置换流水车间进行求解；马邦雄等[12] 运用猫群算法对置换流水车间调度问题进行求解并与其他算法进行比较。针对置换流水车间调度问题，尽管上述研究都取得较好效果，但求解过程中该算法收敛速度较慢，同时，当问题规模变大时容易出现“维数灾难”。为加快寻优速度，同时避免“维数灾难”，本研究提出一种基于分布估计算法的改进猫群算法 (EDA-CSO) 用于解决 PFSP。

目前，分布估计算法 (estimation of distribution algorithms, EDA) 已应用于智能学习、函数优化、图像识别、多目标规划[13] 等多个领域。EDA 算法利用概率矩阵模型描述解序列在空间的分布，利用统计学知识分析概率模型，同时，利用概率矩阵产生子群体。为得到更好的解，TZENG 等[14] 将蚁群算法与 EDA 算法相结合，用于求解 PFSP；Chang 等[15] 将 EDA 算法与遗传算法相结合来解决旅行商问题 (traveling salesman problem, TSP)；同时，Chang 等[16] 将区块模型与分布估计算法 (block-based estimation distribution algorithm, BBEDA) 相结合来求解 PFSP 问题；鉴于 EDA 良好的收敛速度，本研究将 CSO 算法与 EDA 相结合，得到高适应度的人工解，同时利用 3 种变异算子对解序列重组以提高解序列的质量和多样性，并通过仿真实例验证 EDA-CSO 算法的有效性及良好的鲁棒性。

1　置换流水车间调度问题数学模型

PFSP 是许多生产调度问题的简化模型，该问题主要研究了 n 个工件 $\{N_1, N_2, \cdots, N_n\}$ 在 m 台机器 $\{M_1, M_2, \cdots, M_m\}$ 上加工，并且每个工件有 m 道加工工序，目标是求使某项生产指标达到最优的方案。相关约束条件如下：1) 所有工件按照相同顺序，即 1, 2, …, m 在机器上加工；2) 某一时刻，任一机器只能加工一个工件；3) 某一时刻，任一工件只能在一台机器上进行加工；4) 任何工件在加工过程中不能中断；5) 所有工件在初始时刻都可以加工。

若 PFSP 以最大完工时间为目标，则该问题可以用 n、m、prmu、$C_{\max}$这几个符号来表示，其中，n 代表总工件数，m 代表总机器数，prmu 代表所有工件在所有机器上的加工顺序相同，$C_{\max}$ 代表最大完工时间。

上述问题的数学描述如下：令 $t(\pi_i, j)$ 代表工件 i 在机器 j 上的加工时间，$C(\pi_i, j)$ 代表工件 i 在机器 j 上的完工时间。 $\Gamma_\xi = \{\pi_1, \pi_2, \cdots, \pi_n\}$ 代表全部工件的一个加工顺序，Γ 代表全部的排序集合。具体的 PFSP 模型可以表示为

$$
\begin{gathered}
C(\pi_1,1) = t(\pi_1,1) \\
C(\pi_i,1) = C(\pi_{i-1},1) + t(\pi_i,1) \quad i = 2,3,\cdots,n \\
C(\pi_1,j) = C(\pi_1,j-1) + (\pi_1,j) \quad j = 2,3,\cdots,n \\
C(\pi_i,j) = \max\{C(\pi_{i-1},j), C(\pi_i,j-1)\} + t(\pi_i,j) \\
i = 2,3,\cdots,n; j = 2,3,\cdots,m \\
C_{\max} = C(\pi_n,m)
\end{gathered}
$$

该问题的目标是得到最优的工件加工顺序 π^*，因而对于其他任意的工件加工顺序 π 都有：$C_{\max}(\pi^*) \leqslant C_{\max}(\pi)$。

2　改进的猫群算法

基于以上优化目标及约束条件，改进的猫群算法求解 PFSP 的流程如下：

1) 初始化种群；

2) 计算猫的适应度值；

3) 改进的猫群算法通过利用混合比率将猫群分为搜寻模式和跟踪模式的 2 个子群，对不同子

群采取不同的方法处理;

4) 在搜寻模式下, 利用概率模型更新记录可行解信息并更新可行解;

5) 先后运用区块挖掘及区块竞争来得到适应度较高的人工解;

6) 为提高人工解的质量和多样性, 运用搜寻算子对人工解进行重组;

7) 在跟踪模式中利用基于二元竞赛法的速度−位移模型对猫的速度、位置更新, 直至达到全局最优;

8) 算法终止条件判断。

基于区块的猫群算法求解 PFSP 的具体流程如图 1 所示。

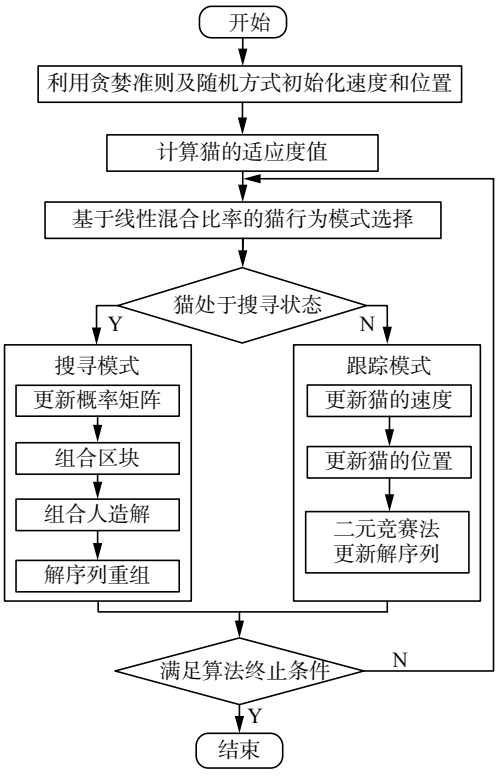


图 1　EDA-CSO 流程图

Fig. 1　The flow chart of EDA-CSO

2.1　复杂度分析

一般而言, 时间复杂度能够度量算法的运行时间及算法的优劣。假设种群规模为 N, 猫处于搜寻模式下的概率为 M_R, 迭代次数为 T, 根据改进猫群算法的步骤进行时间复杂度分析。

步骤 1) 中种群初始化需要 N 次操作, 所以步骤 1) 的时间复杂度为 $O(N)$;

步骤 2) 中用以计算适应度的时间复杂度为 $O(N)$;

步骤 3) 中用以选择猫行为方式的时间复杂度为 $O(N)$;

步骤 4)~6) 中执行搜寻模式的猫, 每次迭代更新概率矩阵 1 次, 挖掘区块 n 次, 区块竞争 1 次, 组合人造解 1 次, 解序列重组 1 次, 故搜寻模式下的时间复杂度为 $O(M_RN+M_R^2N^2+M_RN+M_RN+M_RN)$;

步骤 7) 中执行跟踪模式的猫, 每次迭代速度更新 1 次, 位置更新 1 次, 解序列更新 1 次, 故跟踪模式下的时间复杂度为 $O((1-M_R)(N+N+N))$;

步骤 8) 中终止条件判断需要 1 次操作, 故此步骤的时间复杂度为 $O(1)$。

由以上可知, 该算法的时间复杂度为 $O(T(M_R+5)N+TM_RN^2)$, 故时间复杂度主要与与初始种群规模、混合比率及迭代次数有关。

2.2　初始化种群

传统的猫群算法通过轮盘赌的方式生成初始种群, 由于初始种群个体适应度较低, 在一定程度上制约了算法的收敛速度。本研究利用贪婪准则[16] 寻及轮盘赌相结合的方式优化初始化种群, 以达到加快收敛速度的同时, 增加初始解的多样性。在利用贪婪准则初始化种群时, 随机选取第一个工件 i, 并将其加入到 $\Gamma_\xi=\{\pi_1,\pi_2,\cdots,\pi_n\}$ 里, 然后在其他未加工的工件中搜索, 寻找下一个工件, 其在所有未加工工件中加工时间最短, 将其加至 $\Gamma_\xi=\{\pi_1,\pi_2,\cdots,\pi_n\}$ 里, 并将其作为当前加工工件, 继续搜索并增加下一个加工工件, 直至所有工件都加入加工顺序集 $\Gamma_\xi=\{\pi_1,\pi_2,\cdots,\pi_n\}$。运用贪婪准则产生的是初始解序列, 因而, 需要将初始解序列转变为一定区间内的位置矢量。具体如式 (1):

$$x_{i,j}=x_{\min,j}+\frac{x_{\max,j}-x_{\min,j}}{n}\cdot(s_{i,j}-1+r_3),\quad j=1,2,\cdots,m \tag{1}$$

式中: $x_{i,j}$ 代表猫在 j 维的位置值; $s_{i,j}$ 代表初始解序列的第 j 维工件序号; $x_{\min,j}$ 及 $x_{\max,j}$ 表示猫在连续空间中位置矢量的上下界值; r_3 代表 [0, 1] 区间内随机产生的随机数。

初始位置及速度产生方式为

$$X_i=X_{\min}+(X_{\max}-X_{\min})r_1 \tag{2}$$

$$V_i=V_{\min}+(V_{\max}-V_{\min})r_2 \tag{3}$$

式中: X_i 在区间 $[X_{\min},X_{\max}]$ 内连续变化; V_i 在区间 $[V_{\min},V_{\max}]$ 内连续变化。

2.3　基于混合比率的猫行为模式选择

传统的猫群算法不能根据算法的迭代次数合理分配局部搜索和全局搜索的比重, 若在算法迭代前期采用较大的混合比率的跟踪猫, 可以增加算法的全局搜索能力, 但在算法迭代后期搜寻猫所占比率较大, 可以提高解精度和收敛性。因

此，本研究采用文献 [9] 中的一种猫行为混合比率选择法，具体如图 2 所示。

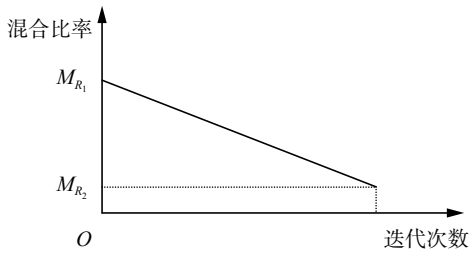


图 2　混合比率分配

Fig. 2　The allocation of mixed ratio

该线性混合比率的计算公式为

$$M_R = M_{R_1} + \frac{(M_{R_2} - M_{R_1}) \times T}{T_0} \tag{4}$$

式中：T 代表迭代次数；T_0 代表最大迭代次数。

2.4　搜寻模式

2.4.1　构建概率模型

本研究采用位置矩阵和相依矩阵来记载不同的加工信息，位置矩阵用来表示工件和所处加工位置之间的关系，相依矩阵用来表示任意两个工件之间的加工前后顺序。

对初始解的适应度函数值从小到大排序，选择前 s 个优秀解组成优秀解集合 ∂，$\partial = \{\Gamma_1, \Gamma_2, \cdots, \Gamma_s\}$，同时位置矩阵及相依矩阵按以下方式更新：

$$B_{ij}^k = \begin{cases} 1, & \text{如果工件}i\text{位于}j\text{位置上} \\ 0, & \text{否则} \end{cases} \tag{5}$$

$$i = 1,2,\cdots,n;\ j = 1,2,\cdots,m;\ k = 1,2,\cdots,s$$

$$T_{ij}(t) = T_{ij}(t-1) + \sum_{k=1}^{s} B_{ij}^k, \tag{6}$$

$$i = 1,2,\cdots,n;\ j = 1,2,\cdots,m;\ k = 1,2,\cdots,s$$

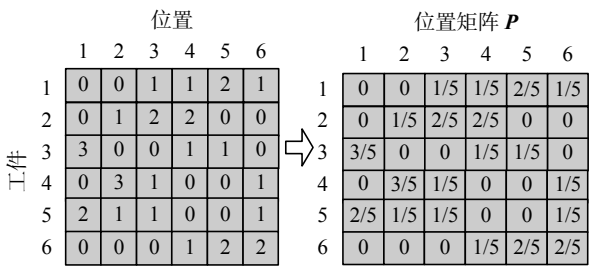
$$Y_{il}^k = \begin{cases} 1, & \text{如果工件}i\text{紧邻}l \\ 0, & \text{否则} \end{cases} \tag{7}$$

$$i = 1,2,\cdots,n;\ l = 1,2,\cdots,n;\ k = 1,2,\cdots,s$$

$$T_{ij}(t) = T_{ij}(t-1) + \sum_{k=1}^{s} Y_{ij}^k, \tag{8}$$

$$i,l = 1,2,\cdots,n;\ j = 1,2,\cdots,m;\ k = 1,2,\cdots,s$$

具体位置、相依矩阵更新方式如图 3、图 4 所示。

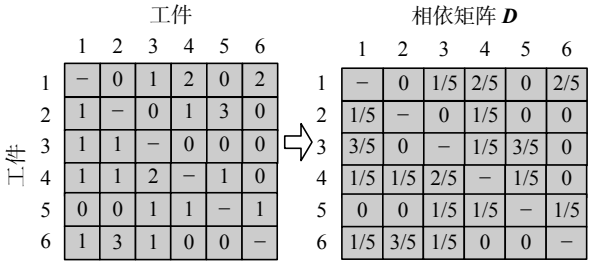


工件	1	2	3	4	5	6
1	0	0	1	1	2	1
2	0	1	2	2	0	0
3	3	0	0	1	1	0
4	0	3	1	0	0	1
5	2	1	1	0	0	1
6	0	0	0	1	2	2

工件	1	2	3	4	5	6
1	0	0	1/5	1/5	2/5	1/5
2	0	1/5	2/5	2/5	0	0
3	3/5	0	0	1/5	1/5	0
4	0	3/5	1/5	0	0	1/5
5	2/5	1/5	1/5	0	0	1/5
6	0	0	0	1/5	2/5	2/5

图 3　位置矩阵更新方式

Fig. 3　Updating method of position matrix



工件	1	2	3	4	5	6
1	–	0	1	2	0	2
2	1	–	0	1	3	0
3	1	1	–	0	0	0
4	1	1	2	–	1	0
5	0	0	1	1	–	1
6	1	3	1	0	0	–

工件	1	2	3	4	5	6
1	–	0	1/5	2/5	0	2/5
2	1/5	–	0	1/5	0	0
3	3/5	0	–	1/5	3/5	0
4	1/5	1/5	2/5	–	1/5	0
5	0	0	1/5	1/5	–	1/5
6	1/5	3/5	1/5	0	0	–

图 4　相依矩阵更新方式

Fig. 4　Updating method of dependency matrix

2.4.2　组合区块

组合区块能够降低种群迭代复杂度，降低解的维度，加快收敛速度。以单个机器 10 个工件的排序为例，如图 5 所示，未组合区块之前母体中的工件{工件 1，工件 2，…，工件 10}为单独的基因，此时可产生 10! 种排列组合，而组合区块后排列组合变为 5! 种，在很大程度上降低了解的维度。为找出含有高竞争优势的区块，本研究从区块挖掘与区块竞争两个步骤来组合区块。

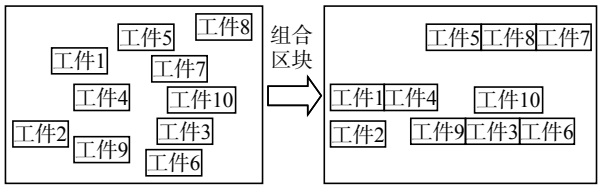


图 5　种群迭代复杂度比较

Fig. 5　The comparison of iterative complexity of population

1) 区块挖掘

根据位置矩阵和相依矩阵模型提供的相关信息，进行相应的区块挖掘。以随机的方式选择区块的起始位置，产生一个符合最小长度的空白区块，设区块的最小长度为 3。

在空白区块中放入最适合的工件，为计算方便，将各工件在位置矩阵与相依矩阵中的累计结果转化为概率，同时，将两矩阵的概率整合成一个合并概率 (combination probability, CP)，利用轮盘赌对合并概率进行挑选，其中，起始位置按照依据位置矩阵的概率，以轮盘赌进行选择。选出第 1 个工件后，以合并概率对第 2 个、第 3 个工件等进行选择。经一系列研究发现，在进化前期前后关系相比位置关系更能影响解序列的适应度高低[17]，相反，在进化中后期位置关系比工件前后关系更重要，因而在进化的不同阶段，令位置矩阵的权重随着世代数由 0.3 增加到 0.7，反之，相依矩阵由 0.7 递减到 0.3。合并概率的计算如式 (9) 所示，其中，i 代表工件编码、γ 表示 i 紧前工件号码、j 表示工件 i 所在的位置、n 表示工件总

数、CP_i 表示工件 i 的合并概率、W_{dom} 与 W_{dep} 分别表示当前位置矩阵与相依矩阵的合并权重值、$P_{i,j}^{dom}$ 为位置矩阵中工件 i 处于位置 j 上的概率、$P_{i,\gamma}^{dep}$ 为相依矩阵中工件 j 紧前于工件 i 的概率。

$$CP_i = (W_{dom} \times P_{i,j}^{dom}) + (W_{dep} \times P_{i,\gamma}^{dep}) \quad i,\gamma = 1,2,\cdots,n;\ j = 1,2,\cdots,m \tag{9}$$

计算出所有工件的合并概率, 并运用轮盘赌选择出区块的第 2 个工件及第 3 个工件等, 具体如图 6 所示。

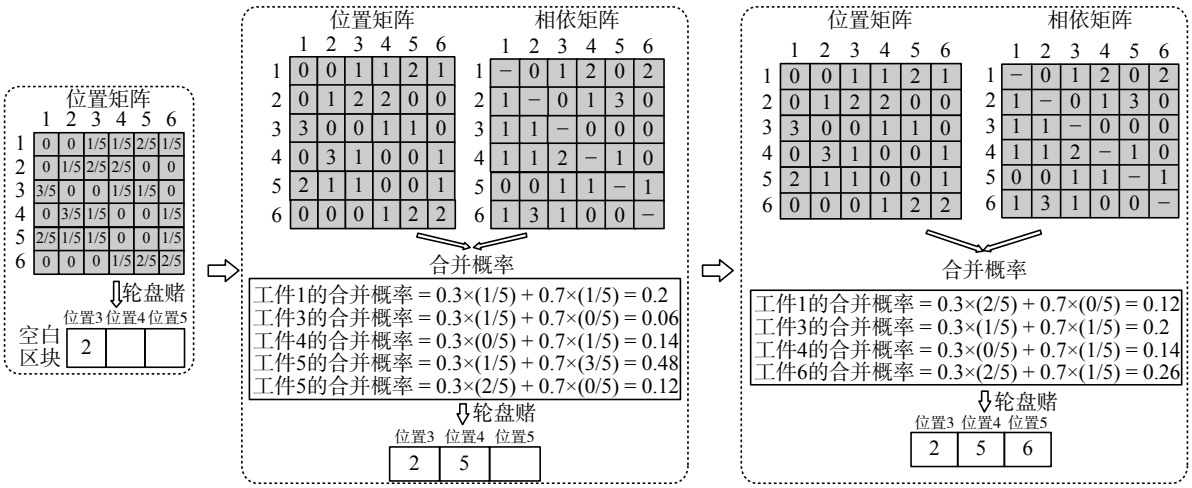


图 6　以合并概率挖掘区块

Fig. 6　Mining blocks by combining probability

随着区块长度的增长, 总概率逐渐降低, 即错误的概率越大。例如, 一个由 5 个工件{工件 1, 工件 2, 工件 3, 工件 4, 工件 5}组成的区块, 其概率分别为 0.5、0.8、0.6、0.4、0.3, 此区块的总概率 0.5×0.8×0.6×0.4×0.3=0.028 8, 若该区块由 3 个工件{工件 1, 工件 2, 工件 3}组成, 则总概率为 0.5×0.8×0.6=0.24, 由此可见, 错误概率随着区块长度的增长而变大。为保证区块的质量不会随着区块长度的增加而降低, 设计一个阈值用以筛选上述挖掘的区块, 阈值随着迭代次数的增加由 0.24 增到 0.8。将符合阈值的区块暂存在区块库中, 区块库中保留着本次迭代中全部符合阈值的区块。

2) 区块竞争

区块挖掘完成后, 利用区块竞争选择竞争力较大的区块, 每迭代一次, 其产生的区块与区块库中的区块进行比较, 最终选择其中竞争优势较大的区块, 更新区块库。区块进行竞争时, 如果几个区块之间工件重复或者区块之间包含的位置出现重复, 利用平均概率对这几个区块比较, 淘汰概率较小的区块。

平均概率的计算方法: 区块的第一个工件位置矩阵概率加上其余工件的合并概率之和除以区块总长度, 即可求出该区块的平均概率。平均概率的计算如式 (10) 所示:

$$P_{B^i}^{AVG} = \frac{P_{B_l^h}^{dom} + \sum_{l=2}^{n_1} CP_{B_l^h}}{n_1} \tag{10}$$

式中: h 为区块编号; B_l^h 为第 i 个区块的第 l 个工件; j 为 B_l^h 的位置; n_1 为当前区块长度。

具体的区块竞争如图 7 所示, 新产生的 2 个区块{D_1, D_2}, 区块库包含 3 个区块{D_3, D_4, D_5}, 新产生的区块 D_1 与区块库中的 D_3 重复出现工件 6, 此时对 D_1 与 D_3 的平均概率进行比较, D_1 的平均概率为 0.26, 大于 D_3 的平均概率 0.23, 故 D_1 取代 D_3 进入区块库中, D_3 被淘汰删除。D_2 与 D_4 在位置 11 重叠, 比较与 D_4 的平均 D_2 概率由图可知 D_2 高于 D_4, 故淘汰 D_4。D_5 未与任何区块发生工件或位置重复, 因此继续保存于区块库中。

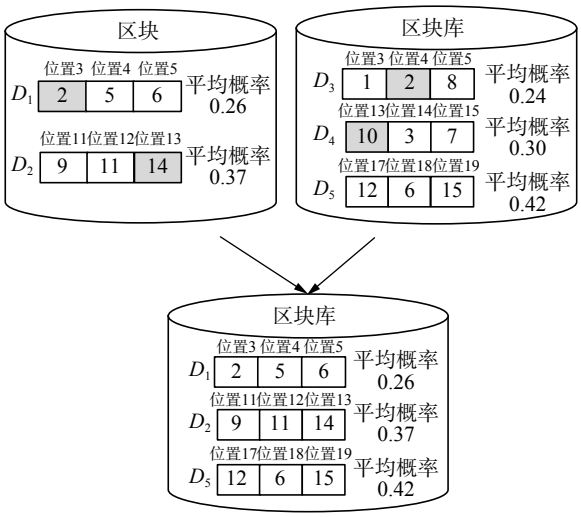


图 7　区块竞争

Fig. 7　The competition of blocks

2.4.3　组合人工解

为提高解序列的质量，在完成区块挖掘后，利用区块库中保留的区块组合人工解，具体步骤如下：

1) 从人工解的第一个位置开始挖掘，挖掘方法与上述区块挖掘相同，第一个位置利用位置矩阵中概率以轮盘赌的方式选择工件；

2) 其余 N−1 个位置以轮盘赌的形式对合并概率进行选择；

3) 每选出一个工件，将该工件与所有其他区块的起始位置进行比较，若与其他区块的位置及工件都相同，则将此区块直接复制到人工解中，然后从下个位置继续进行选择和比较，直至人工解组合完成。

具体的操作方式如图 8 所示。

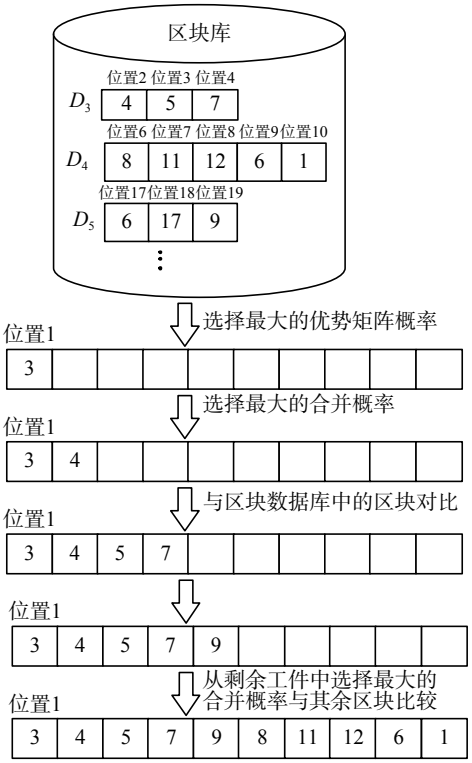


图 8　人工解组合过程

Fig. 8　The combination process of artificial solution

2.4.4　解序列重组

为了增加该算法寻找最优解的概率，本研究对每只猫即解序列进行相应的重组操作。首先每只猫在记忆池中将自身位置复制 H 份，对记忆池中的猫即解序列进行相应的重组操作，使得所有猫能够到达新位置，计算记忆池中所有猫的适应度，然后将适应度最高的猫代替当前猫，以完成猫的位置更新。在记忆池进行变异操作时，为避免局部最优及增加解序列的多样性，本研究将解序列随机切成 N 个片段，选取最短的两个片段，利用遗传算法中的插入搜寻算子操作，对解序列重组，使得两个最短片段形成一个基因片段。若重组之后的片段不再是最短片段，则对解序列中的最长片段及该片段按照合并概率重组，若区块库含有以被选择的工件开头的区块，则直接插入，具体如图 9 和图 10 所示。

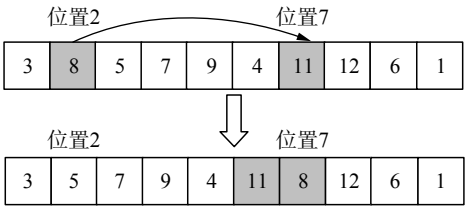


图 9　插入搜寻算子

Fig. 9　Insert search operator

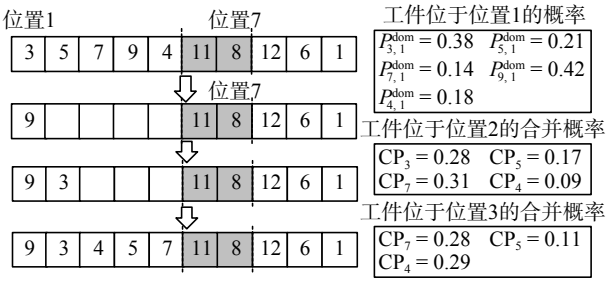


图 10　利用概率重组解序列

Fig. 10　Using probability recombination solution sequence

2.5　跟踪模式

猫的跟踪模式是为了使个体靠近全局最优解，在该模式下，猫与群体最优位置进行比较来更新个体速度与位置[18]。跟踪模式可以根据以下两步进行。

1) 速度的更新。

任意时刻，每个猫都有一个速度，第 i 只猫的当前速度可表示为 $V_i=\{v_{i1},v_{i2},\cdots,v_{il}\}$，所有猫的速度按照式 (11) 进行更新：

$$V_i(n+1)=V_i(n)\times w+c\times \text{rand}\times[X_{\text{best}}(n)-V(n)] \tag{11}$$

$$w=w_{\max}-(w_{\max}-w_{\min})\times(t/t_{\max}) \tag{12}$$

式中：$V_i(n+1)$ 表示更新后第 i 只猫的速度；w 为惯性权重，w 的大小由式 (12) 决定；t 表示迭代信息；$t_{\max}$ 表示最大迭代次数；c 代表加速度常数；rand 服从 [0, 1] 均匀分布。

2) 位置更新。

每只猫的位置更新由式（13）决定：

$$X_i(n+1)=X_i(n)+V_i(n+1) \tag{13}$$

式中 $X_i(n+1)$ 代表第 i 只猫更新后的位置。

3) 如果第 i 只猫分的新位置超出搜索空间，则将速度乘以−1，从反方向继续搜索。

4) 利用二元竞赛法更新解序列，即将母体种群与子群的适应度两两对比并将适应度最高的子群代替母体种群，进行下一次搜索。

3　仿真测试

3.1　参数设置

为了验证基于区块的猫群算法的性能, 本研究测试数据采用 Carlier[19] 中的 Car 类基准例题集进行测试, 应用该算法求解这些案例, 并与其他文献中的算法进行比较。本研究程序利用 Visual Studio2015 中的 C++, 在操作系统为 Windows8, 处理器的主频为 2.71 G 的 Intel(R)Core(TM)i5-6400 处理器 8 G 内存的电脑上进行仿真测试。参数设计: 仿真测试次数为 20, 初始种群的数量为 100, 最大的迭代次数为 100。本研究将各个算法的最优相对误差 (BRE)、平均相对误差 (ARE) 及最差相对误差 (WRE) 进行比较。其中, C^*表示已知算例的最优解, $C_{\min}$ 表示所求解的最优值, $\overline{C}$ 表示所求解的平均值, $C_{\max}$ 表示所求解的最差值。

$$\mathrm{BRE}=\frac{(C_{\min}-C^*)}{C^*}\times 100\% \tag{14}$$

$$\mathrm{ARE}=\frac{(\overline{C}-C^*)}{C^*}\times 100\% \tag{15}$$

$$\mathrm{WRE}=\frac{(C_{\max}-C^*)}{C^*}\times 100\% \tag{16}$$

3.2　实验测试与比较

针对 Carlier 例题, 本研究将基于分布估计算法的改进猫群算法 (EDA-CSO) 与猫群算法 (CSO)[12]、标准粒子群算法 (PSO)[12]、蝙蝠算法 (BA)[12] 进行比较。具体比较结果如表 1 所示。

由表 1 和图 11、图 12 可知, 尽管这 4 种算法在 Carlier 案例中均能求出问题的最优解 (BRE 为 0), 然而 EDA-CSO 的 ARE 均优于其他算法。表明了本算法的整体求解性能优于其他算法。此外, EDA-CSO 显然比 CSO、PSO、BA 的 ARE 和 WRE 更小, 这是因为通过调整猫的混合比率, 迭代前期采用较大的混合比率的跟踪猫, 增加了算法的全局搜索能力, 在算法迭代后期增加搜寻猫所占比率较大, 增加了局部搜索能力, 减少了求解误差, 提高了精度和收敛性, 因此 EDA-CSO 算法明显优于文中其他算法。

表 1　EDA-CSO、CSO、PSO 和 BA 的测试结果对比

Table 1　Comparison of test results for EDA-CSO、CSO、PSO and BA

算例	n、m	C^*	EDA-CSO			CSO			PSO			BA		
			BRE	ARE	WRE	BRE	ARE	WRE	BRE	ARE	WRE	BRE	ARE	WRE
Car_1	11、5	7 038	0.00	0.01	0.20	0.00	0.02	0.27	0.00	0.52	2.72	0.00	0.32	1.68
Car_2	13、4	7 166	0.00	1.37	5.00	0.00	2.76	6.01	0.00	5.22	10.8	0.00	3.68	6.29
Car_3	12、5	7 312	0.00	1.85	3.17	0.00	2.03	3.86	0.00	3.77	9.04	0.00	2.36	3.87
Car_4	14、4	8 003	0.00	0.37	4.98	0.00	0.44	5.25	0.00	3.86	6.55	0.00	2.64	5.25
Car_5	10、7	7 720	0.00	0.18	1.09	0.00	0.29	1.31	0.00	1.24	2.12	0.00	1.01	1.84
Car_6	8、9	8 505	0.00	0.52	2.12	0.00	0.77	2.47	0.00	2.06	5.7	0.00	1.44	3.39
Car_7	7、7	6 590	0.00	0.15	2.03	0.00	0.24	2.47	0.00	1.61	4.51	0.00	0.91	2.58
Car_8	8、8	8 366	0.00	0.09	1.01	0.00	0.29	1.33	0.00	3.85	8.88	0.00	1.08	2.51

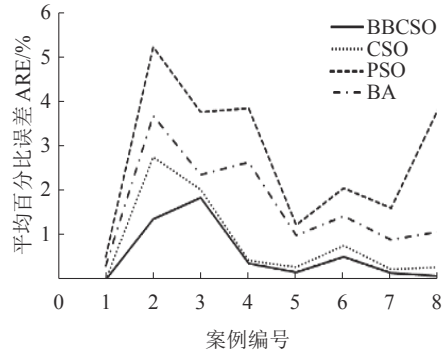


图 11　平均相对误差比较

Fig. 11　Comparison of average relative error

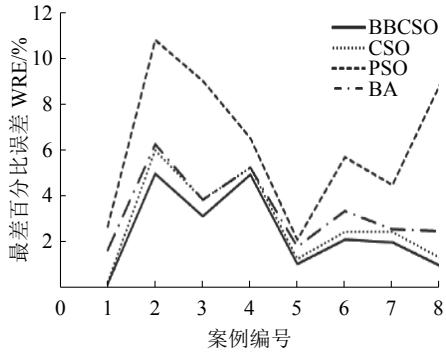


图 12　最差相对误差比较

Fig. 12　Comparison of the worst relative error

为进一步验证算法的有效性, 针对 Reeves[20] 例题, 将基于猫群算法的改进估计分布算法 (EDA-CSO) 与猫群算法 (CSO)[10]、基于区块的分布估计算法 (BBEDA)[17]、基于粒子群算法的分布估计算法 (PSO-EDA)[21] 各迭代 20 000 次并进行比较。具体结果如表 2 所示。其中, $C_{\min}$ 表示所求解的最优值, s; G 表示算法的运行时间, s。

表 2　Reeves 案例测试结果比较

Table 2　Performance comparison on Reeves's instances

算例	n、m	C*	EDA-CSO			CSO			PSO-EDA			BBEDA		
			C_{min}/s	BRE	G/s	C_{min}/s	BRE	G/s	C_{min}/s	BRE	G/s	C_{min}/s	BRE	G/s
Rec01	20、5	1 247	1 247	0.00	2.5	1 247	0.00	2.0	1 247	0.00	3.0	1 247	0.00	2.7
Rec03	20、5	1 109	1 109	0.00	3.0	1 109	0.00	3.0	1 109	0.00	2.9	1 109	0.00	3.2
Rec05	20、5	1 242	1 242	0.00	2.1	1 245	0.24	1.0	1 245	0.24	1.3	1 242	0.00	1.9
Rec07	20、10	1 566	1 566	0.00	3.1	1 566	0.00	3.0	1 566	0.00	2.5	1 566	0.00	3.6
Rec09	20、10	1 537	1 537	0.00	2.0	1 537	0.00	2.0	1 537	0.00	2.3	1 537	0.00	2.5
Rec11	20、10	1 431	1 431	0.00	1.0	1 431	0.00	1.0	1 431	0.00	2.1	1 431	0.00	1.8
Rec13	20、15	1 930	1 930	0.00	114.0	1 930	0.00	178.0	1 930	0.10	181.1	1 930	0.00	119.0
Rec15	20、15	1 950	1 950	0.00	99.0	1 950	0.00	101.0	1 950	0.00	101.7	1 950	0.00	100.0
Rec17	20、15	1 902	1 902	0.00	108.0	1 902	0.00	115.8	1 902	0.00	119.0	1 902	0.00	110.2
Rec19	30、10	2 093	2 097	0.21	14.0	2 099	0.29	18.1	2 099	0.29	37.0	2 099	0.29	36.9
Rec21	30、10	2 017	2 019	0.10	369.0	2 020	0.15	486.2	2 040	1.14	484.0	2 036	0.94	374.0
Rec23	30、10	2 011	2 017	0.30	13.0	2 020	0.45	21.9	2 019	0.40	26.3	2 020	0.45	15.2
Rec25	30、15	2 513	2 515	0.11	341.1	2 525	0.48	377.1	2 520	0.28	369	2 530	0.68	352.6
Rec27	30、15	2 373	2 373	0.00	53.3	2 396	0.97	60.8	2 396	0.97	59.8	2 379	0.25	69.4
Rec29	30、15	2 287	2 289	0.09	259.0	2 305	0.79	332.1	2 295	0.35	267.7	2 292	0.22	268.5
Rec31	50、10	3 045	3 051	0.22	604.1	3 058	0.43	900.0	3 053	0.26	608.5	3 056	0.36	610.2
Rec33	50、10	3 114	3 114	0.00	123.8	3 114	0.00	163.0	3 114	0.00	125.9	3 114	0.00	131.7
Rec35	50、10	3 277	3 277	0.00	4.6	3 277	0.00	7.1	3 277	0.00	8.7	3 277	0.00	7.5
Rec37	75、20	4 951	5 004	1.08	943.0	5 096	2.93	1 583.0	5 041	1.84	850.0	5 111	3.23	867.1
Rec39	75、20	5 087	5 125	0.74	395.0	5 161	1.45	568.1	5 134	0.92	639.0	5 189	2.01	644.0
Rec41	75、20	4 960	5 008	0.98	1 542.0	5 087	2.56	2 071.0	5 050	1.82	1 601.0	5 115	4.26	1 550.1

由表 2 可知，由于增加算子，尽管本算法在求解 Rec01、Rec05、Rec07 时，处理时间上慢于 CSO 算法，但 BRE 均优于 CSO、BBEDA 及 PSO-EDA 算法。这是因为虽然迭代过程中增加了算子，但由于区块具有降维的作用，且随着问题复杂度的增加，效果越明显，因而在一定程度上降低了运算时间。为直观体现本算法降低复杂度，加快收敛速度的性能，以 EDA-CSO、BBEDA 求解 Rec09、Rec11、Rec13、Rec15 为例，具体如图 13~16 所示。

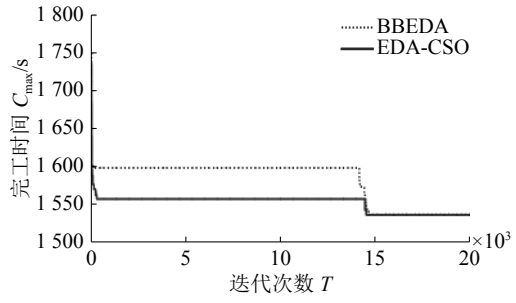


图 13　Rec09 收敛图

Fig. 13　Rec09 convergence graph

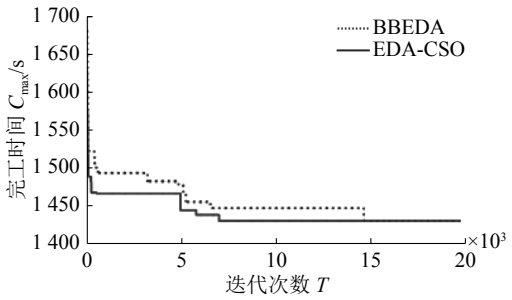


图 14　Rec11 收敛图

Fig. 14　Rec11 convergence graph

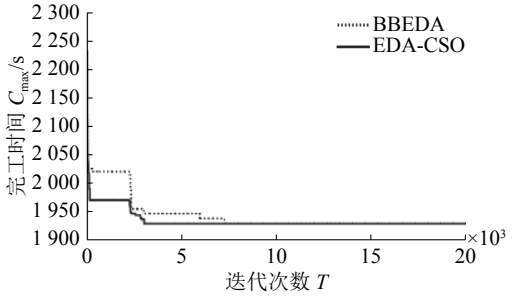


图 15　Rec13 收敛图

Fig. 15　Rec13 convergence graph

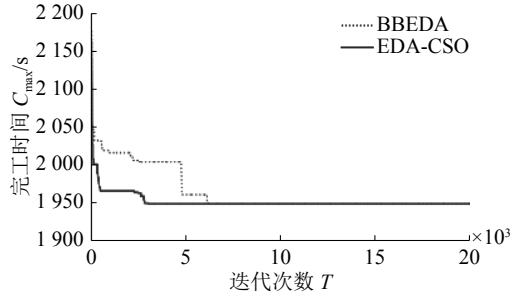


图 16　Rec15 收敛图

Fig. 16　Rec15 convergence graph

从图 13~16 可以看出, 在相同的迭代次数下, EDA-CSO 与 BBED 算法相比, 尽管两者最优误差均为 0, 但 EDA-CSO 的收敛速度均优于 BBEDA 算法的收敛速度。

4　结束语

本研究在猫群算法的基础上进行改进, 提出了基于分布估计算法的改进猫群算法, 用以解决置换流水车间调度问题。在猫搜寻阶段, 运用贪婪准则于轮盘赌相结合的方式初始化种群, 加快收敛速度; 采用位置矩阵与相依矩阵结合的方式挖掘区块; 利用区块竞争产生人工解; 在不同的进化阶段采用不同的变异方式以提高人工解的质量和多样性; 同时, 为了使个体靠近全体最优解, 通过与群体最优位置进行比较来更新个体速度与位置。针对 Carlier 和 Reeves 标准案例运用该算法进行求解, 最后, 将各个算法的实验结果进行比较, 验证了该算法的有效性和鲁棒性。

本研究仅将该算法应用于置换流水车间调度问题, 未将该算法应用于混流生产线、TSP 等其他组合优化问题, 今后可以进一步从这几个方面展开研究。

参考文献:

[1] GAREY M R, JOHNSON D S, SETHI R. The complexity of flowshop and jobshop scheduling[J]. Mathematics of operations research, 1976, 1(2): 117–129.

[2] 李小缤, 白焰, 耿林霄. 求解置换流水车间调度问题的改进遗传算法 [J]. 计算机应用, 2013, 33(12): 3576–3579.
LI Xiaobin, BAI Yan, GENG Linxiao, et al. Improved genetic algorithm for solving permutation flow shop scheduling problem[J]. Journal of computer applications, 2013, 33(12): 3576–3579.

[3] CHU Shuchuan, TSAI P W, PAN J S. Cat swarm optimization[C]//Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence. Guilin, China, 2006, 4099: 854–858.

[4] GANAPATI P, PRADHAN P M, MAJHI B. ⅡR system identification using cat swarm optimization[J]. Expert systems with applications, 2011, 38(10): 12671–12683.

[5] PRADHAN P M, PANDA G. Solving multiobjective problems using cat swarm optimization[J]. Expert systems with applications, 2012, 39(3): 2956–2964.

[6] GUO Lei, MENG Zhuo, SUN Yize, et al. A modified cat swarm optimization based maximum power point tracking method for photovoltaic system under partially shaded condition[J]. Energy, 2018, 144: 501–514.

[7] PAPPULA L, GHOSH D. Cat swarm optimization with normal mutation for fast convergence of multimodal functions[J]. Applied soft computing, 2018, 66: 473–491.

[8] TSAI P W, PAN J S, CHEN S M, et al. Parallel cat swarm optimization[C]//Proceedings of 2008 International Conference on Machine Learning and Cybernetics. Kunming, China, 2008: 3328–3333.

[9] 刘琼, 范正伟, 张超勇, 等. 基于多目标猫群算法的混流装配线排序问题 [J]. 计算机集成制造系统, 2014, 20(2): 333–342.
LIU Qiong, FAN Zhengwei, ZHANG Chaoyong, et al. Mixed model assembly line sequencing problem based on multi-objective cat swarm optimization[J]. Computer integrated manufacturing system, 2014, 20(2): 333–342.

[10] BOUZIDI A, RIFFI M E. Cat swarm optimization to solve flow shop scheduling problem[J]. Journal of theoretical and applied information technology, 2015, 72(2): 239–243.

[11] BOUZIDI A, RIFFI M E. Cat swarm optimization to solve job shop scheduling problem[C]//Proceedings of the Third IEEE International Colloquium in Information Science and Technology. Tetouan, Morocco, 2015: 202–205.

[12] 马邦雄, 叶春明. 利用猫群算法求解流水车间调度问题 [J]. 现代制造工程, 2014(6): 12–15, 71.
MA Bangxiong, YE Chunming. The research of flowshop scheduling problem based on cat swarm optimization[J]. Modern manufacturing engineering, 2014(6): 12–15, 71.

[13] 陶新民, 徐鹏, 刘福荣, 等. 组合分布估计和差分进化的多目标优化算法 [J]. 智能系统学报, 2013, 8(1): 39–45.
TAO Xinmin, XU Peng, LIU Furong, et al. Multi-objective optimization algorithm composed of estimation of distribution and differential evolution[J]. CAAI transactions on intelligent systems, 2013, 8(1): 39–45.

[14] TZENG Y R, CHEN C L, CHEN C L. A hybrid EDA

with ACS for solving permutation flow shop scheduling [J]. The international journal of advanced manufacturing technology, 2012, 60(9/10/11/12): 1139–1147.

[15] CHANG P C, HUANG W H, TING C J. Dynamic diversity control in genetic algorithm for mining unsearched solution space in TSP problems[J]. Expert systems with applications, 2010, 37(3): 1863–1878.

[16] KIM D, HALDAR J P. Greedy algorithms for nonnegativity-constrained simultaneous sparse recovery[J]. Signal processing, 2016, 125: 274–289.

[17] CHANG P C, CHEN Menghui. A block based estimation of distribution algorithm using bivariate model for scheduling problems[J]. Soft computing, 2014, 18(6): 1177–1188.

[18] RAUTRAY R, BALABANTARAY R C. Cat swarm optimization based evolutionary framework for multi document summarization[J]. Physica a: statistical mechanics and its applications, 2017, 477: 174–186.

[19] CARLIER J. Ordonnancements à contraintes disjonctives [J]. RAIRO-operations research, 1978, 12(4): 333–350.

[20] REEVES C R. A genetic algorithm for flowshop sequencing[J]. Computers and operations research, 1995, 22(1): 5–13.

[21] LIU Hongcheng, GAO Liang, PAN Quanke. A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem[J]. Expert systems with applications, 2011, 38(4): 4348–4360.

作者简介：

裴小兵，男，1965 年生，教授，博士，天津工业工程学会理事，主要研究方向为生产调度、系统仿真。发表学术论文 30 余篇。

于秀燕，女，1992 年生，硕士研究生，主要研究方向为生产调度、系统仿真。