求解总拖期时间最小化流水车间调度问题的多智能体进化算法

王大志 [1, 2]　刘士新 [1, 2]　郭希旺 [1, 2]

摘　要　针对总拖期时间最小化的置换流水车间调度问题 (Total tardiness permutation flow-shop scheduling problem) 提出了一种基于多智能体的进化搜索算法. 在该算法中, 采用基于延迟时间排序的学习搜索策略 (Tardiness rank based learning), 快速产生高质量的新个体, 并根据概率更新模型进行智能体网格的更新进化. 同时通过实验设计的方法探讨了算法参数设置对算法性能的影响. 为了验证算法的性能, 求解了 Vallada 标准测试集中 540 个测试问题, 并将测试结果与一些代表算法进行比较, 验证了该算法的有效性.

关键词　流水车间调度, 总拖期时间, 多智能体, 进化算法

引用格式　王大志, 刘士新, 郭希旺. 求解总拖期时间最小化流水车间调度问题的多智能体进化算法. 自动化学报, 2014, **40**(3): 548−555

DOI　10.3724/SP.J.1004.2014.00548

A Multi-agent Evolutionary Algorithm for Solving Total Tardiness Permutation Flow-shop Scheduling Problem

WANG Da-Zhi[1, 2]　LIU Shi-Xin[1, 2]　GUO Xi-Wang[1, 2]

Abstract　In this research, we propose a multi-agent evolutionary algorithm for the permutation flow-shop scheduling problem (PFSP) considering the total tardiness minimization criterion. The algorithm includes the tardiness rank based learning scheme to generate high quality solution by using the specific knowledge of the related problem. We also develop and integrate the probability acceptance model into the proposed algorithm to evolve the whole agent lattice network. A complete calibration of the different parameters of the proposed algorithm by means of a design of experiment approach is given. Using the 540 benchmark problems, a comparative evaluation with other heuristic methods in the literature have been carried out. The results show that the proposed algorithm is effective and competitive.

Key words　Permutation flow-shop scheduling problem (PFSP), total tardiness, multi-agent, evolutionary algorithm

Citation　Wang Da-Zhi, Liu Shi-Xin, Guo Xi-Wang. A multi-agent evolutionary algorithm for solving total tardiness permutation flow-shop scheduling problem. *Acta Automatica Sinica*, 2014, **40**(3): 548−555

置换流水车间调度问题 (Permutation flow shop scheduling problem, PFSP) 是一类被广泛研究的组合优化问题, 同时也是一个典型的 NP-hard 问题. 由于 PFSP 问题是许多现场生产调度问题简化后的模型, 因此, 其研究具有重要的理论指导意义和工程应用价值. 置换流水车间调度问题可以描述为: n 个工件的集合 $N = \{1, 2, \cdots, n\}$, 依次在 m 台机器的集合 $M = \{1, 2, \cdots, m\}$ 上进行加工, 已知工件 $j\,(1 \le j \le n)$ 在机器 $i\,(1 \le i \le m)$ 上的加工时间为 p_{ij}, 优化目标是确定 n 个工件在某项指标下的最优调度方案, 并且要求所有的工件以相同顺序在 m 台机器上进行加工, 一般假设为 $1, 2, \cdots, m$, 同时满足每个工件在每台机器上只加工一次, 每台机器在某个时刻只能加工一个工件这些约束条件.

在研究调度问题中, 调度的质量可以用多个性能指标来衡量. PFSP 问题按照不同的优化指标又可以分为以下几类[1]: 1) 考虑制造周期 makespan($C_{\max}$) 最小化的 PFSP 问题, 即最小化最后一个工件在第 m 台机器上的完工时间, 记为 $F/prmu/C_{\max}$; 2) 总完工时间最小化的 PFSP 问题, 记为 $F/prmu/\sum C_j$, 其中 C_j 为第 j 个工件的完工时间; 3) 总拖期时间最小化的 PFSP 问题, 记为 $F/prmu/\sum T_j$, 其中 T_j 为第 j 个工件的拖期评价指标, 即 $T_j = \max\{C_j - d_j, 0\}$, d_j 为工件 j 的交货时间. 在过去的几十年里, 国内外学者针对 PFSP 问题做了大量深入细致的研究工作, 针对不同的求解目标设计了许多高效的求解算法. 在当前的工业生产活动中, 及时满足客户的订货需求对企业来说已变得尤为重要, 因而与交货时间/拖期相关的评价指标成为近期 PFSP 研究热点. Vallada 等在文献 [2] 中对总拖期时间最小化的 PFSP 问题进行详尽综述并且指出, 只有部分用于求解 Makespan 指标的 PFSP 经典算法可用于求解总拖期时间最小化的 PFSP 问题.

总拖期时间最小化的 PFSP 求解方法包括: 精确算法、构造性启发式算法和元启发式算法三类[3]. 由于此类问题的 NP-hard 性质, 精确算法虽然在理论上能够保证得到最优解, 但其计算时间通常难以接受, 一般只适用于小规模问题. 启发式方法可以快速完成解的构造, 但是解的质量通常较差. 近年, 以模拟退火算法[4]、禁忌搜索算法[5]、遗传算法[6] 和群体智能算法[7] 为代表的元启发式方法相继应用于求解总拖期时间最小化和多目标 PFSP 问题.

多智能体进化算法 (Multi-agent evolutionary algorithm, MAEA) 是受多智能体系统演化过程启发而提出的一种元启发式算法[8]. 近些年, 由于 MAEA 算法在求解各种复杂优化问题中表现出良好的优化特性而受到学者的广泛关注[9−10]. 目前 MAEA 算法已经在数值优化、动态优化、组合优化等问题上得到研究与应用. 然而, MAEA 在总拖期时间最小化的 PFSP 上应用至今尚无研究. 本文结合多智能体系统和进化算法的各自优点, 提出了一种多智能体进化算法. 该算法针对总拖期时间最小化的 PFSP 问题, 采用基于延迟时间排列的邻域搜索策略, 快速生成高质量的新个体并设计了相应的概率更新模型及相关的个体更新机制, 使算法在分散搜索和精细搜索之间达到合理的平衡. 为了获得算法运行最佳参数, 通过实验设计方法对 MAEA 算法的主要参数设置进行实验设计 (Design of experiments, DOE) 分析. 最后, 针对 Vallada 提出的标准测试集进行仿真计算, 其中多个测试问题找到了新的最优解, 证实了该算法的有效性.

1　问题模型

总拖期时间最小化的流水车间调度问题的每个调度方案可用工件集合 $\boldsymbol{N}$ 上的一个排列 $\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_n)$ 来表示. 假设用 Π 代表所有的调度方案, $T(\boldsymbol{\pi})$ 是所有 n 个工件在调

收稿日期 2012-12-12　录用日期 2013-04-19
Manuscript received December 12, 2012; accepted April 19, 2013
国家自然科学基金 (61333006, 71171038), 中央高校基本科研业务费 (N110404024) 资助
Supported by National Natural Science Foundation of China (61333006, 71171038), the Fundamental Research Funds for the Central Universities (N110404024)
本文责任编委 王红卫
Recommended by Associate Editor WANG Hong-Wei
1. 东北大学信息科学与工程学院系统工程研究所 沈阳 110819　2. 流程工业综合自动化教育部重点实验室 沈阳 110819
1. Institute of Systems Engineering, School of Information Science and Engineering, Northeastern University, Shenyang 110819 2. State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819

度方案 $\boldsymbol{\pi}$ 下的总拖期时间, 最优调度方案是使目标函数 $T(\boldsymbol{\pi})$ 最小的排列 $\boldsymbol{\pi}^*$, 即

$$T(\boldsymbol{\pi}^*) = \min_{\pi \in \Pi} T(\boldsymbol{\pi}) \tag{1}$$

其中, $T(\boldsymbol{\pi})$ 的计算公式如下:

$$T(\boldsymbol{\pi}) = \sum_{j=1}^{n} \max(C_{\pi_j m} - d_{\pi_j}, 0) \tag{2}$$

工件 π_j 在机器 i 上的完工时间 $C_{\pi_j i}$ 可通过下面的公式求出:

$$C_{\pi_j i} = \max(C_{\pi_{j-1} i}, C_{\pi_j i-1}) + p_{\pi_j i} \tag{3}$$

并且

$$C_{\pi_1 1} = p_{\pi_1 1} \tag{4}$$

$$C_{\pi_j 1} = C_{\pi_{j-1} 1} + t_{\pi_j 1} \tag{5}$$

$$C_{1\pi_j} = C_{1\pi_{j-1}} + t_{1\pi_j} \tag{6}$$

其中, $j = 2, 3, \cdots, n$; $i = 2, 3, \cdots, m$. 通常来说, 流水车间调度问题的解空间大小为 $(n!)^m$. 但是由于 PFSP 问题要求每个工件在各台机器上的加工顺序相同, 因而只需要考虑 $n!$ 个调度方案.

1.1　求解总拖期时间最小化 PFSP 的多智能体进化算法

多智能体进化算法是一种新型的基于多智能体网格环境下的协同群体进化算法. 一般来说, 多智能系统体包括三个基本概念: 1) 生存环境; 2) 具有信息交互功能的智能体; 3) 每个智能体的生存进化规则. 多智能体系统通过模拟自然界中的鸟类飞行、鱼群捕食、蚂蚁觅食等行为而产生群体智能决策行为来求解现实环境下的复杂优化问题. 本文提出的多智能体搜索算法通过模仿社会组织中个体之间的竞争、协作规则来求解组合优化问题, 整个智能体网格如图 1 所示.

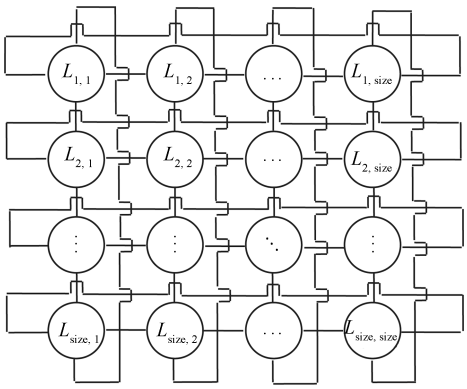

图 1　智能体网络

Fig. 1　Agent lattice

从智能体系统求解问题的进化角度看, 每个智能体相当于进化算法中的个体并且具有局部信息感知、竞争协作和自我学习的能力. 在整个网格系统中, 每个智能体充分保留了分工合作决策的特点去自动求解问题, 并且强调每个智能体的自我进化、自我更新的自治行为而不是按照某个具体的目标进化. 在多智能体进化算法中, 每个智能体代表优化问题搜索空间中的一个候选解, 而网格环境下的所有智能体构成一个种群 (Swarm). 在当前种群中的每个智能体通过竞争的方式与其邻域中的最优智能体 (Localbest) 进行学习、信息交换, 以此来完成其自身和整个智能体系统的进化过程.

1.2　智能体的学习方式

在 MAEA 中, 所有的智能体生存网格定义为 L, 网格大小为 $L_{\text{size}} \times L_{\text{size}}$. 处于第 i 行, 第 j 列的智能体为 L_{ij} $(i, j = 1, 2, \cdots, L_{\text{size}})$, 则智能体 L_{ij} 的邻域为

$$L_{ij}^{\text{neighborhood}} = \{L_{i',j}, L_{i,j'}, L_{i'',j}, L_{i,j''}\} \tag{7}$$

其中

$$i' = \begin{cases} i-1, & i \neq 1 \\ L_{\text{size}}, & i = 1 \end{cases}, \quad j' = \begin{cases} j-1, & j \neq 1 \\ L_{\text{size}}, & j = 1 \end{cases}$$

$$i'' = \begin{cases} i+1, & i \neq L_{\text{size}} \\ 1, & i = L_{\text{size}} \end{cases}, \quad j'' = \begin{cases} j+1, & j \neq L_{\text{size}} \\ 1, & i = L_{\text{size}} \end{cases}$$

设智能体 L_{ij} 的邻域最优个体为 $L_{ij}^{lb} = \{\pi_1^{lb}, \pi_2^{lb}, \cdots, \pi_n^{lb}\}$, 则 $\forall L_{ij} \in L_{ij}^{\text{neighborhood}}, T(L_{ij}) \geq T(_{ij}^{lb})$. 在智能体的竞争学习过程中, 针对总拖期时间最小化的 PFSP 特点, 采用基于拖期时间排序的学习方式 (Tardiness rank based learning, TRBL) 进行智能体的更新演化. TRBL 使每个待更新的智能体向其邻域最优智能体学习, 进而使整个智能体网格朝目标函数最小的方向逐步演化. TRBL 的具体更新策略如下, 假设当前进行学习的智能体为 L_{ij}, 其邻域最优智能体为 L_{ij}^{lb}, 第一阶段, 按照 L_{ij}^{lb} 中工件的加工顺序 $\{\pi_1^{lb}, \pi_2^{lb}, \cdots, \pi_n^{lb}\}$ 评估 L_{ij}^{lb} 中每个工件的拖期时间 $(T_j = C_j - d_j)$. 如果某个工件的拖期时间为正, 则说明此工件的完工时间晚于交货时间, 如果为负, 则说明此工件的完工时间早于交货时间. 从每个智能体的竞争学习角度看, 邻域最优智能体中拖期时间最小的工件集合对于当前智能体的学习过程具有重要的信息指导作用. 设 L_{ij}^{lb} 中拖期时间最小工件集合为 $\prod_{ij}^{m} = \{\pi_{m_1}^{lb}, \pi_{m_2}^{lb}, \cdots, \pi_{m_k}^{lb}\}\,(k < n)$. 将 $\prod_{ij}^{m}$ 中所有工件从智能体 L_{ij} 选出, 则剩余工件集合定义为 $\prod_{ij}^{r}$. 第二阶段, 将集合 $\prod_{ij}^{m}$ 中的工件逐一插入到集合 $\prod_{ij}^{r}$ 中, 并且确保插入后总拖期时间最小化, 此过程反复进行, 直到集合 $\prod_{ij}^{m}$ 为空. 每个待更新的智能体经过 TRBL 学习以后, 将以极高的概率搜索到高质量的新解. 为了说明 TRBL 学习过程的有效性, 下面给出一个 20 个工件 5 台机器的实例 (Taillard 001).

假设当前准备进行 TRBL 学习的智能体 $L_{22} = \{6, 16, 4, 12, 5, 15, 13, 1, 10, 8, 19, 14, 18, 11, 17, 9, 20, 2, 7, 3\}$, 经过计算得出其总拖期时间为 7 113 单位时间. 按照 TRBL 方式进行学习时, 其邻域感知范围 $L_{22}^{\text{neighbourhood}} = \{L_{12}, L_{21}, L_{23}, L_{32}\}$, 则 $L_{22}^{lb} = L_{21}$, 见表 1. 邻域最优智能体 L_{21} 所对应的各工件的拖期时间如表 2 所示.

智能体 L_{22} 采用 TRBL 进行学习时, 设拖期时间最小工件集合的大小 $k = 4$, 则 $\prod_{22}^{m} = \{18, 6, 15, 19\}$, $\prod_{22}^{r} = \{16, 4, 12, 5, 13, 1, 10, 8, 14, 11, 17, 9, 20, 2, 7, 3\}$. 智能体 L_{22} 在第二阶段逐步插入集合 $\prod_{22}^{m}$ 中各个工件之后, 各阶段所对应的解如下:

$L_{22}^{(1)} = \{16, 4, 12, 5, 13, 1, 10, 8, 14, 11, 17, 9, 20, 2, 18, 7, 3\}$, Total tardiness = 4 483;

$L_{22}^{(2)} = \{6, 16, 4, 12, 5, 13, 1, 10, 8, 14, 11, 17, 9, 20, 2, 18, 7, 3\}$, Total tardiness = 4 954;

$L_{22}^{(3)} = \{15, 6, 16, 4, 12, 5, 13, 1, 10, 8, 14, 11, 17, 9, 20, 2, 18, 7, 3\}$, Total tardiness = 5 125;

$L_{22}^{(4)} = \{15, 19, 6, 16, 4, 12, 5, 13, 1, 10, 8, 14, 11, 17, 9, 20, 2, 18, 7, 3\}$, Total tardiness = 6 053;

表 1 智能体 L_{22} 的邻域

Table 1 Neighbourhood of agent L_{22}

智能体 (L_{ij})	Total tardiness
$L_{12}=\{12,20,13,1,15,6,9,8,18,3,4,10,14,19,7,5,17,11,2\}$	7 824
$L_{21}=\{19,6,8,2,12,10,18,14,7,15,1,16,11,5,9,17,20,13,3,4\}$	6 088
$L_{23}=\{12,20,13,1,15,6,9,8,18,3,4,10,14,19,7,5,17,11,2\}$	6 582
$L_{32}=\{6,13,17,9,5,18,2,3,11,4,14,10,19,12,7,1,20,8,16,15\}$	6 760

表 2 智能体 L_{21} 中各工件的拖期时间

Table 2 Tardiness values of jobs according to agent L_{21}

Job (j)	Completion time (C_j)	Due date (d_j)	Tardiness (C_j-d_j)
19	269	334	−65
6	345	690	−345
8	410	289	121
2	483	325	158
12	555	268	287
10	621	342	279
18	746	1 205	−459
14	818	646	172
7	876	602	274
15	976	1 111	−135
1	1 053	468	585
16	1 140	965	175
11	1 226	764	462
5	1 279	1 070	209
9	1 348	873	475
17	1 406	703	703
20	1 434	1 111	323
13	1 442	1 158	284
3	1 462	923	539
4	1 555	513	1 042

1.3 智能体网络概率更新模型

MAEA 算法通过概率更新模型决定是否接受经过 TRBL 学习后产生的新智能体. 为了防止所有的智能体过早陷入早熟状态以及最大限度保证智能体网络的多样性, 适应性差的智能体将以一定的概率替换掉原有的智能体继续参与整个智能体网格的进化. MAEA 采用的概率更新模型类似于恒温机制下的模拟退火算法[11−12], 并且恒温参数的选取决定于特定测试问题中的加工时间, 其计算公式如下:

$$Temp=\frac{T\sum_{i=1}^{m}\sum_{j=1}^{n}p_{ij}}{n\times m\times 10} \tag{8}$$

其中, T 是自定义的调节参数. 当智能体 L_{ij} 经过 TRBL 学习后新产生的智能体为 $L_{ij}^{''}$, 如果 TotalTardiness ($L_{ij}^{''}$) > TotalTardiness(L_{ij}), 并且随机数 Random $\in(0,1)\le \mathrm{e}^{(\mathrm{TotalTardiness(L_{ij})-TotalTardiness(L_{ij}^{''})})/\mathrm{Temp}}$, 则用 $L_{ij}^{''}$ 替换掉原来的解 L_{ij}. MAEA 算法的伪代码见附录.

2 仿真测试及比较分析

2.1 算法参数设置

本文提出的 MAEA 算法有 3 个参数需要设置: 网格大小 L_{size}, 拖期时间最小工件集合大小 k 和温度参数 T. 为了确定合适的参数, 我们利用 Taillard 标准测试集中工件数 $n=50$, 机器数 $m=10$ 中的实例 (Taillard 041) 采用实验设计的方法 (DOE)[13] 讨论参数对算法性能的影响. 算法在每种参数设置下独立运行 10 次, 各参数取 4 种水平, 见表 3. 设置算法的运行时间为 $(n\cdot m/2\cdot 100)$ ms 作为终止条件. 10 次运行的平均值 AVG (Average) 作为评价指标. 选择规模为 $L_{16}(4^3)$ 的正交实验, 正交表和计算的平均值 AVG 如表 4 所示. 算法的参数极差及影响程度等级见表 5, 其中各参数对算法性能的影响如图 2 和 3 所示.

表 3 参数水平

Table 3 Parameter levels

参数	水平			
	1	2	3	4
L_{size}	5	10	15	20
k	2	4	6	8
T	0.1	0.2	0.4	0.8

从表 5 可见, 参数 k 对 MAEA 算法影响最大, 而温度参数 T 对算法影响最小. 过大或过小的参数 k 都会导致算法性能下降. 影响算法性能的另一参数是网格大小 L_{size}. 在限制

表 4　正交表和 AVG 统计值

Table 4　Orthogonal table and average statistical value

实验序列	水平			
	L_{size}	k	T	平均值
1	1	1	1	11 730.6
2	1	2	2	11 327.6
3	1	3	3	11 015.9
4	1	4	4	11 008.3
5	2	1	2	11 369.2
6	2	2	1	11 020.9
7	2	3	4	10 888.0
8	2	4	3	10 840.2
9	3	1	3	11 310.7
10	3	2	4	10 888.0
11	3	3	1	10 653.1
12	3	4	2	10 807.1
13	4	1	4	11 314.4
14	4	2	3	10 865.2
15	4	3	2	10 755.6
16	4	4	1	10 923.5

表 5　均值的响应值

Table 5　Response values

等级	L_{size}	k	T
1	11 271	11 431	11 082
2	11 030	11 025	11 065
3	10 915	10 828	11 008
4	10 965	10 895	11 025
δ	356	603	74
Rank	2	1	3

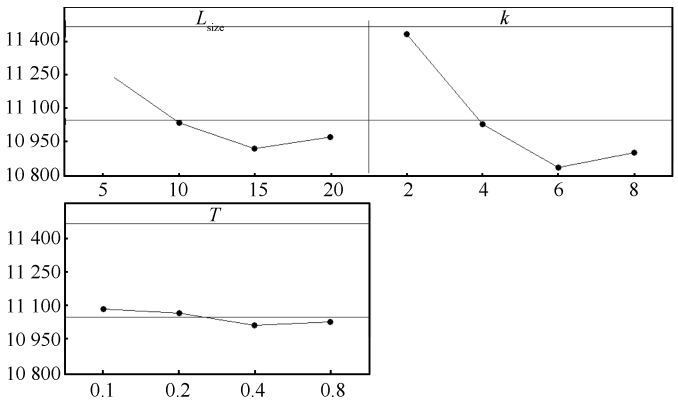

图 2　主效应方差分析

Fig. 2　Main effect plots of analysis of variance (ANOVA) analysis

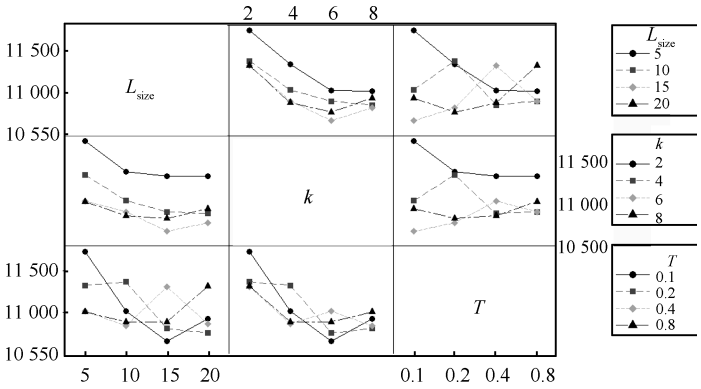

图 3 交叉效应方差分析

Fig. 3 Interaction plots of analysis of variance (ANOVA) analysis

计算时间后, 过大的 L_{size} 会导致算法迭代次数的减少而丧失其深度搜索的能力, 过小则不能保证对问题采样空间的有效探索. 综上分析, 设置 MAEA 算法的参数为: $L_{\text{size}} = 15$, $k = 6$, $T = 0.4$.

2.2 实验计算及比较

为了验证本文提出的 MAEA 算法, 选取了 Vallada 等提出的标准测试问题进行计算. 这些标准测试问题涵盖了工件数从 50 变化到 350 ($n = \{50, 150, 250, 350\}$), 机器数从 10 变化到 50 ($m = \{10, 30, 50\}$) 一共 12 种规模的测试问题, 并且每个测试问题根据两个不同参数: 拖期因子 T (Tardiness factor) 和交货周期范围 R (Due date range) 随机生成 5 个测试数据文件. 拖期因子 T 取值范围 $T = \{0.2, 0.4, 0.6\}$, 交货期 R 的范围为 $R = \{0.2, 0.6, 1.0\}$. 例如测试文件 "I_0, 2_0, 6_50_10_3" 表示在拖期因子 $T = 0.2$, 交货期范围 $R = 0.6$ 条件下, 工件数为 50, 机器数为 10 的第三个测试数据. 所有测试问题的优化值和最差值可以从网站 http://soa.iti.es/problem-instances 下载得到. MAEA 算法采用 C++ 编写, 每个测试问题独立运行 5 次, 运行环境为: Core 2 Duo 2.66 GHz CPU, 2 GB RAM, 停止准则设置为最大运行时间: $(n \cdot m/2 \cdot 90)$ ms. 其计算结果见表 6 ∼ 8.

为了比较 MAEA 算法与其他算法的性能, 分别编写了 Ruiz 等提出的混合遗传算法 (Hybrid genetic algorithm, HGA)[14], Parthasarathy 等和 Hasija 等提出的模拟退火算法 (SAH, SAP)[15−16], Vallada 等提出带有 path-relinking 的遗传算法 (Genetic algorithm with path relinking, GAPR)[17], Ruiz 等提出的迭代贪婪算法 (Iterated greedy, IG)[18] 进行对比. 评价算法的性能指标采用 Zemel 和 Kim 等[19−21] 提出的相对偏差指数 (Relative deviation index, RDI), 其计算公式如下:

$$\text{RDI} = \frac{\text{Method}_{sol} - \text{Best}}{\text{Worst} - \text{Best}} \times 100 \tag{9}$$

其中, Method_{sol} 是采用某种算法多次运行计算所得的平均值, Best 和 Worst 是已经求解出的最优解和计算过程中最差解. 最后, 将每个 $n \times m$ 组内的 45 个计算结果取均值, 其计算结果见表 9.

由表 6 ∼ 表 8 可见, 本文提出的 MAEA 算法表现出了较好的寻优特性, 特别是当拖期因子 $T = 0.2$ 时, 能搜索到多个测试问题的最优值. TRBL 的学习过程依据插入邻域生成准则, 但首先选择邻域最优智能体中拖期时间最小的工件对当前的智能体进行插入测试, 此种方式能够以较小的计算代价快速进行邻域搜索从而获得高质量的新个体.

表 6 拖期因子 $T = 0.2$ 时测试问题的均值

Table 6 Average computation results for tardiness factor $T = 0.2$

问题	$R = 0.2$			$R = 0.6$			$R = 1.0$		
	最优值	最差值	MAEA	最优值	最差值	MAEA	最优值	最差值	MAEA
050 × 10	2 424.60	19 165.8	2 650.08	0	14 453.2	**0**	0	17 321.0	**0**
050 × 30	19 356.8	50 290.8	20 134.88	16 011.4	44 673	16 724.77	13 985.8	45 397.4	14 305.32
050 × 50	35 620.5	70 388.4	36 519.44	36 863.2	72 768.2	37 788.88	35 591.6	71 118.2	36 238.16
150 × 10	11 763.4	95 980	16 501.24	0	84 216.2	142	0	108 932	**0**
150 × 30	48 447.8	187 340.6	70 860.92	1 742.8	170 785.6	8 887.64	0	206203.6	355.72
150 × 50	99 818.4	263 938.8	118 900.3	52 574.4	290 669	69 325.08	21 133.4	299704.8	28 660.12
250 × 10	23 649.8	200 645	40 297.72	0	209 435.8	**0**	0	287 589.2	**0**
250 × 30	85 109.4	368 803.4	152 116.44	0	391 810.8	16 555.0	0	446 483.2	**0**
250 × 50	177 736.2	518 253.6	275 253.72	40 064.8	570 420.8	113 873.08	0	633 137.2	4 056.28
350 × 10	43 420.8	382 878.6	82 262.96	0	358 705.4	**0**	0	503 362.4	**0**
350 × 30	135 042.2	636 625.2	282 848.12	0	620 397.8	17 095.36	0	760 121.8	**0**
350 × 50	256 322.4	861 901.4	400 706.0	17 048	836 437	90 292.6	0	1 002 993.8	**0.08**

表 7　拖期因子 $T = 0.4$ 时测试问题的均值

Table 7　Average computation results for tardiness factor $T = 0.4$

问题	$R = 0.2$			$R = 0.6$			$R = 1.0$		
	最优值	最差值	MAEA	最优值	最差值	MAEA	最优值	最差值	MAEA
050 × 10	13 430.8	37 860.8	13 876.28	10 587.2	33 207	10 982.36	9 149.2	35 233.4	9 448.28
050 × 30	46 117.6	78 935.8	47 060.0	43 702.6	75 257.2	44 458.44	48 811.4	80 504.2	49 490.84
050 × 50	76 495.4	117 107.2	77 672.08	79 209.6	114 860	80 268.24	78 640.6	114 670.8	79 723.68
150 × 10	74 675.4	221 543	89 806.04	24 822	179 540.8	35 057.12	10 821.6	225 849.6	17 462.12
150 × 30	172 459.6	355 068.6	208 632.8	139 950.4	334 417	172 561.8	135 110.8	373 024.4	168 580.0
150 × 50	289 478.2	493 484.6	317 623.4	261 557.8	471 899	290 444.12	240 930.2	477 893.2	268 781.8
250 × 10	183 950.8	520 188.8	243 565.56	61 664.8	463 212.2	98 671.68	97.4	550 258.8	1 820.88
250 × 30	356 729	732 248.4	466 283.56	233 337.8	766 904.8	344 285.36	122 649.8	794 561.2	214 326.6
250 × 50	561 873.2	971 524.4	702 226.24	458 275.8	1 012 587.2	598 246.2	411 175.2	1 044 721.8	569 355.2
350 × 10	353 194	968 476.6	481 778.48	98 941.8	871 273.6	183 710.48	339.2	1 093 749	9 584.64
350 × 30	637 846	1 286 920.2	880 560.84	332 286.4	1 278 054.6	547 167.2	134 553	1 456 916.8	331 849.76
350 × 50	923 234.2	1 669 820.8	1 146 019.6	653 945.8	1 705 427.4	862 954.8	448 715.2	1 748 009.4	666 831.44

表 8　拖期因子 $T = 0.6$ 时测试问题的均值

Table 8　Average computation results for tardiness factor $T = 0.6$

问题	$R = 0.2$			$R = 0.6$			$R = 1.0$		
	最优值	最差值	MAEA	最优值	最差值	MAEA	最优值	最差值	MAEA
050 × 10	32 483.2	62 234.4	33 162.6	34 097.2	61 448.6	34 684.40	22 934.4	47 735.8	23 509.76
050 × 30	78 954.6	116 744	80 057.48	80 781	1 164 35.8	81 621.32	71 503.4	104 610.2	72 377.32
050 × 50	124 722	169 730.6	125 918.04	129 228.2	170 632	130 231.48	107 246.8	144 350.8	108 103.04
150 × 10	199 752.2	377 114	227 961.36	173 233	349 482.6	202 699.5	101 358.2	301 255.8	120 240.8
150 × 30	356 107.2	552 072.6	395 332.56	358 446.2	552 319.6	398 417.72	305 508	512 570.4	348 691.12
150 × 50	526 596.4	751 100.6	557 232.8	519 977.6	733 149	551 237.56	471 871	683 044.8	503 445.72
250 × 10	504 065.2	953 667.8	626 758.96	366 193.8	850 445.4	460 823.08	233 115.6	802 915.6	319 723.28
250 × 30	825 683.2	1 289 114.8	994 550.32	815 991	1 303 602.4	985 848.92	599 164	1 136 824.8	753 695.8
250 × 50	1 109 572.6	1 591 398.2	1 297 100.6	1 114 187.4	1 588 867	1 296 731.4	888 260.6	1 384 607.6	1065384.8
350 × 10	967 597.6	1 718 083.6	12 000 599.6	744 426.8	1 661 490.25	1 014 083.7	474 323	1 597 448	711 886.72
350 × 30	1 436 852.8	2 201 877.8	1 770 826.8	1 381 130.6	2 254 002.2	1 729 635.4	992 405	2 019 712.6	1 323 750.4
350 × 50	1 909 906.6	2 730 538.8	2 226 762.6	1 850 051	2 714 659.8	2 158 717.4	1 424 312	2 435 983	1 704 504.4

表 9　对比算法的平均相对偏差指数

Table 9　Average relative deviation index for the evaluated methods

问题	HGA	SAH	SAP	GAPR	IG	MAEA
050 × 10	10.33	40.75	30.48	11.61	8.44	**1.42**
050 × 30	13.47	57.42	34.49	12.34	12.28	**2.38**
050 × 50	15.39	57.24	35.75	12.87	13.33	**2.59**
150 × 10	37.71	32.36	44.07	17.33	21.89	**7.66**
150 × 30	39.45	49.57	48.15	18.13	24.85	**15.53**
150 × 50	40.34	62.50	57.27	19.17	27.72	**11.57**
250 × 10	43.64	17.35	31.05	15.91	18.09	**10.95**
250 × 30	49.19	30.72	44.58	**20.13**	26.87	21.37
250 × 50	50.62	43.99	53.21	**22.20**	28.84	26.82
350 × 10	35.78	**7.65**	16.70	16.58	15.62	13.96
350 × 30	39.24	14.36	23.40	**17.35**	22.51	24.85
350 × 50	40.04	24.05	31.13	**18.39**	23.15	22.41
平均	34.60	36.50	37.52	16.83	20.30	**13.46**

在与其他算法对比中发现, GAPR 和 IG 也表现出不错的优化效果, 尤其是 GAPR 针对 350 × 30、350 × 50 测试问题的优化效果好于其他算法. MAEA 算法虽然在 250 工件以下的优化过程中效果突出, 但针对 250 × 50 测试问题优化效果均不如 GAPR 和 IG 算法. HGA、SAH 和 SAP 算法的平均 RDI 值均高于其他算法, 表明这三种算法均不适于求解 Total tardiness PFSP 问题, 但比较特殊的是模拟退火算法 SAH 在 350 × 10 测试中均优于其他算法. 导致这一结果主要原因可能是文献 [14] 中提到的 HGA、SAH 和 SAP 算法是用来求解其他指标的 PFSP 问题, 而并没有充分考虑 Total tardiness 问题特性. 从综合统计数据来看, MAEA 算法优化效果令人满意.

为了进一步验证 MAEA 算法在求解 Total tardiness PFSP 问题中具有更强的鲁棒性和适应性, 本文选取 t 检验的方法 (显著性水平 $\alpha = 0.05$) 对 GAPR、IG、MAEA 三种算法进行两两检验. 测试数据选取 $n = \{50, 350\}$, $m = \{10, 50\}$, $T = \{0.2, 0.6\}$, $R = \{0.2, 0.6, 1.0\}$ 4 种参数配置下的第一个数据文件进行测算. 对于每个测试问题, 每种算法独立运行 30 次. 算法停止准则设置为最大运行时间: $(n \cdot m/2 \cdot 200)$ ms. 表 10 列出了三种算法的检验结果. 其中 "s+" 表示前种算法显著优于后一种算法, "s−" 表示结果相反, "+" 表示前种算法非显著优于后者算法, "−" 表示前种

算法非显著差于后者算法.

表 10 三种算法的 t 检验结果

Table 10 The t-test results of comparing algorithms regarding the RDI value

问题 ($n \times m$)	t 检验结果		
050 × 10, $T = 0.2$	$R = 0.2$	$R = 0.6$	$R = 1.0$
MAEA-GAPR	$s+$	$s+$	$s+$
GAPR-IG	$+$	$-$	$-$
MAEA-IG	$s+$	$s+$	$s+$
050 × 10, $T = 0.6$	$R = 0.2$	$R = 0.6$	$R = 1.0$
MAEA-GAPR	$s+$	$s+$	$s+$
GAPR-IG	$+$	$-$	$-$
MAEA-IG	$s+$	$s+$	$s+$
050 × 50, $T = 0.2$	$R = 0.2$	$R = 0.6$	$R = 1.0$
MAEA-GAPR	$s+$	$s+$	$s+$
GAPR-IG	$+$	$s+$	$+$
MAEA-IG	$s+$	$s+$	$s+$
050 × 50, $T = 0.6$	$R = 0.2$	$R = 0.6$	$R = 1.0$
MAEA-GAPR	$s+$	$s+$	$s+$
GAPR-IG	$+$	$+$	$-$
MAEA-IG	$s+$	$s+$	$s+$
350 × 10, $T = 0.2$	$R = 0.2$	$R = 0.6$	$R = 1.0$
MAEA-GAPR	$s+$	$s+$	$s+$
GAPR-IG	$-$	$-$	$+$
MAEA-IG	$s+$	$s+$	$+$
350 × 10, $T = 0.6$	$R = 0.2$	$R = 0.6$	$R = 1.0$
MAEA-GAPR	$+$	$s+$	$s+$
GAPR-IG	$+$	$-$	$-$
MAEA-IG	$s+$	$+$	$s+$
350 × 50, $T = 0.2$	$R = 0.2$	$R = 0.6$	$R = 1.0$
MAEA-GAPR	$+$	$+$	$-$
GAPR-IG	$s+$	$s+$	$+$
MAEA-IG	$-$	$+$	$-$
350 × 50, $T = 0.6$	$R = 0.2$	$R = 0.6$	$R = 1.0$
MAEA-GAPR	$-$	$-$	$s-$
GAPR-IG	$s+$	$s+$	$+$
MAEA-IG	$+$	$+$	$+$

此外, 我们还比较分析了当前求解 Total tardiness PFSP 问题的几种最好算法的收敛特性. 为了与新近提出的群体智能算法做对比, 我们引入了文献 [7] 中提出的 Hybrid-PSO (Particle swarm optimization) 算法进行测试. 对比算法 GAPR、IG、Hybrid-PSO 均引用各自文献中推荐的参数设置, 只是将 Hybrid-PSO 算法的种群大小设置为 $popsize = 100$, 并且所有算法的运行时间设置为 $(n \cdot m \cdot 100)$ ms. 测试结果见图 4 和 5. 从图 4 可以看到, 在工件数 $n = 50$ 情况下, MAEA 的收敛速度均快于其他对比算法, 并且算法运行后期始终能够搜索到新解. Hybrid-PSO 收敛速度最慢, 这可能由于适于求解连续空间优化问题的 PSO 算法不太适合求解离散空间的调度问题. IG 和 GAPR 的搜索性能相当, 但在算法运行后期 GAPR 的性能要好于 IG. 当工件数 $n = 350$ 时, PSO 的收敛速度最慢, GAPR 与 MAEA 的搜索性能相当, 但均优于 IG 算法.

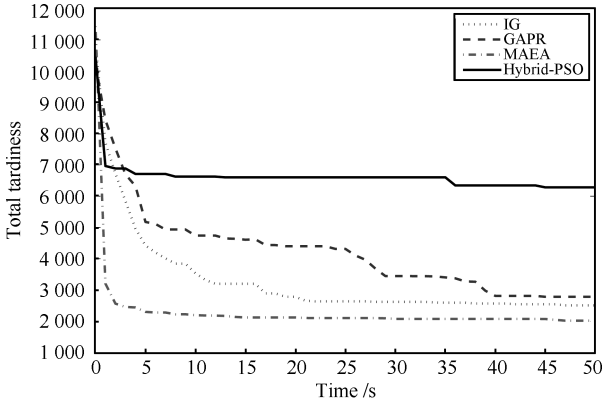

图 4 数据 $'I\_0, 2\_0, 2\_50\_10\_1'$ 收敛曲线

Fig. 4 The convergence curves of instance $'I\_0, 2\_0, 2\_50\_10\_1'$

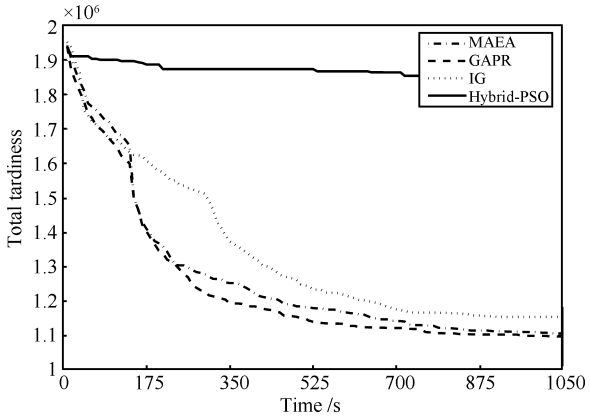

图 5 数据 $'I\_0, 6\_1\_350\_30\_1'$ 收敛曲线

Fig. 5 The convergence curves of instance $'I\_0, 6\_1\_350\_30\_1'$

3 结论

本文针对总拖期时间最小化的 PFSP 问题提出了一种 MAEA 算法进行求解. 通过引入基于 TRBL 学习机制的智能体学习策略与智能体概率更新模型完成整个智能体网格的更新进化并通过实验设计方法评估了不同的算法参数设置对算法求解质量的影响. 通过对 540 个标准测试问题的数值计算和与其他代表算法的比较分析验证了 MAEA 算法的有效性. 今后进一步的研究方向是通过结合问题特性的 MAEA 算法求解其他复杂的组合优化问题.

附录 Vallada 标准测试集中新解

New best solutions for Vallada benchmark instances

1. New best solution for $'I\_0, 2\_0, 2\_50\_10\_1'$ $n = 50, m = 10$, Total tardiness = 1 865

2. New best solution for $'I\_0, 4\_0, 2\_50\_10\_1'$ $n = 50, m = 10$, Total tardiness = 12 338

3. New best solution for $'I\_0, 4\_0, 2\_50\_50\_1'$ $n = 50, m = 50$, Total tardiness = 78 475

4. New best solution for $'I\_0, 6\_0, 6\_50\_30\_3'$ $n = 50, m = 30$, Total tardiness = 80 975

5. New best solution for $'I\_0, 4\_0, 2\_150\_10\_1'$

$n = 150, m = 10$, Total tardiness = 74 597
6. New best solution for $'I\_0, 4\_0, 2\_150\_10\_2'$
$n = 150, m = 10$, Total tardiness = 68 040
7. New best solution for $'I\_0, 4\_0, 2\_150\_10\_4'$
$n = 150, m = 10$, Total tardiness = 83 737
8. New best solution for $'I\_0, 4\_0, 6\_150\_10\_2'$
$n = 150, m = 10$, Total tardiness = 23 344
9. New best solution for $'I\_0, 4\_1\_150\_10\_1'$
$n = 150, m = 10$, Total tardiness = 3 748
10. New best solution for $'I\_0, 4\_1\_150\_10\_2'$
$n = 150, m = 10$, Total tardiness = 11 513

References

1 Vallada E, Rubén R. Cooperative metaheuristics for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 2009, **193**(2): 365−376

2 Vallada E, Rubén R, Gerardo M. Minimising total tardiness in the *m*-machine flowshop problem: a review and evaluation of heuristics and metaheuristics. *Computers & Operations Research*, 2008, **35**(4): 1350−1373

3 Liao J M, Huang C J. Tabu search for non-permutation flowshop scheduling problem with minimizing total tardiness. *Applied Mathematics and Computation*, 2010, **217**(2): 557−567

4 Hasija S, Rajendran C. Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research*, 2004, **42**(11): 2289−2301

5 Armentano V A, Ronconi D P. Tabu search for total tardiness minimization in flowshop scheduling problems. *Computers & Operations Research*, 1999, **26**(3): 219−235

6 Talip K, Bilal T, John W. Elite guided steady-state genetic algorithm for minimizing total tardiness in flowshops. *Computers & Industrial Engineering*, 2010, **58**(2): 300−306

7 Li B B, Wang L, Liu B. An effective PSO-based hybrid algorithm for multi-objective permutation flow shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics — Part A: Systems and Humans*, 2008, **38**(4): 818−831

8 Jiao Li-Cheng, Liu Jing, Zhong Wei-Cai, *Coevolutionary Computation and Multiagent Systems*. Beijng: Science Press, 2006. 205−224
(焦李成, 刘静, 钟伟才. 协同进化计算与多智能体系统. 北京: 科学出版社, 2006. 205−224)

9 Sarker R A, Ray T. *Agent-Based Evolutionary Search*. Berlin: Springer-Verlag, 2010. 97−116

10 Zhong Wei-Cai, Liu Jing, Jiao Li-Cheng. Optimal approximation of linear systems by multi-agent genetic algorithm. *Acta Automatica Sinica*, 2004, **30**(6): 933−938
(钟伟才, 刘静, 焦李成. 多智能体遗传算法用于线性系统逼近. 自动化学报, 2004, **30**(6): 933−938)

11 Stützle T. Applying iterated local search to the permutation flow shop problem, Technical Report, AIDA-98-04, FG Intellektik, FB Informatik, TU Darmstadt, 1998

12 Osman I, Potts C. Simulated annealing for permutation flowshop scheduling. *Omega*, 1989, **17**(6): 551−557

13 Montgomery D C. *Design and Analysis of Experiments* (5th Edition). Hoblken: John Wiley and Sons, 2000

14 Ruiz R, Maroto C, Alcaraz J. Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, 2006, **34**(5): 461−476

15 Parthasarathy S, Rajendran C. A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence-dependent setup times of jobs — a case study. *Production Planning and Control*, 1997, **8**(5): 475−483

16 Hasija S, Rajendran C. Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research*, 2004, **42**(11): 2289−2301

17 Vallada E, Rubén R. Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. *Omega*, 2010, **38**(1−2): 57−67

18 Ruiz R, Stützle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 2007, **177**(3): 2033−2049

19 Zemel E. Measuring the quality of approximate solutions to zero-one programming problems. *Mathematics of Operations Research*, 1981, **6**(3): 319−332

20 Kim Y D. Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal of the Operational Research Society*, 1993, **44**(1): 19−28

21 Kim Y D, Lim H G, Park M W. Search heuristics for a flow shop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research*, 1996, **91**(1): 124−143

王大志 东北大学信息科学与工程学院系统工程研究所讲师. 主要研究方向为智能优化方法, 生产计划与调度. 本文通信作者.
E-mail: wangdazhi1@ise.neu.edu.cn
(**WANG Da-Zhi** Lecturer at the Institute of Systems Engineering, School of Information Science and Engineering, Northeastern University. His research interest covers intelligent optimization methods and production scheduling. Corresponding author of this paper.)

刘士新 东北大学信息科学与工程学院系统工程研究所教授. 主要研究方向为项目管理, 生产计划与调度, 最优化理论与应用.
E-mail: sxliu@mail.neu.edu.cn
(**LIU Shi-Xin** Professor at the the Institute of Systems Engineering, School of Information Science and Engineering, Northeastern University. His research interest covers project management, production planning, and scheduling and optimization theory and applications.)

郭希旺 东北大学系统工程专业博士研究生. 主要研究方向为智能优化方法与生产调度. E-mail: x.w.guo@163.com
(**GUO Xi-Wang** Ph. D. candidate at the Institute of Systems Engineering, School of Information Science and Engineering, Northeastern University. His research interest covers intelligent optimization and production scheduling.)