混合离散教与学算法求解复杂并行机调度问题

何雨洁 [1] 钱 斌 [1] 胡 蓉 [1]

摘 要 针对制造行业中广泛存在的一类复杂并行机调度问题, 即带到达时间、多工序、加工约束和序相关设置时间的并行机调度问题 (Parallel machine scheduling problem with arrival time, multiple operations, process restraints and sequence-dependent setup times, PMSP_AMPS), 建立问题的排序模型并提出一种混合离散教与学优化算法进行求解, 优化目标为最小化最大完工时间. 首先, 根据标准教与学算法 (Teaching-learning-based optimization, TLBO) 中两阶段个体更新公式的特点, 在保留每一阶段个体更新公式框架不变的前提下, 对公式中具体改变实数个体或向量的每个核心操作均用所设计的排列操作进行替换, 使其可直接在离散问题解空间中执行基于标准教与学算法机理的全局搜索, 从而明显提高了原算法的全局搜索效率. 其次, 采用交换操作和插入操作构造了一种简洁有效地变邻域局部搜索, 对全局搜索发现的优质解区域进行细致搜索, 从而进一步增强了算法的性能. 通过对不同测试问题的仿真实验和算法比较, 验证了所提算法可有效求解 PMSP_AMPS.

关键词 并行机调度, 多工序, 序相关设置时间, 到达时间, 离散教与学

引用格式 何雨洁, 钱斌, 胡蓉. 混合离散教与学算法求解复杂并行机调度问题. 自动化学报, 2020, **46**(4): 805−819

DOI 10.16383/j.aas.c180321

Hybrid Discrete Teaching-learning-based Optimization Algorithm for Solving Complex Parallel Machine Scheduling Problem

HE Yu-Jie[1] QIAN Bin[1] HU Rong[1]

Abstract This paper proposes a hybrid discrete teaching-learning-based optimization algorithm and set up the scheduling model for solving the parallel machine scheduling problem with arrival time, multiple operations, process restraints, and sequence-dependent setup times (PMSP_AMPS), which widely exists in the manufacturing process. The criterion is to minimize the maximum completion time. Firstly, according to the characteristics of the two stage individual updating formulas in the standard teaching-learning-based optimization (TLBO) algorithm, under the premise of keeping the framework of individual updating formulas at each stage unchanged, each core operation that specifically changes the real number or vector in the formula is replaced with proposed arrangement operation, so that it can execute the global search directly in the discrete solution space based on the TLBO mechanism, which significantly improve the TLBO′s global exploration ability. Secondly, a simple but effective variable-neighborhood local search is constructed by using interchange and insert operations, which can perform a detailed search of the high-quality solution regions found by the global search, thereby further enhancing the performance of the algorithm. Simulation experiments and comparisons on different instances demonstrate that the proposed algorithm can effectively solve PMSP_AMPS.

Key words Parallel machine scheduling problem (PMSP), multiple operations, sequence-dependent setup times, arrival times, discrete teaching-learning-based optimization

Citation He Yu-Jie, Qian-Bin, Hu Rong. Hybrid discrete teaching-learning-based optimization algorithm for solving complex parallel machine scheduling problem. *Acta Automatica Sinica*, 2020, **46**(4): 805−819

智能制造是解决我国制造业由大变强的根本路径, 有效地生产调度优化算法是提高企业效率和竞争力的重要途径, 属于智能制造的重要研究领域. 并行机调度问题 (Parallel machine scheduling problem, PMSP) 是制造行业中的一类典型生产调度问题. 该类问题不仅需要确定每个工件的加工机器, 还需要确定每台机器上相应工件的加工顺序[1]. Allahverdi[2] 将并行机分为三个类别: 相同并行机, 均匀平行机和不相关并行机. 在实际生产制造或加工过程中, 许多问题可抽象为带不同约束的上述三类并行机调度问题. 其中, 带到达时间、多工序、加工约束和序相关设置时间的并行机调度问题 (Parallel machine scheduling problem with arrival time, multiple operations, process restraints and sequence-dependent setup

收稿日期 2018-05-18 录用日期 2018-08-14
Manuscript received May 18, 2018; accepted August 14, 2018
国家自然科学基金 (51665025, 61963022), 云南省自然科学基金项目 (2015FB136) 资助
Supported by National Natural Science Foundation of China (51665025, 61963022), Applied Basic Research Foundation of Yunnan Province (2015FB136)
本文责任编委 阳春华
Recommended by Associate Editor YANG Chun-Hua
1. 昆明理工大学信息工程与自动化学院 昆明 650500
1. School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500

times, PMSP_AMPS), 是模具制造[3]、半导体芯片终端测试阶段[4]、塑料加工[5] 等行业中广泛存在的一类复杂并行机调度问题, 属于带不同约束的不相关并行机调度问题. 显然, 在计算复杂度上, 并行机问题已被证明为 NP-Complete 问题[6], 而该问题又归约为 PMSP_AMPS, 故 PMSP_AMPS 属于 NP-Complete 问题. PMSP_AMPS 是不相关并行机中非常复杂的一类, 很多其他并行机调度问题均可归约为该类问题. 因此, 研究求解 PMSP_AMPS 的有效算法具有较大的工程意义和理论价值.

并行机调度问题可建立两类问题模型. 一类是 0-1 数学规划模型, 由目标函数和多条显式约束组成, 其对应的求解算法主要为分支定界、列生成[7−10] 等运筹学算法. 该算法可在几分钟至几十分钟内求解较小规模问题 (工件数小于等于 30), 同时经过合适设计, 也可用于求解较大规模问题 (工件数大于等于 100), 但往往求解时间较长. 由于问题具有非凸特性, 目前尚无通用的最优解多项式时间求解算法, 故求解质量取决于具体问题的复杂度以及算法设计者对该问题结构的理解程度, 算法要在较短时间获取满意解的难度较大. 另一类是排序模型, 该类模型相对数学规划模型来说则较为简单, 由几个计算各工件在不同机器上的开始加工时间的式子组成, 约束隐式包含在问题解的排序编码中, 其对应的求解算法主要为基于动态规划的近似算法[11−13] 和各种智能优化算法. 基于动态规划的近似算法需要所求解的问题能提出可构造状态递推方程的、确保最优解不丢失的规则方可使用. 这使其应用主要集中在可行状态较少、能提出相应规则的单机、并行机等调度问题上, 优点在于可确保算法获得问题的最优解, 具有较大理论意义. 但由于该算法本质上仍属于穷举法, 其计算时间复杂度大多为指数型, 难以实际应用. 智能优化算法利用某种拟人、拟物的机制来引导算法执行搜索, 往往在几秒或十几秒就能获得各类调度问题的满意解. 正是由于智能算法具有性能好、求解时间短、适用面广的优点, 使其近年得到广泛研究和应用.

目前, 各种智能优化算法已被设计用于求解包括并行机在内的大量生产调度问题, 并取得良好效果. 但是, 各类文献对所提算法使用少量个体 (一般种群规模为 20 ∼ 100) 运行少量代数 (一般进化代数为 100 ∼ 500 代) 后, 为何都可获得问题的满意解, 基本都只从各自算法本身的机制、特点进行解释, 而忽略了该类算法所采用的排序模型对求解质量的重要影响. 这导致了很多后续研究者过于追求方法和技巧上的创新. 实际上, 智能优化算法为何有效, 是由所采用排序模型的解空间和解的特点, 以及智能优化算法自身机制共同决定的.

在排序模型的解空间方面, 调度问题的目标值变化范围远远小于排序模型解空间的规模. 譬如, 对于智能优化算法求解最多的优化目标为最小化最大完工时间的置换流水线调度问题, 假如有 50 个工件, 5 台机器, 每个工件在每台机器上的加工时间为在 $[1, 100]$ 间均匀分布的随机数, 则其目标值变化范围在 $(1, 25\,000)$ 之内 ($25\,000$ 为各工件在各机器上的加工时间之和, 实际的最大目标值小于此值), 而解空间规模为 50!, 平均每一个具体目标值对应约 1.2×10^{60} 个不同排列或解, 这表明数量巨大的不同解具有相同的目标值. 对于其他类型的调度问题 (包括本文研究的并行机问题) 同样存在上述情况. 因此, 调度问题的排序模型解空间是一个 “巨大” 和 “极为扁平” 的空间. 该空间规模 “巨大”, 内部遍布各种不同的山峰 (峰顶为局部极大解) 和低谷 (谷底为局部极小解), 而最高峰顶 (全局极大解) 和最低谷底 (全局极小解) 之间的差值相对于整个空间的规模来说又非常小, 这使得整个空间呈现出 “极为扁平” 的形态, 即空间中大量不同的位置 (对应不同的解) 会具有相同的目标值. 另外, 就排序模型的解 (即排列或个体) 而言, 解与解之间没有梯度, 两个相似解的目标值也可能差别较大. 排序模型解空间的 “极为扁平” 性, 以及解与解之间 “无梯度” 性, 使得任何带保优的随机搜索只需用较短时间搜索解空间中极小的区域 (类似足球场中 1 根针面积甚至更小的区域), 却可能到达较广的目标值区域. 各种智能优化算法均是在带保优的随机搜索基础上, 增加了自身的寻优机制. 这使此类算法不仅可短时间内搜索较广的目标值区域, 而且其内在的寻优机制可推动算法到达目标值较小的不同区域搜索, 从而获得问题的满意解. 智能优化算法这种通过对排序模型解空间极小区域的搜索来实现对目标值较大、较优区域的搜索, 是其有效地本质原因, 也是现有运筹学算法和基于动态规划的近似算法这些偏遍历或部分遍历解空间的算法难以做到的. 因此, 采用智能优化算法求解复杂调度问题是合理且必要的.

对于基于排序模型的复杂并行机调度问题, 智能优化算法在过去几年中已有较多的研究. Woo 等[14] 针对在处理时间不断恶化下带多个机器维护阶段的并行机调度问题, 提出一种融合了改进启发式操作的基于规则的遗传算法 (Genetic algorithm, GA) 来求解, 以此增强解的高效性. Avalos-Rosales 等[15] 针对带序相关设置时间的不相关并行机调度问题, 提出了一种基于多启动和变邻域下降的元启发式算法. 张嘉琦等[16] 针对并行机动态调度问题, 引入问题分解思想和评价策略, 提出了一种基于差分进化算法与代理模型相结合的快速求解方法. Chen[17] 针对带不相等准备时间和序相关设置时间

的不相关并行机问题, 提出了几种迭代混合元启发式算法进行求解. Damodaran 等[18] 提出一种微粒群优化算法 (Particle swarm optimization, PSO) 求解不相关并行批处理机问题, 测试结果表明该算法在求解大规模问题时, 能在较短时间内找到优质解. Diana[19] 提出了一种基于变邻域下降的混合免疫算法求解带序相关设置时间的不相关并行机调度问题. Sels 等[20] 针对不相关并行机调度问题, 提出了融合禁忌搜索、遗传算法和截断分支定界法的混合算法, 通过标准测试问题表明了所提算法的有效性. Bitar 等[21] 针对半导体制造中的不相关并行机调度问题, 提出了一种文化基因算法, 在合理的测试时间内求得了较好的解. Joo 等[22] 提出了三种分配规则的混合遗传算法, 求解带序相关设置时间和产品约束的不相关并行机调度问题, 通过对比实验验证了算法的高效性和鲁棒性. 罗家祥等[23] 提出一种基于变深度环交邻域结构的迭代搜索算法, 用于求解不相关并行机调度问题, 仿真实验验证了算法计算结果十分接近问题的下界. Lin 等[24] 提出了一种混合人工蜂群算法求解带序相关设置时间的不相关并行机调度问题. 由文献调研可知, 尚无智能优化算法求解 PMSP_AMPS 的相关研究.

对于 PMSP_AMPS 这一类复杂并行机调度问题, 无论是运筹学方法还是基于动态规划的近似算法均难以在较短时间内有效求解, 故有必要设计相应的智能优化算法. 教与学算法 (Teaching-learning-based optimization, TLBO) 是一种基于群体智能的连续随机优化算法, 最早由 Rao 等[25] 于 2011 年提出. 它模拟了教师给学员的教学过程和学员的学习过程, 目的是通过教师的 “教” 和学员之间的相互 “ 学” 来提高学员的学习成绩. TLBO 参数少、算法简单、易理解、搜索能力较强[26]. 对于制造过程调度问题 (即一类典型的离散优化问题), 现有的 TLBO 求解算法分为两类: 1) 以标准的连续 TLBO 为基础, 通过一定的编码和解码方式实现离散问题解空间到连续空间的映射, 在 “教”、“ 学” 阶段, 对个体的更新仍采用传统的连续运算准则; 2) 利用 TLBO 的基本思想, 重新定义 “教”、“学” 阶段中个体的离散更新方式, 使算法可直接用于离散解空间的搜索, 这类算法称为离散 TLBO (Discrete TLBO, DTLBO). 近年 TLBO 和 DTLBO 在生产调度领域均已有初步研究, 在 TLBO 方面, Shao 等[27] 针对零等待流水车间调度问题, 提出了一种基于高斯分布的扩展教与学算法, 建立反向学习机制和高斯概率模型实现对个体的更新, 并加入基于模拟退火和插入操作的局部搜索, 通过对标准测试问题的仿真实验验证了所提算法的有效性. Xu 等[28] 针对带模糊加工时间的柔性作业车间调度问题, 提出了一种有效地 TLBO 算法, 通过加入交叉操作和局部搜索以增强算法的性能. 在 DTLBO 方面, Shao 等[29] 提出了一种混合离散优化算法求解零等待流水车间调度问题, 采用前向、后向插入操作模拟教学阶段, 同时将学习阶段用 2 维分布估计算法替换, 并使用基于 3 种邻域结构的局部搜索来增强算法的搜索能力, 仿真实验验证了其算法的鲁棒性和有效性. Li 等[30] 针对流水线重调度问题, 提出离散教与学算法进行求解, 在 “ 教”、“学” 阶段分别定义了 2 种较复杂的离散操作, 并在每个阶段都采用同一种迭代贪心局部搜索来提高算法性能, 所提算法在与 5 有效算法的对比中优势明显. 文献调研表明, DTLBO 求解并行机调度问题的研究处于空白状态.

已有的 TLBO 和 DTLBO 与大多数文献中的连续域智能调度算法一样 (包括前一段中求解并行机问题的连续域智能算法), 均过于强调原有机制的完整保留, 要么采用一定的编码和解码方式实现原算法从连续空间搜索到离散空间搜索的映射 (譬如文献 [27−28]), 要么在离散化时不够彻底和直接 (譬如文献 [27−28]), 从而导致全局搜索的实际效率并不高. 因此, 本文设计一种离散化更为彻底的 DTLBO, 直接用大部分智能调度算法实际执行搜索的排序操作 (即交叉、插入、交换等) 来对 TLBO 进行离散化, 以实现对解空间更有效率的搜索. 具体来说, 本文提出一种混合离散教与学优化算法 (Hybrid discrete teaching-learning-based optimization, HDTLBO), 求解以最小化最大完工时间为目标的 PMSP_AMPS. HDTLBO 设计了针对问题解的多种排序操作, 在保持标准教与学算法更新机理的前提下实现算法的离散化. 首先, 在算法初始化阶段, 将工件排序 (即问题的解) 作为种群中的个体, 随机初始化种群, 这样有利于确保算法初期搜索的分散性; 其次, 设计了不同的排序操作以实现对标准 TLBO 两阶段更新公式的离散化, 使算法能直接在离散解空间中进行基于 TLBO 机理的有效全局搜索, 明显增强了算法的全局搜索效率; 最后, 采用 *Interchange* 和 *Insert* 邻域操作设计了一种有效地变邻域局部搜索, 进一步增强了算法的局部搜索能力. 通过对不同测试问题的仿真实验和算法比较验证了 HDTLBO 的有效性和鲁棒性.

1 问题描述

第 1.1 节建立 PMSP_AMPS 的排序模型, 第 1.2 节给出该模型的一个示例.

1.1 问题模型

PMSP_AMPS 可描述如下: n 个产品或工件在 m 台设备上进行加工. 每种产品需要 s_j,

$j \in (1,2,\cdots,l)$ 道加工工序, $S_t = [s_1, s_2, \cdots, s_l]$ 表示所有产品的工序数所构成的集合, 同种产品的不同工序需要按先后顺序加工; 所有工序只能由满足加工约束的设备进行加工; 产品的加工时间与加工设备有关, 任何设备同一时刻只能加工一种产品; 不同产品间带序相关设置时间, 其依赖于加工顺序, 同种产品间的设置时间为 0.

令 TS 为所有产品的总工序数, $\pi = [\pi_1, \pi_2, \cdots, \pi_{TS}]$, $\pi_j \in (1,2,\cdots,n)$ 为待加工的 n 个工件基于工序的排列 (该排列中的工件从左往右根据一定的规则及并行加工约束分配到相应的机器上加工), 也是问题的决策变量, T_i 为第 i 台设备上加工的工序总数, $\pi^{T(i)} = [\pi_1^{T(i)}, \pi_2^{T(i)}, \cdots, \pi_{T_i}^{T(i)}], (\pi_k^{T(i)} \in (1,2,\cdots,n), k = 1,2,\cdots,T_i)$ 为第 i 台设备上所加工工件基于工序的排列; $P(\pi_k^{T(i)})$ 为 $\pi_k^{T(i)}$ 的加工时间 ($k = 0,1,\cdots,T_i; P(\pi_0^{T(i)}) = 0$); $St(\pi_k^{T(i)})$ 为 $\pi_k^{T(i)}$ 的开始加工时间 ($St(\pi_0^{T(i)}) = 0$); $S(\pi_{k-1}^{T(i)}, \pi_k^{T(i)})$ 为 $\pi_{k-1}^{T(i)}$ 与 $\pi_k^{T(i)}$ 之间的序相关设置时间 ($S(\pi_0^{T(i)}, \pi_1^{T(i)}) = 0$, 当 $k > 1$ 且 $\pi_{k-1}^{T(i)} = \pi_k^{T(i)}$ 时 $S(\pi_{k-1}^{T(i)}, \pi_k^{T(i)}) = 0$); $pre\_m(\pi_k^{T(i)})$ 为 $\pi_k^{T(i)}$ 前一次加工所用的设备号, $pre\_k(\pi_k^{T(i)})$ 为 $\pi_k^{T(i)}$ 在前一次加工设备上的排列 $\pi^{T(pre\_m(\pi_k^{T(i)}))}$ 中从左往右的位置, (当 $\pi_k^{T(i)}$ 是首次加工时, $pre\_m(\pi_k^{T(i)}) = 0, pre\_k(\pi_k^{T(i)}) = 0$); $R(\pi_k^{T(i)})$ 是工件 k 首次到达第 i 台设备的时间.

根据上述定义, 建立如下排序模型, 优化目标为在所有产品排序的集合 Π 中找到一个最优排序 π^*, 使得最早完工时间 $C_{\max}(\pi)$ 最小, 即

$$C_{\max}(\pi) = \max\{St(\pi_{T_1}^{T(1)}) + P(\pi_{T_1}^{T(1)}), St(\pi_{T_2}^{T(2)}) + P(\pi_{T_2}^{T(2)}), \cdots, St(\pi_{T_m}^{T(m)}) + P(\pi_{T_m}^{T(m)})\} \tag{1}$$

$$St(\pi_k^{T(i)}) = \begin{cases} \max\{St(\pi_{k-1}^{T(i)}) + P(\pi_{k-1}^{T(i)}) + S(\pi_{k-1}^{T(i)}, \pi_k^{T(i)}), \\ R(\pi_k^{T(i)})\}, \quad \text{若 } pre\_k(\pi_k^{T(i)}) = 0 \\ \max\{St(\pi_{k-1}^{T(i)}) + P(\pi_{k-1}^{T(i)}) + S(\pi_{k-1}^{T(i)}, \pi_k^{T(i)}), \\ St(\pi_{pre\_k(\pi_k^{T(i)})}^{T(pre\_m(\pi_k^{T(i)}))}) + P(\pi_{pre\_k(\pi_k^{T(i)})}^{T(pre\_m(\pi_k^{T(i)}))})\}, \quad \text{其他} \end{cases} \tag{2}$$

$$C_{\max}(\pi^*) = \min_{m \subset \Pi} C_{\max}(\pi) \tag{3}$$

$$\pi^* = \arg\{C_{\max}(\pi)\} \to \min, \quad \forall \pi \in \Pi \tag{4}$$

其中, 式 (1) 和式 (2) 为最大完工时间 $C_{\max}(\pi)$ 的计算公式, 式 (3) 和式 (4) 表示在所有产品集合 Π 中找到最优排序 π^*, 使得 $C_{\max}(\pi)$ 最小.

1.2 甘特图示例

由于 PMSP_AMPS 中部分产品需多工序加工, 故 HDTLBO 采用基于操作的编码方式来表示个体或解. 例如, 对于 $n = 5$, $m = 3$, 工序集合 $S_t = [3,1,3,2,1]$ 的问题, $\pi = [1\ 3\ 2\ 5\ 4\ 1\ 3\ 1\ 3\ 4]$ 可以表示该问题采用基于操作的编码方式的一个编码排列, 即第一个加工的工件为工件 1, 共有 3 个加工工序, 第二个加工的工件是工件 3, 共有 3 个加工工序, 以此类推. 本文采用最早完工时间 (Early completion time, ECT) 规则进行解码. 现对 $\pi = [1\ 3\ 2\ 5\ 4\ 1\ 3\ 1\ 3\ 4]$ 的解码操作举例说明, 排列 π 中各工件的工序加工约束、序相关设置时间 (s) 和工件到达时间 (s) 如表 1 所示, 其中每个工件的不同操作可在 0 台或多台机器上加工, 工序加工时间如表 2 所示. 根据表 1、表 2 中所给数据, 结合 ECT 规则可得 $T_1 = 4$, $T_2 = 4$, $T_3 = 3$, 3 台机器分别对应的工件排序为: $\pi^{T(1)} = [\pi_1^{T(1)}, \pi_2^{T(1)}, \pi_3^{T(1)}, \pi_4^{T(1)}] = [2,4,1,4]$; $\pi^{T(2)} = [\pi_1^{T(2)}, \pi_2^{T(2)}, \pi_3^{T(2)}, \pi_4^{T(2)}] = [1,1,3,3]$; $\pi^{T(3)} = [\pi_1^{T(3)}, \pi_2^{T(3)}] = [3,5]$, 对应的甘特图如图 1 所示.

2 HDTLBO 算法

本节将对 HDTLBO 中的关键环节, 即离散 "教"、" 学" 阶段以及 HDTLBO 流程进行介绍. 根据标准 TLBO 两阶段的个体更新机制, HDTLBO 分别在离散 "教"、"学" 阶段设计了不同的排序操作 (基于交叉、插入、交换等), 以此替换标准 TLBO 中对实数个体的核心更新方式, 进而实现对 TLBO 的离散化和对排序模型解空间的有效搜索. 这使得 HDTLBO 能够直接用于求解 PMSP_AMPS, 并且在全局搜索能力上得到明显提高.

令 X_i^{gen} 是第 gen 代中的第 i 个个体, X_{mean}^{gen} 是第 gen 代的种群均值, $X_{teacher}^{gen}$ 是第 gen 代种群中的最优个体即 "教师", TF 为教学因子, $TF = round[1 + rand(0,1)]$, r_i 为学习概率, $r_i = rand(0,1)$, $rand[0,1]$ 是区间 $[0,1]$ 上分布的随机数, $f(X_i^{gen})$ 为第 gen 代中第 i 个个体的评价函数值. 根据上述定义我们提出以下第 2.1 节 ~ 第 2.3 节的 DTLBO.

2.1 种群初始化

本文采用随机初始化方法产生初始种群, 以保证解的多样性和分散性.

2.2 离散 "教" 阶段

本阶段学生 (个体) 通过自身与教师 (种群中当前最优个体) 之间的差异交互来获得知识, 每个个体

表 1　工序加工约束、序相关设置时间和工件到达时间表 ($\pi = [1\ 3\ 2\ 5\ 4\ 1\ 3\ 1\ 3\ 4]$)

Table 1　The schedule of process constraint, sequence setup time and arrival time ($\pi = [1\ 3\ 2\ 5\ 4\ 1\ 3\ 1\ 3\ 4]$)

	可执行操作的设备			序相关设置时间					首次到达设备的时间		
	操作 1	操作 2	操作 3	工件 1	工件 2	工件 3	工件 4	工件 5	m1	m2	m3
工件 1	m1/m2	m1/m2/m3	m1/m3	—	83	38	39	47	35	34	23
工件 2	m1	—	—	53	—	66	45	25	38	78	98
工件 3	m3	m2/m3	m2	64	57	—	72	46	52	23	32
工件 4	m1/m3	m1	—	46	67	83	—	55	132	131	98
工件 5	m1/m3	—	—	40	66	83	77	—	114	99	112

表 2　工件加工时间表 ($\pi = [1\ 3\ 2\ 5\ 4\ 1\ 3\ 1\ 3\ 4]$) ($t$/s)

Table 2　The processing time table ($\pi = [1\ 3\ 2\ 5\ 4\ 1\ 3\ 1\ 3\ 4]$) ($t$/s)

	操作 1			操作 2			操作 3		
	m1	m2	m3	m1	m2	m3	m1	m2	m3
工件 1	62	44	—	78	86	26	48	—	87
工件 2	41	—	—	—	—	—	—	—	—
工件 3	—	—	31	—	58	65	—	42	—
工件 4	32	—	31	27	—	—	—	—	—
工件 5	74	—	36	—	—	—	—	—	—

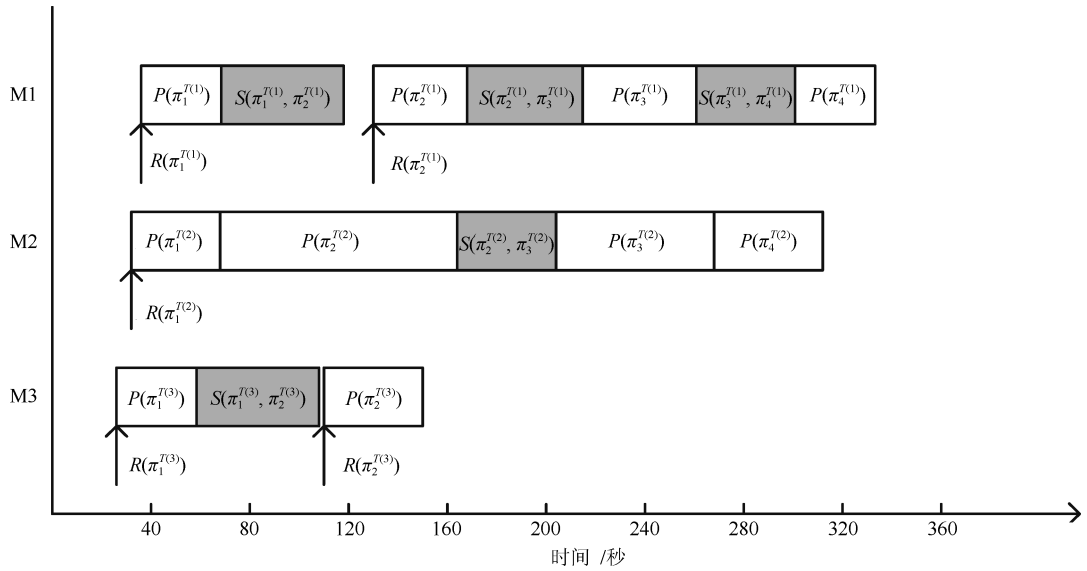


图 1　工序为 $\pi = [1\ 3\ 2\ 5\ 4\ 1\ 3\ 1\ 3\ 4]$ 时的甘特图

Fig. 1　The Gantt chart of $\pi = [1\ 3\ 2\ 5\ 4\ 1\ 3\ 1\ 3\ 4]$

通过最优个体 (教师) 的教导提升自身的水平进而提高整个种群的均值. 针对调度问题解空间的特点, 设计不同的排序交叉操作实现教学活动, 使个体能直接从教师中获取优质解信息. 定义种群均值为当前种群中处于中间大小的目标值所对应的个体或排序. 根据上述定义, 提出如式 (5) 所示的个体更新公式.

$$X_i^{gen+1} = \{r_m \odot (X_i^{gen}), TF \odot [X_{teacher}^{gen}, X_{mean}^{gen}]^{r[a,b]}\}^{r[c]} \tag{5}$$

显然, 式 (5) 中依次执行的三个核心操作分别为 $r_m \odot (X_i^{gen})$、$TF \odot [X_{teacher}^{gen}, X_{mean}^{gen}]^{r[a,b]}$ 和 $\{r_m \odot (X_i^{gen}), TF \odot [X_{teacher}^{gen}, X_{mean}^{gen}]^{r[a,b]}\}^{r[c]}$, 第 2.2.1 节 ∼ 第 2.2.3 节将对这三个核心操作进行具体描述.

2.2.1 $r_m \odot (X_i^{gen})$ 描述

$r_m \odot (X_i^{gen})$ 表示以概率 r_m 对 X_i^{gen} 执行变异操作, 引入变异概率 r_m, 模拟个体在 "教" 之前的自学习过程, 设计 *Interchang* 和 *Insert* 邻域操作实现自学习. 令 $Interchange(\pi, u, v)$ 为将解 π 中的第 u 个位置上的工件与第 v 个位置上的工件进行交换, $Insert(\pi, u, v)$ 为将解 π 中的第 u 个位置上的工件插入到第 v 个位置上, u 和 v 为随机生成的不同整数. 自学习过程如式 (6) 所示:

$$\overline{X_i^{gen}} = r_m \odot (X_i^{gen}) = \begin{cases} (X_i^{gen}), & 若\ rand[0,1] < r_m \\ X_i^{gen}, & 否则 \end{cases} \tag{6}$$

当产生的 $rand[0,1]$ 小于 r_m 时, 执行变异操作 $\overline{X_i^{gen}} = (X_i^{gen})$, 反之执行 $\overline{X_i^{gen}} = X_i^{gen}$. 变异操作采用如下分阶段的方式:

$$(X_i^{gen}) = \begin{cases} Interchange(X_i^{gen}, u, v), & 若\ gen \le 50 \\ Insert(X_i^{gen}, u, v), & 否则 \end{cases} \tag{7}$$

即在算法运行的前 50 代执行 $Interchange(X_i^{gen}, u, v)$ 操作, 之后算法执行 $Insert(X_i^{gen}, u, v)$ 操作, 这有利于搜索问题解空间中的不同区域. 执行 $r_m \odot (X_i^{gen})$ 后, 式 (5) 变为如下所示:

$$X_i^{gen+1} = \{\overline{X_i^{gen}}, TF \odot [X_{teacher}^{gen}, X_{mean}^{gen}]^{r[a,b]}\}^{r[c]} \tag{8}$$

2.2.2 $TF \odot [X_{teacher}^{gen}, X_{mean}^{gen}]^{r[a,b]}$ 描述

$$X_{mean\_new}^{gen} = TF \odot [X_{teacher}^{gen}, X_{mean}^{gen}]^{r[a,b]} = \begin{cases} [X_{teacher}^{gen}, X_{mean}^{gen}]^{r[a,b]}, & 若\ TF = 2 \\ X_{mean}^{gen}, & 否则 \end{cases} \tag{9}$$

该操作为种群均值的离散化更新方式, 种群中每个个体在 "教" 阶段通过不断学习提高自身的评价值进而提高整个种群均值. 其中, 教学因子 TF 决定了均值的选取, $[X_{teacher}^{gen}, X_{mean}^{gen}]^{r[a,b]}$ 为对 $X_{teacher}^{gen}$ 和 X_{mean}^{gen} 执行基于 $r[a,b]$ 的交叉操作, $r[a,b]$ 为随机选取整数 a, b $(1 \le a \le b \le TS)$ 以构成区间 $[a,b]$. 采用类似文献 [31] 中基于顺序的交叉法 (Order-based crossover, OBX) 实现 $TF = 2$ 时新的均值个体的产生, 具体交叉方法如图 2 中例子所示. 执行完此节操作后, 式 (5) 变为如下所示:

$$X_i^{gen+1} = \{\overline{X_i^{gen}}, X_{mean\_new}^{gen}\}^{r[c]} \tag{10}$$

2.2.3 $\{\overline{X_i^{gen}}, X_{mean\_new}^{gen}\}^{r[c]}$ 描述

该操作为 "教" 阶段的最后一个操作, 根据当前学生与教师水平之间的差异对解进行更新. $r[c]$ 为随机选取整数 c $(1 \le c \le n)$, $\{\overline{X_i^{gen}}, X_{mean\_new}^{gen}\}^{r[c]}$ 表示对 $\overline{X_i^{gen}}$ 和 $X_{mean\_new}^{gen}$ 执行基于 $r[c]$ 的交叉操作, 该操作可举例说明, 譬如 $\overline{X_i^{gen}} = [2\ 3\ 5\ 4\ 3\ 4\ 1\ 1\ 3\ 1]$, $X_{mean\_new}^{gen} = [5\ 4\ 1\ 3\ 3\ 4\ 2\ 1\ 3\ 1]$, $c = 3$ 时, 将 $\overline{X_i^{gen}}$ 中为 3 的元素均置 0, 可得第一个中间排列 $temp1 = [2\ 0\ 5\ 4\ 0\ 4\ 1\ 1\ 0\ 1]$, 然后将 $X_{mean\_new}^{gen}$ 中不为 3 的元素均置 0, 可得第二个中间排列 $temp2 = [0\ 0\ 0\ 3\ 3\ 0\ 0\ 0\ 3\ 0]$, 然后将 $temp1$ 中不为 0 的元素从左至右依次替换 $temp2$ 中为 0 的元素, 即可生成交叉后的排列 $X_i^{gen+1} = [2\ 5\ 4\ 3\ 3\ 4\ 1\ 1\ 3\ 1]$. 在执行完最后一个操作后, 将所得 X_i^{gen+1} 和 X_i^{gen} 进行比较, 把两者中较优者保存至 X_i^{gen+1}, 从而完成了离散 "教" 阶段的个体更新.

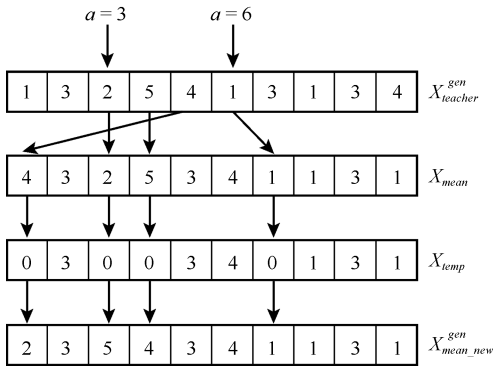


图 2 基于顺序的交叉法

Fig. 2 The order-based crossover (OBX)

2.3 离散 "学" 阶段

首先把 "教" 阶段的 X_i^{gen+1} 赋值给本阶段的 X_i^{gen}, 然后本阶段班级中学生之间互相学习, 通过学生间的互相交流, 学生可从比自己知识多的学生 (即对应目标值更小的个体) 那获取新的知识. 对任一学生 X_i^{gen}, 随机选取另一学习对象 X_j^{gen}, 以一定的学习概率 r_i 向学员 X_j^{gen} 获取新知识. 本节通过不同的交叉操作实现学生之间的互相学习, 以此缩小学生之间的差异, 解的更新公式如式 (11) 所示.

$$X_i^{gen+1} = r_i \odot \{X_i^{gen}, [X_i^{gen}, X_j^{gen}]^{r[d,e]}\}^{r[c]} \tag{11}$$

2.3.1 $[X_i^{gen}, X_j^{gen}]^{r[d,e]}$ 描述

$$X_{d\_temp}^{gen} = [X_i^{gen}, X_j^{gen}]^{r[d,e]} = \begin{cases} [X_i^{gen}, X_j^{gen}]^{r[d,e]}, & 若\ f(X_i^{gen}) < f(X_j^{gen}) \\ [X_j^{gen}, X_i^{gen}]^{r[d,e]}, & 否则 \end{cases} \tag{12}$$

该操作中, 种群中两个个体通过互相学习, 差的个体从较优个体获取知识实现取长补短. 由于每个个体在学习时, 是在小范围的学生之间进行, 不会过早向全局最优点方向聚集, 因此能够有效保持学员的多样性特征, 从而保证算法在搜索空间的全局探索能力. 利用类似文献 [32] 中的顺序交叉 (Order crossover, OX) 法实现个体之间互相学习. $[X_i^{gen}, X_j^{gen}]^{r[d,e]}$ 表示对 X_i^{gen} 和 X_j^{gen} 执行基于 $r[d,e]$ 的交叉操作, $r[d,e]$ 表示随机选取整数 d,e $(1 \le d \le e \le TS)$ 以构成区间 $[d,e]$, 具体操作如图 3 中例子所示. 执行此节操作后, 式 (11) 变为如下所示:

$$X_i^{gen+1} = r_i \odot \{X_i^{gen}, X_{d\_temp}^{gen}\}^{r[c]} \tag{13}$$

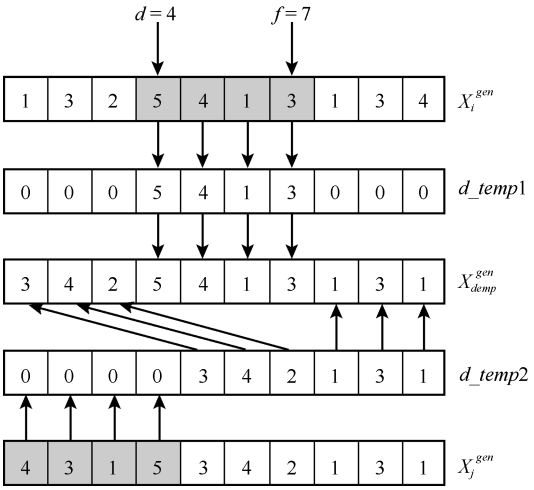


图 3　顺序交叉法

Fig. 3　Order crossover (OX)

2.3.2　$r_i \odot \{X_i^{gen}, X_{d\_temp}^{gen}\}^{r[c]}$ 描述

在学生互相学习的过程中, 学生对已获得的知识可能会存在一定程度上的误解, 如果在个体相互学习过程中对已获取的知识完全信任, 学生就会盲目信任自己已获取知识, 此时算法易陷入局部最优. 因此, 引入学习概率 r_i 控制解的更新方式, 避免学生相互学习中的盲目信任. 具体更新方式如式 (14) 所示.

$$X_i^{gen+1} = r_i \odot \{X_i^{gen}, X_{d\_temp}^{gen}\}^{r[c]} = \begin{cases} \{X_i^{gen}, X_{d\_temp}^{gen}\}^{r[c]}, & \text{若 } r_i < 0.5 \\ X_{d\_temp}^{gen}, & \text{否则} \end{cases} \tag{14}$$

该操作是 “学” 阶段的最后一个操作, 其中的 $\{X_i^{gen}, X_{d\_temp}^{gen}\}^{r[c]}$ 与 “教” 阶段第 2.2.3 节的操作相似. 在执行完最后一个操作后, 将 X_i^{gen+1} 和 X_i^{gen} 中较优者保存至 X_i^{gen+1}, 进而完成了离散 “学” 阶段的个体更新. 至此整个 DTLBO 两阶段离散搜索执行完毕.

2.4　基于 *Interchange* 和 *Insert* 的局部搜索

为增加算法的搜索能力, 需对优质解的近邻区域进行较细致的局部搜索. 在各种常用搜索操作或邻域中, 交换操作 *Interchange* 和插入操作 *Insert* 构成的问题解空间中解之间的距离是最短的[33]. 基于这两种邻域构造的局部搜索可引导算法在更为紧凑的解空间中进行搜索, 有利于提高算法的性能. 因此, HDTLBO 对全局搜索之后产生的新种群中前 20 个优质个体 X_i^{gen+1} 执行基于这两种邻域的局部搜索. 具体来说, 对每个进行局部搜索的个体 X_i^{gen+1} 执行 1 次扰动操作 $tmp1 = Interchange(X_i^{gen+1}, u, v)$ 后, 均对 $tmp1$ 执行 n 次插入操作 $tmp2 = Insert(tmp1, u, v)$. 每次插入操作时, 如 $tmp2$ 优于 $tmp1$ 则令 $tmp1 = tmp2$. 执行完 n 次插入操作后, 如 $tmp1$ 优于 X_i^{gen+1} 则令 $X_i^{gen+1} = tmp1$. 其中, $u, v\,(u \ne v)$ 为随机数, n 为工件数, 每次执行交换、插入操作时均重新生成 u 和 v.

2.5　HDTLBO 流程

令 $P(gen)$ 为算法第 gen 代种群, $popsize$ 为种群大小, $gen\_\max$ 为算法最大运行代数. 根据第 2.1 节 ∼ 第 2.4 节描述, HDTLBO 的具体步骤如下:

步骤 1. 初始化种群. 令 $gen = 1$, 设定 $popsize$ 和 $gen\_\max$, 随机生成初始种群 $P(0)$, 并计算种群中每个个体的适配值 f;

步骤 2. 根据第 2.1 节 ∼ 第 2.3 节内容生成 $P(gen)$ 中的每一个个体, 并进行保优更新;

步骤 3. 根据第 2.4 节内容对种群中前 20 个个体执行基于 *Interchange* 和 *Insert* 的局部搜索;

步骤 4. 令 $gen = gen + 1$;

步骤 5. 如果 $gen \le gen\_\max$, 转到步骤 2, 否则输出当前种群中的最优个体 X_{best}^{gen}.

由上述算法步骤可知, 步骤 2 直接负责在基于排列的问题解空间中进行全局搜索, 用于发现解空间中的优质区域, 步骤 3 对全局搜索找到的优质区域再进行较为细致的局部搜索, 从而增强算法的性能, 也使算法在全局和局部搜索之间达到较好的平衡, 完整算法流程图如图 4 所示. 由此知, HDTLBO 有望成为求解 PMSP_AMPS 的有效算法.

2.6　算法复杂度分析

群智能算法的计算复杂度是指算法执行加、减、乘、除的次数. 对于种群规模为 $popsize$, 迭代次数为 $gen\_\max$, 问题规模为 $n \times m$ 的问题, HDTLBO 的计算复杂度分析如下:

由第 1 节可知, 计算个体适应度 (即计算 $C_{\max}(\pi)$) 的计算复杂度为 $O(mn)$. 初始化种群的

计算复杂度为 O($popsize \times mn$). 初始化种群之后, 每一次迭代均需经过离散 "教" 阶段、离散 "学" 阶段和局部搜索三个步骤. 离散 "教" 阶段的计算复杂度为 O($popsize \times mn + popsize^2$) (O($popsize^2$) 为

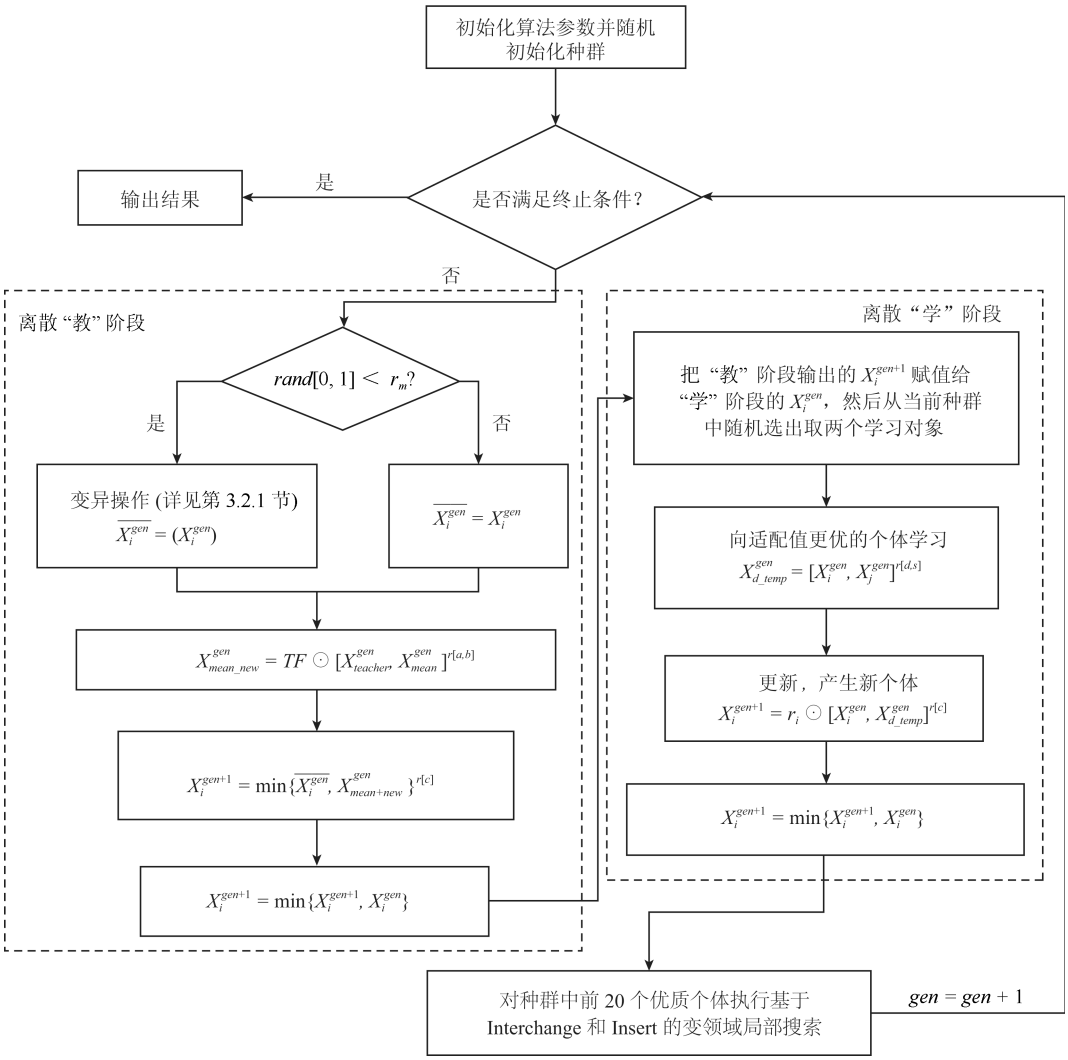


图 4 HDTLBO 流程图

Fig. 4 HDTLBO's flow chart

采用冒泡排序计算种群均值时的计算复杂度), 离散 "学" 阶段的计算复杂度为 O($popsize \times mn$), 局部搜索的复杂度为 O($20 \times mn^2 + popsize^2$) (O($popsize^2$) 为采用冒泡排序计算种群前 20 较优个体的计算复杂度). 由此整个算法的时间复杂度为:

$$\begin{aligned}&\mathrm{O}(popsize, gen\_\max, n \times m) = \\ &\quad \mathrm{O}(popsize \times mn) + gen\_\max \times \\ &\quad (\mathrm{O}(popsize \times mn + popsize^2) + \mathrm{O}(popsize \times \\ &\quad mn) + \mathrm{O}(20 \times mn^2 + popsize^2)) = \\ &\quad gen\_\max \times (\mathrm{O}(popsize \times mn + popsize^2) + \\ &\quad \mathrm{O}(20 \times mn^2 + popsize^2)) \end{aligned} \tag{15}$$

从式 (15) 可以看出, HDTLBO 整体计算复杂度的最高次项是问题输入规模 n 和种群大小 $popsize$ 的平方 (实际 $popsize$ 是一个较小值, 本文为 30), 故具有较低计算复杂度.

3 仿真实验及比较

测试问题采用随机方式生成, 工件需要加工的工序在 [1, 3] 上随机产生, 工件第 i 个操作的加工时间在 [1, 100] 上随机产生, 工件间的设置时间在 [1, 30] 上随机产生, 工件的到达时间在 [1, 100] 上随机产生, 测试问题的规模 $n \times m$ 为: 10 × 5, 20 × 5, 30 × 5, 40 × 5, 50 × 5, 40 × 10, 50 × 10, 60 × 10, 70 × 10, 80 × 10, 80 ×

20, 90 × 20, 100 × 20, 150 × 20 和 200 × 20 共有 15 个测试问题. 所有算法和测试均由 Delphi 2010 编程实现, 操作系统为 Window 10, CPU 为 2.2 GHz, 内存为 4 GB. 每种算法对每个测试问题均在相同时间下独立运行 20 次, 其中, Fit 表示算法独立运行 1 次输出的最优值, 则 $AVG = \sum_{i=1}^{20} Fit_i/20$ 表示算法独立运行 20 次输出结果的平均值, $BST = \min\ \{Fit_i \,|\, i = 1, \cdots, 20\}$ 表示算法独立运行 20 次输出结果的最优值, $WST = \max\ \{Fit_i \,|\, i = 1, \cdots, 20\}$ 表示算法独立运行 20 次输出结果的最差值.

表 3　参数水平设置表

Table 3　Combinations of parameter values

参数	水平			
	1	2	3	4
$popsize$	10	20	30	40
TF	0	1	2	3
r_m	0.1	0.4	0.7	0.9

3.1　参数设置探讨

HDTLBO 的关键参数为种群规模 $popsize$、教学因子 TF 和变异概率 r_m. 本节将针对一组中等规模问题 (60 × 10) 采用实验设计方法 DOE 进行实验分析, 进而确定 HDTLBO 的参数设置. 上述 3 个参数均取 4 个水平 (见表 3), 选择规模为 $L16\,(4^3)$ 的正交实验, 每种参数组合下的测试问题均分别进行 20 次独立运行实验. HDTLBO 在各个参数组合下的运行时间均为 $100 \times n$ (ms), 以 AVG 为评价指标进行测试实验, 参数响应值如表 5 所示, 各参数的相应趋势如图 5 所示.

由表 3 ∼表 5 和图 5 可知, $popsize$、TF 和 r_m 对算法的影响性能等级. 说明若参数选择过大会使得算法偏离优质区域过远, 过小则会使得算法的搜索区域过窄. 综合考虑, HDTLBO 的参数设置为: $popsize = 30$, $TF = 2$, $r_m = 0.9$, 此时, 算法可表现出良好的性能.

表 4　正交表和 AVG 统计

Table 4　Orthogonal array and AVG

参数组合	水平			AVG
	$popsize$	TF	r_m	
1	1	1	1	500.65
2	1	2	2	503.90
3	1	3	3	497.30
4	1	4	4	498.55
5	2	1	2	489.35
6	2	2	3	491.20
7	2	3	4	485.50
8	2	4	1	488.35
9	3	1	3	482.55
10	3	2	4	481.80
11	3	3	1	482.25
12	3	4	2	483.25
13	4	1	4	483.35
14	4	2	1	484.00
15	4	3	2	483.60
16	4	4	3	482.20

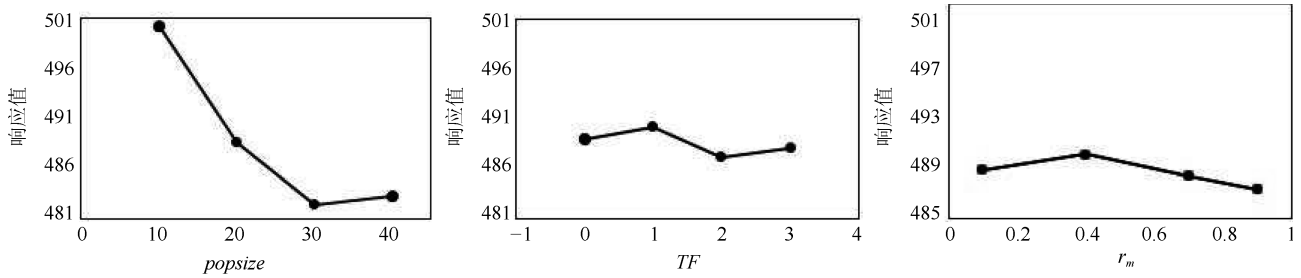
表 5　各参数响应值

Table 5　Average response value and rank of each parameter

水平	$popsize$	TF	r_m
1	500.10	488.98	488.81
2	488.60	490.23	490.02
3	482.46	487.16	488.31
4	483.29	488.09	487.30
等级	1	3	2

3.2　DTLBO 与其他算法的全局比较[1]

为了验证 DTLBO 的有效性, 将 DTLBO 与重要国际期刊中的 PSO[18]、DTLBO-I[30]、标准 TLBO[26]、CIWO[4] 进行算法全局搜索性能的仿真



(a) $popsize$ 的响应趋势　(b) TF 的响应趋势　(c) r_m 的响应趋势

(a) The influence trend of $popsize$　(b) The influence trend of TF　(c) The influence trend of r_m

图 5　各参数响应趋势

Fig. 5　The influence trend of each parameter

[1]本节表 6 与下节表 7 均给出了 $\rho = 4$ 时的各算法对比结果, $\rho = 2$ 及 $\rho = 6$ 时的算法对比结果及实例测试问题集可在 https://pan.baidu.com/s/1Foa_Vry7xxibW9jcoVBgXQ 下载, 若链接失效请联系通信作者.

比较, 其中 PSO[18] 求解的是一类不相关并行机调度问题. 设定每种算法的运行时间均为 $n \times m \times \rho$ 毫秒, ρ 的取值分别为 2、4、6, $\rho = 4$ 时的测试结果如表 6 所示. 为能清楚展现每个算法的性能, 表 6 中每个问题对应的最优结果用粗体表示.

由表 6 可见, DTLBO 通过对标准 TLBO[26] 的更新机制进行离散化处理后, 使算法在全局搜索能力上得以明显提高, 这验证了采用排序操作替换标准 TLBO 中核心操作的合理性. 同时, DTLBO 优于另一离散教与学优化算法 DTLBO-I[30], 这表明 DTLBO 中设计的排序操作更加有效. 此外, DTLBO 也优于具有较强全局搜索能力的 PSO[18], 这说明 DTLBO 可搜索到更多存在优质解的区域.

为了进一步验证 DTLBO 与其他算法差异性的显著与否, 根据 ρ 在 2、4、6 三种取值下的算法全局对比结果, 采用 Tukey's HSD 方法对表 6 中 DTLBO 等 5 种对比算法的 AVG (均值) 进行方差分析 (ANOVA). 图 6 显示了 5 种算法在不同终止运行时间下的均值变化线及 95 % 置信度下 Tukey's HSD 检验的置信区间. 从图 6 中可以看出, DTLBO 在均值水平上与其他算法均有显著差异, 且随着时间的增大, DTLBO 自身的均值水平并无明显的差异变化. 结合算法对比实验及方差分析结果可知, DTLBO 的全局搜索能力强, 能更有效地求解 PMAP_AMPS.

3.3 HDTLBO 与其他算法的比较

为进一步测试 HDTLBO 的性能, 本节将 HDTLBO 和重要国内外期刊中的有效算法 DPSO[34]、DTLBO-II[30]、GA_DR_C[22]、CCIWO[4] 进行比较. DPSO[34] 是求解零等待流水线问题公认的有效算法, GA_DR_C[22] 和 CCIWO[4] 是近年求解复杂并行机调度问题的有效算法. 设定每种算法的运行时间均为 $n \times m \times \rho$ 毫秒, ρ 的取值分别为 2、4、6, 表 7 给出了 $\rho = 4$ 时的算法运行结果, 表中每个问题对应的最优结果用粗体表示.

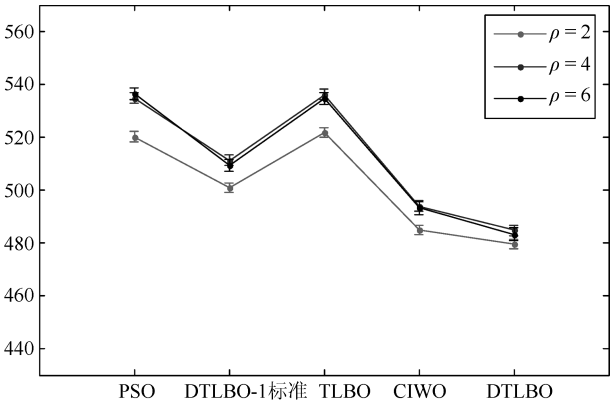


图 6 各算法在不同时间参数下的均值线及 95 % 置信度下 Tukey's HSD 检验的置信区间

Fig. 6 Means plot and 95 % Tukey's HSD confidence intervals for the interaction between the algorithms and the different time factor ρ

表 6 DTLBO 与 PSO、DTLBO-I、标准 TLBO 和 CIWO 的比较 ($\rho = 4$)

Table 6 Comparisons of DTLBO, DTLBO-I, PSO, CIWO, and standard TLBO ($\rho = 4$)

测试问题	PSO			DTLBO-I			标准 TLBO			CIWO			DTLBO		
	BST	*WST*	*AVG*	*BST*	*WST*	*AVG*	*BST*	*WST*	*AVG*	*BST*	*WST*	*AVG*	*BST*	*WST*	*AVG*
10×5	229	235	230.15	228	**231**	229.45	228	233	230.05	**227**	246	229.95	**227**	244	**229.3**
20×5	560	587	576.7	546	579	560.3	558	589	574.5	**503**	554	526.5	506	**534**	**521.4**
30×5	682	708	694.8	638	685	662.3	675	709	697	615	665	635.5	**607**	**638**	**622.4**
40×5	752	784	771.4	696	748	725.7	728	787	767.75	654	716	**690.05**	**661**	**716**	692.8
50×5	1 132	1 171	1 157.45	1 093	1 140	1 114.3	1148	1 175	1 161.5	**1 019**	1 091	1 059.2	**1 019**	**1 078**	**1 051.85**
40×10	340	357	348.2	307	336	321.3	340	360	351.15	293	326	311.9	**284**	**317**	**304.3**
50×10	410	429	419.7	373	411	392.75	413	429	421.1	362	391	376.3	**353**	**386**	**370.75**
60×10	523	548	538.8	497	529	516	538	552	544.3	485	**516**	**499.95**	**481**	**516**	500.6
70×10	658	682	672	626	653	641.8	660	684	672.65	604	639	622	**597**	**636**	**621.75**
80×10	596	619	612.15	567	601	585.55	597	621	612.95	**553**	589	571.45	**553**	**584**	**569.5**
80×20	286	298	292.2	265	289	278.85	285	298	291.8	**257**	282	271.65	259	**272**	**264.7**
90×20	287	299	294.65	271	289	280.55	288	301	297.4	**255**	284	273.1	**255**	**279**	**269.05**
100×20	329	340	336	309	328	320.5	333	343	338	304	323	312.65	**299**	**313**	**306.25**
150×20	479	496	489.2	460	486	472	487	499	492.9	454	476	464.95	**449**	**477**	**458.45**
200×20	575	593	587.25	557	576	568.5	577	598	589.85	559	575	564.4	**547**	**559**	**549.55**
Average	522.53	543.06	534.71	495.53	525.4	511.32	523.66	545.2	536.19	476.27	511.53	493.97	**473.13**	**503.27**	**488.84**

表 7　HDTLBO 与 DPSO、DTLBO-II、GA_DR_C 和 CCIWO 的比较 ($\rho = 4$)

Table 7　Comparisons of HDTLBO, DPSO, DTLBO-II, GA_DR_C, and CCIWO ($\rho = 4$)

测试问题	DPSO			DTLBO-II			GA_DR_C			CCIWO			HDTLBO		
	BST	*WST*	*AVG*	*BST*	*WST*	*AVG*	*BST*	*WST*	*AVG*	*BST*	*WST*	*AVG*	*BST*	*WST*	*AVG*
10×5	**227**	**230**	228.75	**227**	**230**	228.45	**227**	**230**	228.2	**227**	**230**	228.2	**227**	**230**	**227.3**
20×5	521	547	527.15	533	554	543.1	514	551	536.15	**494**	**536**	518.1	499	540	**520.95**
30×5	612	647	624.1	630	658	645.65	625	656	645.85	617	644	621.95	**603**	**638**	**617.5**
40×5	652	**692**	686.75	695	733	710.15	695	735	715.8	657	715	686.15	**645**	**692**	**681.65**
50×5	1 014	1 074	1 058.4	1 064	1 111	1 090	1 070	1107	1 094.75	1 024	1 088	1 057.05	**1 006**	**1 055**	**1 035.9**
40×10	303	317	306.14	303	328	316.2	305	325	316.4	292	320	307.7	**290**	**316**	**303.65**
50×10	365	387	370.95	370	395	387.95	374	400	388.2	362	390	373.85	**355**	**382**	**369**
60×10	481	503	498.85	498	519	510.9	497	584	541.15	**481**	515	496.6	**481**	**501**	**493.05**
70×10	601	639	615.4	626	656	640.15	636	716	694.55	**597**	639	621.7	**597**	**635**	**606.75**
80×10	556	590	579	571	596	584.55	561	590	582.75	**548**	586	568.45	554	**575**	**564.85**
80×20	274	285	277.41	269	285	278	273	285	279.9	275	280	**269.3**	**262**	**279**	273.7
90×20	269	290	273.33	275	288	282.1	277	292	285.8	268	285	276.25	**266**	**282**	**272.58**
100×20	304	**323**	313.7	313	327	320.85	317	328	322.85	300	**323**	314.6	**297**	**323**	**310.65**
150×20	472	489	475.34	469	486	476.45	470	488	480.55	469	486	470.7	**464**	**483**	**469.89**
200×20	553	**579**	566.5	564	**579**	573	562	587	578.1	557	583	570.05	**549**	**579**	**565.2**
Average	480.27	506.13	493.45	493.8	516.33	505.83	493.53	524.93	521.73	477.87	508	492.04	**473**	**500.67**	**487.51**

从表 7 中可知, HDTLBO 在绝大部分问题上的测试结果都明显优于其他比较算法, 这验证了 HDTLBO 具有良好的性能. 此外, 和表 6 中各算法比较, HDTLBO 也明显占优. 由于表 6 和表 7 中各算法运行时间相同, 故在 DTLBO 中加入局部搜索的必要性也得以验证.

为了进一步验证各个算法组内差异, 根据 ρ 在 2、4、6 三种取值下整体算法对比结果, 采用 Tukey′s HSD 分析方法对表 7 中 HDTLBO 等 5 种对比算法的 *AVG* 进行方差分析 (ANOVA). 图 7 显示了 5 种算法在不同运行时间下的均值变化线及 95 % 置信度下 Tukey′s HSD 的置信区间, 可以看出, HDTLBO 在均值水平上与其他算法有显著差异, 且随着时间参数 ρ 的变化, HDTLBO 在均值上并无显著差异, 其三个不同时间下的置信区间明显存在重叠部分. 综合算法对比及方差分析结果可知, HDTLBO 能更高效地求解 PMAP_AMPS.

3.4　实际问题求解

在第 3.1 节～第 3.3 节的基础上, 采用云南某模具制造公司生产过程的实例数据, 进一步对 HDTLBO 的性能进行验证. 模具制造需要经过模架加工、模芯加工、电机加工和模具零部件加工等步骤, 其加工工艺主要铸造、切削加工、特种加工三种方法. 切削加工作为最常用方法, 其加工工序主要为粗加工、半精加工、精加工 3 种. 该企业根据不同的客户订单要求进行模具生产, 不同种模具制造过程中有不同的加工工艺路线. 现有 20 种待生产的模具产品, 5 台不同类型 (即刀具、工艺不同) 的加工设备用于生产该订单中所需产品.

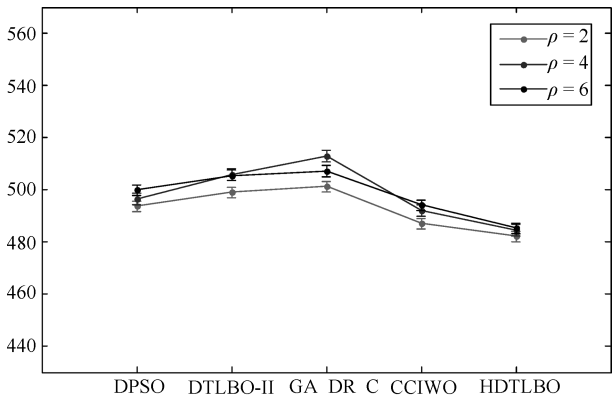


图 7　各算法在不同时间参数下的均值线及 95 % 置信度下 Tukey′s HSD 检验的置信区间

Fig. 7　Means plot and 95 % Tukey′s HSD confidence intervals for the interaction between the algorithms and the different time factor ρ

在对 20 种模具进行生产时, 考虑如下因素: 1) 模具种类不同, 其加工工艺路线也不同, 即在各台设备上工艺操作不同; 2) 加工时, 根据不同模具的工艺要求应选择不同的加工设备, 同种模具在进行不同

表 8 模具各工序加工约束及首次到达时间表

Table 8 The schedule of mold process constraint and arrival time

模具	可执行操作的设备			模具首次到达设备时间				
	操作 1	操作 2	操作 3	m1	m2	m3	m4	m5
1	m1/m3/m5	m3/m4	m2/m3/m4	7	4	8	4	6
2	m1/m3/m6	m1	—	1	6	2	3	3
3	m3	—	—	3	1	8	4	6
4	m2/m3/m5	m1/m2/m3/m4/m5	m2/m3/m4	8	1	6	8	7
5	m1/m3/m4	—	—	1	2	6	7	3
6	m2/m5	m1/m4/m5	m1/m4/m5	5	8	3	1	5
7	m2/m4/m5	m2/m3/m4/m5	—	3	8	3	3	6
8	m1/m2/m3/m5	—	—	2	6	8	7	7
9	m4/m5	m3	m3/m4	6	3	5	3	7
10	m2	m2/m4	m1/m2/m3	7	4	1	4	6
11	m1/m5	m1	—	6	5	1	8	4
12	m4/m5	m1/m2/m3/m4/m5	m1/m2/m4	5	3	1	8	8
13	m2/m4/m5	m3	m4/m5	7	1	6	6	6
14		m3	m1/m3/m5	2	1	4	6	3
15	m3/m4/m5	—	—	8	6	5	3	2
16	m1/m3	m3	—	8	6	6	7	2
17	m3	—	—	8	4	6	7	1
18	m4/m5	m2/m4	m5	4	3	5	2	6
19	m1/m2/m4/m5	—	—	3	6	2	1	3
20	m1/m4	m3/m5	m2/m4/m5	4	7	5	7	7

表 9 各模具产品的序相关设置时间表

Table 9 The schedule of sequence setup time between each mold

模具	序相关设置时间																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	—	2	3	3	4	3	3	3	3	2	1	3	1	4	4	4	2	4	2	1
2	2	—	2	2	3	2	2	1	3	1	2	2	4	4	3	3	4	4	2	4
3	2	3	—	4	3	3	4	2	1	2	3	3	1	4	2	1	3	3	4	3
4	3	1	3	—	4	1	4	2	2	2	1	1	4	1	3	2	2	1	2	3
5	4	4	4	2	—	1	1	1	4	1	1	1	3	2	1	3	2	4	3	3
6	1	2	3	2	3	—	3	2	2	3	4	1	1	3	3	2	4	2	4	1
7	3	4	4	4	1	2	—	1	1	4	2	2	3	2	2	2	4	1	1	2
8	4	2	4	1	4	1	2	—	4	3	3	3	2	4	1	3	4	1	4	1
9	4	2	3	3	2	4	4	1	—	2	2	3	2	1	2	4	2	1	4	3
10	2	4	1	2	2	4	4	3	3	—	1	1	2	3	1	2	1	4	2	1
11	3	4	3	2	2	2	1	2	2	2	—	2	4	3	3	2	1	2	2	2
12	2	1	4	2	4	2	3	2	1	2	3	—	1	1	3	2	3	2	1	2
13	2	1	2	4	2	1	4	2	4	2	4	3	—	4	4	3	4	3	4	1
14	4	4	1	3	2	2	2	4	1	4	1	2	1	—	2	2	1	3	4	1
15	3	2	2	3	1	2	3	4	3	1	2	1	3	1	—	2	3	1	4	4
16	1	4	1	1	4	3	1	3	1	2	4	4	2	3	4	—	4	4	4	1
17	4	3	1	1	3	2	3	1	3	4	1	1	3	2	1	1	—	2	1	1
18	1	2	4	4	2	1	4	1	2	1	4	4	3	2	1	3	1	—	2	2
19	2	3	2	2	2	3	4	1	1	3	2	2	2	3	3	2	4	1	—	2
20	2	2	2	2	2	2	2	1	2	1	2	4	1	4	1	1	1	4	1	—

表 10　模具各工序的加工时间表

Table 10　The processing time table of each mold

模具	操作 1					操作 2					操作 3				
	m1	m2	m3	m4	m5	m1	m2	m3	m4	m5	m1	m2	m3	m4	m5
1	27	—	18	—	29	—	—	27	17	—	—	23	29	19	—
2	13	—	26	—	11	13	—	—	—	—	—	—	—	—	—
3	—	—	23	—	—	—	—	—	—	—	—	—	—	—	—
4	—	10	13	—	12	18	25	11	10	26	—	21	31	27	—
5	9	—	8	16	—	—	—	—	—	—	—	—	—	—	—
6	—	9	—	—	29	19	—	—	25	10	30	—	—	26	19
7	—	19	—	22	28	—	10	28	24	27	—	—	—	—	—
8	30	23	28	—	17	—	—	—	—	—	—	—	—	—	—
9	—	—	—	31	24	—	—	9	—	—	—	—	8	28	—
10	—	25	—	—	—	—	27	—	13	—	31	20	26	—	—
11	25	—	—	—	24	12	—	—	—	—	—	—	—	—	—
12	—	—	—	23	—	17	13	16	—	—	22	8	—	30	—
13	—	16	—	19	22	—	—	28	—	—	—	—	—	27	23
14	28	—	16	—	20	—	—	18	—	—	16	—	16	—	21
15	—	—	28	29	23	—	—	—	—	—	—	—	—	—	—
16	27	—	10	—	—	—	—	8	—	—	—	—	—	—	—
17	—	—	26	—	—	—	—	—	—	—	—	—	—	—	—
18	—	—	—	12	24	—	13	—	27	—	—	—	—	—	17
19	24	19	—	12	14	—	—	—	—	—	—	—	—	—	—
20	17	—	—	10	—	—	—	31	—	20	—	30	—	9	24

表 11　实例仿真结果

Table 11　Simulation results of the instance

运行次数	DTLBO-II	GA	DPSO	CCIWO	HDTLBO
1	167	166	166	**163**	**163**
2	164	166	164	**163**	166
3	168	167	167	167	167
4	169	166	164	166	**163**
5	**163**	**163**	165	**163**	165
6	167	165	166	167	**163**
7	167	168	**163**	168	166
8	166	168	167	164	168
9	168	167	164	**163**	**163**
10	169	168	165	164	165
11	168	**163**	164	168	166
12	169	168	169	167	**163**
13	168	167	165	166	167
14	**163**	170	166	170	166
15	167	166	168	169	**163**
16	166	167	166	171	**163**
17	167	170	**163**	168	**163**
18	169	168	164	170	165
19	165	168	**163**	165	168
20	167	168	164	167	**163**
Average	166.85	166.95	165.15	166.45	**164.8**

工艺加工时需要选择不同的加工设备; 3) 由于工序更换, 加工设备需要一定的设置时间进行更换夹具等所需动作; 4) 根据客户的实时订单要求, 将模具毛坯从仓库中心运送到加工设备上需要可考虑为工件首次到达时间; 5) 模具加工时间 $P_{ij}(k)$ (表示第 i 种模具毛坯料的第 j 个工艺操作在设备 k 上的加工时间) 取决于加工设备所处的工艺步骤. 设有 20 种待加工的模具产品, 工序数为 S_t = [3, 2, 1, 3, 1, 3, 2, 1, 3, 3, 2, 3, 3, 3, 1, 2, 1, 3, 1, 3], 模具加工约束、首达时间 (min) 如表 8 所示, 不同模具产品间的设置时间 (min) 如表 9 所示, 各工序在各台设备上的加工时间 (min) 如表 10 所示.

对于上述实例, 分别使用表 7 中各算法 (即 DPSO、DTLBO-II、GA_DR_C、CCIWO、HDTLBO) 进行求解, 设定算法运行时间均为 $n \times m \times 40$ (ms) 即 4 秒. 对于上述 5 种算法分别独立运行 20 次, 每次运行结果如表 11 所示.

由表 11 可见, HDTLBO 在 20 次独立运行中 9 次找到了最大完工时间 $C_{\max}$ 为 163 的最小解, 明显大于其他 4 种算法找到最小解的次数, 且 HDTLBO 运行 20 次的均值 Average = 164.8 小于其他算法的均值, 这也再次验证了 HDTLBO 具有良好的搜索性能.

综上可知, HDTLBO 能十分有效地求解 PMSP_AMPS.

4 结论

本文针对一类复杂并行机调度问题, 即带到达时间、多工序、加工约束和序相关设置时间的并行机调度问题 (PMSP_AMPS), 建立了排序模型并提出一种混合离散教与学优化算法 (HDTLBO) 进行求解. 这是首次运用基于 TLBO 的算法求解 PMSP_AMPS. 在 HDTLBO 的全局搜索部分, 设计了不同的排序操作, 以此替换标准教与学算法的个体更新操作, 实现了算法的离散化, 使其能直接在问题的离散解空间中执行搜索, 从而在保留 TLBO 更新机理的前提下提高了算法的全局搜索效率. 在局部搜索部分, 构造了基于 *Interchange* 和 *Insert* 邻域操作的变邻域局部搜索, 对全局搜索得到的优质区域执行较为细致的搜索, 使算法在全局和局部搜索之间达到较好平衡, 从而增强了算法的整体搜索能力. 通过不同测试问题上的仿真实验和算法比较, 验证了 HDTLBO 是求解 PMSP_AMPS 的有效算法. 现有的智能调度算法, 大多基于在连续优化领域取得成功应用的智能优化算法, 并在其中加入编码规则来实现解连续空间到问题离散空间的映射, 从而可求解相关调度问题. 这一方式本质上是否有效, 尚无人进行探讨和分析. 本文所提 HDTLBO 在保留标准 TLBO 进化框架基础上, 直接设计各种排序操作来替换原算法的核心操作, 取得了更好的搜索性能. 这对其他智能调度算法的离散化设计具有一定的借鉴意义, 也有利于理解智能调度算法为何有效的本质原因. 下一步的工作将把 HDTLBO 拓展用于求解智能制造中的绿色生产调度问题.

References

1 Pinedo M. Scheduling: theory, algorithms, and systems. Berlin Heidelberg: Springer, 2012.

2 Allahverdi A. Two-machine proportionate flowshop scheduling with breakdowns maximum lateness. *Computers & Operations Research*, 1996, **23**(10): 909−916

3 Lan X J, Su D D. Research of a mold job shop scheduling optimization based on particle swarm optimization algorithm. *Applied Mechanics & Materials*, 2015, **757**(2): 201−207

4 Sang H Y, Duan P Y, Li J Q. An effective invasive weed optimization algorithm for scheduling semiconductor final testing problem. *Swarm & Evolutionary Computation*, 2018, **38**: 42−53

5 Hu H, Ng KKH, Qin Y C. Robust parallel machine scheduling problem with uncertainties and sequence-dependent setup time. *Scientific Programming*, 2016: 1−13

6 Li C L, Cheng T C E. The parallel machine min-max weighted absolute lateness scheduling problem. *Naval Research Logistics*, 1994, **41**(1): 33−46

7 Ranjbar M, Davari M, Leus R. Two branch-and-bound algorithms for the robust parallel machine scheduling problem. *Computers & Operations Research*, 2012, **39**(7): 1652−1660

8 Lee W C, Wang J Y, Lin M C. A branch-and-bound algorithm for minimizing the total weighted completion time on parallel identical machines with two competing agents. *Knowledge-Based Systems*, 2016, **105**(C): 68−82

9 Wu L, Wang S. Exact and heuristic methods to solve the parallel machine scheduling problem with multi-processor tasks. *International Journal of Production Economics*, 2018, **201**: 26−40

10 Chen Z L, Powell W B. A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. *European Journal of Operational Research*, 1999, **116**(1): 220−232

11 Yin Y, Ye D, Zhang G. Single machine batch scheduling to minimize the sum of total flow time and batch delivery cost with an unavailability interval. *Information Sciences*, 2014, **274**(8): 310−322

12 Li D W, Lu X W. Two-agent parallel-machine scheduling with rejection. *Theoretical Computer Science*, 2017, **703**: 66−75

13 Yin Y, Cheng S R, Cheng T C E, et al. Just-in-time scheduling with two competing agents on unrelated parallel machines. *Omega*, 2016, **63**: 41−47

14 Woo Y B, Jung S, Kim B S. A rule-based genetic algorithm with an improvement heuristic for unrelated parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities. *Computers & Industrial Engineering*, 2017, **109**: 179−190

15 Avalos-Rosales O, Angel-Bello F, Alvarez A. Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *International Journal of Advanced Manufacturing Technology*, 2015, **76**(9−12): 1705−1718

16 Zhang Jia-Qi, Cao Zheng-Cai, Liu Min. Integrated differential evolution algorithm with surrogate model for dynamic parallel machine scheduling. *Computer Integrated Manufacturing Systems*, 2017, **23**(1): 75−81
(张嘉琦, 曹政才, 刘民. 融合代理模型和差分进化算法的并行机动态调度方法. 计算机集成制造系统, 2017, **23**(1): 75−81)

17 Chen C L. Iterated hybrid metaheuristic algorithms for unrelated parallel machines problem with unequal ready times and sequence-dependent setup times. *International Journal of Advanced Manufacturing Technology*, 2012, **60**(5−8): 693−705

18 Damodaran P, Diyadawagamage D A, Ghrayeb O. A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines. *International Journal of Advanced Manufacturing Technology*, 2012, **58**(9−12): 1131−1140

19 Diana R O M. An immune-inspired algorithm for an unrelated parallel machines′ scheduling problem with sequence and machine dependent setup-times for makespan minimisation. *Neurocomputing*, 2015, **163**(C): 94−105

20 Sels V, Coelho J, Dias A M, et al. Hybrid tabu search and a truncated branch-and-bound for the unrelated parallel machine scheduling problem. *Computers & Operations Research*, 2015, **53**: 107−117

21 Bitar A, Yugma C, Roussel R. A memetic algorithm to solve an unrelated parallel machine scheduling problem with auxiliary resources in semiconductor manufacturing. *Journal of Scheduling*, 2016, **19**(4): 367−376

22 Joo C M, Kim B S. Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability. *Computers & Industrial Engineering*, 2015, **85**(C): 102−109

23 Luo Jia-Xiang, Tang Li-Xin. A new ILS & SS algorithm for parallel-machine scheduling problem. *Acta Automatica Sinica*, 2005, **31**(6): 917−924
(罗家祥, 唐立新. 带释放时间的并行机调度. 问题的 ILS & SS 算法. 自动化学报, 2005, **31**(6): 917−924)

24 Lin S W, Ying K C. ABC-based manufacturing scheduling for unrelated parallel machines with machine-dependent and job sequence-dependent setup times. *Computers & Operations Research*, 2014, **51**(3): 172−181

25 Rao R V, Savsani V J. and Vakharia D P. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 2011, **43**(3): 303−315

26 Waghmare, G. Comments on "a note on teaching-learning-based optimization algorithm". *Information Sciences*, 2013, 229: 159−169

27 Shao S W, Pi D C, Shao Z. An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem. *Applied Soft Computing*, 2017, **61**: 193−210

28 Xu Y, Wang L, Wang S Y, et al. An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time. *Neurocomputing*, 2015, 148: 260−268

29 Shao S W, Pi D C, Shao Z. A hybrid discrete optimization algorithm based on teaching-probabilistic learning mechanism for no-wait flow shop scheduling. *Knowledge-Based Systems*, 2016, **107**: 219−234

30 Li J Q, Pan Q K, Mao K. A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems. *Engineering Applications of Artificial Intelligence*, 2015, **37**: 279−292

31 Ma Yong-Jie, Yun Wen-Xia. Research progress of genetic algorithm. *Application Rehash of Computers*, 2012, **29**(4): 1201−1206
(马永杰, 云文霞. 遗传算法研究进展. 计算机应用研究, 2012, **29**(4): 1201−1206)

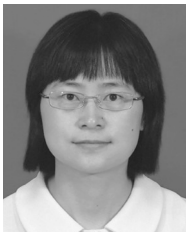
32 Abdoun O, Abouchabaka J. A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem. *Computer Science*, 2011, **31**(11): 49−57

33 Wang Ling, Qian Bin. Hybrid differential evolution and scheduling algorithms. Beijing: Tsinghua University Press, 2012.
(王凌, 钱斌. 混合差分进化与调度算法. 北京: 清华大学出版社, 2012.)

34 Pan Quan-Ke, Wang Ling, Zhao Bao-Hua. Discrete particle swarm optimization for no-idle flow shop problem. *Control and Decision*, 2008, **23**(2): 191−194
(潘全科, 王凌, 赵保华. 解决零空闲流水线调度问题的离散粒子群算法. 控制与决策, 2008, **23**(2): 191−194)

何雨洁 昆明理工大学信息工程与自动化学院硕士研究生. 2016 年获得昆明理工大学信息工程与自动化学院自动化系学士学位. 主要研究方向为调度与智能优化算法.
E-mail: Hyujie_one@163.com
(**HE Yu-Jie** Master student at the School of Information Engineering and Automation, Kunming University of Science and Technology. She received her bachelor degree from Kunming University of Science and Technology in 2016. Her research interest covers scheduling and intelligent optimization algorithms.)

钱 斌 昆明理工大学信息工程与自动化学院教授. 2009 年获得清华大学自动化系博士学位. 主要研究方向为调度与优化. 本文通信作者.
E-mail: bin.qian@vip.163.com
(**QIAN Bin** Professor at the School of Information Engineering and Automation, Kunming University of Science and Technology. He received his Ph. D. degree from Tsinghua University in 2009. His research interest covers scheduling and optimization. Corresponding author of this paper.)



胡 蓉 昆明理工大学信息工程与自动化学院副教授. 2004 年获得清华大学自动化系硕士学位. 主要研究方向为优化方法和决策支持系统.
E-mail: ronghu@vip.163.com
(**HU Rong** Associate professor at the School of Information Engineering and Automation, Kunming University of Science and Technology. She received her master degree from Tsinghua University in 2004. Her research interest covers optimization methods and decision support systems.)