DOI: 10.3901/JME.2021.17.263

求解分布式柔性作业车间调度的混合蛙跳算法^{*}

孟磊磊¹ 张彪¹ 任亚平² 张超勇³

(1. 聊城大学计算机学院 聊城 252059;

2. 暨南大学智能科学与工程学院 珠海 519070;

3. 华中科技大学数字制造装备与技术国家重点实验室 武汉 430074)

摘要: 针对分布式柔性作业车间(Distributed flexible job shop scheduling problem, DFJSP)最小化最大完工时间问题, 提出一种混合蛙跳算法。在该算法中, 编码采用基于工序序列和工厂序列的部分解空间编码方案, 机床选择在解码过程中通过规则确定。引入变邻域搜索算法提升蛙跳算法的局部搜索能力。为了弥补部分解空间编码不能探索整个解空间的缺点, 引入针对关键工厂的全解空间禁忌搜索, 从而扩大算法解空间、进一步提升算法的局部搜索能力。通过对基准实例的求解, 并与现有的最先进算法进行对比, 验证了所提算法的有效性和优越性。

关键词: 分布式调度; 柔性作业车间; 蛙跳算法; 最大完工时间

中图分类号: TP18

Hybrid Shuffled Frog-leaping Algorithm for Distributed Flexible Job Shop Scheduling

MENG Leilei¹ ZHANG Biao¹ REN Yaping² ZHANG Chaoyong³

(1. School of Computer Science, Liaocheng University, Liaocheng 252059;

2. School of Intelligent Systems Science and Engineering, Jinan University, Zhuhai 519070;

3. State Key Lab of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074)

Abstract: Aiming at the distributed flexible job shop scheduling problem (DFJSP) with minimizing makespan, a hybrid shuffled frog-leaping algorithm (HSFLA) is proposed. In this algorithm, the encoding adopts the scheme of partial solution space based on the operation and factory sequences, and the machine selection is determined in the decoding process with a rule. A variable neighborhood search (VNS) algorithm is introduced to improve the local search ability of HSFLA. To make up for the shortcoming that the partial space encoding method cannot search the full solution space, a tabu search algorithm of full solution space is introduced in critical factory to expand the solution space and further improve the local search ability of HSFLA. By solving benchmark instances and comparing HSFLA with the existing state-of-the-art algorithms, the effectiveness and superiority of HSFLA are proved.

Key words: distributed scheduling; flexible job shop; shuffled frog-leaping algorithm; makespan

0 前言

随着经济全球化不断发展,制造业也逐步全球化,制造企业正面临前所未有的竞争。在此背景下,为了提高企业竞争能力,需要从传统单一工

厂生产模式向多工厂生产模式进行转换^[1-4]。企业可以综合利用分布在不同地方的工厂,从而提高生产效率、生产稳定性,降低成本,进而提升企业竞争力。

柔性作业车间调度问题(Flexible job shop scheduling problem, FJSP)广泛存在于生产实践中,其需要解决工序排序和机床选择两个子问题。与FJSP相比,分布式柔性作业车间(Distributed flexible

^{*} 国家自然科学基金资助项目(51861165202, 51575211, 51705263)。20200905 收到初稿, 20201105 收到修改稿

job shop scheduling problem, DFJSP)考虑多个工厂\车间,每个工厂\车间是一个 FJSP 问题。因此,DFJSP 要解决工厂选择、机器选择以及工序排序等 3 个子问题。

针对 DFJSP 的求解方法主要分为精确算法和近似算法两种。其中,精确算法的求解时间随问题规模的增大而剧增,不适合求解较大规模问题,实际应用较少^[5-7]。近似算法包括智能优化算法以及规则方法,求解效率高,得到了广泛的应用^[6, 8-13]。香港大学的 CHAN 等^[11]首次研究了 DFJSP 问题,以最小化最大完工时间为目标,提出了基于主导基因的遗传算法(GADG),构建了针对 DFJSP 问题的全解空间编码(SCH)。SCH 包括工厂、机器、工件、工序以及主导基因 5 部分信息。CHAN 等^[10]研究了考虑机床维修的 DFJSP 问题,将维修信息集成在编码中。GIOVANNI 和 PEZZELLA^[12]提出了一种改进的遗传算法(IGA),并且设计了一种部分解空间编码(SGP)。SGP 只包括工厂和工序信息,机器选择在解码过程中通过规则来确定。并且,作者设计了针对关键工厂的局部搜索算法以提高遗传算法的局部寻优能力。MOHSEN^[14]针对 DFJSP 问题特性,设计了一种快速启发式规则。LU 等^[15]提出了一种基于工件编码方法(SJS)的遗传算法(GA_JS),SJS 中只包含工件信息,工厂选择、机器选择以及工序排序 3 个子问题在解码中通过规则确定。CHANG 等^[16]提出了一种基于概率编码的全解空间编码(SCL)的混合遗传算法(GA_CL)。WU 等^[17]对现有 SCH、SGP、SJS、SCL 等 4 种解码方法进行了分析,提出了基于工序序列的编码(SOP),即编码序列只有一条工序序列,工厂选择以及机床选择通过使用相应的解码规则确定。LIN 等^[18]分析了全解空间编码与部分解空间编码的优缺点,针对考虑机床维修的 DFJSP 问题,提出了两种部分解空间编码方法,编码序列只包括工序排序以及维修信息,工厂选择与机器选择通过解码规则确定。

除使用遗传算法求解 DFJSP 外,黄英杰等^[19]提出了一种基于目标级联法和粒子群算法的混合算法,杨江波等^[13]提出了一种基于目标级联和遗传算法的求解方法。吴锐等^[9]提出了一种改进的人工蜂群算法,并设计了基于工序、机器以及工厂的三层编码方案。吴秀丽等^[8]提出了一种改进的多目标混合差分进化算法,同时优化 DFJSP 总成本和提前\延期惩罚。

蛙跳算法最早由 MUZAFFAR 和 KEVIN^[20]提

出,其通过模拟自然界中青蛙在觅食过程中种群所体现出的交流与合作行为,具有控制参数少、流程简单、易于实现等优点。近年来,蛙跳算法逐步应用于求解车间调度问题,如流水车间^[21]、作业车间^[22]以及柔性作业车间^[23],并取得不错的效果。然而,针对本文 DFJSP,并未看到相关研究。因此,本文将蛙跳算法应用于求解 DFJSP,并根据 DFJSP 特性,设计蛙跳算法的操作算子。针对 FJSP 问题,多个基于问题特性的局部算法已被提出,然而针对 DFJSP 问题,相应的研究还比较匮乏。因此,我们提出针对 DFJSP 问题的高效局部搜索策略,提升算法的局部搜索能力。本文的创新点主要为:(1)设计了一种变邻域搜索算法提升蛙跳算法的局部搜索能力;(2)为了弥补部分解空间编码不能探索整个解空间的缺点,引入针对关键工厂的全解空间禁忌搜索,从而扩大算法解空间、进一步提升算法的局部搜索能力。

1 分布式柔性作业车间调度

1.1 问题描述

DFJSP 可描述如下:给定的 n 个加工工件,在 N_f 个工厂中加工,每个工厂有 m_f 台机床。每个工件 i 可选择在任一工厂加工。并且,每个工件 i 包含 n_i 道工序,工序 $O_{i,j}$ 可选在工厂 f 中的 $m_{i,j,f}$ 个加工机床上加工。工件加工完成后运输到 I/O 中心需要一定的时间。该问题需要满足如下约束条件。

(1) 初始状态约束:所有机床 0 时刻可用,所有工件 0 时刻可加工。

(2) 机器占用约束:每台机床在同一时刻最多只能加工一道工序。

(3) 工件加工约束:任一工件在任一时刻只能在一台机床上加工,同一工件的不同工序按照给定的先后顺序进行加工,不同工件的工序之间是独立的,没有加工先后约束。

(4) 工序连续加工约束:任一工序一旦开始加工不允许中断,即为非抢占式加工。

(5) 工厂选择约束:任一工件只能选择在一个工厂中加工,即同一工件的不同工序必须在同一工厂中加工。

(6) 其他约束:忽略同一机床上不同工件间的转换时间以及同一工件在不同机床间的运输时间,加工时间等信息都是给定的。

本文以最小化最大完工时间为目标,即以所有工厂最大完工时间的最大值最小化为目标。

1.2 符号说明

本文 DFJSP 问题建模所用到的符号说明如下。

表 1 DFJSP 问题建模符号含义

符号	含义
i, i'	工件索引
n	工件总数
I	工件集合, $I=\{1,2,\dots,n\}$
j, j'	工序索引
n_i	工件 i 的工序数
J_i	工件 i 的工序集合, $J_i=\{1,2,\dots,n_i\}$
J'_i	工件 i 的前 n_i-1 道工序集合, $J'_i=\{1,2,\dots,n_i-1\}$
$o_{i,j}$	工件 i 的第 j 道工序
f	工厂索引
N_f	工厂总数
F	工厂集合, $F=\{1,2,\dots,N_f\}$
K	机床索引
m_f	工厂 f 中的机床总数
K_f	工厂 f 的机床集合, $K_f=\{1,2,\dots,m_f\}$
$m_{i,j,f}$	在工厂 f 中, 可加工工序 $o_{i,j}$ 的机床总数
$K_{i,j,f}$	在工厂 f 中, 可加工工序 $o_{i,j}$ 的机床集合
$pt_{i,j,f,k}$	工序 $o_{i,j}$ 在机器 k 上的加工时间
M	一个极大的正数
$x_{i,j,f,k}$	0-1 常量, 当 $x_{i,j,f,k}=1$ 时, 表示工序 $o_{i,j}$ 可以选择在工厂 f 中机床 k 上加工, 否则 $x_{i,j,f,k}=0$
$Tr_{i,f}$	将工件 i 从工厂 f 运送到 I/O 中心所需要的时间, 为建模简单, 本文将 $Tr_{i,f}$ 合并到工件的最后一道工序加工时间 $pt_{i,n_i,f,k}$ 内
决策变量	
$X_{i,j,f,k}$	0-1 决策变量, 当工序 $o_{i,j}$ 选择在工厂 f 中的机床 k 上加工时, $X_{i,j,f,k}=1$, 否则 $X_{i,j,f,k}=0$
$Z_{i,f}$	0-1 决策变量, 如果工件 i 选择在工厂 f 中加工, 则 $Z_{i,f}=1$, 否则 $Z_{i,f}=0$
$Y_{i,j,j',f}$	0-1 决策变量, 当工序 $o_{i,j}$ 在工序 $o_{i,j'}$ 之前加工时, $Y_{i,j,j',f}=1$, 否则 $Y_{i,j,j',f}=0$
$B_{i,j}$	连续决策变量, 表示工序 $o_{i,j}$ 的开始加工时间
$C_{\max}$	连续决策变量, 表示最大完工时间

1.3 DFJSP 数学模型

DFJSP 问题的数学模型如下所述, 该模型是基于工序先后关系的建模思想, 通过确定两工序(相邻或者不相邻)的先后顺序来实现工序排序目的^[5, 6]。

0-1 决策变量 $Y_{i,j,i',j'}$ 、 $Z_{i,f}$ 以及 $x_{i,j,f,k}$ 分别用来确定工序先后关系、工厂选择以及机床选择子问题。

$$\min C_{\max} \quad (1)$$

$$\sum_{f \in F} Z_{i,f} = 1, \forall i \in I \quad (2)$$

$$Z_{i,f} = \sum_{k \in K_{i,j,f}} X_{i,j,f,k}, \forall i \in I, j \in J_i, f \in F \quad (3)$$

$$B_{i,j} + \sum_{f \in F} \sum_{k \in K_{i,j,f}} (pt_{i,j,f,k} X_{i,j,f,k}) \leq B_{i,j+1}, \forall i \in I, j \in J'_i \quad (4)$$

$$B_{i,j} + pt_{i,j,f,k} X_{i,j,f,k} \leq B_{i',j'} + M(3 - Y_{i,j,i',j'} - X_{i,j,f,k} - X_{i',j',f,k}) \\ \forall i \in I, i' \in I, i < i', j \in J_i, j' \in J_{i'}, f \in F, k \in K_{i,j,f} \cap K_{i',j',f} \quad (5)$$

$$B_{i',j'} + pt_{i',j',f,k} X_{i',j',f,k} \leq B_{i,j} + M(2 - Y_{i,j,i',j'} - X_{i,j,f,k} - X_{i',j',f,k}) \\ \forall i \in I, i' \in I, i < i', j \in J_i, j' \in J_{i'}, f \in F, k \in K_{i,j,f} \cap K_{i',j',f} \quad (6)$$

$$C_{\max} \geq B_{i,n_i} + \sum_{f \in F} \sum_{k \in K_{i,n_i,f}} (pt_{i,n_i,f,k} X_{i,n_i,f,k}), \forall i \in I \quad (7)$$

$$B_{i,j} \geq 0, \forall i \in I, j \in J_i \quad (8)$$

其中, 式(1)表示目标函数, 式(2)表示任一工件只能选择在一个工厂中加工, 式(3)表示同一工件的所有工序必须在同一工厂中加工, 式(4)用来约束同一工件不同工序间的先后关系, 对偶约束(5)~(6)确保同一机器上不同工序间加工的非重叠性, 式(7)表示最大完工时间约束, 式(8)表示工序开始时间的非负性。

2 混合蛙跳算法求解 DFJSP

2.1 蛙跳算法介绍

蛙跳算法是基于种群进化的元启发式算法之一, 通过模拟自然界中青蛙觅食过程中种群所体现出的交流与合作行为, 以实现对问题的求解^[20]。在一片湿地中, 分布着一群青蛙, 每只青蛙有自己的想法, 每只青蛙的想法则被定义为一个解。每只青蛙找到食物时, 都会向其周围的青蛙传递信息, 从而影响周围青蛙向其靠近, 形成一个子种群(模因组)。可见, 湿地中青蛙种群将形成多个这样的子种群。子种群内的青蛙相互影响, 从而实现组内的局部搜索。当青蛙跳跃一定次数后, 为了寻找更多的食物, 所有青蛙将重新混合后再次形成子种群, 寻找更多的食物。至此, 完成种群的一次全局搜索。子种群局部搜索与整个种群全局搜索交替进行, 从而实现对问题的求解。

基本蛙跳算法分为初始化、模因组划分、模因组进化、模因组合并 4 个操作。其中, 初始化用来确定算法可选参数以及初始化种群, 模因组划分将所有青蛙按照一定规则分为多个模因组, 模因组进化为实现模因组内青蛙个体的交流与进化, 模因组合并为实现整个青蛙种群的信息交换。

2.2 编码与解码

针对 DFJSP, 目前主要有 4 种离散编码方法。其中, 第一种为基于工件序列的编码(SJS), 由 LU 等^[15]提出, 编码中只有工件序列, 工序排序、机器选择以及工厂选择 3 个子问题都在解码过程中分别通过相应的规则确定。可见, SJS 通过解码过程实现从一维编码空间到三维解码空间的转换。工厂选

择、工序排序以及机器选择分别基于工厂最大完工时间最小规则(FMM)、最长剩余加工时间优先(LWKR)规则以及最小负载机床优先(MLM)规则确定。

第二种为基于工序序列的编码(SOP), 由 WU 等^[17]所提出。SOP 中只有一条工序序列, 用来确定工序的先后顺序, 工厂选择以及机器选择在解码过程中通过规则方法进行确定。

第三种为基于工序和工厂序列的编码(SGP), SGP 可确定工序排序和工厂选择问题, 机器选择在解码中通过当前最大完工时间最小化的规则(MCM)确定^[12]。MCM 规则如下: 从工序序列的第一个工序开始, 依次将其安排到使所在工厂当前最大完工时间最小的机床上。工厂当前最大完工时间是指安排完当前工序后工厂的最大完工时间。例如: 某工厂包括两个机床, M1 和 M2 的当前完工时间分别为 5 和 3, 工序 O 在 M1 和 M2 上的加工时间分别为 5 和 8。如果工序 O 分配到 M1, 则 M1 的完工时间增大为 10, 此时工厂的当前完工时间为 $\max\{10, 8\}=10$; 如果分配到 M2, 则 M2 的完工时间增大为 11, 此时工厂的当前完工时间为 $\max\{5, 11\}=11$ 。因此, 工序 O 将选择使工厂当前完工时间为较小值 10 的机床 M1 上。图 1 给出了 MCM 规则实例示意图。

图 1 MCM 解码实例示意图

图 2 给出了以上三种部分解空间编码的实例。部分解空间编码只能确定 DFJSP 部分子问题, 其他子问题需要在解码过程中使用规则方法确定, 从而称之为部分解空间编码。

第四种编码包含工序序列、机器序列和工厂序列, 可见工序排序、机床选择和工厂选择 3 个子问题都由编码确定, 为全解空间编码^[9]。解码可使用 FJSP 问题中的半主动解码、主动解码与全主动解码方法。图 3 给出了全解空间编码实例。

不同编码方法对算法性能影响较大。虽然部分解空间编码有可能不能遍历所有解空间, 但是采用该编码方案的算法收敛性、稳定性往往较好。全解空间编码虽然保证了能够搜索整个解空间的可能性, 但是, 由于解空间极大, 如果算法设计不理想, 算法结果不能保证。

图 2 SJS、SOP 和 SGP 部分解空间编码

图 3 全解空间编码

为了综合考虑算法收敛性以及解空间, 本文采用类似 SGP 的部分解空间编码。编码中只考虑图 3 中的工序序列和工厂序列, 工序序列长度为所有工件总工序数, 工厂序列长度为工件数。机床选择通过使用 MCM 规则确定。为了弥补部分解空间编码不能遍历整个解空间的劣势, 通过引入针对 FJSP 问题的禁忌搜索算法 MG-TS^[24], 弥补 MCM 规则确定机器选择的不足, 从而有利于探索整个解空间, 进而寻得更好解。

2.3 初始化

针对所有个体, 工序序列随机初始化, 50% 个体的工厂选择采用 FMM 规则确定以保证各个工厂负载均衡, 剩下 50% 个体的工厂选择随机初始化, 每个工序的机器选择在解码过程中通过 MCM 规则确定。FMM 规则首先通过工序序列来确定工件序列; 然后, 依据工件的所有工序在工厂机床的平均加工时间, 按照类似 MCM 规则依次尝试将工件分配到每个工厂, 选择当前完工时间最小的工厂。其中, FMM 规则具体流程参照文献^[17]。

2.4 模因组划分

针对模因组划分, 首先, 对整个种群个体按照最大完工时间从好到差进行排序。然后, 按照以下规则进行模因组划分: 第 1 个青蛙划分到第 1 个模型组, 第 2 个青蛙划分到第 2 个模型组, ..., 第 $N_p + 1$ 个青蛙划分到第 1 个模因组, 第 $N_p + 2$ 个青蛙划分到第 2 个模因组, 以此类推, 直至所有青蛙被分到 N_m 个模因组。

2.5 模因组进化

模因组进化通过全局最好个体、模因组内最好个体和通过二元锦标赛选择的个体的交叉、随机产生解等 3 种方式来更新模因组最差个体。针对每个模因组, 按照最大完工时间的好坏确定种群全局最好解 X_{gb} 、组内局部最好个体 X_b 与局部最差个体 X_w 。然后在该组内按照二元锦标赛原则选择一个个体 X_r , $X_r \neq X_b$, 然后让 X_b 或者 X_{gb} 与 X_r 进行交叉来更新最差个体 X_w 。工序序列使用优先工序交叉算子(Precedence operation crossover, POX)^[25]与两点交叉算子(Two point crossover, TPX), 两种算子随机选择, 从而综合利用两种算子的优势。工厂序列采用均匀交叉算子(Uniform crossover, UX)^[25]。以 30% 的概率选择工厂序列交叉, 以 70% 的概率选择工序序列交叉(35% 概率选择 POX, 35% 概率选择 TPX)。

POX 算子示意图如图 4 所示, 其步骤如下所述。

(1) 随机将所有工件分为两部分 Jset1 和 Jset2。

(2) 将父代 P1 中属于 Jset1 的工序赋值给子代 C1 的相同位置, 将父代 P2 中属于 Jset2 的工序按照先后顺序赋值给子代 C2 的相同位置。

(3) 将父代 P2 中属于 Jset2 的工序按照先后顺序赋值给子代 C1 的其余位置, 将父代 P1 中属于 Jset1 的工序按照先后顺序赋值给子代 C2 的其余位置。

图 4 POX 交叉算子

TPX 算子示意图如图 5 所示, 其步骤如下所述。

(1) 随机产生两随机数 rand1 和 rand2 将工序序列分为两部分, $\text{rand1} < \text{rand2}$ 。

(2) 将父代 P1 中 rand1 之前和 rand2 之后的工序赋值给子代 C1 的相同位置, 将父代 P2 中 rand1 之前和 rand2 之后的工序按照先后顺序赋值给子代 C2 的相同位置。

(3) 将父代 P2 中除步骤 2 中以外的工序按照先后顺序赋值给子代 C1 的其余位置, 得到子代 C1。将父代 P1 中除步骤 2 中以外的工序按照先后顺序赋值给子代 C2 的其余位置, 得到子代 C2。

图 5 两点交叉算子

UX 交叉算子的示意图如图 6 所示, 其操作步骤如下所述。

(1) 随机产生长度为工序总数的 0-1 数列。

(2) 将父代 P1 中对应 0-1 数列为 0 位置的工序赋值给子代 C1, 保持位置不变。同理, 将父代 P2 中对应 0-1 数列为 0 位置的工序赋值给子代 C2, 保持位置不变。

(3) 将父代 P2 中对应 0-1 数列为 1 位置的工序赋值给子代 C1, 保持位置不变。同理, 将父代 P1 中对应 0-1 数列为 1 位置的工序赋值给子代 C2, 保持位置不变。

图 6 均匀交叉算子

模因组进化具体操作步骤如下所述。

(1) 设置 $t=1$, 重复以下步骤(2)~(6)直至 $t > N_t$ 。

(2) 随机选择三种交叉算子种的一个对 X_b 与 X_r 进行交叉, 得到子代个体 X_1^{new} 与 X_2^{new} , 取 X^{new} 为 X_1^{new} 与 X_2^{new} 二者中较好个体。

(3) 如果 X^{new} 好于 X_w , 则 $X_w = X^{new}$, 转至步骤(6); 否则, 转至步骤(4)。

(4) 与步骤(2)类似, 使用 X_{gb} 与 X_r 进行交叉得到较好子个体 X^{new} 。

(5) 如果 X^{new} 好于 X_w , 则 $X_w = X^{new}$, 转至步骤(6); 否则, 按照第 2.3 节的初始化方法随机产生一个解代替 X_w , 转至步骤(6)。

(6) $t=t+1$ 。

2.6 种群多样性检测

当种群最优解 γ 次迭代未得到更新时, 模因组合并之后检查种群多样性。本文两个个体相同条件为:

两个体目标函数值相同、工序的机器选择相同以及工厂序列相同。重复个体工序序列随机初始化,工厂选择按照 FMM 规则确定。

2.7 局部搜索算法

使用两种局部搜索方法,第一种为 VNS,提升蛙跳算法的局部搜索能力;第二种为全解空间禁忌搜索算法(MG-TS),弥补部分解空间编码的劣势,有利于扩大解空间,从而获得更优异的解。因为 DFJSP 的最大完工时间由所有工厂中完工时间最大的工厂(关键工厂)决定。因此,两种局部搜索操作都是针对关键工厂而进行的。针对关键工厂,提出四个邻域结构,其中邻域 1-3 针对工序序列,邻域 4 针对工厂序列,4 个邻域结构如下。

邻域 1: 随机交换关键工厂中的两不同工序(其中一个为关键工序)的位置。

邻域 2: 随机选择关键工厂中的两不同工序(其中一个为关键工序),将位置靠后工序插入到位置靠前工序前边。

邻域 3: 随机选择关键工厂中的两不同工序(其中一个为关键工序),将两工序位置间的工序逆序。

邻域 4: 随机选择关键工厂中的一工件,随机改变其工厂分配。

VNS 搜索过程如下所述。

(1) 初始化。确定初始解 x 、邻域结构 $N_k(x), k=1, \dots, k_{\max}$ 以及停止条件。

(2) 定 $k=1$, 重复步骤(3)~(5)直至 $k > k_{\max}$ 。

(3) 扰动: 随机产生 x 的邻域解 x' , $x' \in N_k(x)$ 。

(4) 针对 x' 使用如下局部搜索算法获得局部最优解 x'_1 :

1) 置 $t=1$, $x'_1 = x'$, 重复以下步骤直至 $t > t_{\text{VNS}}$ 。

2) 产生 x'_1 的邻域解 x'' , $x'' \in N_k(x'_1)$ 。

3) 如果 x'' 好于 x'_1 , 则 $x'_1 = x''$, $t = t + 1$; 否则, $t = t + 1$ 。

(5) 邻域变换: 如果 x'_1 好于 x , 则将 x'_1 赋予 x , $x = x'_1$, 设定 $k=1$; 否则, 设定 $k = k + 1$ 。

对于 MG-TS, 其针对关键工厂中的关键工序进行操作, 其具体算法流程见文献[22]。MG-TS 算法采用基于移动一道关键工序的邻域结构, 针对 FJSP 问题求解效果极好^[23]。

2.8 算法流程

HSFLA 的算法流程依次为初始化、模因组划分、模因组进化、模因组合并、种群多样性检测以及局部搜索操作。局部搜索操作为: 对种群从好到差进行排序, 针对种群中 10% 较好的个体实行 VNS 搜索。为了充分利用基于部分解空间编码算法收敛

快的优点, 并不是每次迭代都执行 MG-TS, 而是在当前最好解连续 γ 次迭代无法更新时才执行。算法停止条件为达到最大求解时间限制或者求得问题下界。算法流程图如图 7 所示。

图 7 HSFLA 算法流程图

3 试验验证

使用 De Giovanni 和 Pezzella 所用 la01-20、mt06、mt10 和 mt20 等 23 组实例, 与现有 IGA、Heu、GA_JS、GA_OP、DTSMA、CRO、IDESAA 以及 IGWO8 种算法进行对比。对比实例分为 2 工厂和 3 工厂, 每个工厂加工信息完全一致。对于 HSFLA 算法参数, 模因组内青蛙数 N_p 从[10,15,20]、模因组个数 N_m 从[5,8,10]、模因组进化次数 N_t 从[3,6,9]、VNS 算法参数 t_{VNS} 从[5,10,20], 算法最大求解时间为 2 nm。通过大量试验证明, 当选择 $N_p = 20$, $N_m = 10$, $N_t = 3$, $t_{\text{VNS}} = 10$ 时, 算法性能较好, 因此采用该参数组合。其他参数设置为: $\gamma = 30$, MG-TS 算法的迭代次数为 5nm。

3.1 局部搜索有效性验证

为了验证所提 VNS 和全解空间 MG-TS 算法的有效性, 针对 2 工厂实例, 表 1 给出了不同算法的对比结果。表 1 中, SFLA 表示不考虑 VNS 以及 MG-TS 的算法, SFLA-VNS 表示只考虑 VNS 的算法, HSFLA 表示同时考虑 VNS 和 MG-TS 的算法。SFLA 以及 SFLA-VNS 算法参数以及停止条件同 HSFLA。每种算法重复运行 20 次, 记录最好值 MK

表 1 SFLA、SFLA-VNS 以及 HSFLA 对比结果

实例	SFLA		SFLA-VNS		HSFLA	
	MK	AV	MK	AV	MK	AV
la01	413	413.0	413	413.0	413	413.0
la02	394	394.0	394	394.0	394	394.0
la03	349	349.0	349	349.0	349	349.0
la04	369	369.0	369	369.0	369	369.0
la05	380	380.0	380	380.0	380	380.0
la06	420	438.0	419	426.3	413	420.4
la07	399	411.0	391	396.6	386	393.0
la08	410	415.4	398	405.7	391	400.4
la09	462	467.0	444	448.7	438	444.6
la10	443	443.0	443	443.0	443	443.0
la11	552	561.2	543	543.6	538	542.6
la12	482	485.6	473	478.9	471	473.2
la13	533	542.8	529	533.5	522	525.8
la14	559	563.0	543	547.6	538	540.4
la15	572	576.3	557	561.3	550	554.5
la16	717	717.0	717	717.0	717	717.0
la17	646	646.0	646	646.0	646	646.0
la18	663	663.0	663	663.0	663	663.0
la19	617	617.0	617	617.0	617	617.0
la20	756	756.0	756	756.0	756	756.0
mt06	47	47.0	47	47.0	47	47.0
mt10	655	655.0	655	655.0	655	655.0
mt20	543	550.0	525	533.6	518	524.2

以及均值 AV。从表 1 中可以看出, SFLA 结果最差, HSFLA 最好, SFLA-VNS 处于中间水平。SFLA-VNS 针对 23 组实例的 MK 以及 AV 都不差于 SFLA, 体现了所设计 VNS 算法的有效性。这是因为通过引入 VNS 作为局部搜索, 增强了 SFLA

的局部搜索能力。针对 23 组实例的 MK 以及 AV, HSFLA 不差于 SFLA-VNS, 体现了加入全解空间 MG-TS 的有效性。这是因为全解空间 MG-TS 可以弥补部分 SFLA-VNS 部分解空间编码、解码的劣势, 从而有利于搜寻整个解空间, 进而获得更加优异的解。

3.2 HSFLA 有效性验证

表 2 和表 3 分别给出了针对 2 工厂以及 3 工厂实例的对比结果。表 2 和 3 中, Best 表示可求最好解的个数, Sum_RPI 表示可求解与下届 LB 相对偏差之和, 相对偏差计算公式为 $RPI=(S-LB)/LB$, S 表示可求解。

从表 2 中可以看出, 本文所提 HSFLA 算法刷新了 la07-09、la11、la13 和 mt20 等实例的现有算法可求最好解。在所有 9 个算法中, 本文算法 Best 最大(为 23), Sum_RPI 最小(为 2.06), 体现了所提算法的优越性。从表 3 中可以看出, 本文所提 HSFLA 算法除实例 la15 的求解结果 390 略差于 IDESAA 的结果 389 外, 可求得其他所有实例的最好解。针对 Best 指标, 本文 HSFLA、IDESAA 以及 IGWO 都为 22, 即都可求得 22 组实例的最好解。其中, IDESAA 不可求得 la13 的最好解, IGWO 不可求得 la15 的最好解。对于 Sum_RPI 指标, 本文 HSFLA 算法最小(为 0.03), 好于其他对比算法, 体现了其优越性。

表 2 2 工厂实例测试结果

实例	LB	IGA	Heu	GA_JS	GA_OP	DTSMA	CRO	IDESAA	IGWO	HSFLA
la01	413	413	413	413	413	413	413	413	413	413
la02	394	394	394	394	394	394	394	394	394	394
la03	349	349	349	349	349	349	349	349	349	349
la04	369	369	369	369	369	369	369	369	369	369
la05	380	380	380	380	380	380	380	380	380	380
la06	413	445	463	421	424	424	424	419	413	413
la07	376	412	417	396	390	398	398	392	389	386*
la08	369	420	436	406	397	406	420	399	393	391*
la09	382	469	485	447	444	463	473	446	439	438*
la10	443	445	500	443	443	445	445	443	443	443
la11	413	570	583	548	541	553	600	543	539	538*
la12	408	504	508	483	474	500	504	479	471	471
la13	382	542	572	530	529	551	537	530	523	522*
la14	443	570	589	545	544	581	576	546	538	538
la15	378	584	595	554	554	597	584	555	550	550
la16	717	717	717	717	717	717	717	717	717	717
la17	646	646	646	646	646	646	646	646	646	646
la18	663	663	663	663	663	663	663	663	663	663
la19	617	617	682	617	617	617	617	617	617	617
la20	756	756	756	756	756	756	756	756	756	756
mt06	47	47	47	47	47	—	—	47	47	47
mt10	655	655	655	655	655	—	—	655	655	655
mt20	387	560	596	530	525	—	—	523	519	518*
Best		12	11	13	13	10	10	13	17	23
Sum_RPI		2.86	3.51	2.31	2.21	2.30	2.41	2.23	2.09	2.06

表 3 3 工厂实例测试结果

实例	LB	IGA	Heu	GA_JS	GA_OP	DTSMA	CRO	IDESAA	IGWO	HSFLA
la01	413	413	413	413	413	413	413	413	413	413
la02	394	394	394	394	394	394	394	394	394	394
la03	349	349	349	349	349	349	349	349	349	349
la04	369	369	369	369	369	369	369	369	369	369
la05	380	380	380	380	380	380	380	380	380	380
la06	413	413	413	413	413	413	413	413	413	413
la07	376	376	376	376	376	376	381	376	376	376
la08	369	369	370	369	369	369	403	369	369	369
la09	382	382	417	382	382	382	437	382	382	382
la10	443	443	443	443	443	445	443	443	443	443
la11	413	425	469	413	413	417	590	413	413	413
la12	408	408	417	408	408	423	483	408	408	408
la13	382	419	454	396	395	401	419	392	382	382
la14	443	443	445	443	443	462	470	443	443	443
la15	378	451	477	413	417	434	451	389	396	390
la16	717	717	717	717	717	717	717	717	717	717
la17	646	646	646	646	646	646	646	646	646	646
la18	663	663	663	663	663	663	663	663	663	663
la19	617	617	617	617	617	617	617	617	617	617
la20	756	756	756	756	756	756	756	756	756	756
mt06	47	47	47	47	47	—	—	47	47	47
mt10	655	655	655	655	655	—	—	655	655	655
mt20	387	439	450	407	397	—	—	387	387	387
Best		19	15	20	20	14	12	22	22	22
Sum_RPI		0.45	0.87	0.18	0.16	0.29	1.21	0.06	0.05	0.03

使用配对 t 检验对 2 工厂实例测试结果进行统计分析, 显著水平设置为 0.05。通过使用 MINITAB 统计分析软件, 求得 HSFLA 与 IGA、Heu、GA_JS、GA_OP、DTSMA、CRO、IDESAA 以及 IGWO 的 p 值分别为 0.001、0、0.001、0.001、0、0、0.001 以及 0.002 5, 均小于 0.05,

说明本文 HSFLA 在统计意义上好于现有 8 种算法。

图 8 给出了 2 工厂实例 mt20 解为 518 的甘特图, 图 9 给出了 3 工厂实例 la15 解为 390 的甘特图。图 8 和图 9 中, 标*的工序表示关键工序, 关键工序只存在于关键工厂中。

图 8 2 工厂实例 mt20 甘特图($C_{\max}=518$)

图 9 3 工厂实例 la15 甘特图($C_{\max}=390$)

4 结论

本文针对 DFJSP 问题, 提出了一种高效的混合蛙跳算法 HSFLA。该混合算法基于部分解空间编码, 通过引入变邻域搜索算法提升了其的局部搜索能力。通过引入针对关键工厂的全解空间禁忌搜索扩大了算法解空间, 弥补了部分解空间编码的缺陷, 从而获得了更加优异的解。本文所提算法刷新了多个 DFJSP 问题 benchmark 实例的当前最好解。

在今后的研究中, 我们将继续从 DFJSP 邻域结构、蛙跳算法的模因组划分以及进化方法等方面对算法改进, 从而求得更加优异的调度方案。除此之外, 我们将尝试使用其他算法, 如候鸟优化算法、鲸鱼群算法来求解 DFJSP 问题。

参 考 文 献

- [1] LUO Q, DENG Q, GONG G, et al. An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers[J]. Expert Systems with Applications, 2020, 160: 113721.
- [2] HUANG J, PAN Q, GAO L. An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times[J]. Swarm and Evolutionary Computation, 2020, 59: 100742.
- [3] WANG G, GAO L, LI X, et al. Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm[J]. Swarm and Evolutionary Computation, 2020, 57: 100716.
- [4] PAN Q, GAO L, WANG L, et al. Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem[J]. Expert Systems with Applications, 2019, 124: 309-324.
- [5] MENG L, ZHANG C, REN Y, et al. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem[J]. Computers & Industrial Engineering, 2020, 142: 106347.
- [6] NADERI B, AZAB A. An improved model and novel simulated annealing for distributed job shop problems[J]. The International Journal of Advanced Manufacturing Technology, 2015, 81(1-4): 693-703.
- [7] NADERI B, AZAB A. Modeling and heuristics for scheduling of distributed job shops[J]. Expert Systems with Applications, 2014, 41(17): 7754-7763.
- [8] 吴秀丽, 刘夏晶. 差分进化算法求解分布式柔性作业车间调度问题[J]. 计算机集成制造系统, 2019, 25(10): 2539-2558.
- [9] 吴锐, 郭顺生, 李益兵, 等. 改进人工蜂群算法求解分布式柔性作业车间调度问题[J]. 控制与决策, 2019, 34(12): 2527-2536.
- WU Xiuli, LIU Xiajing. Differential evolution algorithm for solving distributed flexible job shop scheduling problem[J]. Computer Integrated Manufacturing Systems, 2019, 25(10): 2539-2558.
- WU Rui, GUO Shunsheng, LI Yibing, et al. An improved artificial bee colony algorithm for distributed and flexible

- job-shop scheduling problem[J]. Control and Decision, 2019, 34(12): 2527-2536.
- [10] CHAN F T S, CHUNG S H, CHAN L Y, et al. Solving distributed FMS scheduling problems subject to maintenance: Genetic algorithms approach[J]. Robotics and Computer-Integrated Manufacturing, 2006, 22(5-6): 493-504.
- [11] CHAN F T S, CHUNG S H, CHAN P L Y. Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems[J]. International Journal of Production Research, 2006, 44(3): 523-543.
- [12] De GIOVANNI L, PEZZELLA F. An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem[J]. European Journal of Operational Research, 2010, 200(2): 395-408.
- [13] 杨江波, 陈友玲, 曹楠. 面向柔性作业分布式车间的分层调度模型研究[J]. 计算机工程与应用, 2014, 50(23): 239-244.
- YANG Jiangbo, CHEN Youling, CAO Nan. Research on grading scheduling model of flexible job distributed workshop[J]. Computer Engineering and Applications, 2014, 50(23): 239-244.
- [14] ZIAEE M. A heuristic algorithm for the distributed and flexible job-shop scheduling problem[J]. The Journal of Supercomputing, 2014, 67(1): 69-83.
- [15] LU P, WU M, TAN H, et al. A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems[J]. Journal of Intelligent Manufacturing, 2018, 29(1): 19-34.
- [16] CHANG H, LIU T. Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms[J]. Journal of Intelligent Manufacturing, 2017, 28(8): 1973-1986.
- [17] WU M, LIN C, LIN C, et al. Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems[J]. Computers & Operations Research, 2017, 80: 101-112.
- [18] LIN C, LEE I, WU M. Merits of using chromosome representations and shadow chromosomes in genetic algorithms for solving scheduling problems[J]. Robotics and Computer-Integrated Manufacturing, 2019, 58: 196-207.
- [19] 黄英杰, 姚锡凡. 基于目标级联法和粒子群算法的柔性分布式车间调度[J]. 中南大学学报, 2012, 43(1): 151-158.
- HUANG Yingjie, YAO Xifan. Planning and scheduling of multiple flexible-shops based on analytical target cascading and particle swarm optimization. Journal of Central South University, 2012, 43(1): 151-158.
- [20] EUSUFF M M, LANSEY K E. Optimization of water distribution network design using the shuffled frog leaping algorithm[J]. Journal of Water Resources Planning & Management, 2003, 129(3): 210-225.
- [21] PAN Q, WANG L, GAO L, et al. An effective shuffled frog-leaping algorithm for lot-streaming flow shop scheduling problem[J]. The International Journal of Advanced Manufacturing Technology, 2011, 52(5-8): 699-713.
- [22] LEI D, GUO X. A shuffled frog-leaping algorithm for job shop scheduling with outsourcing options[J]. International Journal of Production Research, 2016, 54(16): 4793-4804.
- [23] LEI D, ZHENG Y, GUO X. A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption[J]. International Journal of Production Research, 2016, 55(11): 3126-3140.
- [24] MASTROLILLI M, GAMBARDELLA L M. Effective neighbourhood functions for the flexible job shop problem[J]. Journal of Scheduling, 2000, 3(1): 3-20.
- [25] LI X, GAO L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem[J]. International Journal of Production Economics, 2016, 174: 93-110.

作者简介: 孟磊磊, 男, 1991 年出生, 博士, 讲师。主要研究方向为车间调度、刀具磨损、绿色制造。

E-mail: mengleilei@lzu-cs.com

张超勇(通信作者), 男, 1972 年出生, 博士, 教授, 博士研究生导师。主要研究方向为车间调度、网络化制造和绿色制造。

E-mail: zcyhust@mail.hust.edu.cn