文章编号: 1001-0920(2017)09-1583-08 DOI: 10.13195/j.kzyjc.2016.0620

求解离散调度问题的双机制头脑风暴优化算法

吴秀丽[1†], 张志强[1], 李俊青[2]

(1. 北京科技大学 机械工程学院，北京 100083；2. 聊城大学 计算机学院，山东 聊城 252000)

摘 要: 为了探讨头脑风暴算法对离散调度问题的求解能力,以柔性作业车间调度问题为应用场景,提出集成种群多样性机制和讨论机制的头脑风暴优化算法. 首先,建立柔性作业车间调度模型;然后,提出双机制头脑风暴优化算法,包含增加种群多样性机制和讨论机制,并深入分析算法的关键参数,设计关键操作,提出基于扩展工序的编码方式,设计聚类算法、扰动算子和合并算子;最后,对典型算例进行仿真计算,结果表明,增加种群多样性和讨论机制的头脑风暴优化算法表现最为优异,能够有效避免算法早熟,显著提高该系列算法的寻优能力.

关键词: 头脑风暴优化算法；种群多样性；讨论机制；柔性作业车间调度

中图分类号: TP18 **文献标志码:** A

A brain storm optimization algorithm integrating diversity and discussion mechanism for solving discrete production scheduling problem

WU Xiu-li[1†], *ZHANG Zhi-qiang*[1], *LI Jun-qing*[2]

(1. School of Mechanic Engineering，University of Science and Technology Beijing，Beijing 100083，China；2. School of Computer Science, Liaocheng University, Liaocheng 252000，China)

Abstract: The paper aims to present a brain storm optimization(BSO) algorithm integrating the population diversity and discussion mechanism(PD-DMBSO) to solve the flexible job shop scheduling problem(FJSP). Firstly, a math optimization model for FJSP is built. Secondly, the flowchart of the PD-DMBSO is proposed. Then, the key parameters of the PD-DMBSO are discussed. Considering the characters of the FJSP, the searching operators are designed. An extended operation encoding method is proposed. The K-means clustering algorithm is employed to cluster the individuals. A perturbation operator and a combining operator are designed to generate new individuals. Finally, a group of experiments are conducted to compare the four algorithms. The statistics analysis of the experiment results shows that the PB-DMBSO performs best among the four algorithms because it can effectively avoid the premature convergence and ensure to explore more solution space for discrete production scheduling problem.

Keywords: brain storm optimization algorithm; population diversity; discussion mechanism; flexible job shop scheduling

0 引 言

生产调度是提高企业运作效率和竞争力的关键环节. 为此,半个多世纪以来,众多学者对其展开了系列研究并取得重要成果.该问题包含多种不同的生产加工类型,因此Ying等[1]详细综述了调度问题,其中包括单机调度问题、并行机调度问题、流水车间调度问题、开放车间调度问题以及作业调度问题(JSP).

近年来,许多企业为了快速响应市场变化而引进柔性制造系统,进而Bruker等[2]提出了柔性作业车间调度问题(FJSP)并对其展开研究,该问题可以被分解为两个子问题:路由问题和调度问题.路由子问题是指将工序分配到可用机器上,调度子问题是指对工序的排序.FJSP是JSP的重要拓展和延伸,同时也是典型的组合优化问题,其更接近实际生产调度环境,因为减少了机器约束,增加了调度灵活性,但问题复杂度比JSP更高,已被证明属于多项式复杂度的非确定性(NP)难题[3].

传统数学优化方法难以在有限的时间求解FJSP,在Bruker等[2]开创性地采用多项式算法求解两工件的FJSP后,众多学者陆续采用多种智能优化方法对其进行求解. Brandimarte等[3]提出了一种基于禁忌搜索的算法;Chen等[4]设计了一种分别负责机器

收稿日期: 2016-05-18；**修回日期:** 2016-08-02.

基金项目: 国家自然科学基金项目(51305024, 61573178).

作者简介: 吴秀丽(1977−), 女, 副教授, 博士, 从事制造过程智能优化调度算法等研究；张志强(1990−), 男, 硕士生, 从事演化算法在调度问题中应用的研究.

†通讯作者. E-mail: wuxiuli@ustb.edu.cn

分配和工序排序的两部分编码方式的遗传算法; Ho等[5-6]提出了一种结合启发式方法和遗传算法的混合算法和一种解决学习型FJSP的高效的体系结构; Gao等[7]提出了一种结合遗传算法和变邻域下降算法的混合算法; Xing等[8]设计了一种有效整合蚁群优化模型和知识模型的算法; Yazdani等[9]将多种独立搜索算子与多种邻域搜索算子结合得到一种混合算法; Wang等[10]将一种基于有效双种群的分布估计方法应用于求解该问题; Wang等[11]提出了一种有效的人工蜂群算法; Yuan等[12]采用了一种混合差分进化算法; Rossi[13]开发了一种具有增强信息素关系的蚁群算法; Ziaee[14]设计了一种启发式方法; Wu等[15-16]设计了一种采用精英策略的量子进化算法和一种细菌觅食算法的改进算法.

由于智能优化算法的日新月异和FJSP问题的强NP难特性,国内外学者不断探索应用新的算法求解该问题. 头脑风暴优化算法(BSO)[17]是2011年出现的一种新型智能优化方法. BSO算法与常见的进化算法来源于仿生学不同,其概念和理论源于对人类会议过程中头脑风暴法的模拟. Shi[17]借鉴该流程设计了BSO算法,该算法已经被成功应用于许多领域,如电力调度问题[18]、直流无刷电机效率问题[19]、最优卫星编队重构问题[20]、多无人机编队飞行滚动时域控制[21]、股票指数预测[22]以及多模态优化问题[23]; Zhou等[24]改进了BSO算法,通过修正步长和个体产生机制来增加算法的寻优能力; 杨玉婷等[25]提出了一种基于讨论机制的头脑风暴优化算法,增加了个体产生的机制,平衡了局部搜索和全局搜索; 由于BSO算法易陷入局部最优,导致优化结果不理想, Cheng等[26-27]先后提出了两种在BSO算法中保持种群多样性的策略,以避免该算法无法跳出局部搜索.

BSO系列算法已经应用于很多连续函数优化问题及部分离散优化问题,但是在车间调度这类组合优化问题上尚未见到相关研究. 为此,本文在探讨已有的BSO系列算法的基础上,提出集成种群多样性和讨论机制的双机制BSO算法,并用于求解FJSP问题,为BSO算法在离散优化领域的应用推广提供重要的参考依据.

1 柔性作业车间调度问题

索引号: $i,g=1,2,\cdots,n$ 为工件号索引; $j,h=1,2,\cdots,n_i$ 为工序号索引; $k=1,2,\cdots,m$ 为机器号索引. 变量: N 为工件总数, M 为机器总数, n_i 为工件 i 的工序总数, O_{ij} 为工件 i 的第 j 道工序, O_{ijk} 表示工序 O_{ij} 选择在机器 k 上加工, S_{ij} 为工序 O_{ij} 的可用机器集合, p_{ijk} 为工序 O_{ij} 在机器 k 上的加工时间, C_{ij} 为工序 O_{ij} 的完工时间. 决策变量: X_{ijk} 表示若工序 O_{ij} 选择在机器 k 上加工,则决策变量 $X_{ijk}=1$,否则 $X_{ijk}=0$. 若工序 O_{ij} 与 O_{gh} 前后相邻,则决策变量 $y_1=-1$; 若工序 O_{gh} 与 O_{ij} 前后相邻,则 $y_1=1$; 若工序 O_{ij} 与 O_{gh} 不相邻,则 $y_1=0$.

1.1 问题描述

FJSP问题描述如下: n 个工件在 m 台机器上进行加工,每个工件 i 包含 n_i 道工序,工序之间遵循工艺约束; O_{ij} 表示工件 i 的第 j 道工序,每道工序 O_{ij} 可由多台机器加工; 工序 O_{ij} 在机器 k 上的加工时间为 p_{ijk}. FJSP包含两个子问题: 路由问题和调度问题. 路由问题是指为每道工序如何分配机器,调度问题是指如何安排工序的加工顺序. 通常情况下,在生产调度环节,大多数生产企业的第一目标是高效快速完成生产任务,因此设定目标函数为完工时间最短.

表1为一个3工件4机器的FJSP的例子,其中: J_i 表示工件号, O_{ij} 表示工件 i 的第 j 道工序, M_k 表示加工的机器号,数字代表工序的加工时间, "–"表示该机器不能用来加工该道工序.

表1 FJSP示例

工件	工序	M_1	M_2	M_3	M_4
J_1	O_{11}	1	3	4	–
	O_{12}	–	3	2	4
	O_{13}	4	1	–	–
J_2	O_{21}	3	4	–	1
	O_{22}	4	3	3	2
	O_{23}	5	4	7	2
J_3	O_{31}	–	–	5	3
	O_{32}	6	–	1	4
	O_{33}	4	3	–	4

FJSP需要满足以下假设: 1) 初始状态: 所有机器在 $t=0$ 时刻都可用,所有工件在 $t=0$ 时刻都可被加工; 2) 工艺确定性: 同一工件不同工序的加工顺序和在不同机器上的加工时间都是固定的; 3) 工件加工的唯一性: 每道工序只能选择一台机器加工,且只能被加工一次; 4) 机器占用的唯一性: 每台机器在同一时刻只能处理一道工序; 5) 非抢占式: 所有工序一旦开始加工便不能中断.

1.2 优化模型

$$\min\ C_{\max}=\min(C_{in_i}). \tag{1}$$

$$\text{s.t. } C_{ij}-C_{i(j-1)}\geqslant P_{ijk}X_{ijk},\ j=2,3,\cdots,n_i; \tag{2}$$

$$(C_{ij}-C_{gh}-P_{ijk})X_{ijk}X_{ghk}\Big(\frac{1}{2}y_1(y_1+1)\Big)+$$

$$(C_{gh}-C_{ij}-P_{ghk})X_{ijk}X_{ghk}\Big(\frac{1}{2}y_1(y_1-1)\Big)\geqslant 0; \tag{3}$$

$$\sum_k X_{ijk}=1,\ k\in S_{ij},\ \forall i,j; \tag{4}$$

$$X_{ijk}\in\{0,1\}; \tag{5}$$

$$y_1\in\{-1,0,1\}. \tag{6}$$

其中: 式(1)是目标函数, 表示最小化最大完工时间; 约束条件(2)是工艺约束, 表示同一工件不同工序的加工顺序要求; 约束条件(3)是机器约束, 表示一台机器在同一时刻只能加工一个工件; 约束条件(4)表示一道工序只能在一台机器上进行加工; 约束条件(5)和(6)表示决策变量的取值范围.

2　双机制头脑风暴优化算法求解FJSP

2.1　BSO系列算法综述

BSO算法是受会议过程中会议成员通过头脑风暴法产生优质方案的启发. 当面对一个较为复杂、一个人难以解决的问题时, 如果组织一群具有不同背景、不同经验和不同思维方式的人, 让他们提出不同见解、互相讨论并最终达成一致意见, 则更有助于高效地解决问题.

在一个头脑风暴会议中有3种角色: 主持人、问题拥有者或提出者、头脑风暴专家小组. 主持人的作用是负责主持整个流程, 确保头脑风暴小组在一定的规则(表2)下产生尽可能多的解决方案. 问题拥有者的作用是在当前已有方案中选择较好的方案, 以使头脑风暴小组能够集中精力在好的方案的基础上再进行思考和拓展. 头脑风暴专家小组的作用是根据自身的背景、知识、经验和思维方式等产生新方案. 表2中的规则是为了让头脑风暴小组中的成员保持足够的开放和发散, 产生尽可能多样性的方案, 以免陷入已有方案中, 无法跳出思维定式.

表 2　头脑风暴新方案产生规则

规则1：Suspend Judgment	所有方案都是好方案
规则2：Anything Goes	所有方案都要记录和分享
规则3：Cross-fertilize	所有方案应该基于现存方案, 每个方案都应该产生更多的方案
规则4：Go for Quantity	尽量产生较多的新方案

随着BSO算法的不断推广, 已经发展出一系列算法, 其中包括基本BSO算法[17]、改进的头脑风暴优化算法[24](MBSO)、基于讨论机制的头脑风暴优化算法[25](DMBSO). BSO系列算法容易出现算法早熟, 在测试已有算法过程中也发现常常出现种群最优解和种群所有解的均值相等的情况, 出现这种情况有两种可能: 1) 种群陷入局部最优, 种群中大部分个体都相同; 2) 种群大部分个体虽然不同, 但是所有个体的适应值相同. 为此, 本文在DMBSO算法的基础上增加一个种群多样性的机制, 即集成种群多样性和讨论机制的头脑风暴优化算法(PD-DMBSO).

2.2　PD-DMBSO算法总体设计

PD-DMBSO是在DMBSO算法的基础上增加了种群多样性保持机制. 当种群早熟时, 算法自动触发多样性策略. 本文设计了两种多样性策略: 1) 随机初始化部分个体, 即产生一定比例的随机个体, 用以替换原种群中相应数量的个体; 2) 对最优个体进行局部搜索, 产生一定数量的邻域个体, 用以替换原种群中相应数量的个体.

PD-DMBSO算法总体流程如图1所示.

Step 1: 初始化种群.

Step 2: 聚类.

1) 将popsize个个体聚为K类, 计算适应值, 根据适应值对类内个体排序, 最佳个体作为聚类中心;

2) 产生一个随机数$R_a\in(0,1)$, 如果R_a小于概率参数P_a, 则进入Step 2中的步骤3), 否则进入Step 3;

3) 随机选择一个聚类中心, 并随机生成一个新个体将其替换.

Step 3: 组内讨论.

1) 依次选择一个类, 并依次选择其中一个个体作为当前个体(记为old_1);

2) 随机产生一个随机数$R_{b1}\in(0,1)$, 如果该随机数小于概率参数P_{b1}, 则进入Step 3中的步骤3), 否则进入Step 3中的步骤4);

3) 对该聚类中心进行随机扰动产生新个体;

4) 产生一个随机数$R_{b2}\in(0,1)$, 如果该随机数小于概率参数P_{b2}, 则进入Step 3中的步骤5), 否则进入Step 3中的步骤6);

5) 在该类中随机选择一个个体进行随机扰动产生新个体:

6) 在该类中随机选择两个个体, 合并, 然后再进行随机扰动, 产生新个体:

7) 新个体与old_1比较, 若比old_1优秀则替换;

8) 判断是否到达最大组内讨论次数, 是则进入Step 4, 否则循环Step 3.

Step 4: 组间讨论.

1) 随机选择两个类, 并随机选择一个个体作为当前个体(记为old_2);

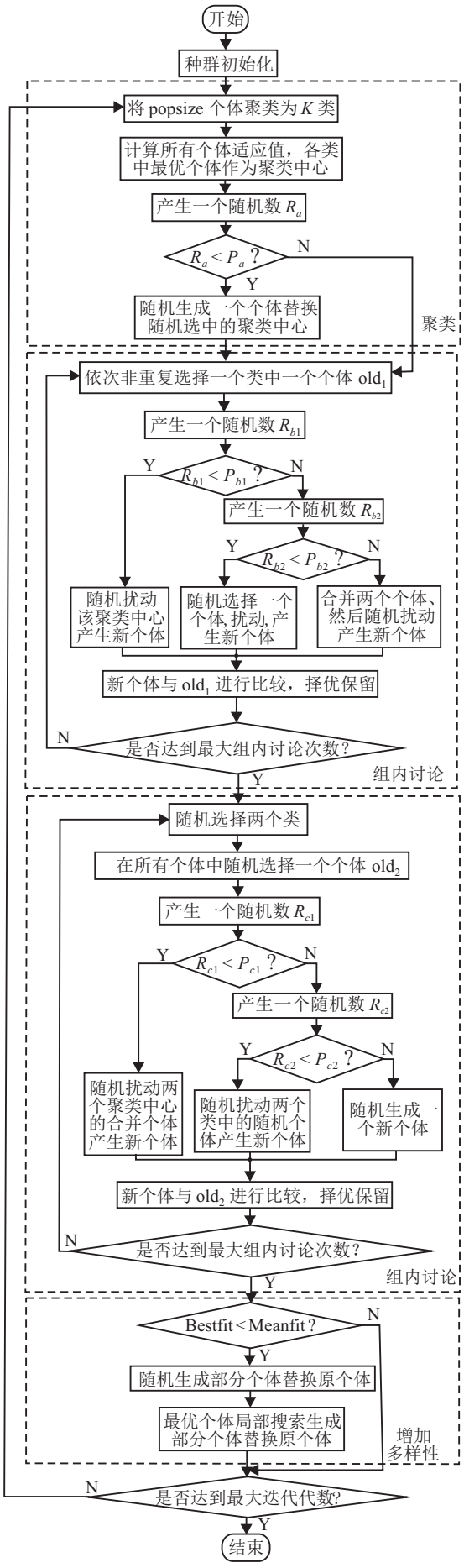

图 1 PM-DMBSO算法

2) 产生一个随机数 $R_{c1} \in (0,1)$, 如果该随机数小于概率参数 P_{c1}, 则进入Step 4中的步骤3), 否则进入Step 4中的步骤4);

3) 合并两个聚类中心并随机扰动产生新个体;

4) 产生一个随机数 $R_{c2} \in (0,1)$, 如果该随机数小于概率参数 P_{c2}, 则进入Step 4中的步骤5), 否则进入Step 4中的步骤6);

5) 在两个类中分别随机选择一个个体, 合并后进行随机扰动产生新个体;

6) 随机生成一个新个体;

7) 新个体与 old_2 比较, 若比 old_2 优秀则替换;

8) 判断是否达到最大组间讨论次数, 是则进入Step 5, 否则循环Step 4.

Step 5: 多样性策略.

1) 判断种群最佳适应值与种群平均适应值是否相等, 相等则进入Step 5中的步骤2), 否则进入Step 6;

2) 随机生成一定数量(种群规模*α)的新个体, 随机替换种群中该数量的个体;

3) 选择一个最优解进行局部搜索, 产生一定数量(种群规模*β)的新个体, 随机替换种群中该数量的个体.

Step 6: 迭代.

如果达到最大迭代代数, 则进入Step 7, 否则进入Step 2继续迭代.

Step 7: 终止.

在Step 2中, 适应值相近的个体被划分为一类, 适应值最佳的个体为聚类中心. 其中 P_a 决定了改变聚类中心的可能性, 即对算法进行扰动的强度, P_a 越大, 直接改变聚类中心的可能性越大, 对算法起到的扰动作用也越强.

Step 3中的组内讨论是在一个类内进行操作的, 其中 P_{b1} 和 P_{b2} 共同决定了算法是选择对一个类中的聚类中心进行扰动、对随机个体进行扰动还是随机选择两个个体进行合并产生新个体. 组内讨论有利于局部搜索, 促进算法收敛, 而其中对聚类中心进行扰动有利于局部搜索过程中的集中探索, 对随机个体进行扰动或对两个随机个体进行合并有利于保持局部搜索过程中的种群多样性.

Step 4中的组间讨论是两个类之间的交互操作, 其中 P_{c1} 和 P_{c2} 共同决定了算法是合并两个聚类中心、合并两个随机个体还是随机生成一个新个体. 组间讨论可以增加种群多样性, 有利于全局搜索, 避免算法早熟. 其中合并两个聚类中心有利于全局搜索过程中集成两个类的优势, 合并两个随机个体或随机生成一个新个体有利于增加全局搜索中的种群多样

性,保证算法在更广阔的解空间进行搜索.

组内讨论和组间讨论共同作用产生新个体,通过组内讨论和组间讨论的顺序执行可以极大促进算法的寻优能力[25].

Step 5中参数α和β是比例系数,决定了算法多样性保持的程度.系数越大,对种群多样性造成的影响越大,种群更容易跳出局部最优,同时也有可能破坏原种群的优势. 其中: $\alpha \in (0,1)$是随机初始化部分个体的比例系数,对算法的破坏较大; $\beta \in (0,1)$是最优个体局部搜索产生新个体的比例系数,对算法的破坏较小.

2.3　PD-DMBSO算法详细设计

1) 种群初始化. PD-DMBSO算法以种群的形式迭代优化,因此首先需要讨论种群中个体的表现形式. 在利用启发式算法求解实际问题的数学模型时,必须建立算法与模型之间的映射关系,可以通过编码实现. 本文采用的是扩展的基于工序的编码方式[28],工序用工件号表示,工件号出现顺序即为该工件的工序顺序. 以表1中问题为例,一个解的编码和其对应的工序调度顺序如图2所示,并依据活动化调度方式[15]解码成具体的调度方案.

个体编码　　[2 1 2 3 3 1 2 3 1]

调度顺序　　[O_{21} O_{11} O_{22} O_{31} O_{32} O_{12} O_{23} O_{33} O_{13}]

图2　个体编码与工序高度顺序

2) 聚类算法. PD-DMBSO算法采用K-means聚类算法,即找出K个聚类中心,使得每个个体与其最接近的聚类中心的平方距离和最小化. 具体流程如下: a. 初始化: 在所有个体中随机选择K个个体作为初始化的聚类中心; b. 聚类: 计算每个个体与每个聚类中心的平方距离,按照距离最近原则将个体进行聚类; c. 更新: 选择每个类中适应值为中位数的个体作为新的聚类中心; d. 迭代: 若Step 3中聚类中心发生变化,则进入Step 2进行迭代,否则结束聚类算法.

3) 扰动算子. 无论是在组内讨论还是组间讨论,都是通过扰动产生新个体. 本文提出一种移码变异的扰动算子,通过随机产生一个位置和一个距离,将该位置信息向右移动相应距离,如果超过最后一个位置则转到第一个位置继续. 图3为个体随机扰动的例子,假如选定原个体第3个位置,位移距离为4,即将位置3的工序2向右移动4个位置,插入位置7和8之间.

原个体　　[2 1 2 3 3 1 2 3 1]

扰动个体　　[2 1 3 3 1 2 2 3 1]

图3　扰动算子举例

在种群多样性策略部分,局部搜索算法亦采用该扰动算子产生邻域个体.

4) 合并算子. 在组内和组间讨论部分,还需要合并两个个体的优势部分. 本文提出一种基于位置的合并算子,在个体1中随机产生多个位置点,将选定位置点的信息复制到子代个体对应位置,并将个体2从左到右依次剔除个体1中选定位置点的信息,然后将剔除后的个体2信息依次填入子代个体的空缺位置以得到完整的子代个体. 两个体合并的原理如图4所示.

个体1　　[2 1 2 3 3 1 2 3 1]

个体2　　[1 2 3 1 3 1 2 2 3]

子代个体　　[3 3 2 1 2 1 2 3 1]

图4　合并算子举例

3　数值实验

3.1　实验设计

PD-DMBSO算法运行环境: Intel Core i3-3240 CPU 3.40 GHz, RAM 4.00 GB, Window 7, 64位操作系统和Matlab 2013b. 本文分别对BSO系列算法(BSO、MBSO、DMBSO)以及本文提出的PD-DMBSO四种算法进行对比实验,其中每个算法有多个参数,本文借鉴各参数的经验值[17, 24-25],力图将几种算法放在同一可对比的水平,最终确定了各算法的参数,结果如表3所示.

表3　各算法参数

算法	K	P_a	P_b	P_{b1}	P_{b2}	P_{c1}	P_{C2}	α	β
BSO	5	0.2	0.2	0.5	0.5	–	–	–	–
MBSO	5	0.2	–	–	–	–	–	–	–
DMBSO	5	0.2	–	0.6	0.5	0.5	0.7	–	–
PD-DMBSO	5	0.2	–	0.6	0.5	0.5	0.7	0.2	0.2

为了测试算法的性能,分别对FJSP国际标准算例Kacem算例[29]和Brandimarte算例[3]进行数值实验,并且与目前最优值进行对比分析; 以Brandimarte算例中6个算例为例,画出了4种算法30组实验的箱线图,以直观地观察30组实验结果的数据分布情况; 对所有算例进行统计分析,对比算法之间优化能力是否存在显著性差异. 通过以上实验和分析可以更为全面地观察4种算法的表现,为客观评价4种算法提供依据.

3.2　实验结果

Kacem算例和Brandimarte算例的计算结果分别如表4和表5所示.

表 4 Kacem算例计算结果

算例	工件 × 机器	最优解[3,30]	BSO	MBSO	DMBSO	PD-DMBSO
Kac1	4 × 5	11	11	11	11	11
Kac2	8 × 8	14	14	14	14	14
Kac3	10 × 7	11	11	11	11	11
Kac4	10 × 10	7	7	7	7	7
Kac5	15 × 10	11	11	11	11	11

表 5 Brandimarte算例计算结果

算例	工件 × 机器	最优解[3,30]	BSO	MBSO	DMBSO	PD-DMBSO
MK01	10 × 6	40	40	40	40	40
MK02	10 × 6	26	27	27	27	26
MK03	15 × 8	204	204	204	204	204
MK04	15 × 8	60	66	63	63	60
MK05	15 × 4	172	176	174	176	173
MK06	10 × 15	58	63	62	63	60
MK07	20 × 5	139	144	141	144	141
MK08	20 × 10	523	523	523	523	523
MK09	20 × 10	307	307	307	307	307
MK10	20 × 15	197	223	223	221	218

在Brandimarte算例中,MK01、MK03、MK08和MK09极易找到最优解,不同算法很难通过最优值对比得到相关结论,因此本文用4种算法计算其余6个算例,30组实验结果的箱线图如图5～图10所示.

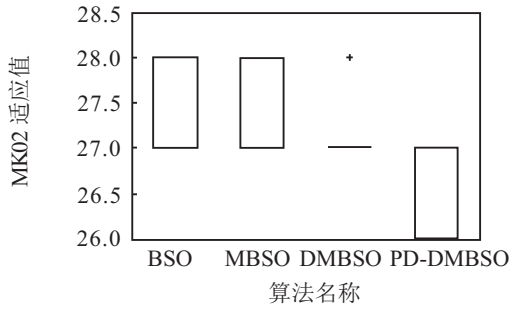

图 5 MK02问题箱线图

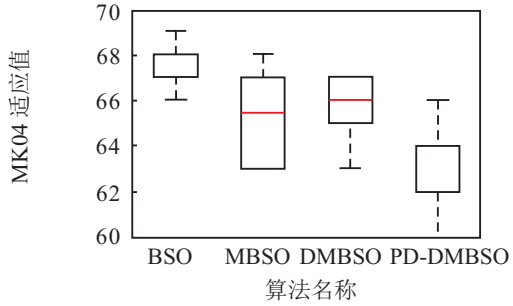

图 6 MK04问题箱线图

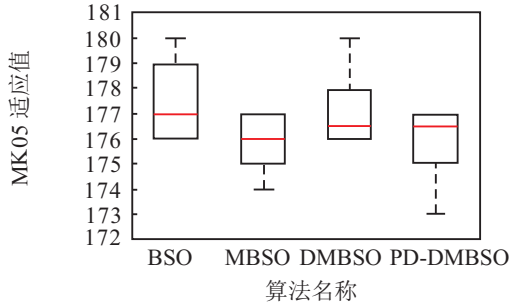

图 7 MK05问题箱线图

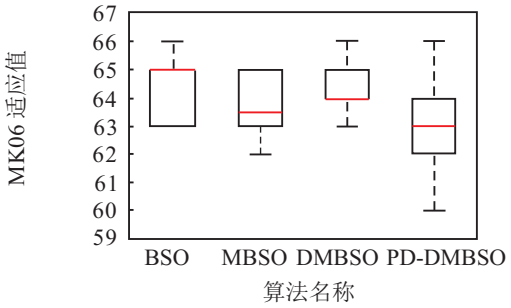

图 8 MK06问题箱线图

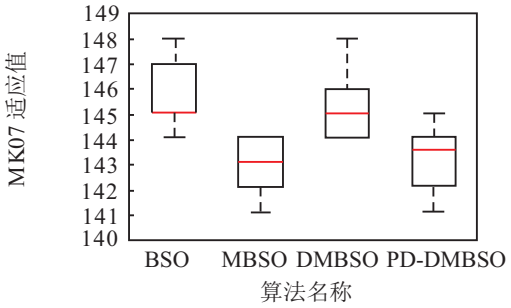

图 9 MK07问题箱线图

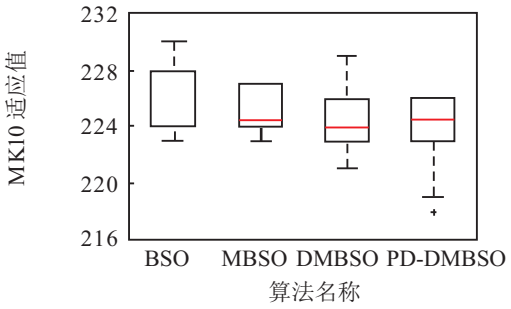

图 10 MK10问题箱线图

箱线图只能体现实验数据的部分特征,无法精确反应数据的全部信息,为此本文通过对30组实验数据进行Wilcoxon配对测试得到统计分析数据. 30组实验均值($C_{\max}$)和计算时间(cpu)如表6所示,不同算法之间的显著性水平如表7所示(置信水平0.05).

表 6 30组实验均值

算例/算法	BSO		MBSO		DMBSO		PD-DMBSO	
	$C_{\max}$	cpu	$C_{\max}$	cpu	$C_{\max}$	cpu	$C_{\max}$	cpu
Kac1	11	163	11	255	11	196	11	188
Kac2	14	218	14	344	14	232	14	226
Kac3	11	343	11	588	11	313	11	308
Kac4	7	387	7	658	7	351	7	366
Kac5	11	498	11	852	11	447	11	456
MK01	40	458	40	1 261	40	536	40	485
MK02	27.7	384	27.3	1 251	27.1	423	26.7	434
MK03	204	1 600	204	4 363	204	1 814	204	1 709
MK04	67.2	500	65.2	1 578	65.6	533	63.1	540
MK05	177.4	482	175.9	1 568	177	561	176	575
MK06	64.5	980	63.7	3 262	64.4	1 148	62.9	1 173
MK07	145.8	557	142.7	1 796	145	649	143	654
MK08	523	1955	523	5 592	523	2 250	523	2 223
MK09	307	2467	307	5 368	307	3 009	307	3 105
MK10	226.4	1630	225	5 162	225	1 965	224	1 998

表 7　算法显著性水平 ($\alpha = 0.05$)

	MBSO	DMBSO	PD-DMBSO
BSO	1	1	1
MBSO		0	0
DMBSO			1

MBSO、DMBSO和PD-DMBSO对BSO显著性水平均为1,表示MBSO、DMBSO以及PD-DMBSO这3种算法性能显著好于BSO算法;DMBSO和PD-DMBSO对MBSO显著性水平均为0,表示DMBSO和PD-DMBSO算法与MBSO算法性能没有显著性差异;PD-DMBSO对DMBSO显著性水平为1,表示PD-DMBSO算法显著好于DMBSO算法.

3.3　实验结论

就4种算法的实验结果而言:基本BSO算法性能最差;DMBSO算法性能好于BSO算法,与MBSO算法相近,两者没有显著性差异;MBSO算法在部分算例(如MK05、MK07)上表现突出,显著好于DMBSO算法和基本BSO算法,仅次于PD-DMBSO算法;PD-DMBSO算法显著好于BSO算法和DMBSO算法,与MBSO算法没有显著性差异,但就最优值而言,PD-DMBSO算法能够比其他3种算法找到更好的解. 然而,就箱线图而言,MBSO算法性能不稳定,对于部分算例(如MK05、MK07)表现突出,但是部分算例(如MK02、MK10)则表现欠佳,而且就计算时间而言,该算法由于过度依赖于随机搜索,算法流程循环过多,导致其计算耗时远大于其他算法. 因此,综合考虑最优解、箱线图和统计分析显著性水平,4种算法的性能排序为BSO < DMBSO < MBSO < PD-DMBSO.

4　结　论

本文着眼于离散生产调度优化问题,以柔性作业车间调度问题为例. 首先,建立了柔性作业车间调度问题的优化模型;然后,基于头脑风暴优化算法思想,提出了求解该类调度问题的双机制头脑风暴优化算法;最后,通过数值仿真实验表明,PD-DMBSO算法由于兼具了种群多样性机制和讨论机制,能够很好地避免算法陷入局部最优,同时也保证了充分的全局搜索能力,所以在benchmark问题的求解结果上比BSO系列其他算法表现都好.

另外,就搜索各个benchmark问题最优解的能力而言,4种BSO算法并未表现出更大的优越性,与其他智能优化算法寻优能力相当. 由此可以在以下方面进一步展开研究: 1)跳出纯粹BSO算法框架的束缚,集成其他优秀的演化算法,形成混合算法,有望进一步提高算法的整体性能; 2)针对离散生产调度问题的多目标特征,探讨多目标头脑风暴优化算法.

参考文献(References)

[1] Ma Y, Chu C, Zuo C. A survey of scheduling with deterministic machine availability constraints[J]. Computers & Industrial Engineering, 2010, 58(2): 199-211.

[2] Brucker P, Schlie R. Job-shop scheduling with multi-purpose machines[J]. Computing, 1990, 45(4): 369-375.

[3] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations Research, 1993, 41(3): 157-183.

[4] Chen H, Ihlow J, Lehmann C. A genetic algorithm for flexible job-shop scheduling[C]. 1999: IEEE Int Conf on Robotics and Automation. IEEE, 1999: 1120-1125.

[5] Ho N B, Tay J C. Genace: An efficient cultural algorithm for solving the flexible job-shop problem[C]. Congress on Evolutionary Computation. IEEE, 2004: 1759-1766.

[6] Ho N B, Tay J C, Lai E M K. An effective architecture for learning and evolving flexible job-shop schedules[J]. European J of Operational Research, 2007, 179(2): 316-333.

[7] Gao J, Sun L Y, Gen M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems[J]. Computer & Operations Research, 2008, 35(9): 2892-2907.

[8] Xing L N, Chen Y W, Wang P, et al. A knowledge-based ant colony optimization for flexible job shop scheduling problems[J]. Applied Soft Computing, 2010, 10(3): 888-896.

[9] Yazdani M, Amiri M, Zandieh M. Flexible job-shop scheduling with parallel variable neighborhood search algorithm[J]. Expert Systems with Applications, 2010, 37(1): 678-687.

[10] Wang L, Wang S, Xu Y, et al. A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem[J]. Computers & Industrial Engineering, 2012, 62(4): 917-926.

[11] Wang L, Zhou G, Xu Y. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem[J]. Int J of Advanced Manufacturing Technology, 2012, 60(1/2/3/4): 303-315.

[12] Yuan Y, Xu H. Flexible job shop scheduling using hybrid differential evolution algorithms[J]. Computers & Industrial Engineering, 2013, 65(2): 246-260.

[13] Rossi A. Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships[J]. Int J of Production Economics, 2014, 153: 253-267.

[14] Ziaee M. A heuristic algorithm for solving flexible job shop scheduling problem[J]. The Int J of Advanced Manufacturing Technology, 2014, 71(1/2/3/4): 519-528.

[15] 吴秀丽, 张志强, 杜彦华, 等. 改进细菌觅食算法求解柔性作业车间调度问题[J]. 计算机集成制造系统, 2015, 21(5): 1262-1270.
(Wu X L, Zhang Z Q, Du Y H, et al. Improved bacteria foraging optimization algorithm for flexible job shop scheduling problem[J]. Computer Integrated Manufacturing Systems. 2015, 21(5): 1262-1270.)

[16] Wu X, Wu S. An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem[J]. J of Intelligent Manufacturing, 2015: 1-17.

[17] Shi Y. Brain storm optimization algorithm[J]. Lecture Notes in Computer Science, 2011, 4(3): 303-309.

[18] Ramanand K R, Krishnanand K R, Panigrahi B K, et al. Brain storming incorporated teaching-learning - based algorithm with application to electric power dispatch[C]. Swarm, Evolutionary, and Memetic Computing. Berlin Heidelberg: Springer, 2012: 476-483.

[19] Duan H, Li S, Shi Y. Predator-prey brain storm optimization for DC brushless motor[J]. IEEE Trans on Magnetics, 2013, 49(10): 5336-5340.

[20] Sun C, Duan H, Shi Y. Optimal satellite formation reconfiguration based on closed-loop brain storm optimization[J]. IEEE Computational Intelligence Magazine, 2013, 8(4): 39-51.

[21] Qiu H, Duan H. Receding horizon control for multiple UAV formation flight based on modified brain storm optimization[J]. Nonlinear Dynamics, 2014, 78(3): 1973 -1988.

[22] Sun Y. A hybrid approach by integrating brain storm optimization algorithm with grey neural network for stock index forecasting[J]. Abstract & Applied Analysis, 2014, 2014(1): 1-10.

[23] Guo X, Wu Y, Xie L. Modified brain storm optimization algorithm for multimodal optimization[C]. Advances in Swarm Intelligence. Hefei: Springer Int Publishing, 2014: 340-351.

[24] Zhou D, Shi Y, Cheng S. Brain storm optimization algorithm with modified step-size and individual generation[C]. Advances in Swarm Intelligence. Berlin Heidelberg: Springer, 2012: 243-252.

[25] 杨玉婷, 史玉回, 夏顺仁. 基于讨论机制的头脑风暴优化算法[J]. 浙江大学学报: 工学版, 2013, 47(10): 1705-1711.
(Yang Y T, Shi Y H, Xia S R. Discussion mechanism based brain storm optimization algorithm[J]. J of Zhejiang University: Engineering Science, 2013, 47(10): 1705-1711.)

[26] Cheng S, Shi Y, Qin Q, et al. Maintaining population diversity in brain storm optimization algorithm[C]. 2014 IEEE Congress on Evolutionary Computation(CEC). IEEE, 2014: 3230-3237.

[27] Cheng S, Shi Y, Qin Q, et al. Population diversity maintenance in brain storm optimization algorithm[J]. J of Artificial Intelligence & Soft Computing Research, 2015, 4(3): 83-97.

[28] 吴秀丽, 孙树栋, 余建军, 等. 多目标柔性作业车间调度优化研究[J]. 计算机集成制造系统, 2006, 12(5): 731-736.
(Wu X L, Sun S D, Yu J J, et al. Research on multi-objective optimization for flexible job shop scheduling[J]. Computer Integrated Manufacturing Systems, 2006, 12(5): 731-736.)

[29] Kacem I, Hammadi S, Borne P. Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems[J]. IEEE Trans on Systems, Man, and Cybernetics, Part C, 2002, 32: 1-13.

[30] Hmida A B, Haouari M, Huguet M J, et al. Discrepancy search for the flexible job shop scheduling problem[J]. Computers and Operations Research, 2010, 37: 2192 - 2201.

（责任编辑：闫 妍）