文章编号: 1001-0920(2017)03-0474-07 DOI: 10.13195/j.kzyjc.2015.1503

零等待约束下多产品间歇过程的多目标调度

邓冠龙[1†], 田广东[2], 顾幸生[3]

(1. 鲁东大学 信息与电气工程学院，山东 烟台 264025；2. 吉林大学 交通学院，长春 130022；
3. 华东理工大学 化工过程先进控制和优化技术教育部重点实验室，上海 200237)

摘 要: 针对零等待约束下多产品间歇过程的总流程时间和完工时间最小化问题,提出一种多目标离散组搜索算法求解.在采用启发式规则产生初始解的基础上,通过发现者、追随者和巡逻者的操作设计,算法不断更新Pareto前沿,同时,混合了基于插入邻域的多目标局部搜索方法.大量计算实验表明,所提出的算法获得的非支配解集在IGD和Set Coverage指标上优于非支配排序遗传算法和模拟退火算法,可为多目标决策者提供更好的决策依据,利于间歇生产过程的优化运行.

关键词: 遗传算法；生产；优化；间歇过程；多目标

中图分类号: TP273 **文献标志码:** A

Multi-objective scheduling for multi-product batch process with no-wait constraint

DENG Guan-long[1†], *TIAN Guang-dong*[2], *GU Xing-sheng*[3]

(1. School of Information and Electrical Engineering，Ludong University，Yantai 264025，China；2. Transportation College，Jilin University，Changchun 130022，China；3. Key Laboratory of Advanced Control and Optimization for Chemical Processes，Ministry of Education，East China University of Science and Technology，Shanghai 200237，China)

Abstract: The multi-objective scheduling problem for the multi-product batch process with no-wait constraint is considered with makespan and total flow time minimizations, and a multi-objective discrete group search optimizer is proposed. The algorithm employs a well-known heuristic to generate initial solutions, and updates a non-dominated solution set by operators of producers, scroungers and rangers.Besides, an insertion-based local search procedure is hybridized. A bunch of computational experiments reveal that the proposed algorithm yields a non-dominated solution set with better IGD and Set Coverage values than the non-dominated sorting genetic algorithm and the bi-objective multi-start simulated annealing algorithm. It is also shown that, the algorithm can provide better basis for multi-objective decision-makers, which benefits the operation optimization of batch process production.

Keywords: genetic algorithm；production；optimization；batch process；multi-objective

0 引 言

间歇生产过程是现代化工过程中很重要的一个分支,它非常适合于小批量、多品种的精细化工生产过程[1-3],近年来越来越受到研究者的重视.间歇生产方式广泛存在于各种生产环境中,如制药、纺织、食品等行业[4].

间歇过程的调度是指如何将各个产品的生产安排到各台设备上,以达到某个或某些性能指标最优,如生产周期最短、能耗最小等.合理有效的调度方案可显著提高生产效率、节约成本,因此,间歇过程的调度已得到了较多的关注[5-8].作为较特殊的情形之一,产品生产时的零等待约束大量存在于各种间歇过程中[9].例如,在钢铁生产过程中,物料烧融后要立即铸造; 在制药过程中,中间产物必须立即进入下一工序以防止发生次生反应.因此,对零等待约束下间歇过程调度的研究具有重要的理论和实际意义.

目前,该问题的研究已有一些成果,杨玉珍等[10]考虑了零等待多目的间歇过程中的多目标调度问题,提出带存储的完全局部搜索方法进行求解.Dong等[11]在零等待多产品间歇过程的基础上,考虑了准

收稿日期: 2015-12-04；**修回日期:** 2016-02-27.

基金项目: 国家自然科学基金项目(61403180, 51405075, 61573144).

作者简介: 邓冠龙(1985−), 男, 讲师, 从事生产计划与调度、智能优化算法等研究；顾幸生(1960−), 男, 教授, 博士生导师, 从事生产计划与调度、复杂工业过程建模、控制和优化等研究.

†通讯作者. E-mail: dglag@163.com

备时间的影响,提出了混合差分进化算法,可求解大规模问题. 文献[12-13]也考虑了零等待间歇过程调度, 并取得了较好的效果. 这些研究包括针对单一目标的优化,也包括针对多目的间歇过程的求解. 然而,多目标优化难度较大,且多产品间歇过程具有多种复杂约束,因此,多产品间歇过程的多目标调度研究目前还较少. 其中,文献[14]考虑了具有准备时间的柔性流水线多目标调度,文献[15]给出了一种差分进化算法,可实现零等待流水车间最大完工时间和最大滞后为目标的调度. 本文考虑零等待约束下的多产品间歇过程调度,并对最大完工时间和总流程时间进行多目标优化,引入了组搜索算法(GSO)来求解相应问题.

作为一种新型群智能优化算法,GSO算法是He等[16]受动物集体捕猎行为启发而提出的,算法种群中的个体按照“发现-参与”的不同分工扮演不同的角色:发现者、追随者或巡逻者. 组搜索算法将种群分为3种不同角色的机制,有利于实现种群多样化的目的,这3种角色的进化机制充分体现了其在寻优过程中的分工,发现者用于探索较优位置附近区域,追随者的分享机制体现了种群的开发性,而巡逻者的随机搜索则体现了算法对未知区域的探索性. 因此,本文考虑引入该算法的进化机制,设计发现者、追随者和巡逻者的操作,为适应离散的调度问题求解,采用离散的工件编码,提出一种离散域下进化的多目标离散组搜索算法(MDGSO),并将其应用于零等待约束下多产品间歇过程的多目标调度中.

1　零等待约束下多产品间歇过程的多目标调度

1.1　零等待约束下的多产品间歇过程

在多产品间歇过程中,有n个产品和m个设备,每个产品需要依次经过第1台设备,第2台设备,直到第m台设备. 产品$j(j = 1,2,\cdots,n)$在第$k(k = 1,2,\cdots,m)$台设备上的处理过程记为工序O_{jk},对应的处理时间为$p(j,k)$. 为满足零等待的约束,工序O_{jk}的完成时刻必须等于工序$O_{j,k+1}$的开始时刻. 这意味着每个产品一旦在第一台设备上开始处理,则需要不停歇地进行下去,直到完成最后一台设备上的处理过程. 为满足这一约束,产品的开始处理时刻很可能后延,从而导致总的完成时间相比于传统的多产品间歇过程要长. 为使问题描述严格,还需作如下假设:

1)所有产品和设备在零时刻是可用的;

2)处理时间大于零,且是已知的;

3)产品的准备时间、在设备间的转移时间已经包括在处理时间内.

图1给出了零等待约束下的某三产品三设备的多产品间歇过程处理甘特图,其中U表示设备,J表示产品.

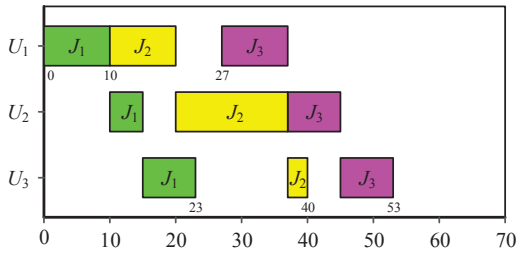

图1　某零等待多产品间歇过程甘特图

1.2　多目标调度模型

由图1可见,若产品的开始处理时刻已确定,则该产品的最终完成时间也相应确定. 图中给出的是产品按照产品号1-2-3的先后次序处理的结果,最大完工时间(makespan)为53. 若产品的处理次序改变,则makespan也随之变化. 由于总流程时间(TFT)是一个很重要的指标,本文考虑以makespan和TFT为目标,寻找Pareto最优解集. 注意,makespan最小的解并不一定能达到TFT最小,反之亦然. 例如,若有两个调度解A和B,具有不同的处理次序,且解A的makespan要小于解B,则一种可能的情况是:由于解A中某些产品的开始时刻要大于解B,使得解A中这些产品的完成时刻也大于解B,从而导致解A的TFT大于解B. 因此,这两个目标是有冲突的. 对该多目标问题,每个解可表示为$\pi = \{\pi_1,\pi_2,\cdots,\pi_n\}$,$\pi_j$表示产品号$(j = 1,2,\cdots,n)$. 用$d(\pi_{j-1},\pi_j)$表示第一台机器上的产品$\pi_{j-1}$和$\pi_j$开始时刻的最小延迟,$C(\pi_j,m)$表示$\pi_j$在设备$m$上的完成时间,$C_{\max}(\pi)$表示$\pi$对应的makespan,则可通过各个工序的处理时间$p(j,k)$计算

$$d(\pi_{j-1},\pi_j) = p(\pi_{j-1},1) + \max\left[0,\max_{2\leqslant k\leqslant m}\left\{\sum_{h=2}^{k}p(\pi_{j-1},h) - \sum_{h=1}^{k-1}p(\pi_j,h)\right\}\right]. \tag{1}$$

在此基础上可计算

$$C(\pi_j,m) = \sum_{i=2}^{j}d(\pi_{i-1},\pi_i) + \sum_{k=1}^{m}p(\pi_j,k). \tag{2}$$

于是可得两个目标函数

$$f_1(\pi) = C_{\max}(\pi) = C(\pi_n,m), \tag{3}$$

$$f_2(\pi) = \mathrm{TFT}(\pi) = \sum_{j=1}^{n}C(\pi_j,m). \tag{4}$$

令所有的π构成集合Π,则问题为

$$\min\ \{f_1(\pi),f_2(\pi)\},\ \forall\pi\in\Pi. \tag{5}$$

2 多目标离散组搜索算法

组搜索算法最初是针对连续的函数优化问题提出的[16],算法内的个体由发现者、追随者和巡逻者3种角色组成,不同角色有不同的更新机制.本文所解决的问题为组合优化问题,产品号的排列可直接映射到活跃调度,因此本文采用离散排列π作为种群个体编码,相应地,发现者、追随者、巡逻者的操作也进行重新设计.另一方面,本文对所考虑的多目标问题采用Pareto方法处理,提出一种多目标离散组搜索算法(MDGSO),尝试搜索问题对应的Pareto最优解集合.

2.1 种群初始化

组搜索算法在每一次的迭代中始终维持大小为p_s的种群,记为$P_L = \{L_1, L_2, \cdots, L_{p_s}\}$,在算法进入个体角色操作之前,需初始化种群中的每个个体.为得到性能较好的初始种群,采用NEH启发式规则[17]产生一个初始个体$\pi_{\rm NEH}$,该规则是针对makespan目标提出的性能较好的启发式规则,因此$\pi_{\rm NEH}$的构建过程以makespan为目标.注意到算法要进行多目标搜索,这里采用一个NEH规则的改进方法——NEH_WPT规则[18]产生另一个初始个体$\pi_{\rm NEH_WPT}$,该规则是针对带阻塞的情形并以TFT为目标提出的,因此$\pi_{\rm NEH_WPT}$的产生过程以TFT为目标.初始种群中剩下的其他p_s-2个个体通过随机方式产生.这样,可使初始种群具有较好性能的同时也兼具多样性.

由于算法进行多目标搜索,其始终维持一个非支配解集合,记为$N_S = \{S_1, S_2, \cdots, S_{n_b}\}$,$n_b$为当前非支配解集的大小.在初始化种群之后,将$P_L$集合的Pareto前沿解作为初始的$N_S$集合.为避免重复的局部搜索,$N_S$中的每个解都做“未搜索"或“已搜索”的标记,初始化时,N_S中的解都标记为“未搜索”.

2.2 发现者操作

组搜索算法发现者的作用是探索较优位置附近区域.然而对多目标问题,较优个体不是单一的,而是集合N_S.因此,发现者使用N_S中的某个解,其操作设计为:判断N_S中是否存在“未搜索”的解,若存在则执行基于插入的Pareto局部搜索(IPLS)过程;若不存在,则随机选择N_S中一个解作为X,然后对X进行d次随机插入操作,再执行IPLS过程.随机插入操作是指对考虑的X,随机选择一个产品,将其随机插入除当前位置外的其他位置上.由于集合N_S中的解通常具有较好的性能,但在进化后期很可能是局部最优,进行d次随机插入操作可得到一邻域解,该邻域解可保持较好的解结构,同时又可能跳出局部最优.发现者的执行过程如下:

Step 1: 若非支配集合N_S中存在标记为“未搜索”的个体(假设为S_k),则令$X = S_k$,转Step 3.

Step 2: 随机选择N_S中一个个体(假设为S_k),令$X = S_k$,对X进行d次随机插入操作,再对X执行IPLS.

Step 3: 对X执行IPLS,若IPLS未能改变X,则将对应的N_S中解S_k标记为“已搜索”.

IPLS过程在原有解X的基础上,对其某个产品进行插入,若插入操作可获得支配X的解,则改变产品继续进行下去,直到对所有产品都不能再找到更优的支配解.步骤如下:

Step 1: 产生一个随机的产品排列$\pi r = \{\pi r_1, \pi r_2, \cdots, \pi r_n\}$,令$i = 0, j = 1$.

Step 2: 找出X中产品πr_j的位置,将X中产品πr_j插入X的其他$n-1$个位置上,可得$n-1$个解,通过计算目标值可得这$n-1$个解的局部非支配集合,记为$L_{\rm NS}$.若存在$X' \in L_{\rm NS}$且$X' \prec X$,则$L_{\rm NS} = L_{\rm NS} \backslash X', X = X'$并置$i = 1$; 否则,$i = i + 1$,置$j = (j+1)\%n$(%表示取余).

Step 3: 利用$L_{\rm NS}$更新N_S,若N_S中有新解加入,则标记为“未搜索”.

Step 4: 若$i < n$,则转Step 2; 否则,利用X更新N_S,若X加入N_S中,则标记为“已搜索”.

上述过程可持续搜索支配X的解X',直到插入邻域不再能找到X'为止.

2.3 追随者操作

组搜索算法中除了发现者之外,种群中其他p_s个个体按照概率p被选择为追随者,若不是追随者,则按巡逻者处理.根据组搜索算法的思想,追随者需要快速向发现者靠近,其分享机制要体现种群的开发性.因此利用遗传算法交叉思想,追随者设计为与较优个体进行交叉.具体地,从N_S中随机选择一个解与当前追随者对应的解进行交叉,交叉方法使用遗传算法中常用的部分匹配交叉(PMX).任意两个不同个体A和B之间存在3种关系: 1) A被B支配; 2) B被A支配; 3) 互不支配.追随者交叉之后,根据子代个体与追随者个体间的关系决定是否更新追随者个体,其关系如图2所示.具体操作步骤如下:

Step 1: 从非支配集合N_S中随机选择一个个体(假设为S_k)和追随者个体(假设为L_t)交叉,得到X_1和X_2.

Step 2: 利用X_1和X_2更新非支配集合N_S,若N_S中有新解加入,则标记为“未搜索”.

Step 3: 若$L_t \prec X_1$且$L_t \prec X_2$,如图2(a)所示,则L_t不变;若$L_t \prec X_1$(或X_2)但L_t不支配X_2(或X_1),如图2(b)所示,则X_2(或X_1)替换L_t;若L_t不支配X_1且L_t不支配X_2,则看X_1与X_2的支配关系.若X_1(或X_2)支配X_2(或X_1),如图2(c)所示,则X_1(或X_2)替换

L_t; 否则, 如图2(d)所示, 随机选择X_1或X_2中一个替换L_t.

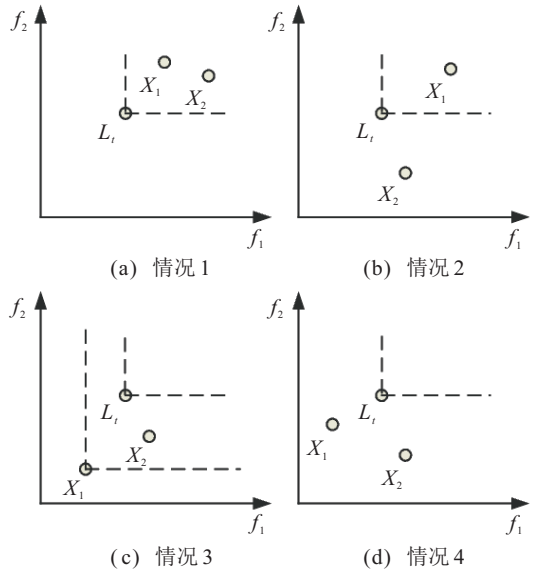

图 2　子代个体支配关系

2.4　巡逻者操作

巡逻者的作用是在搜索空间进行随机探索, 以增强算法的全局搜索能力. 本文提出的离散组搜索算法中, 巡逻者并不进行完全的随机探索, 而是在非支配解集N_S的基础上进行较大的扰动得到. 原因在于利用现有的非支配解集可保持部分较好的解结构特征, 而若完全随机搜索, 则巡逻者找到较好区域的可能性大大降低. 具体地, 先从N_S中随机选择一个解作为X, 考察X的插入邻域$N_b(X)$, 若存在$X' \in N_b(X)$且某个目标函数值要低于X, 则下降方向选定为该目标, 并置$X = X'$; 然后再次考察X的插入邻域$N_b(X)$, 并尝试沿着原有的下降方向继续进行下去. 其中$N_b(X)$是指将某个位置上的产品插入其他位置上得到的解的集合, 大小为$(n-1)^2$. 巡逻者操作实现步骤如下:

Step 1: 从非支配集合N_S中随机选择一个个体(假设为S_k), 令$X = S_k$.

Step 2: 评估X的插入邻域$N_b(X)$. 若存在解$X' \in N_b(X)$, 且$f_1(X') < f_1(X)$成立, 转Step 3; 若存在解$X' \in N_b(X)$, 且$f_2(X') < f_2(X)$成立, 转Step 4; 否则, 将S_k标记为“已搜索”, 结束.

Step 3: $N_b(X) = N_b(X) \backslash X'$, 利用$N_b(X)$更新$N_S$, 若$N_S$中有新解加入, 则标记为“未搜索”. $X = X'$, 评估X的插入邻域$N_b(X)$, 若存在$X' \in N_b(X)$, 且$f_1(X') < f_1(X)$成立, 转Step 3; 否则, 转Step 5.

Step 4: $N_b(X) = N_b(X) \backslash X'$, 利用$N_b(X)$更新$N_S$, 若$N_S$中有新解加入, 则标记为“未搜索”. $X = X'$, 评估X的插入邻域$N_b(X)$, 若存在$X' \in N_b(X)$, 且$f_2(X') < f_2(X)$成立, 转Step 4; 否则, 转Step 5.

Step 5: 利用X更新N_S, 若X加入N_S中, 则标记为“已搜索”. 采用X替换当前的巡逻者个体(假设为L_t), 结束.

2.5　MDGSO算法流程

在给出了MDGSO的种群初始化、个体角色操作和嵌入的局部搜索方法之后, 整个算法流程便可清晰地用图3描述.

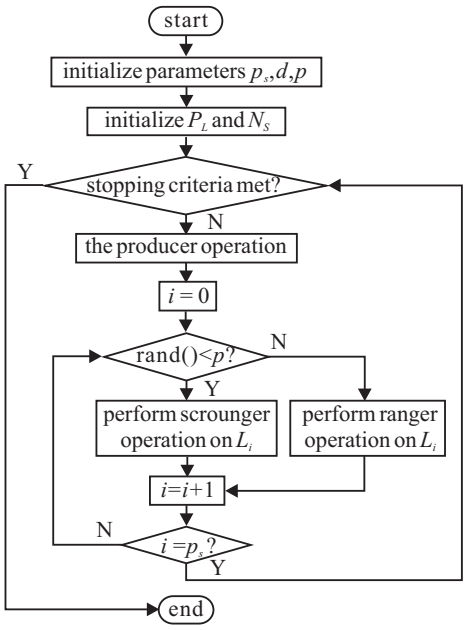

图 3　MDGSO流程

算法采用离散的产品排列, 首先利用π_{NEH}(makespan为目标)、$\pi_{\mathrm{NEH_WPT}}$(TFT为目标)和随机的方式产生初始种群P_L, 并初始化非支配解集N_S, 然后进入迭代. 在每一代中, 算法首先结合基于插入的Pareto局部搜索, 进行发现者操作, 然后对种群的p_s个个体执行追随者或巡逻者操作. 算法涉及的参数有: 扰动规模d, 追随者概率p, 种群大小p_s.

3　仿真研究

3.1　算例选择和运行设置

为了验证MDGSO算法的效果, 本文使用Taillard给出的算例[19]. 该算例集合包括120个算例, 本文使用其中90个算例, 产品数从20至100, 机器数从5至20, 基本涵盖了实际工业应用上可能遇到的问题规模. 本文所有的算法均采用C++语言编程实现, 运行环境是4 GB内存的Intel(R) Core(TM) i7-2600 CPU @3.06 GHz PC.

3.2　算法评估指标

与单目标优化不同, 多目标优化时算法找到的是一个非支配解的集合. 因此, 算法效果优劣评价方式有多种, 本文使用以下指标: Inverted Generational Distance (IGD)[20]和Set Coverage[21].

1) IGD. 假设P^*为已知的某个参考解集, A为某

算法找到的非支配解集.解$x \in A$和解$y \in P^*$的归一化欧氏距离为

$$d(x,y)=\sqrt{\sum_{i=1}^{2}\left(\frac{f_i(x)-f_i(y)}{f_i^{\max}-f_i^{\min}}\right)^2}. \tag{6}$$

其中: $f_i(\cdot)$为解的第i个目标函数值, $f_i^{\max}$和$f_i^{\min}$分别为P^*中该目标函数的最大值和最小值.

在此基础上,可计算A的IGD值为

$$\mathrm{IGD}(A,P^*)=\frac{1}{|P^*|}\sum_{y\in P^*}\min_{x\in A}d(x,y). \tag{7}$$

可见,IGD是A到P^*的所有解的最短距离的平均,该指标可反映A的多样性和收敛性.

2) Set Coverage.假设A和B是两个非支配解集,Set Coverage计算为

$$C(A,B)=\frac{|\{x\in B|\exists y\in A: y\prec x\}|}{|B|}. \tag{8}$$

可见,$C(A,B)$表示B中解被A中某个解支配的比例,若$C(A,B)$较大而$C(B,A)$较小,则表明在某种意义上A要优于B.

3.3 算法参数设置

MDGSO算法共有3个参数,其中p_s为种群大小,该参数设置越大,算法的计算代价越大,算法求解效果越好.通过初步实验,对产品数在20～100间的算例,取值在10～20间是较合适的,本文取p_s=15.而对另两个参数扰动规模d和追随者概率p,在相当的计算代价下,取值不同对算法效果可能有较大影响,因此,可通过计算实验来确定其取值.具体地,扰动规模d取值为2, 6, 10,而追随者概率p取值为0.2, 0.5, 0.8,这样构成9种组合.使用算法选择Ta11、Ta41、Ta71共3个算例,每个算例运行 10次,算法终止条件为$50mn$ ms,结果的IGD指标如表1所示.

表1 不同参数组合下算法所得的IGD值

d	p	IGD		
		Ta11	Ta41	Ta71
2	0.2	0.01	0.32	0.22
	0.5	0.00	0.24	0.17
	0.8	0.00	0.25	0.16
6	0.2	0.00	0.16	0.14
	0.5	0.00	0.08	0.00
	0.8	0.00	0.03	0.01
10	0.2	0.00	0.14	0.10
	0.5	0.00	0.18	0.07
	0.8	0.02	0.11	0.08

表1的结果显示,参数d和p对算法性能有较大影响.对于考虑的算例Ta11,不同参数组合均能较好地求解,这很可能是因为该算例规模较小(产品数为20).而对于另外两个较大规模的算例(产品数为50和100),在一定程度上可反映参数差异.总体上,$d=6$所得的IGD较小,而取值为2和10获得的IGD较大;对于参数p而言,取值为0.5和0.8总体上好于0.2,而$d=6$时,取0.8是更好的选择.另外,为分析参数耦合的影响,将实验得到的IGD值进行方差分析(ANOVA),结果见表2.由表2中最后一行对应的F值和p值可知,两个参数的交互作用对算法性能的影响并不显著.综上,本文MDGSO参数选择为: $d=6, p=0.8, p_s=15$.

表2 不同参数组合下算法性能的ANOVA结果

source		Sum of squares	Df	Mean square	F-value	p-value
Main effects	A: d	139.86	2	69.93	5.47	<0.001
	B: p	92.14	2	46.07	3.96	<0.001
Interactions	AB	148.46	4	37.12	1.13	0.149 0

3.4 MDGSO与NSGA-II、BMSA的性能比较

为验证算法效果,将MDGSO与两种较先进的多目标优化算法进行比较.一种是非支配排序遗传算法[22](NSGA-II), NSGA-II是一种采用精英策略和非支配排序的多目标遗传算法,在多目标优化领域得到广泛应用[23-24];另一种是bi-objective multi-start simulated annealing algorithm (BMSA), BMSA是一种采用多个初始点的多目标模拟退火算法[25].在算法性能测试环节,可将每个算法对每个算例运行10次,算法终止条件为$50mn$ ms.对每个算例,将各个算法10次运行得到的非支配解集合并,作为算法获得的非支配解集A,并将所有运行结果的非支配解集合并,作为参考解集P^*.以此为基础,计算IGD和Set Coverage.由于算例较多,按照算例规模,将计算结果求平均值后列于表3和表4,其中加粗字体表示在该算例规模下对应算法性能指标最好.

表3给出了3种算法在各种规模算例下获得的IGD值.根据IGD值的定义,该值越接近于零,表示算法找到的非支配解集在收敛性和多样性上越接近参考解集.可见,对于20个产品的小规模算例,在本文运行条件下MDGSO与BMSA、NSGA-II并无明显

表3 3种算法在各种规模算例下获得的IGD值

Instance size	IGD		
	MDGSO	BMSA	NSGA-II
20×5	**0.00**	**0.00**	**0.00**
20×10	**0.00**	0.03	0.03
20×20	**0.00**	**0.00**	**0.00**
50×5	**0.02**	0.29	0.18
50×10	**0.03**	0.12	0.12
50×20	**0.02**	0.11	0.14
100×5	**0.02**	0.40	0.84
100×10	**0.01**	0.44	0.58
100×20	**0.02**	0.29	0.49
Average	**0.01**	0.19	0.27

表 4　MDGSO与另外两种算法在各种规模算例下的Set Coverage值

Instance size	MDGSO (A) vs BMSA (B)		MDGSO (A) vs NSGA-II (C)	
	$C(A,B)$	$C(B,A)$	$C(A,C)$	$C(C,A)$
20×5	0.00	**0.01**	**0.00**	**0.00**
20×10	**0.00**	**0.00**	**0.00**	**0.00**
20×20	**0.00**	**0.00**	**0.00**	**0.00**
50×5	**0.80**	0.15	**0.79**	0.18
50×10	**0.64**	0.28	**0.66**	0.20
50×20	**0.73**	0.21	**0.80**	0.10
100×5	**0.97**	0.00	**0.96**	0.02
100×10	**0.96**	0.04	**0.95**	0.02
100×20	**0.86**	0.14	**0.99**	0.00
Average	**0.55**	0.09	**0.57**	0.06

差异,而对于50个和100个产品的中大规模算例而言,MDGSO的IGD值均小于另外两种算法,在性能上具有明显优势.由IGD值接近于零可推断,MDGSO所得解集非常接近参考解集.由表3还可看出,与NSGA-II相比,BMSA在小规模算例上效果相当,但随着规模增大,BMSA亦具有一定的优势.

表4从另一角度揭示了算法间的性能差异.将MDGSO与BMSA相比,对于规模为50和100的算例而言,$C(A,B)$较大而$C(B,A)$较小,表明MDGSO获得的解集要优于BMSA.MDGSO与NSGA-II相比,也有类似的结果.不过对于小规模算例,在这一指标上亦看不出MDGSO相对于另外两种算法的优势,甚至对20×5的算例,可能由于算法的随机性,BMSA指标竟略优于MDGSO.从平均指数上看,MDGSO总体上获得了优于另两种算法的结果.

为了更直观地表现算法的性能差异,这里选取Ta60、Ta80两个算例,画出各个算法获得的非支配解集,分别如图4和图5所示.注意,为使比较相对公平,图中的非支配解集并非单次运行结果的比较,而是对应算法运行10次所得解集的合并.另外,对于较小规模的算例,由于3种算法最终获得的解集基本无异,这里并未给出图示.由图4可见,相比于其他算法,在该算例下MDGSO算法获得的Pareto前沿的分布更具多样性,解的质量也相对较好.在图5中,BMSA算法和NSGA-II算法找到的Pareto前沿分布较为集中,

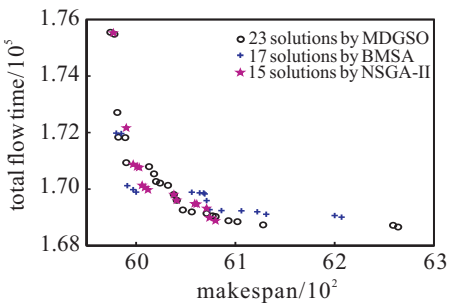

图 4　3种算法求解Ta60获得的非支配解集

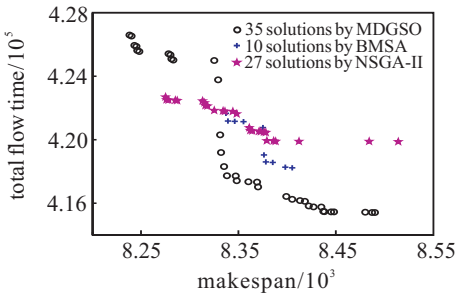

图 5　3种算法求解Ta80获得的非支配解集

且解的数量较少,因此,从Ta80算例的图示看,前述结果更为明显.

由MDGSO算法的种群初始化方法可知,初始的种群中既有启发式规则产生的性能较好解,又有随机产生的解,因此初始种群既具有多样性又具有较好的性能.可以认为,这是算法取得较好效果的原因之一.另一方面,在种群的每一代进化中,发现者在当前的某个前沿解上进一步探索,而追随者则将当前种群解与前沿解进行交叉,以获取较好的搜索区域;同时,巡逻者以一定的几率进行邻域搜索.3种角色的设计可以较好地平衡算法的探索性和开发性,这是算法具有较好性能的主要原因.值得一提的是,算法的每一代进化可能需要多次执行局部搜索,这将带来较大的计算开销,这也许是该算法对较小规模算例求解优势不突出的原因.

4　结　论

针对化工间歇过程中存在的零等待特性以及目前多目标优化的现状,本文研究了一类零等待约束下的多产品间歇过程多目标调度问题.考虑到生产周期和总流程时间对生产过程具有重要影响,以两者为最小化目标,提出了一种新型的多目标离散组搜索算法进行求解.算法采用NEH启发式规则及其变种产生初始解,通过嵌入基于插入的局部搜索设计了发现者、追随者和巡逻者的操作,从而不断搜寻Pareto非支配解集.在仿真研究中,首先对算法的参数进行了初步讨论,然后采用大量多产品间歇过程的调度算例进行了求解.结果表明,所提出的算法能较好地解决各种规模大小的零等待多产品间歇过程多目标调度问题,能获得质量较好的非支配解集.与非支配排序遗传算法和模拟退火算法相比,所提出的算法获得的非支配解集具有较好的多样性和优越性,特别是在中大规模的问题上,算法具有更显著的效果.

参考文献(References)

[1]　李宏光,刘骥鹏,黄静雯.自适应变步长迭代动态规划方法及其在间歇过程优化中的应用[J].控制与决策,2015,30(11):2048-2054.
(Li H G, Liu J P, Huang J W. Self-adaptive variable-step

approach for iterative dynamic programming with applications in batch process optimization[J]. Control and Decision, 2015, 30(11): 2048-2054.)

[2] 陆宁云, 王福利, 高福荣, 等. 间歇过程的统计建模与在线监测[J]. 自动化学报, 2006, 32(3): 400-410.
(Lu N Y, Wang F L, Gao F R, et al. Statistical modeling and online monitoring for batch processes[J]. Acta Automatic Sinica, 2006, 32(3): 400-410.)

[3] 关守平, 付冲, 侯金芬. 一种新的间歇过程多模式辨识方法[J]. 东北大学学报: 自然科学版, 2015, 36(3): 327-358.
(Guan S P, Fu C, Hou J F. A new multi-mode identification method of batch process[J]. J of Northeastern University: Natural Science, 2015, 36(3): 327-358.)

[4] 王静, 胡益, 侍洪波. 基于GMM的间歇过程故障检测[J]. 自动化学报, 2015, 14(5): 889-905.
(Wang J, Hu Y, Shi H B. Fault detection for batch processes based on gaussian mixture model[J]. Acta Automatic Sinica, 2015, 14(5): 889-905.)

[5] Mendez C A, Cerda J, Grossmann I E, et al. State-of-the-art review of optimization methods for short-term scheduling of batch processes[J]. Computers & Chemical Engineering, 2006, 30(6): 913-946.

[6] 唐琦, 汪恭书, 苏丽杰. 化工工业多品种成批轮番生产的集成分批与调度[J]. 控制与决策, 2015, 30(2): 289-295.
(Tang Q, Wang G S, Su L J. Integrated lotsizing and scheduling for multi-variety batch production by turns in chemical industry[J]. Control and Decision, 2015, 30(2): 289-295.)

[7] Belaid R, T'kindt V, Esswein C. Scheduling batches in flowshop with limited buffers in the shampoo industry[J]. European J of Operational Research, 2012, 223(2): 560-572.

[8] 徐震浩, 李青青, 顾幸生. 基于DEPSO的模糊时间ZW多产品厂间歇调度[J]. 控制与决策, 2015, 30(12): 2275-2279.
(Xu Z H, Li Q Q, Gu X S. A study of the DEPSO-based multiproduct plantsbatch scheduling under uncertainty with zero wait[J]. Control and Decision, 2015, 30(12): 2275-2279.)

[9] Hall N G, Sriskandarayah C. A survey of machine scheduling problems with blocking and no-wait in process[J]. Operations Research, 1996, 44(3): 510-525.

[10] 杨玉珍, 顾幸生. 多目标零等待间歇生产过程多任务调度[J]. 化工学报, 2013, 64(12): 4578-4584.
(Yang Y Z, Gu X S. Multi-objective no-wait multi-task scheduling problem of batch process[J]. CIESC J, 2013, 64(12): 4578-4584.)

[11] Dong M, Wang N. A novel hybrid differential evolution approach to scheduling of large-scale zero-wait batch processes with setup times[J]. Computers & Chemical Engineering, 2012, 45: 72-83.

[12] Wang S, Liu M. A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem[J]. Computers & Operations Research, 2013, 40(4): 1064-1075.

[13] Qian B, Wang L, Hu R, et al.A DE-based approach to no-wait flow-shop scheduling[J]. Computers & Industrial Engineering, 2009, 57(3): 787-805.

[14] Khalili M, Naderi B. A bi-objective imperialist competitive algorithm for no-wait flexible flow lines with sequence dependent setup times[J]. Int J of Advanced Manufacturing Technology, 2015, 76(1): 461-469.

[15] Pan Q K, Wang L, Qian B. A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems[J]. Computers & Operations Research, 2009, 36(8): 2498-2511.

[16] He S, Wu Q H, Saunders J R. A novel group search optimizer inspired by animal behavioral ecology[C]. IEEE CEC'2006. Vancouver: IEEE Press, 2006: 1272-1278.

[17] Nawaz M, Enscore J E E, Ham I. A heuristic algorithm for the m-machine n-job flow-shop sequencing problem[J]. OMEGA, 1983, 11(1): 91-95.

[18] Wang L, Pan Q K, Tasgetiren M F. Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms[J]. Expert Systems with Applications, 2010, 37(12): 7929-7936.

[19] Taillard E. Benchmarks for basic scheduling-problems[J]. European J of Operations Research, 1993, 64(2): 275-285.

[20] Coello C, Cortés N. Solving multiobjective optimization problems using an artificial immune system[J]. Genetic Programming and Evolvable Machines, 2005, 6(2): 163-190.

[21] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results[J]. Evolutionary Computation, 2000, 8(2): 173-195.

[22] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Trans on Evolutionary Computation, 2002, 6(2): 182-197.

[23] 李海燕, 井元伟. 基于NSGA-II的具有多目标子学科的协同优化方法[J]. 控制与决策, 2015, 30(8): 1497-1503.
(Li H Y, Jing Y W. Multi-objective collaborative optimization method based on NSGA-II for MDO problems with multi-objective subsystem[J]. Control and Decision, 2015, 30(8): 1497-1503.)

[24] Yuan Y, Xu H. Multiobjective flexible job shop scheduling using memetic algorithms[J]. IEEE Trans on Automation Science and Engineering, 2015, 12(1): 336-353.

[25] Lin S W, Ying K C. Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated-annealing algorithm[J]. Computers & Operations Research, 2013, 40(6): 1625-1647.

（责任编辑：孙艺红）