Journal of System Simulation

Volume 32 | Issue 11 Article 20

11-17-2020

Hyper-heuristic DE Algorithm for Solving Zero-wait Fermentation Process Schedulinge

Shen Peng
Engineering Research Center of Internet of Things Technology Applications Ministry of Education,Wuxi 214122,China;

Wang Yan
Engineering Research Center of Internet of Things Technology Applications Ministry of Education,Wuxi 214122,China;

Zhicheng Ji
Engineering Research Center of Internet of Things Technology Applications Ministry of Education,Wuxi 214122,China;

Jianhua Zhang
Engineering Research Center of Internet of Things Technology Applications Ministry of Education,Wuxi 214122,China;

Hyper-heuristic DE Algorithm for Solving Zero-wait Fermentation Process Schedulinge

Abstract

Abstract: *A class of zero-wait fermentation process scheduling issues with batch process characteristics are researched. In order to solve the problem of easy deterioration in the process, a super heuristic difference algorithm is proposed, and the maximum makespan is minimized as the optimization goal.* The algorithm is divided into two layers. The upper layer is an improved adaptive differential evolution algorithm to select and sort the heuristic operations in lower layer. The lower layer is combined and sorted into a new algorithm to operate on the problem domain, adding simulated annealing algorithm to avoid falling into local optimization. The method has a learning mechanism and the strong generalization ability for different problems. The effectiveness of the algorithm is verified by comparing with the actual production. The result shows that the algorithm has better performance than the traditional heuristic algorithm.

Keywords

batch process, hyper-heuristic evolution algorithm, zero-wait, simulated annealing

Recommended Citation

Shen Peng, Wang Yan, Ji Zhicheng, Zhang Jianhua. Hyper-heuristic DE Algorithm for Solving Zero-wait Fermentation Process Schedulinge[J]. Journal of System Simulation, 2020, 32(11): 2235-2243.

第 32 卷第 11 期 系统仿真学报© Vol. 32 No. 11
2020 年 11 月 Journal of System Simulation Nov., 2020

超启发式 DE 算法求解零等待发酵工艺调度

沈鹏，王艳，纪志成，张建华
(江南大学 物联网技术应用教育部工程研究中心，江苏 无锡 214122)

摘要：*为解决具备间歇过程特点的零等待发酵工艺调度过程中易变质的难点，提出了一种超启发式差分算法，并将最小化最大完工时间设置为优化目标。*此算法分为 2 层，高层为改进的自适应差分进化算法，来对低层的启发式操作进行选择排序。而低层组合排序成了新的算法对问题域进行操作，加入模拟退火算法避免陷入局部最优。所提出的方法具有学习的机制，对不同问题具有较强的泛化能力。通过测试算例和实际生产进行算法的比较和求解验证了此算法的有效性，结果表明所提出的算法比传统的启发式算法性能更优。

关键词：间歇过程；超启发式算法；零等待；模拟退火
中图分类号：TP278 文献标识码：A 文章编号：1004-731X (2020) 11-2235-09
DOI: 10.16182/j.issn1004731x.joss.20-FZ0399

Hyper-heuristic DE Algorithm for Solving Zero-wait Fermentation Process Schedulinge

Shen Peng, *Wang Yan*, *Ji Zhicheng*, *Zhang Jianhua*
(Engineering Research Center of Internet of Things Technology Applications Ministry of Education, Wuxi 214122, China)

Abstract: *A class of zero-wait fermentation process scheduling issues with batch process characteristics are researched. In order to solve the problem of easy deterioration in the process, a super heuristic difference algorithm is proposed, and the maximum makespan is minimized as the optimization goal.* The algorithm is divided into two layers. The upper layer is an improved adaptive differential evolution algorithm to select and sort the heuristic operations in lower layer. The lower layer is combined and sorted into a new algorithm to operate on the problem domain, adding simulated annealing algorithm to avoid falling into local optimization. The method has a learning mechanism and the strong generalization ability for different problems. The effectiveness of the algorithm is verified by comparing with the actual production. The result shows that the algorithm has better performance than the traditional heuristic algorithm.

Keywords: batch process; hyper-heuristic evolution algorithm; zero-wait; simulated annealing

引言

发酵工艺调度是一种典型的间歇过程。间歇过程是指将有限量的物料，按规定的加工顺序，在一个或多个设备中加工，以获得有限量的产品。间歇生产过程适合于小批量、多品种的精细化工生产过程[1-3]，同时具有连续过程和离散过程的特点，在化工生产中占据重要的地位，受到越来越多研究者的关注[4-6]。目前间歇过程广泛应用于制药、纺织、食品等行业。

发酵工艺的调度问题在于研究如何将各个产品的生产安排到每台设备上，从而针对某些需求的

收稿日期：2020-06-23 修回日期：2020-07-13；
基金项目：国家自然科学基金(61973138)，国家重点研发计划(2018YFB1701903)；
作者简介：沈鹏(1994-)，男，江苏泰州，硕士生，研究方向为智能调度与群智能算法；王艳(1978-)，女，江苏盐城，博士，教授，研究方向为制造系统能效优化。

性能指标进行有针对性的优化，如生产周期最短、能耗最小等。合理有效的调度方案可以显著提高生产效率、节约成本，是现代制造技术和现代管理技术的核心技术，因此发酵工艺中的间歇过程调度研究很受重视。发酵工艺中的间歇过程物料转移策略可按照中间储罐设置方式的不同分为零等待策略无中间储罐、有限中间储罐、无限中间储罐以及含有多种储罐设置方式的混合中间储罐[7]。当设备中的中间产物不在中间储罐中停留直接转移到下一个设备中，其中没有延迟，这种生产就需要采用零等待策略。零等待约束应用于各种间歇过程之中。例如在冶金过程的炼钢-连铸过程中，零等待约束可以减少钢水在空气中的温降，这对于热送热装的生产方式十分重要，同时铁被烧热要立刻铸型，否则会影响冷却后的形状和质量；在化工生产中，很多工序需要采用零等待策略以防止产品污染、变质和挥发等影响；在食品生产中为了防止变质和细菌的感染也需要进行连续的生产。因此，研究该问题具有重要的理论价值和实际意义。

类比于流水作业(flow-shop)问题，对于发酵工艺中的间歇过程调度问题的解决方法主要可以分为 2 类：精确方法和近似方法。精确方法包括混合整数线性规划法、分支定界法等，但是由于此类问题在数学上是 NP 完全的将会导致组合爆炸，所以精确方法只能使用于小规模问题。20 世纪 70 年代之后，人们不再追求用精确的算法求出问题的最优解，而是用近似算法在可以接受的时间内寻求问题的满意解，因此提出用启发式算法来解决此类问题。

然而启发式算法对于不同领域的操作规则是不同的，缺乏泛化能力，有较高的实现难度，而且改进的效果也难以保证。而超启发式算法是一类新型智能优化算法，通过高层的启发式操作组成不同的低层启发式操作，因此超启发式算法的高层算法是对一系列操作进行操作，不直接作用于具体问题，从而超启发式算法可以脱离具体问题情境，避免启发式算法的缺点。同时超启发式算法是对低层操作组合的算法，具有学习的机制，整体性能要优于基于单一算法的性能。

针对此问题，已经涌现出了一些研究成果，邓冠龙等[8]考虑了零等待约束下多产品间歇过程的多目标调度问题，利用多目标离散组搜索算法进行求解。杨玉珍等[9]则在多任务间歇过程的调度问题上提出了带存储的完全搜索方法。文献[10]通过两阶段遗传算法来解决零等待流水车间调度问题，并取得了较好的效果。文献[11]考虑了具有准备时间的柔性流水车间多目标调度。

上述算法虽然在部分问题的求解中能够取得较好的结果，但是由于采用固定的算法，对于不同的场景取得的效果可能差异很大。本文以最大完工时间为指标，采用混合超启发式差分进化算法进行优化调度来求解相应问题，能够通过低层算法的组合进行优化，以高层启发式算法进行组合求解，从而对不同的问题都有较好的效果。

1 零等待多产品间歇过程问题概述

1.1 调度模型

n 个产品在 m 台设备上进行加工，每个产品需要按照第 1 台设备，第 2 台设备，直到第 m 台设备的顺序依次加工。n 个加工的产品 $J_i(i=1, 2,..., n)$ 在第 $k(k=1, 2,..., m)$台设备 M_k 上的处理过程记为工序 O_{ik}，对应的处理时间为 $p(i, k)$，为满足零等待的约束，工序 O_{ik} 的完成时刻必须等于工序 $O_{i,k+1}$ 的开始时刻，零等待代表着产品从第一台设备上开始加工之后，将会沿着设备顺序无延迟的一直进行，直到最后一台设备上的产品加工完成。在零等待策略下，每个产品的完工时间总是大于前一个工序的产品的完工时间，从而导致总的完成时间会比其他策略下的多产品间歇过程要长。另外，在零等待多产品间歇过程中，还应该设定下列约束条件：

1) 所有产品和设备在零时刻是可用的；

2) 处理时间>0，且是确定的；

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。

DOI: 10.16182/j.issn1004731x.joss.20-FZ0399

在实际的间歇过程调度中的主要目的就是尽可能的让最大完工时间最小。如图 1 所示，以某三产品三设备的零等待间歇流程的甘特图为例，其中 M 代表设备，J 代表产品。

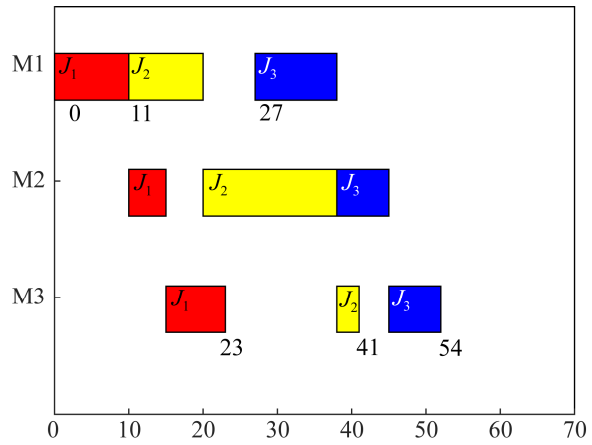


图 1 某零等待多产品间歇过程甘特图

Fig.1 Gantt chart of a zero-wait multi-product batch process

根据图 1 的产品号 J_1–J_2–J_3 的先后次序，在零等待间歇过程调度中采用 Schuster[12]提出的非延迟时间表分配策略。最大完工时间为 54 s。若采用不同的产品处理次序，最大完工时间也将随之改变。由此可见对于不同的调度方案将会直接的影响产品的最大完工时间。

1.2 参数定义

一些用于描述间歇调度问题和数学模型的变量符号归纳总结为：

J_i：第 i 个加工产品，i=1,2,...,n

M_k：第 k 台加工设备，k=1,2,...,m

O_{jk}：产品 J_i 在第 k 台设备上的工序

π_j：产品号，j=1,2,...,n

π=$\{\pi_1,\pi_2,...,\pi_n\}$：一个调度解

$d(\pi_{j-1},\pi_j)$：第一台机器上的产品 π_{j-1} 和 π_j 开始时刻的最小延迟

$C(\pi_j,m)$：π_j 在设备 m 上的完成时间

$C_{\max}(\pi)$：π 对应的最大完工时间

$p(j,k)$：产品 π_j 在设备 M_k 上的处理时间

d_j：π_j 的任务交货期

1.3 优化函数

最大完工时间(Makespan，简称 Ms)指完成所有所需加工工件的全部时间。为了使调度解 π 对应的最大完工时间最小，则应该在约束条件下尽早的开始加工产品。则有

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

在此基础上可以计算

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

因此其计算公式为：

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

2 零等待多产品间歇过程优化调度

2.1 超启发式算法

在调度问题领域已有不少学者采用各种启发式算法进行解决，但是正如文献[13]中指出，每一种启发式算法都是在特定情况下表现出优异的结果。针对此种情况，“寻找启发式算法的启发式算法”超启发式算法，其有良好的通用性，适用于来解决调度问题。

超启发式包括 2 层：低层问题域是不同的启发式搜索策略，高层策略域为调用低层启发式策略的选择器。根据反馈学习的来源可将其分为不学习、离线学习和在线学习 3 种，在线学习又包括元启发、强化学习、函数选择 3 种选择方法。事实证明，用启发式算法作为高层搜索策略已取得良好的结果，这也是为什么“寻找启发式算法的启发式算法”这种更一般定义的产生。

本文选择自适应差分进化算法来选择低层启发式算法，差分进化(Differential Evolution, DE)算法是一种基于群体的启发式算法，具有高效的全局优化能力，并且有待定参数少、收敛速度快和不易陷入局部最优的特点。

2.2 编码与解码

在高层策略域中，通过对于 6 种低层启发式操作的组合来成为一种新的启发式算法，染色体的长度等于低层启发式的个数，染色体中可以出现相同的启发式操作。解码操作就是低层的个体按照染色体中从左到右的顺序依次执行染色体基因中对应的启发式操作。若高层中一个个体为 1-2-6-4-3-3，则按照随机交换、相邻交换、截取、后项插入、前项插入、前项插入的顺序对低层个体进行操作。由于超启发式算法的低层启发式操作的操作次数较多，会在一定程度上导致染色体被过多重组变异，从而降低算法速度，因此提出一种精英保留规则：若新解的适应值小于旧解的适应值，则用新解代替旧解并且跳过剩余的低层启发式操作，否则继续执行。

由于零等待多产品间歇过程调度需要优化的是产品序列，因此在低层问题域中个体的染色体编码直接采用序列本身进行编码。解码时根据非延时表分配策略进行解码，便能得到此个体对应的各个目标函数的值。

2.3 差分进化算法

2.3.1 基本差分进化算法

差分进化算法是一种基于群体的启发式算法，具有高效的全局优化能力，并且有待定参数少、收敛速度快和不易陷入局部最优的特点。与遗传算法有相似的进化流程，包含变异、交叉和选择操作。

1) 生成初始种群

对于个体总数为 N，个体维数为 D 维的种群初始化，定义个体为 X，其生成种群公式为：

$$x_{i,j}=l_j+r\cdot(u_j-l_j),j=1,2,\cdots,D \tag{4}$$

式中：r 为一个随机生成的(0, 1)之间的随机数；u_j 和 l_j 分别为第 j 个染色体取值的上界和下界。

2) 变异操作

定义种群个体为 X，个体总数为 N，对于 m 维的个体，第 G 代的第 i 个个体表示为 $X_{i,j}^G=(\mathrm{x}_{i,1}^G,\mathrm{x}_{i,2}^G,\cdots,\mathrm{x}_{i,D}^G)$。从中随机选择 3 个个体 X_{r1}^G，X_{r2}^G 和 X_{r3}^G，其中 $r1\neq r2\neq r3$，则变异操作为：

$$V_i^{G+1}=X_{r1}^G+F\cdot(X_{r2}^G-X_{r3}^G) \tag{5}$$

式中：F 为缩放因子，对差分向量进行缩放，取值范围为[0, 2]，通常取 0.5。

3) 交叉操作

交叉操作通过概率的方式生成新的个体，交叉操作为：

$$U_{i,j}^G=\begin{cases}V_{i,j}^{G+1},r\leqslant CR\ or\ j=rand(1,D)\\X_{i,j}^G,r>CR\ and\ j\neq rand(1,D)\end{cases} \tag{6}$$

式中：CR 为交叉概率，范围在[0, 1]，r 为[0, 1]内生成的随机数；$rand[1, D]$为在[1, D]内生成的均匀分布的整数，从而能确保 U_i^G 中至少有一维的分量是从变异个体 V_i^G 中取得，而避免与父代个体 X_i^G 相同。

4) 选择操作

DE 算法采用贪婪选择模式，从父代个体和实验个体中选择较优目标值的个体进入下一代种群。通常将较小的目标值视为较优，选择操作为：

$$X_i^{G+1}=\begin{cases}U_i^G,f(U_i^G)\leqslant f(X_i^G)\\X_i^G,f(U_i^G)>f(X_i^G)\end{cases} \tag{7}$$

2.3.2 改进的自适应策略

1) 变异策略

缩放因子 F 是影响算法的种群多样性和收敛速度的重要参数，传统的差分进化算法由于其采用固定取值，容易导致算法结果早熟与收敛速度变慢。本文采用自适应策略调整缩放因子 F：

$$F=(F_{\max}-F_{\min})\frac{G_T-G_t}{G_T}+F_{\min} \tag{8}$$

式中：$F_{\max}$ 为设置的变异因子的最大值；$F_{\min}$ 为设置的变异因子的最小值；G_T 为最大迭代次数；G_t 为当前迭代次数。根据式(8)可以使得在算法初期的时候 F 取值较大，有利于维持种群的多样性，同时 F 随着迭代次数的将会逐渐变小，有利于提高算法的收敛速度和计算精度。

2) 交叉策略

种群的交叉程度由交叉因子 CR 决定，CR 取值越小，则种群的多样性越小，但是 CR 越大也会导致种群收敛过慢，本文采用自适应策略调整交叉

DOI: 10.16182/j.issn1004731x.joss.20-FZ0399

因子 CR：

$$CR = (CR_{\max} - CR_{\min})\frac{G_t}{G_T} + CR_{\min} \tag{9}$$

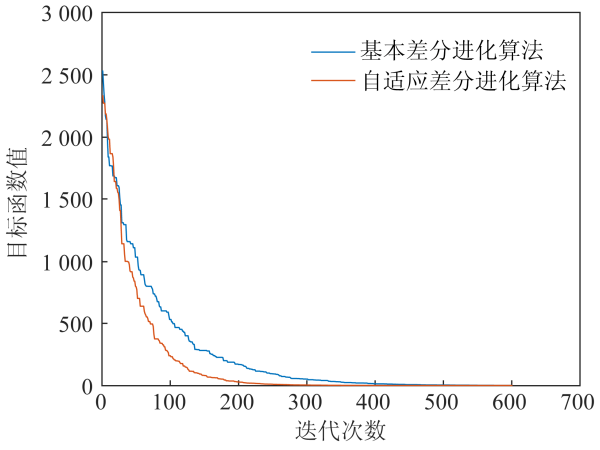
式中：$CR_{\max}$ 为设置的变异因子的最大值取；$CR_{\min}$ 为设置的变异因子的最小值；G_T 为最大迭代次数；G_t 为当前迭代次数。交叉因子可根据迭代次数自适应调整，能够维持种群多样性和收敛速度的平衡。

为了验证自适应差分进化算法的优越性，选择 2 个标准函数进行测试，f_1，f_2 分别为：

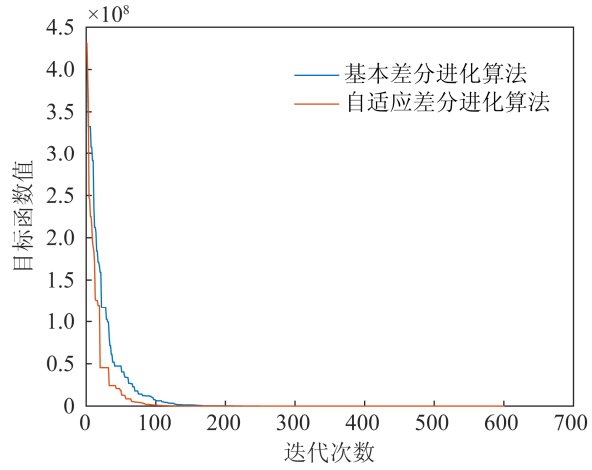
$$f_1 = \frac{1}{4\,000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{10}$$

$$f_2 = \sum_{i=1}^{D-1}[(x_i - 1)^2 + 100(x_i^2 - x_{i+1})^2] \tag{11}$$

标准函数的测试结果分别如图 2(a)，(b)所示。



(a) f_1



(b) f_2

图 2 算法性能对比

Fig. 2 Performance comparison of algorithms

从图 2 可以明显看出，在 2 个标准函数下自适应差分进化算法相对于基本差分进化算法都能能够更快的搜索到理想值，拥有更高的收敛精度，从而证明了自适应差分进化算法的优越性。

2.4 低层启发式操作

低层启发式操作被高层的启发式操作用来选择组成新的算法用于搜索解空间，组成的新算法的好坏和搜索能力与解的质量有关。由于复杂度较高的底层启发式操作会降低算法的运行效率从而加长求解时间，所以在设计低层启发式操作时不宜太过复杂。

本文的低层启发式操作如下：

LLH_1：随机交换操作。从工序排序中随机选取两位进行交换。

LLH_2：相邻交换操作。随机选择一个位置，将它与后一个工序交换，如果选择的位置是最后一个工序就与第一个工序交换。

LLH_3：前项插入操作。随机从工序排列序列中选择一个位置，将随机选择的一个工序插入到这个位置之前。

LLH_4：后项插入操作。随机从工序排列序列中选择一个位置，在这个位置之后插入一个随机选择的工序。

LLH_5：逆序操作。在工序排列序列中选择 2 个不同的随机位置，将它们之间的连续的子序列逆序排序。

LLH_6：截取操作。在工序排列序列中选择 2 个不同的随机位置，将它们之间的连续子序列插入到工序排列序列首端。

2.5 模拟退火算法

模拟退火搜索过程中有概率突跳的能力，通过迭代策略，较小的概率接受较差的解来避免陷入局部最优。通过将模拟退火算法引入超启发式算法，实现冷却机制，每次迭代根据冷却系数 η 来降温，温度变化公式(12)为：

$$T_0 = T \times \eta \tag{12}$$

式中：T_0 为下次迭代温度；T 为当前温度，初始温度 T_S 取 200，冷却系数 η 取 0.9，同时按照公式(13)在低层启发式算法中进行选择：

$$P=\exp\left[-\frac{\Delta f}{T}\right] \tag{13}$$

式中：Δf 循环中每次的新解与旧解之差，否则随机生成一个 0~1 之间的实数与概率 p 进行比较，若大于 p 则接受新解，否则保留旧解。

2.6 算法流程

如上述的整个的算法流程，整个算法的框架为：

混合超启发式算法框架

1：初始化策略域种群和问题域种群
2：设定初始温度
3：评价 2 个种群
4：while 终止条件不满足 do
5：　对策略域种群进行选择、交叉、变异
6：　while 评价完所有个体 or 用完当前低层操作 do
7：　根据策略个体中的低层启发式操作获得新的解
8：　if 新解优于旧解 then
9：　替换旧解
10：　else if 大于接受概率 P then
11：　用新解替换旧解
12：　else 保留旧解
13：　end if
14：end while
15：对策略域种群和问题域种群进行保优
16：降温
17：end while

3 算法测试

3.1 混合超启发式差分进化算法

为测试超启发式差分进化算法的性能，本文选取 Taillard 算例集[14]和 Reeves 算例集[15]中的算例进行算法测试。

仿真实验的编程语言为 MatlabR2014a，电脑硬件参数为：处理器 Intel(R) Core(TM) i5-4210U CPU @ 3.70 GHz，内存 8 GB。

将本文算法与其他智能算法进行比较，本文以平均相对百分比偏差(Average Relative Percentage Deviation，ARPD)为评价标准，ARPD 计算公式为：

$$ARPD=\frac{1}{M}\sum_{i=1}^{M}\frac{C_i-C^*}{C^*}\times 100 \tag{14}$$

式中：C_i 为算法在第 i 此实验中得到的解；C^*为文献[16]中计算的结果；M 为实验次数。当 $ARPD$ 值越小时便代表算法的结果越优。

在本文中，对于种群大小，该参数设置的越大，算法的计算量越大，求解效果越好，经过对 Reeves 算例的初步试验可知对于产品数在 20~100 间的算例，把种群大小设定在 10~20 之间是比较合适的，本文将种群大小设定为 15。而算法中差分进化算法部分的缩放因子 F，交叉因子 CR 和模拟退火机制中的冷却系数 η 为影响算法效果的 3 个主要参数，为了考察这些参数对于算法求解结果质量的影响，通过计算试验来确定其取值，其中缩放因子 F 取值为 0.1~0.5 和 0.3~0.6，交叉因子 CR 取值为 0.4~0.8 和 0.6~0.9，冷却系数 η 取值为 0.85 和 0.9。不同参数组合在 Rec7，Rec33 和 Rec39 下所得结果如表 1 所示。

表 1　不同参数下的 ARPD 值
Tab. 1　ARPD values with different parameters

F	CR	η	ARPD		
			Rec7	Rec33	Rec39
0.1~0.5	0.4~0.8	0.85	0.04	0.37	0.36
0.1~0.5	0.4~0.8	0.9	0.01	0.23	0.23
0.1~0.5	0.4~0.8	0.95	0.04	0.37	0.28
0.1~0.5	0.6~0.9	0.85	0.03	0.37	0.34
0.1~0.5	0.6~0.9	0.9	**0.00**	0.23	0.21
0.1~0.5	0.6~0.9	0.95	0.02	0.36	0.27
0.3~0.6	0.4~0.8	0.85	0.02	0.35	0.32
0.3~0.6	0.4~0.8	0.9	**0.00**	**0.21**	0.21
0.3~0.6	0.4~0.8	0.95	0.02	0.35	0.27
0.3~0.6	0.6~0.9	0.85	0.01	0.35	0.31
0.3~0.6	0.6~0.9	0.9	**0.00**	**0.21**	**0.19**
0.3~0.6	0.6~0.9	0.95	0.02	0.34	0.26

DOI: 10.16182/j.issn1004731x.joss.20-FZ0399

表 1 结果显示，影响结果最大的是算法中模拟退火机制中的冷却系数 η 对求解结果的质量影响最大，因为降温系数作用于低层启发式操作，直接影响着算法对解空间的搜索能力，降温系数过小会降低局部搜索的能力，若过大将导致低层启发式操作能够接受更多的较差解，前期有助于跳出局部最优，但是会让后期的收敛性变差。

因此，根据表 1 的结果，算法中的缩放因子 F 为 0.3~0.6 之间，交叉因子 CR 设置为 0.6~0.9 之间，冷却系数 η 取 0.9。

在 Taillard 算例集中选取的从产品数 20、机器数 5 到产品数 500、机器数 20 的 90 个算例，它们在本文算法与文献[17]中的 HGA 算法和 TMIIG 算法的 ARPD 比较结果如表 2 所示，其中每个算例运行 10 次。从中可以看出本文算法在小规模到大规模中的产品算例中都有较好的表现，求解质量要超过了其余智能算法。

表 2 算法比较结果
Tab. 2 Comparison results of algorithms

Instance size	ARPD		
	HGA	TMIIG	本文算法
20×5	0.03	**0.00**	**0.00**
20×10	0.06	**0.00**	**0.00**
20×20	0.03	**0.00**	**0.00**
50×5	0.55	0.18	**0.09**
50×10	0.47	**0.06**	**0.06**
50×20	0.47	**0.05**	**0.05**
100×5	0.84	0.42	**0.21**
100×10	0.85	0.12	**0.06**
100×20	0.91	0.10	**0.08**
200×10	1.68	0.39	**0.29**
200×20	1.52	0.21	**0.18**

3.2 实际案例分析

3.2.1 N-乙酰氨基葡萄糖工艺流程

以用微生物法合成 N-乙酰氨基葡萄糖[18]为例，从种子培养到最后的包装入库分为 3 个大的过程：发酵、浓缩和提取。在每个过程中又包括各种操作，每个操作的周期都要进行控制，为防止在合成过程中出现污染、变质等情况，需要间歇的进行。合成 N-乙酰氨基葡萄糖的基本步骤如图 3 所示。

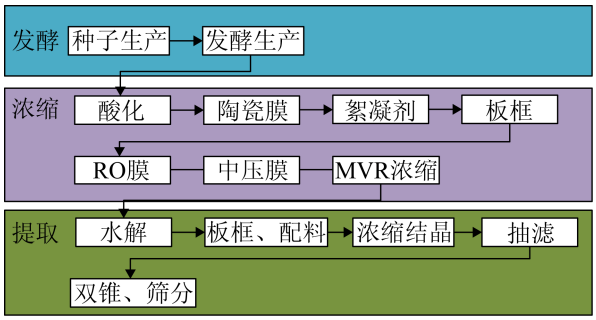


图 3 N-乙酰氨基葡萄糖工艺流程
Fig. 3 Technological process of N-acetylglucosamine

3.2.2 工艺流程调度分析

为了进一步验证本文提出的算法的有效性，本节将其用于求解山东泰安某生物发酵公司的 N-乙酰氨基葡萄糖生产调度问题。因为在发酵、浓缩、提取过程中需要严格的把控时间，防止产物的变质或发生次生反应，在上一个阶段完成之后便立即转入下一个阶段。此公司某月份的 10 个产品总共 14 个步骤建立零等待间歇过程调度模型，根据表 3 的产品加工信息利用本文算法求解得出的其中一个最优序列为：{4，3，9，7，6，8，5，2，10，1}。根据此序列的调度方案得到的最大完工时间为 591，公司的调度员提供的调度方案为{8，6，1，5，2，3，9，4，0，7}，此调度方案求解出的最大完工时间为 615。可以看出由本文算法求解得出的调度方案明显优于调度员提供的调度方案，此外调度员的计算方案时间为 13 min，而本文算法运行 20 次耗时为 32 s，说明本文所提出的混合超启发式算法在实际的间歇过程调度中具有较好的应用前景。

表 3 产品加工信息
Tab.3 Product processing information

流程	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
种子生产	11	13	16	9	11	12	15	14	17	12
发酵生产	57	46	54	48	53	49	41	52	54	48
酸化	2	4	2	4	3	2	3	4	3	1
陶瓷膜	5	7	6	3	6	7	5	4	2	2
絮凝剂	1	3	1	4	3	4	2	3	1	2
板框	2	6	4	8	1	4	3	1	7	2
RO 膜	5	2	8	1	7	4	6	3	2	6
中压膜	5	7	3	7	5	6	4	8	3	5
MVR 浓缩	6	20	12	3	4	18	6	23	15	17
水解	10	14	9	11	10	12	13	13	6	8
板框、配料	12	20	4	23	14	8	21	23	15	17
浓缩结晶	20	13	26	24	16	23	20	17	19	22
抽滤	8	5	12	10	9	8	12	13	5	11
双锥、筛分	4	7	6	3	5	6	4	7	8	6

4 结论

针对发酵过程中的零等待约束下如何最小化最大完工时间问题。由于启发式算法对具体问题操作需要进行不同程度的适应，采用超启发式算法直接对高层启发式操作进行操作，低层启发式算法根据具体的问题进行设计，能够对有效的操作进行选优。在上层问题域采用改进的差分进化算法，具有高效的全局优化能力，同时考虑到算法在求解过程中容易陷入局部最优，嵌入模拟退火算法来增加算法在陷入局部最优时跳出的能力，增加算法的求解效果和深度。对比实验表明，该算法在不同规模的算例下都有较好的表现，同时在实际的间歇过程调度中也能够求解出质量较高的解，在解决此类零等待间歇调度问题上具有一定的有效性。

参考文献:

[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 Hongguang, Liu Jipeng, Huang Jingwen. 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] 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): 327-330.
Guan Shouping, Fu chong, Hou Jinfen. A New Multi-mode Identification Method of Batch Process[J]. Journal of Northeastern University (Natural Science), 2015, 36(3): 327-330.

[3] 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 Ningyun, Wang Fuli, Gao Furong, et al. Statistical Modeling and Online Monitoring for Batch Processes[J]. Acta Automatica Sinica, 2006, 32(3):400-410.

[4] Belaid R, T Kindt V, Esswein C. Scheduling Batches in Flowshop with Limited Buffers in the Shampoo Industry[J]. European Journal of Operational Research (S0377-2217), 2012, 223(2): 560-572.

[5] 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 qi, Wang Gongshu, Su Lijie. Integrated Lotsizing and Scheduling for Multi-variety Batch Production by Turns in Chemical industry[J]. Control and Decision, 2015, 30(2): 289-295.

[6] 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 Zhenhao, Li Qingqing, Gu Xingsheng. A Study of the DEPSO-based Multiproduct Plants Batch Scheduling under Uncertainty with Zero Wait[J]. Control and Decision, 2015, 30(12): 2275-2279.

[7] 李凡华. 基于遗传算法的间歇化工过程调度及其在线

调整研究[D]. 青岛: 青岛科技大学, 2006.
Li Fanhua. Study on Batch Chemical Processes Scheduling and On-line Modification Based on Genetic Algorithms[D]. Qingdao: Qingdao University of Science and Technology, 2006.

[8] 邓冠龙, 田广东, 顾幸生. 零等待约束下多产品间歇过程的多目标调度[J]. 控制与决策, 2017, 32(3): 474-480.
Deng Guanlong, Tian Guangdong, Gu Xingsheng. Multi-objective Scheduling for Multi-product Batch Process with No-wait Constraint[J]. Control and Decision, 2017, 32(3): 474-480.

[9] 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 Yuzhen, Gu Xingsheng. Multi-objective No-wait Multi-task Scheduling Problem of Batch Process[J]. CIESC Journal, 2013, 64(12): 4578-4584.

[10] Wang S, Liu M. A Genetic Algorithm for Two-stage No-wait Hybrid Flow Shop Scheduling Problem[J]. Computers & Operations Research (S0305-0548), 2013, 40(4): 1064-1075.

[11] Khalili M, Naderi B. A Bi-objective Imperialist Competitive Algorithm for No-wait Flexible Flow Lines with Sequence Dependent Setup Times[J]. The International Journal of Advanced Manufacturing Technology (S0268-3768), 2015, 76(1): 461-469.

[12] Schuster C. No-wait Job Shop Scheduling: Tabu Search and Complexity of Subproblems[J]. Mathematical Methods of Operations Research (ZOR) (S1432-2994), 2006, 63(3): 473-491.

[13] 韩亚娟, 彭运芳, 魏航, 等. 超启发式遗传算法求解带软时间窗的车辆路径问题[J]. 计算机集成制造系统, 2019, 25(10): 2571-2579.
Han Yajuan, Peng Yunfang, Wei hang, et al. Hyper-heuristic Genetic Algorithm for Vehicle Routing Problem with Soft time Windows[J]. Computer Integrated Manufacturing Systems, 2019, 25(10): 2571-2579.

[14] Taillard E. Benchmarks for Basic Scheduling Problems[J]. European Journal of Operational Research (S0377-2217), 1993, 64(2): 278-285.

[15] Reeves C R. A Genetic Algorithm for Flowshop Sequencing[J]. Computers and Operations Research (S0305-0548), 1995, 22(1): 5-13.

[16] Jarboui B, Jarboui B, Eddaly M, et al. A Hybrid Genetic Algorithm for Solving No-wait Flowshop Scheduling Problems[J]. The International Journal of Advanced Manufacturing Technology (S0268-3768), 2011, 54(9/12): 1129-1143.

[17] Ding J, Song S, Gupta J N D, et al. An Improved Iterated Greedy Algorithm with a Tabu-based Reconstruction Strategy for the No-wait Flowshop Scheduling Problem[J]. Applied Soft Computing (S1568-4946), 2015, 30: 604-613.

[18] 牛腾飞, 李江华, 堵国成, 等. 微生物法合成N-乙酰氨基葡萄糖及其衍生物的研究进展[J]. 食品与发酵工业, 2019, 46(1): 274-279.
Niu Tengfei, Li Jianghua, Du Guocheng, et al. Research Progress on Microbiological Synthesis of N-acetylglucosamine and Its Derivatives[J]. Food and Fermentation Industries (S0253-990X), 2019, 46(1): 274-279.