文章编号:1006－5911(2013)03－0596－12

自适应编码蜂群算法求解连续批量统一模型

张永韡,汪　镭,吴启迪
(同济大学 电信学院,上海　201804)

摘　要:针对连续生产过程与间歇生产过程混合的生产调度问题,建立了统一优化模型。使用随机比例法对调度任务序列进行二次编码,并使用波动修正稳定设备生产效率。对最大完工时间评价准则进行修正,引入最小停机次数准则,细化解的评价层次。使用蜂群算法使种群搜索最优解,并通过逆向解码得到调度序列。将所提算法应用于化工企业烧碱生产过程,并与文献所给出的结果进行比较分析,证明了算法的有效性。

关键词:生产调度;混合调度;连续批量;蜂群算法;编码算法;评价准则
中图分类号:TP278　　文献标志码:A

Artificial bees colony scheduling algorithm based on adaptive encoding method for unified batch/continuous models

ZHANG Yong-wei, *WANG Lei*, *WU Qi-di*
(College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China)

Abstract: A unified optimization model was proposed for mixed batch/continuous process scheduling problems. The Two Stage Random Ratio method was introduced to encode the job sequence, and the production velocity was stabilized by fluctuation adjustment. The makespan criterion was amended by introducing minimum machine halt criterion to refine the criteria system of solutions. The Artificial Bees Colony algorithm was used to search the optimum solution and scheduling sequence was acquired by reverse decoding. The proposed algorithm was applied in a caustic soda production process of chemical industry, and compared with the results in literatures, which confirmed the feasibility of proposed algorithm.

Key words: production scheduling; mixed scheduling; continuous/batch; artificial bees colony algorithm; encode algorithm; evaluation criterion

0　引言

混合流程生产系统是现代化生产中具有普遍意义的加工方式,它包括连续与间歇两类生产过程,常见于石油、化工、冶金及医药等行业。近些年来,混合流程生产调度得到了研究者的广泛关注,混合流程生产过程建模与调度问题的求解是目前研究的热点与难点。Ierapetritou 等[1]提出一种基于连续时间表述的模型,用来描述同时具有批量和连续过程的生产系统。这种建模主要有两个问题[2]:①生成的问题规模过大,随着事件点的增加,0－1 变量和约束均会增加;②这种结构不利于控制完成生产任务所需的设备批次,由此产生的调度计划批次数过多,实用性不强。Castro[3]等使用一种基于资源—任务网络(Resource-Task Network,RTN)表示的通用数学方法描述多功能混合过程。该方法基于统

收稿日期:2012-07-29;修订日期:2012-10-15。Received 29 July 2012; accepted 15 Oct. 2012.
基金项目:高等学校博士点基金资助项目(20100072110038);国家自然科学基金资助项目(70871091,61075064,61034004,61005090);教育部新世纪优秀人才支持计划资助项目(NECT-10-0633)。**Foundation items**: Project supported by the Research Fund for the Doctoral Program of Higher Education, China (No. 20100072110038), the National Natural Science Foundation, China (No. 70871091, 61075064, 61034004, 61005090), and the Program for New Century Excellent Talents in University, China (No. NECT-10-0633).

一时间间隔和连续时间表示,可得到混合整数线性规划(Mixed Integer Linear Programming,MILP)模型。但该模型无法对定长时间区间内的事件数量进行建模。徐智[4]等提出带缓冲区的混合生产的一种调度模型,将离散生产所需的半成品原料的生产分解为连续生产过程各生产线的分段式生产任务,并给出快速调度方法,再利用遗传算法和分派规则求解离散生产调度问题。K. J. Shaw[5]等以一个制糖企业为例,面向混合批量/连续生产过程,设计了一种以启发式遗传算法为核心的单目标/多目标调度算法;Tang[6]等以炼钢连铸过程为背景,建立了半连续/批量调度模型,并使用分散搜索算法进行求解;廖伟志[7]等针对 MILP 方法在解决混合生产过程调度中存在的问题,提出一种混合间歇/连续生产过程的时间约束混杂 Petri 网模型。但是基于混杂 Petri 网建立的调度问题模型过于繁杂,计算效率难以满足实际调度需求,制约了其应用。王海燕[8]等提出改进差分进化算法求解化工间歇与连续混合生产过程调度问题,该方法在寻优性与稳定性上有很大的提高,但随着问题规模的增大,算法性能与收敛速度也随着下降。

由于混合生产过程的复杂性,现有建模技术与求解算法各有优缺点。目前主流的建模方法是分别考虑连续过程与批量过程,以流速作为连续过程的决策变量,以批量作为批量过程的决策变量。这种将连续与批量过程分别进行讨论的方法在建立模型、仿真以及求解中需要分别建立不同的处理进程,带来了不必要的麻烦。为简化模型,本文将连续过程看作处理时间为调度间隔的批量过程,使用广义批量作为唯一的决策变量,建立统一的批量/连续生产过程模型。在该模型的基础上,采用人工蜂群(Artificial Bee Colony,ABC)算法[9]对调度问题进行求解。相比于遗传算法(Genetic Algorithm,GA)与差分进化算法(Differential Evolution,DE),蜂群算法结构新颖,本质上决定了求解过程不会限制在局部区域太久,具有更强的全局搜索能力;而且蜂群算法所需的外部参数少,不需要变异概率与交叉概率等参数。ABC 算法的这些特点激起了研究者广泛的研究兴趣,并已成功使用在求解复杂优化的问题中[10-13]。

在调度计划求解过程中,编码与解码方案的设计是首先要解决的问题。同一种算法框架因为编码与解码方案不同,其性能可能有很大区别。典型的分块编码方案[8](如图 1 左)有如下缺陷:①段内变异可能造成新解不合法或者不可行;②整段变异造成过多染色体信息丢失;③段间交叉潜在地增加了染色体长度。针对上述问题,本研究引入随机比例法进行编码,在机理上保证了变异与交叉后解的可行性。同时,由于编码长度也作为编码信息的一部分随种群演化,编码长度可以在算法求解过程中进行自适应调整,从而更好地控制编码的长度。

此外,大多数流程工业调度研究[3-4, 8, 14-15]仅以时间作为唯一的评价准则。为了在编码长度和调度精度之间进行取舍,往往选择较大的时间粒度,使得大量不同的调度方案按照时间粒度的等级划分为同一档次。从应用的观点来看,相同完工时间的调度计划并没有本质的不同。但是以进化计算的观点看,较粗的评价体系不但不利于甄别种群中优秀的个体,而且制约了种群多样性,进而影响算法收敛。为克服这种困难,通常在最大完工时间的基础上添加辅助评价策略。辅助评价策略的选择与待解问题相关,通常在相同完工时间的情况下,使用改善潜力、附加效应、系统压力或者实施的难易程度等对调度计划做进一步评价,如最小活动准则[2]。本研究将在最小完工时间的基础上,通过最小停机次数准则细化评价体系来提高算法性能。

综上,本文在建立批量/连续统一模型的基础上,使用随机比例法和 Random Key 方法对任务序列进行二次映射,使用改进的基于最大完工时间的最小停机评价准则,在人工蜂群算法的框架下对化工过程调度问题进行求解。实验显示,所述改进相对于传统编码方案以及 GA 和 DE 算法,可以显著缩短最大完工时间,并减小设备停机次数。

1　问题描述

传统的时间离散化方案将调度时间范围划分为等长的区间,每个区间长度为 Δt。由于需要区分连续与批量过程生产工序,并采用不同的处理方式,这种操作给后续的编码与解码造成了麻烦。

为简化数学模型与计算机仿真,本文提出批量/连续过程的统一离散化模型。规定更改生产速度、启停设备等事件只能发生在区间边界。对于连续生产工序,区间内的流速不变,区间的产量等于流速乘以区间长度。对于给定的区间长度 Δt,将处理流速为 VF 的连续过程看作处理量为 $VF \cdot \Delta t$ 的批量过程。

1.1 术语表

为方便讨论，下面给出模型中所使用符号的含义：

N 为产品数；

AM_i 为表示产品 i 的需求量，$i \in \{1,\cdots,N\}$；

H 为生产线上包含的设备数，包括连续生产过程加工设备和间歇生产过程加工设备；

EVL_i 为设备 i 在 Δt 时间内的最小处理量（连续设备）或者最小容量（间歇设备），$i \in \{1,\cdots,H\}$；

EVU_i 为设备 i 在 Δt 时间内的最大处理量（连续设备）或者最大容量（间歇设备），$i \in \{1,\cdots,H\}$；

ET_i 为设备 i 的加工时间，$i \in \{1,\cdots,H\}$，假设间歇生产设备的生产时间与设备上安排的生产批量无关，对于连续生产设备，加工时间为调度时间区间长度 Δt；

E_i 为设备空闲标志，当设备空闲时为 1，否则为 0，$i \in \{1,\cdots,H\}$；

EM_i 为设备生产时间系数，$i \in \{1,\cdots,H\}$，对于连续生产设备取值为 Δt，对于批量生产设备则取值为 1。

M 为生产线上的工序数，包括连续生产过程工序和间歇生产过程工序，为便于排产操作，将生产线上的工序从末端到始端进行排序标号；

OML_i 为工序 i 在一个时间区间内的最小处理量，$i \in \{1,\cdots,M\}$；

OMU_i 为工序 i 在一个时间区间内的最大处理量，$i \in \{1,\cdots,M\}$；

IMT_i 为工序 i 的输入物料集，$i \in \{1,\cdots,M\}$；

OMT_i 为工序 i 的输出物料集，$i \in \{1,\cdots,M\}$；

OP_i 为工序 i 的类型，$OP_i = 1$ 表示工序 i 为连续生产过程，$OP_i = 0$ 表示工序 i 为间歇生产过程；

OE_i 为工序的加工设备集，$i \in \{1,\cdots,M\}$；

n_i 为产品 i 包含的工序数，$n_i \leqslant M$，$i \in \{1,\cdots,N\}$；

L 为生产线上包含的物料数，包括原料、中间品和产品；

IOR_{ij} 为工序 i 加工时物料 j 的输入（$j \in IMT_i$）/产出比（$j \in OMT_i$），$i \in \{1,\cdots,M\}$，$j \in \{1,\cdots,L\}$；

DMT_i^t 为 t 时刻物料 i 的储量，$i \in \{1,\cdots,L\}$；

UMO_i 为需要消耗的物料 i 的工序集，$i \in \{1,\cdots,L\}$；

CMO_i 为生成物料 i 的工序集，$i \in \{1,\cdots,L\}$；

F 为生产线上包含的贮槽数，假设生产线上紧邻的工序之间都存在贮槽，且每个贮槽只能存储一种物料；

DV_i 为贮槽 i 的容量，$i \in \{1,\cdots,F\}$；

DIN_i^t 为时刻 t 贮槽 i 的储量，$i \in \{1,\cdots,F\}$。

决策变量为广义设备批量 CN_i^j（对于连续生产设备，$CN_i^j = VF_i^j \cdot \Delta t$ 表示设备 i 上 $[k\times\Delta t,(k+1)\times\Delta t)]$ 时段内的处理量；对于批量生产设备，表示设备 i 上第 j 个批次的处理量），通用生产批次数 b_i（对于批量过程，b_i 表示为了完成所有的调度任务，设备 i 上安排的任务批次数目；对于连续过程，b_i 表示设备非空闲的时间区间 Δt 的总数），开始时间 TBN_i^j 和结束时间 TEN_i^j，$j \in \{1,\cdots,b_i\}$。

1.2 数学模型

(1)设备能力约束

$$EVL_i \leqslant CN_i^k \leqslant EVU_i,$$
$$i \in \{1,\cdots,H\},\ k \in \{k \mid 1 < k < b_i\}。\quad (1)$$

(2)物料平衡

$$DMT_i^{t+1} = DMT_i^t - \sum_{j\in UMO_i}\sum_{k\in OE_j}\sum_{\substack{TBN_k^p = t\\ p=1,\cdots,b_k}}(CN_k^p \cdot IOR_{ji}) + \sum_{j\in CMO_i}\sum_{k\in OE_j}\sum_{\substack{TBN_k^p = t\\ p=1,\cdots,b_k}}(CN_k^p \cdot IOR_{ji})。\quad (2)$$

式中右侧第二部分表示在 t 时刻所有工序所有设备上将要开始的任务对物料 i 的消耗总量。对于单一原料的生产设备，相应的 $IOR_{ij} = 1$，表明批量 CN_k^p 全部由设备 k 在第 p 个批次完成。

(3)供求约束

$$DMT_i^t \geqslant CN_k^p \cdot IOR_{ji},\ i \in \{1,\cdots,L\},$$
$$j \in UMD_i, k \in OE_j, p = 1,\cdots,b_k。\quad (3)$$

对于连续和间歇设备，如果物料 i 的储量不足以供应设备 k 以规定的流速或者批量生产，则设备 k 不安排生产。

(4)存储约束

$$0 \leqslant DIN_i^t \leqslant DV_i,\ i \in \{1,\cdots,F\},\ t \geqslant 0。\quad (4)$$

若 t 时刻储罐 i 紧邻的前置工序产生物料 i 的总量，加上 t 时刻的储罐储量再减去 t 时刻储罐 i 紧邻的所有后续工序对物料 i 的最大处理量总和后，物料 i 的数量仍然大于储罐的存储能力，则 t 时刻相应的任务不予排产。

(5)任务分配约束

任何设备必须在完成当前批量后才能加工新的批量。对于连续生产设备，因为每次加工的时间长

度只有 Δt,在每个时间点必定已经完成当前批量的加工,所以该约束自然满足。对于间歇生产设备,$\forall k \in \{1,\cdots,b_i\}$, $j \in \{j \mid k \leqslant j \leqslant b_i\}$,均满足 $TEN_i^j \leqslant TBN_i^k$,对于 $\forall j \in \{1,\cdots,b_i\}$ 均满足 $TBN_i^j \leqslant TEN_i^j$。

(6)生产计划约束

$$\underset{t\to\infty}{DIN_i^t} \geqslant AM_i, i \in \{1,\cdots,N\}。 \tag{5}$$

当生产结束时,相应的存储罐中每种产品的储量应该满足该产品的需求量。

(7)目标函数

以最小化最大完工时间为目标,目标函数为

$$\min T_{\max} = \max_{i\in\{1,\cdots,H\}} (TEN_i^{b_i})。 \tag{6}$$

2　编码,解码与评价准则

2.1　编码

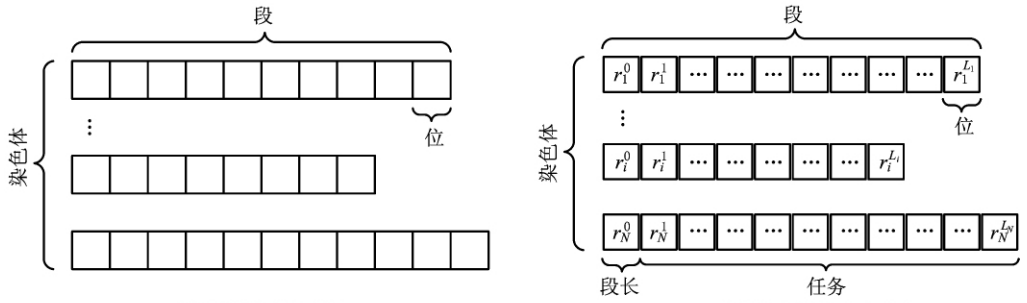
为便于下文论述,先给出一些术语的定义。

(1)位　编码序列中最小的单位,代表一台机器在特定时间段内处理的任务量。一个位代表一个任务。

(2)段　由一系列任务(位)组成的串称为段,一个段代表一个工序队列。

(3)染色体　由一系列段组成的编码集合称为染色体。染色体可以由段首尾相接组成,并行排列的染色体结构如图 1a 所示。段的长度各不相同,组成的染色体外形也不规则,其形态类似一把梳子,一个染色体代表一个完整的调度计划。符号 G_{ij}^k 或者 $G_{i,j}^k$ 表示第 k 个染色体第 i 段中的第 j 位,为了简化表示,下文以 $G^k = [G_1^k,\cdots,G_M^k]$ 表示第 k 个染色体整体,$G_i = [G_{i,1},\cdots,G_{i,L_i}]$ 表示某个染色体内的第 i 段整体。位、段和染色体的组成关系如图 1 所示。

这里使用染色体这个术语,仅仅为了形象地表示编码,实际算法结构与遗传算法无关。在下文中,如未加说明,“食物源”、“解”与“染色体”均表示与图 1 一致的编码结构。



a 传统编码染色体结构　　b 随机比例法染色体结构

图1　传统编码与随机比例法染色体结构

为了确定每台设备上的任务量,首先需要每个工序处理的任务总量。因此,根据产品需求量 AM,从最后一个工序开始,逆序计算出每道工序计划处理的物料总量 PL_i, $i \in \{1,\cdots,M\}$,使其在可行解范围内满足式(5),然后随机生成满足下式的任务队列 $G_i = [G_{i,1},G_{i,2},\cdots,G_{i,L_i}]$:

$$PL_i = \sum_{j=1}^{L_i} G_{ij},\ i \in \{1,\cdots,M\}; \tag{7}$$

$$OML_i \leqslant G_{ij} \leqslant OMU_i,$$

$$i \in \{1,\cdots,M\}, j \in \{1,\cdots,L_i\}; \tag{8}$$

$$OML_i = \begin{cases} \min\limits_{j\in OE_i} EVL_j \cdot \Delta t, & OP_i = 1; \\ \min\limits_{j\in OE_i} EVL_j, & \text{其他}。 \end{cases} \tag{9}$$

$$OMU_i = \begin{cases} \sum\limits_{j\in OE_i} EVU_j \cdot \Delta t, & OP_i = 1; \\ \sum\limits_{j\in OE_i} EVU_j, & \text{其他}。 \end{cases} \tag{10}$$

使用这种编码方式,染色体的每一位代表了任务量的具体数值,在任何算法的交叉或者变异阶段,一个任务的变异可能导致整个任务串的任务总量不满足计划约束。文献[8]通过在染色体之间交换段和整段变异的方法解决该问题,导致一个段内的任务序列在交叉时无法得到改变,而任务序列变异也只是整段的随机初始化,无法利用其他染色体中相应段内的信息;此外,在对两个染色体内实施段间进化操作时,对于不同长度的段,采用在短的段后面添加 0 使二者等长的方法,以使种群在进化过程中,染色体内的段长度有变长的趋势,而每一个段可能的

最小长度在初始化的阶段就已经确定，进化过程中无法使段变得更短。由于任务序列的长短在很大程度上决定了完工时间的长短，整段进行操作无疑降低了算法找到更短完工时间解的能力。

2.1.1 随机比例法

为了解决上述问题，本文引入随机比例原则对任务量进行二次编码，方法如下：

对于工序 i，首先确定在该工序的最大和最小生产能力范围内，任务串可能出现的最长和最短值，记为 $L_i^{\max}$ 和 $L_i^{\min}$，然后在 $[L_i^{\min}, L_i^{\max}]$ 之间生成表示任务串长度(染色体段长度)的随机整数 L_i，并通过下式转化为[0,1]之间的小数 r_{i0}，

$$r_i^0 = \frac{L_i - L_i^{\min}}{L_i^{\max} - L_i^{\min}}, i \in \{1, \cdots, M\}。 \quad (11)$$

生成长度为 L_i 的取值在[0,1]的均匀分布随机数序列 $r_i = [r_i^1, \cdots, r_i^{L_i}]$，得到工序 i 的染色体 $R_i = [r_i^0, r_i^1, \cdots, r_i^{L_i}] = [R_{i1}, R_{i2}, \cdots, R_{i,L_i+1}]$，然后根据 $1 \sim L_i$ 位随机数占该随机数序列之和的比例确定每一个任务生产量占总计划量的比例，以此得出每个任务具体的生产量

$$G_{ij} = \frac{r_i^j PL_i}{\sum_{k=1}^{L_i} r_i^k},$$

$$i \in \{1, \cdots, M\},\ j \in \{1, \cdots, L_i\}。 \quad (12)$$

染色体中的每个块由序列长度标志位和序列组成，其结构如图 1b 所示。

随机比例法得到的染色体中的每一位均为[0,1]之间的数字，可以任意进行按位的交叉变异等操作，而不会影响解的合法性。

2.1.2 基于 Random Key 的任务编码

随机比例在分配任务时通常按照随机序列的顺序进行串行处理，对于一个长度为 L_i 的随机序列，在确定了前 $L_i - 1$ 个任务后，第 L_i 个任务自然确定，即 $G_{i,L_i} = PL_i - \sum_{k \in [1, L_i - 1]} G_{i,k}$。由于随机序列的不确定性，任务的最后一位往往过大或过小。为了改善这种情况，引入 Random Key[16] 对任务分配顺序进行随机排列。Random Key 编码对随机数序列进行排序，得到不重复的自然数序列，由在 2.1.1 节得到的该序列得到任务分配的顺序。对于随机序列 x，Random Key 得到将 x 按升序排列后的序号 β，最终的任务分配序列由下式决定：

$$RK_{\beta_i} = i,\ i \in \{1, \cdots, L_i\}。 \quad (13)$$

式(13)表明任务序列的第 β_i 位将分配第 i 个任务。例如随机序列 $x = [0.4, 0.8, 0.5, 0.6, 0.3]$，排序后得到 $\beta = [5,1,3,4,2]$，$RK = [2,5,3,4,1]$，对于段 G_i，其中的基因位将按照 $G_{i,2}$，$G_{i,5}$，$G_{i,3}$，$G_{i,4}$，$G_{i,1}$ 的顺序进行分配。随机序列 R_i 的结构以及通过 Random Key 编码将 R_i 映射为任务序列的示意如图 2 所示。

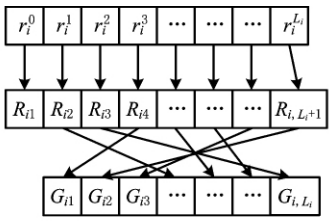


图2 随机序列到任务序列的二次映射和Random Key编码

2.1.3 动态上下限

使用随机比例得到的二次编码虽然增强了解的可行性，但是在顺序地将随机码转换为任务量的过程中，任务量的大小受到工序设备集生产能力的限制，并不能完全按照随机数序列的比例分配计划任务。一种简单的修正方法是当任务量小于设备生产能力下限时，将任务量置为设备生产能力下限，当任务量高于设备生产能力的上限时将其置为上限值。而这又会导致新的问题，当一个任务违反设备能力约束并进行修正时，该修正将导致已计算出的任务总量与计划任务总量不一致。为解决该问题，本文在随机码转换为任务量的过程中，根据已经转换的任务量和设备上下限，动态确定当前任务量的上下限：

$$OMU_i^D = \min(PL_i - OML_i \times (Nl - 1), OMU_i); \quad (14)$$

$$OML_i^D = \max(PL_i - OMU_i \times (Nl - 1), OML_i)。 \quad (15)$$

式中 Nl 是当前任务编码序列中未分配任务的编码数量。式(14)表示当前任务的(动态)上限为假定后续的 $NL - 1$ 任务均以设备能力下限分配的情况下所允许的最大任务量，与该工序设备能力上限之间的较小值；式(15)表示当前任务的(动态)下限为假定 $Nl - 1$ 个后续任务均以设备能力上限分配的情况下所允许的最小任务量，与该工序最小设备容量两者之间的较大值。

当 $OMU_i^D < OML_i^D$ 时，表明动态上下限已无法完成对该随机序列的可行编码，出现这种情况的两种可能及处理方法如下：

(1) $OMU_i^D < OML_i$,表明后续任务即使均以设备能力的下限分配,所剩余的任务量仍然低于设备处理能力的下限。当然,这种情况在实际中不可能出现,因为设备处理能力下限通常为 0。但是,当设备能力下限不为 0 时,出现这种情况后的处理方法为,将当前任务至第 $Nl-1$ 个未分配的任务均以设备处理能力的下限分配。

(2) $OML_i^D > OMU_i$,表明后续任务即使均以设备能力的上限分配,所剩余的任务量仍然高于设备处理能力的上限。同样,这种极端的情况在实际中也不太可能出现。但是,如果出现,则将当前任务至第 $Nl-1$ 个未分配的任务均以设备最大能力分配。

如果上述情况均没有出现,则按照当前任务的随机数与剩余任务随机序列和的比例分配剩余的计划任务。在每个任务分配结束后,将该分配量从总计划量中减去,然后开始下一个任务的分配,直到 L_i-1 个任务分配完毕,最后剩余的未分配计划量就是第 L_i 个任务量。

算法 1　动态上下限法。

步骤 1　由随机比例法得到 R_i,计算 R_i 中第 2 项至 L_i+1 项的和 S,计算 Random Key 序列 RK,设置未分配任务的编码数量 $Nl = L_i$,设置任务序号 $j = 1$。

步骤 2　根据式(14)和式(15)计算当前动态上下限 OML_i^D 和 OMU_i^D。

步骤 3　如果 $OMU_i^D < OML_i^D$,则进入步骤 4,否则执行步骤 6。

步骤 4　如果 $OMU_i^D < OML_i$,则第 j 至 L_i-1 任务量均为 OML_i,剩余计划处理量 $PL_i = PL_i - OML_i \cdot (NL_i - 1)$,执行步骤 8,否则执行步骤 5。

步骤 5　如果 $OML_i^D > OMU_i$,则第 j 至 L_i-1 任务量均为 OMU_i,$PL_i = PL_i - OMU_i \cdot (NL_i - 1)$,执行步骤 8,否则执行步骤 6。

步骤 6　按照 RK 中规定的顺序,按比例在剩余计划量中分配第 j 个任务量。如果所得任务量超出动态上下限约束,则将任务量设为约束值。更新 S、剩余计划处理量 PL_i 和未分配编码数量 Nl。

步骤 7　若 $j < L_i+1$,则 $j = j+1$,返回步骤 2,否则执行步骤 8。

步骤 8　最后一个任务量等于剩余计划处理量。

2.1.4　比例波动修正

定义 1　对于均匀分布随机变量 $x \sim U(R_{\min}, R_{\max}) \geqslant 0$,其波动率定义为

$$d = \frac{R_{\max} - R_{\min}}{R_{\max}}。\tag{16}$$

当生成[0,1]之间的均匀分布随机数,计算每个随机数占总体的比例,继而确定生产量时,由于波动量较大,任务块之间的比例会有很大差异,如 0.01 和 0.99 之间的差距达到了 99 倍。显然,按此生成的生产量变化幅度较大,不利于生产。为了减小波动率,可以将均匀分布向右侧进行偏移:

$$U(R_{\min}, R_{\max}) \rightarrow U(R_{\min}+C, R_{\max}+C)。\tag{17}$$

式中 C 是由原均匀分布上下限和期望波动率 p 决定的正实数,

$$C = \frac{R_{\max}(1-p) - R_{\min}}{p}。\tag{18}$$

例如,希望 0～1 之间的随机数波动量小于等于 0.2,则有 $C \geqslant 4$。当增加 C 后,随机数变化范围为[4,5],最大波动量为(5−4)/5=0.2。

2.1.5　设备能力下限修正

在编码阶段,如果按照实际设备生产能力的下限产生任务量,则在初始化阶段将产生大量小任务量。单个任务量的减小势必增加编码长度和增长任务序列,从而增加计算复杂度和完工时间。同时,由于很多群体智能算法的性能随问题规模的增大而急剧下降,也使得小任务量的编码方式失去实用价值。在某些实现中,通过在编码阶段使用较大的设备处理能力下限的方法(通常取为某工序设备集中容量最小者的一定比例)增加每一个任务量的数值,从而减少序列长度:

$$OML_i = \max\left(\min_{k \in OE_i} EVL_k, \alpha \min_{k \in OE_i} EVU_k\right)_{EM_j}, \quad j \in \forall OE_i。\tag{19}$$

式中:$j \in \forall OE_i$ 表示 j 为设备集 OE_i 中的任意一个设备序号,$\alpha \in [0,1]$ 为设备能力系数。当 $\alpha = 1$ 时,显然生成的任务值均为并行设备能力上限的最小值。为了最大限度地降低任务序列长度,通常将 α 取为较大的值,如 0.8。这样虽然缩短了任务队列,但是解空间也被人为缩小,甚至一些包含最优解的空间也被排除在外,使解的评价降低。为此,在第 4 章将详细研究设备能力下限对算法的影响。

2.2　解码

由于采用了二次编码方案,解码时也需要通过两次映射才能得到调度表。首先,检查当前解码的任务量 G_{ij} 是否满足如下约束(从解码的角度,下列

约束条件表示为与任务量相关的形式)：

(1)设备能力约束

$$\sum_{j\in OE_i} E_j \cdot EVL_j \cdot EM_j \leqslant G_{ij} \leqslant \sum_{j\in OE_i} E_j \cdot EVU_j \cdot EM_j \text{ 。} \tag{20}$$

式(20)是式(1)一种动态的表达，表示当前工序可用设备的生产能力可以对任务量 G_{ij} 进行排产。

(2)供求约束

$$DMT_j^t \geqslant G_{ij} \cdot IOR_{ij} \text{ 。} \tag{21}$$

如果工序 i 所需的物料储量中的任一种不满足当前任务量，则该任务量不排产。

(3)存储约束

$$DIN_j^t + G_{ij} \cdot IOR_{ij} \leqslant DV_j,\ j \in OMT_i \text{ 。} \tag{22}$$

如果 G_{ij} 的输出物料中有任何一种破坏存储约束，则该任务量不排产。

当上述约束条件均满足时，对任务 G_{ij} 进行排产。物料平衡约束在计算物料变化时自然地满足，任务分配约束在进行设备能力约束时通过检查空闲设备隐含地满足，生产计划约束在编码阶段由倒推得出的各工序生产计划 PL_i 自然地满足。

在约束条件均满足的情况下，由时刻 0 开始，首先分配[0，Δt]时段内的任务。按照从左至右的顺序逐个检查序列中是否有满足各项约束的任务，如果有，则安排设备排产，刷新设备状态，更新释放时间以及存储设备状态，然后检查下一个工序的任务序列；如果任务序列中没有符合条件的任务，则当前时间段工序 i 不排产。当所有工序都检查一遍后，在时段[Δt，$2\Delta t$]重复上述步骤，直到 G^k 上的所有任务都安排生产。

如果某时段内工序 i 的任务队列中一项任务 G_{ij} 有多台设备可用，则涉及到并行设备选择问题。本文使用最小总容量原则，即在可用设备的所有可能组合中，选择能够完成任务设备集合的最小总容量组合，且按照每台选中设备与所有选中设备最大容量和的比例分配任务。

2.3 评价

对于传统的调度算法，最小完工时间是最直观和最简便的评价方案。但是这种方案评价体系非常粗糙，不利于在进化过程中指导算法演化。例如，对于 Δt=0.5 h 且最短完工时间大约为 10 h 的生产过程而言，大多数可行解的最大完工时间大致位于[8，15]之间，而在该区间，完工时间只有[8，8.5，…，15]共 15 种取值。显然，这样的评价体系并不适合在巨量的可行解中区别优劣。因此，本文在最大完工时间 $T_{\max}$ 的基础上引入第二个评价指标——最小停机次数，并使用新的评价指标

$$object = \min\{T_{\max} + n_{\text{stop}}/100\} \text{ 。} \tag{23}$$

式中 n_{stop} 为调度计划的停机次数。如果在调度计划中，某台机器在一次停机后，经过有限的时间间隔再次启动，则记一次停机次数。停机次数在甘特图上直观地表示为任务序列之间的间隙数目。由于调度的首要目标是最小化最大完工时间，如果一个解的完工时间较大，则不论停机次数多么少，这个解也不能优于完工时间较小的解。式(23)中的除数项 100 就是为了保证相同完工时间的解，其评价位于同一个数量级，而同一等级的解则通过停机次数判断其优劣。

3 基于统一模型的蜂群算法求解方案

3.1 蜂群算法

人工蜂群算法由 Karaboga[9] 于 2005 年提出。该算法基于蜜蜂觅食行为，将蜂群中的个体划分为工蜂、观察蜂和侦查蜂三个种类。由工蜂探查食物源(解的邻域)，并由食物的数量(解的适应度)决定工蜂“跳舞”区域的大小；观察蜂依据工蜂跳舞区域的大小，按概率对食物源进行再次探查；当食物源耗尽时，侦查蜂负责在可行空间中确定新的食物源。其基本算法流程可参见文献[9]。

3.2 调度算法流程

本文提出的调度算法主要分为初始化、编码、解码和进化操作几个模块，下面介绍每个模块的算法流程。

3.2.1 目标问题初始化

目标问题初始化步骤如下：

步骤 1 根据具体问题，给定调度时间间隔 Δt，N，EVL，EVU，AM，H，L，M 和 DMT 等参数。

步骤 2 将每种产品的需求量折算成每道工序的计划处理量 PL_i，$i = 1,\cdots,M$。

步骤 3 根据式(9)和式(10)计算每道工序的最小与最大处理量 OML_i 和 OMU_i。

步骤 4 计算染色体的最大与最小长度 $L_i^{\max} = |PL_i/OML_i|$，$L_i^{\min} = |PL_i/OMU_i|$。

3.2.2 编码算法

编码算法步骤如下：

步骤 1　设置期望波动率 p 以及偏移常数 C，令工序号 $i=1$。

步骤 2　在 $[L_i^{\min},L_i^{\max}]$ 之间随机选择工序 i 的编码长度 L_i，根据式(11)得到 r_i^0。

步骤 3　生成长度为 L_i 的随机数列 $r_i=[r_i^1,r_i^2,\cdots,r_i^{L_i}]\sim \mathrm{U}(0,1)+C$，最终编码段为 $R_i=[r_i^0,r_i^1,r_i^2,\cdots,r_i^{L_i}]$。

步骤 4　对 R_i 排序，其序号数列记为 β_i；根据式(13)计算 Random Key RK_i，$i\in\{1,\cdots,L_i\}$。

步骤 5　使用算法 1 确定第 i 道工序的染色体编码 G_i。

步骤 6　若 $i<M$，则 $i=i+1$，返回步骤 2；否则完成编码，退出。

3.2.3　解码算法

解码算法步骤如下：

步骤 1　令 $i=1$，$j=1$，$t=0$。

步骤 2　得到当前任务可用设备集，对于连续设备，认为总是可用。

步骤 3　检查 G_{ij} 是否满足约束式(20)～式(22)，若均满足，则转步骤 5，进行排产，否则进入步骤 4。

步骤 4　判断段内的未分配任务是否均检查过，若是，则转步骤 7；否则 $j=j+1$，转步骤 3。

步骤 5　选择满足 $G_{i,j}$ 的最小总容量设备集，按照设备容量比例排产。

步骤 6　更新通用设备批量 CN_i^j、任务开始时间 TBN_i^j 和任务结束时间 TEN_i^j，刷新设备状态。

步骤 7　若 $i<M$，则 $i=i+1$，返回步骤 2，否则执行步骤 8。

步骤 8　若 G 内所有任务没有全部解码完毕，则刷新当前物料平衡状态，$t=t+1$，返回步骤 1；否则解码完成，退出。

3.2.4　进化操作

蜂群算法涉及的进化操作包括指派工蜂、观察蜂和侦查蜂三类，流程分别如下：

(1)工蜂阶段

步骤 1　初始化工蜂序号 $i=1$、设置工蜂数量 N_e，令食物源访问次数 $v_i=0$。

步骤 2　随机生成 $k\in\{1,N_e\}\neq i$ 以及 $j\in\{1,\cdots,N\}$，使第 k 和第 i 个食物源的第 j 个工序 R_j^k 与 R_j^i 进行交叉，得到 Rn_j^i：

①记两个工序串的最大长度为 Rl，较短的工序串后面补 0，使其长度变为 Rl。

②对两个编码段中的每一位进行交叉，得到新食物源的第 j 个编码段 Rn_j^i。

③新工序段的编码长度 $L_j=L_j^{\min}+|(L_j^{\max}-L_j^{\min}+1)\times Rn_{j1}^i|$。

④若 $L_j>Rl-1$，则在 Rn_j^i 后补上长度为 L_j-Rl+1 的随机序列 $r_i\sim \mathrm{U}(0,1)+C$；若 $L_j<Rl-1$，则只取 Rn_j^i 的前 L_j+1 位。

步骤 3　对 Rn^i 进行解码，计算适应度，如果适应度更大，则更新食物源，$v_i=0$，否则 $v_i=v_i+1$。

步骤 4　若 $i<N_e$，则令 $i=i+1$，返回步骤 2，否则退出工蜂阶段。

(2)观察蜂阶段

步骤 1　初始化观察蜂序号 $i=1$，设置观察蜂数量 N_o。

步骤 2　按照轮盘赌方法选择食物源。假设选择第 j 个食物源，随机生成 $k\in\{1,N_e\}\neq i$ 以及 $m\in\{1,\cdots,N\}$，按照工蜂阶段的步骤 2①～步骤 2④使第 k 和第 j 个食物源的第 m 个序列 R_m^k 与 R_m^j 进行交叉，得到 Rn_m^j。

步骤 3　对 Rn^j 进行解码，计算适应度，如果适应度更大，则更新食物源，$v_i=0$，否则 $v_i=v_i+1$。

步骤 4　若 $i<N_o$，则 $i=i+1$，返回步骤 3，否则退出观察蜂阶段。

(3)侦查蜂阶段

步骤 1　初始化食物源序号 $i=1$。

步骤 2　检查第 i 个食物源的访问次数，如果 $v_i\geqslant v_{\max}$，则按照 3.2.2 节编码算法生成随机长度随机序列 R^i，作为新食物源替换原食物源，否则转步骤 4。

步骤 3　对 R^i 进行解码，得到调度计划，并计算适应度，令 $v_i=0$。

步骤 4　若 $i<N_e$（食物源数量与工蜂数量相等），则令 $i=i+1$，返回步骤 2，否则退出侦查蜂阶段。

3.2.5　算法总体流程

算法总体流程如图 3 所示，每个流程后标号内的编号表示流程在文中所在的章节。

4　仿真实验

以某化工企业隔膜烧碱车间的生产过程为例进行仿真实验，问题取自文献[8]，其工艺流程如图 4 所示，其中圆柱体为存储槽，方块为工序，圆圈表示产品，工序后的数字表示该工序产品的产率。蜂群算法参数如下：进化代数为 50，蜂群规模为 80，其中工蜂 40，观察蜂 40，侦查蜂数量不计入种群规模。食

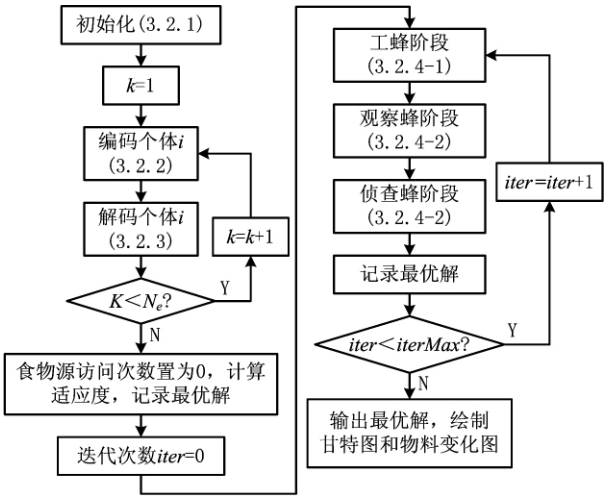


图3 算法总体流程

物源访问次数上限为 15，表明任何一个食物源被工蜂或者观察蜂访问 15 次后，不论其适应度在种群中如何，都将被蜂群遗忘，并由侦查蜂在可行解空间生成新的食物源。

产品需求量、设备—工序情况、储槽情况及其他实验参数参考文献[8]，其中时间段长度 $t=2$ h。为了验证改进策略对算法性能的影响，本文分别将不使用任何改进策略的基本蜂群算法(ABC1)、改进评价准则的蜂群算法(ABC2)、使用随机比例法编码的蜂群算法(ABC3)以及同时应用改进评价和编码策略的蜂群算法(ABC4)应用于上述问题，并与文献[8]中的遗传算法(GA)、差分进化(DE)以及改进 DE 算法进行比较，因为使用评价函数不同，所以不采用原文献中的适应度数值进行比较，而是换算为完工时间。同时，为了研究解空间大小对搜索算法的影响，在设备能力系数 α 分别取值为 0.2，0.5，0.8 时，对四种不同组合的 ABC 算法运行 10 次，运行结果、均值和标准差如表 1 所示，其中完工时间后的括号内为总停机次数。算法名称采用如下命名规则：算法结构(设备能力系数)。

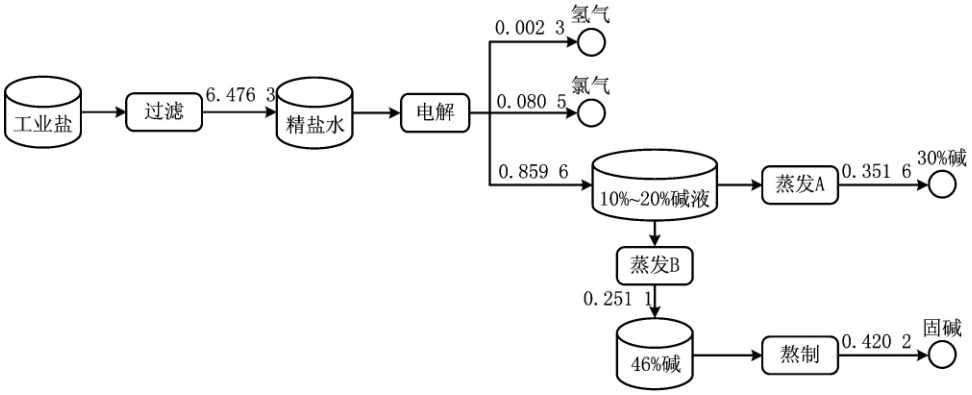


图4 烧碱工艺流程图

表 1 6 种算法 10 次运行结果比较

算法	最大完工时间											
	1	2	3	4	5	6	7	8	9	10	平均值	标准值
GA(0.8)	46	36	36	34	36	36	38	36	38	36	37.2	3.124 1
DE(0.8)	34	46	34	36	48	48	36	34	34	36	38.6	5.800 0
改进 DE(0.8)	34	34	36	36	36	34	34	48	34	36	36.2	4.044 7
ABC1(0.8)	38(3)	38(2)	38(1)	38(4)	38(2)	38(3)	36(3)	38(3)	38(2)	38(4)	37.8(2.7)	0.60(0.90)
ABC2(0.8)	36(1)	36(1)	36(2)	36(1)	38(3)	38(2)	38(3)	38(1)	38(3)	38(2)	37.2(1.9)	0.98(0.83)
ABC3(0.8)	30(2)	30(3)	30(1)	30(4)	30(5)	30(1)	30(1)	30(2)	30(4)	30(1)	30.0(2.4)	0.00(1.43)
ABC4(0.8)	30(1)	30(1)	30(1)	30(1)	30(1)	30(1)	30(1)	30(1)	30(1)	30(1)	30.0(1.0)	0.00(0.00)

续表 1

ABC1(0.5)	38(5)	38(4)	40(5)	40(7)	38(5)	40(5)	36(3)	38(4)	38(4)	38(4)	38.4(4.6)	1.20(1.02)
ABC2(0.5)	38(4)	38(2)	40(1)	38(2)	40(2)	40(2)	40(3)	40(2)	38(2)	38(3)	39.0(2.3)	1.00(0.78)
ABC3(0.5)	30(2)	30(1)	30(2)	30(0)	30(1)	30(2)	30(2)	30(4)	30(1)	30(3)	30.0(1.8)	0.00(1.08)
ABC4(0.5)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30.0(0.0)	0.00(0.00)
ABC1(0.2)	42(6)	42(5)	42(4)	42(5)	42(7)	42(7)	42(6)	40(6)	40(3)	42(6)	41.6(5.5)	0.80(1.20)
ABC2(0.2)	40(2)	42(4)	42(2)	42(2)	40(3)	42(2)	40(5)	40(7)	42(4)	38(3)	40.8(3.4)	1.33(1.56)
ABC3(0.2)	30(2)	30(3)	30(3)	30(2)	30(2)	30(4)	30(5)	30(3)	30(0)	30(1)	30.0(2.5)	0.00(1.36)
ABC4(0.2)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30(0)	30.0(0.0)	0.00(0.00)

可以看出:①在不使用任何改进策略的情况下,ABC1 的表现略低于文献[8]中的改进 DE 算法,但是表现与基本 DE 算法持平。此外,10 次运行结果的标准差显示算法稳定性较高。由运行结果的停机次数来看,尽管每次的停机次数不尽相同,但是由于没有在评价时加以区分,使种群失去了进一步进化的能力,最终导致算法潜能无法完全发挥。②使用 ABC2 使 10 次运行的完工时间均值略为下降。③使用 ABC3 的最大完工时间为 30 h,相比原文中的改进 DE 算法缩短了 4 h,即两个调度周期。④同时采用改进评价和改进编码方案的 ABC4 在完工时间上与 ABC3 无区别,但是 10 次运行均找到了最小停机次数的调度计划。⑤对于未使用改进编码策略的 ABC1 和 ABC2 框架,随着 α 的降低,求解性能逐渐恶化。由于参与比较的 GA 与 DE 也使用缩小的解码空间($\alpha = 0.8$),显然其性能也会随着求解空间的增大而下降。从实验结果可以看出,对于使用改进编码策略的 ABC4 框架,性能没有随着解空间的增大而下降,同时停机次数降为 0。由此验证了本研究中自适应任务序列长度编码方案的有效性。

综上,改进评价策略单独作用下算法的性能提升较为有限,编码方案的改进对性能改善有较大的影响。使用蜂群算法求解,不论是否使用评价策略,数值稳定性均高于 GA,DE 或改进 DE,证明本文所提出的编码方案与搜索算法具有更好的求解能力。其中一次最大完工时间为 30 h 的无停机调度方案相应的机器甘特图、产品产量以及存储量变化分别如图 5 和图 6 所示。图 5 中的 $E_1 \sim E_8$ 分别表示原盐精制机组、电解槽、三效顺流蒸发机组、三效逆流蒸发机组#1 和 2#,以及熬碱大锅#1、#2 和#3。图 6a 的曲线 a～e 分别为氢气、氯气、30%碱液、46%碱液和固碱的产量;图 6b 的曲线 a～c 分别为原盐、电解液和 10～20%碱液的存储情况。

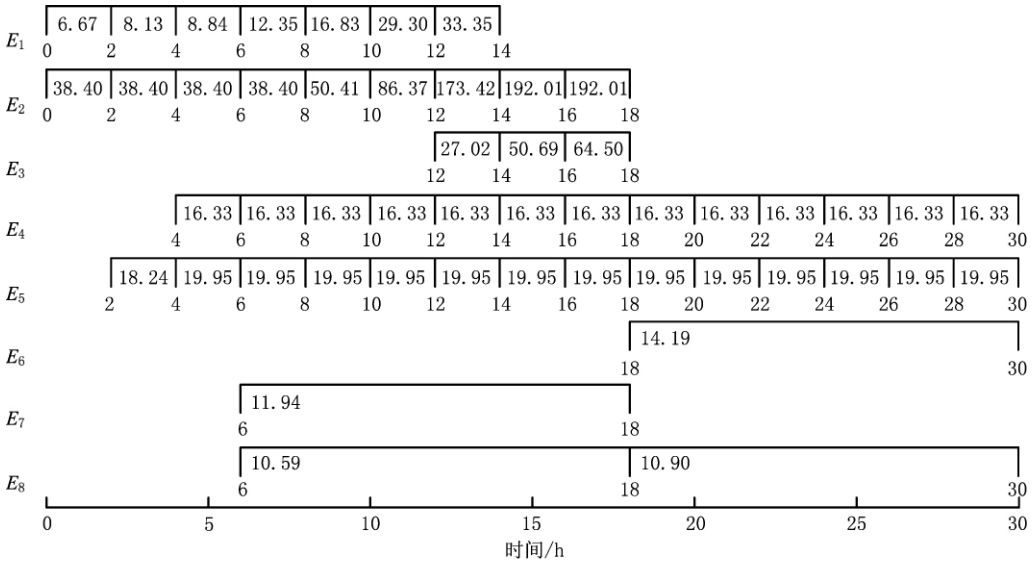
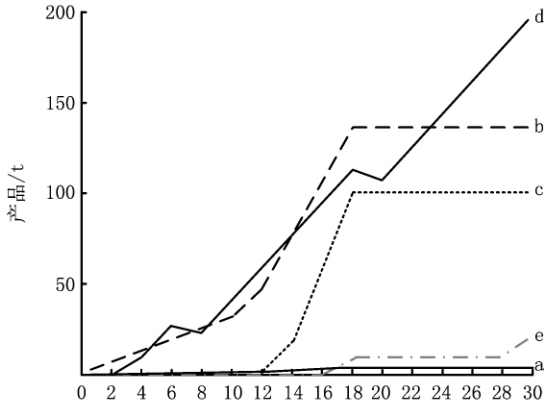
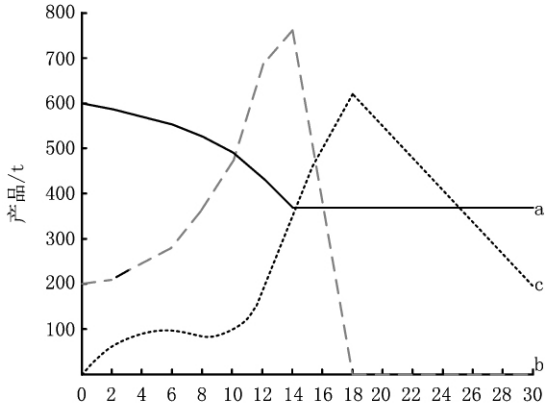


图5　机器甘特图



a 产品产量变化曲线



b 存储变化曲线

图6 产品产量变化曲线和存储变化曲线

由图 5 可以看出，本文所提出的方案得到的机器甘特图，除了最大完工时间显著缩短外，还完全消除了停机现象。对比原文中使用改进 DE 算法得到的机器甘特图可以发现，除了电解槽在第 10 h 发生停机外，原盐精制机组和 2 号熬碱大锅也分别在第 6 h 和第 16 h 停机一个调度周期。仔细研究这三个工序对应的储槽储量可以发现，三次停机均是因为储槽达到或接近了设备存储能力上限。本例中，原盐精制机组(E_1)和电解槽(E_2)在调度表前期(0 h～10 h)均安排了较小的任务量，减小存储设备负担的同时又满足了后续工序的物料需求，使两种需求得到了有效的平衡。

5 结束语

本文将连续生产过程看作处理时间等于调度时间的批量过程，建立了关于流程工业的批量－连续统一模型，采用统一的符号和规则对生产过程进行约束、优化与仿真，简化了模型的建立与分析过程。使用随机比例法编码、以最小化最大完工时间为主要优化目标，最小停机次数为辅助优化目标，最后使用蜂群算法求解本文提出的连续/批量统一模型的最优调度序列。实验证明，所提算法相比 GA 与 DE 算法有更稳定的运行表现和更强的求解能力。

引入随机比例法编码对生产任务序列进行二次映射，使进化操作直接作用于位于 [0，1] 区间的向量上，这种编码方案避免了进化操作对解的可行性的破坏，并且使任务序列在进化过程中可以动态变化，避免了一次编码在进化过程中任务序列变长的趋势。应该注意的是，交叉操作仍然建立在段—段对应的基础上，即只有相同工序的段才进行交叉。但这种交叉是在段内按照单独的位进行，因此段内基因可以通过进化操作传递给下一代。

在评价函数中加入停机次数的考虑，进一步细分了对所得解的评价等级，有助于算法在同样完工时间的解中进行比较。由于停机次数的引入，最终得到的调度方案消除了停机现象，减少了设备启停损耗，是一种较为现实和经济的评价方案。此外，由于在现实生产过程中，设备启停通常伴随着巨大的能源消耗，在保证完工时间的基础上减少设备启停次数，无疑具有极强的现实意义。

参考文献：

[1] IERAPETRITOU M G, FLOUDAS C A. Effective continuous-time formulation for short-term scheduling. 2. continuous and semicontinuous processes[J]. Industrial & Engineering Chemistry Research, 1998, 37(11): 4341-4359.

[2] MENDES J J M, GONCALVES J F, RESENDE M G C. A random key based genetic algorithm for the resource constrained project scheduling problem[J]. Computers & Operations Research, 2009, 36(1): 92-109.

[3] CASTRO P M, BARBOSA-POVOA A P, MATOS H A, et al. Simple continuous-time formulation for short-term scheduling of batch and continuous processes[J]. Industrial and Engineering Chemistry Research, 2004, 43(1): 105-118.

[4] XU Zhi, XI Yugeng, HAN Bing. Scheduling hybrid production with buffer based on genetic algotithm[J]. Control Theory and Application, 2001, 18(5): 675-680 (in Chinese). [徐 智，席裕庚，韩 兵. 基于遗传算法的一类带缓冲区的混合生产调度[J]. 控制理论与应用, 2001,18(5): 675-680.]

[5] SHAW K J, LEE P L, NOTT H P, et al. Genetic algorithms for multiobjective scheduling of combined batch/continuous process plants [C]//Proceedings of the 2000 Congress on Evolutionary Computation. Washington, D. C., USA: IEEE, 2000: 293-300.

[6] LIXIN T, XIANPENG W. A scatter search algorithm for a multistage production scheduling problem with blocking and

semi-continuous batching machine[J]. IEEE Transactions on Control Systems Technology, 2011, 19(5): 976-989.

[7] LIAO Weizhi, GU Tianlong. Scheduling and modeling of mixed production processes based on hybrid Petri nets[J]. Control and Decision, 2010, 25(6): 884-888 (in Chinese). [廖伟志, 古天龙. 一类混合生产过程的混杂 Petri 网建模与调度[J]. 控制与决策, 2010,25(6): 884-888.]

[8] WANG Haiyan, ZHAO Yanwei, XU Xinli, et al. Scheduling batch and continuous process production based on an improved differential evolution algorithm[J]. Systems Engineering—Theory and Practice, 2009, 29(11): 157-167 (in Chinese). [王海燕, 赵燕伟, 徐新黎, 等. 改进差分进化算法求解化工间歇与连续混合生产过程调度问题[J]. 系统工程理论与实践, 2009,29(11): 157-167.]

[9] KARABOGA D. An idea based on honey bee swarm for numerical optimization [R]. Kayseri, Turkey: Erciyes University, 2005.

[10] AKAY B, KARABOGA D. A modified artificial bee colony algorithm for real-parameter optimization[J]. Information Sciences, 2010,DOI:10. 1016/j. ins. 2010. 07. 015.

[11] AYAN K, KILIÇ U. Artificial bee colony algorithm solution for optimal reactive power flow[J]. Applied Soft Computing, 2012, 12(5): 1477-1482.

[12] KARABOGA D, AKAY B. A comparative study of artificial bee colony algorithm[J]. Applied Mathematics and Computation, 2009, 214(1): 108-132.

[13] LI Xiulin, LU Jianxia, CHAI Guozhong, et al. Hybrid bee colony algorithm for flexible Job Shop scheduling problem [J]. Computer Integrated Manufacturing Systems, 2011, 17 (7): 1495-1500 (in Chinese). [李修琳, 鲁建厦, 柴国钟, 等. 混合蜂群算法求解柔性作业车间调度问题[J]. 计算机集成制造系统, 2011,17(7): 1495-1500.]

[14] WANG Wanliang, ZHOU Ming, XU Xinli, et al. Research of ion exchange membrances shop scheduling problem based on improved PSO[J]. Control and Decision, 2010,25(7): 1021-1025 (in Chinese). [王万良, 周　明, 徐新黎, 等. 基于改进粒子群算法的离子膜车间调度问题研究[J]. 控制与决策, 2010,25(7): 1021-1025.]

[15] NOTT H P, LEE P L. An optimal control approach for scheduling mixed batch/continuous process plants with variable cycle time[J]. Computers & Chemical Engineering, 1999, 23(7): 907-917.

[16] BEAN J C. Genetic algorithms and random keys for sequencing and optimization. [J]. ORSA Journal on Computing, 1994, 6(2): 154.

作者简介:

张永韡(1983－),男,甘肃兰州人,博士研究生,研究方向:自然计算、群体智能,E-mail: yongwzhang@gmail. com;

汪　镭(1970－),男,江苏无锡人,教授,博士生导师,研究方向:群体智能、智能控制;

吴启迪(1947－),女,浙江永嘉人,教授,博士生导师,研究方向:智能控制。