排序理论专题

DOI:10.11721/cqnuj20240205

面向饲料加工的排产优化方法研究*

亓祥波[1]，王宏伟[1]，王浩毅[1]，马志强[1]，张　浩[2]
(1. 沈阳大学 机械工程学院，沈阳 110044；2. 中国科学院沈阳自动化研究所，沈阳 110016)

摘要：为了提高饲料企业在成本和质量上的优势，需要采用更加科学的方法制定排产计划。首先根据饲料加工排产的特点构建了基于批量组织生产的排产模型；其次，针对布谷鸟搜索(cuckoo search，CS)算法收敛速度慢与局部搜索能力弱的问题，提出不同的改进策略形成改进 CS 算法求解了排产模型，改进算法运用 NEH 方法、Logistic 混沌映射方法以及随机方法生成初始解，使用了动态改变步长的策略以平衡算法探索能力与开发能力，增加基于差分进化的交叉阶段以增强最优解的挖掘能力。采用改进 CS 算法，以最小化总流经时间为求解目标，在 40 个 Taillard 测试集实例和实际饲料排产数据上进行了实验，验证了改进 CS 算法的寻优能力。结果证明了改进 CS 算法在求解流水线式生产车间排产问题上的有效性。

关键词：置换流水车间调度；布谷鸟搜索算法；差分进化
中图分类号：O224；TP301　　**文献标志码**：A　　**文章编号**：1672-6693(2024)02-0065-09

随着智慧农业和智慧畜牧业的发展，饲料加工业内部竞争日趋激烈，促使饲料加工企业持续提高生产效率、降低生产成本并提升产品质量。近些年来，饲料加工业的生产模式已逐渐由小作坊式向工厂流水线式转变。流水线式生产模式在企业中的应用非常广泛，而且在车间生产中排产计划即使有很小的改进，车间的生产效率也可以有很大提高，生产过程中的浪费也会大大减少。因此，越来越多的饲料加工企业希望能够采用更加科学的方法制定排产计划。在考虑到目前典型的饲料加工生产流程(图 1)的基础上，可将饲料加工排产问题抽象为基于批量组织生产的流水车间调度问题。

图 1　饲料加工的典型流程

Fig. 1　Typical process of feed processing

群智能算法是一种新兴的元启发式计算技术，在求解流水车间调度这类复杂优化问题方面有很大优势，已成为越来越多研究者的关注焦点。布谷鸟搜索(cuckoo search，CS)算法是文献[1]中提出的一种基于杜鹃(*Cuculus*)的巢寄生性和 Lévy 飞行机制的群智能优化算法。在 CS 算法中有 2 种更新解的方式：一种是杜鹃寻找寄生巢产卵的方式，另一种是被杜鹃寄生的鸟以一定概率发现外来蛋后重新筑巢的方式。前一种方式采用了 Lévy 飞行的方式，后一种方式采用随机方式或者 Lévy 飞行的方式。Lévy 飞行方式是一种长步长与短步长相结合的方式[2]。CS 算法在求解置换流水车间调度问题(permutation flow scheduling problem，PFSP)上得到了应用[3]。文献[4]将 CS 算法与差分进化(differential evolution，DE)算法[5]相结合，提出了混合的 CSDE 算法并求解了工程优化问题。

本文面向饲料加工排产问题，在深入研究饲料生产特点的基础上构建了基于批量组织生产的排产模型。在原始 CS 算法基础上进行改进，提出了动态改变步长的策略，提高了解的探索与开发的平衡能力。在 CS 算法的随机重生阶段之后增加了基于 DE 算子的交叉阶段，对群体进行最优解的挖掘，提高了算法的寻优能力。最后，用改进 CS 算法对提出的排产模型进行求解。

* **收稿日期**：2022-02-25　**修回日期**：2023-07-05　**网络出版时间**：2023-07-05T18:22
资助项目：国家自然科学基金青年科学基金项目(No. 61803367)；辽宁省教育厅基本科研项目(No. LJKQZ2021164)
第一作者简介：亓祥波，男，副教授，博士，研究方向为智能优化算法及应用，E-mail：ustcdragon@syu.edu.cn
网络出版地址：https://link.cnki.net/urlid/50.1165.N.20230705.1640.004

1 饲料加工排产建模

在1个排产周期里,多个不同订单依次经过粉碎机、配料机、混合机、制粒机、分级筛和包装机。由于混合机设计产能的限制,大于产能限制的订单都要被拆成小的批次进行生产。制粒机上会根据不同订单的要求更换环模;由于制粒机在工作中温度很高,更换环模的时候需要彻底降温后才可以进行,频繁更换环模会降低效率。所以,在排产时尽量保持同尺寸环模生产。同时,因为饲料属食品类产品,不允许在其中添加防腐剂,所以饲料有一定保质期。另外,如果过多的在制饲料存放于车间内,不仅占用生产场地,也占用生产资金。综合上述情况,鉴于同一订单的多个批次之间不存在更换环模问题,故应尽量快速生产完毕;为此同一订单的多个批次应采用集中生产的方式,且可用最小最大完工时间的PFSP模型(模型Ⅰ)描述。对于多个不同订单,在排产时尽量减少更换环模次数的同时应让饲料的在制品库存最小,可以用最小总流经时间的PFSP模型(模型Ⅱ)描述。

在上述2个模型中,设n个作业$J=\{J_1,J_2,\cdots,J_n\}$在一系列m台机器$M=\{M_1,M_2,\cdots,M_m\}$上依次处理。每个作业由1组操作组成,每个操作都将在特定的机器上执行,所有作业在任何一台机器上的加工顺序都是相同的。机器M_j上作业J_i的处理时间用$P_{i,j}(i=1,2,\cdots,n;j=1,2,\cdots,m)$表示。每个作业只能在1台机器上处理,每台机器1次只能处理1个作业。

对于模型Ⅰ,将作业排列表示为$\pi=\{\pi_1,\pi_2,\cdots,\pi_n\}$,$C(\pi_i,m)$表示$\pi_i$在机器$m$上的作业完成时间,$P_{\pi_i,j}$表示机器$M_j$上作业$\pi_i$的处理时间,因此有:

$$C(\pi_1,1)=P_{\pi_1,1},$$
$$C(\pi_i,1)=C(\pi_{i-1},1)+P_{\pi_i,1},i=2,\cdots,n,$$
$$C(\pi_1,j)=C(\pi_1,j-1)+P_{\pi_1,1},j=2,\cdots,m,$$
$$C(\pi_i,j)=\max(C(\pi_{i-1},j),C(\pi_i,j-1))+P_{\pi_i,j},i=2,\cdots,n,j=2,\cdots,m,$$

从而同一订单的最大完工时间定义为:

$$C_{\max}(\pi)=C(\pi_n,m)。$$

对于模型Ⅱ,将k个订单排列表示为$O=\{O_1,O_2,\cdots,O_k\}$,$C(O_i,m)$表示O_i在机器m上的作业完成时间,k个订单所用环模型号为$D=\{D_1,D_2,\cdots,D_k\}$,$P_{O_i,j}$表示机器M_j上订单O_i的处理时间,$T_{O_i,j}$表示机器M_j上处理订单O_i时的换模时间,从而有:

$$C(O_1,1)=P_{O_1,1},$$
$$C(O_i,1)=C(O_{i-1},1)+P_{O_i,1},i=2,\cdots,k,$$
$$C(O_1,j)=C(O_1,j-1)+P_{O_1,1},j=2,\cdots,m,$$
$$T_{O_i,j}=\begin{cases}T,D_{j-1}\neq D_j,\\0,D_{j-1}=D_j,\end{cases}$$
$$C(O_i,j)=\max(C(O_{i-1},j),C(O_i,j-1))+P_{O_i,j}+T_{O_i,j},i=2,\cdots,k,j=2,\cdots,m,$$

从而不同订单的总流经时间(T_{FT})定义为:

$$T_{\mathrm{FT}}=\sum_{i=1}^{k}C(O_i,m)。$$

2 改进CS算法

2.1 原始CS算法

原始CS算法是受到杜鹃寄生行为的启发而形成的一种群智能算法[1]。假设优化问题搜索空间的维度为D',种群大小为N,则问题的解可以用向量$\boldsymbol{x}^l=\{x_1^l,x_2^l,\cdots,x_{D'}^l\}$表示,其中$l$是一个[1,$N$]内的随机数。

CS算法的伪代码如表1所示。从该伪代码可以看出,CS算法包括2个主要阶段:Lévy飞行阶段与随机重生阶段。

在Lévy飞行阶段,个体解向量($\boldsymbol{x}_p,p\in(1,2,\cdots,N)$)根据下式进行更新:

$$\boldsymbol{x}_p^{k+1}=\boldsymbol{x}_p^k+\alpha\otimes\mathrm{Levy}(\gamma),\ \mathrm{Levy}(\gamma)\sim u=t^{-1-\gamma},0<\gamma\leqslant 2。$$

其中:上标k是当前迭代次数,α是与优化问题相关的步长,$\otimes$表示按项相乘,Levy(γ)表示Lévy飞行函数,γ表示gamma函数。在随机重生阶段,根据发现概率选择部分个体,在优化问题的搜索空间内重新生成解向量替换

选择的个体。

2.2　解的表达

原始 CS 算法主要用来求解连续问题，而连续形式的解不适合求解排产问题。为了让 CS 算法能够求解排产问题，可根据最大位置值规则实现连续解向量向离散的排产问题解转换。表 2 给出了该方法的一个实例：在连续解向量中，最大值为 40.2，所以把它对应的位置 3 作为第 1 个处理的作业；在连续解向量中，第 2 大的值为 2.3，所以把它对应的位置 5 作为第 2 个处理的作业；以此类推。

2.3　解的初始化

原始 CS 算法的初始解完全依赖于随机方式，解的质量不确定性太大。为了提高初始解的质量，在原有的随机方法产生解的基础上，额外采用启发式方法 NEH 与 Logistic 混沌映射产生解。NEH 是一种求解 PFSP 模型的有效的启发式方法，运用该方法生成 1 个解。Logistic 混沌映射就是一种比较简单的一维混沌系统，该系统方程为：

$$x_{k+1}=u\times x_k\times(1-x_k),\qquad(1)$$

其中：x_k 为第 k 次迭代后的数值，$k=0,1,\cdots,n-1$，n 为作业个数；u 是控制参数，本文取 $u=4$，表示系统处于完全混沌状态；运用该方法生成 40% 的初始解。

运用 Logistic 混沌映射产生解的算法的伪代码如表 3 所示。

2.4　参数步长的设计

在原始 CS 算法中，步长是决定 Lévy 飞行的重要参数；步长值的大小起着平衡局部搜索与全局搜索的关键作用，与优化问题设计变量的范围相关：若设计变量范围越大，则步长值越大；若设计变量范围越小，则步长值越小。在迭代早期，算法尽量多地探索搜索空间，步长值应该大一些；在迭代后期，算法尽量多地开发质量高的解的邻域，步长值应该小一些。在原始 CS 算法中，步长值被设置为常数 1。为了平衡算法的探索能力与开发能力，本文提出一种动态调整步长的方法，具体为：

$$\alpha=\alpha^{\min}+\exp(-\mu\times(t/t^{\max})^2)\times(\alpha^{\max}-\alpha^{\min}),\qquad(2)$$

其中：$\alpha^{\max}$、$\alpha^{\min}$ 分别为步长的最大值和最小值；$t_{\max}$ 为算法的最大迭代次数；t 为算法当前的迭代次数，$t=1.8$；μ 为 1.8。图 2 是步长根据式(2)从 0.01 动态调整到 0.001 的效果。

2.5　改进 CS 算法设计

为了增强 CS 算法的局部开发能力，在原始 CS 算法的随机重生阶段后加入基于 DE[5] 的解的开发阶段，形成一种改进 CS 算法。有文献指出，DE 算法具有很强的收敛能力[6]。为了在后期对比参数步长动态设计带来的改进，把改进 CS 算法中去掉参数步长动态设计后的算法称为 SCSDE，把改进 CS 算法中去掉 NEH 与 Logistic 混沌映射产生初始解的算法称为 RCSDE。表 4 给出了交叉阶段的伪代码，图 3 给出了改进 CS 算法的流程。

表 1　CS 算法

Tab. 1　CS algorithm

行号	伪代码
1	初始化种群、发现概率
2	评估种群的适应度，记录最优个体
3	repeat
4	遍历每个个体
5	运用 Lévy 飞行产生新的个体
6	评估新个体的适应度
7	if(新个体的适应度好于原来个体的适应度)
8	对 2 个个体采用贪婪选择
9	endif
10	结束遍历个体
11	以一定的发现概率重新生成新的个体替换某些个体
12	更新当前最优个体
13	until(终止条件)

表 2　基于最大位置值规则的编码方式

Tab. 2　Coding method based on largest position rule

维度	位置 1	位置 2	位置 3	位置 4	位置 5	位置 6
个体	1.7	−10.9	40.2	0.8	2.3	−20
SPV 值	3	5	1	4	2	6

表 3　Logistic 混沌映射产生解的算法

Tab. 3　Algorithm for generating solutions based on logistic chaotic mapping

行号	伪代码
1	设置 $u=4$，$k=1$，n 为工件数目
2	随机产生初始值 x_1
3	repeat
4	根据式(1)生成新的值赋给 x_{k+1}
5	until($k=n$)

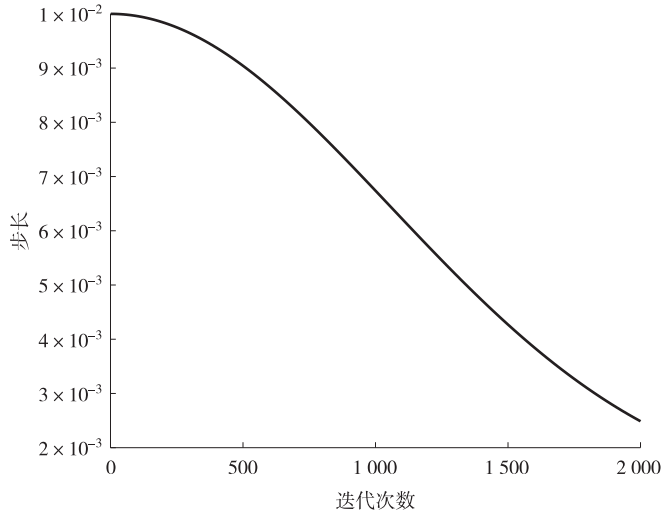


图2 步长动态调整效果

Fig. 2 Dynamic effect of step size adjustment

表4 交叉算法

Tab. 4 Cross algorithm

行号	伪代码
1	设置参数交叉概率与差分权重
2	开始遍历个体
3	开始遍历每个维度
4	生成随机数
5	if(生成的随机数小于交叉概率)
6	根据式(3)生成新个体
7	end if
8	结束遍历维度
9	计算新个体适应度
10	在新个体与原个体之间采用贪婪选择
11	结束遍历个体

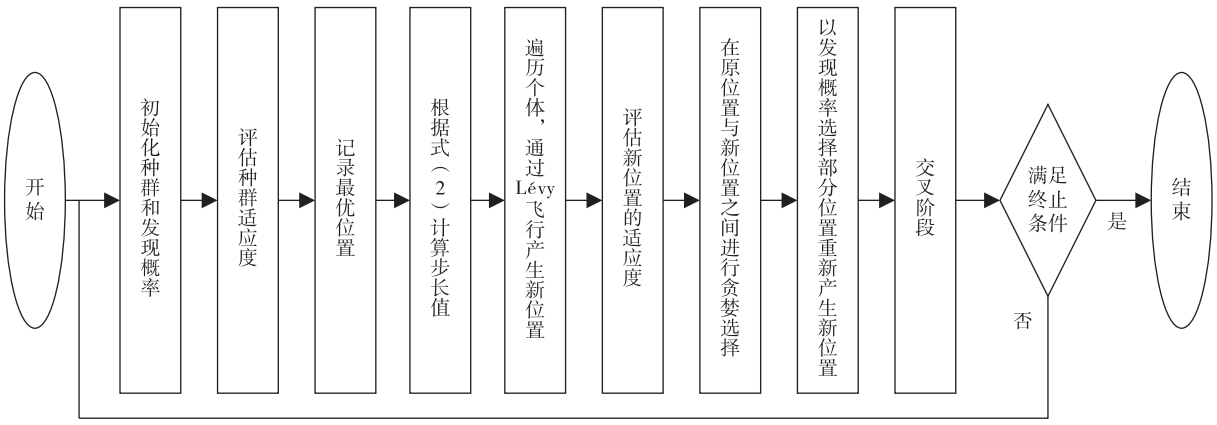


图3 改进CS算法流程图

Fig. 3 Flow chart of the improved CS algorithm

在交叉阶段,差分变异算子是主要的操作,该算子的公式如下:

$$x_{cd}=x_{rd}+F(x_{pd}-x_{qd}), \tag{3}$$

其中:x_{cd}、x_{rd}、x_{pd} 和 x_{qd} 分别为4个解向量的第 d 维,下标 c、r、p 和 q 是4个区间$[1,N]$上的不同数字,F 是差分权重。交叉阶段的每个个体根据交叉概率都有机会执行差分变异算子,所以该阶段可以大范围挖掘最优解。

3 仿真实验

3.1 实验方案

Taillard测试集是一种中到大规模问题测试集[7-8]。实验以该测试集作为基准测试实例,实验结果用每组测试集实例的最佳值相对偏差(V_{BRE})与平均值相对偏差(V_{ARE})表示,它们的计算公式为:

$$V_{\mathrm{BRE}}=\left[\sum_{I=1}^{K}(\min(T_I)-T_I^*)\Big/T_I^*\right]\Big/K, V_{\mathrm{ARE}}=\left[\sum_{I=1}^{K}(\mathrm{avg}(T_I)-T_I^*)\Big/T_I^*\right]\Big/K,$$

其中:T_I^* 是在参考文献中已知的最优结果[7],T_I 为算法在第 I 个测试实例上取得的目标函数值,K 是测试集的测试实例个数。

将CS算法[1]、CSDE算法[4]、DE算法[5]、CHIO算法[9]、PSO算法[10]、ABC算法[11]、SCSDE算法、RCSDE算法和经典的NEH算法为对比算法。所有算法采用最大位置值的方式实现PFSP解的编码与解码。对比算法中的参数按照参考文献[1,4,5,9,10,11]进行设置。使用适应度评估次数作为算法终止条件,最大评估次数设置为50 000。为了得到有效的实验结果,每种算法独立运行30次。

3.2　实验结果与分析

以最小化总流经时间为优化目标在 Taillard 测试集实例中进行实验。表 5 与表 6 分别给出了各个算法在 4 组 Taillard 测试集实例的最佳值相对偏差与平均值相对偏差，从表 5 和表 6 可知，改进 CS 算法在 3 组测试集实例中取得了最好的结果，在 4 组测试集实例中取得了最好的平均值。

表 5　算法在 Taillard 测试实例上的最佳值的平均相对偏差

Tab. 5　The average relative deviation of the best value of algorithms on Taillard instances

算法	作业个数(n)×机器台数(m)				算法	作业个数(n)×机器台数(m)			
	20×5	20×10	20×20	50×5		20×5	20×10	20×20	50×5
SCSDE	0.194 29	0.204 22	0.151 47	**3.088 69**	CSDE	1.079 64	1.021 08	0.887 37	6.152 88
CHIO	0.837 33	0.765 51	0.513 69	4.825 55	PSO	6.404 28	5.192 06	4.766 22	14.374 95
改进 CS	**0.064 56**	**0.147 37**	**0.092 25**	3.111 05	DE	0.268 14	0.534 20	0.248 68	5.376 94
CS	2.595 78	2.587 85	2.046 93	10.237 69	NEH	4.873 31	4.643 63	4.080 45	9.450 86
ABC	0.502 31	0.678 53	0.498 41	3.794 93	RCSDE	0.275 21	0.339 44	0.267 59	3.274 13

注：加粗数据表示最佳结果，下同。

表 6　算法在 Taillard 测试实例上的平均值的平均相对偏差

Tab. 6　The average relative deviation of the average value of algorithms on Taillard instances

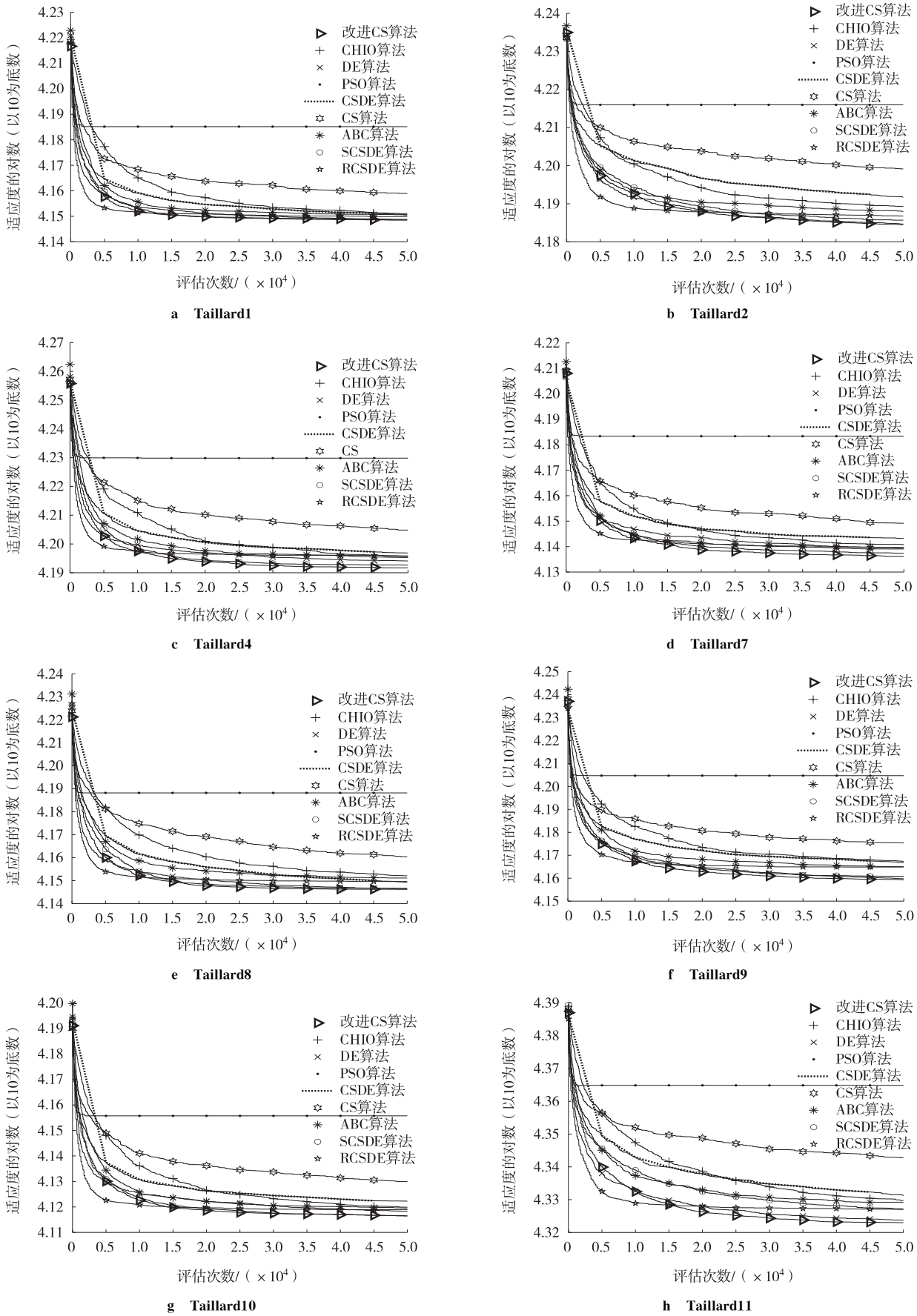
算法	作业个数(n)×机器台数(m)				算法	作业个数(n)×机器台数(m)			
	20×5	20×10	20×20	50×5		20×5	20×10	20×20	50×5
SCSDE	0.832 08	0.994 88	0.808 26	4.397 18	CSDE	1.838 64	1.964 28	1.629 31	7.146 56
CHIO	1.770 26	1.873 93	1.435 18	6.109 85	PSO	11.058 96	10.128 32	7.560 78	18.882 80
改进 CS	**0.685 37**	**0.764 76**	**0.720 12**	**4.079 42**	DE	0.992 62	1.103 67	0.817 53	6.111 44
CS	3.914 33	4.019 99	3.169 74	12.130 52	NEH	4.873 31	4.643 63	4.080 45	9.450 86
ABC	1.543 29	1.746 49	1.328 24	5.285 81	RCSDE	1.525 74	1.665 96	1.283 92	5.125 95

改进 CS 算法在 40 个 Taillard 测试集实例中的 28 个单实例上取得了最好的结果，图 6 和图 7 给出了不同算法在部分实例上的收敛曲线。从这 2 个图中可以看到，相对于其他 7 种算法，改进 CS 算法具有更强的收敛性。从 CS、RCSDE、SCSDE 与改进 CS 算法的实验数据与收敛曲线还可以看出：相对于原始 CS 算法，改进 CS 算法的初始解产生方式、动态改变步长策略以及基于 DE 的交叉策略在提升寻优精度方面有更好的改善效果；相对于 RCSDE 与 SCSDE 算法，改进 CS 算法的最佳值相对偏差和平均值相对偏差均更小，说明改进 CS 算法中采用 NEH 与 Logistic 混沌映射生成初始解的方式以及参数步长动态设计方式在提高解的精度方面有更好的改善效果。

4　求解饲料加工排产实例

表 7 给出了 24 个订单单个批次在不同设备上的加工时间，24 个订单的分批次个数依次为 3、5、10、4、10、4、3、5、10、4、13、4、3、5、10、4、12、4、3、5、10、4、5 和 4；24 个订单所用模具型号依次为 3、3.5、3.5、3.5、3.5、4.5、3、3.5、3.5、3.5、3.5、4.5、3、3.5、3.5、3.5、3.5、4.5、3、3.5、3.5、3.5、3.5 和 4.5。

分别采用 CSDE、CS、DE 与改进 CS 算法对上述 24 个订单的排产问题进行求解。实验条件与本文第 3 节中的设置一致。CSDE、CS、DE 与改进 CS 算法的平均总流经时间分别为 20 239、20 782、20 279 和 20 225 min。改进 CS 算法求解的最佳订单排序为：22、16、4、10、14、8、2、20、23、15、21、9、3、5、17、11、1、7、19、13、12、18、6、24。在上述订单排序中，模具更换次数只有 2 次。



a Taillard1　b Taillard2

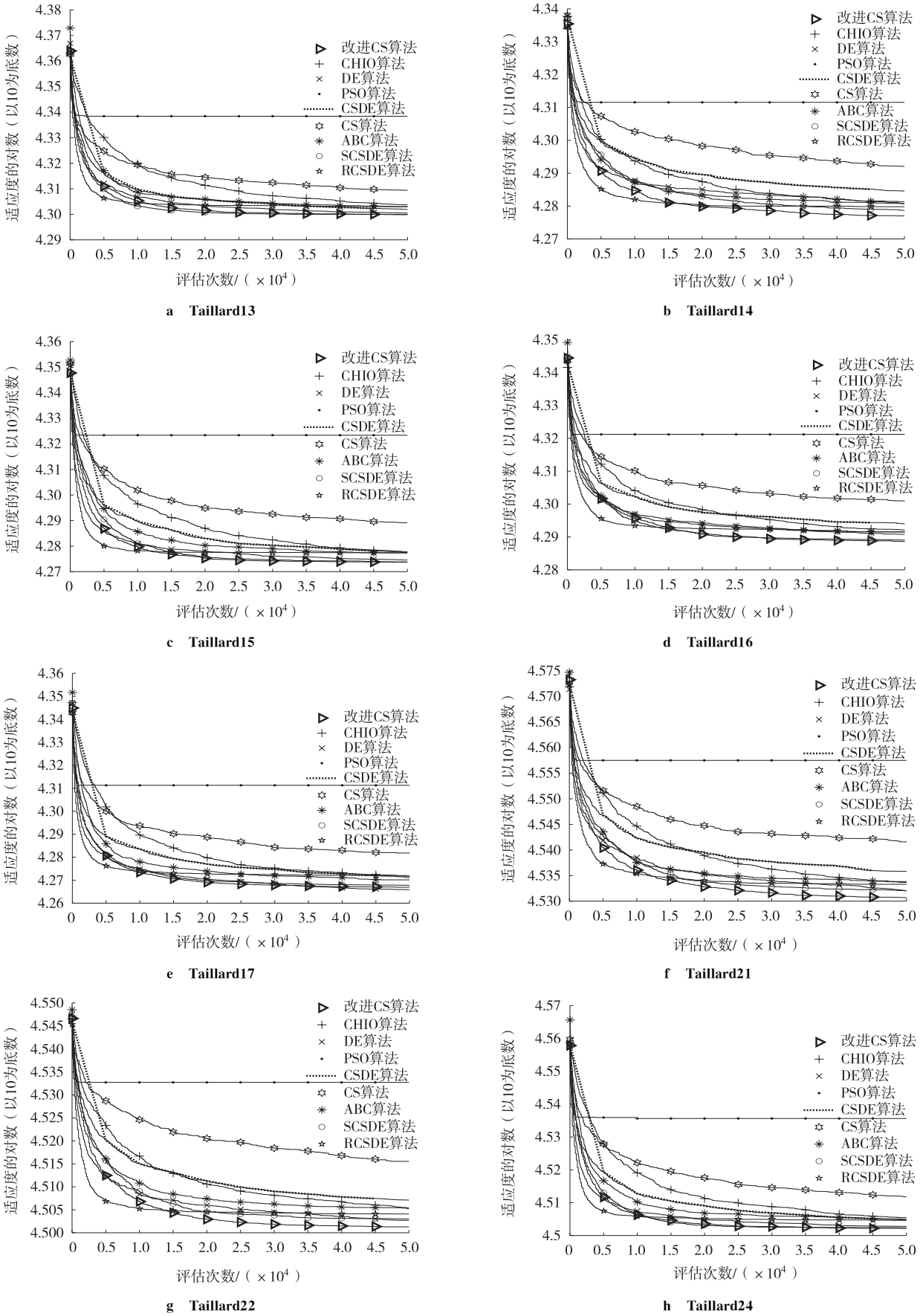
c Taillard4　d Taillard7

e Taillard8　f Taillard9

g Taillard10　h Taillard11

图6 算法在部分Taillard实例上的收敛曲线(Ⅰ)

Fig. 6 Convergence curve of algorithms on partial Taillard instances (Ⅰ)



a　Taillard13　　b　Taillard14

c　Taillard15　　d　Taillard16

e　Taillard17　　f　Taillard21

g　Taillard22　　h　Taillard24

图 7　算法在部分 Taillard 实例上的收敛曲线（Ⅱ）

Fig. 7　Convergence curve of algorithms on partial Taillard instances（Ⅱ）

表7 批次在设备上的加工时间

Tab.7 Processing time of batch on equipments min

批次编号	粉碎时间	配料时间	混合时间	制粒时间	分级筛时间	包装时间	批次编号	粉碎时间	配料时间	混合时间	制粒时间	分级筛时间	包装时间
1	5.6	2.35	3.5	8	8	6	13	5.6	2.35	3.5	8	8	6
2	5.6	2.35	3.5	8	8	6	14	5.6	2.35	3.5	8	8	6
3	5.6	2.35	3.5	4.8	4.8	6	15	5.6	2.35	3.5	4.8	4.8	6
4	5.6	2.35	3.5	4.8	4.8	6	16	5.6	2.35	3.5	4.8	4.8	6
5	5.6	2.35	3.5	10	10	6	17	5.6	2.35	3.5	10	10	6
6	5.6	2.35	3.5	8	8	6	18	5.6	2.35	3.5	8	8	6
7	5.6	2.35	3.5	8	8	6	19	5.6	2.35	3.5	8	8	6
8	5.6	2.35	3.5	8	8	6	20	5.6	2.35	3.5	8	8	6
9	5.6	2.35	3.5	4.8	4.8	6	21	5.6	2.35	3.5	4.8	4.8	6
10	5.6	2.35	3.5	4.8	4.8	6	22	5.6	2.35	3.5	4.8	4.8	6
11	5.6	2.35	3.5	10	10	6	23	5.6	2.35	3.5	10	10	6
12	5.6	2.35	3.5	8	8	6	24	5.6	2.35	3.5	8	8	6

5 结束语

本文根据饲料加工生产与工艺特点将饲料加工排产问题抽象为基于批量组织生产的流水车间调度问题,构建了相应排产模型。在原始CS算法基础上提出了一种改进CS算法,该算法采用动态改变步长的策略以平衡解的探索能力与开发能力,增加基于DE的交叉阶段以增强算法对最优解的开发能力。针对PFSP模型,以最小化总流经时间为优化目标,以原始CS算法以及其他6个智能算法作为对比算法,在40个Taillard测试集实例上进行了仿真实验,实验结果表明本文提出的有关改进策略有效提高了解的寻优能力和收敛速度。最后,以实际的饲料加工排产数据为依据,采用改进CS算法对饲料加工排产问题进行求解,结果显示该算法能够给出更好的排产序列。

参考文献:

[1] YANG X S,DEB S. Cuckoo search via Lévy flights[C]//2009 World Congress on Nature and Biologically Inspired Computing. Coimbatore:IEEE,2009:210-214.

[2] VISWANATHAN G M. Fish in Lévy-flight foraging[J]. Nature,2010,465(7301):1018-1019.

[3] 刘长平,叶春明. 求解置换流水车间调度问题的布谷鸟算法[J]. 上海理工大学学报,2013,35(1):17-20.
LIU C P,YE C M. Cuckoo search algorithm for the problem of permutation flow shop scheduling[J]. Journal of University of Shanghai for Science and Technology,2013,35(1):17-20.

[4] ZHANG Z C,DING S F,JIA W K. A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems[J]. Engineering Applications of Artificial Intelligence,2019,85(6):254-268.

[5] STORN R,PRICE K. Differential evolution:a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization,1997,11(4):341-359.

[6] GONG W Y,CAI Z H,LING C X. DE/BBO:a hybrid differential evolution with biogeographybased optimization for global numerical optimization[J]. Soft Computing,2011,15(4):645-665.

[7] ZHANG Y,LI X P,WANG Q. Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization[J]. European Journal of Operational Research,2009,196(3):869-876.

[8] WANG L,PAN Q K,SUGANTHAN P N,et al. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems[J]. Computers & Operations Research,2010,37(3):509-520.

[9] AL-BETAR M A,ALYASSERI Z A A,AWADALLAH M A,et al. Coronavirus herd immunity optimizer (CHIO)[J]. Neural Computing & Applications,2021,33:5011-5042.

[10] KENNEDY J, EBERHART R. Particle swarm optimization[C]//Proceedings of ICCN'95-International Conference on Neural Networks. Perth: IEEE, 1995.

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

Special Topics on Scheduling Theory

Study on Production Scheduling Optimization Method for Feed Processing

QI Xiangbo[1], WANG Hongwei[1], WANG Haoyi[1], MA Zhiqiang[1], ZHANG Hao[2]

(1. College of Mechanical Engineering, Shenyang University, Shenyang110044;

2. Shenyang Institute of Automation Chinese Academy of Sciences, Shenyang 110016, China)

Abstract: In order to improve the cost and quality advantages of feed enterprises, more scientific methods can be used to formulate production scheduling plans. Firstly, according to the characteristics of feed processing scheduling, a scheduling model based on batch organization production is constructed. Secondly, aiming at the problems of slow convergence speed and weak local search ability of cuckoo algorithm, an improved cuckoo optimization algorithm with different improvement strategies is formed. The NEH method, the Logistic chaotic mapping method and the random method were employed to create initial solution. The strategy of dynamically changing the step size is used to balance the exploration ability and exploitation ability of the algorithm. The crossover stage based on differential evolution is added to enhance the mining ability of the optimal solution. The improved cuckoo search algorithm is used to solve 40 Taillard test problems and an actual feed scheduling problem with minimization of total flow time. The experimental results show the effectiveness of the proposed algorithm in solving flow shop scheduling problem.

Keywords: permutation flow shop scheduling; cuckoo search; differential evolution

(责任编辑 方 兴)