

基于混合离散蝙蝠算法的跨工序协同调度问题


鲁建厦<sup>1,2</sup> 李晋青<sup>1</sup> 汤洪涛<sup>1</sup>

1. 浙江工业大学机械工程学院,杭州,310023
- 2.浙江汇智物流装备技术有限公司,湖州,313028

**摘要:**针对跨工序的生产与配送协同调度问题,构建了前工序单机批加工、后工序多产线逐订单加工,且工序之间采用自动引导车循环配送的协同调度模型。以最小化最大完工时间和后工序前的在制品等待时间为调度目标,设计了融合模拟退火算法与解串算法的混合离散蝙蝠算法,与改进的离散粒子群算法和Ullrich遗传算法相比,该算法能很好地减少后工序产线前的队列等待时间,缩短产品的生产周期。

**关键词:**协同调度;跨工序;混合离散蝙蝠算法;自动引导车

**中图分类号:**TP242

**DOI:**10.3969/j.issn.1004-132X.2020.06.013      **开放科学(资源服务)标识码(OSID):** 

Cross Process Coordinated Scheduling Problem Based  
on Hybrid Discrete Bat Algorithm

LU Jiansha<sup>1,2</sup> LI Jinqing<sup>1</sup> TANG Hongtao<sup>1</sup>

1. College of Mechanical Engineering, Zhejiang University of Technology, Hangzhou, 310023
- 2.Zhejiang Huizhi Logistics Equipment Technology Co., Ltd., Huzhou, Zhejiang, 313028

**Abstract:** Aiming at the problem of collaborative scheduling of production and distribution across processes, a collaborative scheduling model of single-machine batch processing in former process and order-by-order processing in latter process was constructed, and AGV circular distribution was adopted among the processes. A hybrid discrete bat algorithm, which is combined with simulated annealing algorithm and modified unstringing and stringing algorithm, was designed to minimize the maximum completion time and waiting time before subsequent process. Compared with the improved discrete particle swarm optimization and Ullrich genetic algorithm, the proposed algorithm may reduce queue waiting time and production cycle of products.

**Key words:** coordinated scheduling; cross process; hybrid discrete bat algorithm; automated guided vehicle(AGV)

0 引言

近年来,自动引导车(AGV)已在智能物流系统中大量使用,它与设备、产线之间如何协同调度显得越来越重要,但这方面的研究比较欠缺,且绝大部分协同调度研究集中于解决前工序的生产调度及工序间物料配送的问题,忽略了后工序的加工特性。MAGGU 等<sup>[1]</sup>在具有无限配送车辆情况下,解决了的生产线上下游协同调度的问题。ARMENTANO 等<sup>[2]</sup>通过禁忌搜索算法解决了具有有限配送周期和有限容积车辆的生产与库存

成本的最小化问题。方伯芃等<sup>[3]</sup>针对具有随机性、并发性、模糊性的不确定环境下的供应链系统,通过启发式寻优得到产业链生产与配送协同调度的最优方案。李凯等<sup>[4]</sup>在研究单机多车情况下的生产与配送问题时,将配送车辆容量进行了限制,在模型中考虑了制造商信誉惩罚成本和配送成本,建立了以总成本最低为目标的协同调度模型,并构造了模拟退火算法进行求解。关静等<sup>[5]</sup>针对钢铁企业中工件的生产与生产前运输的协同调度问题,考虑具有温降属性的工件在等待加工时会使处理时间延长等因素,建立了以最小化最大完工时间为目标的单机多车流水生产的调度模型,并证明了该问题属于强 NP 难问题。马文琼等<sup>[6]</sup>分析了多段装配流水车间工件加工和成品配送的协同调度特征,将产品的加工次序和配送批次作为决策变量,建立了以产品交付时间和

收稿日期:2018-10-08      修回日期:2019-11-27

基金项目:国家重点研发计划资助项目(2018YFB1308100);浙江省重点研发计划资助项目(2018C01003);特种装备制造与先进加工技术教育部/浙江省重点实验室开放基金资助项目(EM201720104)

万方数据

运输费用整体最优为目标的调度模型,并提出了一种基于遗传算法和反向变邻域搜索的混合智能优化算法进行求解。刘玲<sup>[7]</sup>针对单机多车且未给定订单生产完成时间的协同调度问题,建立了以最小化最晚订单交付时间为目标的生产与配送协同调度模型,提出了一种混合禁忌搜索算法进行求解。卓雪雪<sup>[8]</sup>研究了并行批处理机环境下的协同调度问题,将该问题模型划分为组成批次阶段与各批次调度阶段,大型实例仿真表明在解决该类问题时,融合了下界算法的改进型最大最小蚁群算法(min max ant system,MMAS)的整体性能优于混合蚁群优化(hybrid ant colony optimization,HACO)算法。

当前,国内外学者对协同调度问题的研究更多地是将前工序的生产调度与物料配送相结合,单纯把后工序当作配送需求点而不考虑其加工特性,忽视了前工序、物料配送、后工序三者之间的关联调度。按需生产的拉式生产系统中,各订单的生产需求指令是由后工序发送到前工序的,后工序如何与前工序以及中间配送环节进行协同调度显得十分重要。本文研究了一种同时考虑配送车辆容量和数量限制且存在在制品缓存区的跨工序协同调度问题,假定前工序为单机批加工,后工序为多产线逐订单加工,各订单不可拆分且与后工序产线的对应关系已知。当前加工工序的一个订单加工批量小于批加工工序产能时,将多订单组合为一个批次进行加工。订单在前加工工序加工完成后,AGV 就位装车,然后根据所设计的配送顺序配送到多个位置的后工序产线,订单在后工序各产线依照先进先出原则排队逐个进行加工。生产周期和产线前的缓冲队列等待时间对生产系统效率的影响很大,为此将调度目标定为整体最小化最大订单完工时间及各订单在产线队列缓冲时间。该类问题属于 NP 难问题,为此设计了一种融合模拟退火算法及解串算法的混合离散蝙蝠算法。

1 问题描述与模型建立

以往研究所建立的模型更多地是将后工序看作单纯的配送点,忽略了后工序的生产可能影响前工序的调度及工序间物料配送,造成实际应用中的在制品大量堆积,显著延长了产品的生产周期。本文研究的跨工序生产与配送协同调度问题的最主要特征是考虑了后工序的加工特性。如图 1 所示,跨工序协同调度模型可分为单机批加工

工序(前加工工序)、工序间物料配送、多产线逐订单加工工序(后加工工序)3 个阶段。

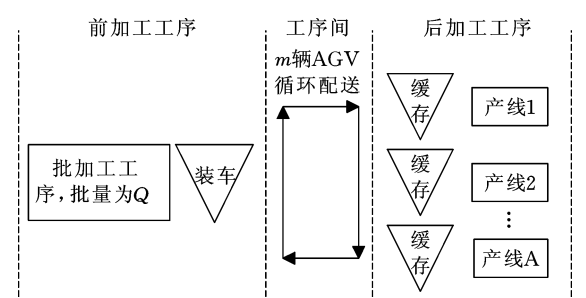


图 1 跨工序协同调度模型

Fig.1 Cross process collaborative scheduling model

首先,后工序各产线发出多个订单需求至前工序,并确定加工批次与加工顺序。同批次内的订单同时开始加工,加工时间取决于该批次内加工时间最长的订单。一批订单在前工序加工完成后,若 AGV 已就位则可依次装车,否则需要等待。AGV 的数量和容量有限,当 AGV 装载的在制品达到其最大容量时开始配送,并根据算法设计的顺序配送到后工序所需产线。当该运输批次所有配送任务完成后,返回到前工序等待下一次配送。后工序各产线依照订单需求进行加工,且加工时间为固定值。当在制品到达后工序时,若该产线处于忙碌状态,则需排队等待加工,加工完毕后,该订单整个加工流程结束。

本模型调度的目标是同时最小化最大订单完工时间与各订单在产线队列的缓冲时间。

假定:初始状态下,前工序、AGV 以及各产线皆为闲置。 $A$  条产线发出的  $n(A > n)$  个订单需求分别为  $J_1、J_2、\cdots、J_n$ ,订单与产线所属关系已知,各订单容量分别为  $W_1、W_2、\cdots、W_n$ 。前工序的单批次加工时间  $R_x$  为该批次加工时间最长的订单时间, $x$  为批次号,前工序最大加工批量为  $Q$ 。零件经过前工序后,可能的装车等待运输时间分别为  $D_1、D_2、\cdots、D_n$ ,而后经  $m$  辆 AGV 采用循环方式运输至后工序指定的产线进行加工。后工序产线前可能存在的队列缓冲时间为  $E_1、E_2、\cdots、E_n$ ,对应的产线加工时间分别为  $T_1、T_2、\cdots、T_A$ 。每台 AGV 装载容量均为  $Z$ 。设产线  $a$  到产线  $p$  的运输时间为  $t_{ap}$  ( $a, p \in [0, A]$ ),往返时间相同即  $t_{ap} = t_{pa}, t_{aa} = t_{pp} = 0$ ,前工序至产线  $a$  的运输时间为  $t_{0a}$ 。订单在加工和运输过程中不可中断。队列缓存策略为先进先出(FIFO)。最后一个订单加工完成的时间记为  $C_{\max}$ ,目标是最小化  $C_{\max}$  以及各订单在产线前的队列等待时间之和  $\sum_{i=1}^n E_i$ 。

跨工序的协同调度模型如下。

目标函数即最小化的最大完工时间及各订单在产线前队列等待时间之和为

$$F(x) = w_1 \min(\max_{j \in [1, n]} C_j) + w_2 \min(\sum_{i=1}^n E_i) \quad (1)$$

式中,  $w_1$ 、 $w_2$  分别为目标函数中最小化最大完工时间  $C_{\max}$  及各订单在产线前等待时间之和  $\sum_{i=1}^n E_i$  的权重;  $C_j$  为订单  $j$  的完工时间。

模型的约束如下:

(1) 在批加工工序生产一批产品的时间取决于该批次加工时间最长的订单, 即

$$R_x = \max_{j \in [1, n]} e_{jx} p_j \quad (2)$$

式中,  $p_j$  为订单  $j$  在批加工工序的加工时间;  $e_{jx}$  为 0-1 变量, 当订单  $j$  分配到第  $x$  加工批次进行加工时为 1, 其余为 0。

(2) 某订单的运输装载装车时间等于它的批加工工序结束时间与等待 AGV 到达时间之和, 即

$$q_j = \sum_{i \in [1, j]} R_L + D_j \quad (3)$$

式中,  $q_j$  为订单  $j$  的 AGV 装车时间,  $j = 1, 2, \dots, n$ ;  $L$  为订单  $j$  所分到的加工订单批次;  $R_L$  为订单  $j$  的批加工时间。

(3) 各订单只能分配给一个批加工批次, 即

$$\sum_{x=1}^X e_{jx} = 1 \quad (4)$$

(4) 批加工工序的加工能力限制为

$$\sum_{j=1}^n e_{jx} w_j \leq Q \quad (5)$$

(5) 批加工工序包含的订单数量之和等于总订单数, 即

$$\sum_{j=1}^n \sum_{x=1}^X e_{jx} = n \quad (6)$$

(6) 批加工工序的加工开始时间等于上一批加工完工时间, 即

$$h_x = h_{x-1} + R_{x-1} \quad (7)$$

式中,  $h_x$  为前工序第  $x$  批次订单加工开始时间,  $h_0 = 0$ 。

(7) 每个订单只能分配给一个运输批次, 即

$$\sum_{k=1}^K r_{jk} = 1 \quad (8)$$

式中,  $r_{jk}$  为 0-1 变量, 当订单  $j$  分配到运输批次  $k$  进行运输时为 1, 其余为 0。

(8) 所有运输批次包含的订单数量之和等于订单总数, 即

$$\sum_{j=1}^n \sum_{k=1}^K r_{jk} = n \quad (9)$$

(9) 运输批次  $k$  的开始时间为所属该运输批次订单的最晚装车时间与所属该运输批次 AGV 到达初始位置时间的较大值, 即

$$a_k = \max(\max_{j \in [1, n]} (r_{jk} q_j), b_k) \quad (10)$$

万方数据

$$k = \sum_{m=1}^M z_{km} \delta_{ml} \quad l = \sum_{m=1}^M z_{km} l_{km} - 1$$

式中,  $a_k$  为运输第  $k$  批订单的开始时间,  $k = 1, 2, \dots, K$ ;  $K$  为最大运输批次;  $\delta_{ml}$  为第  $m$  辆 AGV 输送次序为  $l$  的运输批次;  $z_{km}$  为 0-1 变量, 当第  $k$  运输批次由第  $m$  ( $m = 1, 2, \dots, M$ ) 辆 AGV 运输时为 1, 其余为 0;  $M$  为 AGV 数量;  $b_k$  为第  $k$  运输批次运输完毕后返回到批加工车间准备下一轮运输的时间;  $l_{km}$  为第  $k$  批次在第  $m$  辆 AGV 上的输送次序。

(10) 每个运输批次只能分配给 1 辆 AGV, 即

$$\sum_{m=1}^M z_{km} = 1 \quad (11)$$

(11) 所有 AGV 运输的批次之和为

$$\sum_{k=1}^K \sum_{m=1}^M z_{km} = K \quad (12)$$

(12) 运输批次应满足其 AGV 装载容量限制, 即

$$\sum_{j=1}^n r_{jk} w_j \leq Z \quad (13)$$

(13) 各并行产线以订单为生产单元, 即

$$\sum_{a=1}^A u_{ai} u_{aj} (C_i - C_j + \varphi_{ij}^{(a)} P_N) \geq \sum_{a=1}^A u_{ai} u_{aj} T_a \quad (14)$$

$$\sum_{a=1}^A u_{ai} u_{aj} [C_j - C_i + (1 - \varphi_{ij}^{(a)}) P_N] \geq \sum_{a=1}^A u_{ai} u_{aj} T_a \quad (15)$$

式中,  $u_{aj}$  为 0-1 变量, 当订单  $j$  是由产线  $a$  提出需求时为 1, 否则为 0;  $\varphi_{ij}^{(a)}$  为 0-1 变量, 当产线  $a$  上订单  $i$  先于订单  $j$  加工为 1, 否则为 0;  $P_N$  为极大数。

(14) 每个订单到达该订单所属产线的时间为

$$b_{kj} = \begin{cases} a_k + \sum_{a=1}^A t_{0a} u_{aj} & v_{jk} = 1 \\ b_{ki} + \sum_{p=1}^A \sum_{a=1}^A t_{ap} u_{ai} t_{pj} & v_{jk} \neq 1 \end{cases} \quad (16)$$

式中,  $b_{kj}$  为第  $k$  运输批次中订单  $j$  的达到指定产线的时间;  $v_{jk}$  为在第  $k$  运输批次中, 订单  $j$  的运输序号,  $v_{ik} = v_{jk} - 1$ 。

(15) 某一订单的最终加工完成时间为

$$C_j = b_{kj} + E_j + \sum_{a=1}^A u_{aj} T_a \quad (17)$$

(16) 某一运输批次完成运输返回到批加工工序的时间为

$$b_k = \sum_{a=0}^A \sum_{p=0}^A t_{ap} g_{k,ap} + a_k \quad (18)$$

式中,  $g_{k,ap}$  为 0-1 变量, 当任意运输批次  $k$  经过产线  $a$  至产线  $p$  之间时为 1, 否则为 0。

## 2 算法设计

蝙蝠算法作为一种新型的启发式算法, 在继承和声算法、粒子群算法优点的同时, 利用回声定位原理能同时搜索待优化空间中的多个区域,

通过频率调谐原理实现动态控制全局搜索与局部搜索的动态转换,从而获得比其他算法更好的收敛性。但蝙蝠算法只能解决连续域寻优问题<sup>[9]</sup>,而本文所研究的协同调度问题属于离散排列组合问题,因此需要在传统蝙蝠算法的基础上进行离散化。此外,为了避免蝙蝠算法后期收敛速度慢、易陷入局部最优等缺点,该算法融合了变温度的模拟退火算法来提高其全局搜索能力和后期收敛效率。解串算法在解决路径规划问题时具有结构简单、计算效率高、便于实现等特点,所以本文将解串算法嵌套在蝙蝠算法来解决跨工序协同调度问题。

2.1 编码方式

混合离散蝙蝠算法求解跨工序协同调度问题时,首先要考虑编码问题,由于研究的问题包括生产调度和物料配送,因此采用单层整数编码方式。编码长度为  $n$ ,各订单的编码位代表该订单在批加工工序的生产顺序,编码形式表示为

$$\mathbf{A}=(m_1,m_2,\cdots,m_n) \tag{19}$$

式中, $m_i(i=1,2,\cdots,n)$ 为批加工工序生产排序为  $i$  的订单。

加工批次及运输批次划分需在满足批加工批量和运输批量要求的基础上进行划分。如图 2 所示,9 个订单的订单容量分别为 2、3、4、3、1、3、2、3、1,批加工工序加工批量为 6,AGV 的运输批量为 4,假定一个工序编码  $\mathbf{A}=(2,5,3,7,4,6,8,1,9)$ 。按照生产序列以及加工批量限制,将  $J_2、J_5$  分配到加工批次 1,  $J_3、J_7$  分配到加工批次 2,  $J_4、J_6$  分配到加工批次 3,  $J_1、J_8、J_9$  分配到加工批次 4。根据 AGV 运输批量限制,将  $J_2、J_5$  分配到运输批次 1,  $J_3$  分配到运输批次 2,  $J_7$  分配到运输批次 3,  $J_4$  分配到运输批次 4,  $J_6$  分配到运输批次 5,  $J_8$  分配到运输批次 6,  $J_1、J_9$  分配到运输批次 7。

加工批次1		加工批次2		加工批次3		加工批次4		
2	5	3	7	4	6	8	1	9
运输批次1	运输批次2	运输批次3	运输批次4	运输批次5	运输批次6	运输批次7		

图 2 编码方式

Fig.2 Coding mode

2.2 混合离散蝙蝠算法设计

针对标准的蝙蝠算法易陷入局部最优,后期收敛速度慢等缺点,本文设计了一种混合离散蝙蝠算法(hybrid discrete bat algorithm, HDBA)来

解决跨工序协同调度问题。

标准蝙蝠算法中,每只虚拟蝙蝠在  $n$  维(研究问题的维数)搜索空间主要通过位置矢量和速度矢量进行表达。搜索过程中,蝙蝠个体在迭代过程中经过位置矢量和速度矢量的更新后会逐渐逼近最优解。第  $t$  代蝙蝠个体  $i$  的速度矢量和位置矢量表达式为

$$\mathbf{v}_{i,t}=\mathbf{v}_{i,t-1}+(\mathbf{X}_{i,t}-\mathbf{X}^*) \tag{20}$$

$$\mathbf{X}_{i,t}=\mathbf{X}_{i,t-1}+\mathbf{v}_{i,t} \tag{21}$$

$$f_i=f_{\min}+\beta(f_{\max}-f_{\min}) \tag{22}$$

式中, $f_i$  表示频率属性,蝙蝠个体的频率在  $[f_{\min}, f_{\max}]$  随机选择; $f_{\max}、f_{\min}$  分别为脉冲频率的上下限; $\beta$  为  $0\sim 1$  之间的随机变量; $\mathbf{X}^*$  为当前全局最佳位置。

假定协同调度问题的解决方案有  $n$  维,那么第  $i$  个蝙蝠的  $n$  维位置矢量  $\mathbf{X}_i=(X_{i1},X_{i2},\cdots,X_{in})$ 。矢量  $\mathbf{v}_{i,t}=(v_{1,t},v_{2,t},\cdots,v_{k,t})$  可以看作  $k$  维方向矢量,全局最优解  $\mathbf{X}^*=(X_1^*,X_2^*,\cdots,X_n^*)$ 。标准位置矢量和速度矢量公式可更新为

$$\mathbf{v}_{i,t}=\varphi(\mathbf{X}_{i,t-1},\mathbf{X}^*,f_i,M_{i,t}) \tag{23}$$

$$\mathbf{X}_{i,t}=\omega(\mathbf{X}_{i,t-1},\mathbf{v}_{i,t}) \tag{24}$$

$f_i$  是一个大小位于  $f_{\min}$  与  $f_{\max}$  之间的随机整数,它代表了子代从父代  $\mathbf{X}^*$  和  $\mathbf{X}_{i,t-1}$  复制单位的大小,将式(22)更新为

$$f_i=\mu(f_{\max},f_{\min}) \tag{25}$$

变量  $M_{i,t}$  是介于 1 与  $\mathbf{X}_{i,t-1}$  和  $\mathbf{X}^*$  的汉明距离的随机整数, $M_{i,0}=0$ 。向上圆整函数  $R_{UP}(M_{i,t}/f_i)$  为返回一系列排列  $\mathbf{v}_{i,t}$  的个数  $k$ 。

式(23)包含参数  $\mathbf{X}_{i,t-1}、\mathbf{X}^*、f_i、M_{i,t}$ ,速度矢量如图 3 所示,首先左至右扫描父代的染色体,如果某一位基因在所有父代中的基因位相同,则将该基因复制到子代的同一位置。然后,未被安排的基因以  $f_i$  为编组单元大小,从任意父代基因位进行交叉操作,将改组基因拷贝到子代相应基因位上(如果父代出现重复基因,则从任意父代未

编组大小即频率  $f=2, 10$  维搜索空间,  $M_i=8$

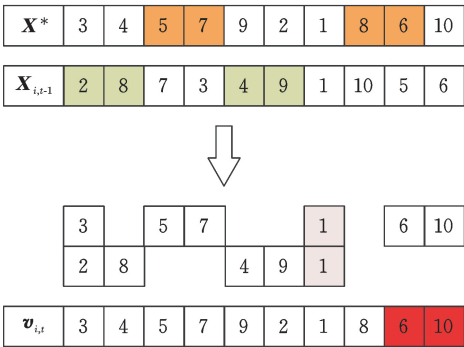


图 3 速度矢量更新

Fig.3 Speed vector update



出现的基因上随机选择一个基因并将该基因复制到子代相同基因位上)。最后,子代染色体  $\mathbf{v}_{i,t}$  从左至右依照编组单元大小进行划分( $\mathbf{v}_{i,t}$  存在最后一部分基因分组数量小于频率  $f_i$  情况),根据  $M_{i,t}$  得到  $\mathbf{v}_{i,t}$ 。图 2 示例的最终输出结果  $\mathbf{v}_{i,t} = (3,4,5,7,9,2,1,8)$ 。

式(24)包含 2 个参数,蝙蝠  $i$  新的位置矢量为  $\mathbf{X}_{i,t}$ ,父代  $\mathbf{X}_{i,t}$  根据式(23)所得的一系列组合  $\mathbf{v}_{i,t}$  进行对应基因位的交叉,具体流程如图 4 所示。 $f_i \geq 3$  时,按照  $\mathbf{v}_{i,t}$  的编组大小,对  $\mathbf{X}_{i,t}$  进行组内基因位随机交换。多次交换后,得到蝙蝠  $i$  在迭代  $t$  次后的位置矢量  $\mathbf{X}_{i,t}$ 。

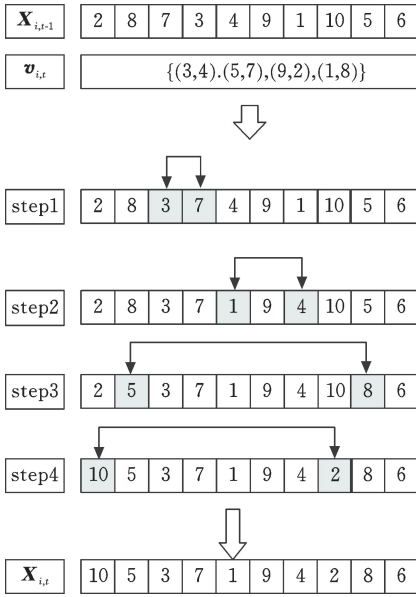


图 4 位置矢量更新

Fig.4 Position vector update

蝙蝠算法在进行个体局部搜索时,它的新解决方案会在当前最优解的随机游过程中就近产生:

$$\mathbf{X}_{\text{new}} = \mathbf{X}_{\text{old}} + \epsilon A_t \quad (26)$$

式中,  $\epsilon$  是  $-1 \sim 1$  之间的随机变异因子;  $A_t$  为所有蝙蝠在  $t$  代的平均响度,  $A_t = \langle A_{i,t} \rangle$ ;  $A_{i,t}$  为蝙蝠  $i$  在第  $t$  次迭代时的脉冲响度;  $\mathbf{X}_{\text{old}}$  为局部搜索前的当前最优解;  $\mathbf{X}_{\text{new}}$  为局部搜索后产生的解。

如式(26)所示,传统蝙蝠算法邻域搜索中,  $\epsilon$  和  $A_t$  作为一维连续变量并不适用于解决离散问题,为了解决离散问题同时提高局部搜索能力,则可更新局部搜索产生的解  $\mathbf{X}_{\text{mid}}$  为

$$\mathbf{X}_{\text{mid}} = \begin{cases} \text{InsertionFun}(\mathbf{X}_e, |R_{\text{UP}}(\epsilon A_t)|) & \epsilon A_t \geq 0 \\ \text{ExchangeFun}(\mathbf{X}_e, |R_{\text{DOWN}}(\epsilon A_t)|) & \epsilon A_t < 0 \end{cases} \quad (27)$$

式中,  $\mathbf{X}_e$  为当前最优解;  $R_{\text{UP}}(\cdot)$  表示向上圆整函数;  $R_{\text{DOWN}}(\cdot)$  表示向下圆整函数。

$\epsilon A_t \geq 0$  时,  $\text{InsertionFun}(\mathbf{X}_e, |R_{\text{UP}}(\epsilon A_t)|)$  表示对  $\mathbf{X}_e$  进行  $|R_{\text{UP}}(\epsilon A_t)|$  次随机基因插入操作

若  $\mathbf{X}_e = (10, 5, 3, 7, 1, 9, 4, 2, 8, 6)$ ,  $|R_{\text{UP}}(\epsilon A_t)| = 2$ , 则对  $\mathbf{X}_e$  进行 2 次随机基因插入操作。2 次随机插入操作后更新的解为

$$\mathbf{X}_{e,1} = (10, 5, 7, 1, 9, 3, 4, 2, 8, 6)$$

$$\mathbf{X}_{\text{mid}} = \mathbf{X}_{e,2} = (10, 5, 7, 1, 3, 4, 2, 8, 9, 6)$$

$\epsilon A_t < 0$  时,  $\text{ExchangeFun}(\mathbf{X}_e, |R_{\text{DOWN}}(\epsilon A_t)|)$  表示对  $\mathbf{X}_e$  进行  $|R_{\text{DOWN}}(\epsilon A_t)|$  次随机基因交换操作。若  $\mathbf{X}_e = (10, 5, 3, 7, 1, 9, 4, 2, 8, 6)$ ,  $|R_{\text{DOWN}}(\epsilon A_t)| = 2$ , 则对  $\mathbf{X}_e$  进行 2 次随机基因交换操作。2 次随机交换操作后更新的解为

$$\mathbf{X}_{e,1} = (10, 5, 7, 2, 9, 3, 4, 1, 8, 6)$$

$$\mathbf{X}_{\text{mid}} = \mathbf{X}_{e,2} = (10, 2, 7, 5, 9, 3, 4, 1, 8, 6)$$

经过局部搜索得到  $\mathbf{X}_{\text{mid}}$  后,为提高精英群体的质量,采用变温度的模拟退火算法对  $\mathbf{X}_{\text{mid}}$  进行退火操作。若  $\mathbf{X}_{\text{mid}}$  优于  $\mathbf{X}_e$  则保留新解,否则以概率  $\exp(-\Delta\theta'/\theta)$  接受新解,其中,  $\theta$  为当前退火温度,  $\Delta\theta'$  为  $\mathbf{X}_e$  与  $\mathbf{X}_{\text{mid}}$  适应度函数的差值。本算法通过初始温度  $\theta_0$ 、终止温度  $\theta_{\text{end}}$ 、退温系数  $\lambda$  来控制算法进程。该局部搜索方法提高了局部搜索能力,但也增加了时间计算成本,因此采用变温度的模拟退火算法以缩短计算时间,当前温度为

$$\theta = R_{\text{DOWN}}(\theta_0(1-t/(T+1))) \quad (28)$$

采用变温度的模拟退火算法,前期可提高算法的全局搜索能力,后期可加快算法的收敛,从而在总体上提高该算法的计算效率。

混合离散蝙蝠算法中,响度  $A_{i,t}$  和脉冲率  $r_{i,t}$  同传统蝙蝠算法一样也要随着迭代过程进行更新:

$$A_{i,t} = \alpha A_{i,t-1} \quad (29)$$

$$r_{i,t} = r_{i,0}(1-e^{-\gamma}) \quad (30)$$

其中,  $\gamma$  为脉冲频度的增加系数;  $\alpha$  为一个恒量,对于任何  $0 < \alpha < 1$  都有

$$\left. \begin{aligned} A_{i,t} &\rightarrow 0 \\ r_{i,t} &\rightarrow r_{i,0} \end{aligned} \right\} \quad (31)$$

$r_{i,t}$  直接影响个体执行局部搜索的概率,  $A_{i,t}$  决定局部搜索中解的偏移程度,初始时刻个体的响度较大,脉冲率较小;随着迭代的不断推进,响度不断减小,脉冲率不断增大,从而使算法结果逐渐最优化。

HDBA 算法的伪代码如下:

Begin

1. 初始化。设置迭代次数  $t=1$ ; 初始化蝙蝠种群的位置  $\mathbf{X}_i$ 、速度  $\mathbf{v}_i$ 、频率  $f_i$ 、响度  $A_i$ 、 $\alpha$ 、 $\gamma$ 、初始温度  $\theta_0$ 、终止温度  $\theta_{\text{end}}$ 、退温系数  $\lambda$ ;
2. 计算初始解  $\text{fitness}(\mathbf{X}) = 1/f(\mathbf{X})$
3. while  $t < \text{Max Generation do}$   
排列蝙蝠并找到最佳  $\mathbf{X}^*$ ;  
for  $i=1:N$  (所有节点数)

```
fi = μ(fmax, fmin);
vi,t = φ(Xi,t-1, X*, fi, Mi,t);
Xi,t = ω(Xi,t-1, vi,t);
if(rand > r) then
    Xmid = { InsertionFun(Xe, |RUP(εAt)| | εAt ≥ 0
           { ExchangeFun(Xe, |RDOWN(εAt)| | εAt < 0
           if(fitness(Xmid) > fitness(Xe) || rand <
exp(-Δθ'/θ)) then
    Xe = Xmid;
    End if
    End if
    if(rand < A & fitness(Xmid) < fitness(Xe))
then
    X* = Xe;
    fitness(Xmid) = f(Xe);
    Ai,t = αAi,t-1;
    ri,t = ri,0(1 - e-n);
    End if
    t = t + 1;
    End for
End while
可视化结果
End
```

2.3 MUS 算法

协同调度问题中,每个运输批次内的配送顺序问题都可以看作是一个是车辆路径规划问题,可采用 LI 等<sup>[10]</sup> 针对库存一路径问题所修改的 MUS 算法求解,具体的伪代码如下:

```
Begin
    初始化。设置迭代次数 t=1,配送路径中目标点的数目为 M, Li 为配送路径中的任意一个需求点;
    While(t ≤ 最大迭代次数) do
        i = 0;
        While(i < M) do
            Li+1 为运输路径中需求点 Li 的下一个物料需求点,找出 Li+1 周边 u 个相邻点,并从中随机选择一个目标点作为 Lj;
            Li-1 是运输路径中需求点 Li 的上一个物料需求点,找出 Li-1 周边 u 个相邻点,并从中随机选择一个目标点作为 Lk;
            如图 5 所示,从当前的配送路线中删除物料需求点 Li;删除 Li 后,在重新生成的路径中找出距离目标点 Li 最近的 2 个点 Lp 和 Lq;
            点 Lp+1 表示运输路径中需求点 Lp 的下一个物料需求点,找出 Lp+1 周边 m 个相邻点,并从中随机选择一个目标点作为 Lz;
            如图 6 所示,将点 Li 重新插入到当前路径中;
```

```
i = i + 1
    计算 AGV 到达各产线上料点时间 bkj;
    End while
    t = t + 1
    End while
```

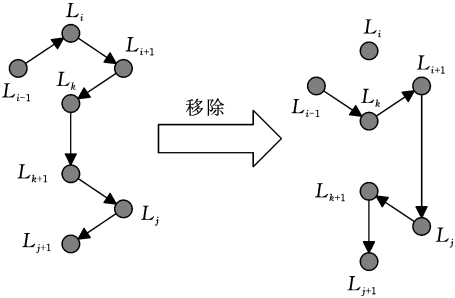


图 5 点的移除操作  
Fig.5 Point remove operation

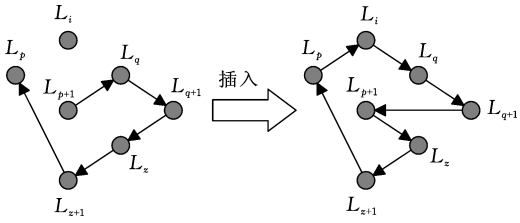


图 6 点的插入操作  
Fig.6 Point insertion operation

图 7 为基于 HDBA + MUS 算法的实现流程图。

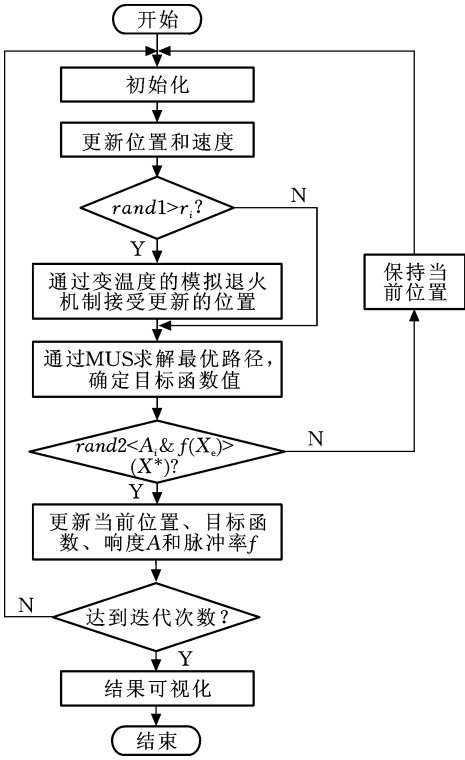


图 7 基于 HDBA + MUS 算法流程图  
Fig.7 Flow chart of algorithm based on HDBA + MUS

3 仿真研究

拟采用仿真实验来验证混合离散蝙蝠算法的有效性。实验对象选取某厨具生产企业自动化生产车间。该车间生产加工主要由两部分组成：辊淋涂批加工工序主要用于对原材料铝片表面上色；多条拉伸成形产线主要用于将铝片拉伸成锅。原材料在经过辊淋涂工序后由 AGV 送到各成形段需求产线进行加工。本仿真所研究的目的是求得其最优协同调度方案，从而使该生产系统具有较高的生产效率。仿真平台为 MATLAB7.11.0。各项基本数据如下：产线 A~F 的订单数量分别为 4、4、5、5、4、5；各订单大小在 1、2、3 中随机出现，在辊淋涂段加工时间在  $[0.1\text{ h}, 0.6\text{ h}]$  区间内随机出现；辊淋涂段加工最大批量为 10；产线 A~F 的加工时间分别为 0.15 h、0.2 h、0.1 h、0.15 h、0.05 h、0.1 h；AGV 数量为 3；每台 AGV 最大运输批量为 7。AGV 在各产线间行驶时间如表 1 所示。

表 1 AGV 各产线行驶时间

Tab.1 AGV running time of each production line h

	辊淋涂段	产线 A	产线 B	产线 C	产线 D	产线 E	产线 F
辊淋涂段	0	0.10	0.13	0.18	0.18	0.21	0.17
产线 A	0.10	0	0.05	0.07	0.08	0.12	0.10
产线 B	0.13	0.05	0	0.05	0.09	0.10	0.10
产线 C	0.18	0.07	0.05	0	0.02	0.10	0.10
产线 D	0.18	0.08	0.09	0.02	0	0.05	0.05
产线 E	0.21	0.12	0.10	0.10	0.05	0	0.07
产线 F	0.17	0.10	0.10	0.10	0.05	0.07	0

为证明 HDBA + MUS 算法的有效性，选取解决协同调度问题中具有代表性的离散粒子群算法(MBPSO)算法<sup>[11]</sup>与 Ullrich-GA 算法<sup>[12]</sup>进行比较，迭代次数均为 200，种群规模均为 50。为了简化计算，通过蝙蝠算法求解调度问题时，多将  $f_{\max}$ 、 $f_{\min}$  设置在 0~3 之间<sup>[13]</sup>，为了增加种群多样性，避免过早收敛设置  $f_{\max} = 3$ ， $f_{\min} = 0$ ， $\alpha = \gamma = 0.9$ <sup>[14-15]</sup>；采用求解车间内部调度问题的模拟退火操作中的参数值  $\theta_0 = 10$ ， $\theta_{\text{end}} = 0.1$ ， $\lambda = 0.9$ 。由于缩短订单的生产周期对协同调度问题来说至关重要，因此文献[16]均将其作为目标函数。针对后工序具有加工特性的协同调度模型的首要目标是尽量缩短后工序队列等待时间<sup>[17]</sup>，考虑到最小化最大完工时间及各订单在产线前的队列等待时

间对本文所研究的问题均十分重要，且具有相同量纲，设置目标函数权重  $w_1 = w_2 = 1$ 。

编码初始化采用随机全排列。为了对 HDBA + MUS 算法的收敛性进行分析，在经过 200 次迭代后得到 HDBA + MUS、MBPSO、Ullrich-GA 的仿真收敛曲线，见图 8。

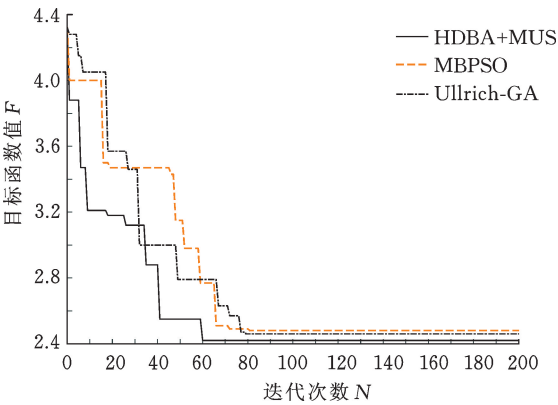


图 8 仿真收敛曲线  
Fig.8 Simulation convergence curve

从收敛曲线可以看出，HDBA + MUS 的收敛能力要强于 MBPSO 与 Ullrich-GA。这是因为 MBPSO 只选择适应度前 20% 的粒子进行随机两两交叉，同时其基于适应度变化的变异操作对平均适应度要求较高，一旦平均适应度偏差较大就会造成一部分较好的值跳不出局部最优。Ullrich-GA 在解决跨工序协同调度问题时，只单纯地对生产调度与车辆路径规划进行随机编码而没有考虑两者之间的联系，可能导致每辆 AGV 配送开始时间较晚，进而延迟了订单的完工。HDBA + MUS 在继承蝙蝠算法优秀的局部搜索能力的同时，通过与模拟退火操作相结合极大地提高了全局搜索能力，它在对染色体进行编码与解码时协同考虑了生产调度与车辆路径规划的关系，提高了可行解的适应度。

表 2 给出了混合离散蝙蝠算法所生成的具体最优生产配送调度方案。 $T$ 、 $T'$  分别为此次配送的开始时间与返回到批加工工序的时间， $V$  为所用 AGV 编号， $C_1$  为各个订单在批加工工序完工时间， $C_2$  为各个订单的到达各产线时间， $C_3$  为各个订单的成形段加工完工时间，可由整体配送顺序方案得到批加工工序生产顺序， $\Omega_{\beta,\gamma}$  表示产线  $\Omega$  生产的第  $\beta$  个订单位于批加工工序批次  $\gamma$  中， $\Omega = A, B, \dots, F$ 。  $C_1[F_{1,1}]$  表示产线 F 所安排生产的第 1 个订单在批加工工序所处批次 1 的完工时间，箭头连接表示这个配送批次里各订单的配送顺序，如  $F_{1,1} \rightarrow D_{1,1} \rightarrow E_{1,1} \rightarrow C_{1,1}$  所示， $C_2[F_{1,1}]$  表示该

订单配送到产线 F 的时间,  $C_3[F_{1,1}]$  表示该订单在产线 F 加工完成的时间。

表 2 最优生产配送调度方案

Tab.2 Optimal production and distribution scheduling scheme

批次 1	$T=0.1; T'=0.65; V=1$ $F_{1,1} \rightarrow D_{1,1} \rightarrow E_{1,1} \rightarrow C_{1,1}$ $C_1[F_{1,1}] = C_1[D_{1,1}] = C_1[E_{1,1}] = C_1[C_{1,1}] = 0.1;$ $C_2[F_{1,1}] = 0.27; C_2[D_{1,1}] = 0.32; C_2[E_{1,1}] = 0.37;$ $C_2[C_{1,1}] = C_3[F_{1,1}] = 0.47; C_3[D_{1,1}] = 0.62; C_3[E_{1,1}] = 0.42; C_3[C_{1,1}] = 0.67$
批次 2	$T=0.3; T'=0.83; V=2$ $F_{2,1} \rightarrow A_{1,1} \rightarrow D_{2,2}$ $C_1[F_{2,1}] = C_1[A_{1,1}] = 0.1; C_1[D_{2,2}] = 0.3; C_2[F_{2,1}] = 0.47; C_2[A_{1,1}] = C_3[F_{2,1}] = 0.57; C_2[D_{2,2}] = 0.65;$ $C_3[A_{1,1}] = 0.87; C_3[D_{2,2}] = 1.1$
批次 3	$T=0.3; T'=0.7; V=3$ $B_{1,2} \rightarrow F_{3,2}$ $C_1[B_{1,2}] = C_1[F_{3,2}] = 0.3; C_2[B_{1,2}] = 0.43; C_2[F_{3,2}] = 0.53; C_3[B_{1,2}] = 1.03; C_3[F_{3,2}] = 0.87$
批次 4	$T=0.7; T'=1.35; V=1$ $C_{2,3} \rightarrow E_{2,3} \rightarrow B_{2,3} \rightarrow D_{3,3}$ $C_1[C_{2,3}] = C_1[E_{2,3}] = C_1[B_{2,3}] = C_1[D_{3,3}] = 0.7;$ $C_2[C_{2,3}] = 0.88; C_2[E_{2,3}] = 0.98; C_2[B_{2,3}] = C_3[C_{2,3}] = C_3[E_{2,3}] = 1.08; C_2[D_{3,3}] = 1.17; C_3[B_{2,3}] = 1.18;$ $C_3[D_{3,3}] = 1.47$
批次 5	$T=1.2; T'=1.81; V=2$ $F_{4,3} \rightarrow C_{3,3} \rightarrow D_{4,3} \rightarrow B_{3,4} \rightarrow C_{4,4}$ $C_1[F_{4,3}] = C_1[C_{3,3}] = C_1[D_{4,3}] = 0.7; C_1[B_{3,4}] = C_1[C_{4,4}] = 1.2; C_2[F_{4,3}] = 1.37; C_2[C_{3,3}] = C_3[F_{4,3}] = 1.47; C_2[D_{4,3}] = 1.49; C_2[B_{3,4}] = 1.58; C_2[C_{4,4}] = 1.63; C_3[C_{3,3}] = 1.57; C_3[D_{4,3}] = 1.64; C_3[B_{3,4}] = 2.18; C_3[C_{4,4}] = 1.73$
批次 6	$T=1.2; T'=1.69; V=3$ $A_{4,2} \rightarrow E_{4,3} \rightarrow F_{5,4} \rightarrow A_{3,4}$ $C_1[A_{4,2}] = C_1[E_{4,3}] = C_1[F_{5,4}] = C_1[A_{3,4}] = 1.2;$ $C_2[A_{4,2}] = 1.3; C_2[E_{4,3}] = 1.42; C_2[F_{5,4}] = 1.49;$ $C_2[A_{3,4}] = 1.59; C_3[A_{4,2}] = 1.6; C_3[E_{4,3}] = 1.47;$ $C_3[F_{5,4}] = 1.69; C_3[A_{3,4}] = 1.75$
批次 7	$T=1.8; T'=2.44; V=1$ $C_{5,5} \rightarrow D_{5,5} \rightarrow A_{4,5} \rightarrow B_{4,5} \rightarrow E_{4,5}$ $C_1[C_{5,5}] = C_1[D_{5,5}] = C_1[A_{4,5}] = C_1[B_{4,5}] = C_1[E_{4,5}] = 1.8; C_2[C_{5,5}] = 1.98; C_2[D_{5,5}] = 2;$ $C_2[A_{4,5}] = 2.08; C_2[E_{4,5}] = 2.23; C_3[C_{5,5}] = 2.18;$ $C_3[D_{5,5}] = 2.15; C_3[A_{4,5}] = 2.23; C_3[E_{4,5}] = 2.38;$ $C_3[E_{4,5}] = 2.33$
目标函数	$\min(\max C_j) = 2.38; \min \sum_{i=1}^n E_i = 0.1;$ $w_1 \min(\max C_j) + w_2 \min \sum_{i=1}^n E_i = 2.48$

4 结论

(1)建立了考虑后工序加工特性的跨工序生产与配送协同调度模型。该模型以最小化最大完工时间和产线前等待时间为目标,假设前工序采用批加工,后工序多产线并行加工,两工序之间通过有限运输能力的 AGV 配送,且产线前存在队列缓冲时间。

(2)设计了混合离散蝙蝠算法。该算法在继承标准蝙蝠算法的基础上针对离散问题进行了改进,并与模拟退火算法相融合,提高了全局寻优能力和收敛效率,并嵌套 MUS 算法以解决车辆路径规划问题。与 MBPSO 算法及 Ullrich-GA 算法相比,本文算法极大缩短了产品的生产周期和生产系统中的队列等待时间。

在下一阶段,可以进一步考虑车间实际情况将产线负荷平整化,以及各工序扰动性等因素进行分析研究。

参考文献:

[1] MAGGU P L, DAS G. On  $2 \times n$  Sequencing Problem with Transportation Times of Jobs[J]. Pure and Applied Mathematica, Science, 1980,129(12): 1-6.

[2] ARMENTANO V A, SHIGUEMOTO A L, LØKKETANGEN A. Tabu Search with Path Relinking for an Integrated Production-distribution Problem[J]. Computers & Operations Research, 2011,38(1):320-327.

[3] 方伯芑, 孙林夫. 不确定环境下产业链生产与配送协同调度优化[J]. 计算机集成制造系统, 2018, 24(1):224-244.

FANG Bopeng, SUN Linfu. Coordinated Scheduling Optimization of Production and Distribution of Industrial Chain under Uncertain Environment[J]. Computer Integrated Manufacturing System, 2018, 24(1): 224-244.

[4] 李凯, 王明星. 单机多车情形生产与配送协同调度算法[J]. 计算机集成制造系统, 2014, 20(12): 3011-3019.

LI Kai, WANG Mingxing. Coordinated Scheduling Algorithm of Production and Distribution in the Case of Single Machine and Multi-vehicle[J]. Computer Integrated Manufacturing System, 2014, 20(12):3011-3019.

[5] 关静, 唐立新. 工件带有温降的生产与前运输协调调度问题[J]. 系统工程学报, 2007, 22(6):639-643.



- GUAN Jing, TANG Lixin. Inbound Transportation and Production Coordinated Scheduling Problem with Temperature Reduction Jobs [J]. Journal of Systems & Management, 2007, 22(6):639-643.
- [6] 马文琼, 王凯. 两阶段装配流水车间加工与配送协同调度研究[J]. 工业工程与管理, 2016, 21(6): 103-117.
- MA Wenqiong, WANG Kai. Coordinated Scheduling Problem for Two-stage Assembly Flowshop Production and Distribution[J]. Industrial Engineering and Management, 2016, 21(6):103-117.
- [7] 刘玲. 单机器生产与车辆路径协同调度问题建模与算法研究[D]. 武汉:华中科技大学, 2016.
- LIU Ling. Model and Algorithms for the Integrated Single Machine Scheduling and Vehicle Routing Problem[D]. Wuhan: Huazhong University of Science and Technology, 2016.
- [8] 卓雪雪. 批处理机环境下两阶段集成调度算法研究[D]. 合肥:安徽大学, 2018.
- ZHUO Xuexue. Research on Batch Scheduling Processing on Parallel Machines with Two-stage Integrated Scheduling [D]. Hefei: Anhui University, 2018.
- [9] YANG X S. A New Metaheuristic Bat-inspired Algorithm[M]//Nature Inspired Cooperative Strategies for Optimization (NICSO2010). Berlin:Springer, 2010:65-74.
- [10] LI K, CHEN B, SIVAKUMAR A I, et al. An Inventory-routing Problem with the Objective of Travel Time Minimization[J]. European Journal of Operational Research, 2014, 236(3):936-945.
- [11] 薛梅, 周志平. 批处理机环境下生产与两阶段运输协同调度问题研究[J]. 中国管理科学, 2016, 24: 22-28.
- XUE Mei, Zhou Zhiping. Coordinated Scheduling Problem of Production and Two-stage Transportation with a Batch-processing Machine[J]. Chinese Journal of Management Science, 2016, 24:22-28.
- [12] ULLRICH C A. Integrated Machine Scheduling and Vehicle Routing with Time Windows[J]. European Journal of Operational Research, 2013, 227(1): 152-165.
- [13] 尹建津, 张贝克. 改进蝙蝠算法解决 FFSP 问题及其应用研究[J]. 计算机工程应用, 2019, 55(9): 243-247.
- YIN Jianjin, ZHANG Beike. Improved Bat Algorithm for Solving Flexible Flow Shop Scheduling Problem and Its Application[J]. Computer Engineering and Applications, 2019, 55(9):243-247.
- [14] 徐华, 张庭. 混合离散蝙蝠算法求解多目标柔性作业车间调度[J]. 机械工程学报, 2016, 52(18): 201-212.
- XU Hua, ZHANG Ting. Hybrid Discrete Bat Algorithm for Solving the Multi-objective Flexible JobShop Scheduling Problem[J]. Journal of Mechanical Engineering, 2016, 52(18):201-212.
- [15] 韩忠华, 朱伯秋. 基于改进蝙蝠算法的柔性流水车间排产优化问题研究[J]. 计算机应用研究, 2017, 34(7): 1935-1938.
- HAN Zhonghua, ZHU Boqiu. Study for Flexible Flow Shop Scheduling Problem Based on Advanced Bat Algorithm[J]. Application Research of Computer, 2017, 34(7): 1935-1938.
- [16] 刘玲, 李昆鹏. 生产和运输协同调度问题的模型和算法[J]. 工业工程与管理, 2016, 21(2):86-91.
- LIU Ling, Li Kunpeng. Model and Algorithms for the Integrated Production and Transportation Scheduling[J]. Industrial Engineering and Management, 2016, 21(2): 86-91.
- [17] 李修琳, 鲁建厦. 基于混合遗传算法的混流混合车间协同调度问题[J]. 中国机械工程, 2012, 23(8):935-940.
- LI Xiulin, LU Jiansha. Hybrid Genetic Algorithm for Mixed-model Hybrid-shop Scheduling Problem [J]. China Mechanical Engineering, 2012, 23(8): 935-940.

(编辑 张 洋)

作者简介:鲁建厦,男,1963年生,教授、博士研究生导师。研究方向为精益生产、智能制造。发表论文 99 篇。E-mail: ljs@zjut.edu.cn.