控制与决策
Control and Decision

离散蝙蝠算法在三阶段装配流水线调度问题的应用

王艺霖， 郑建国

引用本文:
王艺霖, 郑建国. 离散蝙蝠算法在三阶段装配流水线调度问题的应用[J]. 控制与决策, 2021, 36(9): 2267−2278.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2020.0054

离散蝙蝠算法在三阶段装配流水线调度问题的应用

王艺霖†, 郑建国

(东华大学 旭日工商管理学院，上海200050)

摘 要: 为了解决三阶段装配流水线调度问题,提出一种改进的离散型蝙蝠算法(DBA). 针对所提问题的瓶颈期,提出下限理论,改进三阶段瓶颈期的下限公式,并引入调度模型生成初始种群,重新划分蝙蝠的捕食范围(HR),通过捕食行为、迁移行为的改进提高局部搜索能力,以有效提高离散蝙蝠算法的性能. 改进K-means聚类算法,将具有最高相似性的蝙蝠进行分组,缩短计算时间,加快算法收敛速度. 通过对不同规模实例的仿真实验与对比分析,对机器、产品和组的数量进行测试,验证了DBA的总体性能比其他算法更优;在算法的有效性和解的质量方面,通过对动态控制参数、DHR和精英策略的改进,有效地增强了算法的搜索能力.

关键词: 离散型蝙蝠算法；三阶段装配流水线调度；K-means聚类算法；精英策略

中图分类号: C93 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2020.0054 **开放科学(资源服务)标识码(OSID):**

引用格式: 王艺霖,郑建国. 离散蝙蝠算法在三阶段装配流水线调度问题的应用[J]. 控制与决策, 2021, 36(9): 2267-2278.

Discrete bat algorithm in three-stage assembly flowshop scheduling problem

WANG Yi-lin†, ZHENG Jian-guo

(Glorious Sun School of Business and Management，Donghua University，Shanghai 200050，China)

Abstract: In order to solve the three-stage assembly flowshop scheduling problem, this paper proposes an improved discrete bat algorithm (DBA). Aiming at the bottleneck period of the question, this paper proposes the lower limit theory and improves the lower limit formula of the three-stage bottleneck period. At the same time, a scheduling model is introduced to generate the initial population, and the bat's hunting range (HR) is re-divided. Through the improvement of predation behavior and migration behavior, the local search ability of the algorithm is improved, and the performance of the discrete bat algorithm is effectively improved. The K-means clustering algorithm is improved to group the bats with the highest similarity, shortening the calculation time and speed up the algorithm convergence speed. Through simulation experiments and comparative analysis of examples of different scales, the number of machines, products and sets is tested, and the overall performance of the DBA is verified to be better than other algorithms; in terms of the effectiveness of the algorithm and the quality of the solution, the improvements for the dynamic control parameters, DHR and elite strategy effectively enhance the algorithm's search capabilities.

Keywords: discrete bat algorithm; three-stage assembly flowshop problem; K-means clustering algorithm; elite strategy

0 引 言

在过去的几十年中,装配流水线问题(assembly flowshop problem, AFSP)已成为国内外学者深入研究的调度优化问题之一. AFSP在不同的行业领域均有应用,包括制造业[1]、化工[2]、纺织业[3]等. 因此,对三阶段装配流水线调度问题(3sAFS)进行研究具有重要的工程意义. 同时,由于3sAFS本身具有强NP-hard[4],对其研究亦具有较高的学术价值.

国内外学者已经广泛研究了两阶段装配流水线的一般情况,其中第1级具有不同优化标准的m个相同的并联机器. 之前的研究涉及完工时间[5-6]、总完工时间[7]和迟到[8]等研究目标. Koulamas等[4]将两阶段问题扩展为三阶段装配问题,同时推广了串行三机流水车间问题和两阶段装配流水线调度问题,具有较强的NP-hard,考虑了三阶段装配流水线调度问题,分析了几种启发式方法的最坏情况比率,指出启发式算法逼近最优. Hatami等[9]考虑了一个三阶段的带有两个目标的装配流水线调度问题,即平均

收稿日期: 2020-01-11；**修回日期:** 2020-05-08.
基金项目: 东华大学博士研究生创新基金项目(18D310804).
†通讯作者. E-mail: 15800351523@163.com.

流动时间和最大延误,提出了模拟退火和禁忌搜索算法解决随机产生的许多测试问题;Maleki等[10]研究了具有有限缓冲和设置时间的三阶段装配流水线,提出了用于解决此类问题的禁忌搜索. 此后,Maleki等[11]讨论了最小化工作时间和工作平均完成时间的加权总和,提出了变邻域搜索(VNS)算法和禁忌搜索. Campos等[12]提出了结合迭代贪婪策略的Elitist非主导排序遗传算法(NSGA-II)解决三阶段装配流水线调度问题. 针对某种三阶段装配流水线调度问题,即具有序列相关设置时间的三阶段装配流程调度问题(TSAFSP_SDST), Li等[13]提出了一种自适应混合估计分布算法(AHEDA)来最小化加权和,引入了基于插入的邻域搜索提高局部搜索能力. Yang等[14]研究了分布式装配置换流水车间问题(DAPFSP),通过考虑实际约束条件(包括无等待、无怠速和到期日,在此后标记为DAPFSP-T)研究了一类关于最小化总延迟的复杂DAPFSP. Komaki等[15]提出了布谷鸟优化算法解决三阶段装配流水线调度问题.

大多数调度问题都是NP-hard的,如B&B和动态规划等精确方法已经无法在合理的计算时间内找到最优解[4]. 因此,启发式和元启发式算法被应用于解决装配流水线调度问题,如模拟退火[9]、禁忌搜索[9-12]、遗传算法[12]、粒子群算法[16]、路径重新链接[17]以及基于生物启发的元启发式算法(如布谷鸟搜索[15]和蝙蝠算法[18])等.

本文针对三阶段装配流水线调度问题的改进,提出了一种新颖的离散型蝙蝠算法,其创新点如下:

1) 为了解决三阶段装配流水线调度的瓶颈期问题,提出下限理论,改进三阶段瓶颈期的下限公式;

2) 引入调度模型生成初始种群,并重新划分蝙蝠的捕食范围(HR),通过捕食行为、迁移行为的改进提高局部搜索的能力,同时有效地提高离散蝙蝠算法的性能;

3) 根据最高相似性对蝙蝠进行种群分组,改进K-means聚类算法,将具有最高相似性的蝙蝠进行分组,缩短计算时间,提高离散蝙蝠算法的效率,引入精英策略,提高离散蝙蝠算法的局部搜索能力,防止过早收敛.

1 3sAFS问题的数学描述

假设两阶段装配流水线从第1阶段到第2阶段收集和运输了m个组件,其所需的时间忽略不计. 实际生产中,具有多工厂设备和最终装配工厂的生产系统是不现实的[19-20],其中收集过程、运输过程和车辆的装卸过程都需要损耗时间成本. 因此在文献[4]的基础上,本文研究三阶段装配流水线问题,即收集阶段(具有m个并行的相同机器)、运输阶段和组装阶段. 在收集完成一项工件的所有零件后,运输车以预设顺序访问每一个并行机器,加载产品组件/零件;车辆将零部件运输至组装机,在等待区域完成卸载操作;最后在组装阶段将组件安装在一起.

三阶段装配流水线问题的数学描述可以表示为工件$J_j\,(j=1,2,\dots,n)$由操作$\{O_{i,j},\dots,O_{m,j}\}$、$O_{T,j}$和$O_{A,j}$组成. 收集操作$O_{ij}$由机器$M_i(i=1,2,\dots,m)$处理,需要$p_{ij}$个时间单位,机器$M_{ij}$一次最多可以处理一个工件;运输操作$O_{T,j}$在机器$M_T$上处理,耗用$p_{T,j}$个时间单位;组装操作$O_{A,j}$在机器$M_A$上处理,需要$p_{A,j}$个时间单位. 其问题是NP-hard,假设为:

1) 产品的所有组件/零件在零时间可用,且其处理时间是预先已知的.

2) 不允许抢占,即一旦组件/零件已经开始处理,机器必须完成该过程后才可以进入下一阶段,且过程不允许中断. 操作$O_{T,j}$仅在完成所有并行操作$O_{1,j},\dots,O_{m,j}$之后才开始,同时操作$O_{A,j}$仅在$O_{T,j}$操作完成之后才可以开始.

3) 所有机器都可在整个调度范围内使用,无需细分,运输阶段M_T可以一次处理最多一个工件的组件,同样地,组装机器M_A可以一次组装最多一个工件的组件.

4) 一旦机器空闲,就开始处理可用的组件/产品.

5) 所有机器一次可以且只可以处理一个组件/零件. 对于任何i和$k, i=1,2,\dots,m, k=1,2,\dots,m, i\neq k$,操作$O_{ij}$和$O_{kj}(j=1,2,\dots,n)$允许同时处理.

基于以上假设,令$C_{A,j}$为M_A上J_j(操作$O_{A,j}$)的完成时间. 目标是最小化最大完成时间(完工时间)$C_{\max}=\max C_{A,j}$. 对于调度问题,将第1阶段m个机器的三阶段装配流水线问题表示为$\mathrm{AF}(m,1,1)//C\max$,其公式为

$$C_{\max}=\max_{1\leqslant k_1\leqslant k_2\leqslant n}\left\{\max_{1\leqslant i\leqslant m}\sum_{k=1}^{k_1}p_{ij}+\sum_{k=k_1}^{k_2}p_{Tj}+\sum_{k=1}^{n}p_{Aj}\right\}. \tag{1}$$

目标是找到最小化完工时间$\min(C\max)$的产品序列. $\mathrm{AF}(m,1,1)//C_{\max}$问题概括了$F_3//C_{\max}$串行三机流水车间问题和$\mathrm{AF}(m,1)//C_{\max}$两级装配流水线调度问题;三阶段装配问题$\mathrm{AF}(m,1,1)//C_{\max}$可以表示为增加了收集和运输过程的两阶段装配问题$\mathrm{AF}(m,1)//C_{\max}$,或允许第1阶段并行操作

的$F_3//C_{\max}$或$F_2//C_{\max}$与$\mathrm{AF}(m,1,1)//C_{\max}$的混合模型.

2　下限和调度模型

本节描述了文献[4]提出的启发式算法H_0和H_3,进而提出下限(lower bound)和调度模型(dispatching rules). 启发式算法H_0和H_3的最差性能比率分别为2和$2-1/m+\varepsilon$.

1) 启发式算法H_0[4].

step 1: 将Johnson算法应用于$F_2//C_{\max}$问题,处理时间为$J_j(j=1,2,\ldots,n)$,即第1阶段为$p_{T,j}$、第2阶段为$p_{A,j}$. 令$\pi_0=(J_1,J_2,\ldots,J_n)$为已知的产品序列.

step 2: 使用置换π_0为$\mathrm{AF}(m,1,1)//C\max$问题构建时间序列$S_{H_0}$,运行时间为$O(n\log n)$.

2) 启发式算法H_3[4].

step 1: 给出一个实例$\mathrm{AF}(m,1,1)//C\max$,定义相应的$F_3//C_{\max}$问题,工件$J_j(j=1,2,\ldots,n)$的处理时间为$(p_{il}+p_{i2}+\ldots+p_{im})/m$(第1阶段)、$p_{T,j}$(第2阶段)和$p_{A,j}$(第3阶段).

step 2: 设H是$F_3//C_{\max}$问题的任意启发式算子,将H应用于step 1生成的问题,由此产生产品序列$\pi_3=(j_1,j_2,\ldots,j_n)$.

step 3: 使用置换p_{i3}为$\mathrm{AF}(m,1,1)//C_{\max}$问题构建时间序列$S_{H_3}$.

2.1　下　限

在调度问题中,瓶颈期的平均工作量最大、完工时间最长,导致工期停滞,因此本节提出了一个基于瓶颈期的下限(lower bound, LB),根据每个阶段工作量也可以分为3个阶段.

当第1阶段为瓶颈时,式(1)中$k_1=k_2=n$,此时公式演变为

$$\mathrm{LB}_1=\max_{1\leqslant i\leqslant m}\sum_{j=1}^{n}p_{ij}+\min_{1\leqslant j\leqslant n}p_{Tj}+\min_{1\leqslant j\leqslant n}p_{Aj}. \quad (2)$$

第2阶段为瓶颈,即第1阶段结束之前进入瓶颈期. 第1阶段和第3阶段的最短完工时间最小,式(1)中$k_1=1$且$k_2=n$,此时公式为

$$\mathrm{LB}_2=\min_{1\leqslant j\leqslant n}\{\max_{1\leqslant i\leqslant m}p_{ij}\}+\sum_{j=1}^{n}p_{Tj}+\min_{1\leqslant j\leqslant n}\{p_{Aj}\}. \quad (3)$$

装配阶段为瓶颈期,此时的第1、第2阶段达到最佳状态,完工时间最小,式(1)中$k_1=k_2=1$,则

$$\mathrm{LB}_3=\min_{1\leqslant j\leqslant n}\{\max_{1\leqslant i\leqslant m}p_{ij}\}+\min_{1\leqslant j\leqslant n}p_{Tj}+\sum_{j=1}^{n}p_{Aj}. \quad (4)$$

综上分析,下限的公式为

$$\mathrm{LB}=\max\{\mathrm{LB}_1,\mathrm{LB}_2,\mathrm{LB}_3\}. \quad (5)$$

2.2　调度模型

基于瓶颈期的下限,为了方便查找每个阶段的产品序列,引入产品索引并提出调度模型(dispatching model, DM),按照非递减顺序对产品索引的处理时间进行排序,其模型为

$$\mathrm{DM}_1=\max_{1\leqslant i\leqslant m}\{p_{ij}\}, \quad (6)$$

$$\mathrm{DM}_2=\max_{1\leqslant j\leqslant n}\{p_{Tj}\}, \quad (7)$$

$$\mathrm{DM}_3=\max_{1\leqslant j\leqslant n}\{p_{Aj}\}, \quad (8)$$

$$\mathrm{DM}=\mathrm{DM}_1+\mathrm{DM}_2+\mathrm{DM}_3. \quad (9)$$

DM_1, DM_2和DM_3分别基于第1、第2和第3阶段的产品处理时间,而DM是每个产品所有处理时间的总和,因此举例说明LB的性能和调度规则.

根据文献[4]的示例,第1阶段有m台机器,两个产品的处理时间如下:

产品1: $p_{1,j}=1, p_{1,T}=k, p_{1,A}=1/k$;

产品2: $p_{2,j}=k, p_{2,T}=1, p_{2,d}=2/k$.

其中: $j=1,2,\ldots,m, k>2$,最优调度为$C_{\max}=k+2+2/k$. 这种情况下,有

$$\mathrm{LB}_1=\max\{1+k\}+\min\{k+1/k,1+2/k\}=k+2+2/k,$$

$$\mathrm{LB}_2=\min\{1,k\}+k+1+\min\{1/k,2/k\}=k+2+1/k,$$

$$\mathrm{LB}_3=\min\{1+k,k+1\}+(1/k+2/k)=k+1+3/k,$$

$$\mathrm{LB}=\max\{\mathrm{LB}_1,\mathrm{LB}_2,\mathrm{LB}_3\}=\max_{\{k+2+2/k,k+2+1/k,k+1+3/k\}}=k+2+2/k.$$

此外, DM_1将1分配给产品1,将k分配给产品2,因此产品序列为1-2,这与最优解相同. 基于DM_2的产品序列为2-1,有效期为$2k+1+1/k$[4]. 同样,基于DM_3和DM的产品序列为1-2,这与最优解相同.

根据这些示例,可以认为LB_1相对于其他下限具有更好的性能, DM_1优于其他调度规则.

3　改进的离散型蝙蝠算法

3.1　蝙蝠算法的问题描述

蝙蝠算法[21-22]是在理想状态下,利用蝙蝠在觅食时所发出的脉冲的频率f、响度A_i、脉冲发射率r_i的变化而模拟设计出的一种新型群体智能算法.

通过频率f的调整可以引起波长λ的变化,波长λ的大小与蝙蝠所捕食猎物的大小相一致,从而可

定位目标. 频率越高,波长和行程的距离越短. 一般频率f在$[f_{\min}, f_{\max}]$范围内变化,并对应于波长范围$[\lambda_{\min}, \lambda_{\max}]$. 蝙蝠按脉冲发射率$r_i \in [0,1]$和响度$A_i$发出声波脉冲. 蝙蝠在定位过程中发现目标时会增加脉冲发射率,减小响度,逼近目标,从而捕食猎物.

对于蝙蝠i,假设其位置为x_i,速度为v_i. 在D维空间中,蝙蝠i在t时刻的位置为x_i^t、速度为v_i^t,有

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \tag{10}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i, \tag{11}$$

$$x_i^t = x_i^{t-1} + v_i^t. \tag{12}$$

其中x_*是在t时刻的当前全局最优解. 在局部搜索阶段,一旦当前最优解选中了一个解,那么每只蝙蝠按照随机游走法产生局部新解,即

$$x_{\text{new}}(i) = x_{\text{old}} + \varepsilon A^t. \tag{13}$$

其中:x_{old}为从当前最优解集中随机选择的一个解,$\varepsilon \in [-1,1]$为一个服从均匀分布的随机数,$A^t = \langle A_i^t \rangle$为$t$时刻蝙蝠群体的平均响度,$N$为蝙蝠数量.

在迭代过程中,响度A_i、脉冲发射率r_i也会发生改变. 当蝙蝠越来越接近猎物时,其响度便会降低,脉冲发射率会增加,公式为

$$A_i^{t+1} = \alpha A_i^t, \tag{14}$$

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)], \tag{15}$$

其中α和γ为常数. 这里,α类似于模拟退火中冷却时间的冷却因子,对于任意$0 < \alpha < 1$, $\gamma > 0$,有

$$\text{当} t \to \infty \text{时},\ A_i^t \to 0,\ r_i^t \to r_i^0. \tag{16}$$

响度A_i^0的初始值通常为[1,2],由式(15)发射率的初始值r_i^0趋近于0或$r_i^0 \in [0,1]$的任意值,只有在可行解不断改进的情况下,响度和发射率才会更新,此时趋于最优解.

3.2 离散型蝙蝠算法

由于标准蝙蝠算法不能直接应用于离散解空间中的优化问题,也不适合排列问题,有必要对标准连续蝙蝠算法进行改进,以处理三阶段装配流水线调度问题的优化. 为此,提出了一个新的离散型蝙蝠算法,即DBA,对其进行例如捕食行为、觅食后的种群聚类和基于离散解表示的蝙蝠迁移行为. DBA中的迁移行为和捕食机制有效地提高了全局搜索和局部搜索能力,这些特征机制通过定向和随机移动增强了算法的性能. DBA算法流程如图1所示.

3.2.1 初始化栖息地

为了解决调度问题,蝙蝠算法必须是离散型,生成大小为$N_{\text{pop}} \times n$的矩阵形式的初始候选栖息地,

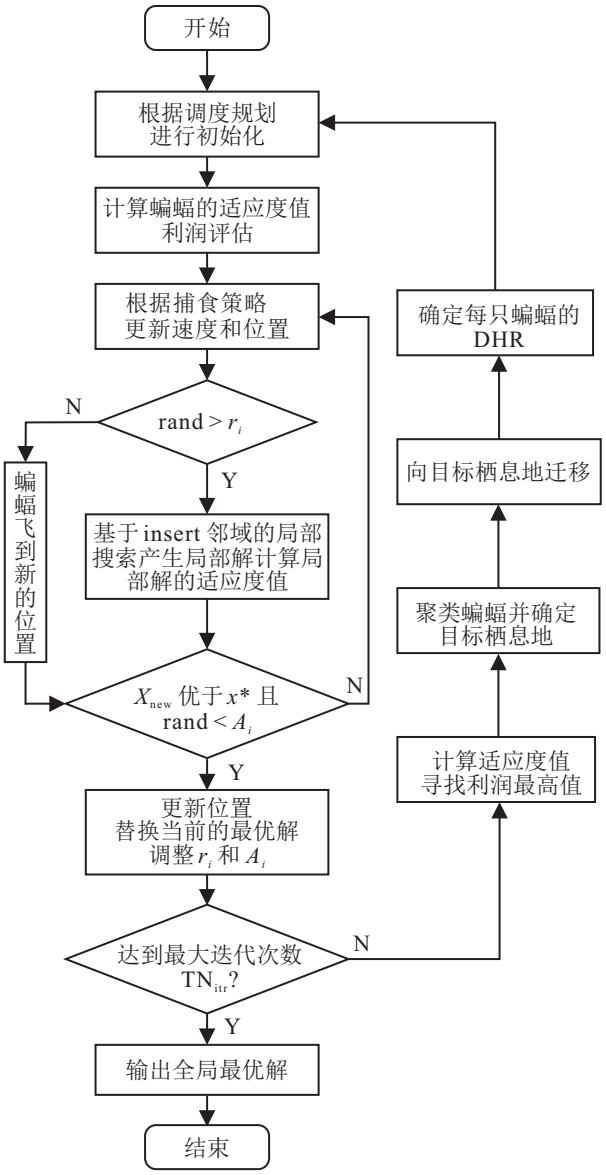

图1 DBA算法流程

其中N_{pop}和n分别为初始种群的大小和产品的数量. 为了生成高质量的初始解,采用调度问题中置换工件的编码表示,但是属于同一产品的零件不会与其他产品混合,因此编码也隐含产品的顺序. 由调度模型随机生成其余的序列,索引顺序,根据编码将产品及其所有零件分配给工期最短的工厂. 例如,解由4,3,1,2,8,6,7,5编码,零件的顺序为4,3,1,2,8,6,7,5,产品的顺序为2,1,4,3,因此将产品2及其零件4和3分配给工厂1,将产品1及其零件1和2分配给工厂2,将产品4及其零件8、6和7分配给工厂2,将产品3及其零件5分配给工厂1. 具体示意见图2.

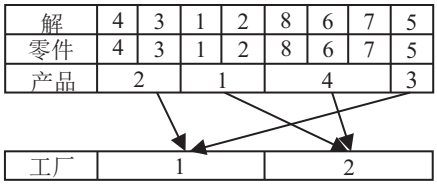

图2 编码示意图

3.2.2 利润评估

与遗传算法的染色体类似,在蝙蝠算法中,栖息地(habitat)表示问题的解,其质量由利润表示. 利润(适应度)评估的目标是计算栖息地关于目标函数的良好性,即测量提高捕食率的解的适用率.

由于蝙蝠算法试图最大化利润函数,目标函数为最小化完工时间,利润函数与其相应的目标函数具有相反的关系. 每个栖息地/解的利润值可以计算如下:

$$\text{profit}(\text{habitat}) = -\text{makespan}(\text{habitat}). \tag{17}$$

其中每个解的完工时间由前文计算可得,利润值越高,栖息地的表现越好,因此,具有较高适应值的候选栖息地有更多机会成为可行解.

3.2.3 捕食策略

定义一个新的概念:蝙蝠在当前的栖息地范围内捕食觅食,该范围或最大距离称为“捕食半径”(hunting radius, HR),它取决于猎物的数量(N_{kc})、猎物的总数(TN_{kc})以及问题变量的上限和下限(U_{var}、L_{var}),有

$$\text{HR} = \alpha \times \frac{N_{\text{kc}}}{\text{TN}_{\text{kc}}} \times (U_{\text{var}} - L_{\text{var}}). \tag{18}$$

在蝙蝠识别出栖息地后,每个个体开始在捕食半径(HR)内随机地捕食小型猎物,称为“捕食机制”,这种机制将会提高局部搜索的能力和速度. 由此本节提出在问题解(栖息地)上应用局部搜索的邻域结构,基于其利润值的蝙蝠种群可以捕获一定数量的猎物N_{kc},计算如下:

$$N_{\text{kc}} = w_i\Big(\frac{f_i}{f_{\text{best}}}\Big). \tag{19}$$

其中:$w_i \sim U(L_{\text{kc}}, U_{\text{kc}})$,$L_{\text{kc}}$、$U_{\text{kc}}$分别为迭代过程中蝙蝠捕食数量的上限和下限.

由于解决问题的蝙蝠算法是离散型,要求兼容解空间和解的适应度,引入“动态捕食半径(dynamic hunting radius, DHR)”的概念,与三阶段装配流水线问题中解的大小(在该问题中为产品的数量n)、猎物的总数、捕获猎物的数量和动态控制参数α_{itr}成比例,其公式为

$$\text{DHR} = \alpha_{\text{itr}} \times n \times \frac{N_{\text{kc}}}{\text{TN}_{\text{kc}}}, \tag{20}$$

其中α_{itr}为控制参数. 较大的α_{itr}会更好地提高全局优化能力,同时DBA的计算时间将会增加. 因此,为了维持两者的平衡,引入动态α_{itr}值而不是常数值,这意味着α_{itr}会随着时间获得更好的收敛、提高DBA的全局搜索能力,并避免陷入早期收敛. 为此,定义一个线性变化机制,如模拟退火(SA)中的冷却函数[9],即在迭代中得出α_{itr}的值,其公式为

$$\alpha_{\text{itr}} = \alpha_{\max} - N_{\text{itr}}\frac{\alpha_{\max} - \alpha_{\min}}{\text{TN}_{\text{itr}}}. \tag{21}$$

其中:α_min和α_max分别为最小、最大极值,N_{itr}和TN_{itr}分别为迭代次数和总迭代次数.

同时,为了执行捕食策略,引入解空间的邻域结构. 文献[13, 23]指出,插入移动比交换移动更适合领域结构. 如2.2节和3.2.1节所述,本文使用的解表示基于产品的索引序列,因此在邻域结构中只考虑插入移动k-insert算子,$k \leqslant$DHR,随机选择k个产品并重新插入随机位置. 使用k-insert算子,使得现有的栖息地迭代为当前蝙蝠在其DHR内的新解.

3.2.4 识别类型

蝙蝠利用超声波辨别目标猎物的位置和速度,在当前捕食半径范围内进行觅食. 在觅食过程中,蝙蝠需要识别容易捕获的小型猎物,并避开大型猎物转向小型猎物,或者直接放弃该路径继续寻找新的猎物. 通常,$P\%$大型猎物将被蝙蝠识别并避开,其余的小型猎物会被捕获,这个概率通常为10%.

3.2.5 迁移、分组和评估

新栖息地旨在拥有更多的食物来源,需要蝙蝠不断地进行移动和飞行,这种行为称为“迁移行为”. 在迁移之前,蝙蝠在不同地域组成种群,利润最高的种群则为迁移的目标点. 蝙蝠在进行觅食后,很难识别属于哪个种群,因此使用K-means聚类方法[24-25]对蝙蝠进行重新分组,通常设置[3, 5]即可;计算每组的平均利润,确定迁移的目标点.

在迁移过程中,蝙蝠只能经过某些路径,且与当前栖息地到目的地的直接路径有些许偏差. 如,蝙蝠飞过当前栖息地到目的地的直接路径的概率为$\lambda\%$,偏差为ψ,则$\lambda \sim U(0,1)$,$\psi \sim (-w, w)$,$\lambda \sim U(0,1)$,$\psi \sim (-w, w)$,其中U为0、1之间的均匀分布,$w = \pi/6$是合理的偏差,可以模拟蝙蝠的迁移过程. 在迁移过程完成并确定新的栖息地后,蝙蝠可以根据DHR规则随机进行捕获和觅食行为.

为了对觅食后的蝙蝠种群进行分组,结合DBA算法的特性对K-means聚类算法[26]进行改进,使用相似性度量分组觅食后的蝙蝠,将具有最高相似性的蝙蝠分组. 因此,在每对蝙蝠x与y之间定义新的相似系数$\text{csc}(x, y)$,有

$$\text{csc}(x, y) = \Big(\sum_{y=1}^{n}\sum_{x=1}^{m} S_{ij} + 2Z_y^x\Big)\Big/n. \tag{22}$$

其中:n为产品数量,$\sum_{y=1}^{n}\sum_{x=1}^{m} S_{ij}$为解$x$与$y$之间相似的对部分(边)的数量,$Z_y^x$为两个解之间排序串中相

同元素的数量或者在同一位置的数量. 步骤如下.

step 1: 建立一个相似矩阵, 其元素为所有$\csc(x,y)$的值.

step 2: 确定k个差异最大可行解或最小的csc值作为k聚类中心.

step 3: 将新的个体i加到种群集合$A(A \in \{1,2,\ldots,k\})$中, 并求得$\max\sum\csc(x,y)$.

step 4: 如果有未分组的个体, 则转至step 3, 否则结束.

选择具有最高利润价值的栖息地作为蝙蝠迁移的目标点, 每个组的利润如下:

$$\text{groupprofit} = a \times \text{最优解的利润值} + b \times \text{平均利润}.$$

这些利润的最大值确定了目标群体, 因此其栖息地是迁移蝙蝠的新目的地.

3.2.6 精英策略

在蝙蝠算法中, $\text{NC}_{\max}$是一个控制解空间和限制环境中最大个体数量的参数, 假设只有$\text{NC}_{\max}$数量的蝙蝠生存下来, 才具有更好的利润价值, 由此, 根据利润值提出概率P_e为在种群中选择不是精英并且必须灭亡的个体. 忽略标记为精英的最佳蝙蝠N_e的一组设置为S_e, 如果没有被概率P_e选中消亡, 则可以存活, 有

$$P_e(\xi_i) = \frac{\text{profit}(1/\xi_i)}{\sum\limits_{j\neq S_e}\text{profit}(1/\xi_j)}. \tag{23}$$

该函数不仅可以避免算法在早期迭代中过早收敛, 而且可以在一定程度上保留解空间的随机性. 依据文献[4-5, 27], 在算法的每次迭代中对精英解进行η插入移动, 可以改善初始序列, 接受所得到的解. 最终, 若达到最大迭代次数后算法没有任何改进则终止.

4 仿真实验与结果

4.1 参数设置

蝙蝠算法(BA)[21]和本文提出的离散型蝙蝠算法(DBA)参数如表1所示, N_e和NI_{itr}仅适用于DBA. 其中DBA的参数N_e为当前种群中的精英解决方案的数量, 其应该被保存用于下一次迭代, 因此定义为种群的5%.

基于随机生成的实例测试算法的性能, 产品处理时间列于表2, 其中组1、2和3来自文献[4], 组4中所有阶段的处理时间适合于相同的范围. 此外, 第1阶段的机器数量定义为2、4、6、8, 产品数量为60、120、180、240. 运行64个实例, 每个实例包含100个生成的测试问题.

表1 算法参数及其最佳值

参数	范围	BA	DBA
初始种群大小	(2, 4, 5, 6)	5	5
最大允许蝙蝠数量($\text{NC}_{\max}$)	(20, 40, 60)	40	40
下限(L_{kc})	–	5	5
上限(U_{kc})	–	20	20
识别率(P_e)	–	10%	10%
当前种群中精英解数量(N_e)	(3%, 5%, 7%)	–	5%
α_{itr}的下限($\alpha_{\min}$)	(0.3, 0.4, 0.5)	0.5	0.5
α_{itr}的上限($\alpha_{\max}$)	(1.7, 2, 2.2)	1.7	2
迭代次数(NI_{itr})	(3, 5, 7)	–	5
最大迭代次数(TN_{itr})	–	400	400

表2 处理不同阶段产品的时间区间

问题	第1阶段p_{ij}	第2阶段p_{Tj}	第3阶段p_{Aj}
组1	[0, 100]	[0, 10]	[0, 100]
组2	[0, 100]	[0, 50]	[100, 200]
组3	[100, 200]	[0, 10]	[0, 100]
组4	[0, 100]	[0, 100]	[0, 100]

4.2 与其他算法对比

将DBA与禁忌搜索(SA)[10]、变邻域搜索算法(VNS)[11]和文献[4]提出的启发式算法进行比较, 所有算法均在Matlab 7.5.0中编程, 并在具有2.66 GHz Intel Core 2 Duo处理器和3 GB RAM内存的PC上运行. 由于元启发式算法的随机性, 每个组运行25次.

4.2.1 最优解的偏差计算DVL

最优解偏差DVL计算如下:

$$\text{DVL} = \frac{x^* - \text{LB}_i}{x^*} \times 100\%. \tag{24}$$

其中: LB_i为下限, x^*为DBA算法的最优值. 基于产品数量和机器数量的LB偏差最优解如表3所示.

表3 基于产品数量和机器数量的LB偏差最优解

产品数量					机器数量				
n	LB_1	LB_2	LB_3	LB	m	LB_1	LB_2	LB_3	LB
60	17.46	69.74	21.86	0.48	2	18.64	68.89	19.72	0.66
120	18.13	70.54	22.97	0.54	4	18.17	68.37	19.36	0.68
180	18.17	68.91	20.5	0.46	6	17.67	68.97	20.57	0.59
240	17.97	68.79	20.08	1.02	8	17.52	68.52	20.62	0.57
平均值	**17.932 5**	69.495	21.352 5	**0.625**	平均值	**18**	68.69	20.07	**0.625**

由表 3 可见, 下限 LB 的总体性能为 0.625 %, 表明 LB 的提出会对 3sAFS 问题的整体性能产生积极的作用, 同时, 其偏差最优解中 LB1 的平均值最优. 由表 3 机器数量部分可知, 基于机器数量的 LB_1 性能最佳, 而 LB_2 性能最差, 此时下限的平均值与产品数量平均值相同, 再次验证了下限 LB 提高了 3sAFS 问题的整体性能.

4.2.2　相对百分比偏差 RPD

为了比较算法之间的效率, 计算相对百分比偏差 (relative percentage deviations, RPD), 将 DBA 与其他算法进行对比, 有

$$\mathrm{RPD} = \frac{x^* - \mathrm{LB}}{\mathrm{LB}} \times 100\,\%. \tag{25}$$

由此可见, RPD 越低算法性能越好. 算法平均 RPD 如表 4 所示. 由表 4 可得各个算法的 RPD 值, DBA 的平均值为 0.568 %, 明显低于包括原蝙蝠算法在内的其他算法的平均值, 表明 DBA 算法的效率高于其他元启发式算法; BA 表现次之, 平均 RPD 为 0.53 %. 此外, 平均值为 0.792 % 的 VNS 明显优于 SA 和其他 4 个调度模型. 与其他调度模型相比, DM_1 的性能最好, 并优于 H_0; DM_3 的平均值高达 15.798 %, 表明该调度模型的效率最低、性能最差.

对最优解的平均频率 (frequency of resulting in the best solution, FBS) 进行仿真实验, 每个元启发式算法运行 25 次, 公式如下:

$$\text{FBS的平均值} = [\text{FBS的总数}/\text{总运行次数}],$$

$$\text{总运行次数} = 100 \times 25,$$

其中 $[X]$ 表示大于实数 X 的最小整数.

算法平均 FBS 如表 5 所示. 由表 5 可见, DBA 具有最高的平均频率, DM_3 具有最低的平均频率, 表明 DBA 性能最好, 同时验证了表 4 关于算法性能的结论. 此外, BA 和 VNS 的平均频率与 DBA 相对接近, 其中 VNS 具有比 BA 和 SA 更好的平均频率.

4.2.3　非参数统计检验

对比结果为统计分析中的平均值、方差和参数测试, 如方差分析 (ANOVA)[28]. 自 2007 年起, 非参数统计分析也逐渐用于算法的结果分析, 基于此, 本节使用参数或非参数统计测试分析所涉及算法在优化问题上的结果. 为了检测算法之间的差异是否具有统计学意义, 对算法的 RPD 结果进行方差分析, 从而确认结果的有效性, 同时判定哪种算法最优. 为了使用参数测试, 有必要检查 3 个条件: 残差的独立性、正态性和同方差性[29]. 用 Kolmogorov-Smirnov 测试[30] 所得结果是否满足正态性检验, 该测试的 P 值 (sig.) 等于零并且明显小于显著性水平 (0.05), 因此所得的结果不是正态分布, 这种情况下不能使用参数测试 (方差分析), 应使用非参数测试.

作为测试样本是否来自相同分布的非参数检验方法, 采用 Kruskal-Wallis 单因素方差进行分析[31], 其结果如图 3 所示. Kruskal-Wallis 检验零假设的假设样本来自于相同的组, 而备选假设的假设样本属于不同的组, 预测来自种群的平均样本等级是相同的. 最小值的等级为 1, 下一个最小值的等级为 2, 依此类推. 由于 Kruskal-Wallis 是非参数统计检验, 不可能以统计方式检验 DBA、VNS 和 SA 算法是否具有显著不同, 即该测试仅考虑所有组, 当检验的零假设被拒绝时, 表示至少一种算法随机支配至少一种其他算法.

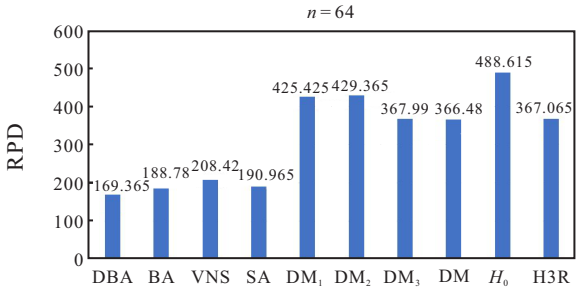

图 3　Kruskal-Wallis 算法排序

根据图 3 可知, DBA 在所有算法中为第 1 等级, 而 DM 在它们中产生最差的结果. DBA、BA、VNS 和 SA 相对于其他算法具有相当低的平均等级. 此外, H3R、DM_1 和 H_0 在某种程度上具有相同的平均等级. 通过定向和随机移动增强的迁移策略和精英机制有助于蝙蝠算法在局部搜索和全局搜索中更有效地进行搜索, 这与局部搜索机制相比提供了更多机会改进可行解.

如果 Kruskal-Wallis 检验的结果不显著, 则没有证据表明样本之间存在随机优势. 在应用 Kruskal-Wallis 检验后, P 值 (sig.) 等于零, 因此零假设被拒绝, 这意味着所采用的算法性能之间存在统计上显著的差异. 拒绝这种差异归因于随机抽样的想法, 并且可以得出群体具有不同分布的结论.

图 4 ~ 图 6 为基于机器、产品和组的数量测试所有算法的性能结果. 可以看出, DBA 和 DM_3 分别在所有其他算法中具有最佳和最差性能. 图 4 和图 5 中, 基于产品数量和机器数量的算法 DBA、BA、SA 和 VNS 均比其他几个算法更优, 其中 DBA 算法在产品数量和机器数量上体现最优. 图 4 第 2 组和第 3 组算法性能很突出, 但第 1 组和第 4 组性能发生了显著变化.

表 4 算法平均RPD

序号	n	m	机器	对比算法									
				DBA	BA	SA	VNS	DM_1	DM_2	DM_3	DM	H_0	H3R
1	60	2	1	0.096	0.248	0.496	0.42	1.23	5.08	23	11.91	3.51	1.21
2	60	2	2	0.007	0.009	0.013	0.01	0.11	0.51	0.71	0.11	0.51	0.51
3	60	2	3	0.008	0.009	0.008	0.008	0.53	0.59	1.07	1.04	0.06	0.45
4	60	2	4	1.294	1.962	2.57	2.04	3.75	22.83	27.3	14.28	17.52	10.94
5	60	4	1	0.093	0.172	0.38	0.26	2.53	5.04	22.56	17.54	3.15	0.95
6	60	4	2	0	0	0	0	0.09	0.38	0.61	0.13	0.38	0.38
7	60	4	3	0	0	0	0	0.51	0.55	1.06	1.04	0.04	0.47
8	60	4	4	1.798	1.954	2.261	2.09	4.47	22.07	25.07	17.83	16.09	10.33
9	60	6	1	0.126	0.188	0.339	0.28	6.81	4.61	22.54	20.13	3.06	1
10	60	6	2	0	0.002	0.002	0.003	0.1	0.26	0.54	0.13	0.26	0.26
11	60	6	3	0	0.002	0.002	0.002	0.52	0.56	1.06	1.05	0.03	0.54
12	60	6	4	0.63	0.755	1.98	0.85	15.01	22.44	24.46	18.71	16.03	10.9
13	60	8	1	0.042	0.116	0.22	0.19	2.83	4.33	21.27	19.42	2.7	0.79
14	60	8	2	0	0.001	0.004	0.002	0.1	0.19	0.51	0.14	0.19	0.19
15	60	8	3	0	0.006	0.004	0.003	0.51	0.54	1.05	1.04	0.03	0.46
16	60	8	4	0.498	0.759	1.763	1.14	5.08	22.01	24.88	19.12	15.85	10.41
17	120	2	1	1.113	1.141	1.581	1.576	5.267	11.547	29.76	18.28	8.987	3.88
18	120	2	2	0	0	0.083	0.068	0.307	1.533	2.28	0.48	1.533	1.533
19	120	2	3	1.013	1.032	0.056	0.056	2.04	2.453	3.933	3.693	0.933	1.76
20	120	2	4	3.653	4.627	7.467	7.343	11.867	31.893	37.52	24.813	21.16	13.547
21	120	4	1	1.48	1.596	0.992	0.813	8.24	12.267	29.053	23.147	9.947	3.56
22	120	4	2	0	0	0.011	0.012	0.347	1	1.92	0.627	1	1
23	120	4	3	0	0	0.012	0.015	2.04	2.013	3.84	3.747	0.733	1.747
24	120	4	4	4.012	4.816	5.836	5.604	26.187	30.453	36.533	28.547	19.973	13.627
25	120	6	1	1.631	1.644	0.947	0.912	6.907	11.293	29.147	25.733	8.907	2.853
26	120	6	2	0	0	0.003	0.011	0.36	0.707	1.667	0.467	0.707	0.707
27	120	6	3	0	0	0.012	0.012	2.013	1.96	3.813	3.707	0.76	1.48
28	120	6	4	3.118	2.273	5.227	4.6	14.107	31.093	37.347	30.773	19.72	13.933
29	120	8	1	1.127	0.215	0.619	0.493	8.12	10.027	27.907	25.92	7.52	2.387
30	120	8	2	0	0	0.012	0.011	0.48	0.453	1.573	0.427	0.453	0.453
31	120	8	3	0	0	0.009	0.008	1.96	1.933	3.773	3.707	0.68	1.493
32	120	8	4	2.994	3.047	4.683	4.267	15.213	30.987	36.573	30.76	20.613	14.36
33	180	2	1	0.128	0.331	0.661	0.56	1.64	6.773	30.667	15.88	4.68	1.613
34	180	2	2	0.009	0.011	0.017	0	0.147	0.68	0.947	0.147	0.68	0.68
35	180	2	3	0.011	0.011	0.011	0.011	0.707	0.787	1.427	1.387	0.08	0.6
36	180	2	4	1.725	2.616	3.427	2.72	5	30.44	36.4	19.04	23.36	14.587
37	180	4	1	0.124	0.229	0.507	0.347	3.373	6.72	30.08	23.387	4.2	1.267
38	180	4	2	0	0	0	0	0.12	0.507	0.813	0.173	0.507	0.507
39	180	4	3	0	0	0	0	0.68	0.733	1.413	1.387	0.053	0.627
40	180	4	4	2.397	2.605	3.015	2.787	5.96	29.427	33.427	23.773	21.453	13.773
41	180	6	1	0.168	0.251	0.452	0.373	9.08	6.147	30.053	26.84	4.08	1.333
42	180	6	2	0	0.003	0.003	0.004	0.133	0.347	0.72	0.173	0.347	0.347
43	180	6	3	0	0.002	0.003	0.003	0.693	0.747	1.413	1.4	0.04	0.72
44	180	6	4	0.84	1.007	2.64	1.133	20.013	29.92	32.613	24.947	21.373	14.533
45	180	8	1	0.056	0.155	0.293	0.253	3.773	5.773	28.36	25.893	3.6	1.053
46	180	8	2	0	0.001	0.005	0.003	0.133	0.253	0.68	0.187	0.253	0.253
47	180	8	3	0	0.008	0.005	0.004	0.68	0.72	1.4	1.387	0.04	0.613
48	180	8	4	0.664	1.012	2.351	1.52	6.773	29.347	33.173	25.493	21.133	13.88
49	240	2	1	0.14	0.21	0.477	0.32	3.027	5.333	30.813	15.307	3.6	1.147
50	240	2	2	0.005	0.009	0.016	0.005	0.12	0.52	0.733	0.133	0.52	0.52
51	240	2	3	0.004	0.005	0.004	0.004	0.52	0.667	1.08	1.04	0	0.507
52	240	2	4	1.347	1.787	2.819	1.893	4.4	30.827	34.787	17.8	23.24	14.12
53	240	4	1	0.12	0.187	0.413	0.293	2.28	5.693	30.933	24.333	3.52	1
54	240	4	2	0	0	0	0	0.12	0.413	0.707	0.147	0.413	0.413
55	240	4	3	0	0	0	0	0.52	0.56	1.08	1.053	0.04	0.48
56	240	4	4	1.309	1.548	2.204	1.827	18.2	28.413	33.88	21.613	21.187	13.667
57	240	6	1	0	0.055	0.344	0.12	2.867	5.627	30.187	26.56	3.667	0.973
58	240	6	2	0	0	0.005	0.005	0.107	0.267	0.533	0.147	0	0.267
59	240	6	3	0	0.009	0.005	0.004	0.627	0.56	1.067	1.053	0.027	0.453
60	240	6	4	1.453	1.58	2.156	1.84	5.373	29.147	32.573	23.547	21	13.933
61	240	8	1	0.093	0.127	0.235	0.147	2.96	5.307	29.867	27.373	3.107	0.947
62	240	8	2	0	0	0	0.005	0.107	0.227	0.56	0.147	0.227	0.227
63	240	8	3	0	0	0	0	0.493	0.627	1.067	1.053	0	0.44
64	240	8	4	1.013	1.207	2.089	1.427	20.213	30.067	33.307	25.427	21.96	14.76
	平均值/%			**0.568**	0.649	0.965	0.792	4.226	9.175	15.798	11.667	6.428	4.037

表 5　算法平均FBS

序号	n	m	机器	DBA	BA	SA	VNS	DM_1	DM_2	DM_3	DM	H_0	H3R
1	60	2	1	95	91	84	88	22	0	0	0	13	5
2	60	2	2	96	95	95	96	44	4	1	31	4	4
3	60	2	3	100	99	100	100	1	0	0	0	51	4
4	60	2	4	87	68	74	79	0	0	0	0	0	0
5	60	4	1	100	92	76	91	2	0	0	0	16	2
6	60	4	2	100	99	99	100	44	6	2	33	6	6
7	60	4	3	100	100	100	100	2	0	0	0	45	5
8	60	4	4	88	81	72	85	0	0	0	0	0	0
9	60	6	1	93	89	81	90	4	0	0	0	14	2
10	60	6	2	100	99	99	100	34	5	0	27	5	5
11	60	6	3	100	100	100	100	2	0	0	0	57	5
12	60	6	4	83	60	62	76	0	0	0	0	0	1
13	60	8	1	94	79	85	94	2	0	0	0	21	6
14	60	8	2	100	100	100	100	43	15	1	29	15	15
15	60	8	3	100	99	100	100	1	0	0	0	55	4
16	60	8	4	86	84	71	77	0	0	0	0	0	0
17	120	2	1	96	88	53	61	21	1	0	0	9	4
18	120	2	2	100	96	93	97	51	11	4	42	11	11
19	120	2	3	96	90	90	93	6	1	0	0	48	9
20	120	2	4	100	85	66	72	2	0	0	0	1	4
21	120	4	1	88	84	60	63	6	0	0	0	14	8
22	120	4	2	100	100	100	100	54	15	4	36	15	15
23	120	4	3	100	100	100	100	2	2	0	0	49	9
24	120	4	4	97	96	77	82	1	0	0	0	1	3
25	120	6	1	90	83	51	82	7	0	0	0	16	10
26	120	6	2	100	100	100	100	53	12	4	35	12	12
27	120	6	3	100	100	100	100	5	1	0	0	54	10
28	120	6	4	99	86	83	93	0	0	0	0	2	4
29	120	8	1	85	89	75	85	3	1	0	0	16	12
30	120	8	2	100	100	100	100	43	21	5	32	21	21
31	120	8	3	100	100	100	100	3	1	0	0	54	8
32	120	8	4	93	80	71	75	0	0	0	0	0	3
33	180	2	1	100	88	83	87	22	1	0	0	11	5
34	180	2	2	98	97	96	97	45	7	3	37	7	7
35	180	2	3	100	99	99	100	3	0	0	0	50	7
36	180	2	4	97	81	81	86	1	0	0	0	0	1
37	180	4	1	98	90	81	91	3	0	0	0	17	6
38	180	4	2	100	100	100	100	51	9	2	38	9	9
39	180	4	3	100	100	100	100	2	0	0	0	47	7
40	180	4	4	92	82	81	89	0	0	0	0	0	0
41	180	6	1	97	87	66	86	4	0	0	0	19	6
42	180	6	2	100	100	100	100	46	6	2	31	6	6
43	180	6	3	100	100	100	100	2	0	0	0	61	5
44	180	6	4	95	72	71	84	0	0	0	0	0	1
45	180	8	1	100	90	89	96	2	1	0	0	21	6
46	180	8	2	100	100	100	100	48	15	2	29	15	15
47	180	8	3	100	100	100	100	4	0	0	0	56	6
48	180	8	4	98	85	72	78	0	0	0	0	0	1
49	240	2	1	100	100	94	98	0	0	0	0	0	0
50	240	2	2	100	100	100	100	40	7	0	31	7	7
51	240	2	3	100	100	100	100	2	0	0	0	51	0
52	240	2	4	87	69	62	78	0	0	0	0	0	0
53	240	4	1	100	87	81	89	12	0	0	0	18	4
54	240	4	2	100	100	100	100	35	5	0	33	5	5
55	240	4	3	100	100	100	100	2	0	0	0	61	5
56	240	4	4	91	72	67	81	1	0	0	0	0	1
57	240	6	1	99	97	88	95	3	0	0	0	15	3
58	240	6	2	100	100	100	100	45	9	1	26	9	9
59	240	6	3	100	87	100	100	0	0	0	0	52	4
60	240	6	4	89	86	79	84	1	0	0	0	0	0
61	240	8	1	97	81	78	87	3	0	0	0	23	4
62	240	8	2	100	100	100	100	45	11	1	23	11	11
63	240	8	3	100	100	100	100	5	0	0	0	58	7
64	240	8	4	95	93	74	85	0	0	0	0	0	0
	平均值/%			**96.85**	91.54	86.91	91.8	13.7	2.64	0.48	8.16	19.69	5.57

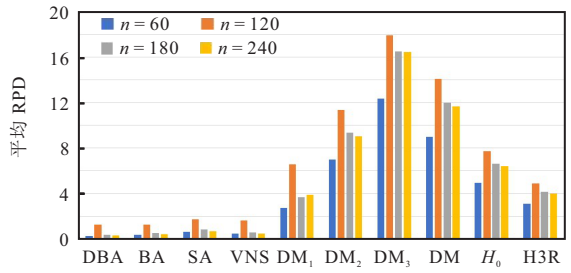

图 4 基于产品数量的算法RPD

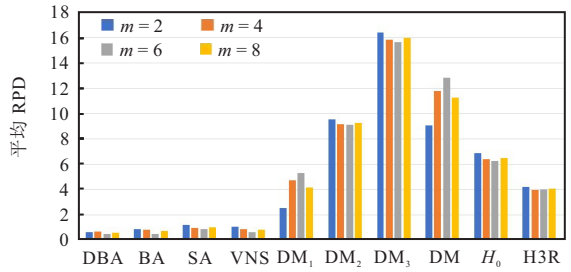

图 5 基于机器数量的算法RPD

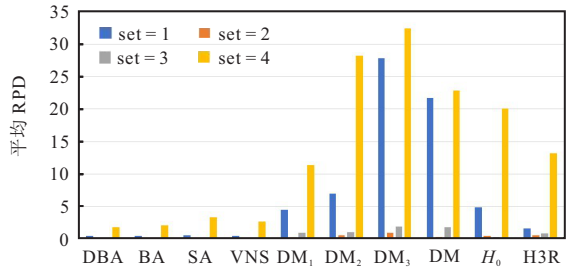

图 6 基于组的算法RPD

4.3 组合特征的性能效果

为了研究“种群分组”“组群评估”“迁移行为”和“精英策略”对DBA的影响,选取表4和表5中问题33～48作为样本,即$n=60$,研究“种群分组”“DHR”和“精英策略”等特征对DBA的影响. 由于这16组数据较大,可以有效区分几个特征的引入对DBA算法局部搜索能力的影响.

仿真实验在有无该特征的情况下进行模拟,如表6所示,所得结果以x/y的形式给出. 其中:x表示平均RPD,y表示计算时间. 种群分组通过对比改进K-means聚类算法和原K-means聚类算法得到结论:改进后聚类算法的平均RPD和计算时间略优于K-means;引入动态控制参数α、步长DHR和该区域的可行解,这些特征在有效性和效率方面增强了算法的搜索能力;精英策略方面,虽舍弃最好的策略使得算法计算速度更快,但会降低算法性能;在计算时间可控的前提下,局部搜索在一定程度上提高了DBA的性能. 由此可知,从种群分组、捕食策略(DHR)、精英策略等方面改进原蝙蝠算法,使得DBA算法提高了局部搜索能力.

为了研究DBA特征之间是否存在统计学差异,ANOVA以0.05的显著性水平进行测试. 表7给出了ANOVA测试结果,其中Y和N分别表示是和否,即Y表示给定的特征存在显著差异,N表示给定特征没有显著差异. 由表7可以得到结论:在“种群分组”技术中,除问题36和45外,其他问题之间彼此不存在显著性差异;在计算时间方面,大多数问题都存在显著性差异. DHR的总体结果中,Y对应于平均RPD,N对应于计算时间. DHR的总体结果是显著的,则(1)和(2),

表 6 基于平均RPD和时间对DBA每项特征的影响

问题				种群分组		DHR (动态α & Q_f)			精英策略		局部搜索	
序号	参数			改进聚类算法	K-means聚类算法	动态控制参数α		(3)可行解	P_e	最优值	有	无
	n	m	组			(1)有Q_f	(2)无Q_f					
33	60	2	1	0.18/25	0.16/27	0.16/25	0.17/26	0.22/27	0.16/25	0.16/25	0.16/25	0.33/23
34	60	2	2	0/27	0.01/30	0.01/27	0.01/28	0.01/28	0.01/27	0.01/26	0.01/27	0.01/26
35	60	2	3	0/35	0.00/38	0.00/35	0.00/36	0.00/36	0.00/35	0.00/33	0.00/35	0.00/32
36	60	2	4	1.69/42	1.66/45	1.79/42	1.94/44	2.15/47	1.79/42	1.82/40	1.79/42	2.04/38
37	60	4	1	0.19/43	0.19/49	0.19/46	0.22/46	0.22/49	0.19/46	0.19/43	0.19/46	0.23/43
38	60	4	2	0/46	0.00/51	0.00/46	0.00/47	0.00/50	0.00/46	0.00/43	0.00/46	0.00/44
39	60	4	3	0/45	0.00/52	0.00/45	0.00/49	0.00/50	0.00/45	0.00/44	0.00/45	0.00/46
40	60	4	4	1.97/47	2.01/55	2.08/48	2.09/48	2.09/55	2.08/48	2.10/46	2.08/48	2.09/51
41	60	6	1	0/34	0.19/35	0.20/34	0.21/34	0.21/37	0.20/34	0.23/32	0.20/34	0.22/32
42	60	6	2	0/36	0.00/44	0.00/36	0.00/37	0.00/39	0.00/36	0.00/35	0.00/36	0.00/33
43	60	6	3	0/38	0.00/45	0.00/38	0.00/40	0.00/44	0.00/38	0.00/36	0.00/38	0.00/34
44	60	6	4	0.64/29	0.77/37	0.79/30	0.83/37	0.84/39	0.79/30	0.81/26	0.79/30	0.80/35
45	60	8	1	0.18/36	0.18/39	0.18/36	0.18/38	0.18/40	0.18/36	0.18/34	0.18/36	0.19/34
46	60	8	2	0/38	0.00/46	0.00/38	0.00/41	0.00/47	0.00/38	0.00/36	0.00/38	0.00/35
47	60	8	3	0/40	0.00/51	0.00/40	0.01/45	0.01/49	0.00/40	0.01/36	0.00/40	0.01/37
48	60	8	4	0.69/42	0.67/60	0.71/42	0.79/46	0.93/57	0.71/42	0.77/37	0.71/42	0.96/38
平均值				**0.35/38**	0.37/44	0.38/38	0.40/40	0.43/43	0.38/38	0.39/36	0.38/38	0.43/36

表 7　对表6实验进行ANOVA测试,显着性水平为0.05

问题序号	种群分组	DHR (α & Q_f)				精英策略	局部搜索
		总体	(1) & (2)	(1) & (3)	(2) & (3)		
33	N/N	Y/N	N/–	Y/–	N/–	N/N	Y/N
34	N/N	N/N	–/–	–/–	–/–	N/N	N/N
35	N/N	N/N	–/–	–/–	–/–	N/N	N/N
36	Y/Y	Y/Y	Y/N	Y/Y	Y/Y	N/N	Y/Y
37	N/N	Y/N	Y/–	Y/–	N/–	N/N	Y/N
38	N/Y	N/N	–/–	–/–	–/–	N/N	N/N
39	N/Y	N/N	–/–	–/–	–/–	N/N	N/N
40	N/Y	N/Y	–/N	–/Y	–/Y	N/N	Y/Y
41	N/N	N/N	–/–	–/–	–/–	Y/N	Y/N
42	N/Y	N/N	–/–	–/–	–/–	N/N	N/Y
43	N/Y	N/Y	–/N	–/Y	–/Y	N/N	N/Y
44	N/Y	Y/Y	Y/Y	Y/Y	N/N	N/Y	N/Y
45	Y/Y	N/N	–/–	–/–	–/–	N/N	Y/N
46	N/Y	N/Y	–/N	–/Y	–/Y	N/N	N/Y
47	N/Y	Y/Y	Y/Y	Y/Y	N/N	N/Y	N/Y
48	N/Y	Y/Y	N/N	Y/Y	Y/Y	Y/Y	Y/Y

(1) 和 (3) 或 (2) 和 (3) 特征中至少有一个具有显著差异; 相反, 如果DHR的总体结果不显著, 则上述成对特征中没有一个是显著的, 用 "–" 表示. 因此, 问题规模的大小、优化特征在求解过程中都具有重要的工件, 决定了算法的计算时间和运作效率.

5　结　论

本文针对三阶段装配流水线调度问题, 对蝙蝠算法进行改进, 提出一种新颖的离散型蝙蝠算法 (DBA). 仿真实验表明:

1) 引入调度模型提高了产品和机器的平均性能, 进而提高了3sAFS问题的整体性能, 由该模型随机生成初始解序列, 根据编码将3sAFS产品及其所有零件分配给工期最短的工厂.

2) 在3sAFS问题上, 基于产品数量、机器数量和组的测试中, 与其他算法相比DBA性能更优.

3) 为了更好地解决种群分组问题, 改进了 K-means聚类算法, 与原 K-means聚类算法相比, 改进算法平均RPD和计算时间略优于 K-means; 动态控制参数 α 的引入优化了捕食策略 (DHR), 在有效性和效率方面增强了算法的搜索能力; 精英策略的提出改进了原蝙蝠算法, 在计算时间可控的前提下提高了DBA的局部搜索能力, 防止过早收敛.

参考文献 (References)

[1] Ma W G, Wang K. Coordinated scheduling problem for two-stage assembly flowshop production and distribution[J]. Industrial Engineering and Management, 2016, 21(6): 103-110.

[2] Bautista J, Alfaro R. Transformation of a mixed model assembly line in a regular flow workshop: Case study at the Nissan factory in Barcelona[J]. Direccion Y Organizacion, 2019, 69: 82-98.

[3] Wang D J, Qiu H X, Wu C C, et al. Dominance rule and opposition-based particle swarm optimization for two-stage assembly scheduling with time cumulated learning effect[J]. Soft Computing, 2019, 23(19): 9617-9628.

[4] Koulamas C, Kyparisis G. The three-stage assembly flowshop scheduling problem[J]. Computers & Operations Research, 2001, 28(7): 689-704.

[5] Allahverdi A, Al-Anzi F S. Evolutionary heuristics and an algorithm for the two-stage assembly scheduling problem to minimize makespan with setup times[J]. International Journal of Production Research, 2006, 44(22): 4713-4735.

[6] Marichelvam M K. An improved hybrid cuckoo search (IHCS) metaheuristics algorithm for permutation flow shop scheduling problems[J]. International Journal of Bio-Inspired Computation, 2012, 4(4): 200-205.

[7] Wu C C, Chen J Y, Lin W C, et al. A two-stage three-machine assembly scheduling flowshop problem with both two-agent and learning phenomenon[J]. Computers & Industrial Engineering, 2019, 130: 485-499.

[8] Chung T P, Chen F. A complete immunoglobulin based artificial immune system algorithm for two-stage assembly flowshop scheduling problem with part splitting and distinct due windows[J]. International Journal of Production Research, 2019, 57(10): 3219-3237.

[9] Hatami S, Ebrahimnejad S, Tavakkoli M R. Two meta-heuristics for three-stage assembly flowshop scheduling with sequence-dependent setup times[J]. International Journal of Advanced Manufacturing Technology, 2010, 50(9/10/11/12): 1153-1164.

[10] Maleki D A, Modiri M, Tavakkoli M R, et al. A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times[J]. Journal of Industrial Engineering International, 2012, 8(1): 1-7.

[11] Maleki D A, Seyedi I. Taguchi method for three-stage assembly flow shop scheduling problem with blocking and sequence dependent set up times[J]. Journal of Engineering Science and Technology, 2013, 8(5), 603-622.

[12] Campos S C, Claudio A, Jose E. NSGA-II with iterated greedy for a bi-objective three-stage assembly flowshop scheduling problem[C]. Proceedings of the 2014 Genetic and Evolutionary Computation Conference. New York: Assco Computing Machinery, 2014: 429-436.

[13] Li Z H, Qian B, Hu R, et al. Adaptive hybrid estimation of distribution algorithm for solving a certain kind of three-stage assembly flowshop scheduling problem[J]. Computer Integrated Manufacturing Systems, 2015, 21(7): 1829-1845.

[14] Yang Y X, Li P, Wang S Y, et al. Scatter search for distributed assembly flowshop scheduling to minimize total tardiness[C]. IEEE Congress on Evolutionary Computation. New York: IEEE, 2017: 861-868.

[15] Komaki G M, Ehsan T, Vahid K. Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem[J]. Computers & Industrial Engineering, 2017, 105: 158-173.

[16] Shahdi-Pashaki S, Teymourian E, Tavakkoli R. New approach based on group technology for the consolidation problem in cloud computing-mathematical model and genetic algorithm[J]. Computational & Applied Mathematics, 2018, 37(1): 693-718.

[17] Shahvari O, Logendran, R. An enhanced tabu search algorithm to minimize a bi-criteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes[J]. Computers & Operations Research, 2017, 77: 154-176.

[18] Sirikarn C, Ponnapa M, Pupong P. A hybrid discrete bat algorithm with krill herd-based advanced planning and scheduling tool for the capital goods industry[J]. International Journal of Production Research, 2019, 57(21): 6705-6726.

[19] Zhang G H, Xing K Y. Differential evolution metaheuristics for distributed limited-buffer flowshop scheduling with makespan criterion[J]. Computers & Operations Research, 2019, 108: 33-43.

[20] Nejati M, Mahdavi I, Hassanzadeh R, et al. Lot streaming in a two-stage assembly hybrid flow shop scheduling problem with a work shift constraint[J]. Journal of Industrial and Production Engineering, 2016, 33(7): 459-471.

[21] Yang X S. A new metaheuristic bat-Inspired algorithm[C]. Nature Inspired Cooperative Strategies for Optimization. Berlin: Springer-Verlag, 2010: 65-74.

[22] Yang X S. Nature-Inpsired metaheursitic algorithms[M]. Frome: Luniver Press, 2008: 1-128.

[23] Grabowski J, Pempera J. Some local search algorithms for no-wait flowshop problem with makespan criterion[J]. Computers & Operations Research, 2015, 32(8): 2197-2212.

[24] Bishnu N, Motoi Y, Hiroya S. Analysis of building electricity use pattern using K-means clustering algorithm by determination of better initial centroids and number of clusters[J]. Energies, 2019, 12(12): 2451-2468.

[25] Yang M S, Sinaga K P. A feature-reduction multiview k-means clustering algorithm[J]. IEEE Access, 2019, 7: 114472-114486.

[26] Jiang Z L, Guo N, Jin Y B, et al. Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing[J]. Information Sciences, 2020, 518: 168-180.

[27] Xiao S Y, Wang W J, Wang H. An improved artificial bee colony algorithm based on elite strategy and dimension learning[J]. Mathematics, 2019, 7(3): 289-306.

[28] Luengo J, García S, Herrera F. A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests[J]. Expert Systems with Applications, 2009, 36(4): 7798-7808.

[29] Ozcelik B, Erzurumlu T. Comparison of the warpage optimization in the plastic injection molding using ANOVA, neural network model and genetic algorithm[J]. Journal of Materials Processing Technology, 2006, 171(3): 437-445.

[30] Filippatos A, Langkamp A, Kostka P, et al. A sequence-based damage identification method for composite rotors by applying the kullback-leibler divergence, a two-sample kolmogorov-smirnov test and a statistical hidden markov model[J]. Entropy, 2019, 21(7): 690.

[31] Maimaiti P, Sen L F, Aisilahong G, et al. Statistical analysis with Kruskal Wallis test for patients with joint contracture[J]. Future Generation Computer Systems — The International Journal of Escience, 2019, 92: 419-423.

作者简介

王艺霖 (1989−), 女, 博士生, 从事智能决策与知识管理的研究, E-mail: 15800351523@163.com;

郑建国 (1962−), 男, 教授, 博士生导师, 从事智能决策与数据挖掘、技术经济分析、进化计算与优化算法等研究, E-mail: zjg@dhu.edu.cn.

(责任编辑：郑晓蕾)