Computer Science and Application 计算机科学与应用, 2024, 14(11), 45-59
Published Online November 2024 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2024.1411215

Hans汉斯

多维度改进的蜣螂优化算法及其应用

郭　兵

天津工业大学软件学院，天津

收稿日期：2024年10月12日；录用日期：2024年11月11日；发布日期：2024年11月20日

摘　要

蜣螂优化算法(DBO)作为一种新兴的智能优化算法，在求解复杂优化问题中显示出巨大潜力。然而，其在收敛精度和易陷入局部最优方面的局限性限制了其应用范围。本文提出了一种多策略改进的蜣螂优化算法(ODBO)，通过佳点集群初始化和周期突变机制增强种群多样性，引入Beta分布动态生成反射解决方案以探索更广的搜索空间，并采用莱维飞行处理边界违规问题。进一步融合鲸鱼优化算法的螺旋搜索更新机制，结合随机策略更新位置，显著提升了算法的收敛精度和鲁棒性。当算法陷入停滞时，引入t分布扰动变异策略，有效提高了算法跳出局部最优解的能力。通过在17个基准测试函数验证了改进策略的有效性。此外，本文还将ODBO应用于车间排产调度问题，进一步证实了其在解决实际工程问题中的有效性和可靠性。

关键词

蜣螂优化算法，局部最优解，鲸鱼优化算法，t分布扰动变异，车间排产调度问题，鲁棒性分析

The Multi-Dimensional Improved Dung Beetle Optimization Algorithm and Its Applications

Bing Guo

School of Software, Tiangong University, Tianjin

Received: Oct. 12th, 2024; accepted: Nov. 11th, 2024; published: Nov. 20th, 2024

Abstract

The Dung Beetle Optimization algorithm (DBO), as an emerging intelligent optimization algorithm,

文章引用: 郭兵. 多维度改进的蜣螂优化算法及其应用[J]. 计算机科学与应用, 2024, 14(11): 45-59.
DOI: 10.12677/csa.2024.1411215

shows great potential in solving complex optimization problems. However, its limitations in convergence precision and susceptibility to local optima hinder its broader application. This paper proposes an improved multi-strategy Dung Beetle Optimization algorithm (ODBO), which enhances population diversity through good point set initialization and periodic mutation mechanisms. A Beta distribution is introduced to dynamically generate reflected solutions to explore a wider search space, and Lévy flight is applied to handle boundary violation issues. Additionally, the spiral search update mechanism from the Whale Optimization Algorithm is integrated, along with random strategy updates for position, significantly improving the algorithm's convergence accuracy and robustness. When the algorithm stagnates, a t-distribution mutation perturbation strategy is introduced, effectively enhancing its ability to escape local optima. Simulations on 17 benchmark functions test functions validate the effectiveness of the improved strategies. Moreover, the application of ODBO to the job-shop scheduling problem confirms its effectiveness and reliability in solving real-world engineering problems.

Keywords

Dung Beetle Optimization, Local Optima, Whale Optimization Algorithm, T-Distribution Mutation Perturbation, Job-Shop Scheduling Problem, Robustness Analysis

1. 引言

在生产调度领域，面对日益复杂的问题，传统的精确求解策略如整数规划和网络流算法，虽然能确保找到最优解，但其计算成本却随着问题规模的扩大而迅速攀升。为了解决这一问题，研究者们开始转向启发式算法[1]，这些算法在迭代过程中通过避免陷入局部最优解来提高效率。蜣螂优化算法(DBO) [2]，由 Xue 等人在 2022 年提出，是一种新兴的群体智能优化技术。该算法已经在多个实际问题中得到应用，例如生物医学和能源系统规划。与传统的群体智能算法相比，DBO 算法的位置更新是并行进行的，它模拟了蜣螂的四种行为：滚球、繁育、觅食和偷窃。这种并行更新机制使得 DBO 算法在处理大规模问题时具有潜在的优势。然而，DBO 算法也面临着一些挑战，如算法稳定性、可解释性以及高纬度问题处理困难。

为了克服这些挑战，本文提出了以下改进措施：首先，通过使用佳点集群初始化[3]方法来增强种群的多样性，并引入周期性突变机制[4]来帮助算法跳出局部最优。其次，采用 Beta 分布来动态生成反射解[5]，以更有效地探索解空间，并利用莱维飞行策略[6]来处理边界违规问题[7]。最后，结合鲸鱼优化算法中的螺旋搜索机制[8]，通过随机策略来更新位置，从而提高算法的收敛精度。当算法遇到停滞状态时，采用 t 分布扰动变异策略[9]来进一步增强算法跳出局部最优解的能力。这些改进在基准函数测试和车间排产优化问题中的应用中得到了验证，证明了其在工程领域的实用性。

2. 蜣螂优化算法

受自然界中蜣螂独特生活习性的启发，科研人员开发了一种创新的优化策略。在自然界里，蜣螂展现出滚动粪球、精心挑选繁殖地、积极觅食以及巧妙偷窃等多种行为。在算法设计中，这些复杂行为被巧妙地提炼为四种各具特色的搜索策略，它们分别对应于算法中的四种蜣螂角色：滚动者、繁殖者、觅

食者和偷窃者。它们遵循各自独特的位置更新机制，协同合作，共同在广阔的解空间中探索，旨在发现全局最优解。

(1) 滚球蜣螂

在滚动的过程中，蜣螂必须通过天体的线索来导航，以保持粪球在直线上滚动。因此，滚球蜣螂的位置被更新，并且可以表示为

$$x_i(t+1)=x_i(t)+\alpha\times k\times x_i(t\text{-}1)+b\times\Delta x \tag{1}$$

$$\Delta x=\left|x_i(t)-X^w\right| \tag{2}$$

其中 t 表示当前迭代次数；$x_i(t)$ 表示第 t 次迭代期间，第 i 只蜣螂的位置信息；$k\in(0,0.2]$是表示偏转系数的常数值；b 是属于$(0,1)$的常数值；α 是分配给 1 或−1 的自然系数；X^w 是全局最差位置。当一只蜣螂遇到阻碍它前进的障碍物时，它会通过跳舞来改变自己的方向，以获得一条新的路线。正切函数模拟舞动行为以获得新的滚动方向。因此，滚球蜣螂的位置更新如下：

$$x_i(t+1)=x_i(t)+\tan\theta\left|x_i(t)-x_i(t-1)\right| \tag{3}$$

(2) 繁殖蜣螂

选择一个合适的产卵地点对蜣螂为后代提供一个安全的环境至关重要。因此，在 DBO 中，提出了一种边界选择策略来模拟雌性蜣螂的产卵区，其定义如下：

$$\begin{aligned}Lb^*&=\max\left(X^*\times(1-R),Lb\right)\\Ub^*&=\min\left(X^*\times(1-R),Ub\right)\end{aligned} \tag{4}$$

其中，X 表示当前的局部最优位置；Lb 和 Ub 表示产卵区域的上下界；$R=1-t/\mathrm{Tmax}$；Tmax 表示最大迭代次数；Lb 和Ub 分别是可行区域的上下界 S。Ub^*、Lb^* 为安全区域的上界和下界。假设每只雌性蜣螂在每次迭代中只产一个卵。产卵区域的边界范围随值动态变化，因此在迭代过程中，蛋球的位置也动态变化，表示如下：

$$B_i(t+1)=X^*+b_1\times\left(B_i(t)-Lb^*\right)+b_2\times\left(B_i(t)-Ub^*\right) \tag{5}$$

其中 $B_i(t)$是第 t 次迭代时第 i 个球体的位置；b_1 和 b_2 是两个独立的随机向量，大小为 1 × D；D 是维数。

(3) 小蜣螂

一些成年的蜣螂会从地下钻出来寻找食物，这种蜣螂被称为小蜣螂。小蜣螂最佳觅食区的边界定义如下：

$$\begin{aligned}Lb^b&=\max\left(X^b\times(1-R),Lb\right)\\Ub^b&=\min\left(X^b\times(1-R),Ub\right)\end{aligned} \tag{6}$$

其中 X^b 表示全局最优位置；Lb^b 和 Ub^b 分别是最优觅食区域的下限和上限。因此，小蜣螂的位置更新如下：

$$x_i(t+1)=x_i(t)+C_1\times\left(x_i(t)-Lb^b\right)+C_2\times\left(x_i(t)-Ub^b\right) \tag{7}$$

其中 $x_i(t)$ 表示第 t 次迭代时第 i 只蜣螂的位置；C_1 为服从正态分布的随机数；C_2 为$(0,1)$范围内的随机向量。

(4) 偷窃蜣螂

有些被称为小偷的蜣螂会从其他蜣螂那里偷粪球。从等式(5)可以观察到，X^b 是最佳食物源。假设

X^b 周围的区域是竞争食物的最佳位置。在迭代过程中，小偷的位置信息更新如下：

$$x_i(t+1)=X^b+S\times g\times\left(\left|x_i(t)-X^*\right|+\left|x_i(t)-X^b\right|\right) \tag{8}$$

其中，$x_i(t)$ 表示第 t 次迭代时第 i 个小偷的位置信息；g 为正态分布的随机向量，大小为 1 × D；S 为常数值。

3. 多策略改进的蜣螂优化算法

本章节介绍的改进型蜣螂优化算法(ODBO)融合了三项关键策略：首先是优势点集群的初始化方法和周期性突变机制；其次是动态生成反向解的策略，以及莱维飞行在处理边界冲突时的应用；最后是结合了鲸鱼搜索算法的螺旋搜索技术，并引入了基于 t 分布的扰动变异策略。

3.1. 佳点集群初始化策略和周期突变机制

为了增强算法种群的遗传多样性，本研究采取了一种周期性突变策略，这一策略使得优化算法能够周期性地在解空间内进行大幅度跳跃，有效地扩展了搜索范围。这种周期性跳跃有助于算法在广阔的解空间中进行深入探索，从而提高找到全局最优解的几率。蜣螂的四种种群更新方式为：

$$x_i(t+1)=x_i(t)\times[1+A\times(0.5-rand)\times\sigma \tag{9}$$

$$\sigma=\begin{Bmatrix}1\cdots t=nT\\0\cdots t\neq nT\end{Bmatrix}n=1,2,3,\cdots \tag{10}$$

其中 A 为突变幅度；$rand$ 表示按照均分布 N (0, 1)随机生成的实数；T 为突变周期，其值小于迭代次数；σ 是根据突变周期从一个区域搜索到另一个区域的冲量。

3.2. 动态反向学习策略

DBO 算法中采用的动态区域选择策略，虽然在特定情况下有效，但可能会将搜索过程局限在较小的局部区域内，从而限制了对全局最优解的探索。为了克服这一局限，本研究提出了一种基于反向动态计算的策略，该策略通过扩展搜索边界，有助于算法跳出局部最优的局限，实现对更广阔解空间的探索。传统的初始化方法，如基于均匀分布的随机位置生成，虽然操作简便，但可能不足以形成对潜在最优解的有效“包围”。为了改善这一状况，本文引入了一种创新的方法，即利用 Beta 分布的特性来生成初始解，从而在解空间中形成对最优解的更紧密包围，类似 Beta 分布函数定义为：

$$f(x)=\frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1},x\in[0,1] \tag{11}$$

3.3. 边界违规处理

处理单个粒子位置超出预定边界的情况是一个关键问题。一种常见的简单处理方法是将超出的粒子位置直接设置为边界值，这种方法实现起来直截了当。然而，由于边界点往往不是全局最优解，这种直接的策略可能会限制优化过程的效果。这种方法的优点是易于实现。为了引入更多的随机性并提高搜索效率，Levy 飞行作为一种数学方法被用来生成具有长尾分布的随机步长。Levy 飞行能够模拟粒子在解空间中的随机游走，其步长遵循 Levy 分布。然而，Levy 飞行可能产生的过长步长可能会导致粒子跳过潜在的最优解区域。为了平衡这一点，Levy 飞行的步长会乘以一个适当的常数 CC 以降低其步长长度。本文采用一种新的基于 Levy 分布映射的新型边界处理方法。具体操作如下：

levy 飞行机制结束后，采用贪婪策略，如果粒子的适应度小于前一代适应度，则对粒子进行替换。

$$x_{ij}^{t}=\begin{cases}\min\left(Ub^{j}\otimes Levy\left(\lambda\right),Ub^{j}\right),x_{ij}>Ub^{j}\\ \max\left(Lb^{j},Lb^{j}\otimes Levy\left(\lambda\right)\right),x_{ij}<Lb^{j}\end{cases}\tag{12}$$

其中 $\boldsymbol{x}_{ij}$ 是边界处理后的粒子位置；λ 是常数；Levy 是随机搜索路径，其随机步长是 Levy 分布。符号是条目式乘法。图 1 直观地提供了一种直观的视角来展示粒子边界处理方法。当粒子 $\boldsymbol{x}_{ij}$ 跳出上边界和下边界时，通过 Levy 飞行调整其位置，确保有效搜索解空间。

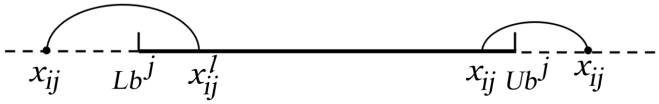

Figure 1. Diagram for particle boundary processing
图 1. 粒子边界处理方法图

3.4. 引入鲸鱼优化算法螺旋搜索策略和 t 分布扰动变异策略

3.4.1. 鲸鱼优化算法螺旋搜索更新机制

在蜣螂优化算法(DBO)中，如果按照当前的在产卵区域内繁殖雏球，这肯定会使种群在短时间内快速收敛，但也会导致种群的多样性降低，容易使算法陷入局部最优；因此对蜣螂算法繁殖阶段的这个公式进行改进，在蜣螂优化算法(DBO)中，盗窃蜣螂的行为对于种群多样性和算法的全局搜索能力有着重要影响。然而，传统的盗窃行为可能过于依赖当前搜索区域，导致种群多样性的降低，增加了算法陷入局部最优解的风险。为提升算法的全局探索性及种群的遗传多样性，本研究对蜣螂优化算法中的盗窃蜣螂行为引入了螺旋搜索策略。该策略仿照自然界中某些生物的螺旋形运动轨迹，使个体能够沿螺旋路径在解空间内进行探索，有效拓宽了搜索视野，增加了发现更优解的可能性。在经改进的 DBO 算法框架下，盗窃蜣螂在更新位置时，放弃了传统的简单复制或随机变异手段，转而依据其当前位置与历史搜索数据，采纳螺旋搜索策略来开辟新的探索路径。在每一轮迭代过程中，盗窃蜣螂依据自身的当前坐标和过往搜索经验，确定螺旋探索的起始点、半径及旋转角度，继而沿此螺旋轨迹进行搜索，并依据所获解的适应度来优化自身位置。这种更新机制使得盗窃蜣螂在维持与种群当前状态的联系的基础上，能够突破局部搜索局限，向更广阔的解空间扩展探索，从而为种群多样性的增加和算法全局搜索能力的提升做出了贡献。螺旋搜索策略的融入，不仅助力算法摆脱局部最优的困境，也为种群在持续搜索过程中的多样性维持提供了保障，显著增强了算法解决复杂优化问题的综合效能。

$$\mathrm{B_i}\left(\mathrm{t}+1\right)=\mathrm{X}^{*}+\mathrm{b_1}\times\left(\mathrm{B_i}\left(\mathrm{t}\right)-\mathrm{Lb}^{*}\right)+\mathrm{b_2}\times\left(\mathrm{B_i}\left(\mathrm{t}\right)-\mathrm{Ub}^{*}\right)\tag{13}$$

鲸鱼优化算法(WOA)借鉴了鲸鱼的社会捕食模式，通过群体中的领导者引导，实现快速而多样的搜索。算法中，领导者通过发泡网攻击策略，指引其他成员，占群体的 10%到 20%。追随者根据以下公式更新位置：

受到鲸鱼算法中头鲸种群围捕猎物的启发，鲸鱼算法在迭代过程中，个体鲸鱼会使用螺旋搜索策略来更新与猎物的位置，这不仅保证了算法的收敛速度，而且可以增加个体的多样性。鲸鱼围捕猎物阶段公式如下：

$$X\left(t+1\right)=D^{t}\cdot e^{cl}\cdot\cos\left(2\pi l\right)+X^{*}\left(t\right)$$

$$D'=\left|X^{*}\left(t\right)-X\left(t\right)\right|$$

$$Pos_{new}=pos_{b}\left(t\right)+\left(pos_{b}\left(t\right)-pos_{c}\left(t\right)\right)\times C\times Levy\tag{14}$$

但该策略很容易受到定义参数 c 的影响。较大的 c 会使算法衰减过快，导致算法局部最优，较小的 c 会导致算法收敛缓慢。为了解决这个问题，引入了动态螺旋搜索形状的参数 r。

$$r = e^{cgcos\left(\frac{\pi \cdot t}{Max\_iteration}\right)} \tag{15}$$

式中：$|\vec{X}best(t) - \vec{X}(t)|$表示当前鲸鱼个体与最佳位置鲸鱼之间的距离向量；常量系数 b 决定了鲸鱼个体螺旋前进时的螺旋线形状，b 取值为 1 时即为普通的对数螺旋线，l 是[−1, 1]之间的随机数。WOA 算法中鲸鱼个体位置向量 $\vec{X}(t)$的每一个分量在种群更新时均独立进行。

3.4.2. T 分布扰动变异

在 T 分布中，随着自由度值的减小，曲线的形状越来越平，曲线的中部越低，曲线两侧的尾端翘起的幅度越大。使用 T 分布扰动蜣螂个体的位置，实现种群的变异，然后更新变异后的蜣螂个体位置的目标函数值，对比前一代最优函数值，如果结果更好，则改当前最优值。T 分布扰动公式如下：

$$x_i^t = x_i + x_i \cdot t(iter) \tag{16}$$

式中：x，是变异后种群中第 i 个布谷鸟窝新的位置；x 是变异前的个体位置；$t(iter)$为 T 分布值，以迭代次数为自由度。算法迭代前期，$iter$ 数值很小，由 T 分布函数生成的结果近似于柯西变异，全局搜索能力强，晚期 $iter$ 数值偏大，类似高斯变异，拥有良好的局部搜索能力，进而使算法的搜索精度得以提高。

3.5. 算法复杂度分析

在深入探究 DBO (Dung Beetle Optimizer，即蜣螂优化算法)的时间复杂度时，我们可以将其分解为三个核心组成部分：种群的初始化过程、适应度的评估环节，以及个体的位置更新机制。假设我们有一个最大迭代次数 T、种群规模 N，以及问题维度 D 的设定。对于种群的初始化，其时间复杂度为 O (N × D)，这是因为每个个体都需要在 D 维空间中设定一个初始位置。接着，适应度评估的时间复杂度为 O (N × T)，因为每个个体在每次迭代时都需要计算其适应度值。在个体的位置更新方面，DBO 算法模拟了四种不同类型的蜣螂行为：滚球、繁育、觅食(此处可能原指“小蜣螂”的某种觅食行为，为简化表述而调整)、以及偷窃。每种蜣螂在种群中都有一定的数量 N，且这些行为在每次迭代时都会对所有个体进行更新。尽管每种蜣螂的更新方式可能有所不同，但总体上，这些更新操作的时间复杂度最高可达到 O (max (N_i) × T × D)。然而，由于所有 N_i 的总和等于种群规模 N，因此我们可以将 DBO 算法的整体时间复杂度简化为 O (max (N_i) × T × D)。

接下来，我们介绍 ODBO 算法(Omnifarious Dung Beetle Optimizer，即多维度蜣螂优化算法)，它是在 DBO 的基础上进行了扩展，并融入了三种新的策略。首先，佳点集群初始化策略和周期突变机制主要作用于初始化阶段，它们并没有增加额外的位置更新或适应度评估步骤，因此 ODBO 算法在这一阶段的时间复杂度保持不变。其次，ODBO 算法还引入了动态反向学习和边界违规处理策略，并结合了鲸鱼优化算法和 t 分布扰动变异。尽管这些新策略增加了算法的复杂性，但它们在位置更新和适应度评估方面的最大时间复杂度仍然分别为 O (N × D)和 O (N × T)。因此，综合考虑所有因素，ODBO 算法的整体时间复杂度仍然保持在 O (N × T × D)，与 DBO 算法相同。

3.6. 算法复杂度分析

所提出的 ODBO 算法的复杂度分析如下图 2。优化器的复杂度是衡量算法效率的关键指标之一。ODBO 算法的复杂性主要包括以下几个过程，如下图 3：初始化；通过 Beta 分布计算反射解；周期突变

机制；螺旋搜索；t 分布扰动变异；以及使用新的边界处理方法更新蜣螂的位置。假设在具有 D 个变量的问题中，涉及 N 个解和 T 次迭代。首先，算法的初始化复杂度为 O (N)。然后，计算 N 个反射解的复杂度为 O (T * N)。在 ODBO 中，每个周期需要评估 N 组解，总评估次数为 4 * T * N。因此，蜣螂寻找最佳位置的时间复杂度约为 O (4 * T * N)，简写为 O (T * N)。综上所述，ODBO 的总体时间复杂度为 O (ODBO) = O (N + T * N * (1 + D + N))。尽管相对于 DBO 算法增加了一些时间开销，但性能得到了显著提升。

Algorithm 1 The framework of the ODBO algorithm

Require: The maximum iteration T, the size of the particle's population N, and the problem dimension D
Ensure: Optimal position X^b and its fitness value f_b

```
Initialize the particle's population i ← 1, 2, ..., N using the good point set strategy
Calculate the fitness of the initial population
while t ≤ T do
    for i = 1 to N do
        Generate opposite solution using dynamic reflective learning based on Beta distribution
    end for
    for i = 1 to N do
        Record position and fitness history
        if t mod C == 0 then
            Apply periodic mutation mechanism
        else
            Update positions based on behavior strategies:
            if individual type is ball-rolling dung beetle then
                Update position using Eq.(1)
            else if individual type is brood ball dung beetle then
                Update position using Eq.(2)
            else if individual type is small dung beetle then
                Update position using Eq.(3)
            else
                Update position using Eq.(4) and spiral search strategy
            end if
        end if
        Handle boundary violations using Levy flight
        Evaluate the new position
        Greedy selection: update individual if new position is better
    end for
    Apply t-distribution mutation to the best individual
    Store the best cost found so far
    Display the best cost for the current iteration
end while
Return X^b and its fitness value f_b
```

Figure 2. Pseudocode diagram for algorithm
图 2. 算法伪代码图

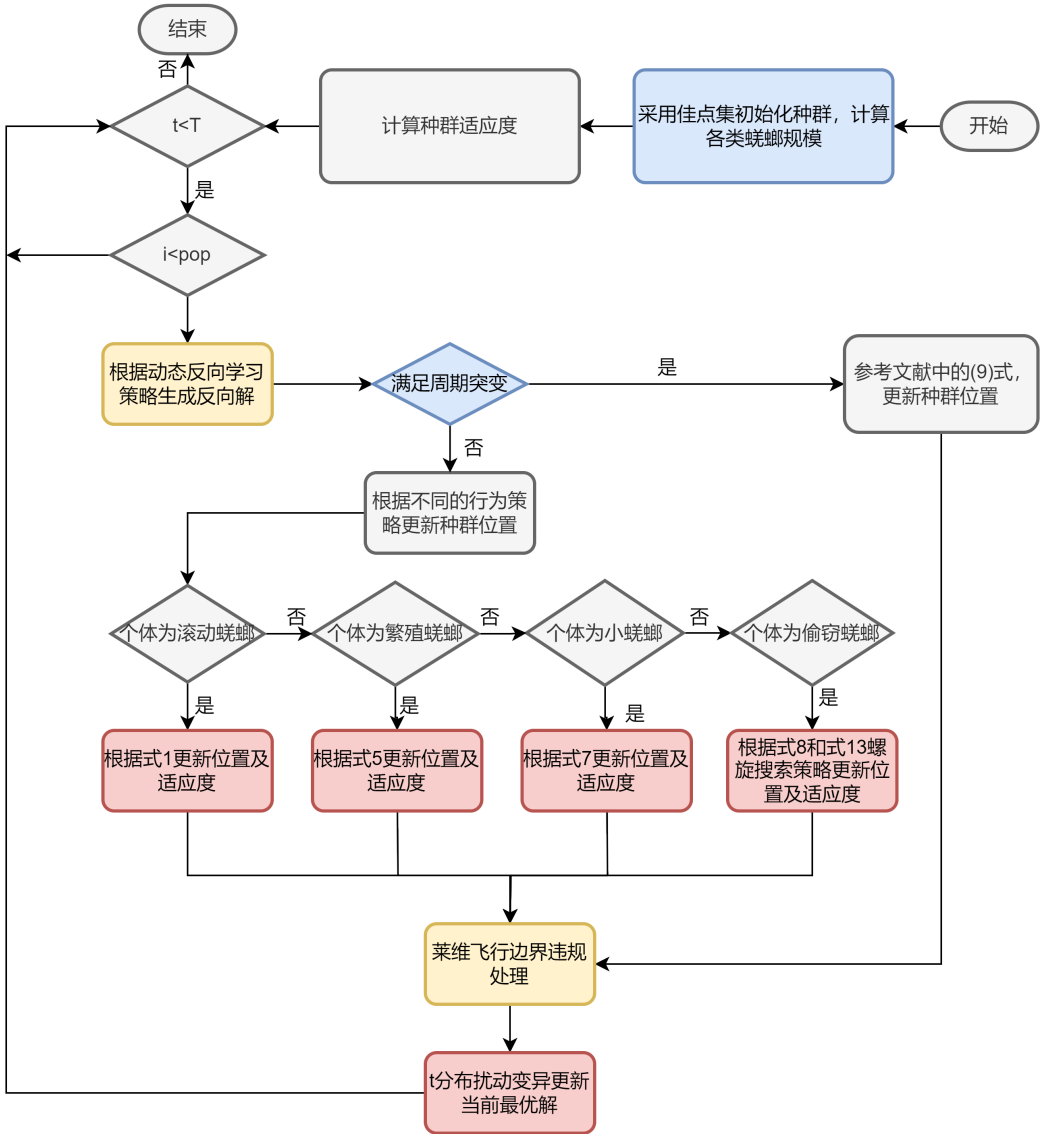

Figure 3. Flowchart of algorithm
图 3. 算法流程图

4. 仿真实验及结果分析

4.1. 测试函数说明

为了全面评估 ODBO 算法的寻优性能，我们采用了表 1 中列出的 17 个经典测试函数。这些测试函数被分为三类：

1) 高维单峰值函数(F1 至 F7)：这些函数仅含有一个全局最优解，能够有效检验算法的开发(或称为局部搜索)能力。

2) 高维多峰值函数(F8 至 F13)：这些函数不仅包含一个全局最优解，还包含多个局部最优解，是检验算法全局搜索能力和局部逃逸能力的理想选择。

3) 固定维度多峰函数(F14 至 F17)：此类函数在特定维度下具有多个峰值，有助于验证算法在探索和开发能力之间的平衡。

Table 1. Test functions
表 1. 测试函数

函数	名称	维度	范围	最优值	函数	名称	维度	范围	最优值
F1	Sphere	30/100	[−100, 100]	0	F11	Griewank	30/100	[−600, 600]	0
F2	Schwefel2.22	30/100	[−10, 10]	0	F12	Penalized	30/100	[−50, 50]	0
F3	Schwefel1.2	30/100	[−100, 100]	0	F13	Penalized2	30/100	[−50, 50]	0
F4	Schwefel2.21	30/100	[−100, 100]	0	F14	Foxholes	2	[−65, 65]	0.9980
F6	STEP	30/100	[−100, 100]	0	F15	Kowalik's	4	[−5, 5]	0.0003
F7	Quartic	30/100	[−1.28, 1.28]	0	F21	Shekel5	4	[0, 10]	−10.153
F8	Schwefel	30/100	[−500, 500]	−12569	F22	Shekel7	4	[0, 10]	−10.402
F9	Rastrigin	30/100	[−5.12, 5.12]	0	F23	Shekel10	4	[0, 10]	−10.536
F10	Ackley	30/100	[−32, 32]	0					

为确保实验的公平性和可比较性，所有算法的实验仿真均在相同的软硬件平台上进行，具体使用的是 Matlab 2021a 软件。实验参数统一设置如下：最大迭代次数 T 设为 300，种群数量 pop 设为 30。每种算法均独立运行 30 次，并以其 30 次寻优结果的最好值(Best)、平均值(Mean)和标准差(Std)作为评价指标。

通过这三类不同类型的测试函数，我们旨在全面而深入地评估 ODBO 算法的寻优性能。

4.2. 改进策略有效性分析

ODBO 算法是在 DBO 算法的基础上进行了三项关键性改进，每项改进分别与 DBO 融合，形成了三种不同的变体算法：

1) ODBO1：通过将种群初始化策略和周期突变机制引入到 DBO 中，我们得到了 ODBO1。这一改进旨在提升算法在初始阶段的多样性和在迭代过程中的突变能力。

2) ODBO2：在 DBO 的基础上，我们加入了生成反向解的动态学习反向策略以及莱维飞行边界违规处理机制，从而诞生了 ODBO2。这一改进旨在增强算法在搜索空间中的探索能力和对边界条件的处理能力。

3) ODBO3：通过将螺旋搜索策略和 t 分布扰动变异融入 DBO，我们得到了 ODBO3。这一改进旨在提高算法在局部搜索中的精细度和在全局搜索中的随机性。

为了全面评估 ODBO 及其变体算法的性能，我们将其与 DBO、MSADBO (麻雀搜索算法) [10]等五种算法进行了对比实验。实验采用了表 1 中列出的 17 个测试函数，每种算法均独立运行 30 次，并记录其寻优结果的最好值(Best)、平均值(Mean)和标准差(Std)。其中，最好值反映了算法的精确性，即其能够找到的最优解的质量；平均值和标准差则共同反映了算法的稳定性，即其多次运行结果的一致性和可靠性。实验结果如表 2 所示。

表 2 中的加粗数据表明了相较于 DBO 算法，平均寻优精度有所降低的值。根据表 2 不同策略寻优结果表 2 的数据分析，我们可以得出以下结论：

与 DBO 相比，ODBO1、ODBO2 和 ODBO3 分别在 12 个、9 个和 12 个测试函数上实现了平均寻优精度的提升。而 ODBO 算法在 13 个函数上的平均寻优精度相较于 DBO 有明显提高，尽管在其余 4 个函数上的提升幅度不大，但在寻优结果的最好值和标准差方面却有所优化。具体来看，ODBO1 在函数 F1 至 F4 上的最优收敛值达到了理论最优解，这充分证明了佳点集群初始化策略和周期突变机制对于提升算法收敛精度的有效性。ODBO2 在 F1 至 F11 (除 F6 外)的 10 个测试函数上，无论是最优收敛值还是平均

收敛精度，都实现了大幅提升。特别是在多峰函数 F11 上，ODBO2 的标准差相较于 DBO 更小，这表明在 DBO 中引入反向解策略能够显著提高算法的稳定性。而 ODBO3 在 F9 和 F11 两个函数上的平均值和最优收敛值都达到了理论最优解，要知道，F9 的 Rastrigin 函数是一个高度多模态的函数，具有许多局部极小值，且这些极小值的位置规则分布，优化条件十分苛刻。ODBO3 能够找到其全局最优解，说明将鲸鱼算法的螺旋搜索策略和 t 分布扰动变异融入 DBO 中，可以显著提升算法的寻优性能，并增大跳出局部最优解的概率。对于固定维度函数 F14 至 F23，从 F13、F15 以及 F21 至 F23 的数据可以看出，融入了三种策略的 ODBO 在平均收敛精度上相较于 DBO 有了显著提升。而对于函数 F14，虽然 ODBO 的平均收敛精度略低于 DBO，但在最优精度值上却与 DBO 基本相当。这表明，融入三种策略的 ODBO 能够很好地平衡算法的探索与开发能力。与 DBO 相比，融合了三种改进策略的 ODBO 在 F1 至 F23 (除 4 个函数外)上的最优收敛值、平均收敛精度以及标准差都有所提升。特别是在 F1 至 F2 上的最优收敛值，以及 F9 至 F11 上的平均收敛精度，都达到了理论最优解，且未出现负优化的现象。这充分证明了三种策略融合的有效性。

Table 2. Optimization results of different strategies
表 2. 不同策略寻优结果

函数	指标	DBO	ODBO1	ODBO2	ODBO3	QHDBO	MSADBO	ODBO
F1	Best	6.62E−104	7.5E−214	2.3E−249	9.2E−192	4.8E−249	8.93E−298	0
	Std	2.866E−51	3.2E−138	7.81E−72	2.6E−139	0	0	0
	Mean	5.233E−52	5.9E−139	1.43E−72	4.7E−140	2E−212	4.01E−193	4.7E−283
F2	Best	8.206E−52	7.3E−105	1.1E−125	9.5E−107	1.6E−129	2.19E−148	1.6E−159
	Std	5.89E−32	3.68E−69	1.4E−124	5.13E−72	3E−106	1.78E−105	4.5E−138
	Mean	1.278E−32	6.81E−70	4.7E−125	1.08E−72	5.7E−107	3.27E−106	8.2E−139
F3	Best	3.588E−91	7.9E−242	6E−248	1.9E−183	4.8E−242	1.02E−218	0
	Std	1.427E−53	1.8E−127	0	1.9E−123	0	1.45E−138	0
	Mean	3.413E−54	3.2E−128	1E−245	3.5E−124	3.5E−211	2.64E−139	5.6E−281
F4	Best	4.253E−51	3.5E−111	3.2E−125	5.35E−96	2.7E−127	5.36E−157	6.6E−166
	Std	2.828E−33	8.47E−68	7.4E−126	2.9E−69	1.1E−103	8.875E−93	1.7E−140
	Mean	7.44E−34	1.69E−68	4.3E−125	5.37E−70	2E−104	1.809E−93	3.1E−141
F6	Best	6.694E−19	0.009342	0.027981	0.010332	4.72E−11	2.709E−15	0.018906
	Std	7.375E−14	0.01537	0.195494	0.024425	0.065632	1.219E−12	0.030919
	Mean	4.167E−14	0.032816	0.233292	0.051217	0.073473	9.049E−13	0.063923
F7	Best	0.0001567	9.45E−06	5.07E−05	2.82E−06	8.71E−06	1.931E−05	4.18E−06
	Std	0.0011708	0.000312	0.000209	0.000245	0.000202	0.000144	0.000243
	Mean	0.00172	0.000291	0.000269	0.000287	0.000205	0.0002141	0.000203
F8	Best	−4189.829	−1.3E+53	−2.1E+31	−1.2E+59	−4189.83	−1.09E+37	−1706.04
	Std	532.17475	2.33E+52	3.78E+30	2.26E+58	797.5957	1.985E+36	184.1298
	Mean	−3249.935	−4.5E+51	−6.9E+29	−4.1E+57	−3183.34	−3.68E+35	−1305.42

续表

F9	Best	0	0	0	0	0	0	0
	Std	8.5886184	0	0	0	0	0	0
	Mean	2.3126043	0	0	0	0	0	0
F10	Best	8.882E−16	8.88E−16	8.88E−16	8.88E−16	8.88E−16	8.882E−16	8.88E−16
	Std	0	0	0	0	0	0	0
	Mean	8.882E−16	8.88E−16	8.88E−16	8.88E−16	8.88E−16	8.882E−16	8.88E−16
F11	Best	0	0	0	0	0	0	0
	Std	0.0592216	0	0	0	0	0	0
	Mean	0.0318332	0	0	0	0	0	0
F12	Best	6.501E−19	0.006201	0.004179	0.003074	0.00148	2.901E−15	4.71E−32
	Std	4.567E−06	0.004989	0.021025	0.006014	0.029792	1.494E−12	1.13E−05
	Mean	8.339E−07	0.013597	0.027632	0.015356	0.034312	7.693E−13	3E−06
F13	Best	3.68E−16	0.001118	0.264409	0.009342	0.000259	1.551E−13	1.35E−32
	Std	0.0680511	0.008824	0.152237	0.081651	0.005239	0.0335791	1.26E−06
	Mean	0.052793	0.013238	0.640819	0.110192	0.006627	0.0133493	3.44E−07
F14	Best	0.9980038	12.67051	0.998078	0.998004	10.76318	0.9980038	0.998004
	Std	2.0316918	2.31E−13	1.110446	1.022965	0.953117	1.8283697	4.064419
	Mean	1.8846914	12.67051	2.304755	1.490099	11.68512	1.4557832	3.180025
F15	Best	0.0003076	0.000308	0.000435	0.000358	0.000307	0.0003075	0.000307
	Std	0.0003926	5.37E−05	0.003715	0.000339	4.07E−05	0.0003167	3.48E−05
	Mean	0.0007954	0.00035	0.003789	0.000779	0.000339	0.0005582	0.000336
F21	Best	−10.1532	−10.0398	−8.64979	−9.97054	−10.1532	−5.055198	−10.1532
	Std	2.8012857	1.496831	1.034111	1.880955	2.375941	2.0796472	0.00699
	Mean	−6.869318	−7.79054	−5.47573	−7.10676	−6.58434	−3.385713	−10.1472
F22	Best	−10.40294	−10.3271	−10.3265	−9.75886	−10.4029	−10.40294	−10.4029
	Std	2.9005639	1.653269	1.82438	1.698345	1.819441	2.7975311	0.026964
	Mean	−8.113965	−7.78766	−6.0879	−6.78902	−5.78918	−3.245011	−10.3929
F23	Best	−10.53641	−10.2633	−9.04942	−10.4956	−10.5364	−5.128481	−10.5363
	Std	2.8747407	1.501277	0.918715	1.865828	2.200108	2.1004129	0.01602
	Mean	−8.099543	−7.88544	−5.47779	−6.9271	−6.21004	−2.945935	−10.5277

5. 车间排产优化问题求解

为了进一步验证 ODBO 算法在实际工程应用中的可靠性，选取某制造企业车间生产排产调度设计问题进行验证。车间工单排产问题需要最小化成型炉占用位置，具体描述如下。

5.1. 模型描述

工单排产问题描述如下：

1) 在某一时间共有 N 个工单等待排产，每个工单包含以下关键信息：胎具号、组数(占用成型炉位置的个数)、圈数、产线信息、温度等。

2) 生产线有 M 个节点，每个节点只有一台加工机器，每台机器上有 P 个位置。

3) 吊炉有 6 个位置，每个位置的最大运转圈数和工艺路线相关略有不同；双层炉有 5 个位置。在这里它们统称成型炉

4) 工单在排产时，应避免同一胎具的不同工单分配到相同班次，除非其运转圈数相加不大于最大运转圈数。(共用)，而胎具合并就是两种不同编号的胎具实际上是合并的关系。

5) 尽量减少胎具的搬运次数。

5.2. 变量定义

N：待排产的工单数量；

M：生产线节点数量；

P：每个节点的位置数量；

x：工单分配矩阵，元素为二元变量，表示工单是否分配到节点的位置；

T：工单的运转圈数矩阵；

C：节点在特定温度下的最大运转圈数矩阵；

S：胎具合并的工单集。

5.3. 目标函数

本文仅考虑单个目标函数，即成型炉位置占满度最大化：

$$\max \sum_{j=1}^{M}\sum_{k=1}^{p}\sum_{i=1}^{N} x_{ijk} \tag{17}$$

该目标函数就是用最少的机器最少的位置来完成当天的生产需求。

5.4. 约束条件

1) 允许多个工单分配到同一个位置，只要它们的总运转圈数不超过位置的最大承载能力

引入符号：

T_{ij}：工单 i 在位置 j 上的运转圈数。

C_j：表示位置 j 的最大承载能力。

约束公式：

$$\sum_{i=1}^{N} T_{ij} x_{ijk} \le C_j \tag{18}$$

2) 不同工艺路线的工单不能分到同一个成型炉上，因为成型炉一旦开工就不切换温度了。

引入符号：

R_i：工单 i 的工艺路线。

x_{ijk}：工单 i 是否分配到成型炉 j 的位置 k 上，二元变量，1 表示分配，0 表示不分配。

约束公式：

$$x_{ijk}R_i = x_{ijk}R_j \tag{19}$$

3) 每个成型炉上分配到的工单所实际占用的位置数须小于等于 6。

$$\sum_{i=1}^{N} x_{ijk} \le 6 \tag{20}$$

5.5. 算例分析

本文以某汽车制造公司的实际生产排程为例，利用所设计的算法求解，算例中涉及 22 个成型炉，并从当天 APS 系统历史抽取出来 70 个某车间待排程工单。

其中前 14 个成型炉是吊炉，后 8 个是双层，这里展示白夜班认为是两个不同的成型炉编号，具体成型炉配置信息如表 3。

Table 3. Configuration information of molding furnace
表 3. 成型炉配置信息

成型炉	位置	圈数	工艺路线
吊炉	6	34	CW
吊炉	6	28	GW
吊炉	6	21	DW
双层炉	5	35	CW
双层炉	5	30	GW

然后这里的工单参数如下表，这里只展示部分工单数据因为量比较大，数据来源如下表 4。

Table 4. Actual work order data
表 4. 实际工单数据

工单编号	组数	圈数	工艺路线	胎具号	成型炉编号
1	2	28	GW	1	DL1
2	1	34	CW	2	DL2
3	3	25	GW	1	DL3
4	5	28	GW	2	DL3
5	6	34	CW	3	DL2
6	1	15	CW	4	DL1
7	2	30	CW	5	DL1
8	1	2	CW	6	DL1
9	1	5	CW	6	DL1
10	1	25	GW	10	DL4

如下图 4 是从企业 APS 系统获取到的原始数据，因实际生产需求，系统排产不满足现状。首先可以看出当天某车间成型排产若干个工单，但是实际上最后分配的实际情况却是很多机器的负载量要么过低要么过高，这里当天机器负载圈数 204。

成型炉编码	工单号	工单状态	物料	旧物料	物料描述	辅助工序	胎具总根数	操作
DL1-CX-1	CXBZ2406080049	已下达	474502063667	5B2CHJH_3540400U73C0_4	成型管 5B2CHJH_3540400U73C0_4		5	
DL1-CX-1	CXBZ2406080050	已下达	474502063666	5B2CHJH_3540400U73C0_5	成型管 5B2CHJH_3540400U73C0_5	OD10×1；截...	5	
DL1-CX-1	CXBZ2406080051	已下达	474502063670	5B2CHJH_3540400U73C0_1	成型管 5B2CHJH_3540400U73C0_1	OD10×1；截...	5	
DL1-CX-1	CXBZ2406080053	已下达	474502063668	5B2CHJH_3540400U73C0_3	成型管 5B2CHJH_3540400U73C0_3	OD10×1；截...	5	
DL1-CX-1	CXSZ2406080074	已下达	474502066194	5B2CHQR_F081104113HD	成型管 5B2CHQR_F081104113HD		25	
DL1-CX-1	CXSZ2406080208	已下达	474502062985	5B2CHJL_PP69J270AB	成型管 5B2CHJL_PP69J270AB		5	
DL1-CX-1	CXSZ2406080324	已下达	474502061624	5B2CHQR_F263541010	成型管 5B2CHQR_F263541010		15	
	合计							
DL1-CX-2	CXSZ2406080077	已下达	474502062997	5B2CHJH_1104240S454Y	成型管 5B2CHJH_1104240S454Y		5	
DL1-CX-2	CXSZ2406080231	已下达	474502064465	5B2CHCR_204000886AA_1	成型管 5B2CHCR_204000886AA_1		5	
	合计							
DL10-CX-2	CXSZ2406080032	已下达	474502061904	5B2CHNW_10210015_1	成型管 5B2CHNW_10210015_1	OD16×1.25；...	25	

总记录数：109

Figure 4. Raw data diagram of unplanned production in an enterprise's internal system
图 4. 某企业内部系统未排产原始数据图

如下图 5 是优化以后的最后结果。

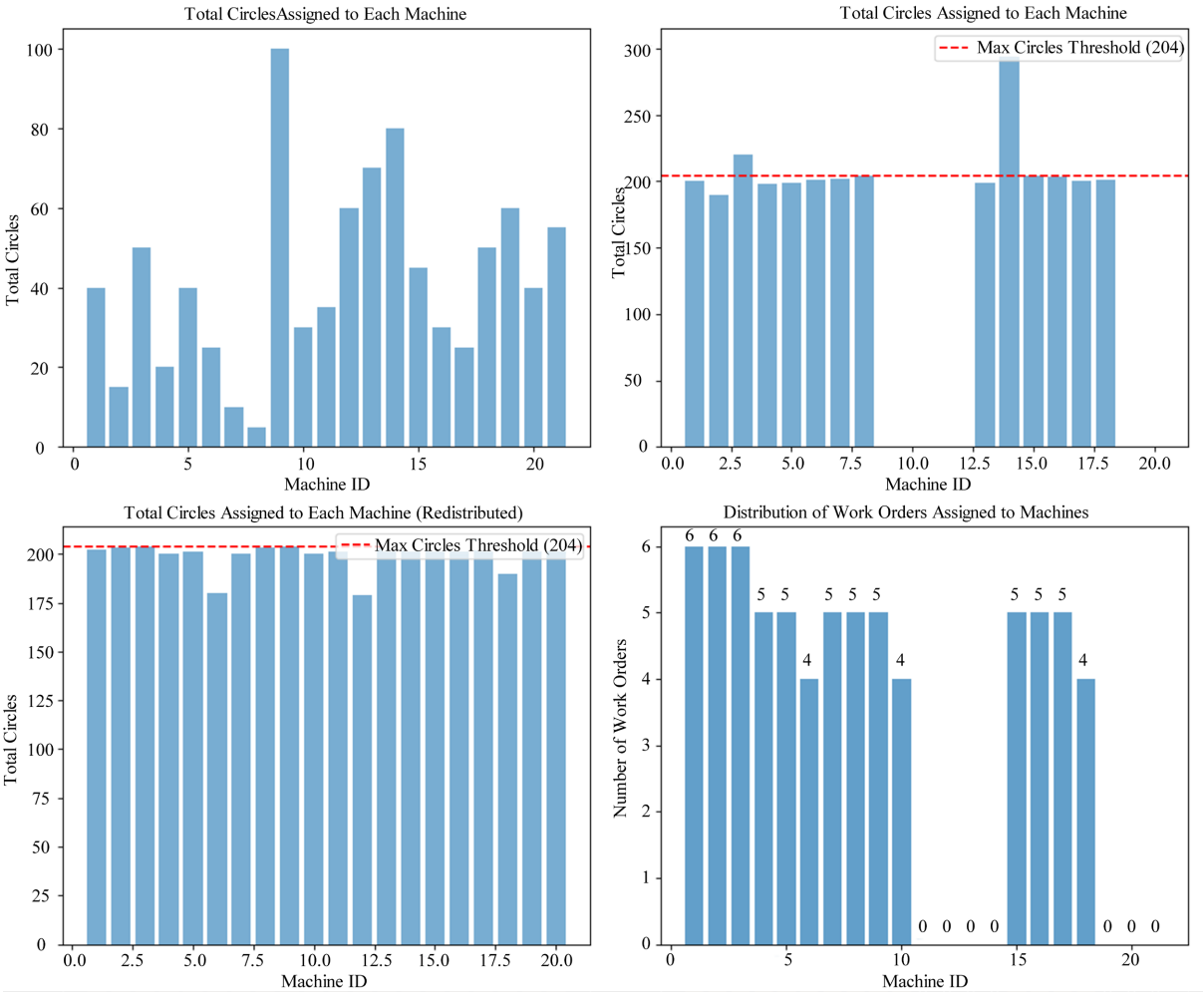

Figure 5. Comparison chart before and after the production scheduling optimizer
图 5. 排产优化器前后对比图

通过优化后的结果可以看出如图 5，工单的分配均衡性得到了显著提升，各机器的负载更加均匀分

布。所有工单的圈数都在限制范围内，且大部分机器的负载接近最大阈值 204 圈，表明资源利用率得到了有效提高。

6. 结束语

本文为提高蜣螂优化算法的优化性能，提出一种多策略改进的蜣螂优化算法(ODBO)，使用标准测试函数验证了改进策略的有效性，将 ODBO 与新型群智能优化算法在 17 个测试函数上的优化结果进行比对，结果表明 ODBO 比原始 DBO 和对比算法具有更高的收敛精度、更快的收敛速度和更强的鲁棒性；再将 ODBO 应用于车间排产优化问题实例，验证了 ODBO 在实际工程方面的适用性和可靠性；在后续的工作中，将继续提高算法的性能，提高算法的效率，减少算法耗时，并考虑把 ODBO 运用在其他实际问题中。通过上述研究与实验，改进的 ODBO 算法在车间生产排产调度问题上展现出了较高的实用性和可靠性。通过实际数据验证，ODBO 算法能够有效减少成型炉的占用位置数，优化生产效率。

参考文献

[1] Gardeyn, J. and Wauters, T. (2022) A Goal-Driven Ruin and Recreate Heuristic for the 2D Variable-Sized Bin Packing Problem with Guillotine Constraints. *European Journal of Operational Research*, **301**, 432-444. https://doi.org/10.1016/j.ejor.2021.11.031

[2] Xue, J. and Shen, B. (2022) Dung Beetle Optimizer: A New Meta-Heuristic Algorithm for Global Optimization. *The Journal of Supercomputing*, **79**, 7305-7336. https://doi.org/10.1007/s11227-022-04959-6

[3] Zhu, F., Li, G., Tang, H., Li, Y., Lv, X. and Wang, X. (2024) Dung Beetle Optimization Algorithm Based on Quantum Computing and Multi-Strategy Fusion for Solving Engineering Problems. *Expert Systems with Applications*, **236**, Article ID: 121219. https://doi.org/10.1016/j.eswa.2023.121219

[4] 周亚中, 何怡刚, 邢致恺, 邵凯旋, 李紫豪, 雷蕾潇. 基于 IDBO-ARIMA 的电力变压器振动信号预测[J]. 电子测量与仪器学报, 2023, 37(8): 11-20.

[5] Shen, Q., Zhang, D., Xie, M. and He, Q. (2023) Multi-Strategy Enhanced Dung Beetle Optimizer and Its Application in Three-Dimensional UAV Path Planning. *Symmetry*, **15**, Article 1432. https://doi.org/10.3390/sym15071432

[6] 吴宇昂, 李永胜, 宣士斌, 等. 基于莱维飞行的自适应乌鸦搜索算法[J]. 微电子学与计算机, 2023, 40(3): 10-19.

[7] 何永康, 李旭芳. 邻域搜索和改进莱维因子的人工蜂鸟优化算法[J]. 建模与仿真, 2024, 13(2): 987-1003.

[8] 郑子威. 基于鲸鱼优化算法的自动排产系统研究与实现[D]: [硕士学位论文]. 邯郸: 河北工程大学, 2021.

[9] 吴超略, 韦文山, 郭羿, 等. 基于 t 分布变异的改进麻雀搜索算法[J]. 信息技术与网络安全, 2022, 41(8): 74-78.

[10] Trojovský, P. and Dehghani, M. (2022) Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications. *Sensors*, **22**, Article 855. https://doi.org/10.3390/s22030855