改进差分进化算法及啤酒灌装机液位控制 PID 参数整定

王超锋[a]，司呈勇[a,b]，沈建强[b]
（上海理工大学 a.光电信息与计算机工程学院 b.中德国际学院，上海 200093）

摘要：目的 针对啤酒液位控制系统存在 PID 参数整定难、非线性、滞后性问题，提出一种改进基于邻域的改进差分进化算法，应用于 PID 参数优化整定中，从而提高灌装机的工作效率和啤酒的质量。**方法** 文中对差分进化算法进行改进，设计一种新型的变异策略，在变异环节引入邻域搜索操作；根据当前种群的分布情况，实时对邻域的个数进行自适应分配，以提升算法全局和局部搜索能力；与 2 种基本差分进化算法和 4 种改进差分进化算法对比，用 18 个测试函数验证文中所提出算法的性能。**结果** 仿真结果表明，相较于基本差分进化算法，使用改进的差分进化算法整定的 PID 参数，调节时间减少 0.22 s，上升时间减少 0.04 s，超调量降低 7.63%。**结论** 通过改进的差分进化算法对啤酒灌装机液位 PID 参数的优化整定，可以显著改善控制系统的超调量、上升时间和稳态误差等性能，实现了液位的稳定控制。

关键词：差分进化算法；啤酒液位控制系统；参数自适应控制；邻域搜索；PID 参数整定
中图分类号：TP273+.2 **文献标识码：**A **文章编号：**1001-3563(2022)19-0310-10
DOI：10.19554/j.cnki.1001-3563.2022.19.038

PID Parameter Setting of Liquid Level Control for Beer Filling Machine Based on Improved Differential Evolution Algorithm

WANG Chao-feng[a], *SI Cheng-yong*[a,b], *SHEN Jian-qiang*[b]
(a. School of Optical-Electrical and Computer Engineering b. Sino-German College, University of Shanghai for Science and Technology, Shanghai 200093, China)

ABSTRACT: The work aims to propose an improved neighborhood based differential evolution algorithm and apply it to PID parameter optimization setting to solve the problems of difficulty, nonlinearity and hysteresis in beer liquid level control system, thus improving the work efficiency of filling machine and the beer quality. The differential evolution algorithm was improved to design a new mutation strategy and introduce the neighborhood search operation in the mutation link. According to the current population distribution, real-time adaptive allocation of the number of neighborhoods was carried out to enhance the global and local search ability of the algorithm. The proposed algorithm was compared with 2 basic differential evolution algorithms and 4 improved differential evolution algorithms and its performance was verified by 18 test functions. From the simulation results, compared with the basic differential evolution algorithm, the PID parameter setting by the improved differential evolution algorithm could reduce the adjustment time by 0.22 s, the rise time by 0.04 s, and the overshoot by 7.63%. Through the improved differential evolution algorithm, the PID parameter setting of the beer filling machine's liquid level is optimized, which significantly improves the overshoot, rise time and steady-state error of the control system and realizes the stable control of the liquid level.

收稿日期：2020–10–13
基金项目：上海市青年科技英才扬帆计划（18YF1417400）
作者简介：王超锋（1997—），男，硕士生，主攻智能优化算法。
通信作者：司呈勇（1986—），男，博士，副教授，主要研究方向为进化计算、约束优化、多目标优化算法等相关人工智能理论及应用。

KEY WORDS: differential evolution algorithm; beer level control system; parameter adaptive control; neighborhood search; PID parameter setting

在整条啤酒包装生产线中，啤酒罐装机扮演着至关重要的角色。灌装机内部的贮液缸液位控制则是一项核心技术。贮液缸液位的精准控制会减少灌装过程中的冒酒、灌不满、液位偏高或低、增氧量和瓶颈空气超出标准等现象发生的概率，从而降低企业包装成本，提高生产效率，因此，贮液缸控制液位的研究显得尤为重要。目前，啤酒生产中液位控制通常采用 PID 控制器进行反馈控制，PID 控制器结构简单、鲁棒性和适应性强，但由于实际对象通常具有非线性、时变不确定性、强干扰等特性，利用常规 PID 控制器难以达到理想的控制效果。PID 控制参数主要有比例、积分和微分系数，如何选取 PID 参数直接影响液位控制效果的好坏。随着经济社会的高质量发展，企业对灌装技术的要求也越来越高，引入进化算法是提高控制精度的一个有效途径。

工程实践中和科学研究中存在很多非线性、非凸、不可微分、多模态、高维等复杂的优化问题，传统数学分析方法如：梯度下降法（Gradient Descent, GD）、共轭方向法（Conjugate Direction Method，GDM）以及拉格朗日乘子法（Lagrange Multiplier Method，LMM）在解决这类问题时效果不佳。进化算法（EA）从自然界中生物进化的得到学习和启发，不需要借助待优化问题的数学特征，并且使用方便、易于理解，鲁棒性较强，已经成为工程技术和研究人员解决优化问题的首选。近些年来，灰狼优化算[1]、布谷鸟算法[2]、萤火虫算[3]等进化算法相继被学者提出。

其中差分进化算法（Differential Evolution, DE）因其具有易于实现、供选择的变体多、收敛速度快、解精度高等优点得到了进化计算领域学者的关注，已经被成功应用到模式识别[4-5]、工业生产设计[6-7]、图像检测[8]、通信系统设计[9]等领域。差分进化算法是一种基于种群的启发式搜索算法，具有对最佳历史位置的记忆功能，可以应用于各种优化场景，包括约束、大规模、多目标、多模态和动态的优化。在 IEEE Congress on Evolutionary Computation （CEC）2021 单目标优化竞赛中，性能最好的前 10 名中有 6 名是在差分进化算法的基础上进行改进，由此可见差分进化算法现在仍然受到学者的青睐，因此，文中采用改进差分进化算法对贮液缸液位 PID 控制参数进行整定，进而提高灌装过程中液位控制的精确度和稳定性。

1　经典 DE

为了寻找非线性、非凸、不可微分、多模态优化问题的全局最优解，Storn 和 Price 于 1995 年提出差分进化算法[10]。差分进化算法是一种基于种群的启发式搜索算法，与其他算法类似，DE 主要由 4 个步骤组成——初始化种群（Initialization）、变异操作（Mutation）、交叉操作（Crossover）以及选择操作（Selection），算法通过不断迭代变异、交叉和选择操作向全局最优解逼近。

1.1　初始化种群

差分进化算法初始化一个包含　个 D 维实值决策变量个体的种群，个体也被称为一条染色体。第 i 个个体表示为 $X_i^t=[X_{i,1}^t,X_{i,2}^t,\cdots,X_{i,D}^t]$，整个种群构成了一个候选解池。初始化的种群应该尽可能地分布在空间，$[X_{\min},X_{\max}]$称为决策空间的上下界，$x_{i,j}^t$ 由式（1）生成。

$$x_{i,j}^t = x_{\min} + rand(0,1)*(x_{\max}-x_{\min}), j \in D \tag{1}$$

式中：$rand[0,1]$为均匀分布的随机数；t 为当前迭代的次数；D 为决策变量的维数；$x_{i,j}^t$ 为决策变量的第 j 维。

1.2　变异操作

变异操作对 DE 的性能有至关重要的影响，DE 的核心思想是利用差分矢量对目标个体 X_i^t 进行变异操作，从而得到新的变异个体V_i^t。

常用 5 种的变异策略如下

1）DE/rand/1：

$$V_i^t = X_i^t + F(X_{r1}{}^t - X_{r2}{}^t) \tag{2}$$

2）DE/best/1：

$$V_i^t = X_{\text{best}}^t + F(X_{r1}{}^t - X_r{}^t) \tag{3}$$

3）DE/current-to-best：

$$V_i^t = X_i^t + F(X_b{}^t - X_i^t) + F(X_{r1}{}^t - X_{r2}{}^t) \tag{4}$$

4）DE/best/2：

$$V_i^t = X_{\text{best}}^t + F(X_{r1}^t - X_{r2}^t) + F(X_{r3}^t - X_{r4}^t) \tag{5}$$

5）DE/rand/2：

$$V_i^t = X_{r1}^t + F(X_{r2}^t - X_{r3}^t) + F(X_{r4}^t - X_{r5}^t) \tag{6}$$

式中：$X_{r1}^t, X_{r2}^t, X_{r3}^t, X_{r4}^t, X_{r5}^t$是从种群中随机选择的个体，索引 $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5$，并且与 X_i^t 不同，X_{best}^t代表种群中适应度值最好的个体，$\Delta p = X_{r1}^t - X_{r2}^t$称为差分矢量，$F \in [0,1]$代表缩放因子，较小的 F 值有利于种群进行细粒化搜索。DE/rand/2 变异策略在二维平面的简单展示见图 1。

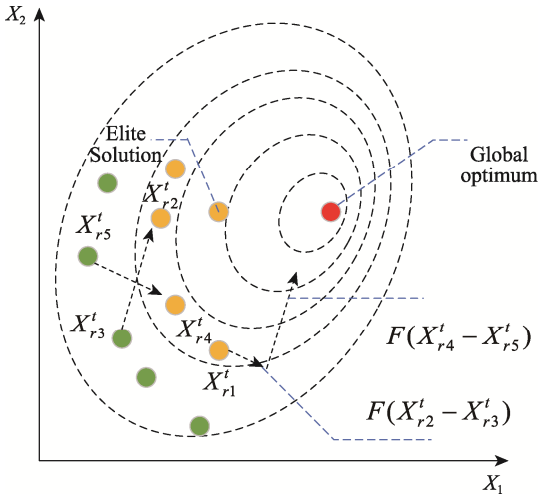

图 1 DE/rand/2 变异策略在二维平面的简单展示
Fig.1 Brief illustration of DE/rand/2 mutation strategy in 2-D parametric space

1.3 交叉操作

在变异操作结束后，交叉操作通过交换个体中某些元素产生试验向量 $U_i^t=[u_{i,1}^t,u_{i,2}^t,\cdots,u_{i,D}^t]$，用于提高种群的多样性。在二进制交叉中，试验个体以如下概率从突变个体中接收随机选择的元素：

$$u_{i,j}^t=\begin{cases}v_{i,j}^t, & \text{if } rand(0,1)<C_R \text{ or } j=j_{rand}\\ x_{i,j}^t, & \text{otherwise}\end{cases} \tag{7}$$

式中：C_R 为交叉概率，通常在[0,1]之间取值；j_{rand} 为[1,D]之间的一个整数以确保试验向量不同于目标向量，较大的 C_R 值有利于算法的勘探能力，较小的 C_R 值有利于算法开发能力。

1.4 选择操作

选择操作在计算目标个体函数值 $f(X_i^t)$ 和试验个体函数值 $f(U_i^t)$ 之后执行方式为：如果试验个体在适应度方面优于或等于目标个体（对于最小化问题），则替换群体中的目标向量。

$$X_i^{t+1}=\begin{cases}U_i^t, & \text{if } f(U_i^t)<f(X_i^t)\\ X_i^t, & \text{otherwise}\end{cases} \tag{8}$$

2 相关工作

和其他算法一样，DE 也存在容易陷入局部最优和过早收敛的情况，许多学者从不同的角度研究如何提高其性能，主要可以包括以下几方面：采用新的初始化种群方法、多子种群多策略、基于参数自适应、利用邻域的信息。荆涛等[11]提出基于蒙特卡洛方法初始化初始种群以提高初始种群的多样性并且在变异环节采用 Lévy flight 概率分布自适应策略，有效的提高了算法的全局搜索能力、鲁棒性能和收敛速度，提高了飞机装配过程中精度同时达到了降低制造成本的目的。李钊等[12]提出基于搜索空间划分与 Canopy K 均值聚类的种群初始化方法，将决策变量的上下界的分割为 n 个子空间，使其可以均匀的分布在决策空间中，并结合 Canopy 和 K 均值聚类搜索到空间中的前景区域，提高了个体寻优的效率，节省计算的时间并且有较好地收敛特性。张静华等[13]提出多种群协同进化的差分进化算法，对种群分别采用不同的策略，利用种群不同个体之间的信息共享机制，吸收适应度值高个体的优点达到共同进化的目的，但缺点是计算时间比较长。孙灿等[14]提出了基于邻域自适应的多策略的差分进化算法（MSDE-NS），根据个体适应度值和社会学中“二八法则”将种群分为 3 个子种群，并分别为每个子种群设置了不同的变异策略、交叉率和变异率。Yi 等[15]提出了一种具有混合变异算子和自适应控制参数的差分进化算法（HSDE），它为每一个个体设置了变异和交叉概率，具有这种反馈机制的自适应控制选择更好的控制参数 F 和 保存到下一代，并且结合 DE/rand/1 和 DE/current-to-best/1 的优点得到混合变异策略，通过竞争的方式决定采用哪种变异策略，这种改进提高了 DE 算法的搜索能力和跳出局部最优的能力。徐英杰等[16]提出基于小波基函数的改进 DE 算法，对缩放因子进行自适应控制，在一定程度上增加算法的多样性，避免了耗时的人为选择参数的问题。Stanovov 等[17]在 Tanabe[18]基础上提出了一种自适应差分进化算法的一种变体（NL-SHADE-RSP），该算法基于保存成功历史的参数和非线性减少种群数目，以保证在进化过程都能获得适合的变异和交叉概率。杨紫晴等[19]提出了 DE 算法和集成协方差矩阵算法（CMADE），融合了 DE 算法的全局收敛能力强和集成协方差局部搜索能力强的优点，取长补短，使算法能够有效的平衡勘探和开发的能力。使用邻域的信息也是提高 DE 算法性能的有效途径之一。

为了提高 DE 算法搜索效率，提出了一种新的变异策略，试图平衡全局搜索和局部搜索。引入个体之间分布信息判断个体收敛性来动态调节算法中的参数，并且对缩放因子和交叉概率采用自适应的策略。

3 改进的差分进化算法

3.1 建立新的变异策略

DE 的性能在很大程度上依赖于变异，差分进化算法的搜索过程从基向量开始，然后将不同的向量作为干扰加入其中，沿着差分向量的方向进行搜索，逐渐向全局最优解收敛。在众多变异策略之中，DE/rand/1 偏向于随机搜索具有很强的多样性，但没有加入种群中最佳解信息，导致后期收敛性能不足；DE/best/1 引入了种群中最佳解信息，但可能会出现陷入局部最优的情况。受 Zhang 等[20]所提出的变异策

略 DE/current-to-pbest/1 和 Wang 等[21]提出的基于邻域的粒子群算法（DNSPSO）的启发，文中提出一种新的变异策略，加入邻域的信息，见式（10）。

DE/current-to-pbest/1 变异公式见式（9）。

$$V_i^t = X_i^t + F(X_{pbest}^t - X_i^t) + F(X_{r1}^t - X_{r2}^t) \tag{9}$$

X_{pbest}^t 是适应度前 100p%的个体，其中 $p \in [0,1]$。

DE/best-to-neighbors/1：

$$V_i^t = X_{nbest}^t + F(X_{best}^t - X_i^t) + F(X_{r1}^t - X_{r2}^t) \tag{10}$$

式中：X_{nbest}^t 为 X_i^t 个体 N 个邻域中适应度值最好的个体；X_{best}^t 为整个种群中适应度值最好的个体，该变异策略结合了局部最优和全局最优解的信息，可以帮助算法跳出局部最优解，表现出较好的收敛性能。N 为 6 时的邻域搜索示意图见图 2。

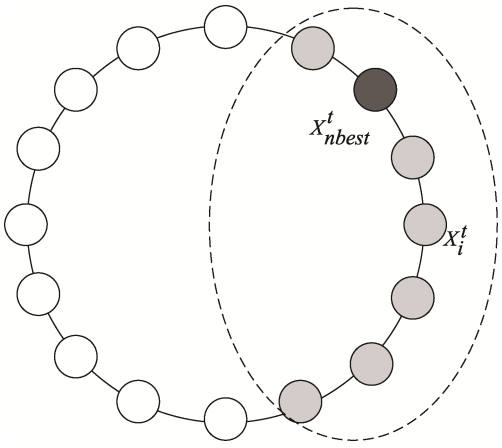

图 2　N 为 6 时的邻域搜索示意图

Fig.2 Schematic diagram of neighborhood search when N is 6

3.2　动态调整邻域因子

其中邻域 N 的大小对算法的开发和探索能力有很大的影响，N 的值较大使算法倾向于 DE/best/1 策略，N 的值较小使算法倾向于 DE/rand/1 策略，因此固定的 N 值不适合于所有函数。由于种群是随机初始化生成，在算法执行的早期阶段种群较为分散，通过变异、选择等操作种群逐渐向全局最优收敛。我们引入个体之间的欧式距离，通过判断种群的分布情况，动态分配的选择邻域个数。在算法整个周期内对于适应度较差个体分配更多的邻居个数，适应度值好的个体则分配更少的邻域个数。

$$Diversity(t) = \frac{1}{N_P}\sum_{i=1}^{N_P}\sqrt{\sum_{j=1}^{D}(X_{i,j}^t - X_{ave}^t)^2} \tag{11}$$

$$X_{ave}^t = \frac{\sum_{i=1}^{N_P} X_{i,j}^t}{N_P} \tag{12}$$

$$N = N_{min} + (N_{max} - N_{min}) * Diversity(t) \tag{13}$$

式中：$Diversity(t)$为第 t 代种群的个体之间距离；X_{ave}^t 为个体的平均位置，通过多次实验比较；N_{max} 和 N_{min} 分别设置为 4 和 12 算法可以取得满意的结果。当算法运行到后期时，理论上 $Diversity(t)$是一个接近于 0 的值。

3.3　F和的 C_R 自适应调整

控制参数 F 和 C_R 的值对同样 DE 性能有着很大的影响，在 2 个参数交叉概率 C_R 和缩放因子 F 之间，C_R 对问题的性质如多模态更为敏感，而 F 对收敛速度更为敏感。根据“天下没有免费的午餐”定理[22]，一个算法在某组优化问题上的性能优于另一个算法，那么在其他优化问题上情况必然相反。如果采用一组固定的参数，算法可能只对某个优化问题有较好的性能，文中采用文献 Jde[23]中所提出自适应改变控制参数的方法，为每一代中的每个个体生成控制参数。

$$C_{Ri}^{t+1} = \begin{cases} r_{and2}, \text{if } f(U_i^t) > f(X_i^t) \\ C_{Ri}^t, \quad \text{otherwise} \end{cases} \tag{14}$$

$$F_i^{t+1} = \begin{cases} F_l + r_{and1} \times F_u, \text{if } f(U_i^t) > f(X_i^t) \\ F_i^t, \quad \text{otherwise} \end{cases} \tag{15}$$

该方法通过和上一代适应度值比较，如果该个体成功进化，则选择这个参数保留至下一代，否则根据上式随机产生一个新的参数。式中：r_{and1} 和 r_{and2} 为 0~1 之间均匀分布的随机数，F_l 和 F_u 是缩放因子 F 的最大和最小值，根据文献中的建议，将 F_l 和 F_u 分别设置为 0.5 和 1。

3.4　算法伪代码

与原始 DE 相比，文中主要从两方面改进：创建一种新的变异方式，融合了邻域和全局最优解的信息，能有效增强种群的多样性；对于邻域个数、缩放因子和交叉概率设置了自适应策略更新参数。

算法 1：NSaDE 伪代码如下：

1.Input：

N_p: population size

D: Problem dimension

F_l: lower of F

F_u: upper of F

T_{max}: Maximum number of function evaluations

2.Randomly initialize N_p individuals

3. Generate control parameter F and C_R for each individual

4. Evaluate the objective function values $f(X_i)$ and FES = N_p

5. **while** FEs < T_{max} **do**

6. Calculate the number of neighbors for each individual according to Eq.(13)

7. **for** i = 1:N_p do

8. Randomly choose two mutually differ-

ent individuals X_{r1}, X_{r2} from N neighbors and X_{nbest}, choose best individual X_{best} from the population

9. Execute the DE/neighbor-to-best/1 according to Eq.(10) to generate a mutate vector V_i

10. Execute the crossover operation to generate a trial vector U_i

11. Evaluate the objection function value $f(U_1), f(U_2),\cdots, f(U_{NP})$

12. FEs = FEs + 1

13. **if** $f(X_i) > f(U_i)$ **then**

14. $X_i = U_i$

15. **else**

16. update C_R and F according to Eq.(14) and Eq.(15)

17. **end if**

18. **end for**

19. **end while**

3.5 数值仿真

仿真实验的运行环境为：AMD Ryzen 7 4800H CPU，主频 2.90 GHz，内存 16 GB, Windows 10，64 位操作系统，实验仿真软件采用 Matlab R2021a。

文中采用的 18 个测试函数来验证所提出算法的有效性，其中 f_1—f_7 为单峰测试函数，可以检验算法收敛速度和精度；f_8—f_{13} 为多模态函数，具有多个局部最优解，可以检测算法跳出局部最优的能力；f_{14}—f_{18} 为 CEC2005 单目标优化竞赛中提出的测试函数，这些函数为带移位的组合函数，其地形较为复杂具有一定的欺骗性，因此有很大的难度。测试函数的函数名、搜索范围以及全局最优值见表 1。

为了评估文中所提出算法的有效性，将与基本 DE，其包含了 2 种常用的变异策略 DE/rand/1 和 DE/best/1；与 4 种改进的算法进行比较，分别是：SaDE[24]、EPSDE [25] 、DEGL[26]、ODE[27]。各算法参数设置如下：DE/rand/1 中 F 和 C_R 分别为 0.5 和 0.9，DE/best/1 中 F 和 C_R 分别为 0.6 和 0.8；SaDE、EPSDE、DEGL、ODE 种群个数 =100，所有函数维数设置为 30 维，其余参数均和原论文中保持一致；适应度函数的最大迭代次数（$T_{\max}$）分别设置为 3×10^5。对每个算法独立运行 30 次，并分别计算获得的函数误差值的平均值和标准偏差。算法优化的结果见表 2 和表 3，采用非参数 Wilcoxon 检验[28]分析算法之间在统计意义上性能差异，其中“−”、“+”和“≈”分别表示算法性能劣于、优于和相当于文中所提出的 NSaDE 算法。

表 1 18 个测试函数

Tab.1 18 test functions

序号	函数名	搜索范围	全局最优值
f_1	Sphere	[−100,100]	0
f_2	Schwefel2.22	[−10,10]	0
f_3	Schwefel1.2	[−100,100]	0
f_4	Schwefel2.21	[−100,100]	0
f_5	Rosenbrock	[−30,30]	0
f_6	Step	[−1.28,1.28]	0
f_7	Quartic with noise	[−100,100]	0
f_8	Schwefel2.26	[−500,500]	$-418.98\cdot D$
f_9	Rastrigin	[−5.12,5.12]	0
f_{10}	Ackley	[−32,32]	0
f_{11}	Griewank	[−600,600]	0
f_{12}	Penalized1	[−50,50]	0
f_{13}	Penalized2	[−50,50]	0
f_{14}	Shift Sphere Function	[−100,100]	−450
f_{15}	Shift Schwefels Problem 1.2	[−100,100]	−450
f_{16}	Shift Rotated High Conditioned Elliptic Function	[−100,100]	−450
f_{17}	Shift Schwefels Problem 1.2 with Noise in Fitness	[−100,100]	−450
f_{18}	Schwefels Problem 2.6 with Global Optimum on Bounds	[−100,100]	−310

表 2　NSaDE 与基本 DE 对比实验结果

Tab.2 Experimental results of comparison between NSADE and basic DE algorithm

函数	DE/rand/1		DE/best/1		NSaDE	
	平均值	标准差	平均值	标准差	平均值	标准差
f_1	3.15×10^{-36}	1.55×10^{-36}–	4.39×10^{-279}	0+	4.51×10^{-105}	3.01×10^{-105}
f_2	5.54×10^{-17}	3.49×10^{-17}–	7.48×10^{-87}	1.37×10^{-86}+	1.18×10^{-51}	1.07×10^{-48}
f_3	3.71×10^{-2}	2.58×10^{-2}–	2.22×10^{-53}	6.83×10^{-51}+	3.98×10^{-16}	1.57×10^{-16}
f_4	2.84×10^{-3}	2.77×10^{-3}–	1.06×10^{-10}	6.59×10^{-10}–	1.17×10^{-11}	6.58×10^{-11}
f_5	3.19×10^{-2}	5.33×10^{-2}–	6.35×10^{-20}	1.88×10^{-20}–	1.43×10^{-25}	3.52×10^{-25}
f_6	0	0≈	7.61×10^{1}	$8.53\mathrm{E}\times10^{1}$–	0	0
f_7	7.53×10^{-3}	1.47×10^{-3}–	4.12×10^{-3}	4.68×10^{-3}–	2.47×10^{-5}	8.07×10^{-6}
f_8	3.82×10^{-4}	6.02×10^{-4}–	1.18×10^{2}	$6.23\mathrm{E}\times10^{2}$–	3.82×10^{-4}	0
f_9	1.40×10^{-7}	2.34×10^{-7}–	9.95×10^{-1}	1.83×10^{0}–	0	0
f_{10}	5.15×10^{-15}	4.47×10^{-15}–	2.22×10^{0}	1.37×10^{0}–	3.55×10^{-15}	0
f_{11}	0	0≈	7.40×10^{-3}	4.98×10^{-2}–	0	0
f_{12}	1.64×10^{-32}	5.16×10^{-48}+	3.70×10^{-12}	2.84×10^{-12}–	2.35×10^{-32}	9.36×10^{-35}
f_{13}	1.84×10^{-32}	4.57×10^{-32}≈	1.45×10^{0}	$1.85\mathrm{E}\times10^{0}$–	1.58×10^{-33}	5.47×10^{-32}
f_{14}	0	0≈	5.63×10^{-13}	3.62×10^{-13}–	0	0
f_{15}	3.17×10^{-5}	7.87×10^{-5}–	1.18×10^{-11}	7.26×10^{-12}–	1.56×10^{-13}	8.19×10^{-13}
f_{16}	5.23×10^{5}	1.86×10^{5}–	1.29×10^{5}	1.08×10^{5}≈	1.56×10^{5}	1.01×10^{5}
f_{17}	2.92×10^{-2}	2.54×10^{-2}–	2.05×10^{2}	$3.15\mathrm{E}\times10^{2}$–	4.11×10^{-4}	6.38×10^{-4}
f_{18}	5.85×10^{1}	6.45×10^{1}+	2.13×10^{1}	7.24×10^{1}+	1.22×10^{2}	1.22×10^{2}
–/+/≈	12/2/4		13/4/1		––	

从表 2 中可以看出，与 DE/rand/1 相比，NSaDE 在 18 个测试函数上有 12 个取得了更好的结果，4 个测试函数上性能接近，仅有 2 个结果比 NSaDE 更优。与 DE/best/1 相比，NSaDE 在 13 个测试函数上取得了更好的结果，而 DE/best/1 在仅 4 个测试函数上表现的比 NSaDE 更优，f_1、f_2 和 f_3 是单峰问题，f_{18} 是移位单峰函数，原因是 DE/best/1 作为一种贪婪策略对这些简单的单峰和多峰问题具有更好的勘探能力，但是在处理具有多个局部最优问题时，可能使算法陷入停滞状态。在表 3 中与 4 种改进算法对比结果看出，对于绝大多数测试函数 NSaDE 取得的结果具有明显的优势。综上所述，文中所提出 NSADE 算法有较强的全局搜索能力和较好的稳定性。

4　啤酒灌装机贮液缸液位 PID 优化控制

为了验证文中提出的改进差分进化算法在啤酒液位灌装机贮液缸控制系统上的有效性和可靠性，在 Matlab/Simulink 中进行仿真实验。传统的 PID 控制原理见图 3，采用比例、积分、微分调节，其中比例系数（K_p）、积分系数（T_i）、和微分系数（T_d）对系统的性能有很大的影响，只有 3 个参数合理配置时，系统才能处于最佳的状态。

文中使用改进的差分进化算法对参数进行整定，旨在使系统的超调量（M_p）、调节时间（t_s）、稳态误

表 3 NSaDE 与 4 种改进算法对比实验结果

Tab.3 Experimental results of comparison between NSADE and 4 improved algorithm

函数	ODE		DEGL		SaDE		EPSDE		NSaDE	
	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差	平均值	标准差
f_1	8.94×10^{-58}	3.18×10^{-57}–	6.01×10^{-101}	2.10×10^{-100}–	2.14×10^{-130}	5.47×10^{-130}+	8.02×10^{-170}	0+	4.51×10^{-105}	3.01×10^{-105}
f_2	5.23×10^{-18}	4.59×10^{-18}–	1.63×10^{-49}	1.53×10^{-49}–	3.56×10^{-79}	2.59×10^{-79}+	5.65×10^{-86}	3.24×10^{-85}+	1.18×10^{-51}	1.07×10^{-48}
f_3	3.16×10^{-5}	3.25×10^{-5}–	3.35×10^{-24}	6.82×10^{-24}+	1.93×10^{-6}	1.28×10^{-6}–	4.38×10^{-36}	2.84×10^{-35}+	3.98×10^{-16}	1.57×10^{-16}
f_4	1.75×10^{-3}	1.27×10^{-2}	5.18×10^{-25}	9.15×10^{-25}+	3.21×10^{-7}	5.49×10^{-6}–	2.65×10^{0}	1.83×10^{0}–	1.17×10^{-11}	6.58×10^{-11}
f_5	2.44×10^{1}	8.21×10^{1}–	6.64×10^{-1}	1.49×10^{0}–	2.24×10^{1}	1.89×10^{1}–	3.99×10^{-1}	1.20×10^{0}–	1.43×10^{-25}	3.52×10^{-25}
f_6	0	0≈	0	0≈	0	0≈	0	0≈	0	0
f_7	9.10×10^{-4}	3.12×10^{-4}+	1.20×10^{-3}	3.52×10^{-4}+	2.86×10^{-3}	1.89×10^{-3}–	8.87×10^{-4}	3.27×10^{-4}≈	2.47×10^{-5}	8.07×10^{-4}
f_8	6.84×10^{3}	5.75×10^{2}–	7.30×10^{3}	2.94×10^{2}–	3.83×10^{3}	0≈	3.82×10^{-4}	0≈	3.82×10^{-4}	0
f_9	3.26×10^{1}	1.35×10^{1}	1.01×10^{-2}	5.12×10^{1}–	3.56×10^{-2}	4.98×10^{-1}–	0	0≈	0	0
f_{10}	3.51×10^{-15}	0≈	3.67×10^{-15}	6.38×10^{-16}≈	9.32×10^{-2}	2.56×10^{-1}–	4.86×10^{-15}	1.71×10^{-15}–	3.55×10^{-15}	0
f_{11}	7.29×10^{-4}	1.35×10^{-3}–	3.61×10^{-3}	5.46×10^{-3}–	3.56×10^{-3}	6.02×10^{-3}–	7.40×10^{-4}	2.22×10^{-3}–	0	0
f_{12}	1.35×10^{-32}	$\times4.57\times10^{-32}$–	1.57×10^{-32}	5.47×10^{-48}≈	2.81×10^{-2}	5.22×10^{-2}–	1.57×10^{-32}	2.13×10^{-35}≈	2.35×10^{-32}	9.36×10^{-35}
f_{13}	1.56×10^{-32}	7.85×10^{-48}–	3.66×10^{-4}	1.97×10^{-3}≈	1.25×10^{-3}	8.52×10^{-3}–	3.66×10^{-4}	1.97×10^{-3}–	1.58×10^{-33}	5.47×10^{-32}
f_{14}	0	0≈	3.79×10^{-15}	1.42×10^{-14}	0	0≈	5.68×10^{-15}	1.71×10^{-15}–	0	0
f_{15}	3.32×10^{-4}	5.24×10^{-4}–	4.74×10^{-15}	2.12×10^{-13}–	8.45×10^{-6}	5.27×10^{-5}–	1.74×10^{-12}	4.60×10^{-12}–	1.56×10^{-13}	8.19×10^{-14}
f_{16}	5.97×10^{5}	4.65×10^{5}–	5.80×10^{4}	3.44×10^{4}–	5.23×10^{5}	5.12×10^{5}–	1.84×10^{6}	4.73×10^{6}–	1.56×10^{5}	1.01×10^{4}
f_{17}	2.14×10^{-1}	2.47×10^{-1}–	6.44×10^{-14}	3.51×10^{-14}+	4.14×10^{2}	1.48×10^{2}–	2.51×10^{1}	1.18×10^{2}–	4.11×10^{-4}	6.38×10^{-4}
f_{18}	1.45×10^{2}	7.03×10^{2}–	5.80×10^{4}	2.60×10^{-1}+	3.30×10^{3}	4.36×10^{2}–	1.98×10^{3}	2.59×10^{2}–	1.22×10^{2}	1.95×10^{2}
–/+/≈	14/1/3		9/5/4		13/2/3		8/3/5		—	

差（E_{ss}）最小化。改进的 PID 控制原理见图 4，图 4 中 K_p、T_i、T_d 是 3 个需要优化的参数。文中采用黄卓超等[29]给出的传递函数形式，式中 K 为过程放大倍数，T 为系数惯性时间常数。在实际工程中由于考虑了储能元件，液位的变化会存在滞后性，因此加入延迟环节 $e^{-\tau s}$，τ 为延迟时间，具体形式见式（18）。

$$G(s)=\frac{K}{Ts+1}e^{-\tau s} \tag{16}$$

$$G(s)=\frac{4.756}{160.436s+1}e^{-3s} \tag{17}$$

采用传统的 Ziegler–Nichols（Z–N）法、基本 DE 算法和文中改进的 NSaDE 算法优化的 PID 控制对啤酒灌装机贮液缸液位进行仿真对比试验。实验中，选择绝对误差积分（ITAE）准则作为改进差分进化算法的适应度函数，见式（18），其值越小越好。通过实数编码将 K_p、T_i、T_d 3 个参数作为一个个体，以 ITAE 作为算法评估函数，通过不断迭代搜索一组最佳的 PID 控制参数。

$$J=\int_0^{\infty}t\left|e(t)\right|\mathrm{d}t \tag{18}$$

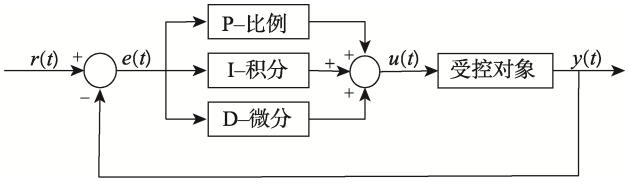

图 3　PID 控制系统原理
Fig.3 Principle of PID control system

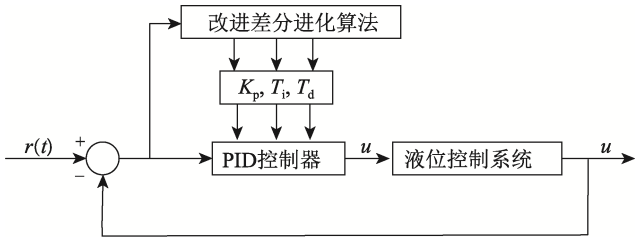

图 4　基于改进差分进化算法的 PID 优化结构
Fig.4 PID optimization structure based on improved differential evolution algorithm

仿真参数设置：以单位阶跃信号作为系统输入，K_p、T_i、T_d 上下界设置为[0.001,100]，仿真时间为 10 s，采样频率为 100 Hz；基本 DE 算法种群规模 N_P=50，F 和 C_R 分别设置为 0.7 和 0.9；NSaDE 种群规模 N_P=50，最大迭代次数（T_{max}）为 1 500 次。

从表 4 的结果来看，经 NSaDE 法整定的系统超调量较小，响应速度更快，调节时间相比于 Z–N 和 DE 法分别缩短了 59.71%和 10.04%，上升时间相比于 Z–N 和 DE 法缩短了 12.5 和 3.44%。从图 5 中可以看出，NSaDE 方法可以获得更好的整定结果。传统的 Z–N 法获得的系统响应曲线振荡次数较多并且始终存在稳态误差，而 DE 和 NSaDE 实现了零误差，具有更高的控制精度。综上所述，采用 NSaDE 方法整定的 PID 参数对啤酒灌装机贮液缸液位能进行稳定的控制，在目标值追踪效果上好于 Z–N 法和基本 DE 方法，其控制器性能能够实现啤酒灌装机的生产要求。

表 4　3 种方法主要性能指标对比
Tab.4 Comparison of main performance indexes of 3 methods

整定方法	K_p	K_i	K_d	适应度值	稳态误差	调节时间（2%）/s	上升时间/s	超调量/%
Z–N	4.96	0.35	0.105	––	0.02	4.88	1.28	53.77
DE	5.93	0.47	1.089	0.164	0	2.19	1.16	11.58
NSaDE	2.257	0.861	0.712	0.042	0	1.97	1.12	3.95

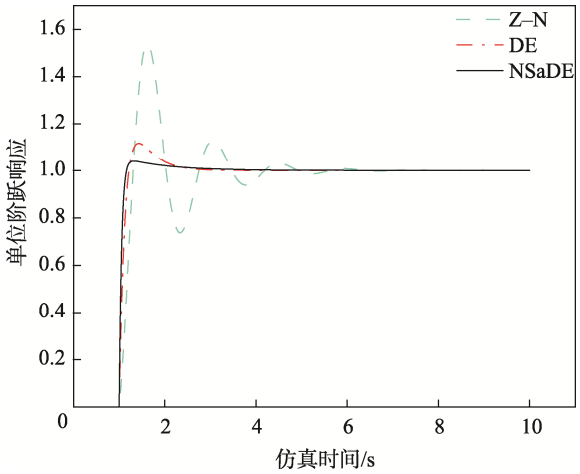

图 5　Z–N 法、DE 法和 NSaDE 法整定后的单位响应曲线
Fig.5 Comparison of setting results between Z-N and DE and NsaDE

5　结语

为了克服差分进化算法搜索性能、容易陷入局部最优解的不足，文中提出了一种基于邻域的自适应差分进化算法——NSaDE 算法。算法设计了一种新的变异方式，引入种群信息对邻域的个数进行自适应更新来提升算法局部和全局搜索的性能。在 18 个测试函数上进行仿真，与 DE/rand/1、DE/best/1、SaDE、EPSDE、DEGL、ODE 算法进行对比，在收敛速度、寻优精度都取得了较好的结果。将改进算法用于解决啤酒灌装机 PID 参数的优化问题，取代了烦琐的反复实验，节约了时间成本。仿真结果表明使用改进算法得到 PID 参数的系统有着更小的超调量、更快的响应速度，能够实现对液位的精准控制，保证啤酒罐装过程中实现稳定运行和啤酒生产的质量和产量。

参考文献：

[1] SEYEDALI M, SEYED MOHAMMAD, ANDREW LEWSIA. Grey Wolf Optimizer[J]. Advances in Engineering Software, 2014: 46-61.

[2] GANDOMI A H, YANG Xin-she, ALAVI A H. Erratum To: Cuckoo Search Algorithm: A Metaheuristic Approach to Solve Structural Optimization Problems[J]. Engineering With Computers, 2013, 29(2): 245.

[3] YANG Xin she. Firefly Algorithm, Stochastic Test Functions and Design Optimization[J]. International

Journal of Bio-Inspired Computation, 2010, 2(2): 78-84.

[4] WANG Li-zhou, HONG Guang-hao. A Singular Critical Point for some Variational Free Boundary Problem in Five Dimensions[J]. Journal of Mathematical Analysis and Applications, 2015, 426(2): 855-863.

[5] 何庆飞, 陈小虎, 姚春江, 等. 基于最小二乘支持向量分类机的齿轮泵故障诊断研究[J]. 流体机械, 2019(8): 32-36.
HE Qing-fei, CHEN Xiao-hu, YAO Chun-jiang, et al. Gear Pump Fault Diagnosis Research Based on Least Squares Support Vector Classification Machine[J]. Fluid Machinery, 2019(8): 32-36.

[6] SEGUNDO E, AMOROSO A L, MARIANI V C, et al. Economic Optimization Design for Shell-and-Tube Heat Exchangers by a Tsallis Differential Evolution[J]. Applied Thermal Engineering, 2017, 111: 143-151.

[7] AYALA H, VICENTE H, MORAIS, et al. Design of Heat Exchangers Using a Novel Multi-Objective Free Search Differential Evolution Paradigm[J]. Applied THERMAL ENGINEERING: Design, Processes, Equipment, Economics, 2016, 94: 170-177.

[8] ZHOU Zhi-li, YANG C N, CHEN Bei-jing, et al. Effective and Efficient Image Copy Detection with Resistance to Arbitrary Rotation[J]. IEICE Transactions on Information and Systems, 2016, 99(6): 1531-1540.

[9] ZHANG, WU Z. Spectrum Allocation by Wave Based Adaptive Differential Evolution Algorithm[J]. Ad hoc networks, 2019, 94: 1-8.

[10] 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.

[11] 荆涛, 田锡天. 基于蒙特卡洛-自适应差分进化算法的飞机容差分配多目标优化方法[J]. 航空学报, 2022, 35(3): 569-580.
JING Tao, TIAN Xi-tian. Multi-Objective Optimization Method for Aircraft Tolerance Allocation Based on Monte Carlo-Adaptive Differential Evolution Algorithm[J]. Acta Aeronautica et Astronautica Sinica, 2022, 35(3): 569-580.

[12] 李钊, 袁文浩, 任崇广. 基于搜索空间划分与 Canopy K-means 聚类的种群初始化方法[J]. 控制与决策, 2020, 35(11): 2767-2772.
LI Zhao, YUAN Wen-hao, REN Chong-guang. Population Initialization Based on Search Space Partition and Canopy Kmeans Clustering[J]. Control and Decision, 2020, 35(11): 2767-2772.

[13] ZHANG Jing-hua, HAN Pu. Multi-Algorithm and Multi-Population Co-Optimization Differential Evolution Algorithm[J]. Journal of System Simulation, 2018, 30(5): 1690-1699.

[14] SUN Can, ZHOU Xin-yu, Wang Ming-wen. A Multi-Strategy Differential Evolution Algorithm Combined with Neighborhood Search[J]. Journal of System Simulation, 2020, 32(6): 1071-1084.

[15] YI Wen-chao, GAO Liang, LI Xin-yu, et al. A New Differential Evolution Algorithm with a Hybrid Mutation Operator and Self-Adapting Control Parameters for Global Optimization Problems[J]. Applied Intelligence, 2015, 42(4): 642-660.

[16] 徐英杰, 阎晓琳, 邓武. 基于小波基函数的差分进化算法缩放因子改进方法及应用[J]. 云南民族大学学报(自然科学版), 2020, 119(1): 33-38.
XU Ying-jie, YAN Xiao-lin, DENG Wu. Scaling Factor Improvement Method of Differential Evolution Based on Wavelet Basis Function and Its Application[J]. Journal of Yunnan Minzu University (Natural Sciences Edition), 2020, 119(1): 33-38.

[17] STANOVOV V, AKHMEDOVA S, SEMEKIN E. NL-SHADE-RSP Algorithm with Adaptive Archive and Selective Pressure for CEC 2021 Numerical Optimization[C]// 2021 IEEE Congress on Evolutionary Computation (CEC), IEEE, Kraków Poland, 2021: 17-24.

[18] TANABE R, FUKUNAGA A. Success-History Based Parameter Adaptation for Differential Evolution[C]// 2013 IEEE Congress on Evolutionary Computation, 2013: 71-78.

[19] 杨紫晴, 姚加林, 伍国华, 等. 集成协方差矩阵自适应进化策略与差分进化的优化算法[J]. 控制理论与应用, 2021, 36(10): 1493-1502.
YANG Zi-qing, YAO Jia-lin, WU Guo-hua, et al. Ensemble Optimization Algorithm from Covariance Matrix Adaptive Evolution Strategy and Differential Evolution[J]. Control Theory & Applications, 2021, 36(10): 1493-1502.

[20] ZHANG Jing-qiao, SANDERSON A C. JADE: Adaptive Differential Evolution with Optional External Archive[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 945-958.

[21] WANG Hui, SUN Hui, LI Chang-he, et al. Diversity Enhanced Particle Swarm Optimization with Neighborhood Search[J]. Information Sciences, 2013, 223: 119-135.

[22] DAVID H WOLPERT, WILLIAM G. MACREADY No Free Lunch Theorems for Optimization[J]. IEEE Transactions on Evolutionary Computation (S1089-778X), 1997, 1(1): 67-82.

[23] BREST J, GREINER S, BOSKOVIC B, et al. Self- Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems[J]. IEEE Trans Evolutionary Computation, 2006, 10(6): 646-657.

[24] QIN A K, HUANG V L, SUGANTHAN P N. Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(2): 398-417.

[25] MALLIPEDDI R, SUGANTHAN P, PAN Q, et al. Differential Evolution Algorithm with Ensemble of Parameters and Mutation Strategies[J]. Applied Soft Computing Journal, 2010, 11(2): 1679-1696.

[26] DAS S, ABRAHAM A. Differential Evolution Using a Neighborhood-Based Mutation Operator[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(3): 526-553.

[27] RAHNAMAYAN S, TIZHOOSH H R, SALAMA M M A. Opposition-Based Differential Evolution[J]. IEEE Transactions on Evolutionary Computation, 2014, 12(1): 64-79.

[28] DERRAC J, GARCÍA S, MOLINA D, et al. A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms[J]. Swarm and Evolutionary Computation, 2011, 1(1): 3-18.

[29] 黄卓超, 张伟, 王亚刚. 改进粒子群算法的啤酒灌装机液位控制 PID 参数整定[J]. 包装工程, 2020, 41(19): 159-165.
HUANG Zhuo-chao, ZHANG Wei, WANG Ya-gang. PID Parameter Setting of Liquid Level Control for Beer Filling Machine Based on Improved Particle Swarm Algorithm[J]. Packaging Engineering, 2020, 41(19): 159-165.

责任编辑：曾钰婵