Journal of System Simulation

Volume 32 | Issue 6 Article 10

6-25-2020

A Multi-strategy Differential Evolution Algorithm Combined with Neighborhood Search

Can Sun
School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China;

Xinyu Zhou
School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China;

Mingwen Wang
School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China;

A Multi-strategy Differential Evolution Algorithm Combined with Neighborhood Search

Abstract

Abstract: The difficulties of designing a multi-strategy differential evolution (DE) algorithm are how to select the mutation strategies and allocate these strategies. A multi-strategy DE algorithm combined with the neighborhood search operator is proposed. The population is divided into three subpopulations according to the fitness values, and each subpopulation employs a different mutation strategy and parameter settings to complement the search ability, to balance the exploration and exploitation ability of the whole population. The subpopulation with the best fitness values employs the neighborhood search operator to exploit possible benefit information to guide the search. Extensive experiments are carried out on 34 test functions to compare with 12 different evolutionary algorithms, which include the 7 DE algorithms. The results show that the algorithm can perform better on most test functions.

Keywords

differential evolution, multiple strategies, neighborhood search, exploration ability, exploitation ability

Recommended Citation

Sun Can, Zhou Xinyu, Wang Mingwen. A Multi-strategy Differential Evolution Algorithm Combined with Neighborhood Search[J]. Journal of System Simulation, 2020, 32(6): 1071-1084.

第 32 卷第 6 期 系统仿真学报© Vol. 32 No. 6
2020 年 6 月 Journal of System Simulation Jun., 2020

一种融合邻域搜索的多策略差分进化算法

孙灿，周新宇，王明文
（江西师范大学计算机信息工程学院，江西 南昌 330022）

摘要：设计多策略差分进化算法的难点在于选择何种变异策略以及如何分配这些策略。*提出一种融合邻域搜索的多策略差分进化算法，根据个体适应度值将种群分为 3 个子种群，每个子种群分别采用不同的变异策略和参数值，使得各子种群的搜索能力可互补，有助于平衡整个种群的勘探和开采能力。同时，对适应度值最好的子种群采用邻域搜索操作，充分挖掘优质个体可能包含的有益信息用于指导搜索。*在 34 个测试函数上实验，与包含 7 种差分进化算法在内的 12 种进化算法进行对比，结果表明该算法在大多数函数上取得了更好性能。

关键词：*差分进化；多策略；邻域搜索；勘探能力；开采能力*
中图分类号：TP391.9 文献标识码：A 文章编号：1004-731X (2020) 06-1071-14
DOI: 10.16182/j.issn1004731x.joss.18-0787

A Multi-strategy Differential Evolution Algorithm Combined with Neighborhood Search

Sun Can, *Zhou Xinyu*, *Wang Mingwen*
(School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China)

Abstract: The difficulties of designing a multi-strategy differential evolution (DE) algorithm are how to select the mutation strategies and allocate these strategies. *A multi-strategy DE algorithm combined with the neighborhood search operator is proposed. The population is divided into three subpopulations according to the fitness values, and each subpopulation employs a different mutation strategy and parameter settings to complement the search ability, to balance the exploration and exploitation ability of the whole population. The subpopulation with the best fitness values employs the neighborhood search operator to exploit possible benefit information to guide the search.* Extensive experiments are carried out on 34 test functions to compare with 12 different evolutionary algorithms, which include the 7 DE algorithms. The results show that the algorithm can perform better on most test functions.

Keywords: differential evolution; multiple strategies; neighborhood search; exploration ability; exploitation ability

引言

差分进化(Differential Evolution，DE)算法是一种基于群体的随机搜索算法[1-2]，它采用变异、交叉、选择 3 种进化操作来生成后代个体，其结构简单、易于实现、且性能优秀，是近年来较为流行的一种进化算法，受到了众多领域中研究人员的广泛关注，已在数字滤波器设计、图像处理和数据挖掘等多方面得到了成功应用[3-4]。

经典 DE 的算法结构仅包含了一种变异策略和固定参数值，虽然该方式可降低算法结构的复杂度，有利于保持算法简洁，但对于复杂的优化问题

收稿日期：2018-11-23 修回日期：2019-07-15；
基金项目：国家自然科学基金(61603163，61966019，61876074)，江西省自然科学基金(20192BAB207030)；
作者简介：孙灿(1989-)，男，河南，硕士生，研究方向为进化算法及其应用；周新宇(通讯作者 1987-)，男，江西，博士，副教授，硕导，研究方向为进化算法及其应用。

而言，特别是多峰类型的优化问题，经典 DE 的求解效果往往难以令人满意。通常，不同类型的变异策略和参数配置适合求解的问题类型一般不同，甚至对同一问题，同一算法在不同的运行阶段所适合采取的变异策略和参数配置也有所不同。例如，DE/best/1 因包含全局最优个体，使得该策略侧重于开采，适合求解单峰类型的问题；而 DE/rand/1 包含的个体均为随机选择，使得该策略的搜索方向具备无偏性，侧重于勘探，适合求解多峰类型的问题。

为增强 DE 性能，多种不同类型的改进 DE 相继被提出，其中主流的改进思路是采用多种变异策略和参数配置。例如，Mallipeddi 等[5]提出了一种集成式的 EPSDE 算法，构建了策略候选池和参数候选池，从中选择策略和参数进行组合，再根据组合的历史表现来设置其被选用的概率；Wang 等[6]提出了一种组合式的 CoDE 算法，将 3 种不同类型的变异策略和 3 组参数配置进行随机组合，用于同时生成 3 个试验个体，再从中选择最好的一个作为后代个体；Zhou 等[7]提出了一种基于角色分配机制的 RADE 算法，根据个体的适应度和个体间的欧式距离对个体进行角色分配，把种群中所有个体划分为 3 类角色，再结合角色的特点来选择不同的变异策略和参数配置。

毋庸置疑，以上述算法为代表的多策略 DE 表明多策略机制可有效提高算法性能，但选择何种变异策略，又如何分配这些策略仍是设计高效多策略 DE 的难点。为此，提出一种融合邻域搜索的多策略 DE 算法(简称 MSDE-NS)。该算法将种群中所有个体按适应度值进行排序，把种群划分为 3 个子种群，每个子种群分别采用不同的变异策略和参数值，使得各子种群具备不同的搜索能力，实现搜索能力相互协调和互补，有利于平衡整个种群的勘探和开采能力。同时，为进一步发挥种群中优质个体的引领作用，为适应度值最好的子种群引入了邻域搜索操作，将部分计算资源用于搜索优质个体的邻域空间，充分挖掘其可能包含的有益信息，加快算法的收敛速度。在 34 个典型的测试函数(包括单峰、多峰、偏移、以及旋转类型)上进行实验，与包含 7 种 DE 算法在内的 12 种进化算法进行对比，结果表明本文算法在大部分测试函数上能获得更好性能。

1 经典 DE

DE 是较为流行的一种进化算法范例，与其他进化算法的结构类似，DE 包含了变异、交叉、以及选择 3 种进化操作。假设 DE 的种群由 NP 个个体组成，第 i 个个体是 $X_i^G=[x_{i,1}^G,x_{i,2}^G,\cdots,x_{i,D}^G]$，其中 D 为优化问题的维度，G 为当前进化代数。DE 的 3 种进化操作介绍如下。

1.1 变异操作

变异操作的主要目的是生成变异个体，使得变异个体可通过交叉操作与目标个体生成试验个体。变异操作是 DE 的亮点，也是区别于其他进化算法范例的关键。假设生成的变异个体是 $V_i^G=[v_{i,1}^G,v_{i,2}^G,\cdots,v_{i,D}^G]$，DE 常用的 5 种变异策略分别是[8]：

1) DE/rand/1

$$V_i^G=X_{r1}^G+F\cdot(X_{r2}^G-X_{r3}^G) \tag{1}$$

2) DE/rand/2

$$V_i^G=X_{r1}^G+F\cdot(X_{r2}^G-X_{r3}^G)+F\cdot(X_{r4}^G-X_{r5}^G) \tag{2}$$

3) DE/best/1

$$V_i^G=X_{\text{best}}^G+F\cdot(X_{r1}^G-X_{r2}^G) \tag{3}$$

4) DE/best/2

$$V_i^G=X_{\text{best}}^G+F\cdot(X_{r1}^G-X_{r2}^G)+F\cdot(X_{r3}^G-X_{r4}^G) \tag{4}$$

5) DE/current-to-best/1

$$V_i^G=X_i^G+F\cdot(X_{best}^G-X_i^G)+F\cdot(X_{r1}^G-X_{r2}^G) \tag{5}$$

式中：X_{r1}^G，X_{r2}^G，X_{r3}^G，X_{r4}^G，X_{r5}^G 是从种群中随机选择的互不相同的个体，并且与目标个体 X_i^G 不相同，X_{best}^G 表示当前种群中适应度最好的个体。参数 F 是尺度系数，用于控制差分向量的步长，其值常取自于区间[0, 1]，较大的 F 值可增

DOI: 10.16182/j.issn1004731x.joss.18-0787

强算法的勘探能力，而较小的 F 值有助于增强开采能力。

1.2 交叉操作

在变异操作生成了变异个体之后，DE 执行交叉操作来生成试验个体。假设试验个体是 $U_i^G = [u_{i,1}^G, u_{i,2}^G, \cdots, u_{i,D}^G]$，常用的二项式交叉操作如下所示[8]：

$$u_{i,j}^G = \begin{cases} v_{i,j}^G & \text{if } rand_j(0,1) \le CR \text{ or } j = jrand \\ x_{i,j}^G & \text{otherwise} \end{cases} \quad (6)$$

式中：参数 CR 是交叉概率，常在[0, 1]内取值，$rand_j(0,1)$是[0, 1]内的随机数，$jrand$ 是随机选择的一个维度，可确保试验个体至少有一维来自于变异个体，从而使得试验个体不同于目标个体。通常，较大的 CR 值有助于增加种群多样性，而较小的 CR 值可加快收敛。

1.3 选择操作

在生成试验个体之后，DE 采用选择操作从试验个体和目标个体中根据适应度选出较优的一个进入下一次迭代。假设适应度函数是 $f(\cdot)$，对最小化优化问题而言，适应度值越小越好，因此选择操作可形式化表达如式(7)所示[8]：

$$X_i^{G+1} = \begin{cases} U_i^G & \text{if } f(U_i^G) < f(X_i^G) \\ X_i^G & \text{otherwise} \end{cases} \quad (7)$$

2 相关工作

多策略DE的研究重点在于如何应用不同的策略。在已有的相关研究工作中，大部分工作可归纳为 3 类[9]：

1) 基于概率模型的多策略 DE；
2) 基于多策略共存的多策略 DE；
3) 基于多子种群的多策略 DE。

在概率模型方式中，通常先根据策略在前期搜索过程中积累的成功历史经验来计算策略在后期被选中的概率，再按该概率为个体选择不同的策略。在多策略共存方式中，通常先对个体同时应用多个策略用于生成多个试验个体，再从中选择适应度最优的试验个体作为后代个体。在多子种群方式中，通常先按某一规则把种群划分为多个子种群，再按子种群的需求或特征来选择不同的策略。下面就这 3 类相关的研究工作做如下简要介绍。

(1) 基于概率模型的多策略 DE

Mallipeddi 等[5]提出了一种集成式的EPSDE算法，分别构建了策略候选池和参数候选池，从中随机选择不同的策略和参数进行组合，再根据组合的历史表现来设置其被选用的概率。Qin 等[10]提出了一种自适应 SaDE 算法，选择了 4 种不同风格的策略用于构建策略候选池，用正太分布函数来生成参数值，通过计算不同策略在先前搜索过程中的成功率来设置策略在后期被选中的概率。Gong 等[11]提出了一种策略自适应机制 SaM，对每个个体设置一个参数用于控制选择何种策略，再通过正太分布函数来自适应调节该参数的值，实现自适应选择策略。Wang 等[12]提出了一种高斯精简的 MGBDE 算法，将提出的高斯变异策略和 DE/best/1 策略分别以 50%的概率分配给个体。类似于 MGBDE，Sun 等[13]也提出了一种改进的高斯变异策略，并结合 DE/rand-worst/1 策略构建了多策略机制，采用“累计得分”的方式来应用这 2 种策略。

(2) 基于多策略共存的多策略 DE

针对约束优化问题，Mezura-Montes 等[14]对种群中所有个体采用改进的 DE/current-to-best/1 策略生成多个试验个体，再按适应度值、可行性、以及约束违反条件这 3 项来选择最恰当的试验个体作为后代个体。Wang 等[6]提出了一种组合式的 CoDE 算法，将 3 种不同类型的变异策略和 3 组参数配置进行随机组合，用于同时生成 3 个试验个体，再从中选择最好的一个作为后代个体。Wang 等[15]提出了一种基于累积种群分布信息的 CPI-DE 框架，采用累积种群分布信息用于建立特征坐标系，对种群中所有个体分别生成基于原坐标系和特征坐标系

的 2 个试验个体，再择优选择一个进入下一代。

(3) 基于多子种群的多策略 DE

Zhou 等[7]提出了一种基于角色分配机制的 RADE 算法，根据个体的适应度和个体间的欧式距离对个体进行角色分配，把种群中所有个体划分为 3 类角色，再结合角色的特点来选择不同的变异策略和参数配置。Wu 等[16]提出了一种基于多子种群的集成式 MPEDE 算法，先把种群随机分为 4 个子种群，其中前 3 个子种群的个体数量较少，但 3 者规模相同，最后一个规模较大的子种群作为奖赏种群，再为 3 个小子种群应用不同的变异策略，经过一定进化代数后将奖赏种群分配给表现最好的小子种群。Zhou 等[17]提出了一种基于相交变异策略的 IMDE 算法，先将种群中所有个体按适应度分为 2 个子种群，再为这 2 个子种群设计了 4 种不同的变异策略，在这些变异策略中，2 个子种群中的个体相互交叉参与变异操作，目的是平衡算法的勘探和开采能力。

从上述相关工作中可看出，虽然实现多策略机制的方式各有不同，但事实上不论采用哪种方式，多策略 DE 的目的都是试图通过组合不同的变异策略和参数配置来弥补经典 DE 仅采用单一策略和固定参数值的不足。

3 融和邻域搜索的多策略 DE 算法

3.1 基于多子种群技术的多策略机制

在解决一些复杂的优化问题时，经典 DE 的求解效果往往难以令人满意。通常，这些复杂的优化问题大多是多峰类型，其适应度地形一般遍布局部极值，求解难度很大。对经典 DE 而言，在算法的求解过程中仅采用单一的变异策略和固定的参数配置，这种方式易导致算法出现收敛速度过慢或陷入局部最优等问题。因此，设计多策略 DE 来求解复杂优化问题是一条有效途径，可发挥不同策略的优势，实现优势互补。

从多策略 DE 的相关研究工作和 DE 自身的特点中，我们注意到对种群中的个体而言，个体的好坏可通过适应度来衡量，并且不同个体在适应度地形上的位置分布也可通过适应度值来粗略推断。一般而言，较好个体可能分布在局部极值或全局极值的周边，而较差个体所处区域则可能离局部极值或全局极值更远。因此，对较好个体宜采用开采能力强的策略，如：DE/best/1，而对较差个体则更适合采用勘探能力强的策略，如：DE/rand/1。不难得出，从个体层次来看，设计多策略机制时应考虑不同个体的优劣，做到有的放矢。

事实上，在多策略 DE 的相关研究工作中，已有不少算法从个体层次来设计多策略机制。例如，在文献[7]中，Zhou 等提出一种基于角色分配机制的 RADE 算法，该算法通过结合个体的适应度和位置信息来对个体进行分组，不同组采用不同的变异策略，实验结果验证了该方式的有效性。然而，RADE 在分组过程中采用算术平均的方式来计算均值，这种方式容易使得超级个体(即适应度相对于其他个体要好很多的个体)对分组结果带来偏差，引起数据噪声的问题。此外，RADE 在分组过程中计算欧式距离可能会使得算法运行时间相对更长，例如在测试函数为 30 维的情况下，RADE 的平均 CPU 时间是 DE/rand/1 的 1.16 倍。在文献[16]中，Wu 等提出了基于多子种群的集成式 MPEDE 算法，该算法将种群随机分为 4 个子种群，设计了一种奖励机制来选择应采用哪种策略变异，实验结果验证了该方式的有效性。然而，MPEDE 在种群划分过程中采用的是随机方式，可能引起的问题是未充分利用优质个体的信息，在一定程度上会导致算法收敛过慢。

受以上相关多策略 DE 的启发，本文也试图从个体层次来设计多策略机制，并结合不同个体的优劣情况来选择策略。具体而言，首先根据个体的适应度值将种群划分为 3 个子种群，分别对应适应度值较好的子种群 A、适应度值一般的子种群 B、以及适应度值较差的子种群 C。显然，这种划分方法

DOI: 10.16182/j.issn1004731x.joss.18-0787

的一个关键问题是如何来确定哪些个体可以认为是较好的，哪些个体又可以认为是较差的。此处，采用简单但却较有效的方式，将种群中个体按适应度值从好到差进行排序，按“二八法则”把前 20%的个体视较好个体，剩下的 80%个体视为一般个体和较差个体。进一步，把剩下的 80%个体中的前 40%视为一般个体，而最后的 40%个体即视为较差个体。“二八法则”是社会学中的一个重要概念，其核心是任何一组东西中，最重要的部分只占 20%，而剩下的 80%却是次要的。通过这种划分方式可将适应度值相似的个体大致分配至同一个子种群，有利于在个体层次选择策略，也有利于发挥优质个体的引领作用。

在种群划分的基础上，如何选择变异策略和设置参数也是设计多策略 DE 的关键问题。从第 1 节的经典 DE 介绍中可看出，不同变异策略因选择的个体不同，使得策略具备不同的搜索能力，例如，DE/best/1 因包含全局最优个体，使得该策略侧重于开采，适合求解单峰类型的问题；而 DE/rand/1 包含的个体均为随机选择，使得该策略的搜索方向具有无偏性，适合求解多峰类型的问题。与此类似的是，参数值也会影响算法的搜索能力，例如，较大的 F 值可增加差分向量的扰动程度，有利于增强算法的勘探能力；而较大的 CR 值可令试验个体继承变异个体的更多信息，同样也可增强算法的勘探能力。虽然，在不少的改进 DE 中，研究人员提出了各种各样的改进变异策略和参数自适应机制，但我们为保持算法的简洁性，对 3 个子种群采用如下所示的变异策略和参数设置：

1) 子种群 A：[DE/best/2，F=0.9，CR=0.1]

2) 子种群 B：[DE/rand/1，F=0.6，CR=0.1]

3) 子种群 C：[DE/rand/1，F=0.7，CR=0.9]

由此看出，对于适应度值较好的子种群 A 采用了开采能力较强的 DE/best/2 策略，并且 CR 值也仅设置为 0.1，目的在于将搜索范围集中在这些优质个体的周边，实现细粒度搜索，但同时为避免搜索过于贪婪，将 F 值设为 0.9。对于子种群 B 和 C 则均采用了搜索方向具备无偏性的 DE/rand/1 策略，目的在于扩大搜索范围，同时通过设置不同的 F 和 CR 值来调整 B 和 C 的搜索行为，使得 C 的搜索行为相对于 B 更偏向勘探。综合来看，子种群 A 侧重于开采，子种群 C 侧重勘探，而子种群 B 则位于 A 和 C 之间，搜索行为更为均衡。通过对这 3 个子种群设置不同的策略和参数，期望能互补不同子种群的搜索能力，力图平衡整个种群的勘探和开采能力。

3.2 融合邻域搜索操作

在上述多子种群划分方法中，我们将种群中适应度值最好的 20%个体划分为子种群 A，并且为子种群 A 选择了开采能力较强的 DE/best/2 策略。对于多峰类型的优化问题，其适应度地形往往“崎岖不平，沟壑纵横”，子种群 A 中的个体很可能分布在局部极值或全局极值附近，如果能对这些个体的邻域空间开展细粒度搜索，显然可提高找到全局最优解的概率。为此，为进一步发挥优质个体的引领作用，我们对子种群 A 融合了一种邻域搜索操作。

邻域搜索操作的目的是对个体的邻域空间进行搜索以期找到更好个体。在不同类型的进化算法中，研究人员设计了多种不同类型的邻域搜索操作。例如，Wang 等[18]针对 PSO 算法存在收敛速度过慢的问题提出了一种邻域搜索操作 LNS，采用基于个体下标索引的环形邻域拓扑结构，在每个个体的 K 邻域内进行搜索。Peng 等[19]提出了一种基于随机邻域的改进 DE 变异策略 DE/neighbor/1，该策略在算法的每次迭代过程中构建一个规模为 N 的随机邻域，从该邻域中选择最优个体作为变异策略的目标个体。这些邻域搜索操作在形式上虽然各有特点，但却都有效地提高了算法的性能。

针对子种群 A 的特点，直接采用 Wang 等提出的邻域搜索操作 LNS，该策略结构简单、搜索效率高，其采用的搜索公式如式(8)所示[18]：

$$TX_i = r_1 \cdot X_i + r_2 \cdot pbest_i + r_3 \cdot (X_a - X_b) \quad (8)$$

式中：TX_i 为经搜索生成的新个体的位置；X_i 为目标个体的位置；$pbest_i$ 为目标个体的历史最优位置；X_a 和 X_b 为从目标个体的 K 邻域范围内随机选择的互不相同的 2 个个体的位置，即下标 a 和 b 均在 $[i-K, i+K]$ 范围内取值，3 个系数 r_1、r_2、r_3 是 [0, 1] 范围内的随机数，并且满足条件 $r_1 + r_2 + r_3 = 1$。

与搜索公式略有不同的是，我们结合 DE 算法的特点，将目标个体的历史最优位置 $pbest_i$ 更改为目标个体的 K 邻域范围内的最优个体的位置 $lbest_i$，经微调后的搜索公式如式(9)所示：

$$TX_i = r_1 \cdot X_i + r_2 \cdot lbest_i + r_3 \cdot (X_a - X_b) \quad (9)$$

式中：$lbest_i$ 为目标个体 X_i 在半径为 K 的邻域范围内的最优个体。

图 1 给出了当邻域半径 K 为 3 的示意图。不难看出，邻域半径 K 是影响邻域搜索操作性能的一个重要参数，如果 K 值越大意味着目标个体的邻域空间范围越接近整个种群，反之 K 值越小则表明邻域空间范围越小。采用邻域搜索操作的目的在于对优质个体的邻域空间进行细粒度搜索，因此 K 值不宜过大，采用经验值将 K 设置为种群规模的 10%。

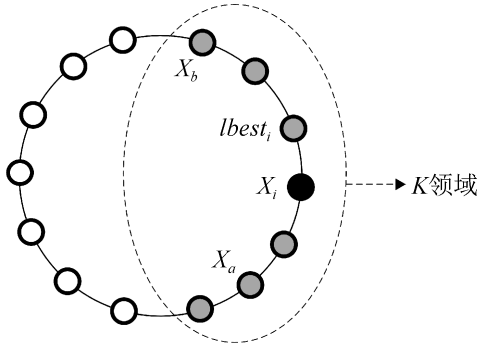

图 1 邻域搜索操作示意图

Fig. 1 Demonstration of neighborhood search operator

通过采用邻域搜索操作对子种群 A 中个体的邻域空间进行搜索，可有效提高找到全局最优解的概率，进一步发挥优质个体的引领作用。至此，对子种群 A 而言，采用了 DE/best/2 和邻域搜索 2 种策略。为协调这 2 种搜索策略的使用，引入参数 P 来控制策略的使用频率。经初步实验，对参数 P 采用固定取值的方式，将其值设置为经验值 0.7。算法 1 给出了对子种群 A 进行操作的伪代码。

算法 1: 对子种群 A 的操作

1. if rand(0，1)≤P then
2. Execute the strategy DE/best/2
3. else
4. Execute the strategy LNS
5. end if

3.2 算法伪代码

针对经典 DE 仅采用一种变异策略和固定参数值存在的不足，本文提出了 MSDE-NS 算法，相对于经典 DE 主要有 2 点改进：

(1) 根据适应度值，将种群划分为 3 个子种群，再结合每个子种群的特点来选择不同的变异策略和参数值；

(2) 为进一步发挥优质个体的引领作用，对具有较好适应度值的子种群采用了邻域搜索操作。

算法 2 给出了伪代码描述，其中 NP 为种群大小，FES 表示适应度函数的评估次数，$MaxFES$ 为适应度函数的最大评估次数。

算法 2：MSDE-NS 算法伪代码

1. Generate an initial population *pop* with *NP* individuals；
2. Evaluate the fitness value of each individual X_i, and set *FES*=*NP*；
3. while *FES* ≤ *MaxFES* do；
4. Divide *pop* into three subpopulations according to the fitness value of each individual；
5. for *i*=1 to *NP* do；
6. if X_i belongs to the subpopulation A do
7. Generate a trial individual for X_i through the Algorithm 1；

DOI: 10.16182/j.issn1004731x.joss.18-0787

8. else；

9. Generate a trial individual for X_i by using the selected mutation strategy and parameter values；

10. end；

11. Select the better one from the trial individual and X_i to enter the next generation；

12. end for；

13. $FEs=FEs+NP$；

14. end while。

4 数值实验与结果分析

为验证本文算法性能，采用 34 个典型的测试函数进行实验，这些函数的相关细节如表 1 所示。

表 1 实验采用的 34 个测试函数

Tab. 1 34 test functions in experiments

序号	函数名	搜索范围	最优值
F01	Sphere	[−100, 100]	0
F02	Schwefel 2.22	[−10, 10]	0
F03	Schwefel 1.2	[−100, 100]	0
F04	Schwefel 2.21	[−100, 100]	0
F05	Rosenbrock	[−30, 30]	0
F06	Step	[−100, 100]	0
F07	Quartic with noise	[−1.28, 1.28]	0
F08	Elliptic	[−100, 100]	0
F09	SumSquare	[−10, 10]	0
F10	SumPower	[−1, 1]	0
F11	Exponential	[−10, 10]	0
F12	Schwefel 2.26	[−500, 500]	−418.98·D
F13	Rastrigin	[−5.12, 5.12]	0
F14	Ackley	[−32, 32]	0
F15	Griewank	[−600, 600]	0
F16	Generalized penalized 1	[−50, 50]	0
F17	Generalized penalized 2	[−50, 50]	0
F18	NCRastrigin	[−5.12, 5.12]	0
F19	Alpine	[−10, 10]	0
F20	Levy	[−10, 10]	0
F21	Bohachevsky_2	[−100, 100]	0
F22	Weierstrass	[−0.5, 0.5]	0
F23	Himmelblau	[−5, 5]	−78.332 36
F24	Michalewicz	[0, π]	−D
F25	Shifted Sphere Function	[−100, 100]	−450
F26	Shifted Schwefels Problem 1.2	[−100, 100]	−450
F27	Shifted Rotated High Conditioned Elliptic Function	[−100, 100]	−450
F28	Shifted Schwefels Problem 1.2 with Noise in Fitness	[−100, 100]	−450
F29	Schwefels Problem 2.6 with Global Optimum on Bounds	[−100, 100]	−310
F30	Shifted Rosenbrocks Function	[−100, 100]	390
F31	Shifted Rotated Griewanks Function without Bounds	[0, 600]	−180
F32	Shifted Rotated Ackleys Function with Global Optimum on Bounds	[−32, 32]	−140
F33	Shifted Rastrigins Function	[−5, 5]	−330
F34	Shifted Rotated Rastrigins Function	[−5, 5]	−330

在表 1 中，F01~F11 是单峰类型的函数，F12~F24 是多峰类型的函数，F25~F34 是 CEC2005 优化竞赛测试函数集中的前 10 个函数[14]，均为带偏移或旋转类型的函数，其中 F05 在维度大于 3 时会转化为多峰类型的函数[15]，F06 是非连续类型的函数，F07 是带噪声类型的函数。在后续实验中，设置 3 组实验来评估本文算法性能，分别是：

(1) 与 4 种经典 DE 算法比较；

(2) 与 3 种相关的改进 DE 算法比较；

(3) 与 5 种改进的其他进化算法比较。

实验中，进行 2 个维度的实验，分别是 30 维和 50 维，适应度函数的最大评估次数分别设置为：

(1) D=30 时，F01~F24 为 200 000 次，F25~F34 为 300 000 次；

(2) D=50 时，F01~F24 为 250 000 次，F25~F34 为 500 000 次。所有算法在每个函数上独立运行 30 次，记录 30 次结果的均值，算法每次运行的结果由 $f(X)-f(X^*)$ 得到，$f(\cdot)$ 为适应度函数，X 为算法得到的最优解，X^*为测试函数的全局最优解。此外，为在统计意义上比较两种算法的性能是否存在显著性差异，对实验结果进行了非参数 Wilcoxon 检验[20-21]，显著性水平设为 0.05，符号“+”、“−”和“≈”分别表示本文算法要优于、劣于和相当于对比算法。

4.1 与 4 种经典 DE 算法比较

本节实验的目的在于评估提出的多策略机制的有效性。在 MSDE-NS 中，为 3 个子种群分别设置了 3 组策略和参数配置，其中包含了 2 种常用的变异策略 DE/best/2 和 DE/rand/1，为此把这 3 组策略和参数配置单独用于构建对应的 DE 算法，将 MSDE-NS 与这 3 种 DE 算法进行对比。此外，还增加了一种文献中广泛使用的经典 DE 进行对比，即应用 DE/rand/1 策略，F 和 CR 分别设为 0.5 和 0.9。因此，本节中参与实验对比的 DE 算法有 4 种，如下所示：

1) DE1：[DE/best/2，F=0.9，CR=0.1]

2) DE2：[DE/rand/1，F=0.6，CR=0.1]

3) DE3：[DE/rand/1，F=0.7，CR=0.9]

4) DE4：[DE/rand/1，F=0.5，CR=0.9]

前 3 种 DE1-DE3 为 MSDE-NS 中采用的 3 组策略和参数配置，DE4 为文献中广泛使用的经典 DE。为公平对比，这 4 种 DE 算法的种群规模和 MSDE-NS 保持相同，均设置为 30。

表 2~3 分别给出了上述 5 种算法在测试函数为 30 维和 50 维的结果。

从表 2 最后一行的 Wilcoxon 检验结果可得出，当函数为 30 维时，MSDE-NS 在大部分函数上的结果要优于 4 种经典 DE 算法。具体而言，与 DE1 相比，MSDE-NS 在 25 个函数上取得了更好结果，在 7 个函数上性能相当，仅在 2 个函数上结果更差。DE1 在函数 F33 上取得了 5 种算法中的最好结果，可能的原因是 F33(带偏移的 Rastrigin 函数)的局部极值数量众多，使得 5 种算法中开采能力最强的 DE/best/2 策略更适合这种类型的适应度地形。与 DE2 相比，MSDE-NS 也仅在 2 个函数上处于劣势，而在其他 32 个函数上取得了更好或相当的结果。与 DE3 相比，MSDE-NS 在 24 个函数上更优，在 5 个函数上性能相当，同样也在 5 个函数上结果更差。值得注意的是，DE3 在单峰类型的偏移函数上性能比 MSDE-NS 略好，可能的原因是较大的 F 和 CR 值使得 DE3 的勘探能力是五种算法中最强的，有利于提高算法的多样性，但从总体性能来看本文提出的 MSDE-NS 还是更优。与 DE4 相比，SDE-NS 在 34 个函数上的性能均要好于或相当于 DE4。当函数维度从 30 增加至 50 维时，从表 3 的结果中可看出，MSDE-NS 依然在大部分函数上的性能要优于参与对比的 4 种经典 DE 算法，这也在一定程度上表明虽然问题的求解难度有所增但 MSDE-NS 的性能却能依旧保持良好。

DOI: 10.16182/j.issn1004731x.joss.18-0787

表 2 测试函数为 30 维时，MSDE-NS 和 4 种经典 DE 的实验结果

Tab. 2 Experimental results of MSDE-NS and four classic DE algorithms when D=30

函数	DE1	DE2	DE3	DE4	MSDE-NS
F01	1.09E−26+	6.11E−84+	4.21E−45+	1.13E+01+	1.66E−182
F02	6.19E−17+	8.51E−49+	8.87E−24+	8.13E−19+	1.44E−92
F03	5.13E+03+	1.37E+03+	2.53E−08+	5.13E−01+	7.31E−112
F04	3.85E−01+	3.26E−06+	9.80E+00+	2.13E+01+	3.27E−69
F05	3.06E+01+	2.82E+01+	4.68E−01+	1.79E+03+	1.88E−07
F06	0.00E+00≈	0.00E+00≈	1.33E−01+	4.69E+01+	0.00E+00
F07	2.14E−02+	5.35E−03+	7.81E−03+	1.26E+05+	5.99E−04
F08	2.76E−26+	5.72E−83+	1.22E−44+	3.80E+01+	1.80E−180
F09	1.82E−31+	1.57E−89+	1.99E−49+	1.16E−02+	9.22E−188
F10	1.55E−58+	8.58E−172+	2.97E−41+	1.06E+06+	1.30E−296
F11	4.65E−06+	5.92E−06+	1.96E−07+	0.00E+00≈	0.00E+00
F12	3.82E−04≈	3.82E−04≈	8.06E+02+	1.47E+03+	3.82E−04
F13	0.00E+00≈	6.63E−02≈	2.71E+01+	2.03E+01+	0.00E+00
F14	1.72E−13+	7.55E−15+	7.19E−15+	2.33E+00+	3.05E−15
F15	0.00E+00≈	0.00E+00≈	1.56E−03+	2.02E−01+	0.00E+00
F16	9.16E−28−	1.57E−32≈	1.73E−02≈	1.27E+03+	3.46E−03
F17	3.51E−27+	1.35E−32≈	7.32E−04≈	1.02E+04+	1.35E−32
F18	3.33E−01+	1.67E+00+	8.07E+00+	0.00E+00≈	0.00E+00
F19	1.39E−06+	1.47E−35−	1.13E−16−	7.04E−13+	8.34E−14
F20	1.21E−25+	1.35E−31≈	3.66E−03≈	6.05E−01+	1.35E−31
F21	0.00E+00≈	0.00E+00≈	3.98E−02≈	3.17E+00+	0.00E+00
F22	1.08E−03+	0.00E+00≈	1.35E+00+	6.56E−01+	0.00E+00
F23	−7.83E+01≈	−7.83E+01≈	−7.79E+01+	−7.59E+01+	−7.83E+01
F24	−2.85E+01+	−2.86E+01≈	−2.62E+01+	−2.66E+01+	−2.86E+01
F25	5.68E−14+	5.49E−14+	2.84E−14+	3.79E+01+	5.68E−15
F26	2.87E+03+	3.76E+02+	2.94E−12−	3.61E+00+	1.31E−08
F27	2.49E+07+	2.30E+07+	1.47E+05−	2.83E+05≈	2.77E+05
F28	1.04E+04+	5.66E+03+	3.91E−03−	1.52E+01+	1.11E−02
F29	5.20E+03+	4.19E+03+	2.70E+02−	2.66E+03+	5.02E+02
F30	4.25E+01+	5.38E+01+	5.32E−01+	3.13E+07+	5.12E−14
F31	2.73E−01+	6.09E−02+	7.63E−03+	6.23E−01+	4.92E−04
F32	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01
F33	5.68E−14−	3.32E−02−	2.60E+01+	2.93E+01+	9.95E−02
F34	1.56E+02+	1.29E+02+	1.26E+02+	4.54E+01≈	4.76E+01
+/≈/−	25/7/2	20/12/2	24/5/5	29/5/0	−−

表 3 测试函数为 50 维时，MSDE-NS 和 4 种经典 DE 的实验结果

Tab. 3 Experimental results of MSDE-NS and four classic DE algorithms when D=50

函数	DE1	DE2	DE3	DE4	MSDE-NS
F01	6.68E−11+	1.67E−61+	4.99E−31+	6.21E+01+	2.94E−204
F02	5.47E−08+	8.53E−37+	1.05E−17+	1.91E−03+	6.57E−105
F03	2.96E+04+	2.28E+04+	1.29E+00+	4.17E−01+	2.85E−106
F04	1.12E+01+	9.82E−03+	2.24E+01+	2.98E+01+	3.41E−81

续表

函数	DE1	DE2	DE3	DE4	MSDE-NS
F05	1.12E+02+	4.63E+01+	5.78E+01+	7.03E+04+	1.72E+01
F06	0.00E+00≈	0.00E+00≈	9.10E+00+	2.92E+02+	0.00E+00
F07	1.12E−01+	1.26E−02+	3.10E−02+	2.09E+06+	5.95E−04
F08	8.90E−11+	1.67E−60+	1.23E−29+	2.93E+02+	7.59E−205
F09	1.94E−14+	1.25E−65+	3.43E−34+	1.96E−01+	6.40E−207
F10	1.39E−28+	1.84E−175+	3.04E+03+	3.25E+13+	0.00E+00
F11	4.72E−06+	6.58E−06+	1.05E−05+	1.63E−06+	7.40E−18
F12	6.36E−04+	7.90E+00≈	2.81E+03+	4.32E+03+	6.36E−04
F13	2.33E+01+	6.63E−02≈	5.23E+01+	4.30E+01+	1.16E+00
F14	8.57E−06+	1.51E−14+	1.79E−01+	4.77E+00+	3.40E−15
F15	4.67E−10+	0.00E+00≈	2.87E−03+	8.84E−01+	0.00E+00
F16	3.53E−11+	9.42E−33≈	8.91E−02+	1.04E+04+	9.42E−33
F17	5.86E−11+	1.35E−32≈	1.10E−03+	1.26E+05+	1.35E−32
F18	3.67E−01+	3.43E+00+	1.55E+01+	0.00E+00≈	0.00E+00
F19	1.21E+00+	8.53E−13+	1.02E−15+	3.80E−04+	4.89E−104
F20	3.62E−08+	1.35E−31≈	7.32E−03+	2.81E+00+	1.35E−31
F21	2.67E−08+	0.00E+00≈	8.09E−01+	7.78E+01+	0.00E+00
F22	1.51E+01+	0.00E+00−	5.04E+00+	5.16E+00+	2.27E−01
F23	−7.83E+01≈	−7.83E+01≈	−7.59E+01+	−7.36E+01+	−7.83E+01
F24	−4.03E+01+	−4.62E+01−	−4.24E+01+	−4.33E+01+	−4.51E+01
F25	1.14E−13+	6.06E−14+	9.28E−14+	2.51E+02+	4.93E−14
F26	2.58E+04+	1.23E+04+	7.08E−04−	1.22E−02−	4.06E−02
F27	1.14E+08+	9.27E+07+	4.17E+05−	5.58E+05≈	5.82E+05
F28	5.08E+04+	3.69E+04+	2.57E+02≈	1.92E+03+	3.41E+02
F29	1.27E+04+	1.06E+04+	2.80E+03−	6.63E+03+	3.05E+03
F30	6.57E+01+	5.81E+01+	3.98E+01+	1.44E+08+	7.82E−04
F31	4.62E−01+	2.20E−02+	5.67E−03+	2.35E+00+	8.21E−04
F32	2.11E+01≈	2.11E+01≈	2.11E+01≈	2.11E+01≈	2.11E+01
F33	1.97E−13−	3.32E−01≈	6.41E+01+	8.19E+01+	3.98E−01
F34	4.07E+02+	3.40E+02+	3.27E+02+	1.02E+02+	9.84E+01
+/≈/−	30/3/1	21/11/2	29/2/3	30/3/1	−−

4.2 与 3 种相关的改进 DE 算法比较

本节对比实验的目的在于检验本文算法的整体性能与其他多策略 DE 相比如何。为此，我们将 MSDE-NS 与 3 种相关的改进 DE 算法进行比较，这 3 种算法分别是：

1) RADE[7]：基于角色分配机制的 DE 算法；

2) MPEDE[16]：基于多子种群机制的集成式 DE 算法；

3) RNDE[19]：基于随机邻域策略的 DE 算法。

可以看出，在上述 3 种算法中，RADE 和 MPEDE 均采用了多子种群方式来实现多策略机制，而 RNDE 采用了邻域搜索操作来增强 DE 性能，因此与这 3 种算法进行对比具备代表性，能够检验 MSDE-NS 的多策略机制和邻域搜索操作是否有效。

为公平对比，这 3 种改进 DE 的所有参数设置均保持与原文献相同，

表 4 给出了它们的参数设置明细。

表 5~6 分别给出了这 4 种 DE 算法在测试函数为 30 维和 50 维的结果。

DOI: 10.16182/j.issn1004731x.joss.18-0787

表 4 3 种相关的改进 DE 算法的参数设置

Tab. 4 Parameter settings of three related improved DE algorithms

算法	参数设置
RADE	NP=30, F=[0.5, 0.7], CR=[0.1, 0.9]
MPEDE	NP=250, $\lambda_1=\lambda_2=\lambda_3$=0.2, ng=20
RNDE	NP=100, F=0.5, CR=0.9, N_{lb}=3, N_{ub}=10

表 5 测试函数为 30 维时，MSDE-NS 和 3 种相关的改进 DE 算法的实验结果

Tab. 5 Experimental results of MSDE-NS and three related DE algorithms when D=30

函数	RADE	MPEDE	RNDE	MSDE-NS
F01	3.73E−87+	6.47E−29+	2.91E−42+	1.66E−182
F02	1.18E−49+	2.45E−14+	1.59E−22+	1.44E−92
F03	3.22E−07+	2.11E−16+	5.58E−05+	7.31E−112
F04	8.26E−09+	1.16E−10+	5.75E−07+	3.27E−69
F05	2.97E−14−	1.96E+00+	2.48E−02+	1.88E−07
F06	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00
F07	3.88E−03+	2.29E−03+	3.83E−03+	5.99E−04
F08	1.16E−95+	1.50E−24+	3.20E−41+	1.80E−180
F09	1.81E−108+	3.53E−69+	1.43E−46+	9.22E−188
F10	3.22E−200+	7.18E−66+	3.54E−81+	1.30E−296
F11	3.79E−06+	4.35E−24+	7.78E−14+	0.00E+00
F12	2.76E+01+	3.82E−04≈	3.82E−04≈	3.82E−04
F13	1.33E−01+	4.26E−05+	2.71E−08+	0.00E+00
F14	6.37E−15+	5.03E−15+	4.12E−15+	3.05E−15
F15	3.29E−04≈	0.00E+00≈	0.00E+00≈	0.00E+00
F16	1.57E−32≈	9.35E−31−	1.57E−32≈	3.46E−03
F17	1.35E−32≈	2.78E−29+	1.35E−32≈	1.35E−32
F18	0.00E+00≈	2.07E+00+	0.00E+00≈	0.00E+00
F19	8.61E−58−	1.06E−04+	7.42E−04+	8.34E−14
F20	1.35E−31≈	6.93E−29+	1.35E−31≈	1.35E−31
F21	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00
F22	0.00E+00≈	3.82E−01+	6.11E−04+	0.00E+00
F23	−7.83E+01≈	−7.83E+01≈	−7.83E+01≈	−7.83E+01
F24	−2.86E+01+	−2.87E+01−	−2.73E+01+	−2.86E+01
F25	1.14E−14≈	0.00E+00≈	0.00E+00≈	5.68E−15
F26	3.13E−10−	2.56E−27−	4.55E−08+	1.31E−08
F27	3.06E+05≈	2.37E+00−	2.44E+05≈	2.77E+05
F28	2.12E−03−	3.48E−17−	1.53E−03−	1.11E−02
F29	3.08E+02−	8.01E−06−	7.93E+01−	5.02E+02
F30	5.12E−14≈	4.67E+00+	7.39E−11+	5.12E−14
F31	2.87E−03≈	5.25E−03≈	9.86E−04≈	4.92E−04
F32	2.09E+01≈	2.09E+01≈	2.09E+01≈	2.09E+01
F33	1.66E−01≈	0.00E+00−	0.00E+00−	9.95E−02
F34	8.72E+01+	2.41E+01−	1.07E+02+	4.76E+01
+/≈/−	14/15/5	18/8/8	18/13/3	

表 6 测试函数为 50 维时，MSDE-NS 和 3 种相关的改进 DE 算法的实验结果

Tab. 6 Experimental results of MSDE-NS and three related DE algorithms when D=50

函数	RADE	MPEDE	RNDE	MSDE-NS
F01	5.93E−61+	3.22E−29+	2.45E−33+	2.94E−204
F02	1.54E−35+	8.93E−15+	1.41E−17+	6.57E−105
F03	4.01E+01+	4.16E−06+	4.88E+00+	2.85E−106
F04	8.88E−04+	1.96E−06+	1.73E−01+	3.41E−81
F05	8.04E+00−	2.41E+01+	2.36E+01+	1.72E+01
F06	0.00E+00≈	0.00E+00≈	0.00E+00≈	0.00E+00
F07	7.78E−03+	4.99E−03+	7.35E−03+	5.95E−04
F08	7.01E−70+	3.30E−23+	1.52E−32+	7.59E−205
F09	4.01E−80+	3.63E−64+	1.51E−36+	6.40E−207
F10	1.29E−169+	2.67E−109+	6.48E−72+	0.00E+00
F11	6.04E−06+	1.26E−23−	1.69E−07+	7.40E−18
F12	9.48E+01+	6.53E−03+	6.36E−04≈	6.36E−04
F13	9.95E−01−	5.43E−01−	5.45E+01+	1.16E+00
F14	8.02E−15+	8.59E−15+	7.19E−15+	3.40E−15
F15	0.00E+00≈	1.72E−03+	0.00E+00≈	0.00E+00
F16	9.42E−33≈	6.22E−03+	9.42E−33≈	9.42E−33
F17	1.35E−32≈	3.66E−04+	1.35E−32≈	1.35E−32
F18	3.33E−02≈	1.41E+01+	0.00E+00≈	0.00E+00
F19	1.87E−34+	7.85E−10+	8.15E−03+	4.89E−104
F20	1.35E−31≈	3.66E−03+	1.35E−31≈	1.35E−31
F21	0.00E+00≈	1.99E−01+	0.00E+00≈	0.00E+00
F22	0.00E+00−	1.24E+00+	3.63E+00+	2.27E−01
F23	−7.83E+01≈	−7.67E+01+	−7.83E+01≈	−7.83E+01
F24	−4.79E+01−	−4.61E+01−	−3.90E+01+	−4.51E+01
F25	5.68E−14+	0.00E+00−	1.33E−14−	4.93E−14
F26	6.07E−02+	6.74E−13−	6.84E−03−	4.06E−02
F27	1.10E+06≈	6.12E+04−	5.25E+05≈	5.82E+05
F28	5.92E+02+	8.90E−01−	1.53E+02−	3.41E+02
F29	2.58E+03−	7.44E+02−	2.30E+03−	3.05E+03
F30	1.33E−01+	1.24E+00+	1.62E−03+	7.82E−04
F31	7.39E−04≈	5.82E−03≈	2.79E−03≈	8.21E−04
F32	2.11E+01≈	2.11E+01≈	2.11E+01≈	2.11E+01
F33	1.19E+00≈	0.00E+00−	1.03E−12≈	3.98E−01
F34	2.48E+02+	4.71E+01−	2.39E+02+	9.84E+01
+/≈/−	17/12/4	21/3/10	17/13/4	

从表 5 的结果可看出，MSDE-NS 在大部分函数上相对于这 3 种相关的改进 DE 算法更有优势。与 RADE 相比，MSDE-NS 在 29 个函数上具备更好或相当的性能，在 5 个函数上 RADE 更优。与 MPEDE 相比，MSDE-NS 在 18 个函数上取得了更好结果，而 MPEDE 仅在 8 个函数上更优。值得注

意的，MPEDE 的性能也非常不错，在 6 个偏移或旋转类型的函数上取得了 4 种算法中的最好结果。与 RNDE 相比，MSDE-NS 仅在 3 个函数上性能更差，但在 31 个函数上取得了更好或相当的结果。表 6 的结果与表 5 类似，MSDE-NS 在整体性能上具备更多优势。

表 7 给出了本节中 4 种算法的 Friedman 排名情况，从中可看出在函数的 2 种维度情况下 MSDE-NS 均取得了最好排名。

表 7 MSDE-NS 和 3 种相关的改进 DE 算法的 Friedman 排名结果

Tab. 7 Friedman ranking results of MSDE-NS and three related DE algorithms

算法	*D*=30	*D*=50
RADE	2.68	2.65
MPEDE	2.65	2.74
RNDE	2.69	2.68
MSDE-NS	**1.99**	**1.94**

4.3 与 5 种改进的其他进化算法比较

为进一步评估 MSDE-NS 的性能，本节对比实验中将 MSDE-NS 与五种改进的其他进化算法进行对比，包括 2 种知名的改进 PSO 算法和 3 种知名的改进 ABC 算法，它们分别是：

1) OLPSO-G[22]：正交学习的 PSO 算法；

2) ALPSO[23]：基于自适应学习策略的混合 PSO 算法；

3) ILABC[24]：基于信息学习机制的 ABC 算法；

4) MPGABC[25]：基于新颖概率模型的改进 gbest 引导的 ABC 算法；

5) ABCLGII[26]：基于局部和全局信息交互的 ABC 算法。

在上述算法中，OLPSO-G 采用正交学习策略用于更新粒子的飞行速度，有利于避免传统速度更新公式可能引起的“震荡”现象。ALPSO 设计了一种基于自学习的候选解生成策略来提高勘探能力，同时也设计了一种基于竞争学习的预测策略用于确保算法的开采能力。ILABC 采用聚类技术将种群划分为多个子种群，每个子种群采用两种解搜索方程用于搜索新的候选解。为提高算法的搜索效率，MPGABC 设计了一种简洁的多策略机制，并且在观察蜂阶段采用了一种新颖的概率模型用于提高优秀蜜源被选中的概率。ABCLGII 在雇佣蜂阶段采用了邻域搜索技术来提高搜索效率，在观察蜂阶段则设计了 2 种新的解搜索方程用来生成候选解，并且对这 2 种解搜索方程采用了适应性选择方式来控制使用频率。

为公平对比，上述 5 种算法的参数设置与其相应的原文献保持相同，具体明细如表 8 所示。

表 8 5 种改进的其他进化算法的参数设置

Tab. 8 Parameter settings of five other improved evolutionary algorithms

算法	参数设置
OLPSO-G	*NP*=40, ω=[04, 0.9], c=2.0, *G*=5, V_{MAXd}=0.2×Range
ALPSO	*NP*=20, ω=[0.4, 0.9], c_1=c_2=2
ILABC	*NP*=100, *limit*=200
MPGABC	*NP=50, limit=NP×D, P=0.3*
ABCLGII	*NP*=50, *limit= NP×D*, *r*=1, *q*=0.2

这 5 种算法和本文算法的实验结果如表 9 所示，最优结果用粗体凸显，其中测试函数的维度为 30 维，适应度函数的最大评估次数为 200 000。需要说明的是，OLPSO-G 和 ILABC 的实验结果来源于文献[24]，ALPSO 的结果来源于其原文献[23]。MPGABC 和 ABCLGII 的结果来源于我们运行算法的源代码。不难看出，与这 5 种改进的其他进化算法相比，MSDE-NS 在大部分测试函数上更优。

同时，表 10 给出了本节实验中 6 种算法的 Friedman 排名情况，MSDE-NS 排名第一，紧随其后的是 ILABC、MPGABC、ABCLGII、ALPSO、以及 OLPSO-G 算法。

DOI: 10.16182/j.issn1004731x.joss.18-0787

表 9 MSDE-NS 和 5 种改进的其他进化算法的实验结果
Tab. 9 Experimental results of MSDE-NS and five other improved evolutionary algorithms

函数	OLPSO-G	ALPSO	ILABC	MPGABC	ABCIGII	MSDE-NS
Sphere	4.12E-54	1.70E-93	2.19E-91	1.02E-98	2.06E-119	**1.66E-182**
Schwefel2.22	9.85E-30	2.11E-28	1.17E-48	3.20E-53	1.69E-61	**1.44E-92**
Rosenbrock	2.15E+01	1.78E+01	8.92E-02	2.70E+00	6.85E+00	**1.88E-07**
Step	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Schwefel2.26	3.84E+02	1.41E+03	**3.82E-04**	**3.82E-04**	7.90E+00	**3.82E-04**
Rastrigin	1.07E+00	5.12E-14	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
NCRastrigin	2.18E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
Ackley	7.98E-15	1.04E-14	5.45E-15	3.24E-14	3.08E-14	**3.05E-15**
Griewank	4.83E-03	**0.00E+00**	**0.00E+00**	3.29E-04	7.40E-04	**0.00E+00**
Generalized penalized 1	**1.59E-32**	**1.57E-32**	**1.57E-32**	**1.57E-32**	**1.57E-32**	3.46E-03
Generalized penalized 2	4.39E-04	**1.35E-32**	**1.35E-32**	**1.35E-32**	**1.35E-32**	**1.35E-32**

表 10 MSDE-NS 和 5 种改进的其他进化算法的 Friedman 排名结果
Tab. 10 Friedman ranking results of MSDE-NS and five other improved evolutionary algorithms

算法	平均排名	算法	平均排名
OLPSO-G	5.23	MPGABC	3.23
ALPSO	4.00	ABCLGII	3.23
ILABC	2.91	MSDE-NS	**2.41**

5 结论

为克服经典DE仅采用一种变异策略和固定参数设置的不足，本文基于多子种群的方式提出了一种多策略 DE 算法，将种群按适应度划分为 3 个子种群，根据每个子种群的特点分别选择了 3 种对应的变异策略和参数值，使得不同子种群的搜索能力可互补，力图平衡整个种群的勘探和开采能力。同时，为进一步发挥有较好适应度值的子种群的引领作用，我们对该子种群采用了邻域搜索操作，在优质个体的邻域空间范围内进行细粒度搜索，提高找到全局最优解的概率。在 34 个测试函数上进行实验验证，与 4 种经典 DE 算法、3 种相关的改进 DE 算法、以及 5 种改进的其他进化算法进行了对比，对比结果表明本文算法在大部分测试函数上更优。下一步研究工作的重点是将本文算法应用于求解实际优化问题，比如：无线传感器网络的覆盖控制问题。

参考文献:

[1] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of global optimization (S0925-5001), 1997, 11(4): 341-359.

[2] Das S, Suganthan P N. Differential Evolution: A Survey of the State-of-the-Art[J]. IEEE Transactions on Evolutionary Computation (S1089-778X), 2010, 15(1): 4-31.

[3] Gong W, Yan X, Liu X, et al. Parameter extraction of different fuel cell models with transferred adaptive differential evolution[J]. Energy (S0360-5442), 2015, 86: 139-151.

[4] 朱李楠, 王万良, 沈国江. 基于改进差分进化算法的云制造资源优化组合方法[J]. 计算机集成制造系统, 2017, 23(1): 203-214.
Zhu Linan, Wang Wanliang, Shen Guojiang. Resource optimization combination method based on improved differential evolution algorithm for cloud manufacturing[J]. Computer Integrated Manufacturing Systems, 2017, 23(1): 203-214.

[5] Mallipeddi R, Suganthan P N, Pan Q K, et al. Differential evolution algorithm with ensemble of parameters and mutation strategies[J]. Applied Soft Computing (S1568-4946), 2011, 11(2): 1679-1696.

[6] Wang Y, Cai Z, Zhang Q. Differential evolution with composite trial vector generation strategies and control parameters[J]. IEEE Transactions on Evolutionary

Computation (S1089-778X), 2011, 15(1): 55-66.

[7] Zhou X, Wu Z, Wang H, et al. Enhancing differential evolution with role assignment scheme[J]. Soft Computing (S1432-7643), 2014, 18(11): 2209-2225.

[8] Das S, Mullick S S, Suganthan P N. Recent advances in differential evolution–An updated survey[J]. Swarm and Evolutionary Computation (S2210-6502), 2016, 27: 1-30.

[9] Zhou X, Zhang G. Differential Evolution With Underestimation-Based Multimutation Strategy[J]. IEEE Transactions on Cybernetics (S2168-2267), 2019, 49(4): 1-12.

[10] 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 (S1089-778X), 2009: 398-417.

[11] Gong W, Cai Z, Ling C X, et al. Enhanced differential evolution with adaptive strategies for numerical optimization[J]. IEEE Transactions on Cybernetics (S2168-2267), 2011, 41(2): 397-413.

[12] Wang H, Rahnamayan S, Sun H, et al. Gaussian Bare-Bones Differential Evolution[J]. IEEE Transactions on Cybernetics (S2168-2267), 2013, 43(2): 634-647.

[13] Sun G, Lan Y, Zhao R. Differential evolution with Gaussian mutation and dynamic parameter adjustment[J]. Soft Computing (S1432-7643), 2019, 23(5): 1615-1642.

[14] Mezura-Montes E, Velazquez-Reyes J, Coello C A C. Modified Differential Evolution for Constrained Optimization[C]//Proceedings of the IEEE International Conference on Evolutionary Computation. Canada: IEEE, 2006: 25-32.

[15] Wang Y, Liu Z, Li J, et al. Utilizing cumulative population distribution information in differential evolution[J]. Applied Soft Computing (S1568-4946), 2016, 48: 329-346.

[16] Wu G, Mallipeddi R, Suganthan P N, et al. Differential evolution with multi-population based ensemble of mutation strategies[J]. Information Sciences (S0020-0255), 2016.

[17] Zhou Y, Li X, Gao L. A differential evolution algorithm with intersect mutation operator[J]. Applied Soft Computing (S1568-4946), 2013, 13(1): 390-401.

[18] Wang H, Sun H, Li C, et al. Diversity enhanced particle swarm optimization with neighborhood search[J]. Information Sciences (S0020-0255), 2013, 223(2): 119-135.

[19] Peng H, Guo Z, Deng C, et al. Enhancing differential evolution with random neighbors based strategy[J]. Journal of Computational Science (S1877-7503), 2018, 26: 501-511.

[20] Garc I A S, Fern A Ndez A, Luengo J, et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power[J]. Information Sciences (S0020-0255), 2010, 180(10): 2044-2064.

[21] Derrac J, Garc I A 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 (S2210-6502), 2011, 1(1): 3-18.

[22] Zhan Z, Zhang J, Li Y, et al. Orthogonal Learning Particle Swarm Optimization[J]. IEEE Transactions on Evolutionary Computation (S1089-778X), 2011, 15(6): 832-847.

[23] Wang F, Zhang H, Li K, et al. A hybrid particle swarm optimization algorithm using adaptive learning strategy[J]. Information Sciences (S0020-0255), 2018, 436-437: 162-177.

[24] Gao W, Huang L, Liu S, et al. Artificial Bee Colony Algorithm Based on Information Learning[J]. IEEE Transactions on Cybernetics (S2168-2267), 2015, 45(12): 2827-2839.

[25] Cui L, Zhang K, Li G, et al. Modified Gbest-guided artificial bee colony algorithm with new probability model[J]. Soft Computing (S1432-7643), 2018, 22(7): 2217-2243.

[26] Lin Q, Zhu M, Li G, et al. A novel artificial bee colony algorithm with local and global information interaction[J]. Applied Soft Computing (S1568-4946), 2018, 62: 702-735.

DOI: 10.16182/j.issn1004731x.joss.18-0787