软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(10):3151−3175 [doi: 10.13328/j.cnki.jos.006030]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于多策略的改进花授粉算法*

肖辉辉[1,2], 万常选[1]

[1](江西财经大学 信息管理学院,江西 南昌 330013)

[2](河池学院 大数据与计算机学院,广西 河池 546300)

通讯作者: 万常选, E-mail: wanchangxuan@263.net

摘 要: 花授粉算法是近年来提出的一种新型的、简单高效的优化算法,已在各个领域得到广泛应用,但其搜索策略存在的不足,制约着其应用范围.为此,提出一种改进的基于多策略的花授粉算法.首先,新全局搜索策略通过利用两组随机个体差异矢量和莱维飞行机制来增加种群多样性并扩大搜索范围,使算法更易跳出局部最优,提升其开采能力;其次,在局部搜索部分引入精英变异策略,并与随机个体变异机制组合成一种新的局部授粉策略,利用精英个体对其他个体的演化方向进行引导,提高算法的搜索速度;通过随机个体变异策略来保持种群的多样性,增强算法的持续优化能力;同时,通过一种线性递减概率规则调节这两种变异策略,使其取长补短,以提高算法的优化能力;最后,对进化中没有得到改善的解,利用余弦函数搜索因子策略产生一个新解加以替换,从而提高算法解的质量.通过 5 类经典测试函数的仿真实验和采用统计学上的分析,证明了该算法的稳定性和有效性;与现有经典的和知名的改进算法进行了对比,实验结果表明,所提出的改进算法是一种富有竞争力的新算法.同时,利用改进算法对军事领域中的无人作战飞行器航线规划问题进行求解,测试结果表明,改进算法在解决实际工程问题时,同样具有一定的优势.

关键词: 花授粉算法;动态调整策略;余弦函数搜索因子;搜索方程;种群多样性

中图法分类号: TP18

中文引用格式: 肖辉辉,万常选.基于多策略的改进花授粉算法.软件学报,2021,32(10):3151−3175. http://www.jos.org.cn/1000-9825/6030.htm

英文引用格式: Xiao HH, Wan CX. Improved flower pollination algorithm based on multi-strategy. Ruan Jian Xue Bao/ Journal of Software, 2021,32(10):3151−3175 (in Chinese). http://www.jos.org.cn/1000-9825/6030.htm

Improved Flower Pollination Algorithm Based on Multi-strategy

XIAO Hui-Hui[1,2], WAN Chang-Xuan[1]

[1](School of Information Technology, Jiangxi University of Finance and Economics, Nanchang 330013, China)

[2](School of Big Data and Computer Science, Hechi University, Hechi 546300, China)

Abstract: The flower pollination algorithm (FPA) is a novel, easy and efficient optimization algorithm proposed in recent years. It has been widely used in various fields, but its search strategy has some defects, which become an impediment to its application. Therefore, this paper introduces an improved flower pollination algorithm based on multi-strategy. First, the new global search strategy was adopted through two groups of random individual difference vectors and Lévy flight to increase the diversity of population and expand the search range, making the algorithm easier to escape the local optimum and improve its exploitation ability. Second, the elite mutation strategy was used in the local search, and a new local pollination strategy was developed by combing it with the random individual mutation mechanism. The elite individuals were used to guide the evolution direction of other individuals and improve the search speed of the

* 基金项目: 国家自然科学基金(61972184, 61562032); 江西省自然科学基金(20152ACB20003); 河池学院高层次人才科研启动项目(2019GCC012)

Foundation item: National Natural Science Foundation of China (61972184, 61562032); Natural Science Foundation of Jiangxi Province (20152ACB20003); High-level Talent Research Start-up Project of Hechi University (2019GCC012)

收稿时间: 2019-04-16; 修改时间: 2019-08-22, 2019-12-19; 采用时间: 2020-01-31

algorithm. The random individual mutation strategy was adopted to keep the population diverse and enhance the continuous optimization capability of the algorithm. In addition, the two mutation strategies were adjusted through linear decreasing probability rule to make them complement with each other and improve the optimization capability of the algorithm. Finally, a new solution was generated by the cosine function search factor strategy to replace the unimproved solution and improve the quality of the solution. The stability and effectiveness of the algorithm were proved by simulation experiments of 5 kinds of classical test functions and statistical analysis. The experimental results show that the improved algorithm proposed in this paper is a novel and competitive algorithm compared with the existing classical and state-of-the-art improved algorithms. At the same time, the proposed algorithm was used to solve the route planning problem of unmanned combat aerial vehicle (UCAV) in the military field. The test results show that the proposed algorithm also has certain advantages in solving practical engineering problems.

Key words: flower pollination algorithm; dynamic adjustment strategy; cosine function search factor; search equation; population diversity

随着当今世界进入大数据时代,各类工程优化问题、科学计算问题等越来越复杂化,各个领域中的复杂优化计算也越发地成为社会急需解决的问题,如电力系统无功优化、物流中最优配送、大数据虚拟资源调度及化工过程最优控制等众多实际问题.传统的优化算法在解决这些大规模、高难度和不确定性的复杂实际工程优化问题时,在收敛精度、鲁棒性和收敛速度等方面都难以获得令人满意的结果.受自然界中各种演化规律及其生物种群行为的启发,众多国内外学者提出了一系列群智能优化算法,如粒子群算法[1]、蜂群优化算法[2]、差分进化算法[3]及教与学优化算法[4]等,并且成功地用于求解上述大量的工程问题.

受显花植物花授粉过程的启发,学者 Yang[5]构建了一种新的优化算法——花授粉算法(flower pollination algorithm,简称 FPA).FPA 算法是模拟自然界显花植物授粉繁殖后代的不同方式,它是一种基于种群集体搜索行为的启发式智能算法,与其他群智能算法相比,由于其概念简单、容易实现、寻优效率较高,并通过参数 p 较好地解决了探测和开采之间的平衡问题等优点,一经提出就引起国内外众多学者的广泛关注.目前,FPA 算法在多目标优化、无线传感器网络生命周期优化、神经网络优化、桁架结构优化、数据挖掘及能源[6−11]等众多领域中得到了广泛应用.然而 FPA 算法在解决众多工程问题时,存在后期收敛速度慢、收敛精度低和易陷入局部极小等不足,尤其是对于具有多局部极值点、高维较复杂的优化问题.为了提高 FPA 算法在实际应用中的优化能力,国内外众多学者对该算法进行了一系列研究和改进.从目前已出版的有关文献梳理结果来看,当前国内外学者对其研究主要从以下几个方面展开.

(1) 对算法的关键参数 p 值的设置及其动态调整策略进行了研究[12].

从算法的仿生原理可知:控制算法性能的参数较少,但关键参数 p 对该算法的优化能力具有较大的影响.

Madasu 等人[13]对 p 在[0.1,1]的取值进行了实验仿真分析,实验结果表明:当 p 在[0.5,0.6]之间取值时,FPA 算法的性能较好.且 Madasu 等人在文献[13]中指出:p=0.55 时,算法的优化性能最好.Draa[14]对关键参数 p 值的设置进行了定量仿真实验分析,仿真结果显示:当 p 取 0.2 时,FPA 算法的优化能力表现最优.

Mahdad 等人[15]根据迭代次数把转换概率 p 进行分阶段的动态自适应调整,同时将分区搜索机制引入到 FPA 算法中,然后利用改进的 FPA 算法求解带安全约束的优化功率流问题,在 IEEE 30-Bus 和 IEEE 57-Bus 电力测试系统上对燃料成本、功率损耗和电压偏差这 3 个不同目标进行优化,并与其他方法进行比较,实验结果表明:其优化效果要优于对比方法,同时,其鲁棒性也得到了较好的提升.

(2) 将其他优秀的算子引入到 FPA 算法中.

这类改进研究是根据著名的“无免费午餐定理”.每种元启发式算法都会存在各自的优点和缺点,故国内外众多学者对算法进行改进时,都把多种算子融合到一起,构成超大型智能优化算法,使其优势互补,提高算法的整体性能.

Zhou 等人[16]首先运用精英反向学习策略对 FPA 算法的全局授粉部分进行改进,提高其种群个体的多样性,从而有利于扩展其搜索领域;其次,他们还把自适应贪婪机制融入到局部搜索,改善其探索能力;最后,参数 p 采用动态调整策略.实验结果显示,FPA 算法的优化能力获得了较大提升.

肖辉辉等人[17]针对花授粉算法种群个体之间缺乏信息交流,容易导致算法陷入早熟的问题;同时,为了将进

化中的最差个体变为较好个体,改善解的质量,构建了基于复合形法的花授粉算法模型.测试结果验证了改进方法的有效性.

Chakraborty 等人[18]在 FPA 算法中融入差分进化思想:先运用差分进化算法来改善 FPA 算法初始解的质量;然后,为了消除 p 对 FPA 算法优化性能的不利影响,增强 FPA 算法的鲁棒性,受粒子群优化算法设计思想的启发,通过引入两个动态参数,把 FPA 算法的两个搜索方程合二为一.测试结果表明,改进策略提高了 FPA 算法的收敛能力.

Wang 等人[19]针对 k-means 算法在求解聚类问题时容易陷入局部最优且解的质量高度依赖初始解等问题,提出了用改进的 FPA 算法求解聚类问题.该算法首先在 FPA 算法的全局搜索部分引入人工蜂群算法中的丢弃解操作,该策略有利于个体跳出局部最优;其次,再将基于精英的变异算子和交叉算子融入到 FPA 算法的局部搜索中,分别用来提高算法的开采能力和增加种群的多样性.

Draa[14]将反向学习方法融入到 FPA 算法中,实验结果显示,该措施能够有效提升算法的优化性能.

肖辉辉[20]构建了一种基于引力搜索机制的花授粉算法,对全局寻优部分进行改进:采用花朵个体间的万有引力和算法本身的莱维飞行共同实现个体位置的更新,使花朵个体受莱维飞行和个体间引力的双重影响.在进化中,种群个体利用万有引力的相互作用实现优化信息的共享及向质量大(最优位置)的个体靠近,且个体间的万有引力也牵制莱维飞行的随机游走;同时又运用莱维飞行的跳跃及不均匀性步长,避免个体陷入局部极值,从而提高算法的寻优能力.实验结果表明,改进方法能够有效提升 FPA 算法的性能.

以上文献中的改进方法均能有效提升 FPA 算法的寻优性能,但都没有从 FPA 算法的搜索策略所存在的固有缺陷这个角度对其进行改进,也没有考虑在进化中劣解对 FPA 算法寻优能力的影响;同时,上述改进算法在求解高维多峰的复杂优化问题时,其优化性能还不够理想,并且有些改进算法修改了 FPA 算法的演化思路,另一些改进算法与 FPA 算法相比,其时间复杂度较高.

针对上述存在的问题,本文构建了一种改进的基于多策略的花授粉算法(an improved flower pollination algorithm based on multi-strategy,简称 MIFPA).该算法在全局搜索部分引入两组随机个体差异矢量以增强算法的扰动性,提高个体在进化后期逃离局部极值的概率;将精英变异策略融入到局部授粉,并通过一个线性递减概率规则与基本 FPA 算法的变异机制构成新的局部搜索策略,在较好地保持算法种群多样性的同时,加速了算法的收敛速度;利用余弦函数搜索因子方法对进化中的劣解进行改进,以提高算法解的质量.上述改进方法的有机融合,构成了 MIFPA 算法的主要框架,这些方法在算法的不同进程中执行,并相互协调地提升 FPA 算法的整体性能.新算法中的参数实施自适应调整策略,增强了算法的灵活性.其实现非常简单,且既不会大幅增加 FPA 算法的时间复杂度,也未改变 FPA 算法的基本演化思想.实验结果表明,MIFPA 算法简单且高效、求精能力强、收敛速度快和鲁棒性好,在单模态高维函数、多模态高维函数、多模态低维函数、带旋转的多模态高维函数和变换旋转的复杂函数上的优化能力与已有典型的改进 FPA 算法和其他知名改进算法相比,显示出其良好的竞争力;同时,在求解实际的复杂工程问题时,与其他对比算法相比,MIFPA 算法的优势也表现突出.

1 基本花授粉算法

FPA 算法概念简单、容易实现及寻优效率较高,且通过参数 p 实现了其异花授粉和自花授粉之间的相互切换,平衡了算法的探索和开发能力.同时,FPA 算法利用了莱维飞行策略来实现对个体位置的更新,扩展其开采领域,从而达到提高算法的优化能力.

为了更好地利用 FPA 算法解决实际优化问题,Yang[5]在算法中假定每株显花植物只开一朵花,每朵花只有一个花粉配子,一个配子对应于求解问题的一个解.此外,还需假定以下 4 条规定.

(1) 显花植物的异花授粉对应于算法的勘探行为,全局授粉是由蜜蜂等传播者携带花粉并采用莱维飞行来实现;

(2) 非生物自花授粉对应于算法的开采行为,局部授粉是通过同种植物的不同花朵之间传粉来实现;

(3) 繁衍概率对应于花的常性,进化中两个个体(花朵)之间的相似性与其值大小具有一定的比例关系;

(4) 参数 $p\in[0,1]$对 FPA 算法的勘探(全局授粉)和开采(局部授粉)之间的相互转换进行调节,且因受花朵个体之间所处位置的邻近性和风等其他自然因素影响,使得授粉更偏重于自花授粉[21].

从上述可知,异花授粉(全局授粉)和自花授粉(局部授粉)是 FPA 算法的核心,算法的全局授粉(优化)通过公式(1)实现,局部授粉(优化)通过公式(5)实现.

全局授粉(优化)的进化公式如下:

$$x_i^{t+1}=x_i^t+\gamma L(\lambda)(x_i^t-x_{best}) \tag{1}$$

其中, x_i^t、x_i^{t+1} 分别对应于迭代 t 次、$(t+1)$次后获得的解,x_{best} 是每次迭代后获得的最优解,γ是控制步长的缩放因子[20],$L(\lambda)$是对应于花朵个体的莱维飞行位移.$L(\lambda)$的计算如公式(2):

$$L(\lambda)\sim\frac{\lambda G(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}(s>>s_0>0) \tag{2}$$

其中,λ=3/2,$G(\lambda)$是标准伽马函数,s 由公式(3)得到:

$$s=\frac{\mu}{|\nu|^{1/\lambda}} \tag{3}$$

其中,$\mu\sim N(0,\sigma^2)$,$\nu\sim N(0,1)$,σ^2 由公式(4)计算获得:

$$\sigma^2=\left\{\frac{G(1+\lambda)}{\lambda G[(1+\lambda)/2]}\cdot\frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}}\right\}^{1/\lambda} \tag{4}$$

花朵个体的局部授粉(优化)可通过公式(5)进行局部搜索:

$$x_i^{t+1}=x_i^t+\varepsilon(x_j^t-x_k^t) \tag{5}$$

其中, x_j^t、x_k^t 是优化过程中两个不同的随机解,ε是区间 0~1 上的一个均匀分布随机数.

2 基于多策略改进的花授粉算法

依据基本 FPA 算法的仿生原理可知:其是由全局搜索和局部搜索通过参数 p 进行融合组成的,并利用公式(1)和公式(5)实现了数学模型化,为其解决一系列复杂的优化问题奠定了理论基础.而对基本 FPA 算法的搜索机制进行定性分析显示:基本 FPA 算法的搜索策略存在的不足制约着算法的收敛效果,包括存在收敛速度慢和易陷入局部最优等缺点.对于这些影响算法性能的不利因素,本文对 FPA 算法进行了多策略改进.

2.1 新全局搜索策略

从技术角度来说,FPA 算法在全局授粉时,莱维飞行和最优个体(x_{best})对种群中的个体同时施加影响.由于受全局最优个体 x_{best} 的吸引,FPA 算法在优化简单问题时具有较快的收敛速度;但在解决复杂的优化问题时,若种群中的个体 x_{best} 陷入到探索领域中的某些局部极小位置,则其他个体受 x_{best} 的影响,也快速移动到 x_{best} 所在的位置,使得($x_i^t-x_{best}$)变得非常小,从而造成个体位置更新公式(1)无效,因为 $x_i^{t+1}=x_i^t+0=x_i^t$. 在这种情况下,种群将停止进化,并且很难逃离局部最优.为了解决该问题,本文利用公式(6)对公式(1)进行改进:

$$x_i^{t+1}=x_i^t+\gamma L(\lambda)(x_i^t-x_{best}+x_{i_1}^t-x_{i_2}^t+x_{i_3}^t-x_{i_4}^t) \tag{6}$$

其中,i、i_1、i_2、i_3、i_4 分别是从当前群体中随机选取的 5 个不同个体的下标,其余变量的含义同公式(1).

从公式(6)可以看出:在莱维飞行机制的基础上,引入了两组差异矢量,增加了种群个体之间的差异性,提高了算法在多维空间的探索能力,有利于抑制算法早熟收敛,提升算法的性能.

2.2 引入精英变异策略的局部搜索策略

在基本花授粉算法的局部搜索部分,利用公式(5)的变异操作产生一个新个体.从公式(5)可知:新个体是在父代个体的基础上加上一个扰动项,其值是一个随机数和两个个体差分矢量的乘积.于是,新个体的产生具有较大的随机性,这使得种群个体的多样性能够较好地得到维持,从而使算法能够保持良好的持续优化能力.但是由于个体缺乏对种群中最优个体良好经验的继承和学习机制,个体的进化方向具有很强的盲目性,这导致了对搜索

全局最优解的计算量增大,降低了算法的收敛速度.

为了解决上述存在的这个问题,基于差分进化算法的思想和 FPA 算法的特性,本文把差分进化算法中的经典变异算子 DE/best/2 的思路融入到局部授粉中,提出一种新的复合型局部搜索机制:如果 $rand<\zeta$,则按式(7)进行处理;否则按式(8)进行处理.

$$v_i = x_i + \delta(x_{r_2} - x_{r_3}) \tag{7}$$

$$v_i = x_{best} + \alpha(x_{r_1} - x_{r_2} + x_{r_3} - x_{r_4}) \tag{8}$$

其中,$rand$ 是[0,1]上服从均匀分布的随机数;$\zeta=1-t/\text{Max\_}iter$,$t$ 是当前迭代次数,Max_$iter$ 为最大迭代次数;参数δ、α是服从高斯分布且均值和标准偏差分别为 0.5、0.1,其作用是用于控制算法的演化速度;n 为种群数,$i\in(1,2,\ldots,n)$为当前个体的下标,r_1、r_2、r_3、r_4 为 4 个不同的随机个体的下标,x_{best} 为当前种群中最优个体.

从上述改进的局部授粉策略可知:公式(7)采用变异策略增加种群个体的差异性,从而达到提升算法的全局优化能力,但算法的搜索速度比较慢;对于公式(8)运用的变异机制,是通过精英个体对其他个体的演化方向进行引导,同时利用精英个体的信息有利于开发其周围领域,提高 FPA 算法的搜索效率,从而提升了算法的收敛速度和开采能力,但这一策略也容易导致 FPA 算法陷入局部极小问题.为了能够使这两种方法优势互补,提高 FPA 算法的寻优能力,本文通过一个线性递减概率规则融合这两种变异机制,构建 MIFPA 算法的新局部搜索策略.

同时,从上述新的局部搜索策略可以看出:在进化过程中,通过线性递减概率规则"$rand<\zeta$"对两种变异策略进行调节,如果 $rand$ 产生的随机数小于ζ,则个体执行公式(7)的变异机制;否则,执行公式(8)的变异策略.

依据$\zeta=1-t/\text{Max_}iter$ 可知:在算法进化初期,选择公式(7)的变异策略的概率要比选择公式(8)的变异机制的概率大,这有利于种群个体在演化初期扩大搜索空间.因为从公式(7)可以发现:在算法进化初期,由于个体之间的差异性比较大,则 $(x_{r_2} - x_{r_3})$ 值较大,这使得种群个体有利于向更广的搜索范围进行扩散,易于算法找到最优值.在算法的演化后期,个体选择公式(8)的变异策略的概率要比选择公式(7)的变异机制的概率大,这有利于加快算法的收敛速度.因为公式(8)利用精英个体对种群其他个体的演化方向进行了引导,促进种群的其他个体加速向精英个体靠近,达到提高算法的搜索速度和计算精度.综上所述,通过线性递减概率规则可使这两种变异策略优势互补,提升算法的收敛能力.

2.3 对劣解的改进策略

在基本花授粉算法中,总是以贪婪式演化策略选择较优个体来保证群体向前进化,即:经过全局搜索或局部搜索后,产生一个新个体,只有当子代的适应度值优于父代才能演化.然而,依据 FPA 算法的具体流程可知:如果子代劣于父代,则父代直接保留到下一代,并没有对其作任何改进措施,算法直接进入下一次进化,这使得本次迭代的计算资源严重浪费,增加了对全局最优解的搜索计算量,降低了算法的收敛速度,且容易造成算法收敛精度不高.

针对基本花授粉算法存在的这一不足,在算法进入下次迭代之前,如果父代没有得到改善,则利用公式(9)重新产生一个新个体,若新解优于原始解,则用新解代替原始解:

$$x^{new}=2\times\cos((\pi\times t)/(2\times\text{Max\_}iter))\times\varphi\times x_t(r) \tag{9}$$

其中,φ是[−1,1]上服从均匀分布的随机数,其作用是使种群个体实现随机游走;Max_$iter$ 为最大迭代次数,$x_t(r)$是迭代次数为 t 时种群中的一个随机个体;x^{new} 为产生的新个体,$t=1,2,\ldots,\text{Max\_}iter$.

由上述可以看出:原始解的质量获得提升的概率得到提高,从而达到提高FPA算法解的质量.下面就公式(9)可改善算法解的质量、提高算法优化能力的原因进行分析.

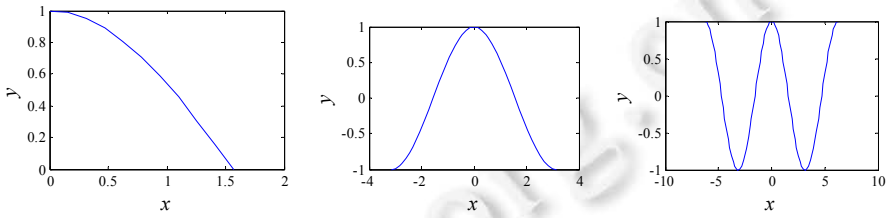
(1) 公式(9)能够有效避免随机产生的新个体(解)的质量太差(新个体远离全局最优个体 x_{best})而影响算法的搜索速度.从公式(9)中可以看出:新个体的产生是以 $x_t(r)$为中心,充分利用原有个体的信息引导新个体在其四周产生,使其产生的新个体的位置距全局最优个体 x_{best} 更近,避免了由于产生的新个体质量差而导致算法的收敛速度慢.

(2) 依据上述新的全局搜索方程(6)和新的局部搜索方程(8)可以发现:在新构建的搜索策略中都采用了最

优个体(x_{best})策略,这增加了算法陷入局部最优的风险.为了有效防止 FPA 算法早熟,本文利用公式(9)来产生一个新个体以替换没有得到改善的父代个体.从公式(9)可以发现:新个体的产生利用了余弦函数因子,使得算法有利于跳出局部最优.这是因为,余弦函数因子与线性递减因子相比,余弦函数因子具有振荡性.

本文通过融入余弦函数搜索因子,利用其振荡特点使得种群个体位置具有振荡性,扩展个体的搜索范围,有效引导个体跳出局部最优,提高解的质量.

图 1 是余弦函数在 x 不同取值范围内的曲线图,从图 1 可知:随着 x 取值的不断扩大,余弦函数的峰值个数增多,这说明余弦函数的振荡性更加剧烈.但是,如果算法的振荡性太强,不利于算法快速收敛,甚至造成算法找不到全局最优解.鉴于此,本文把余弦函数的 x 取值设为$\pi\times t/(2\times \text{Max_}iter)\in(0,\pi/2)$,以减缓个体的振荡性,使算法获得更优质量的解.



(a) $x\in(0,\pi/2)$ (b) $x\in(-\pi,\pi)$ (c) $x\in(-2\pi,2\pi)$

Fig.1 Curve graph of cosine functions within the range of different values of x

图 1 余弦函数在 x 不同取值范围内的曲线图

(3) 同时,利用公式(9)中的余弦函数搜索因子,其作用是对产生的新个体半径范围进行调节.其思想是:在算法进化初期,$\cos(\pi\times t/(2\times \text{Max_}iter))$值较大,则以原有个体 $x_i(r)$为中心的搜索半径较大,使得算法具有良好的全局搜索能力;而随着演化的不断深入,其搜索半径越来越小,因此到进化的后期,新个体更侧重于局部精细化开采.由此可知:通过利用余弦函数因子,可使算法具有良好的全局搜索和局部搜索平衡性,提高了算法的全局优化能力,并找到最优解.

2.4 MIFPA算法的流程

根据第 2.1 节~第 2.3 节描述的算法改进思想,本节给出 MIFPA 算法流程,如算法 1 所示.其中,n 为花朵的个数,x_{best} 是全局最优解,$f_{\min}$ 是 x_{best} 对应的最优值.

Aglorithm 1. MIFPA.

1. Randomly produce n flowers $\{x_i|i=1,2,\ldots,n\}$;
2. Compute the fitness $fun(x_i)$;
3. $f_{\min}=\min(fun(x_i))$;
4. Find the best individual x_{best} in the initial population;
5. $FEs=n$;
6. **while** $FEs\leqslant \text{Max\_}FEs$ **do**
7. **for** $i=1$: n **do**
8. Generate a new p according to Eq.(11);
9. **if** $p>rand(0,1)$ //global optimization
10. Randomly produce four flowers (solutions);
11. produce a new candidate x_i^{t+1} based on Eq.(2)~Eq.(4) and Eq.(6);
12. **else** // local optimization
13. Randomly generate four flowers (solutions);
14. **if** $rand(0,1)<\zeta$

15. Generate a new candidate x_i^{t+1} according to Eq.(7);
16. **else**
17. Generate a new candidate x_i^{t+1} according to Eq.(8);
18. **endif**
19. **endif**
20. Evaluate the newly generated solution x_i^{t+1};
21. $FEs=FEs+1$;
22. **if** $fun(x_i^{t+1}) < fun(x_i^t)$
23. replace x_i^t by x_i^{t+1};
24. **else**
25. Generate a new candidate x_i^{t+1} according to Eq.(9);
26. Evaluate the newly generated solution x_i^{t+1};
27. $FEs=FEs+1$;
28. **if** $fun(x_i^{t+1}) < fun(x_i^t)$
29. replace x_i^t by x_i^{t+1};
30. **endif**
31. **endif**
32. **endfor**
33. Update the current best solution x_{best};
34. **endwhile**

MIFPA 算法与基本 FPA 算法的框架基本类似,但其中主要有 4 个不同之处:① 转换概率 p 采用自适应调整策略,详见第 8 行;② 增加了两组随机个体的差异矢量到全局搜索(全局授粉)中,详见第 10 行、第 11 行;③ 采用一个线性递减概率规则融合两种变异机制进行局部搜索(局部授粉),详见第 14 行~第 18 行;④ 利用余弦函数搜索因子对算法中的劣解进行改进,详见第 25 行.

2.5 MIFPA算法的复杂度分析

衡量改进算法的有效性和可行性,一是算法的寻优性能要有较大的提高,二是算法的时间复杂度也不能比原算法高太多.在群智能优化算法中,时间复杂度是指通过优化算法求解优化问题的最优解或次优解所需要的进化次数.下面对 MIFPA 算法的复杂度进行分析.

假设目标函数设为 $f(x)$,算法的种群规模设为 n,求解问题的维数设为 D,$f(D)$为计算给定解的目标函数的执行时间与求解问题维数 D 有关的函数,则依据算法的描述和符号 O(时间复杂度符号)的计算规则为:

- 在初始化种群并计算得到最优值和最优解阶段的时间复杂度为 $O(n(r_1D+f(D)))=O(D+f(D))$,其中,$r_1$ 为产生均匀分布随机数的执行时间;
- 全局优化部分的时间复杂度为 $O(n(r_2D))=O(D)$,其中,r_2 为式(1)生成新解时一维变量的执行时间;
- 局部优化部分的时间复杂度为 $O(n(r_3D))=O(D)$,其中,r_3 为式(5)生成新解时一维变量的执行时间;
- 更新全局最优值和全局最优解部分的时间复杂度为 $O(n(r_4+r_5D+f(D)))=O(D+f(D))$,其中,$r_4$、$r_5$ 分别为比较新解与旧解和替换一维旧解变量的执行时间.

所以,FPA 算法的时间复杂度[20]表示为 $T(\text{FPA})=2O(D+f(D))+2O(D)=O(D+f(D))$.从第 2.4 节的算法流程可知:公式(6)~公式(9)是改进算法修改之处,即包括新的全局搜索策略、新的局部搜索策略和对质量不好的解的改进方法.依据上述对算法公式的描述和符号 O 的计算规则,可推导出 MIFPA 的时间复杂度为

$$T(\text{MIFPA})=T(\text{FPA})+T(\text{公式(6)})+T(\text{公式(7)})+T(\text{公式(8)})+T(\text{公式(9)})$$
$$=O(D+f(D))+O(n(r_6D))+O(n(r_7D))+O(n(r_8D))+O(n(r_9D))$$

$$=O(D+f(D))+4O(n(D))$$
$$=O(D+f(D)),$$

其中,r_6~r_9分别为公式(6)~公式(9)中生成新解时一维变量的执行时间.因此,MIFPA 算法的时间复杂度与 FPA 算法的时间复杂度属于同一数量级,也就是说,改进算法的复杂度并未比 FPA 算法提高太多.

根据上述对 MIFPA 算法的时间复杂度的理论分析(定性分析)可知,MIFPA 算法的时间复杂度与基本 FPA 算法属于同一数量级.因为 FPA 的迭代次数为 Max_*Gf*=Max_*FES*/*n*,其中,*n* 为种群数,MIFPA 算法的最大迭代次数为 Max_*FES*/(2×*n*)<Max_*Gc*<Max_*FES*/*n*,故 Max_*Gc*<Max_*Gf*.从上述分析可知:在相同的最大函数评估次数 Max_*FES*=10000×*D*(其中,*D* 为函数的维数)下,MIFPA 算法并没有增加算法的时间复杂度.

3 实验仿真及分析

3.1 实验测试函数及参数设置

为了检验 MIFPA 算法的有效性,本节选用 19 个经典的标准测试函数(包含 CEC 测试函数)进行实验,测试函数的函数名称、搜索空间、维数及理论最优值的描述见表 1[22,23].按其特性,可将测试函数分为五大类:f_1~f_4是单模态高维函数;f_5~f_9是非旋转的多模态高维函数;f_{10}~f_{13}是多模态低维函数;f_{14}~f_{16}是带旋转的多模态高维函数;f_{17}~f_{19}是变换旋转高维函数.本文中所用的所有函数都是求解最小值.

Table 1 Experimental test functions in the paper

表 1 本文使用的实验测试函数

测试函数	函数名	搜索范围	维度	最优值 $f(x^*)$
f_1	Sphere	[−100,100]	30,50,100	0
f_2	Schwefel 1.2	[−100,100]	30,50,100	0
f_3	Rosenbrock	[−30,30]	30,50,100	0
f_4	Quartic with noise	[−1.28,1.28]	30,50,100	0
f_5	Rastrigin	[−5.12,5.12]	30,50,100	0
f_6	Ackley	[−32,32]	30,50,100	0
f_7	Griewank	[−600,600]	30,50,100	0
f_8	Generalized penalized 1	[−50,50]	30,50,100	0
f_9	Generalized penalized 2	[−50,50]	30,50,100	0
f_{10}	Kowalik	[−5,5]	4	0.000 307 5
f_{11}	Shekel5	[0,10]	4	−10.153 2
f_{12}	Shekel7	[0,10]	4	−10.402 9
f_{13}	Shekel10	[0,10]	4	−10.536 4
f_{14}	Rotated rosenbrock's function	[−2.048,2.048]	30,50	0
f_{15}	Rotated griewank's function	[−600,600]	30,50	0
f_{16}	Rotated ackley's function	[−32.768,32.768]	30,50	0
f_{17}	Shifted sphere function	[−100,100]	30,50	−450
f_{18}	Shifted rosenbrock's function	[−100,100]	30,50	390
f_{19}	Shifted rotated ackley's function with global optimum on bounds	[−32,32]	30,50	−140

3.2 余弦函数搜索因子对算法性能的影响分析

为了更直观地进一步说明本文引入余弦函数搜索因子能够提高算法解的质量这一点,本节对带余弦函数搜索因子的花授粉算法(flower pollination algorithm with cosine function factor,简称 CFPA)与基本花授粉算法(FPA)进行定量分析.本节利用表 1 中经典的标准测试函数(包含 CEC 测试函数)进行实验测试,其实验参数设置为:种群个数 *n*=50;函数维数 *D*=30 或 4;最大函数评估次数 Max_*FES*=10000×*D*,独立运行 30 次.其中,*Mean_error*(平均值误差)通过公式(10)计算:

$$Mean\_error=f(x)-f(x^*) \tag{10}$$

其中,x 为算法当前获得的解,x^*为算法当前求解到的全局最优解.

由公式(10)可知:*Mean_error* 越小,则算法解的质量越好.

实验结果见表 2,其中,加粗的数值表示该算法在该测试函数上取得的最优计算结果.

Table 2 *Mean_error* of the two algorithms (D=30,4)

表 2 两种算法的 *Mean_error*(D=30,4)

测试函数	*Mean_error*		测试函数	*Mean_error*	
	FPA	CFPA		FPA	CFPA
f_1	5.54E−08	**0.00E+00**	f_{11}	**3.21E−07**	**3.21E−07**
f_2	2.85E−04	**0.00E+00**	f_{12}	**4.06E−05**	**4.06E−05**
f_3	**2.08E+01**	2.59E+01	f_{13}	**9.82E−06**	**9.82E−06**
f_4	1.86E−02	**4.67E−06**	f_{14}	1.17E+03	**5.70E+02**
f_5	6.12E+01	**0.00E+00**	f_{15}	1.74E+02	**0.00E+00**
f_6	1.50E+00	**8.88E−16**	f_{16}	7.67E−01	**0.00E+00**
f_7	8.49E−05	**0.00E+00**	f_{17}	**3.60E−07**	1.83E−01
f_8	1.34E−02	**6.86E−07**	f_{18}	**1.52E+02**	3.65E+04
f_9	**2.09E−05**	1.76E+00	f_{19}	2.10E+01	**2.03E+01**
f_{10}	**1.40E−08**	**1.40E−08**	–	–	–

从表 2 可知:CFPA 算法在 19 个测试函数上取得了 15 个最优值,而 FPA 算法取得 8 个最优值.这表明,本文把余弦函数搜索因子融入到算法中,能够提高算法解的质量.

为了更好地阐述带余弦函数搜索因子的花授粉算法与基本花授粉算法之间的不同,本文利用这两种不同的算法来优化二维的 Griewank 函数,该函数是非线性多模态函数,具有众多局部最小值和一个全局最优值.

基本 FPA 算法和带余弦函数搜索因子的 FPA 算法分别在第 1 次、第 10 次、第 20 次迭代后的种群分布结果如图 2 所示,其中,图 2(a)~图 2(c)是基本 FPA 算法的种群分布图,图 2(d)~图 2(f)是带余弦函数搜索因子的 FPA 算法的种群分布图.

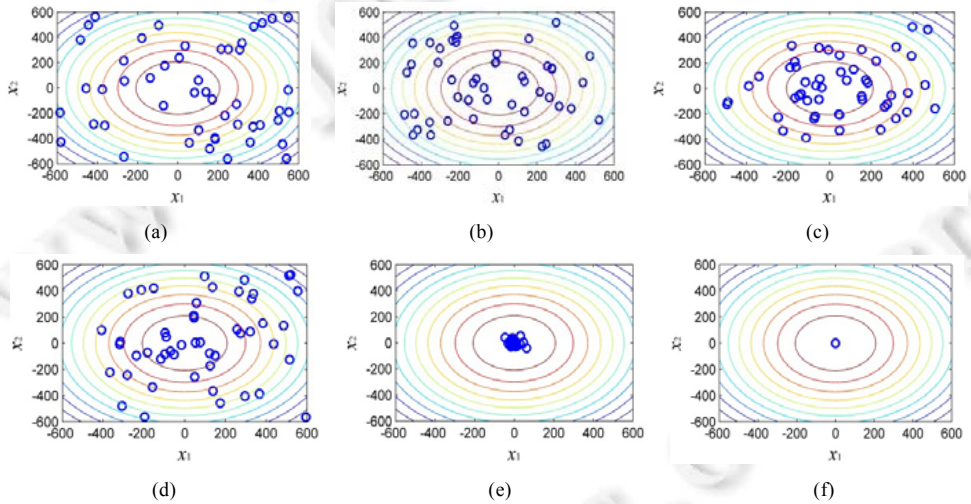


Fig.2 Population distribution of the 2 algorithms after first, 10th and 20th iterations respectively

图 2 两种算法分别在第 1 次、第 10 次、第 20 次迭代后种群的分布

从图 2 可以看出:两种算法在开始进化时(第 1 次迭代后),种群几乎覆盖了整个搜索空间;但是随着进化的不断深入,改进的 FPA 算法的收敛性能显著地优于基本 FPA 算法.这说明,通过引入余弦函数搜索因子来提高算法解的质量,能够帮助 FPA 算法快速地收敛到全局最优解.

3.3 *p*对算法性能的影响分析及动态调整策略

p 是 FPA 算法的重要参数,而依据基本 FPA 算法提供的参数可知:对每个优化问题,p 都设置为固定的值 0.8.受其他智能算法参数研究经验的启发,转换概率参数 p 的取值应该是依赖于不同优化问题,其取值对解决不同的优化问题应该是敏感的.为了研究上述结论是否成立,本节利用基本 FPA 算法对两个经典函数 Sphere(单模态函数)及 Ackley(多模态函数)进行优化,研究 p 的取值对 FPA 性能的影响是否具有敏感性.实验参数设置:p 取值

范围[0.001:0.001:0.01,0.02:0.01:0.1,0.2:0.1:1],每个函数的问题维数设置为 D=30,种群数设置为 n=50,最大评估次数设置为 10000×D;为了减小实验误差,对 p 的每个值,算法都独立执行 30 次,求其平均值的误差.实验结果如图 3 所示,图中的 Fsph 和 Fack 分别表示函数 Sphere 和 Ackley.

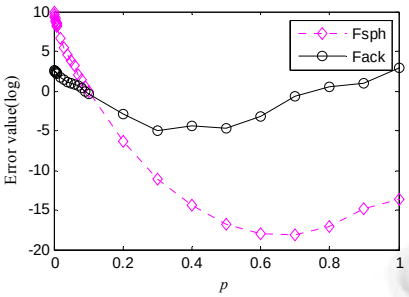


Fig.3 Performances of FPA with different p values

图 3 p 取不同值时,FPA 的性能

从图 3 可以看出:

(1) 对于函数 Ackley,p=0.3 时,FPA 的性能最好;对于函数 Sphere,p=0.7 时,FPA 获得的性能最优;

(2) 如果 p 的取值太大或太小,FPA 的性能都较差;

(3) FPA 在求解不同的优化问题时,其性能依赖于 p 的取值.

从上述实验可知:如果转换概率 p 取固定的值,不利于 FPA 找到全局最优解.为了解决这一不足,本文采用公式(11)对 p 进行自适应调整,提高 FPA 算法的性能和灵活性:

$$p=p_{\min}+(p_{\max}-p_{\min})\times((\text{Max\_}iter-t)/\text{Max\_}iter) \tag{11}$$

其中,$p_{\max}$、$p_{\min}$ 分别是 p 的最大值(本文取 0.9)和最小值(本文取 0.2),Max_$iter$ 是最大迭代次数,t 是当前迭代次数.

根据上述公式(11)和改进算法的流程可知:在算法的进化初期,变量 t 的值较小,则转化概率 p 的取值较大,算法更偏向于异花授粉(全局搜索);当算法进入演化后期,变量 t 的值越来越大,转换概率 p 的值越来越小,则算法更倾向于自花授粉(局部搜索).综上所述,通过 p 的自适应调整策略,能够更有效地解决算法的全局搜索和局部搜索之间的平衡问题,从而更有利于提高算法的全局优化能力.

3.4 MIFPA算法的有效性分析

为了验证本文算法的可行性和正确性等性能,本文在 5 类共 19 个测试函数上,从如下几个方面进行实验对比与分析.

(1) 解的质量对比分析;

(2) 算法的维度扩展性分析;

(3) 算法的鲁棒性和收敛速度分析;

(4) Friedman 与 Wilcoxon 检验;

(5) 运行时间比较分析.

为了检验 MIFPA 算法在性能上是否具有较大的提升,本节除了与基本 FPA 算法对比外,还将其与异构综合学习 PSO(heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation,简称 HCLPSO)算法[24]、基于多种群变异策略集成的差分进化算法(differential evolution with multi-population based ensemble of mutation strategies,简称 MPEDE)[25]、融入精英反向学习策略的 FPA 算法(elite opposition-based flower pollination algorithm,简称 EOFPA)[16]、引入广义反向学习机制的 FPA 算法(modified generalised oppsition-based flower pollination algorithm,简称 MGOFPA)[14]分别从上述几个方面进行比较分析.

3.4.1 解的质量对比分析

为了验证本文算法的有效性,并能较好地防止算法陷入局部最优,佐证其解的质量的优越性,同时为了减少

实验误差,本文对每种比较算法在每个测试函数上都分别独立执行 30 次,计算其 *Mean_error*(平均值误差)、*Std.Dev*(标准方差).实验的其他参数设置为:种群个数 n=50;函数维数 D=30 或 4;最大评估次数为 10000×D;转换概率 p=0.8[5,26–29].

为了对所有算法的优化能力做出公平公正的评估,本文对所有测试函数的测试结果分别利用 Wilcoxon(威尔科克森)秩和检验(a=0.05)进行实验分析,测试结果见表 3.其中,符号"†""≈""‡"分别表示 MIFPA 算法解的质量好于、相当于或差于对比算法,符号"*w*/*t*/*l*"分别表示 MIFPA 算法解的质量有 w 个函数优于对比算法、t 个函数与对比算法相当、l 个函数差于对比算法.

Table 3 Optimal mean error values and standard deviations of the six algorithms (D=30,4)

表 3 6 种算法的优化均值误差和标准差(D=30,4)

测试函数	*Mean_error±Std.Dev*		
	HCLPSO	MPEDE	EOFPA
f_1	2.70E−48±4.43E−48†	3.41E−42±1.10E−41†	**0.00E+00±0.00E+00**≈
f_2	8.18E−05±9.63E−05†	8.05E−28±3.71E−27†	**0.00E+00±0.00E+00**≈
f_3	7.44E+00±4.23E+00†	1.33E−01±7.28E−01†	1.51E+01±1.07E+00†
f_4	2.60E−03±1.00E−03†	8.92E−04±3.09E−04†	2.37E−04±2.14E−04†
f_5	9.66E−14±8.33E−14†	**0.00E+00±0.00E+00**≈	**0.00E+00±0.00E+00**≈
f_6	2.11E−14±5.04E−15†	4.44E−15±0.00E+00†	**8.88E−16±0.00E+00**≈
f_7	8.22E−04±2.50E−03†	3.29E−04±1.80E−03†	**0.00E+00±0.00E+00**≈
f_8	**1.57E−32±5.57E−48**‡	**1.57E−32±5.57E−48**‡	9.17E−28±4.17E−27†
f_9	**1.35E−32±5.57E−48**‡	**1.35E−32±5.57E−48**‡	3.50E−03±1.89E−02†
f_{10}	**1.40E−08±3.31E−18**‡	1.50E−04±2.76E−04†	9.94E−05±2.78E−04†
f_{11}	**3.21E−07±3.24E−16**≈	**3.21E−07±0.00E+00**≈	1.36E+00±2.29E+00†
f_{12}	4.06E−05±5.42E−16≈	4.06E−05±0.00E+00≈	5.32E−01±1.62E+00†
f_{13}	9.82E−06±7.23E−16†	9.82E−06±4.51E−16≈	3.61E−01±1.37E+00†
f_{14}	4.70E+02±2.18E+02‡	**7.18E−24±3.93E−23**‡	8.43E+02±1.62E+02†
f_{15}	1.61E+02±1.42E+01†	1.40E+02±1.07E+01†	**0.00E+00±0.00E+00**≈
f_{16}	5.60E−03±2.36E−02†	**0.00E+00±0.00E+00**≈	**0.00E+00±0.00E+00**≈
f_{17}	7.01E−14±2.45E−14†	**0.00E+00±0.00E+00**≈	**0.00E+00±0.00E+00**≈
f_{18}	5.34E+00±4.32E+00†	3.13E+00±1.27E+01†	2.65E+01±5.50E+01†
f_{19}	2.08E+01±9.64E−02‡	2.09E+01±5.22E−02≈	**2.02E+01±2.71E−01**‡
w/*t*/*l*	12/2/5	9/7/3	10/8/1

Table 3 Optimal mean error values and standard deviations of the six algorithms (D=30,4) (Continued)

表 3 6 种算法的优化均值误差和标准差(D=30,4)(续)

测试函数	*Mean_error±Std.Dev*		
	MGOFPA	FPA	MIFPA
f_1	1.12E−123±2.98E−123†	5.54E−08±4.01E−08†	**0.00E+00±0.00E+00**
f_2	1.48E−39±4.95E−39†	2.85E−04±3.29E−04†	**0.00E+00±0.00E+00**
f_3	2.42E+01±5.97E−01†	2.08E+01±4.41E+00†	**7.53E−04±9.86E−04**
f_4	9.04E−05±6.47E−05†	1.86E−02±7.00E−03†	**4.87E−06±4.14E−06**
f_5	7.94E−01±1.88E+01†	6.12E+01±9.70E+00†	**0.00E+00±0.00E+00**
f_6	4.44E−15±0.00E+00†	1.50E+00±7.25E−01†	**8.88E−16±0.00E+00**
f_7	**0.00E+00±0.00E+00**≈	8.49E−05±1.29E−04†	**0.00E+00±0.00E+00**
f_8	6.18E−05±4.33E−05†	1.34E−02±3.74E−02†	1.69E−32±7.26E−34
f_9	5.26E−05±3.60E−05†	2.09E−05±2.72E−05†	2.79E−32±1.20E−32
f_{10}	1.40E−08±5.31E−14‡	1.40E−08±1.24E−19†	1.40E−08±7.22E−20
f_{11}	3.21E−07±8.85E−16†	**3.21E−07±0.00E+00**≈	**3.21E−07±0.00E+00**
f_{12}	**4.06E−05±8.54E−16**‡	4.06E−05±5.42E−16≈	4.06E−05±4.51E−16
f_{13}	**9.82E−06±1.00E−15**‡	9.82E−06±0.00E+00≈	9.82E−06±0.00E+00
f_{14}	6.49E+02±8.58E+01†	1.17E+03±2.28E+02†	5.74E+02±9.53E+01
f_{15}	**0.00E+00±0.00E+00**≈	1.74E+02±1.04E+01†	**0.00E+00±0.00E+00**
f_{16}	**0.00E+00±0.00E+00**≈	7.67E−01±1.07E−01†	**0.00E+00±0.00E+00**
f_{17}	5.19E+00±4.57E+00†	3.60E−07±3.26E-07†	**0.00E+00±0.00E+00**
f_{18}	1.38E+04±2.59E+04†	1.52E+02±4.52E+02†	**1.75E−01±2.58E−01**
f_{19}	2.10E+01±4.90E−02≈	2.10E+01±4.40E−02≈	2.09E+01±2.77E−01
w/*t*/*l*	12/4/3	15/4/0	–/–/–

表 3 中给出了 6 种对比算法的优化均值误差及标准差,其中,加粗的数值表示该算法在该测试函数上取得

的最优计算结果.从表 3 可知:在相同种群数和评估次数的计算条件下,MIFPA 算法在 19 个测试函数上获得了 12 个最优计算结果,其中,MIFPA 算法在 7 个函数上取得了全局最优解,在 2 个函数上获得了次优结果.表 3 最后一行的 Wilcoxon 秩和检验结果可以直观地显示:在所有对比算法中,MIFPA 算法的优化性能最好且优势非常明显.具体分析如下.

(1) 对于单模态高维的第 1 类函数,MIFPA 算法与对比算法相比显著地提高了解的质量,4 个函数中有 2 个函数取得了全局最优解,其收敛精度高;尤其是对于非凸病态且非常难以找到全局最优值的经典高维单模态测试函数 f_3,除 MIFPA 算法外的其余算法都易收敛到局部极小,求解结果不太理想,而 MIFPA 算法的收敛精度达到 10^{-4},无限接近最优解,其收敛性能远远优于对比算法,尤其是基本 FPA 算法几乎寻优失败.这充分说明改进策略能够有效地提高基本 FPA 算法的优化能力;

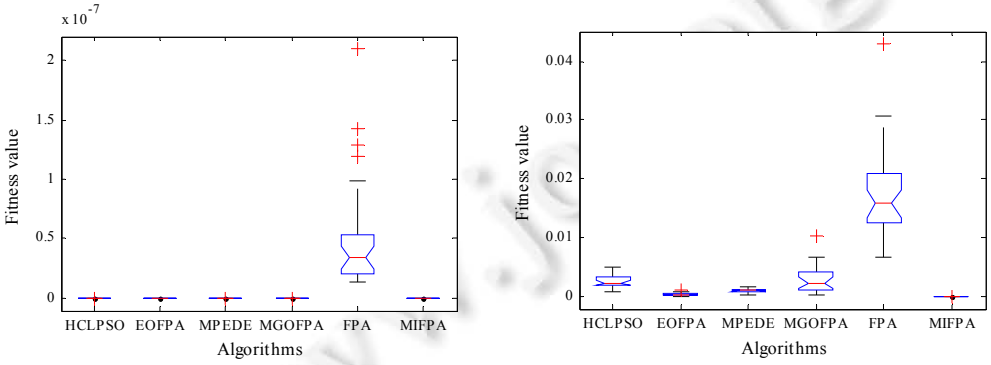
(2) 在具有大量局部极值点的非旋转多模态高维的第 2 类函数上,MIFPA 算法与 HCLPSO、EOFPA、MGOFPA 及 FPA 算法相比,解的质量优于对比算法;与 MPEDE 比较,两者解的质量相当;尤其是对于基本 FPA 算法而言,在此类函数上的优化能力不太理想.这充分表明了本文提出的改进方法能够显著地提高基本 FPA 算法的求解精度;

(3) 对于多模态低维的第 3 类函数,MIFPA 算法与 EOFPA 算法相比显著地提高了解的质量,在 4 个函数上,都优于对比算法;对 HCLPSO、MPEDE 和 FPA 算法而言,只有在函数 f_{10} 上比 HCLPSO 算法略有逊色,剩余函数解的质量要优于或者相当于比较算法;与 MGOFPA 比较,MIFPA 解的质量要稍差一些;

(4) 在第 4 类带旋转的多模态高维函数上,MIFPA 算法与 HCLPSO、FPA 算法相比优势非常明显,解的质量要好于对比算法,尤其是在函数 f_{15} 和 f_{16} 上,MIFPA 算法能够找到理论最优解,而对比算法近乎寻优受挫;与其余 3 种算法对比,MIFPA 算法的优化性能也要优于或相当于对比算法.MIFPA 算法之所以能够在绝大部分函数上提高了解的质量,这是因为新方法能够有效地改进基本 FPA 算法在搜索策略和解的质量方面存在的不足,从而改善算法的全局优化能力;

(5) 在第 5 类变换旋转的高维函数上,在函数 f_{17} 和 f_{18} 上,MIFPA 算法与其他对比算法相比,MIFPA 算法解的质量明显优于或相当于对比算法,尤其是对于函数 f_{18},MIFPA 算法的优势更为突出,它能找到全局最优解,而其他 5 种对比算法无法获得全局最优解;6 种对比算法在优化函数 f_{19} 时,MIFPA 获得解的质量与 MPEDE、MGOFPA 和 FPA 算法相当,但差于其他两种算法.由上述可知:MIFPA 算法与其余算法对比,更适合求解此类问题.

为了更深入、直观地表明 MIFPA 算法的优化性能好于其对比算法,图 4 展示了 6 种算法在部分测试函数上的全局最优值方差分析结果.



(a) f_1 D=30 (b) f_4 D=30

Fig.4 Anova analyses of global minimum for six algorithms on test functions

图 4 6 种算法在测试函数上的全局最优值方差分析

由图 4 可知:MIFPA 算法在两个函数上的收敛精度取得了两个最优结果;只有在函数 f_1 上的收敛精度与基本 EOFPA 算法相当,但 MIFPA 算法在该函数上的求解精度显著地优于其余对比算法.尤其是对于函数 f_4,MIFPA 算法找到的解无限接近于最优解,其解的精度要比其余的 5 种对比算法更优.这表明,MIFPA 算法具有很好的收敛性能,也验证了表 3 中的实验结果.

通过对上述 5 类不同的测试函数进行对比实验,实验结果表明:MIFPA 算法具有很好的收敛能力,显著地提高了 FPA 算法解的质量,不仅适合解决低维问题,而且也适用于优化高维复杂问题.

3.4.2 算法的维度扩展性分析

对于演化算法而言,算法的优化性能会随着问题维度的增加而降低.为了验证 MIFPA 算法在高维上同样具有良好的寻优能力,不会陷入“维灾难”,本节将测试函数的维度由 D=30 扩展到 D=50 和 D=100;所有算法的其他参数设置与上节相同,最大评估次数为 10000×D,在所有函数上每种算法都独立运行 30 次.因为是扩展高维函数的维度,故不对第 3 类函数进行测试.

表 4 列出了 6 种算法在 D=50 时的优化结果,为了客观、公平地对算法性能进行对比,对实验结果从数学角度进行了统计分析.

Table 4 Optimal mean error values and standard deviations of the six algorithms (D=50)

表 4 6 种算法的优化均值误差和标准差(D=50)

测试函数	*Mean_error±Std.Dev*		
	HCLPSO	MPEDE	EOFPA
f_1	7.73E−45±1.30E−44†	1.28E−60±7.01E−60†	**0.00E+00±0.00E+00**≈
f_2	4.80E−02±3.46E−02†	3.75E−14±5.18E−14†	**0.00E+00±0.00E+00**≈
f_3	**1.06E+01±7.02E+00**‡	5.32E−01±1.38E+00‡	3.47E+01±8.15E−01†
f_4	5.80E−03±1.30E−03†	2.30E−03±7.39E−04†	2.92E−04±2.34E−04†
f_5	1.46E−09±5.02E−09†	**0.00E+00±0.00E+00**≈	**0.00E+00±0.00E+00**≈
f_6	4.48E−14±7.83E−15†	7.76E−15±9.01E−16†	**8.88E−16±0.00E+00**≈
f_7	**0.00E+00±0.00E+00**≈	7.40E−04±2.30E−03†	**0.00E+00±0.00E+00**≈
f_8	**9.42E−33±2.78E−48**‡	2.10E−03±1.14E−02†	2.08E−27±4.63E−27†
f_9	**1.35E−32±5.57E−48**‡	**1.35E−32±5.57E−48**‡	3.40E−03±5.10E−03†
f_{14}	1.51E+03±6.68E+02‡	**3.26E−21±1.79E−20**‡	2.55E+03±3.49E+02†
f_{15}	2.30E−02±2.75E−02†	**0.00E+00±0.00E+00**≈	**0.00E+00±0.00E+00**≈
f_{16}	5.53E−14±7.95E−15†	7.11E−15±0.00E+00†	**0.00E+00±0.00E+00**≈
f_{17}	1.67E−13±3.32E−14†	4.36E−14±2.45E−14†	8.93E−11±4.89E−10†
f_{18}	1.45E+01±8.70E+00‡	**9.31E−01±1.72E+00**‡	8.21E+01±8.76E+01†
f_{19}	2.10E+01±7.00E−02≈	2.11E+01±4.56E−02≈	**2.02E+01±1.49E−01**‡
w/t/l	8/2/5	8/3/4	7/7/1

Table 4 Optimal mean error values and standard deviations of the six algorithms (D=50) (Continued)

表 4 6 种算法的优化均值误差和标准差(D=50)(续)

测试函数	*Mean_error±Std.Dev*		
	MGOFPA	FPA	MIFPA
f_1	1.05E−204±0.00E+00†	1.24E+08±8.80E−09†	**0.00E+00±0.00E+00**
f_2	3.46E−62±9.72E−62†	3.02E−01±2.04E−01†	**0.00E+00±0.00E+00**
f_3	4.45E+01±6.96E−01†	5.54E+01±2.42E+01†	1.72E+01±1.99E+00
f_4	4.93E−05±5.44E−05†	3.92E−02±1.27E−02†	**2.62E−06±2.06E−06**
f_5	**0.00E+00±0.00E+00**≈	1.15E+02±1.83E+01†	**0.00E+00±0.00E+00**
f_6	4.44E−15±0.00E+00†	1.57E+00±5.89E−01†	**8.88E−16±0.00E+00**
f_7	**0.00E+00±0.00E+00**≈	9.36E−06±4.49E−05†	**0.00E+00±0.00E+00**
f_8	9.17E−05±6.55E−05†	1.30E−03±5.20E−03†	1.28E−32±1.81E−33
f_9	1.12E−01±1.20E−01†	1.13E−05±1.90E−05‡	1.10E−03±3.40E−03
f_{14}	2.12E+03±2.98E+02‡	3.01E+03±5.36E+02†	2.13E+03±3.44E+02
f_{15}	**0.00E+00±0.00E+00**≈	3.36E+02±1.40E+01†	**0.00E+00±0.00E+00**
f_{16}	3.55E−15±0.00E+00†	9.16E−01±4.17E−02†	**0.00E+00±0.00E+00**
f_{17}	3.19E+01±1.61E+01†	1.29E−07±8.96E−08†	**0.00E+00±0.00E+00**
f_{18}	7.37E+05±5.97E+05†	1.27E+02±9.27E+01†	3.74E+01±2.87E+01
f_{19}	2.11E+01±5.24E−02≈	2.11E+01±3.39E−02≈	2.10E+01±4.07E−01
w/t/l	10/4/1	13/1/1	−/−/−

对表 4 进行分析可知:

(1) 在 15 个高维函数上,MIFPA、HCLPSO、MPEDE、EOFPA、MGOFPA 和 FPA 算法分别取得 9 个、4 个、5 个、8 个、3 个和 0 个最优结果,且 MIFPA 算法有 7 个函数获得全局最优解,2 个函数获得次优解;

(2) 从表 4 最后一行可以直观地看出:在 15 个函数上,MIFPA 算法分别有 8 个、8 个和 13 个函数的实验结果分别优于 HCLPSO、MPEDE 和 FPA 算法,有 2 个、3 个和 1 个函数的实验结果与 HCLPSO、MPEDE 和 FPA 算法相当,有 5 个、4 个和 1 个函数的实验结果逊色于 HCLPSO、MPEDE 和 FPA 算法;MIFPA 算法分别有 7 个、10 个函数的实验结果好于 EOFPA 和 MGOFPA 算法,有 1 个、1 个函数分别稍差于 EOFPA 和 MGOFPA 算法,有 7 个、4 个函数的实验结果与 EOFPA 和 MGOFPA 算法相当.

为了进一步检验 MIFPA 算法在更高维数上也同样具有良好的优化能力,此次实验选择在 100 维上对 MIFPA 算法的优化性能进行验证.由于篇幅所限,此次实验只选择 f_1~f_9 共 9 个函数进行测试,其实验结果见表 5.

Table 5 Optimal mean error values and standard deviations of the six algorithms (D=100)

表 5 6 种算法的优化均值误差和标准差(D=100)

测试函数	*Mean_error±Std.Dev*		
	HCLPSO	MPEDE	EOFPA
f_1	6.34E−36±2.60E−35†	1.40E−78±2.67E−78†	0.00E+00±0.00E+00≈
f_2	1.59E+01±6.43E+00†	7.70E−07±6.44E−07†	**0.00E+00±0.00E+00**≈
f_3	1.85E+01±7.79E+00‡	**6.03E+00±5.63E+00**‡	8.25E+01±7.30E-01†
f_4	1.35E−02±2.60E−03†	5.10E−03±1.30E−03†	9.54E−05±5.60E−05†
f_5	3.32E−02±1.82E−01†	1.66E−01±4.59E−01†	0.00E+00±0.00E+00≈
f_6	1.16E−13±1.14E−14†	3.85E−01±5.20E−01†	8.88E−16±0.00E+00≈
f_7	0.00E+00±0.00E+00≈	8.21E−04±3.20E−03†	0.00E+00±0.00E+00≈
f_8	**4.71E−33±1.39E−48**‡	1.45E−02±2.41E−02†	3.89E−24±1.59E−23†
f_9	**1.35E−32±2.25E−34**‡	1.50E−03±3.80E−03†	5.50E−03±6.90E−03†
w/*t*/*l*	5/1/3	8/0/1	4/5/0

Table 5 Optimal mean error values and standard deviations of the six algorithms (D=100) (Continued)

表 5 6 种算法的优化均值误差和标准差(D=100)(续)

测试函数	*Mean_error±Std.Dev*		
	MGOFPA	FPA	MIFPA
f_1	**0.00E+00±0.00E+00**≈	1.71E−10±9.77E−11†	**0.00E+00±0.00E+00**
f_2	2.50E−98±1.37E−97†	2.01E+01±7.45E+00†	**0.00E+00±0.00E+00**
f_3	9.44E+01±7.03E−01†	1.62E+02±4.73E+01†	7.47E+01±1.13E+00
f_4	1.59E−05±1.16E−05†	1.48E−01±3.36E−02†	**1.31E−06±1.24E−06**
f_5	0.00E+00±0.00E+00≈	1.93E+02±3.49E+01†	**0.00E+00±0.00E+00**
f_6	4.44E−15±0.00E+00†	5.87E−01±6.68E−01†	**8.88E−16±0.00E+00**
f_7	0.00E+00±0.00E+00≈	2.47E−04±1.40E−03†	**0.00E+00±0.00E+00**
f_8	2.77E-04±2.45E−04†	5.20E−03±1.43E−02†	1.94E−31±9.84E−31
f_9	1.58E+00±1.38E+00†	1.10E−03±3.10E−03‡	4.00E−03±8.90E−03
w/*t*/*l*	6/3/0	8/0/1	–/–/–

从表 5 最后一行可以看出:MIFPA 算法的收敛能力在 100 维上与对比算法相比,其优势同样也很突出.尤其是 MIFPA 算法与 FPA 算法相比,其全局优化能力的优势非常显著,只有在函数 f_9 上稍差于 FPA 算法,而在其余 8 个函数上,无论是优化均值误差还是标准差,都明显好于 FPA 算法,说明 MIFPA 算法获得的解的质量和鲁棒性都优于 FPA 算法.同时从表 4 和表 5 可知:MIPFA 在 100 维上的优化能力与在 50 维上相比较,其优化均值误差和标准差变化较小.这表明,MIFPA 算法不会陷入“维灾难”问题.

因此,在 D=50,100 上,无论是在单模态高维函数上,还是在复杂高维多模态函数上,MIFPA 算法都优于对比算法,并且取得了非常不错的优化效果.说明改进算法在高维函数上同样具有较好的优化能力,与对比算法相比具有良好的竞争力.

3.4.3 鲁棒性和收敛速度对比分析

为了检验 MIFPA 算法的鲁棒性和收敛速度的优越性,对其鲁棒性和收敛速度进行对比分析.在实验中,对所

有测试函数分别设定一个计算精度阈值,若算法的运行评估次数高于 10000×*D* 次时还没有达到设定的收敛精度阈值,则认定本次寻优不成功;其余实验参数设置与第 3.4.1 节相同,在所有函数上每种算法都独立运行 30 次;执行成功率 *SR*=找到精度阈值的次数/总的运行次数,*Mean_FEs* 为平均评估次数,其中,"NA"表示寻优失利,最优结果用加粗凸显.实验结果见表 6.

Table 6 Algorithms' results of mean number of *FEs* and success rate of optimization under predefined precision

表 6 在固定精度下算法的函数评估次数平均值及寻优成功率

测试函数	阈值	*Mean_Fes* (*SR*%)		
		HCLPSO	MPEDE	EOFPA
f_1	1.00E−08	1.45E+05(**100**)	7.36E+04(**100**)	3.72E+04(**100**)
f_2	2.00E−04	2.81E+05(93.33)	8.48E+04(**100**)	2.02E+04(**100**)
f_3	2.00E+01	2.10E+05(96.67)	**1.01E+05(100)**	1.34E+05(**100**)
f_4	1.00E−01	7.36E+04(**100**)	1.07E+04(**100**)	8.11E+02(**100**)
f_5	1.00E+01	1.32E+05(**100**)	1.10E+05(**100**)	1.56E+03(**100**)
f_6	2.00E−08	1.68E+05(**100**)	1.11E+05(**100**)	1.99E+04(**100**)
f_7	2.00E−03	1.63E+05(83.33)	7.15E+04(90)	3.34E+04(**100**)
f_8	2.00E−02	1.01E+05(**100**)	2.57E+04(**100**)	2.05E+04(96.67)
f_9	1.00E−03	1.16E+05(**100**)	**4.06E+04(100)**	1.65E+05(73.33)
f_{10}	1.00E−04	3.49E+04(83.33)	4.00E+04(3.33)	2.54E+04(53.33)
f_{11}	1.00E−01	1.20E+04(**100**)	1.46E+04(**100**)	**6.65E+03**(90)
f_{12}	4.00E−05	1.63E+04(**100**)	2.08E+04(**100**)	**1.27E+04**(76.67)
f_{13}	9.00E−06	1.73E+04(**100**)	2.34E+04(**100**)	**1.54E+04**(76.67)
f_{14}	6.00E+02	1.83E+05(66.67)	**2.45E+04**(**100**)	2.97E+05(3.33)
f_{15}	2.00E+02	6.66E+02(**100**)	1.50E+03(**100**)	2.74E+02(**100**)
f_{16}	5.00E−01	1.01E+05(**100**)	2.78E+04(**100**)	2.71E+03(**100**)
f_{17}	3.00E-07	8.92E+04(**100**)	**6.64E+04(100)**	7.56E+04(**100**)
f_{18}	3.00E+01	1.46E+05(**100**)	**8.97E+04**(96.67)	1.47E+05(90)
f_{19}	2.10E+01	1.34E+05(**100**)	1.13E+05(90)	**3.70E+04**(**100**)
Total aver	–	1.12E+05(95.96)	5.53E+04(93.68)	5.54E+04 (87.36)
Total rank	–	5(2)	2(3)	3(4)

Table 6 Algorithms' results of mean number of *FEs* and success rate of optimization under predefined precision (Continued)

表 6 在固定精度下算法的函数评估次数平均值及寻优成功率(续)

测试函数	阈值	*Mean_Fes* (*SR*%)		
		MGOFPA	FPA	MIFPA
f_1	1.00E−08	3.00E+04**(100)**	NA	**1.53E+03(100)**
f_2	2.00E−04	5.33E+04(**100**)	2.90E+05(60)	**9.16E+02(100)**
f_3	2.00E+01	NA	2.66E+05(40)	1.13E+05(**100**)
f_4	1.00E−01	4.67E+03(**100**)	6.27E+04(**100**)	**2.48E+02(100)**
f_5	1.00E+01	5.70E+04**(100)**	NA	**2.17E+02(100)**
f_6	2.00E−08	4.44E+04(**100**)	NA	**2.64E+03(100)**
f_7	2.00E−03	2.07E+04(**100**)	2.24E+05(96.67)	**7.67E+02(100)**
f_8	2.00E−02	1.03E+05(**100**)	2.35E+05(83.33)	**1.71E+04(100)**
f_9	1.00E−03	3.00E+05(3.33)	2.53E+05(**100**)	8.85E+04(**100**)
f_{10}	1.00E−04	3.68E+04(36.67)	5.34E+04(**100**)	**1.78E+04(100)**
f_{11}	1.00E−01	1.98E+04(**100**)	2.19E+04(**100**)	1.29E+04(**100**)
f_{12}	4.00E−05	NA	5.00E+04(**100**)	3.16E+04(**100**)
f_{13}	9.00E−06	NA	6.01E+04(**100**)	3.76E+04(**100**)
f_{14}	6.00E+02	2.54E+05(43.33)	2.99E+05(3.33)	2.04E+05(63.33)
f_{15}	2.00E+02	1.34E+03(**100**)	1.04E+05(**100**)	**1.49E+02(100)**
f_{16}	5.00E−01	1.16E+04(**100**)	NA	**4.17E+02(100)**
f_{17}	3.00E−07	NA	2.93E+05(66.67)	1.49E+05(**100**)
f_{18}	3.00E+01	NA	2.76E+05(46.67)	1.49E+05(**100**)
f_{19}	2.10E+01	1.03E+05(90)	1.36E+05(86.67)	1.64E+05(66.67)
Total aver	–	1.06E+05(61.75)	2.01E+05(62.28)	**5.22E+04(96.32)**
Total rank	–	4(6)	6(5)	**1(1)**

从表 6 可知:

(1) 在第 1 类 4 个单模态多维函数上,对于优化成功率,MIFPA、HCLPSO、MPEDE、EOFPA、MGOFPA

和 FPA 算法分别获得了 4 个、2 个、4 个、4 个、3 个和 1 个最优结果,这表明,MIFPA 算法的鲁棒性强于 HCLPSO、MGOFPA 和 FPA 算法,与 MPEDE、EOFPA 算法相当,但 MIFPA 的平均评估次数优于 MPEDE、EOFPA 算法;对于平均评估次数,MIFPA 算法获得了 3 个最优结果,MPEDE 算法只取得 1 个最优结果,其余算法没有取得最优结果.因此,在第 1 类函数上,本文算法的收敛速度最快,也进一步验证了本文的改进措施能够有效地提高 FPA 算法的收敛速度;

(2) 对于第 2 类高维多模态函数,MIFPA 算法在所有函数上都运行成功,且寻优成功率优于其他对比算法;同时,MIFPA 算法的平均评估次数也好于其他对比算法.因此,本文算法在此类函数上的鲁棒性和搜索速度都有一定的优势;

(3) 对于第 3 类的 4 个低维多峰函数,MIFPA 算法的寻优成功率与基本 FPA 算法相当,但优于其他 4 种对比算法;对于平均评估次数,MIFPA 算法好于 MGOFPA、FPA 算法,与 MPEDE 算法平分秋色,但稍差于其他两种算法.因此,在第 3 类函数上,本文算法的鲁棒性整体要比对比算法更优,且收敛速度要快于 MGOFPA、FPA 算法,与 MPEDE 相当,但要逊色于其他两种算法;

(4) 对于第 4 类带旋转的多模态高维函数,MIFPA 算法的平均评估次数和寻优成功率都优于 EOFPA、MGOFPA 和 FPA 对比算法;与 HCLPSO 和 MPEDE 算法相比,MIFPA 算法在函数 f_{15} 和 f_{16} 上的平均评估次数显著地好于 HCLPSO 和 MPEDE 算法,同时,3 种算法在这两个函数上的寻优成功率相当,对于函数 f_{14},MIFPA 算法的平均评估次数和寻优成功率都稍逊色于 HCLPSO 和 MPEDE 算法.因此,在第 4 类函数上,本文算法的鲁棒性和收敛速度与对比算法相比具有一定的竞争力;

(5) 对于第 5 类变换旋转的高维函数,MIFPA 算法的平均评估次数和寻优成功率都优于 MGOFPA 和 FPA 对比算法;与 EOFPA 算法相比,两者的寻优成功率相当,但本文算法的平均评估次数要差于对比算法;与其余两种算法相比,MIFPA 算法的平均评估次数和寻优成功率都稍逊色于对比算法.

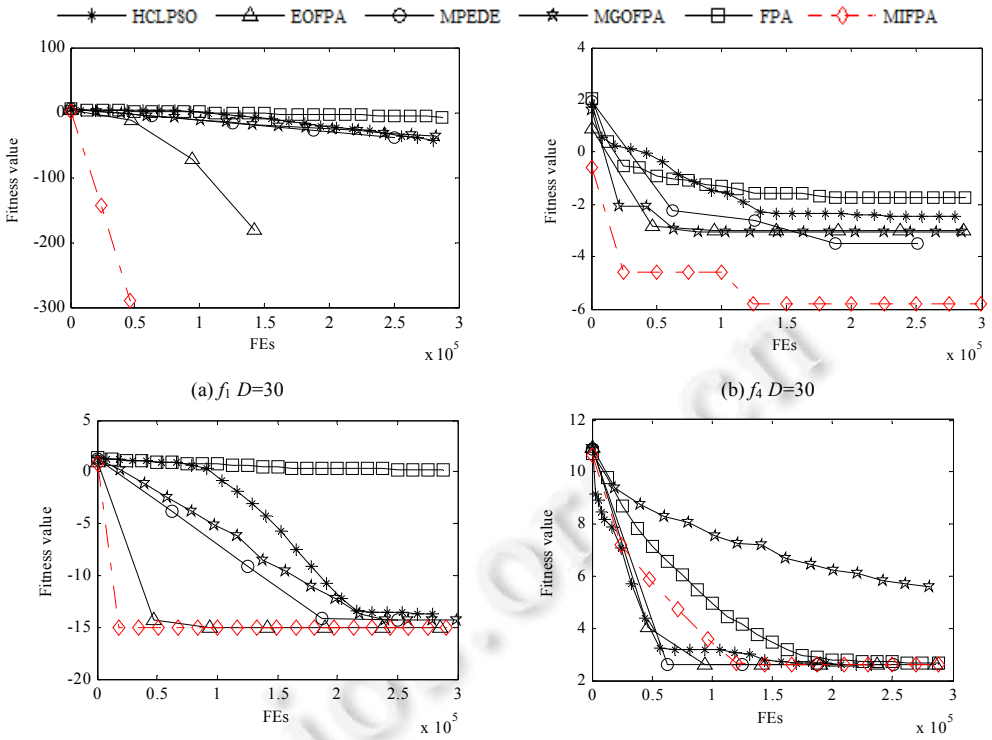
从表 6 倒数第 2 行可以看出:基本 FPA 算法的平均寻优成功率只有 62.28%;但通过对基本 FPA 算法的搜索策略、劣解和转换概率进行改进后,MIFPA 的收敛成功率达到 96.32%,这是基本 FPA 算法成功率的 1.54 倍;与其他 4 种算法相比,寻优成功率至少高出 0.36 个百分点.因此,MIFPA 算法的鲁棒性最强.同时,在总的平均评估次数上,MIFPA 算法的实验结果也是最优的,尤其是对基本 FPA 算法而言,MIFPA 算法总的平均评估次数提高了将近 1 个数量级,这证实了 MIFPA 算法的收敛速度是所有算法中最快的.同时,从表 6 最后一行可知,本文算法的收敛速度和鲁棒性排名第 1,表明其鲁棒性和搜索速度表现突出.

MIFPA 算法之所以能够提高搜索速度,原因是:在算法的演化后期,随着整个种群搜索范围的不断缩小,算法主要侧重于开采;而基本 FPA 算法通过公式(5)随机产生一个新的个体,该策略具有较大的盲目性,产生的新个体有可能处于搜索范围之外,导致无效的搜索次数增加,从而降低了算法的收敛速度.对于 MIFPA 算法而言,在局部搜索部分利用了最优个体搜索策略,使种群中其他个体受最优个体引导作用,快速地向最优解靠近,从而起到了提高算法搜索速度的效果.

为了更深入、直观地显示 MIFPA 算法的稳定性及收敛速度好于其对比算法,图 5 给出了部分函数的收敛曲线图.为了便于分析算法的收敛趋势,这里对测试函数的适应度值取了以 10 为底的对数.

从图 5 可以看出,MIFPA 算法在 4 个函数上的收敛速度都显著地快于基本 FPA 等对比算法;对于函数 f_{18},MIFPA 算法在进化初期搜索速度要慢于 HCLPSO、MPEDE 和 EOFPA 算法,但求精能力要优于 HCLPSO、MPEDE 和 EOFPA 算法,从而考证了表 3 的结果.而与其余算法相比,MIFPA 算法在函数 f_{18} 上的收敛速度明显要快;在另外 3 个函数上,MIFPA 算法的搜索速度都要快于其他对比算法.同时,从图 5 可知,其收敛精度也好于对比算法,因此,进一步佐证了表 3 的实验结果.

图 6 是每种算法分别在两个测试函数上都独立运行 30 次的最优值对比分析图.由图 6 可以看出:MIFPA 算法均未出现波动现象,而其他对比算法都出现了不同程度的波动性.因此,进一步表明 MIFPA 算法的鲁棒性是所有对比算法中最强的,也更深入地验证了本文提出的改进策略是行之有效的.

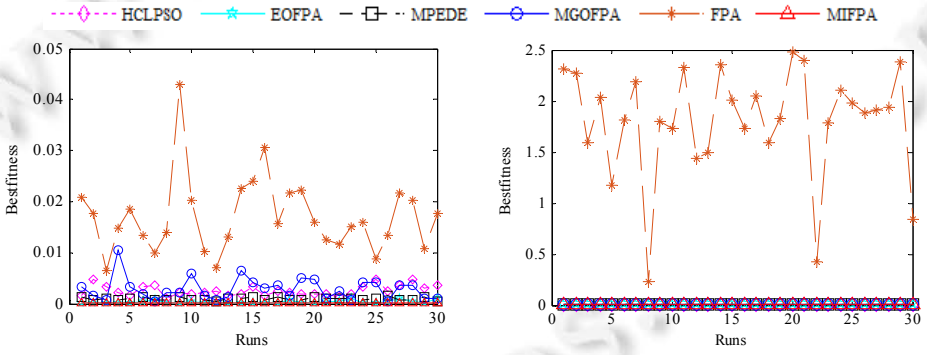


(a) f_1 D=30 (b) f_4 D=30

(c) f_6 D=30 (d) f_{18} D=30

Fig.5 Convergence curve of fitness value for six algorithms on partial functions

图 5 6 种算法在部分函数上的适应度值收敛曲线



(a) f_4 D=30 (b) f_6 D=30

Fig.6 Comparison of optimal fitness value for six algorithms on partial functions

图 6 6 种算法在部分函数上的最优适应度值比较图

3.4.4 Friedman 及 Wilcoxon 检验方法

为了对算法的寻优性能做出更客观的评价,本文运用两种数学统计方法对算法收敛能力进行整体对比分析.表 7 和表 8 是所有算法分别在 19 个函数(维数 D=30 或 4,函数 f_{10}~f_{13} 为低维函数)和 15 个函数(维数 D=50)上的 Friedman 的检验结果.Rankings 的值越小,则算法的优化性能越优,排名越靠前.从表 7 和表 8 可以看出:MIFPA 算法的秩均值(rankings)分别是 2.62 和 2.45,是所有算法中最小的,比 FPA 算法的秩均值分别小 2.52 和 3.02.这表明,MIFPA 算法非常有效地提高了 FPA 算法的性能.对于其他 4 种算法而言,MIFPA 算法的秩均值分别至少要少 0.35 和 0.30.因此,MIFPA 算法无论是在低维还是高维函数上,总体性能都是最好的,与对比算法相比,

MIFPA 算法具有较好的竞争力,这进一步证明了本文提出的改进策略是卓有成效的.

Table 7 Average rankings achieved by Friedman test for the six algorithms (D=30,4)

表 7 6 种算法的 Friedman 秩均值检验(D=30,4)

算法	Rankings
MIFPA	**2.62**
HCLPSO	3.61
MPEDE	2.99
EOFPA	2.97
MGOFPA	3.66
FPA	5.14

Table 8 Average rankings achieved by Friedman test for the six algorithms (D=50)

表 8 6 种算法的 Friedman 秩均值检验(D=50)

算法	Rankings
MIFPA	**2.45**
HCLPSO	3.61
MPEDE	2.78
EOFPA	2.75
MGOFPA	3.94
FPA	5.47

为了判断算法之间的差异性是否显著,本文通过 Wilcoxon 检验对实验结果进行统计分析.表 9 和表 10 分别是 6 种算法在 19 个函数(维数 D=30 或 4,函数 f_{10}~f_{13} 为低维函数)和 15 个函数(维数 D=50)上的 Wilcoxon 检验结果.由表 9 和表 10 可知:在显著性水平 a=0.05 时,虽然 MIFPA 与 MPEDE、EOFPA 算法的显著性差异小于 a,但 MIFPA 算法的秩均值排序要好于 MPEDE、EOFPA 算法;对于其他 3 种算法而言,与 MIFPA 算法的显著性差异较为明显.这进一步证明了 MIFPA 算法的收敛性能优于对比算法,也验证了通过对原有 FPA 算法的搜索策略、转换概率和劣解进行改进,能够有效提高 FPA 算法的寻优能力.

Table 9 Wilcoxon test between MIFPA and other five algorithms (D=30,4)

表 9 MIFPA 与其他 5 种算法的 Wilcoxon 检验(D=30,4)

MIFPA vs.	p-values
HCLPSO	1.30E−18
MPEDE	8.39E−01
EOFPA	1.05E−01
MGOFPA	1.16E−14
FPA	1.57E−68

Table 10 Wilcoxon test between MIFPA and other five algorithms (D=50)

表 10 MIFPA 与其他 5 种算法的 Wilcoxon 检验(D=50)

MIFPA vs.	p-values
HCLPSO	1.14E−15
MPEDE	1.33E−01
EOFPA	1.12E−01
MGOFPA	4.42E−15
FPA	2.47E−52

3.4.5 算法的运行时间比较分析

为了更直观地验证第 2.5 节中的理论分析结果,本节从实验角度对 MIFPA 算法的时间复杂度进行验证,其实验参数的设置与第 3.4.1 节相同.实验结果见表 11,表中 MT 是每种算法在所有测试函数上 CPU 运行时间总的平均值.从表 11 的最后一行可以看出:① FPA 算法的 MT 值比 MIFPA 算法多 1.37s,验证了上述理论剖析的正确性,也说明与 FPA 算法对比,MIFPA 算法性能更好;② MIFPA 算法的 MT 值要比其余对比算法稍大一些,即 CPU 运行的时间要长一些,但仍在可承受的范围内.这是因为,在最大函数评估次数相同的情况下,其他 4 种对比算法的迭代次数要比 MIFPA 算法少.从上述实验结果可以看出,上述的理论分析是正确的.

Table 11 Average CPU run time of six algorithms on functions

表 11 6 种算法在函数上的平均 CPU 运行时间

测试函数	HCLPSO/s	MPEDE/s	EOFPA/s	MGOFPA/s	FPA/s	MIFPA/s
f_1	11.75	4.44	5.24	4.49	9.64	9.38
f_2	22.90	33.10	15.94	15.65	20.27	20.23
f_3	13.05	9.41	6.98	6.30	12.49	9.82
f_4	13.39	7.89	6.85	6.08	11.09	8.93
f_5	14.10	7.44	6.67	7.03	10.66	12.82
f_6	13.12	8.75	7.78	6.33	11.71	12.17
f_7	12.77	7.35	6.52	5.51	11.61	10.45
f_8	16.52	23.08	10.41	9.48	15.27	13.67
f_9	16.39	17.11	10.34	9.80	15.08	13.47
f_{10}	0.50	0.96	8.80	4.90	11.46	7.89
f_{11}	0.74	1.91	10.66	6.78	12.29	9.87
f_{12}	0.83	2.23	11.36	7.56	12.84	10.52
f_{13}	0.99	2.52	12.33	8.38	14.02	11.48
MT	10.54	9.71	9.22	7.56	12.96	11.59

3.5 不同策略的FPA算法优化性能分析

MIFPA 算法对 FPA 算法进行了 4 个方面的改进:在全局授粉部分增加了两组随机个体的差异矢量;通过一个线性递减概率规则,融合两种变异机制对局部授粉部分进行改进;自适应地调整转换概率;引入余弦函数搜索因子对劣解进行改善.为了验证这些改进策略分别对基本 FPA 算法的性能提升的效果,我们分别把这些策略融入到基本 FPA 算法中,并比较这些不同策略对基本 FPA 算法的性能改进效果,从而达到证明这些策略的效用.利用不同方法改进的 FPA 算法如下.

- IGFPA:在基本 FPA 算法的全局搜索部分增加了两组随机个体的差异矢量改进后的算法;
- ILFPA:对基本 FPA 算法的局部搜索部分改进后的算法;
- IPFPA:采用自适应调整转换概率的 FPA 算法;
- CFPA:融入余弦函数搜索因子的 FPA 算法;
- MIFPA:引进上述所有改进策略的 FPA 算法.

在实验中,所有算法都采用第 3.4.1 节相同的参数设置.实验结果见表 12,如果两种算法的优化均值误差相等,则标准差好的算法,其性能更优.

Table 12 Optimal mean error values and standard deviations with different strategies

表 12 不同策略的优化均值误差和标准差

测试函数	评估指标	FPA	IGFPA	ILFPA	IPFPA	CFPA	MIFPA
f_1	*Mean_error*	5.54E−08	5.46E−19	2.59E−11	3.45E−07	0.00E+00	0.00E+00
	Std.Dev	4.01E−08	4.51E−19	3.15E−11	3.41E−07	0.00E+00	0.00E+00
	Rank	4	2	3	5	1	1
f_2	*Mean_error*	2.85E−04	8.24E−05	9.83E−05	1.96E−04	0.00E+00	0.00E+00
	Std.Dev	3.29E−04	5.75E−05	1.19E−04	3.57E−04	0.00E+00	0.00E+00
	Rank	5	2	3	4	1	1
f_3	*Mean_error*	2.08E+01	1.69E+01	2.06E+01	2.04E+01	2.59E+01	7.53E−04
	Std.Dev	4.41E+00	3.83E+00	1.15E+01	3.92E+00	3.04E−01	9.86E−04
	Rank	5	2	4	3	6	1
f_4	*Mean_error*	1.86E−02	1.23E−02	1.25E−02	1.28E−02	4.67E−06	4.87E−06
	Std.Dev	7.00E−03	4.30E−03	5.70E−03	4.00E−03	4.25E−06	4.14E−06
	Rank	6	3	4	5	1	2
f_5	*Mean_error*	6.12E+01	7.15E+01	5.74+01	7.89E+01	0.00E+00	0.00E+00
	Std.Dev	9.70E+00	1.50E+01	8.86E+00	1.21E+01	0.00E+00	0.00E+00
	Rank	3	4	2	5	1	1
f_6	*Mean_error*	1.50E+00	4.55E−05	5.06E−01	1.14E+00	8.88E−16	8.88E−16
	Std.Dev	7.25E−01	9.53E−05	6.77E−01	9.53E−01	0.00E+00	0.00E+00
	Rank	5	2	3	4	1	1
f_7	*Mean_error*	8.49E−05	1.27E−04	2.01E−02	1.80E−03	0.00E+00	0.00E+00
	Std.Dev	1.29E−04	2.51E−04	2.12E−02	2.90E−03	0.00E+00	0.00E+00
	Rank	2	3	5	4	1	1
f_8	*Mean_error*	1.34E−02	2.79E−06	1.05E−07	8.30E−03	6.86E−07	1.69E−32
	Std.Dev	3.74E−02	7.09E−06	5.72E−07	1.68E−02	5.01E−07	7.26E−34
	Rank	6	4	2	5	3	1
f_9	*Mean_error*	2.09E−05	3.73E−13	4.00E−03	4.43E−05	1.76E+00	2.79E−32
	Std.Dev	2.72E−05	5.42E−13	5.40E−03	6.95E−05	1.41E+00	1.20E−32
	Rank	3	2	5	4	6	1
f_{10}	*Mean_error*	1.40E−08	1.40E−08	1.40E−08	1.40E−08	1.40E−08	1.40E−08
	Std.Dev	1.24E−19	4.16E−17	6.36E−20	8.13E−20	4.34E−19	7.22E−20
	Rank	4	6	1	3	5	2
f_{11}	*Mean_error*	3.21E−07	3.21E−07	3.21E−07	3.21E−07	3.21E−07	3.21E−07
	Std.Dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	6.73E−16	0.00E+00
	Rank	1	1	1	1	2	1
f_{12}	*Mean_error*	4.06E−05	4.06E−05	4.06E−05	4.06E−05	4.06E−05	4.06E−05
	Std.Dev	5.42E−16	6.73E−16	0.00E+00	4.51E−16	4.51E−16	4.51E−16
	Rank	3	4	1	2	2	2
f_{13}	*Mean_error*	9.82E−06	9.82E−06	9.82E−06	9.82E−06≈	9.82E−06	9.82E−06
	Std.Dev	0.00E+00	3.24E−16	7.23E−16	0.00E+00	8.54E−16	0.00E+00
	Rank	1	2	3	1	4	1

Table 12 Optimal mean error values and standard deviations with different strategies (Continued)

表 12 不同策略的优化均值误差和标准差(续)

测试函数	评估指标	FPA	IGFPA	ILFPA	IPFPA	CFPA	MIFPA
f_{14}	*Mean_error*	1.17E+03	1.12E+03	8.92E+02	1.12E+03	5.70E+02	5.74E+02
	Std.Dev	2.28E+02	2.71E+02	1.90E+02	2.50E+02	1.01E+02	9.53E+01
	Rank	6	5	3	4	1	2
f_{15}	*Mean_error*	1.74E+02	1.77E+02	1.64E+02	1.75E+02	0.00E+00	0.00E+00
	Std.Dev	1.04E+01	8.28E+00	1.10E+01	1.20E+01	0.00E+00	0.00E+00
	Rank	3	5	2	4	1	1
f_{16}	*Mean_error*	7.67E−01	1.07E−01	8.40E−01	6.74E−01	0.00E+00	0.00E+00
	Std.Dev	1.07E−01	4.13E−02	1.27E−01	1.87E−01	0.00E+00	0.00E+00
	Rank	4	2	5	3	1	1
f_{17}	*Mean_error*	3.60E−07	1.33E−14	3.28E−10	3.18E−06	1.83E−01	0.00E+00
	Std.Dev	3.26E−07	2.44E−14	5.74E−10	2.76E−06	3.13E−01	0.00E+00
	Rank	4	2	3	5	6	1
f_{18}	*Mean_error*	1.52E+02	2.88E+01	1.04E+02	6.24E+01	3.65E+04	1.75E−01
	Std.Dev	4.52E+02	2.33E+01	1.58E+02	5.25E+01	1.32E+05	2.58E−01
	Rank	5	2	4	3	6	1
f_{19}	*Mean_error*	2.10E+01	2.09E+01	2.10E+01	2.10E+01	2.03E+01	2.09E+01
	Std.Dev	4.40E−02	5.72E−02	4.15E−02	3.52E−02	3.04E−01	2.77E−01
	Rank	6	3	5	4	1	2
–	*Average_rank*	4.00	2.95	3.11	3.63	2.63	1.26
	Total_rank	6	3	4	5	2	1

从表 12 中倒数第 2 行可知:基本 FPA 算法取得的 *Average_rank*(排名平均数)值为 4.0,是所有算法中最大的,这说明本文提出的所有改进措施都能改善 FPA 算法的优化性能;在 5 种改进算法中,MIFPA 算法获得的 *Average_rank* 值为 1.26,是 5 种改进算法中最小的.同时,从表 12 最后一行可以看出,MIFPA 算法的 *Total_rank*(表示所有算法性能的排名)的值为 1.这即证明,本文提出的改进策略能够很好地相互协调,并最终提高 FPA 算法的全局优化能力.

4 新算法求解 UCAV 工程优化问题

随着现代战争环境的日趋复杂,如何在现代战争中利用高科技降低战争风险,是现代军事领域面临的一个重要课题.而从近年来的几场典型战役(伊拉克、反恐等)可知,无人机的高效使用是规避现代战争中巨大风险的非常有效的手段之一.因为无人机无需人员驾驶,在现代战争和反恐中,可以通过无人机深入敌方危险区域进行情报收集或者攻击,而无需担忧人员伤亡,降低了战争或反恐的代价.同时,无人机的造价也相对比较低廉,所以无人机在战争环境日益复杂的现代战争和反恐中得到广泛使用.但是随着现代技术的日新月异,威胁无人机的不利因素(导弹、雷达等)日益增多,给无人机的作战、监视、侦查和其他方面的任务带来巨大的挑战.如何获取无人机的最优或次优航线路径,是提高其战斗力的关键问题.

为了进一步验证新算法对复杂的实际工程优化问题的优化能力,本文利用 MIFPA 算法来解决无人作战飞行器(unmanned combat air vehicle,简称 UCAV)的航线规划问题.首先阐述该优化问题的数学模型,再利用本文提出的改进算法求解该优化问题,并进行实验仿真与实验结果分析.

4.1 无人机航线优化问题的数学模型

无人机航线路径规划问题是在满足一定的约束条件下,找到一条从出发点到终点之间的代价最小的最优或者次优的航线路径.最优航线路径是指无人机在避开所有潜在威胁区域且能耗少时所经过的一条从起始点到目的地的最短路径.从上述定义可知,其优化模型的性能评估指标包括威胁代价和能耗代价,故无人机航线路径规划问题的数学模型定义如公式(12):

$$\min J=\vartheta J_t+(1-\vartheta)J_f \tag{12}$$

其中,$\vartheta\in[0,1]$是能耗性能与安全性能之间的平衡系数;J_f 和 J_t 分别表示能耗最小的性能评估指标和威胁最小的性能评估指标,其值分别通过公式(13)和公式(14)获得:

$$\min J_t = \int_0^L w_t \mathrm{d}l \tag{13}$$

$$\min J_f = \int_0^L w_f \mathrm{d}l \tag{14}$$

其中,w_t 和 w_f 分别表示航线路径上每个点的威胁代价和能耗代价,L 表示路径长度.在本文实验中,w_f=1.

当无人机沿着路径 L_{ij} 飞行时,其总的威胁代价通过公式(15)计算取得:

$$w_{t,L_{ij}} = \frac{L_{ij}^5}{5}\sum_{k=1}^{N_t} t_k \left(\frac{1}{d_{0.1,k}^4} + \frac{1}{d_{0.3,k}^4} + \frac{1}{d_{0.5,k}^4} + \frac{1}{d_{0.7,k}^4} + \frac{1}{d_{0.9,k}^4} \right) \tag{15}$$

其中,L_{ij} 表示节点 i 和节点 j 之间的长度,是 L 的子段;N_t 是无人机飞行过程中所要面临的威胁领域的个数;t_k 表示第 k 个威胁领域的威胁度;$d_{0.1,k}$ 表示子段 L_{ij} 上 1/10 处的分割点到第 k 个威胁领域的距离.

4.2 实验结果与分析

4.2.1 测试参数与测试问题

为了验证本文改进算法对 UCAV 航线规划问题的求解是否行之有效,采用表 13、表 14 的两个测试实例 1 和实例 2 进行实验仿真,并与 FPA、EOFPA[16]、MGOFPA[14]、HCLPSO[24]和 MPEDE[25]算法进行实验对比分析.实验参数设置为:种群数和迭代次数分别为 30 和 200,6 种对比算法的其他参数设置同第 3.4.1 节.

Table 13 Details of test case 1
表 13 测试实例 1 的详细信息

威胁源位置	威胁半径	威胁度
(45,50)	10	2
(12,40)	10	10
(32,68)	8	1
(36,26)	12	2
(55,80)	9	3

Table 14 Details of test case 2
表 14 测试实例 2 的详细信息

威胁源位置	威胁半径	威胁度
(59,52)	10	9
(55,80)	9	7
(27,58)	9	3
(24,33)	9	12
(12,48)	12	1
(70,65)	7	5
(70,34)	12	13
(30,70)	10	2

4.2.2 实验结果与分析

为了比较算法的优化能力,采用最优值(*best*)、平均值(*mean*)和最差值(*worst*)这 3 个性能指标对算法的优化能力进行度量.各种问题维度下的实验仿真结果见表 15,最优结果用加粗凸显.

为了减少实验误差,每种算法在每种维数上独立运行 30 次.为了更直观地表达 MIFPA 算法的优化性能,本文给出了 6 种对比算法在测试实例 1 和测试实例 2 上部分不同维度上的最优飞行航线图,分别如图 7、图 8 所示.

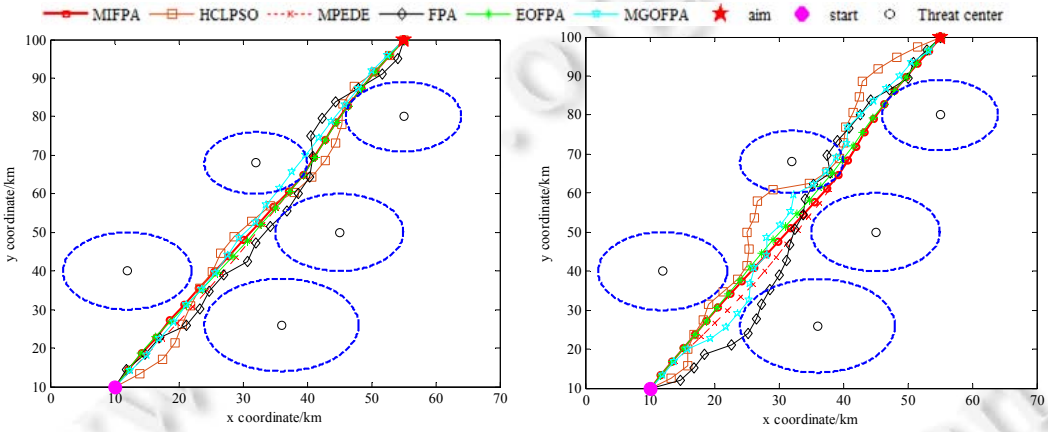
Table 15 Comparison of experimental results of eight algorithms on two test cases
表 15 8 种算法在两个测试实例上的实验结果对比

测试实例	维数	性能指标	MIFPA	HCLPSO	MPEDE	FPA	EOFPA	MGOFPA
测试实例 1	5	*Mean*	**51.349**	51.874	51.438	53.468	51.357	53.825
		best	**51.344**	51.347	51.344	52.998	51.344	51.555
		worst	**51.350**	55.578	54.072	56.095	51.407	57.543
	10	*Mean*	**50.459**	50.633	50.460	52.242	50.466	52.259
		best	**50.459**	50.472	50.459	51.341	50.459	50.925
		worst	**50.460**	53.355	50.477	53.413	50.488	54.873
	15	*Mean*	**50.393**	51.175	50.395	52.551	50.398	52.264
		best	50.389	50.500	**50.388**	51.438	**50.388**	50.755
		worst	**50.400**	57.820	50.420	53.689	50.418	54.881
	20	*Mean*	50.407	52.197	50.411	53.379	**50.319**	52.230
		best	50.381	50.821	**50.373**	51.815	50.377	50.637
		worst	50.561	56.573	50.588	54.769	**50.473**	53.451
	25	*Mean*	50.490	53.845	**50.476**	54.203	50.673	53.362
		best	**50.380**	51.208	50.384	51.593	50.427	50.834
		worst	**50.549**	60.115	50.696	56.765	52.926	57.245

Table 15 Comparison of experimental results of eight algorithms on two test cases (Continued)

表 15 8 种算法在两个测试实例上的实验结果对比(续)

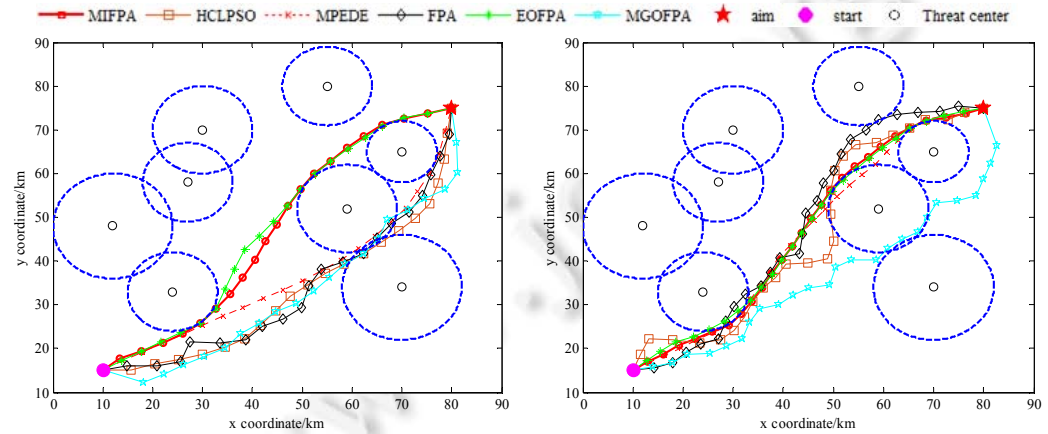
测试实例	维数	性能指标	MIFPA	HCLPSO	MPEDE	FPA	EOFPA	MGOFPA
测试实例 2	5	*Mean*	**50.193**	52.107	51.645	54.914	51.612	58.625
		best	**50.191**	50.198	**50.191**	51.715	**50.191**	51.004
		worst	50.236	61.136	61.092	62.073	77.094	62.388
	10	*Mean*	**50.870**	55.247	52.904	54.012	55.960	59.074
		best	48.103	48.433	**48.093**	49.014	48.126	50.016
		worst	**56.341**	63.249	56.746	58.689	60.581	65.031
	15	*Mean*	**49.466**	51.170	49.888	52.608	50.005	54.046
		best	47.877	48.047	47.862	49.652	**47.847**	51.646
		worst	**50.437**	54.608	50.822	54.317	50.835	56.382
	20	*Mean*	**48.634**	53.305	48.879	52.520	48.863	54.661
		best	47.869	48.946	**47.827**	50.869	47.902	52.286
		worst	49.455	66.527	49.619	54.997	49.608	58.528
	25	*Mean*	**48.575**	54.384	48.956	53.694	48.968	57.497
		best	**47.924**	50.058	47.926	51.538	48.195	53.854
		worst	**49.109**	62.888	50.171	57.937	50.242	61.758



(a) D=20 (b) D=25

Fig.7 Optimal flight route charts of six algorithms D=20 and D=25 on example 1

图 7 6 种算法在实例 1 上 D=20 和 D=25 的最优飞行航线图



(a) D=20 (b) D=25

Fig.8 Optimal flight route charts of six algorithms D=20 and D=25 on example 2

图 8 6 种算法在实例 2 上 D=20 和 D=25 的最优飞行航线图

从图 7 可知:

- 在 D=20 上,除 FPA 和 MGOFPA 算法不能解决 UCAV 问题外,其余算法都能较好地解决 UCAV 航线规划问题,可以使无人机有效地避开危险区域;
- 对于 D=25 而言,MIFPA 和 MPEDE 算法可获得良好的寻优结果,能够使无人机有效地避开危险区域;而其余对比算法都不能使无人机有效地避开威胁区域,寻优失败.

从图 8 可以看出:MIFPA 算法都能较好地解决 UCAV 航线规划问题,可以使无人机有效地避开威胁区域.且从表 15 可知,MIFPA 算法的寻优性能都要好于其余对比算法.

从表 15 可以看出:

(1) 在测试实例 1 上:当 D=5,10,15 时,MIFPA 算法的优化能力显著优于 5 种对比算法;当 D=20 时,MIFPA 算法的优化性能要逊色于 EOFPA 算法,但从图 7 可以看出,MIFPA 算法也可以较好地使无人机有效地避开危险区域,MIFPA 算法的寻优性能要优于其余 4 种对比算法;当 D=25 时,MIFPA 算法的寻优能力要稍差于 MPEDE 算法,但从图 7 可以看出,MIFPA 算法也可以较好地使无人机有效地避开危险区域,MIFPA 算法的寻优性能要优于其余 4 种对比算法.这表明,本文算法在解决 UCAV 航线规划问题时总体上要优于对比算法;

(2) 在测试实例 2 上:MIFPA 算法的优化能力表现得更为突出,MIFPA 算法的寻优能力都要优于对比算法.这说明,MIFPA 算法解决 UCAV 问题的能力更强.

综上所述,本文提出的新算法在求解 UCAV 工程优化问题时,其优化能力同样具有很好的竞争力.

5 结束语

一方面,花授粉算法是一种新型元启发式优化算法,由于该算法实现过程简单且具有较好的全局探索和局部开采平衡能力等优点,使其在函数优化与工程优化等领域取得了成功应用.但又因其搜索方程、进化策略和参数设置方面存在一些不足,导致其收敛精度低、收敛速度慢和易早熟等问题,进而造成其应用范围受到一定程度的限制.另一方面,当前已有的研究没有考虑 FPA 算法的搜索方程存在的不足对算法性能的影响,也没有考虑 FPA 算法采用贪婪式进化机制中的劣解对算法性能的影响,以及转换概率 p 采用固定值对算法性能的影响,同时,也没有对全局搜索方程中运用最优个体策略对算法性能带来的问题进行探索,并且现有的改进 FPA 算法寻优能力还有待提升,其时间复杂度也较高.

本文对基本 FPA 算法的搜索策略进行了定性分析,发现其存在着多方面的不足,制约着该算法的全局优化能力.为了提高算法的收敛能力,本文提出了一种改进的基于多策略的花授粉算法 MIFPA:首先,在全局搜索方程中加入两组差异矢量来增加种群多样性,有利于防止算法早熟,进而提高算法的寻优性能;其次,将精英变异策略和线性递减概率规则引入到局部搜索方程,与原有的搜索方程构建一种新的复合型局部搜索方程,以有效提升算法的收敛能力;最后,利用余弦函数搜索因子对进化中产生的劣解进行改进,从而达到改善解的质量、提高算法优化能力的目的.

为了检验新算法的优越性和可行性,首先,在经典测试函数上进行测试,实验结果显示,MIFPA 算法与知名的 HCLPSO 和 MPEDE 算法、典型的 EOFPA 和 MGOFPA 算法以及基本的 FPA 算法相比,其解的质量、鲁棒性和收敛速度、时间复杂度等方面总体上都要优于对比算法;其次,利用新算法对 UCAV 问题进行求解,实验结果表明,新算法在求解复杂的实际工程优化问题时,其优化能力表现更优.

实验结果表明:本文提出的 MIFPA 算法与对比算法相比,其全局优化能力获得较大的提高;但从表 6 中的函数 f_{14} 和 f_{19} 的优化结果可以看出,其鲁棒性还具有提升的空间.这是因为,本文的参数 p 虽然采用了自适应调整策略,但算法在寻优过程中还是依据 p 的值与一个大于 0 的随机小数进行比较,以随机选择全局搜索或者局部搜索,这导致算法在寻优过程中具有振荡性,从而影响了算法的鲁棒性.因此,在今后的工作中,如何消除参数 p 对算法寻优能力的影响,以及运用新算法来解决更多的实际工程优化问题,推广其应用领域,是我们今后所要研究的内容.

References:

[1] Kennedy J, Eberhart R. Particle swarm optimization. In: Proc. of the IEEE Int'l Conf. on Neural Networks. Piscataway: IEEE, 1995. 942−948. [doi: 10.1109/ICNN.1995.488968]

[2] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization, 2007,39:459−471. [doi: 10.1007/s10898-007-9149-x]

[3] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997,11:341−359. [doi: 10.1023/A:1008202821328]

[4] Rao RV, Savsani VJ, Vakharia DP. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. Computer-aided Design, 2011,43(3):303−315. [doi: 10.1016/j.cad.2010.12.015]

[5] Yang XS. Flower pollination algorithm for global optimization. In: Proc. of the 11th Int'l Conf. on Unconventional Computation and Natural Computation. LNCS 7445, 2012. 240−249. [doi: 10.1007/978-3-642-32894-7_27]

[6] Yang XS, Karamanoglu M, He XS. Multi-objective flower algorithm for optimization. Procedia Computer Science, 2013,18: 861−868. [doi: 10.1016/j.procs.2013.05.251]

[7] Sharawi M, Emary E, Saroit IA, El-Mahdy H. Flower pollination optimization algorithm for wireless sensor network lifetime global optimization. Int'l Journal of Soft Computing and Engineering, 2014,4(3):54−59.

[8] Chiroma H, Khan A, Abubakar AI, Saadi Y, Hamza MF, Shuib L, Gital AY, Herawan T. A new approach for forecasting OPEC petroleum consumption based on neural network train by using flower pollination algorithm. Applied Soft Computing, 2016,48: 50−58. [doi: 10.1016/j.asoc.2016.06.038]

[9] Bekdaş G, Nigdeli SM, Yang XS. Sizing optimization of truss structures using flower pollination algorithm. Applied Soft Computing, 2015,37:322−331. [doi: 10.1016/j.asoc.2015.08.037]

[10] Majidpour H, Gharehchopogh FS. An improved flower pollination algorithm with Adaboost algorithm for feature selection in text documents classification. Journal of Advances in Computer Research, 2018,9(1):29−40.

[11] Peesapati R, Yadav VK, Kumar N. Flower pollination algorithm based multi-objective congestion management considering optimal capacities of distributed generations. Energy, 2018,147:980−994. [doi: 10.1016/j.energy.2018.01.077]

[12] Łukasik S, Kowalski PA. Study of flower pollination algorithm for continuous optimization. Advances in Intelligent Systems and Computing, 2015,322:451−459. [doi: 10.1007/978-3-319-11313-5_40]

[13] Madasu SD, Kumar MLSS, Singh AK. A flower pollination algorithm based automatic generation control of interconnected power system. Ain Shams Engineering Journal, 2018,9(4):1215−1224. [doi: 10.1016/j.asej.2016.06.003]

[14] Draa A. On the performances of the flower pollination algorithm—Qualitative and quantitative analyses. Applied Soft Computing, 2015,34:349−371. [doi: 10.1016/j.asoc.2015.05.015]

[15] Mahdad B, Srairi K. Security constrained optimal power flow solution using new adaptive partitioning flower pollination algorithm. Applied Soft Computing, 2016,46:501−522. [doi: 10.1016/j.asoc.2016.05.027]

[16] Zhou YQ, Wang R, Luo QF. Elite opposition-based flower pollination algorithm. Neurocomputing, 2015,188:294−310. [doi: 10.1016/j.neucom.2015.01.110]

[17] Xiao HH, Wan CX, Duan YM. Flower pollination algorithm based on complex method. Journal of Chinese Computer Systems, 2015,36(6):1373−1378 (in Chinese with English abstract).

[18] Chakraborty D, Saha S, Dutta O. DE-FPA: A hybrid differential evolution-flower pollination algorithm for function minimization. In: Proc. of the Int'l Conf. on High Performance Computing and Applications. Bhubaneswar, 2014. 1−6. [doi: 10.1109/ICHPCA.2014.7045350]

[19] Wang R, Zhou YQ, Qiao SL, Huang K. Flower pollination algorithm with bee pollinator for cluster analysis. Information Processing Letters, 2016,116:1−14. [doi: 10.1016/j.ipl.2015.08.007]

[20] Xiao HH, Wan CX, Duan YM, Tan QL. Flower pollination algorithm based on gravity search mechanism. Acta Automatica Sinica, 2017,43(4):576−594 (in Chinese with English abstract). [doi: 10.16383/j.aas.2017.c160146]

[21] He XS, Yang XS, Karamanoglu M, Zhao YX. Global convergence analysis of the flower pollination algorithm: A discrete-time markov chain approach. Procedia Computer Science, 2017,108:1354−1364. [doi: 10.1016/j.procs.2017.05.020]

[22] Nasir M, Das S, Maity D, Sengupta S, Halder U, Suganthan PN. A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. Information Sciences, 2012,209:16−36. [doi: 10.1016/j.ins.2012.04.028]

[23] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation, 2009,214(1): 108−132. [doi: 10.1016/j.amc.2009.03.090]

[24] Lynn N, Suganthan PN. Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. Swarm & Evolutionary Computation, 2015,24:11−24. [doi: 10.1016/j.swevo.2015.05.002]

[25] Wu GH, Mallipeddi R, Suganthan PN, Wang R, Chen HK. Differential evolution with multi-population based ensemble of mutation strategies. Information Sciences, 2016,329:329−345. [doi: 10.1016/j.ins.2015.09.009]

[26] Nabil E. A modified flower pollination algorithm for global optimization. Expert Systems with Applications, 2016,57:192−203. [doi: 10.1016/j.eswa.2016.03.047]

[27] Ouadfel S, Taleb-Ahmed A. Social spiders optimization and flower pollination algorithm for multilevel image thresholding: A performance study. Expert Systems with Applications, 2016,55:566−584. [doi: 10.1016/j.eswa.2016.02.024]

[28] Sayed SA, Nabil E, Badr A. A binary clonal flower pollination algorithm for feature selection. Pattern Recognition Letters, 2016,77: 21−27. [doi: 10.1016/j.patrec.2016.03.014]

[29] Dubey HM, Pandit M, Panigrahi BK. A biologically inspired modified flower pollination algorithm for solving economic dispatch problems in modern power systems. Cognitive Computation, 2015,7(5):594−608. [doi: 10.1007/s12559-015-9324-1]

附中文参考文献:

[17] 肖辉辉,万常选,段艳明.一种基于复合形法的花朵授粉算法.小型微型计算机系统,2015,36(6):1373−1378.

[20] 肖辉辉,万常选,段艳明,谭黔林.基于引力搜索机制的花朵授粉算法.自动化学报,2017,43(4):576−594. [doi: 10.16383/j.aas.2017.c160146]

肖辉辉(1977－),男,博士,教授,主要研究领域为智能计算及其应用,情感计算.

万常选(1962－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为数据挖掘与知识工程,情感分析,Web 数据管理与信息检索.