doi:10.3969/j.issn.1001-4616.2020.04.017

动态环境下的自适应反向扩散演化算法

曹文梁[1],康岚兰[2],王　石[1]

(1.东莞职业技术学院计算机工程系,广东 东莞 523808)
(2.江西理工大学应用科学学院,江西 赣州 341000)

[摘要]　针对传统演化优化算法难以在动态环境下有效地持续跟踪最优解的问题,本文提出了一种自适应反向扩散演化算法(ARDEA)对动态环境进行寻优. 该算法采用多种群策略对最优解进行跟踪,通过设置子种群全局动态监哨点监测环境变化;并引入一种差分粒子群速度更新公式引导个体在搜索空间内不断寻找最优值;同时,本文提出了一种新的排斥策略以确保种群多样性,以及种群持续跟踪最优解的搜索能力. 该策略包含两个新方法:其一,采用新提出的群间平均马氏距离判断种群间距离,对于群间距过小的两个子种群进一步通过 hill-valley 函数判定它们的搜索空间是否重叠,其二,将重叠搜索空间中的劣势子种群通过反向扩散操作(RD)重新初始化. 新算法与当前性能较优的动态优化算法同时作用于移动峰测试问题,结果表明,ARDEA 算法在动态环境中能更加有效地跟踪最优解,与其它比较算法而言,表现出较强的鲁棒性和适应性.

[关键词]　动态优化,粒子群优化,反向扩散,群间平均马氏距离
[中图分类号]TP181　[文献标志码]A　[文章编号]1001-4616(2020)04-0119-10

An Adaptively Reversed Diffuse Evolutionary Algorithm in Dynamic Environments

Cao Wenliang[1], Kang Lanlan[2], Wang Shi[1]

(1.Department of Computer Engineering, Dongguan Polytechnic, Dongguan 523808, China)
(2.College of Applied Science, Jiangxi University of Science and Technology, Ganzhou 341000, China)

Abstract: To solve the problem that traditional evolutionary optimization algorithm is difficult to effectively keep track of the optimal solution in dynamic environment, this paper proposes an adaptively reversed diffuse evolutionary algorithm (ARDEA). The new algorithm adopts the multi-population strategy to track the optimal solution and monitors the environmental changes by setting the global dynamic sentryin each subgroup. A differential particle swarm velocity update formula is introduced to guide individuals to search for the optimal points in the search space. Meanwhile, in order to ensure the diversity of the population and the search efficiency of the sub-population, a new exclusion strategy is proposed in this paper. This strategy includes two method. Firstly, it uses between-swarms average Mahalanobis distance to judge the inter-population distance. If the distance is too small between two sub-populations, hill-valley function is used to further determine whether they tracked the same peak or not. Secondly, the subpopulations with poor performance in the search overlap will be reinitialized by reverse diffusion operation (RD). The new algorithm is compared with several state-of-artdynamic optimization algorithms on moving peak problem. The results show that the ARDEA algorithm can track the optimal solution more effectively in the dynamic environment. Compared with other algorithms, the ARDEA algorithm shows strong robustness and adaptability.

Key words: dynamic optimization, particle swarm optimization, reversed diffuse, between-swarms average Mahalanobis distance

在复杂的现实世界中,很多优化问题都是动态的,例如:动态车辆路线问题、微网经济调度、网络边缘环境中的资源分配和选矿操作过程优化问题等[1-4]. 这类问题的最优值大小及其位置都可能随着时间的变化而改变,它们被称之为动态优化问题(dynamic optimization problems, DOPs)[5],其定义如下:

收稿日期:2020-07-08.
基金项目:广东省普通高校特色创新(自然科学)项目(2019GKTSCX142、2017GKTSCX101)、东莞职业技术学院示范建设专项资金项目(政 201803)、江西省科技厅自然科学基金面上项目(20202BABL202032)、江西省教育厅科技项目(GJJ181511).
通讯作者:康岚兰,博士,讲师,研究方向:演化计算、机器学习. E-mail:victorykll@163.com

$$F(\boldsymbol{x}) = \max(\text{or } \min) f(\boldsymbol{x}, t),$$
$$\text{s.t.} \begin{cases} x \in X(t), \\ X \subseteq S, \quad t \in T. \end{cases} \tag{1}$$

其中,S 是搜索空间,t 是时间,f 是目标函数,x 是可行解,$X(t)$ 是在 t 时刻可行解集合. 根据问题特性,解决 DOPs 的两个关键点在于:

(1)如何快速感知当前环境变化,并对变化后的新环境做出响应;

(2)如何持续跟踪最优解,不断适应新环境.

以上是传统优化算法在解决动态优化问题时面临的新挑战.

DOPs 可认作是一个动态系统控制问题,它通过环境中各个因素的相互协作与外界不断发生能量交换. 而作为一种源于自然进化的随机启发性算法,演化算法(evolutionary algorithm,EA)通过个体间的相互协作,不断与外界交换信息,从而跟踪问题的最优解. 因此,EA 具备求解动态优化问题的天然优势,作为是进化计算领域的一个研究热点问题[6-7],目前,EA 中的粒子群优化算法(particle swarm optimization,PSO)[8-9]被广泛应用于求解 DOPs 的一种演化算法. 然而,为保持动态环境中种群多样性,以及环境动态变化下粒子寻优动力的持续性问题,需要对一般 PSO 算法展开深入研究和改进,以达到在动态环境中更加有效的持续跟踪最优解的目标.

本文通过认真分析 DOPs 的特点,提出了一种求解该问题的新算法:自适应反向扩散演化算法(adaptively reversed diffuse evolutionary algorithm,ARDEA). 该算法通过设立动态监哨点(sentry)监测环境变化;采用多种群策略对最优解进行跟踪;在每个子种群中,受差分变异思想启发,提出一种新的粒子群速度更新公式对个体进行不断进化;同时,为保证种群多样性,以及子种群搜索效率,本文提出了一种新的排斥策略,将搜索重叠或表现较差的子种群通过反向扩散操作(reversed diffuse,RD)重新初始化. 本文将新算法与一些目前性能较优的动态优化算法同时作用于标准动态测试函数-移动峰测试问题(moving peaks benchmark,MPB)[10]中,实验结果表明,ARDEA 算法在动态环境中能有效地跟踪最优解,与其它比较算法而言,表现出较强的鲁棒性和适应性.

1 相关工作

1.1 动态优化算法

用于求解动态优化问题(DOPs)的算法被称之为动态优化算法[5]. 相较于静态环境中,EA 通过一段时间的运行将收敛(非早熟)到一个固定的解或有限区域的不同,在动态环境中,这种收敛将使得问题在面临新环境的到来时失去探索新区域所必要的多样性,从而无法在新环境下再跟踪到问题的最优解. 因此,在求解这类动态优化问题时,优化算法的目标不再是仅仅为了获得一个满意解,而是在不断变化的环境中具备较强的适应能力,从而尽可能地追踪到最优解的变化轨迹.

PSO 算法是目前用于解决 DOPs 问题较为广泛的演化算法之一[7,11-14]. 文献[7]提出了一种基于多种群和时间关联的粒子群混合算法对动态环境下最优解进行跟踪. 该算法利用多个种群下的粒子子种群收集当前环境信息,并根据环境信息计算环境变化后更换解决方案的置换成本,最终获得新环境下最优解. Cao 等在文献[11]中,将基于邻域的学习策略融入到粒子群优化的速度更新中,采用“最坏替换”策略对群进行更新. Li 等[12]通过对动态多峰问题的分析,提出了一个聚类粒子群动态优化算法(CPSO),CPSO 采用了一种邻学习策略训练粒子,并使分层聚类方法分别跟踪多个峰的最优值. 之后,Li 等[13]对 CPSO 算法进行了改进,利用分层聚类方法生成各个子种群,聚类方法中的具体操作做出一系列调整,以适应多峰问题. 另外,卜晨阳[14]在他的博士论文中针对动态优化问题中的时间关联特征、动态处理机制等展开研究,并针对短期水火电高度问题与梯度算法相结合,提出了一种新的混合 PSO 优化算法(HPSO),并在测试实验中取得良好的效果.

差分进化算法(differential evolution algorithm,DE)是另一个普遍用于动态优化问题中的演化算法[15-18]. 例如:文献[17]中,Mendes 等提出了一种多种群差分演化算法用于求解 DOPs,算法中每个子种群的个体通过 DE 操作进行迭代. Brest 等[18]在 2009 年提出了一个自适应差分演化算法,算法应用一种自适应方法对 DE 算法中参数进行控制,并结合多种群与老化机制对动态环境下 DE 算法进行改进. Hui

等[16]结合贪婪的锦标赛全局搜索方法的自适应变异策略,以及自适应存档方法跟踪动态最优解,对动态环境下的多种群 DE 算法进行改进.

上述方法通过采用多种群、混合算法、多策略融合角度提高算法在动态环境下的应激能力,然而种群在动态环境下对最优解的持续跟踪能力仍有许多提升空间.

1.2 PSO 算法

粒子群优化算法(PSO)是一种基于群体智能的全局随机搜索算法,由 Kennedy 等[19]于 1995 年提出. 在进化算法家族中,因其综合性能表现突出而受到广泛重视. 在 PSO 算法中,每个个体(粒子)代表一个可行解,每个粒子具有速度 $v_{i,j}$ 与位置 $x_{i,j}$ 两个属性(分别由式(1)、(2)表示),粒子通过在搜索空间内的移动来寻找最优解. 其中,速度公式由基于自身运动经验的惯性运动(inertial motion)、基于个体历史经验积累的"认知学习(cognitive learning)"以及对整体种群历史经验认识的"社会学习(social learning)"3 部分构成. 其进化更新行为由式(1)、(2)所描述[20]:

$$v_{i,j}(t+1)=\omega v_{i,j}(t)+c_1 rand_1(pbest_{i,j}-x_{i,j}(t))+ c_2 rand_2(gbest_j-x_{i,j}(t)), \tag{1}$$

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1), \tag{2}$$

其中,$i=1,2,\cdots,N$;$j=1,2,\cdots,D$(N 为种群规模,D 粒子维数);ω 是惯性权值(inertia weight),$\omega\in[0,1]$;c_1 和 c_2 为认知学习因子(cognitive learning factor)和社会学习因子(social learning factor),$c_1,c_2\in[0,2]$;$rand_1$ 和 $rand_2$ 是在(0,1)区间上服从均匀分布的两个随机数,$pbest_i$,$gbest$ 分别为第 i 个粒子的当前个人历史最优位和当前全局最优位.

1.3 标准 DE 算法

标准差分算法(DE)由 Storn 等于 1997 年提出,是一种基于群体差异的智能进化方法,具有原理简单、受控参数少且易实现等优点[21]. 其基本思想是对种群中的每个个体 $\boldsymbol{x}_i$,从当前种群中随机选择 3 个互不相同的个体 $\boldsymbol{x}_{r_1},\boldsymbol{x}_{r_2},\boldsymbol{x}_{r_3}$,以其中一个点为基础、另两个点为参照做一个扰动,所得点与个体交叉后进行自然选择,保留较优者,实现种群的进化. 例如,式(3)和(4)就是两种常用的差分变异操作(在差分进化中,$\boldsymbol{v}_i$ 表示个体 $\boldsymbol{x}_i$ 变异后的个体向量):

DE/rand/1:

$$\boldsymbol{v}_i=\boldsymbol{x}_{r1}+F_1\cdot(\boldsymbol{x}_{r2}-\boldsymbol{x}_{r3}) \tag{3}$$

DE/current-to-best/1:

$$\boldsymbol{v}_i=\boldsymbol{x}_i+F_1\cdot(\boldsymbol{x}_{\text{best}}-\boldsymbol{x}_i)+F_2\cdot(\boldsymbol{x}_{r1}-\boldsymbol{x}_{r2}) \tag{4}$$

其中,$i,r_1,r_2,r_3\in\{1,\cdots,N\}$ 为互不相同的索引,r_1,r_2,r_3 被随机选取,F_1,F_2 为缩放因子,x_{best} 为粒子当前全局最优位.

2 自适应反向扩散演化算法

在动态环境中,对环境变化的感知以及变化后对目标解的持续跟踪能力是求解动态优化问题的关键. 为充分适应动态变化的环境,并更加及时有效地获取信息,从而进一步提高动态优化算法性能,本文提出了一种新的自适应扩散演化算法(ARDEA)对动态环境下的最优解持续寻优. 该算法:(1)受 DE 算法启发,对 PSO 中速度更新公式进行改进;(2)同时采用多种群策略,并提出一种新的排斥策略确保种群多样性;(3)新排斥策略采用新提出的一种"群间平均马氏距离"度量种群间离,对于距离小于阈值 δ 的子种群,进一步通过 hill-valley 函数判定将其中处于劣势子种群移出原搜索区域;(4)对劣势子种群采用新提出的反向扩散操作按比例生成新的子种群参与新环境下的寻优工作.

2.1 速度更新策略

在动态环境中,如何准确快速地捕获环境信息,并指导个体下一步的进化是解决动态优化问题的一个关键问题. 因此,本文针对动态优化问题,引入了一个更加适合于动态环境下的速度更新公式. 由于新公式是受到 DE 变异公式启发,对 PSO 中式(1)做出的修正,因此本文称之为 VD 速度更新公式[20].

$$vd_{i,j}(t+1)=s\cdot(x_{r1,j}(t)-x_{r2,j}(t))+\varphi_2(gbest_j-x_{i,j}(t))+\varphi_1(pbest_{i,j}-x_{i,j}(t)). \tag{5}$$

其中,s 被称之为差分系数(differential coefficient),其数值大小表示粒子向其它个体学习的接受程

度. φ_1 和 φ_2 为两个控制参数,用于控制粒子向群体学习及个体历史信息学习的程度.

与 PSO 中速度更新公式相比,式(10)不再包含惯性部分,取而代之的是利用群体中个体之间的差异信息来引导粒子下一时刻的飞行方向. 公式分解如图 1 所示. 由图 1 可见,VD 将更多地从其它个体以及群体中("Momentum+Social" component)获取信息,而不仅仅依赖于个体历史经验("Inertial+Cognition" component),从而积极拓宽信息获取渠道,加强对搜索空间的探索能力. 新策略的设计更加符合人类对新环境认知与适应的过程,有效减少进化过程中粒子间的耦合度,平衡粒子对搜索领域的勘探与开采能力;避免由于个体经验的不足而导致陷入局部最优等问题,同时提升收敛速度. 设计本身更加适用于动态环境中寻优问题.

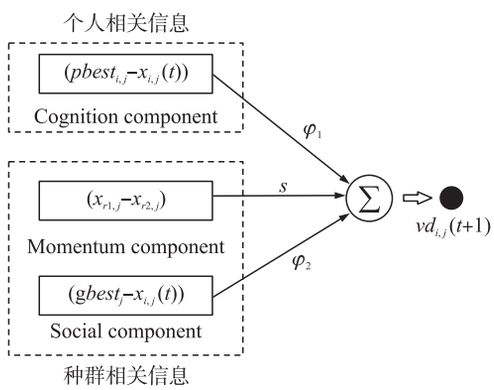

图 1　VD 分解图

Fig. 1　VD decomposition diagram

为防止群体出现搜索停滞现象,算法在每次迭代过程中,对子群中的当前最优解采用差分演化变异(differential evolutionary mutation,DEM)[22]对 *gbest* 进行扰动.

$$m_j = gbest_j + (x_{r1,j} - x_{r2,j}) + (x_{r3,j} - x_{r4,j}), \tag{6}$$

$$gbest_j^* = \begin{cases} m_j, & \text{if } rand \leqslant 0.1 \text{ or } j_{\text{rand}} = j, \\ gbest_j, & \text{otherwise.} \end{cases} \tag{7}$$

其中,$rand$ 是均匀分布的随机数,$rand \in [0,1]$, j_{rand}是一随机整数, $j_{\text{rand}} \in [1,D]$.

2.2　排斥策略

多种群策略已被证明是解决动态多峰问题的一种有效手段[1,4,7],它能有效加强种群的多样性,使不同的子种群同时对多个峰展开搜索,从而更快速地获得最优值. 因此,文本采用多种群策略求解动态多峰问题. 为避免多个子种群同时搜索同一个峰,本文提出一种新的排斥策略,策略通过计算子种群间的距离 $dist$,与阈值 δ 进行比较,若 $dist<\delta$,则需进一步通过 hill-valley[23] 函数来进一步判定两个子种群是否对同一峰进行跟踪. 若是,则将两个子群中性能较差的劣势子种群在搜索范围内以扩散方式尽可能得远离原搜索区域.

新排斥策略两个急待解决的关键点是:(1)如何度量子种群间的距离?(2)对于劣势子种群采用何种方式进行扩散?

2.2.1　子种群间距离的度量

文献[19]中,以当前领域最优个体代表每个子种群,通过欧式距离公式计算每个子群间的距离. 这种简单的以某个点来代表某个种群的方式没有充分考虑种群中个体之间的分布特性,且对种群个体存在较大的极值影响. 为克服上述问题,本文受聚类算法中组间平均链锁法(between-groups linkage)[24]与马氏距离(Mahalanobis distance)[25]启发,提出一种新的度量两种群间距离的计算公式,在此称之为群间平均马氏距离(between-swarms average Mahalanobis distance). 在给出相关定义之前,首先给出个体与种群间的马式距离的定义.

定义 1　个体与种群间的马式距离. 设 X 是一个含有 N 个个体的种群,μ 为种群 X 中所有个体的位置均值,即:$\mu = E(X)$,Σ 为 X 中个体的协方差矩阵,则解空间中任一个体 x_t 到种群 X 的距离定义为:

$$d\_m(x_t, X) = (x_t - u)\Sigma^{-1}(x_t - u)^{\mathrm{T}}. \tag{8}$$

定义 2　群间平均马氏距离(Between-Swarms Average Mahalanobis Distance). 设 pop_k 与 pop_i 为群体中任意两子种群,则 pop_k 到 pop_i 之间的平均马氏距离定义如下:

$$Dist_{ki} = \sum_{t=1}^{N} d\_m(x_t, pop_i)/N. \tag{9}$$

其中,N 为 pop_k 种群规模;x_t 为 pop_k 中任意个体,即:$x_t \in pop_k (t = 1,2,\cdots,N)$. $d\_m(x_t, pop_i)$ 为 pop_k 中个体 x_t 到 pop_i 距离.

由定义 1 可见,与欧氏距离不同,马式距离是基于个体分布的一种距离公式,它充分考虑到种群中个体之间的相关程度,通过种群协方差矩阵,度量个体与种群间的距离(相似度). 而群间平均马式距离则是通过计算种群中每个个体到另一种群间马式距离的和平均来度量两个种群的距离(相似度),从而避免了

种群中离群点对距离判定的极值影响.

利用多种群对多峰问题进行搜索过程中,最理想的状态是:一个子种群对一个峰进行搜索. 因此,若 $dist<\delta$,说明两子种群距离极小,搜索领域可能重叠,从而可以考虑将一个当前处于劣势子种群移出原搜索区域. 但事实上,对于任意两个邻近且相邻距离极小的峰,仅用上述距离判定法而考虑移除某个子种群可能是不完备的. 为避免误判,使每个子种群尽可能地对峰进行一对一地搜索,本文采用 *hill-valley* 原理(见原理示意图 2),利用式(10)对小区域内峰的个数做进一步判断.

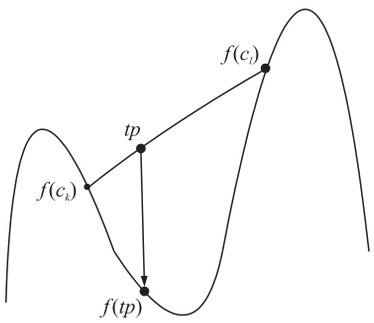

图 2　hill-valley 原理示意图

Fig. 2　hill-valley schematic diagram

实验分析取得经验阈值 $\delta=X/2\cdot\exp(\sqrt[D]{m})$,其中:$X$ 为空间搜索范围,m 是峰的个数. 易见,维数 D 越高,单位空间内峰数 m 越大,δ 越小.

$$tp=\varphi\cdot c_k+(1-\phi)\cdot c_l. \tag{10}$$

其中,φ 是一个任意常数,$\varphi\in(0,1)$,在算法性能分析实验中取 $\varphi=0.5$. c_k 与 c_l 为不同的两个子种群中心,$c_i=(c_{i1},c_{i2},\cdots,c_{iD})$ $(i=1,2,\cdots k,\cdots l,\cdots S)$ (S 为子种群数目),且:

$$c_{kj}=\frac{\sum_{k=1}^{N}x_{kj}}{N},\quad (j=1,2,\cdots,D). \tag{11}$$

若 $f(tp)<\min\{f(c_k),f(c_l)\}$(如图 2 所示),则说明子种群 pop_k 与 pop_i 分别跟踪不同的峰,两子种群都无需移除. 反之,将劣势子种群(适应值较差)进行“重新初始化”操作.

2.2.2　反向扩散操作

对于种群中任意两个子种群,通过距离度量方法(2.2.1),若两者距离小于阈值 δ,即:两子种群搜索重叠,则为了提向搜索效率,维持种群多样性,扩大搜索空间,需对其中劣势子种群进行“重新初始化操作”分为随机初始化与反向扩散操作两部分. 首先,初始化随机产生 $(1-perc)\cdot N$ 个个体($perc$ 为扩散比例);其次,受对偶原理启发,另 $perc\cdot N$ 个个体将以个体原所在位为基准进行反向扩散操作(reversed diffuse, RD)(如示意图 3). 其中,ave 表示劣势子群中所有个体的平均位置;$[L,U]$为空间搜索边界.

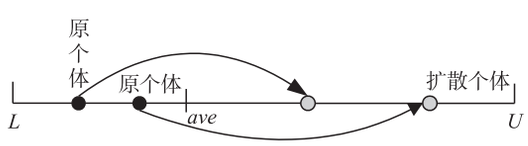

图 3　反向扩散操作示意图

Fig. 3　Reversed diffuse operation diagram

由图 3 可见,反向扩散操作使个体从原位向另一方边界反向扩散. 各维扩散操作具体如下(式(12)和(13)).

设 $x_i^t=(x_{i,1}^t,x_{i,2}^t,\cdots,x_{i,D}^t)$ 为 t 时刻 D 维空间劣势子种群 pop_k 中的任意个体,$x_{i,j}^t\in[l_j,u_j]$,此处,$l_i\in L$,$u_i\in U(i=1,2,\cdots D)$,其平均位置向量 $ave=(ave_1,ave_2,\cdots ave_j,\cdots,ave_D)$ 中每一维定义为:

$$ave_j=\sum_{i=1}^{N}x_{i,j}^t. \tag{12}$$

其每一维在下一时刻以子种群平均位置反向扩散. 反向扩散操作 RD 如式(13)所示:

$$x_{i,j}^{t+1}=\begin{cases}l_j+\dfrac{x_{i,j}^t-ave_j}{u_j-ave_j}\cdot(ave_j-l_j), & x_{i,j}^t\in[ave_j,u_j),\\ ave_j+\dfrac{x_{i,j}^t-l_j}{ave_j-l_j}\cdot(u_j-ave_j), & x_{i,j}^t\in[l_j,ave_j).\end{cases} \tag{13}$$

该操作借助历史信息,使劣势子群中所有个体尽可能地远离原子群位置,向边界反向扩散. 扩散操作二维示意效果图如图 4 所示. 其中,蓝色圆点表示原始种群,扩散后种群以红色圆点表示. 五角星表示各自种群中心. 从图 4 可见,扩散操作后种群中心将远离原始种群中心,且各扩散后的粒子更加分散. 在下一代的进化过程中,扩散后新产生的子种群将开始对“希望”子区域进行新地探索.

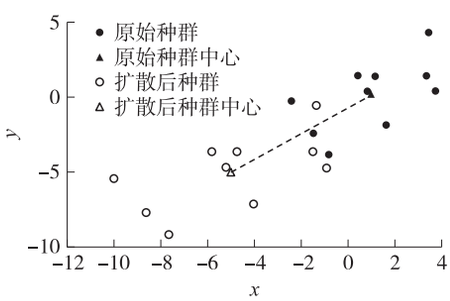

图 4　反向扩散二维示意效果图

Fig. 4　Reversed diffuse 2D effect diagram

新排斥策略对可能重叠搜索同一峰的两子种群借助 hill-valley 思想做出进一步的判定,对其中的劣势子种群采用 RD 策略重新初始化,使得初始化的子种群具有良好的多样性,从而保证整体种群具备持续跟踪动态最优值的能力. 其详细执行步骤如算法 1 所示.

算法 1　ExclusionReinitialize(k,l)基本步骤

for each pair of subswarms(k,l) **do**
  if dist(k,l)<δ **then**
    $tp=\varphi\cdot c_k+(1-\phi)\cdot c_l$;
    if $f(tp)<\min\{f(c_k),f(c_l)\}$ **then**
  if subswarms(k) inferior to subswarms(l) **then**
  rand generateperc·Nparticle for pop_k;
  generate(1-perc)·Nparticle by RD operator for pop_k;
  else generateperc·Nparticle for pop_l;
generate(1-perc)·Nparticle by RD operator for pop_l;
    end if
  end if
  end if
end for

2.3　ARDEA 算法

自适应反向扩散演化算法(ARDEA)由 S 个子种群组成,每个子种群 $pop_k(k=1,\cdots,S)$ 包含 N 个个体;个体速度的取值范围为$[v_{\min},v_{\max}]$;$gbest_k^g$ 为第 k 个子群在第 g 代的最优位置. 每个子群中个体历史最优解被存储在一个优势集 $Sup_k(k=1,\cdots,S)$ 中. 当环境发生变化后,每个子群将利用历史经验对个体进行重新初始化,若 Sup_k 中记忆个体优于初始个体,将取代之. 子种群中所有个体采用改进后的粒子群速度更新操作不断进化,持续跟踪动态最优值. 具体步骤如算法 2 所示.

算法 2　ARDEA 算法基本步骤

%flag:the mark variable of environment change
for $k=1$ to S **then**
  randomly initialize position and velocity of each individual in pop_k;
  calculate the fitness all individuals in pop_k;
  assign pbest with current position of each individual;
  calculate the $gbest_k^g$ in pop_k;
end for
while the stopping criterion is not meet do
  flag=0;
  for $k=1$ to S **then**
    recalculate the fitness of $gbest_k^{g-1}$;
    if the value of $gbest_k^{g-1}$ has change **then**
      flag=1;
    end if
  end for
  if flag==1 **then**
    for $j=1$ to S **then**
      reinitialize pop_k;
      recalculate the fitness of all individuals in pop_j;
      updatepbest of all individuals in pop_j;
      update $gbest_k^g$ of pop_j;
  end for
  end if
  for $k=1$ to S **then**
    for $l=1:N$ **then**
      ExclusionReinitialize(k,l);
    end for
    reperform new improved PSO operators in pop_k;
end for
end while

2.4　算法时间复杂度分析

由算法 2 可知,ARDEA 算法主要包含 4 部分:种群初始化、环境变化的监测与响应、排斥策略以及 VD 速度更新操作. 易知,对 S 个子种群进行初始化的时间复杂度为 $O(S\cdot N\cdot D)$,其中 N 是子群中个体数目,D 为个体维度;对环境监测部分为 $O(S)$,环境响应即初始化,时间复杂度 $O(S\cdot N\cdot D)$;在排斥策略中,主要涉

及到子种群重叠搜索判定、反向扩散初始化操作,前者需在 $O(S^2)$ 时间完成,后者执行反向扩散操作需时间复杂度为 $O(N\cdot D)$;PSO-VD 复杂度为 $O(S\cdot N\cdot D)$. 一般情况下,子种群个数 S 一般取值大于或等于峰的个数 N;当 D 较小时,子种群规模 N 近似 D;若 D 较大时,例如:$D\geqslant 100$,N 通常小于 D. 综合上述分析,ARDEA 的计算复杂度为 $O(S^2\cdot D)$.

3 测试函数

3.1 MPB 问题

移动峰问题(moving peaks benchmark,MPB)[10] 最早由 Branke 在 1999 年提出并得到广泛应用的动态测试标准函数. 本文参考 CEC'2009[26-27] 中对环境动态变化的定义,MPB 中各峰在下一时刻采用动态旋转峰生成器(DRPBG)对 MPB 函数中各峰的高度 $\boldsymbol{H}$、宽度 $\boldsymbol{W}$ 和位置 $\boldsymbol{X}$ 进行旋转动态变化.

MPB 函数 $f(x,\phi,t)$ 定义如(14),$\phi=(\boldsymbol{H},\boldsymbol{W},\boldsymbol{X})$:

$$f(x,\phi,t)=\max_{i=1}^{m}\left(\boldsymbol{H}_i(t)/\left(1+\boldsymbol{W}_i(t)\cdot\sqrt{\sum_{j=1}^{D}\frac{(x_j-\boldsymbol{X}_j^i(t))^2}{D}}\right)\right). \tag{14}$$

函数中,m 是峰的个数,n 是维度. $\boldsymbol{H}$、$\boldsymbol{W}$ 和 $\boldsymbol{X}$ 向量在环境变化的下一时刻如下产生:

$$\boldsymbol{H}(t+1)=\text{DynamicChanges}(\boldsymbol{H}(t)), \tag{15}$$

$$\boldsymbol{W}(t+1)=\text{DynamicChanges }\boldsymbol{W}((t)), \tag{16}$$

$$\boldsymbol{X}(t+1)=\boldsymbol{X}(t)\cdot A(t). \tag{17}$$

文献[27]中,环境被分为 small step(T0)、large step(T1)、random(T2)、chaotic(T3)、recurrent(T4)、recurrent with noisy(T5)和 random with changed dimension(T6)7 种,即:环境变化由小变大,由随机到无序,环境周期变化到带噪声的周期变化,以及环境维度随机变化. DynamicChanges(·)将根据环境变化类型的不同采用不同的变化表达式,具体操作参考文献[27];$\boldsymbol{A}(t)$ 为旋转矩阵,产生方式同样见参见文献[27]. DRPBG 相关参数设置见表 1.

表 1 DRPBG 参数设置

Table 1 DRPBG parameter setting

Parameter	Value
Number of peaks	P=10 or 50
Change frequency	10 000×D
Number of dimensions	D(fixed)=10
Peak heights	∈[10,100]
Peak widths	∈[1,10]
Number of changes	60
Search range	$\in[-5,5]^D$
Sampling frequency	100

3.2 评估标准

本文采用 CEC'2009[20] 中给出的算法评估标准,包括:平均最优值(*Avg_best*)、平均均值(*Avg_mean*)、和标准偏差(STD),具体定义如下:

$$Avg\_best=\sum_{i=1}^{runs}Min_{j=1}^{num\_change}E_{i,j}^{last}(t)/runs, \tag{18}$$

$$Avg\_mean=\sum_{i=1}^{runs}\sum_{j=1}^{num\_change}E_{i,j}^{last}(t)/(runs*num\_change), \tag{19}$$

$$STD=\sqrt{\frac{\sum_{i=1}^{runs}\sum_{j=1}^{num\_change}(E_{i,j}^{last}(t)-Avg\_mean)^2}{runs*num\_change-1}}. \tag{20}$$

其中,*runs* 是运行次数,*num_change* 是环境变化次数,$E_{i,j}^{last}(t)=|f(x_{\text{best}}(t))-f(x^*(t))|$ 指在 t 时该最优个体适应值与全局最优值之间的离线误差.

算法整体性能通过 *performance* 计算:

$$performance=\sum_{k=1}^{number\_of\_leaf\_nodes}mark_k*weight_k, \tag{21}$$

$$mark_k=percentage_k\frac{\sum_{i=1}^{\text{runs}}\sum_{j=1}^{num\_change}r_{i,j}}{num\_change*runs}, \tag{22}$$

$$r_{i,j}=r_{i,j}^{last}\Big/\left(1+\sum_{s=1}^{S}(1-r_{i,j}^{s})/S\right), \tag{23}$$

$$S=change\_frequency/s\_f. \tag{24}$$

以上表达式中,S 为采样次数,$r_{i,j}^{s}$是每次采样最优个体适应值与全局最优值比值,$r_{i,j}^{last}$为每次环境发生变化前最优个体适应值与全局最优值比值. s_f 是采样频率.

4　结果与讨论

本文所有实验统一在 MATLAB2014b 环境下针对 MPB 问题进行性能测试. 实验分为 3 个部分,第一部分,通过平均最优值、平均均值和平均方差评估值 3 个评估指标对 ARDEA 算法性能进行分析;第二部分,将 ARDEA 与其它 3 种主流动态优化演化算法进行实验对比,即:CPSO[13]、jDE[18] 以及 Multi-DEPSO[26];第三部分,对 ARDEA 算法中的关键参数进行了参数敏感性分析.

4.1　性能分析

本文将 MPB 测试问题中峰的个数在取 10 或 50 的情况下分别进行实验($F1:P=10$,$F2:P=50$). 每个问题实例独立运行 20 次,取其平均最优值 *Avg_best*、平均均值 *Avg_mean*、平均方差 *STD* 评估值列于表 3($F1:P=10$)和表 4($F2:P=50$). 每个测试函数最优 *Avg_mean* 值用粗体显示. 在个体利用 *VD* 速度更新公式进行进化的过程中,更新粒子最大速度 $v_{\max}$为搜索空间的一半. 实验在维数 $D=10$ 下进行,具体参数设置如表 2 所示. 为确保算法性能,相关涉及参数优化设置参见文献[20,22,28].

表 2　ARDEA 参数设置

Table 2　ARDEA parameter setting

$c1$	$c2$	s	*perc*	φ	w	D
2	2	0.2	0.5	0.5	0.5	10

从表 3 和表 4 中可得出 ARDEA 在 7 种不同动态环境中总体表现占优. 其中,当峰数 $P=10$ 时(如表 3 所示),在 T1、T4、T5 和 T6 四种动态环境下取得较优的结果;当峰数 $P=50$ 时(如表 4 所示),在 T0、T1、T2、T4、和 T6 5 种动态环境下取得较优的结果. 由此可见,ARDEA 在峰数越高表现结果越好,但在 chaotic(T3)环境中表现均未取得最优结果. 另外,ARDEA 算法的 STD 值普遍较小,说明该算法相较于其它对比算法具有良好的鲁棒性和适应性.

表 3　CPSO、JDE、Multi-DEPSO,ARDEA 算法在 MPB 测试问题 $F1:P=10$ 下的测试结果

Table 3　The test results of CPSO,JDE,Multi-DEPSO and ARDEA on MPB problem $F1:P=10$

算法	测试结果	T0	T1	T2	T3	T4	T5	T6
CPSO	*Avg_best*	1.12E−7	5.23E−8	6.23E−09	9.86E−7	3.03E−7	4.13E−06	5.04E−9
	Avg_mean	0.050	3.650	4.342	0.092	1.964	1.182	4.543
	STD	0.453	4.341	7.896	0.784	5.301	4.866	9.117
jDE	*Avg_best*	0	0	1.17E−16	0	0	0	0
	Avg_mean	0.013	0.269	3.383	0.781	1.979	2.433	1.678
	STD	0.453	4.341	7.896	0.784	5.301	4.866	9.117
Multi-DEPSO	*Avg_best*	1.92E−3	1.59E−3	1.09E−3	1.83E−3	2.54E−3	2.00E−3	1.04E−3
	Avg_mean	**0.005**	2.061	**1.043**	**0.024**	0.281	0.012	2.381
	STD	0.001	1.301	1.231	0.000	0.143	0.002	2.003
ARDEA	*Avg_best*	1.65E−5	1.75E−9	7.21E−9	1.66E−8	3.54E−7	1.05E−7	5.44E−8
	Avg_mean	0.006	**4.03E−4**	1.74	0.030	**0.058**	**0.010**	**0.139**
	STD	0.003	1.011	2.201	0.345	3.143	0.904	2.723

表 4　CPSO、jDE、Multi-DEPSO,ARDEA 算法在 MPB 测试问题 $F2:P=50$ 下的测试结果

Table 4　The test results of CPSO,JDE,Multi-DEPSO and ARDEA on MPB problem $F2:P=50$

算法	测试结果	T0	T1	T2	T3	T4	T5	T6
CPSO	*Avg_best*	2.52E−6	6.00E−7	8.96E−7	5.18E−6	2.03E−6	8.93E−06	5.74E−5
	Avg_mean	0.353	3.100	6.402	0.132	0.905	1.522	6.933
	STD	0.992	4.679	8.465	0.654	1.878	4.516	7.789
jDE	*Avg_best*	0	0	0	0	0	0	0
	Avg_mean	1.117	4.559	4.383	1.758	1.545	3.803	5.738
	STD	1.053	5.891	5.786	5.087	2.675	8.760	7.458
Multi-DEPSO	*Avg_best*	1.44E−2	1.46E−2	1.23E−2	1.34E−2	3.05E−2	1.80E−2	2.98E−3
	Avg_mean	1.231	1.710	4.367	**0.466**	0.876	**0.767**	4.176
	STD	0.076	0.451	1.830	0.024	0.123	0.052	1.103
ARDEA	*Avg_best*	1.67E−3	1.51E−2	2.53E−7	1.72E−2	1.76E−5	2.13E−6	2.84E−3
	Avg_mean	**1.001**	**0.021**	**1.09**	0.830	**0.049**	1.156	**4.103**
	STD	0.047	2.769	1.254	6.895	0.781	0.954	0.678

表 5　CPSO、jDE、Multi-DEPSO,ARDEA 算法综合性能比较

Table 5　Comprehensive performance comparison of CPSO, JDE, Multi-DEPSO and ARDEA

算法	测试函数	T0	T1	T2	T3	T4	T5	T6	mark/%	performance
CPSO	*F*1	0.015 2	0.013 4	0.012 4	0.015 5	0.021 1	0.008 3	0.008 6	9.45	18.37
	*F*2	0.014 3	0.013 3	0.013 4	0.013 5	0.013 6	0.013 1	0.008 0	8.92	
jDE	*F*1	0.014 3	0.014 6	0.013 4	0.013 9	0.013 5	0.014 1	0.009 1	9.29	18.59
	*F*2	0.013 8	0.013 4	0.015 6	0.014 5	0.014 3	0.013 2	0.008 2	9.30	
Multi-DEPSO	*F*1	0.014 5	0.014 8	0.014 5	0.014 5	0.014 9	0.014 9	0.009 3	9.74	19.36
	*F*2	0.014 5	0.014 7	0.014 3	0.013 7	0.014 9	0.014 9	0.009 2	9.62	
ARDEA	*F*1	0.014 6	0.015 6	0.017 2	0.001 3	0.016 3	0.015 6	0.009 9	9.05	**21.19**
	*F*2	0.014 8	0.014 9	0.017 8	0.014 1	0.035 3	0.014 9	0.009 6	12.14	

4.2　对比实验

为进一步获悉 ARDEA 算法在各种动态环境中的综合性能,在接下的实验中,将 ARDEA 与 CPSO、jDE 及 Multi-DEPSO 4 种算法在 7 种动态环境下进一步展开综合性能测试. 统计 mark 与 performance 值(见式(21)、(22)),通过性能对比(如表 5 所示),ARDEA 性能值获得最高值(21.19),表明 ARDEA 在 7 种动态环境中综合性能最优.

4.3　参数敏感性分析

反向扩散操作(RD)是维持 ARDEM 多样性,保证算法在动态环境下对最优解持续跟踪能力的重要策略. 其中,扩散比例参数 *perc* 是影响策略效果的关键参数,本小节将 *perc* 分别取 0.2,0.3,0.4,0.5,0.6,0.7 值,在 7 种动态环境下对 MPB 测试问题(峰数 F1,F2)展开综合性能测试. 统计 mark 与 performance 值于表 6 中,并为便于直观比较,上述数据以图形方式呈现在图 5 中. 图 5 中柱状图分别表示峰值为 10(浅灰柱)和 50(深灰柱)时 *perc* 取不同值下 mark 值;红色空圆点实线表示 performance 值. 的由图 5 和表 6 可见,峰数越高,*perc* 取值越大,算法性能越好;反之,*perc* 取值在 0.3~0.5 之间较稳定.

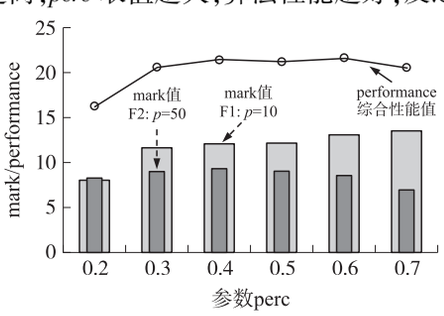

图 5　参数 perc 取不同值 mark 与 performance 性能对比

Fig. 5　The performance comparison of mark and performance when parameter perc takes different values

表 6　参数 perc 取不同值综合性能对比

Table 6　Comprehensive performance comparison when perc takes different values

perc	mark		performance
	*F*1	*F*2	
0.2	8.22	8.07	16.29
0.3	9.01	11.59	20.60
0.4	9.31	12.08	21.39
0.5	9.05	12.14	21.19
0.6	8.56	13.07	21.63
0.7	6.99	13.56	20.55

5　结论

本文针对传统演化优化算法在动态环境下难以有效地持续跟踪最优解的问题,提出了一种自适应反向扩散演化算法(ARDEA)对动态环境进行寻优. 该算法采用多种群策略对最优解进行跟踪,并通过子种群最优位 $gbest_k^g$ 作业动态监哨点(sentry)对环境变化进行动态监测;同时,引入一种差分粒子群速度更新公式(*VD*)对个体在搜索空间内进行搜索;除此之外,为保证种群多样性,以及子种群搜索效率,本文提出了一种新的排斥策略,该策略采用新提出的群间平均马氏距离判断种群间距离,对于群间距过小的两个子种群进一步通过 hill-valley 函数判定两子种群搜索区域是否重叠,并则将重叠区域中的劣势子种群通过新提出的反向扩散操作(RD)重新初始化,扩散后新产生的子种群将开始对“希望”子区域进行新地探索,从而保证整体种群具备持续跟踪动态最优值的能力. 新算法与当前性能较优的 3 种动态优化算法同时作用于移动峰测试问题(MPB),对比实验结果表明,ARDEA 算法在动态环境中能有效地跟踪最优解,与其它比较算法而言,表现出较强的鲁棒性和适应性.

未来将进一步通过更广泛的测试验证 ARDEA 的有效性,并展开算法的相应机制分析,理论证明算法的有效性,发现内在动因,为在实际问题中的推广打下基础.

[参考文献]

[1] LI Q Y,ZOU J,YANG S X,et al. . A predictive strategy based on special points for evolutionary dynamic multi-objective optimization[J]. Soft computing,2019,23(11):3723-3739.

[2] MAURIZIO F,LIBERATI D E,GUALANDI S,et al. Quantum-inspired evolutionary multiobjective optimization for a dynamic production scheduling approach[J]. Multidisciplinary approaches to neural computing,2018,69:191-201.

[3] 简琤峰,陈家炜,张美玉. 面向边缘计算的改进混沌蝙蝠群协同调度算法[J]. 小型微型计算机系统,2019,40(11):2424-2430.

[4] YANG C, DING J. Constrained dynamic multi-objective evolutionary optimization for operational indices of beneficiation process[J]. Journal of intelligent manufacturing,2019,30:2701-2713.

[5] YANG S X, YAO X. Evolutionary computation for dynamic optimization problem [M]. Boston: Spring-Verlag Berlin Heidelberg,2013.

[6] CRUZ C,GONZáLEZ J R,PELTA D A. Optimization in dynamic environments:a survey on problems,methods and measures[J]. Soft computing,2011,15(7):1427-1488.

[7] YAZDANI D,NGUYEN T T,BRANKE J,et al. A multi-objective time-linkage approach for dynamic optimization problems with previous-solution displacement restriction[J]. Lecture notes in computer science,2018,10784:864-878.

[8] 申鼎才,胡声洲. 基于领域搜索的粒子群动态优化算法[J]. 合肥工业大学学报,2017,40(5):628-632.

[9] LI C H,YANG S X. A clustering particle swarm optimizer for dynamic optimization[C]//Proceedings of the IEEE Comgress on Evolutionary Computation,Trondheim,Norway,2009,May:18-21.

[10] BRANKE J. Memory enhanced evolutionary algorithms for changing optimisation problems[C]//IEEE congress on evolutionary computation,Washington,DC,USA,1999:1875-1882.

[11] CAO L,XU L,GOODMAN E D. A neighbor-based learning particle swarm optimizer with short-term and long-term memory for dynamic optimization problems[J]. Information sciences,2018,453:463-485.

[12] LI C,YANG S. A clustering particle swarm optimizer for dynamic optimization[C]//Proceedings of the IEEE Congress on Evolutionary Computation,2009:439-446.

[13] YANG S,LI C. A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments[J]. IEEE Transations on Evolutionary Computation,Proceedings of the IEEE Congress on Evolutionary Computation,2010,6(10):959-974.

[14] 卜晨阳. 演化约束优化及演化动态优化求解算法研究[D]. 合肥:中国科学技术大学,2017.

[15] 李志坚. 改进的差分演化算法及其在动态优化问题中的应用[D]. 武汉:华中师范大学,2016.

[16] HUI S,SUGANTHAN P N. Ensemble differential evolution with dynamic subpopulations and adaptive clearing for solving dynamic optimization problems[C]//IEEE congress on evolutionary computation. Brisbane,Australia,2012:1-8.

[17] MENDES R,MOHAIS A S. DynDE:a differential evolution for dynamic optimization problems[C]//IEEE Congress on Evolutionary Computation,Edinburgh,UK,2005:2808-2815.

[18] BREST J, ZAMUDA A, BOSKOVIC B, et al. Dynamic optimization using self-adaptive differential evolution [C]//IEEE Congress on Evolutionary Computation,Trondheim,Norway,2009:415-422.

[19] KENNEDY J,EBERHART R C. Particle swarm optimization[C]//Proceedings of IEEE International Conference on Neural Networks. Perth,Australia,1995:1942-1948.

[20] 康岚兰,董文永,宋婉娟,等. 无惯性自适应精英变异反向粒子群优化算法[J]. 通信学报,2017,38(8):66-78.

[21] STORN R,PRICE K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of global optimization,1997,11:341-359.

[22] 周新宇,吴志健,王晖,等. 一种精英反向学习的粒子群优化算法. 电子学报,2013,41(8):1647-1652.

[23] BLACKWELL T,BRANKE J. Multi-swarm optimization in dynamic environments[C]//Applications of Evolutionary Computing. Coimbra,Porugal,2004:489-500.

[24] 谢修娟,李香菊,莫凌飞. 基于改进 K-means 算法的微博舆情分析研究[J]. 计算机工程与科学,2018,40(1):155-158.

[25] ZU Z W,LI Q. Mahalanobis distance fuzzy clustering algorithm based on particle swarm optimization[J]. Journal of Chongqing University of posts and telecommunications(natural science edition),2019,31(2):275-284.

[26] ZUO X,XIAO L. A DE and PSO based hybrid algorithm for dynamic optimization problems[J]. Soft computing,2014,18:1405-1424.

[27] LI C,YANG S,NGUYEN T T,et al. Benchmark Generator for the CEC'2009 Competition on Dynamic Optimization[R]. University of Leicester, University of Birmingham, Honda Research Institute Europe, Vorarlberg University of Applied Sciences,Nanyang Technological University,Technical Report,October 26,2008:1-14.

[28] HAN D P,KIRSTEN E,JAN C B,et al. A survey of dynamic parameter setting methods for nature-inspired swarm intelligence algorithms[J]. Neural computing and applications. 2020,32:567-588.

[责任编辑:顾晓天]