基于种群多样性的可变种群缩减差分进化算法

单天羽 管煜旸

(浙江工业大学计算机科学与技术学院 杭州 310023)

摘 要 为了更有效地避免早熟收敛,提高算法的全局搜索能力,提出了基于种群多样性的可变种群缩减差分进化算法(Dapr-DE)。首先,Dapr-DE 使用群体多样性指标控制种群规模缩减;然后,使用聚类将种群分为不同类簇,在类簇中根据适应度值删除个体,既维持了种群的多样性,又减少了由于存在过多相似个体而导致的局部收敛。最后在 CEC14 测试集的 30 个函数优化问题上进行了实验比较,验证了所提算法的有效性。

关键词 差分进化算法,种群多样性,聚类,启发式算法

中图法分类号 TP301 **文献标识码** A

Differential Evolution Algorithm with Adaptive Population Size Reduction Based on Population Diversity

SHAN Tian-yu GUAN Yu-yang

(College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract To avoid premature effectively and improve the capability of global search, an algorithm named differential evolution algorithm with adaptive population size reduction based on population diversity(Dapr-DE) was proposed. Dapr-DE firstly uses population diversity to control the population size reduction. Then, Dapr-DE divides the population into some subpopulations by clustering and deletes the individuals according to their fitness, which keeps population diversity effectively and avoids local convergence. At last, the experimental results validate the effectiveness of the proposed algorithm on 30 real optimization problems in the CEC14 function set.

Keywords Differential evolution algorithm, Population diversity, Clustering, Heuristic algorithm

1 引言

差分进化算法[1-2]是由 Storn 等于 1995 年提出的采用浮点矢量编码,在连续空间内进行随机搜索的智能优化算法。它采用种群的全局搜索策略,加入基于差分的简单变异操作和一对一的竞争生存策略,有效地降低了遗传操作的复杂性。近年来,差分进化算法作为一种性能卓越的优化算法,在约束优化计算[3-4]、模糊控制优化设计[5-7]以及神经网络优化[8-9]等方面得到了广泛应用。

差分进化算法的结构与遗传算法[10-11]相似,但子代的生成方式有所不同,其最大的特点是父代多个个体之间的线性组合生成的子代,而不是遗传算法中传统的父代染色体交叉重组。同时,差分进化算法采用"贪婪"的选择操作,使得算法具有快速的收敛特性,但是增大了算法陷入局部最优或早熟收敛的概率。目前主要的解决方法是增加种群的规模,但这样会增加算法的运算量。为了提高算法的性能,很多学者提出了基于种群规模缩减的改进方法[12-13]。Brest 提出了种群规模缩减差分进化算法(pr-DE)[13],在算法初期给定一个规模很大的种群,人为设定缩减次数,每次随机选取一半个体,达成缩减一半种群规模的目的。这样能够保证算法不过早收敛,同时还能有效地提高算法性能和效率。但是,算法后期可能删除有潜力的个体,导致算法仍可能陷入局部最优;而随机选取个体的策略也不能很好地针对种群具体情况实施,使得种群缩减的效用大打折扣。

针对以上不足,本文将在 pr-DE 算法的基础上,采用种群多样性指标自适应改变种群规模的缩减:在进化初期,即种群的多样性较高时,种群缩减的规模需较大;在进化后期,即种群多样性较低时,种群缩减的规模需较小;同时,在种群缩减时,加入了聚类操作和基于适应度值的删除操作,使得种群中的个体分布更加均匀,从而提高算法的鲁棒性。在 CEC14 测试集上的函数测试结果表明,与传统差分进化算法和目前较流行的改进的差分进化算法相比,本文提出的新算法能够有效避免早熟收敛,不同问题的求解质量都较好。

本文首先介绍了差分进化算法的基本原理和种群规模缩减差分进化算法;接着,提出改进的基于种群多样性的可变种群规模进化算法;然后,采用 CEC14 单目标数值优化竞赛的 30 个基准函数对 4 种算法进行测试,并进行实验分析;最后,对所提算法的特性进行讨论并指出进一步研究的方向。

2 差分进化算法介绍

2.1 概述

差分进化算法[1-2],最早是 1995 年由美国科学家 Store 和 Price 提出的一种计算方法,其凭借简单有效的特性吸引了大量研究者的关注,在过去的 20 余年里得到了充分的发

本文受国家自然科学基金(61573316)资助。

单天羽(1992—),男,硕士生,主要研究方向为进化算法、智能计算,E-mail:martinof92@163.com(通信作者);**管煜旸**(1993—),女,硕士生,主要研究方向为进化算法。

展。差分进化算法的应用也十分广泛，如神经网络优化、约束优化计算等。

2.2 基本原理

设差分进化算法[1-2,14]的种群有 NP 个个体，个体由 D 维向量表示，具体如下：

$$\boldsymbol{X}_i^{(G)}=\{x_{i,1}^{(G)},x_{i,2}^{(G)},\cdots,x_{i,D}^{(G)}\} \tag{1}$$

其中，G 代表进化代数，$i=1,2,\cdots,NP$。

首先，差分进化算法随机生成个体组成初始种群。接着，差分进化算法执行 3 个进化操作：变异操作、交叉操作和选择操作。这一过程使用了 2 个控制参数：差分因子 F 和交叉概率 CR。

1）变异操作

种群中每一个向量都通过变异产生新的变异向量：

$$\boldsymbol{V}_i^{(G)}=\{v_{i,1}^{(G)},v_{i,2}^{(G)},\cdots,v_{i,D}^{G}\} \tag{2}$$

变异向量可以通过某一变异策略得到，本文介绍一种最常用的变异策略 rand/1[15-16]：

$$\boldsymbol{V}_i^{(G)}=x_{r_1}^{(G)}+F*(x_{r_2}^{(G)}-x_{r_3}^{(G)}) \tag{3}$$

其中，下标 r_1,r_2,r_3 代表不同于 i 且在[1，NP]范围中随机产生的整数值，而变异系数 F 的范围为[0，2]，通常小于 1。

2）交叉操作

交叉操作使用式(4)对变异向量和对应的亲本向量逐项选择，得到交叉向量。

$$u_{i,j}^{(G)}=\begin{cases}v_{i,j}^{(G)}, & \text{if rand}(0,1)\leqslant CR \text{ or } j=j_{\text{rand}}\\ x_{i,j}^{(G)}, & \text{otherwise}\end{cases} \tag{4}$$

其中，$j=1,2,\cdots,D$；CR 是交叉概率，$CR\in[0,1)$；下标 j_{rand} 是一个属于[1，D]的随机整数值，它确保变异向量至少有一项参与交叉操作。

为了防止交叉向量没有任何一项超出定义域，一般用修正函数[2,14-15]加以控制，其公式如下：

$$u_{i,j}^{(G)}=\begin{cases}2*x_{j,\text{low}}-u_{i,j}^{(G)}, & \text{if } u_{i,j}^{(G)}<x_{j,\text{low}}\\ 2*x_{j,\text{upp}}-u_{i,j}^{(G)}, & \text{if } u_{i,j}>x_{j,\text{upp}}\end{cases} \tag{5}$$

其中，$x_{j,\text{low}},x_{j,\text{upp}}$ 分别表示第 j 项的定义域上界和下界，$j\in\{1,\cdots,D\}$。

3）选择操作

通过比较交叉向量和对应的亲本向量的适应度值确定其中的较优个体进入下一代（$G+1$ 代）。在求函数最小值的情况下，选择操作由式(6)确定：

$$\boldsymbol{X}_i^{(G+1)}=\begin{cases}\boldsymbol{U}_i^{(G)}, & \text{if } f(\boldsymbol{X}_i^{(G)})<f(\boldsymbol{U}_i^{(G)})\\ \boldsymbol{X}_i^{(G)}, & \text{otherwise}\end{cases} \tag{6}$$

通过这 3 个操作，种群中随机选择的亲本个体和若干其他个体重组产生新的个体，然后比较解新个体和亲本个体的适应度值来选择较好的个体参与接下来的进化，如此逐代改善种群的质量。

2.3 种群规模缩减差分进化算法

Brest 提出的种群规模缩减差分进化算法 pr-DE[13]对种群规模 NP 进行控制。该算法设定有 $p_{\max}$ 次种群缩减，每一次缩减后，种群规模减半。

设 $NP_1=NP_{\text{init}}$，即初始种群规模，而 NP_p（$p=1,2,\cdots,p_{\max}+1$）则是经过 $p-1$ 次种群缩减后的种群规模。当种群规模为 NP_p 时，该算法需执行的进化代数 gen_p 为：

$$gen_p=\left\lfloor\frac{maxnfeval}{p_{\max}*NP_p}\right\rfloor \tag{7}$$

其中，$maxnfeval$ 是算法预设的终止条件评估总数，由人为设定。

得到 gen_p，算法进行种群缩减的时间也就确定了。即当 $G=gen_1,gen_1+gen_2,\cdots,\sum_{p=1}^{p_{\max}}gen_p$ 时，算法执行种群缩减操作。种群规模减半的具体实现是：将个体 X_i 与个体 $X_{i+NP/2}$ 进行适应度值比较，选择较好的个体留下，删除较差的个体。公式如下：

$$\boldsymbol{X}_i^{(G)}=\begin{cases}(\boldsymbol{X}_{i+NP/2}^{(G)}, & \text{if } f(\boldsymbol{X}_i^{(G)})<f(X_{i+NP/2}) \text{ and } G=GR\\ \boldsymbol{X}_i^{(G)}, & \text{otherwise}\end{cases} \tag{8}$$

$$NP^{(G+1)}=\begin{cases}\dfrac{NP^{(G)}}{2}, & \text{if } G=GR\\ NP^{(G)}, & \text{otherwise}\end{cases} \tag{9}$$

其中，$i=1,2,\cdots,NP/2$，$GR\in\{gen_1,gen_1+gen_2,\cdots,\sum_{p=1}^{p_{\max}}gen_p\}$。

3 基于种群多样性的可变种群缩减差分进化算法

基于 Brest 提出的 pr-DE，本文使用控制种群多样性的缩减过程，提出了一种改进算法——基于种群多样性[17-18]的可变种群规模进化算法（Dapr-DE）。该算法对种群大小进行 $p_{\max}$ 次缩减，每一次删减的个体数量根据种群的多样性变化来调整，而不是简单地减半；此外，其使用聚类[19-20]将种群划分为若干子种群，每个子种群根据其总适应度确定要删除的个数，对删除作出进一步的优化，使得种群中的个体分布更加均匀，以保证种群多样性，避免过早收敛，从而提高算法的鲁棒性。

3.1 种群多样性计算

本文使用种群熵[21]作为衡量种群多样性的指标。计算熵 H 的方法如下。

设当前种群有 NP_{cur} 个个体，算法进化过程中迄今得到的最优个体和最差个体的适应度值分别为 $f_{\max}$ 和 $f_{\min}$。

1）解空间表示：$\xi_{\min}=(1-\delta)f_{\min}$，$\xi_{\max}=(1+\delta)f_{\max}$，$\delta=0.1$，则解空间 S 可以用区间[$\xi_{\min}$，$\xi_{\max}$]表示，区间长度 $\lambda=\xi_{\max}-\xi_{\min}$。

2）解空间分割：把区间[$\xi_{\min}$，$\xi_{\max}$]分成 $M(M=NP)$ 个小区间[$\xi_{\min}+\lambda(i-1)/M$，$\xi_{\max}+\lambda(i-1)/M$]，统计每一区间的个体数 N_i，$i=1,2,\cdots,M$。

3）计算个体出现在第 i 区间的概率 p_i：

$$p_i=\frac{M_i}{M} \tag{10}$$

4）种群熵值计算：

$$H=-\sum_{i=1}^{M}p_i\ln p_i \tag{11}$$

5）标准化处理：很多时候多样性的范围较小，会导致删除个体过少，影响算法效率。因此本文采用 sigmoid 函数对多样性数据进行标准化处理：

$$H_{\text{std}}=\frac{1}{1+\exp(-H)} \tag{12}$$

6）阈值处理：为了防止删除全部的个体或太多的个体，加入阈值。

$$H_{\text{std}}=\begin{cases}0.5, & \text{if } H_{\text{std}}>0.5\\ H_{\text{std}}, & \text{otherwise}\end{cases} \tag{13}$$

3.2 聚类处理

加入聚类的主要目的在于,根据个体之间的相似性形成类簇,并在类簇中根据适应度值删除个体,这样既能够维持种群的多样性,也能够减少因存在过多相似个体导致的局部收敛。

为了提高算法效率,本文采用 K-means 算法[20],如算法 1 所示。

算法 1 K-means 算法

1. 随机选取 K 个聚类中心
2. 满足循环条件
3. 对于种群中的每一个个体,执行
4. 计算其到每一聚类中心的距离
5. 把该个体划分到距离最近的类簇
6. 重新计算得到新的聚类中心
7. 输出结果

3.3 删除个体

根据 H_{std}确定种群需要删除的个体数目:

$$NP_{\text{del}}=NP_{\text{cur}}*H_{\text{std}} \tag{14}$$

再根据聚类得到 K 个类簇,然后使用式(15)确定每个类簇中要删除的个数:

$$del_i=NP_{\text{del}}*\frac{NP_i}{NP_{\text{cur}}} \tag{15}$$

其中,del_i 表示第 i 个聚簇集合中要删除的个体数目,NP_{del}表示本次要执行删减的所有个体数目,NP_i 表示第 i 个聚簇集合中的个体数目,NP_{cur}表示当前种群的数目。每个类簇中的个体按照适应度值进行排序,删除最差的 del_i 个个体。

综上,算法的流程如算法 2 所示。

算法 2 Dapr-DE 算法

1. 随机初始化规模为 NP_{init}的种群,G=1,通过式(7)求得 GR
2. 满足循环条件
3. 计算种群中每个个体的适应度值
4. 对于种群中每一个个体,执行
5. 变异操作
6. 交叉操作
7. 评估个体适应度值,执行选择操作
8. if G=GR
9. 由式(10)-式(13)计算种群的熵
10. 使用 K-means 对种群进行聚类,得到 K 个类簇
11. 由式(14)、式(15)得到不同子种群的 del_i,按照适应度值进行排序,删除最差的 del_i 个个体
12. 由式(7)求得 GR′,GR=GR+GR′
13. G=G+1
14. 输出进化结果

4 实验分析

4.1 数据集和对比算法

本文使用 CEC14 单目标数值优化竞赛的 30 个基准函数[22]作为测试的函数数据集,如表 1 所列。这个数据集涵盖了多样的函数优化问题类型。在这个数据集上,将本文所提算法(Dapr-DE)与 TDE,pr-DE,PA-jDE 3 个算法进行对比。

TDE 是经典的差分进化算法,进化过程维持种群大小不变,除了基本的变异、交叉和选择外,没有任何额外操作。

pr-DE 已经在本文第 2 节详细阐述。

PA-jDE 利用种群的总欧氏距离 dG 作为种群多样性指标,当 dG 若干代维持不变时,使用当前最优解对整个种群的个体进行再生成操作。该算法虽未改变种群规模,但作为使用种群多样性的差分进化算法参与实验比较,也具有一定的意义。

表 1 CEC14 测试函数

类型	ID	函数
单峰函数	f_1	Rotated high conditioned elliptic 函数
	f_2	Rotated bent cigar 函数
	f_3	Rotated discus 函数
多峰函数	f_4	Shifted and rotated Rosenbrock 函数
	f_5	Shifted and rotated Ackley's 函数
	f_6	Shifted and rotated Weierstrass 函数
	f_7	Shifted and rotated Griewank's 函数
	f_8	Shifted Rastrigin 函数
	f_9	Shifted and rotated Rastrigin's 函数
	f_{10}	Shifted Schwefel 函数
	f_{11}	Shifted and rotated Schwefel's 函数
	f_{12}	Shifted and rotated Katsuura 函数
	f_{13}	Shifted and rotated HappyCat 函数
	f_{14}	Shifted and rotated HGBat 函数
	f_{15}	Shifted and rotated Expanded Griewank's plus Rosenbrock's 函数
	f_{16}	Shifted and rotated Expanded Scaffer's 函数
混合函数	f_{17}	混合函数 1($f_9;f_8;f_1$)
	f_{18}	混合函数 2($f_2;f_{12};f_8$)
	f_{19}	混合函数 3($f_7;f_6;f_4;f_{14}$)
	f_{20}	混合函数 4($f_{12};f_3;f_{13};f_8$)
	f_{21}	混合函数 5($f_{14};f_{12};f_4;f_9;f_1$)
	f_{22}	混合函数 6($f_{10};f_{11};f_{13};f_9;f_5$)
组合函数	f_{23}	组合函数 1($f_4;f_1;f_2;f_3;f_1$)
	f_{24}	组合函数 2($f_{10};f_9;f_4$)
	f_{25}	组合函数 3($f_{11};f_9;f_1$)
	f_{26}	组合函数 4($f_{11};f_{13};f_1;f_6;f_7$)
	f_{27}	组合函数 5($f_{14};f_9;f_{11};f_6;f_1$)
	f_{28}	组合函数 6($f_{15};f_{13};f_{11};f_{16};f_1$)
	f_{29}	组合函数 7($f_{17};f_{18};f_{19}$)
	f_{30}	组合函数 8($f_{20};f_{21};f_{22}$)

4.2 实验环境和参数设置

本实验使用的硬件环境为:Intel Core i7-4500 M 处理器 8GB DDR3 内存。30 个测试函数设定为 50 维。算法运行的终止条件对于评价算法性能的优劣尤为关键,这里使用最大适应度评估次数作为判断是否终止的条件。当达到最大适应度评估次数后,算法取得的函数最值可以较好地反映算法的性能。同时,对于每个测试函数,4 种算法各自独立运行 50 次取得均值,以此反映各算法的平均性能,从而大大降低实验偶然性对结果的影响。

经过实验发现,pr-DE 和 Dapr-DE 的 $NP_{\text{init}}=200$,$p_{\max}=3$ 时得到的结果最好。为了公平比较,将 4 种方法的 NP_{init}都设为 200。

对于参数 F 和 CR,有很多控制方法,文献[23-24]充分讨论了参数应如何设置以发挥其作用。同文献[23-24]的设置一样,本文讨论的 4 种算法均初始化为 $F=0.5$,$CR=0.9$,并采用自适应方法对其进行控制,具体实现如下:

$$F_i^{(G+1)}=\begin{cases}F_l+rand_1*F_u, & \text{if } rand_2<\tau_1\\ F_i^{(G)}, & \text{otherwise}\end{cases} \tag{16}$$

$$CR_i^{(G+1)}=\begin{cases}rand_3, & \text{if } rand_4<\tau_2\\ CR_i^{(G)}, & \text{otherwise}\end{cases} \tag{17}$$

其中，$rand_j(j\in\{1,2,3,4\})$是处于[0,1]之间的随机数，$\tau_1=\tau_2=F_l=0.1$，$F_u=0.9$。

4.3 实验结果与分析

表 2 列出了单峰、多峰、混合和组合函数的实验结果，其中 best 和 worst 分别表示 50 次实验中最优解中出现的最小值和最大值；mean 则表示 50 次实验最优解的平均值；std 是标准差，代表了算法的稳定性。表中 best 和 mean 中的加粗数据代表在 4 种算法中取得了最好的结果。

表 2　4 种方法在 30 个测试函数上的实验结果

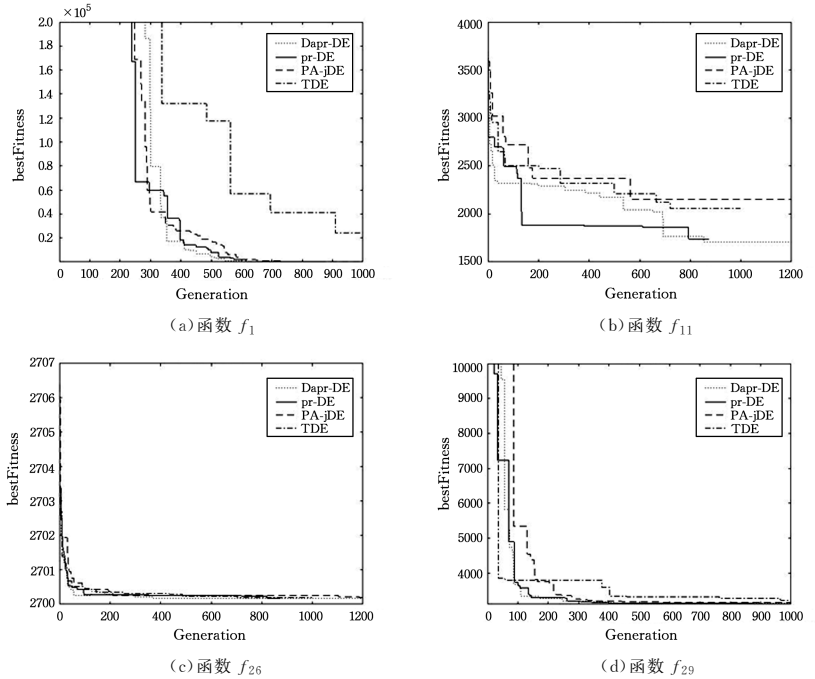
优化函数		Dapr-DE	pr-DE	PA-jDE	TDE
f_1	best	1.00000647E+02	1.07432680E+02	**1.00000012E+02**	1.15235160E+03
	worst	3.69462996E+02	6.29705613E+02	5.10009378E+03	5.83849140E+04
	mean	**1.08430195E+02**	1.82163147E+02	1.75965956E+02	2.95761829E+04
	std	3.78758704E+01	8.09418985E+01	2.99590499E+02	1.04921389E+04
f_2	best	**2.00000000E+02**	**2.00000000E+02**	**2.00000000E+02**	2.00000002E+02
	worst	2.00000000E+02	2.00000002E+02	2.00000000E+02	2.00000021E+02
	mean	**2.00000000E+02**	**2.00000000E+02**	**2.00000000E+02**	2.00000007E+02
	std	0.00000000E+00	4.06962940E−07	0.00000000E+00	4.07886877E−06
f_3	best	**3.00000000E+02**	**3.00000000E+02**	**3.00000000E+02**	**3.00000000E+02**
	worst	3.00000000E+02	3.00000001E+02	3.00000000E+02	3.00000002E+02
	mean	**3.00000000E+02**	**3.00000000E+02**	**3.00000000E+02**	3.00000001E+02
	std	0.00000000E+00	5.16024815E−10	0.00000000E+00	1.75964233E−08
f_4	best	**4.00000000E+02**	4.00000060E+02	**4.00000000E+02**	4.00189982E+02
	worst	4.34780275E+02	4.34780275E+02	4.34780275E+02	4.34780275E+02
	mean	**4.02215917E+02**	4.02475149E+02	4.15009846E+02	4.11314905E+02
	std	8.26579635E+00	8.25898479E+00	1.67710523E+01	1.58779301E+01
f_5	best	5.18247243E+02	5.17837761E+02	**5.00000000E+02**	5.19957603E+02
	worst	5.20293405E+02	5.20218571E+02	5.20325154E+02	5.20323703E+02
	mean	5.20104470E+02	5.20066365E+02	**5.18172266E+02**	5.20227787E+02
	std	3.36382394E−01	3.90883368E−01	6.05157983E+00	7.40545335E−02
f_6	best	**6.00000000E+02**	**6.00000000E+02**	**6.00000000E+02**	**6.00000000E+02**
	worst	6.00011081E+02	6.00006163E+02	6.00894543E+02	6.00037448E+02
	mean	**6.00000237E+02**	6.00000426E+02	6.00035080E+02	6.00000753E+02
	std	1.55498370E−03	1.13542631E−03	1.75365544E−01	5.24138735E−03
f_7	best	**7.00000000E+02**	7.00009861E+02	**7.00000000E+02**	7.00097275E+02
	worst	7.00159226E+02	7.00255292E+02	7.00073884E+02	7.00359758E+02
	mean	7.00039710E+02	7.00121752E+02	**7.00026787E+02**	7.00316938E+02
	std	4.06415707E−02	4.77892216E−02	2.13010386E−02	5.94453219E−02
f_8	best	**8.00000000E+02**	**8.00000000E+02**	**8.00000000E+02**	**8.00000000E+02**
	worst	8.00000000E+02	8.00000000E+02	8.01989918E+02	8.00000542E+02
	mean	**8.00000000E+02**	**8.00000000E+02**	8.00214599E+02	8.00000030E+02
	std	0.00000000E+00	0.00000000E+00	4.58706361E−01	1.52408649E−04
f_9	best	**9.00994959E+02**	9.07173717E+02	**9.00994959E+02**	9.09365029E+02
	worst	9.15719387E+02	9.17148543E+02	9.13929417E+02	9.22658566E+02
	mean	9.06967865E+02	9.12399762E+02	**9.04916269E+02**	9.17426715E+02
	std	3.69471942E+00	2.14212714E+00	2.52030862E+00	2.93954169E+00
f_{10}	best	**1.00000000E+03**	1.00021238E+03	**1.00000000E+03**	1.04090259E+03
	worst	1.00037473E+03	1.00791999E+03	1.01182953E+03	1.17840676E+03
	mean	**1.00015920E+03**	1.00182911E+03	1.00105830E+03	1.10339574E+03
	std	9.20551559E−02	1.47871494E+00	2.42384854E+00	2.92354712E+01
f_{11}	best	1.15169316E+03	1.39618514E+03	**1.10366478E+03**	1.60290053E+03
	worst	1.94496903E+03	2.00568863E+03	1.67243636E+03	2.31141923E+03
	mean	1.60897673E+03	1.73905982E+03	**1.24385643E+03**	2.00862937E+03
	std	1.78693186E+02	1.33530822E+02	1.37398822E+02	1.50550895E+02
f_{12}	best	1.20025082E+03	1.20021320E+03	**1.20000077E+03**	1.20044282E+03
	worst	1.20087756E+03	1.20075996E+03	1.20057185E+03	1.20086927E+03
	mean	**1.20008786E+03**	1.20054588E+03	1.20010783E+03	1.20065747E+03
	std	1.27455540E−01	1.11379650E−01	1.35593560E−01	1.09850690E−01
f_{13}	best	**1.30008893E+03**	1.30006686E+03	1.30015680E+03	1.30010276E+03
	worst	1.30024733E+03	1.30022467E+03	1.30009634E+03	1.30022433E+03
	mean	**1.30010924E+03**	1.30016065E+03	1.30011459E+03	1.30017638E+03
	std	2.84082816E−02	3.21964029E−02	3.58372667E−02	2.54044768E−02
f_{14}	best	**1.40008779E+03**	1.40010916E+03	1.40004856E+03	1.40010464E+03
	worst	1.40031315E+03	1.40024153E+03	1.40021932E+03	1.40024144E+03
	mean	**1.40010121E+03**	1.40017054E+03	1.40012775E+03	1.40016933E+03
	std	4.87212345E−02	3.06430833E−02	3.96044150E−02	3.00310502E−02
f_{15}	best	1.50094766E+03	1.50097723E+03	**1.50040386E+03**	1.50130926E+03
	worst	1.50209163E+03	1.50207429E+03	1.50214074E+03	1.50229973E+03
	mean	1.50157087E+03	1.50147648E+03	**1.50094771E+03**	1.50183072E+03
	std	2.71671276E−01	2.29976423E−01	3.11648383E−01	2.93466115E−01

（续表）

优化函数		Dapr-DE	pr-DE	PA-jDE	TDE
f_{16}	best	1.60071064E+03	1.60178172E+03	**1.60047925E+03**	1.60231546E+03
	worst	1.60312349E+03	1.60285149E+03	1.60277355E+03	1.60311502E+03
	mean	**1.60150436E+03**	1.60253317E+03	1.60164648E+03	1.60275946E+03
	std	2.80167198E−01	2.37609043E−01	4.58196524E−01	1.84216136E−01
f_{17}	best	**1.70020036E+03**	1.70884330E+03	1.70020814E+03	1.73588448E+03
	worst	1.81990639E+03	1.81423462E+03	1.79525729E+03	1.88909134E+03
	mean	**1.71688696E+03**	1.74649110E+03	1.71701285E+03	1.81773186E+03
	std	2.62063105E+01	2.05673477E+01	2.26620095E+01	2.91232632E+01
f_{18}	best	**1.80007775E+03**	1.80040349E+03	1.80013159E+03	1.80230400E+03
	worst	1.80521688E+03	1.80477643E+03	1.80603291E+03	1.80647647E+03
	mean	**1.80160615E+03**	1.80246841E+03	1.80185583E+03	1.80388715E+03
	std	1.23157454E+00	1.05959780E+00	1.17318950E+00	9.48875732E−01
f_{19}	best	1.90010653E+03	1.90031650E+03	**1.90000740E+03**	9.00349920E+02
	worst	1.90116217E+03	1.90085280E+03	1.90036808E+03	1.90101340E+03
	mean	1.90066284E+03	1.90058126E+03	**1.90013441E+03**	1.90066551E+03
	std	2.54478738E−01	1.42520110E−01	1.03951614E−01	1.84454717E−01
f_{20}	best	2.00000148E+03	2.00011795E+03	**2.00000013E+03**	2.00030960E+03
	worst	2.00136593E+03	2.00114654E+03	2.00219865E+03	2.00116948E+03
	mean	**2.00030753E+03**	2.00041646E+03	2.00044115E+03	2.00065301E+03
	std	3.81391778E−01	2.25716650E−01	5.32518305E−01	2.08165064E−01
f_{21}	best	**2.10000002E+03**	2.10016469E+03	2.10000009E+03	2.10191252E+03
	worst	2.10202094E+03	2.10319211E+03	2.15860420E+03	2.10914750E+03
	mean	**2.10034635E+03**	2.10090998E+03	2.10469612E+03	2.10391927E+03
	std	3.74158486E−01	5.41774374E−01	1.41482472E+01	1.42288642E+00
f_{22}	best	**2.20000000E+03**	2.20061157E+03	2.20000001E+03	2.20003884E+03
	worst	2.20039215E+03	2.20096262E+03	2.21708970E+03	2.20026048E+03
	mean	**2.20008716E+03**	2.20056848E+03	2.20118966E+03	2.20009082E+03
	std	1.24997520E−01	3.40480990E−01	3.99166417E+00	6.91339762E−02
f_{23}	best	**2.62945747E+03**	**2.62945747E+03**	**2.62945747E+03**	2.62945747E+03
	worst	2.62945748E+03	2.62945748E+03	2.62945748E+03	2.62945747E+03
	mean	**2.62945747E+03**	2.62945748E+03	**2.62945747E+03**	2.62945747E+03
	std	9.18524503E−13	9.18740844E−13	9.18544623E−13	9.18544623E−13
f_{24}	best	**2.50000000E+03**	2.50000033E+03	2.50000021E+03	2.51600758E+03
	worst	2.52236875E+03	2.52355856E+03	2.52355856E+03	2.53220843E+03
	mean	**2.51142921E+03**	2.51156417E+03	2.51156417E+03	2.52540896E+03
	std	5.15797898E+00	4.07887669E+00	4.07887669E+00	3.13508083E+00
f_{25}	best	2.61528089E+03	2.60837595E+03	**2.60000000E+03**	2.64600970E+03
	worst	2.65747230E+03	2.72464141E+03	2.70160507E+03	2.70103805E+03
	mean	**2.63812186E+03**	2.66899960E+03	2.66031315E+03	2.66985924E+03
	std	8.66985174E+00	6.78622462E+00	3.40684604E+01	1.31810236E+01
f_{26}	best	**2.70008173E+03**	2.60595146E+03	2.70004677E+03	2.70006307E+03
	worst	2.70022784E+03	2.64059294E+03	2.70022548E+03	2.70024380E+03
	mean	2.70016913E+03	2.70016946E+03	**2.70010915E+03**	2.70019089E+03
	std	3.36565643E−02	3.21354846E−02	3.13874073E−02	3.39928039E−02
f_{27}	best	**2.70104353E+03**	2.70011132E+03	2.70101970E+03	2.70183793E+03
	worst	3.00637042E+03	2.70023882E+03	3.10030701E+03	3.00004920E+03
	mean	2.70840119E+03	**2.70275090E+03**	2.91447766E+03	2.76688442E+03
	std	4.25612156E+01	4.28561262E−01	1.69021391E+02	1.23487165E+02
f_{28}	best	**3.15682688E+03**	3.15682689E+03	3.15682698E+03	3.15682703E+03
	worst	3.19946412E+03	3.16006121E+03	3.31860335E+03	3.17620418E+03
	mean	3.15946505E+03	**3.15739768E+03**	3.19979004E+03	3.15795120E+03
	std	6.70031754E+00	1.24523484E+00	5.06963505E+01	3.29247889E+00
f_{29}	best	**3.01410663E+03**	3.06572218E+03	3.04892149E+03	3.15859357E+03
	worst	3.12890872E+03	3.12628056E+03	3.19160845E+03	3.26364320E+03
	mean	**3.11359402E+03**	3.12040376E+03	3.12524762E+03	3.20787628E+03
	std	2.20487837E+01	1.06079472E+01	2.41647485E+01	2.48343370E+01
f_{30}	best	3.46679271E+03	3.46951339E+03	**3.29053808E+03**	3.60229280E+03
	worst	3.56959981E+03	3.56430824E+03	3.74169640E+03	3.74535033E+03
	mean	**3.49901475E+03**	3.51131878E+03	3.52760482E+03	3.64316128E+03
	std	2.32807159E+01	2.06684507E+01	7.93033358E+01	4.55378934E+01

3 个单峰函数存在着螺旋陷阱，多峰、混合和组合函数存在大量的局部最优解，这都使算法的寻优能力面临巨大的考验。结果显示，在 21 个函数（f_1，f_2，f_3，f_4，f_6，f_8，f_{10}，f_{12}，f_{13}，f_{14}，f_{16}，f_{17}，f_{18}，f_{20}，f_{21}，f_{22}，f_{23}，f_{24}，f_{25}，f_{29}，f_{30}）中，Dapr-DE 的平均值是最好的，在 5 个函数（f_7，f_9，f_{11}，f_{26}，f_{27}）中，Dapr-DE 的平均值是第二好的；在余下的 4 个函数（f_5，f_{15}，f_{19}，f_{28}）中，Dapr-DE 的平均值是第三好的。此外，pr-DE 在函数（f_2，f_3，f_8，f_{27}，f_{28}）中取得最好的平均值，而 PA-jDE 在函数（f_2，f_3，f_5，f_7，f_9，f_{11}，f_{15}，f_{19}，f_{23}）取得最好的平均值。

限于篇幅，这里只选取了 4 幅收敛速率对比图，如图 1 所示。4 个函数分别是 1 个单峰函数 f_1、1 个多峰函数 f_{11}、2 个组合函数 f_{26} 和 f_{29}。由图中可知，TDE 算法的收敛速率是所有算法中最快的，但是该算法在陷入局部最优解之后就一直没有跳出。其他 3 种算法在 4 个函数上的表现相差不大，收敛速率比较接近，同时在遇到局部收敛时也能够尽快地调整并跳出陷阱。最后，算法 Dapr-DE 在图 1 的 4 个函数上达到最优解时的收敛代数均稍高于其他 3 种算法，但是最终达到的收敛状态均显示是最好的。此外，由于加入了种群熵的计算和删除操作来进行聚类划分以确定最终删除的个体，Dapr-DE 比起其他算法需要更多的运行时间。但因为种群缩减的次数相较于进化代数小很多，使得这一部分时间花费占 Dapr-DE 算法总运行时间的比例较低，并未给算法带来巨大的计算负担。



(a)函数 f_1　(b)函数 f_{11}

(c)函数 f_{26}　(d)函数 f_{29}

图 1　4 种算法在 4 个函数上的收敛速率比较

综上，对于单峰、多峰、混合和组合 4 种函数类型，Dapr-DE 的总体性能在 4 种比较算法中是最好的，这表明 Dapr-DE 克服陷阱寻找最优解的能力较强，对于不同类型的问题也有着较好的适用性，这主要归因于引入的参数熵可以很好地反映种群的多样性，帮助算法在恰当的时候进行种群大小缩减，从而取得不错的进化结果。但在一些函数上，Dapr-DE 表现稍劣于 PA-jDE，这主要是由于种群大小的缩减减弱了算法的探测性能，一定程度上阻碍了算法跳出局部最优陷阱。在一些函数上，Dapr-DE 表现稍劣于 pr-DE，这主要是由于为了保证种群多样性而减弱了算法的收敛能力。

结束语　本文提出了一种新的差分进化算法——基于种群多样性的可变种群规模进化算法 Dapr-DE，其根据种群多样性来调整每一次删减的个体数量；同时，使用聚类对删除作出进一步的优化，使得种群中的个体分布更加均匀，从而提高了算法的鲁棒性。这些操作改善了以往差分进化算法存在的不足。此外，使用 CEC 14 函数测试集进行实验，得到的结果也证实了 Dapr-DE 有着较好的算法性能。不过，Dapr-DE 也存在一些不足，如由于种群大小的缩减导致探测种群解空间能力的减弱和为保证种群多样性而减弱了算法的收敛能力，如何克服这些不足是我们未来工作的重点部分。此外，将算法用于解决一些实际问题，如 TSP 问题，也是我们未来工作的一个方向。

参 考 文 献

[1] STORN R，PRICE K. Differential Evolution－a simple and efficient adaptive scheme for global optimization over continuous spaces：Technical Report：TR-95-012[R]. 1995.

[2] 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.

[3] MOIRANGTHEM J，KRISHNANAND K R，DASH S S，et al. Adaptive differential evolution algorithm for solving non-linear coordination problem of directional overcurrent relays[J]. IET Generation，Transmission & Distribution，2013，7(4)：329-336.

[4] WANG B C，LI H X，LI J P，et al. Composite Differential Evolution for Constrained Evolutionary Optimization[J]. IEEE Transactions on Systems，Man，and Cybernetics：Systems，2018，99：1-14.

[5] LAI J C，LEUNG F H，LING S H. A New Differential Evolution with self-terminating ability using fuzzy control and k-nearest neighbors[C]// IEEE Congress on Evolutionary Computation. 2010：1-8.

[6] LI S，SUN W. Optimization of fuzzy control rules based on differential evolution algorithm[C]// Proceedings of 2014 IEEE Chinese Guidance，Navigation and Control Conference. 2014：2610-2613.

[7] FENG X，SANDERSON A C，BONISSONE P P，et al. Fuzzy Logic Controlled Multi-Objective Differential Evolution[C]// The 14th IEEE International Conference on Fuzzy Systems. 2005：720-725.

[8] LIU L，YIN J T，ZHOU P T. Fault diagnosis of power transformer based on improved differential evolution-neural network

[C]// 2013 IEEE 4th International Conference on Electronics Information and Emergency Communication. 2013:238-241.

[9] BHATIA S, VISHWAKARMA V P. Feed forward neural network optimization using self adaptive differential evolution for pattern classification[C]// 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). 2016:184-188.

[10] GOLDBERG D E. Genetic Algorithms in Search, Optimization, and Machine Learning[R]. Addison-Wesley, Reading, MA, 1989.

[11] MENGSHOEL O J, GOLDBERG D E. The crowding approach to niching in genetic algorithms[J]. Evolutionary Computation, 2008, 16(3):315-354.

[12] SHI E C, LEUNG F H F, BONNIE N F. Differential Evolution with adaptive population size[C]// Law 2014 19th International Conference on Digital Signal Processing. 2014:876-881.

[13] BREST J, ZAMUDA A, FISTER I, et al. Large scale global optimization using self-adaptive differential evolution algorithm[C]// Evolutionary Computation (CEC). 2010:1-8.

[14] RONKKONEN J, KUKKONEN J, PRICE K V. Real-Parameter Optimization with Differential Evolution[C]// The 2005 IEEE Congresson Evolutionary Computation. 2005:506-513.

[15] PRICE K V, STORN R M, LAMPINEN J A. Differential Evolution, A Practical Approach to Global Optimization[R]. Springer, 2005.

[16] FEOKTISTOV V. Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)[R]. Springer-Verlag NewYork, Inc., Secaucus, NJ, USA, 2006.

[17] VOLKOVAS R, FAIRBANK M, PEREZ-LIEBANA D. Diversity maintenance using a population of repelling random-mutation hill climbers[C]// 2017 9th Computer Science and Electronic Engineering (CEEC). 2017:37-42

[18] CHEN L. An Adaptive Genetic Algorithm Based on Population Diversity Strategy[C]// 2009 Third International Conference on Genetic and Evolutionary Computing. 2009:93-96.

[19] WAGSTAFF K, CARDIE C. Constrained K-means clustering with background knowledge[C]// The Eighteenth International Conference on Machine Learning. 2001:577-584.

[20] FREY B J, DUECK D. Clustering by Passing Messages Between Data Points[J]. Science, 2007, 315:972-976.

[21] CUI X X, LI M, FANG T J. Study of population diversity of multiobjective evolutionary algorithm based on immune and entropy principles[C]// Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat). 2001:1316-1321.

[22] LIANG J J, QU B Y, SUGANTHAN P N. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective realparameter numerical optimization[R]. Tech. Rep. 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China, 2014.

[23] LIU J, LAMPINEN J. A Fuzzy Adaptive Differential Evolution Algorithm[J]. Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2005, 9(6):448-462.

[24] ZHANG X, YE Z W, YANG J, et al. An approach for learning the optimal "tuned" masks based on differential evolution algorithm[C]// 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC). 2017:585-590.

(上接第 148 页)

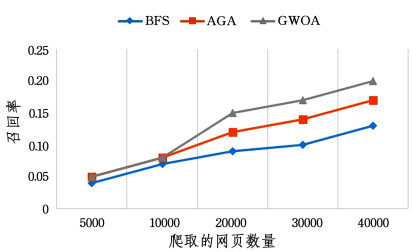


图 5 不同搜索方法下网页的召回率

结束语 基于灰狼算法的主题爬虫能够有效地提高抓取网页的准确率和召回率，提高搜索的精度。但是灰狼算法还存在迭代过程中容易限于局部最优无法完全覆盖网页的情况。改进灰狼算法的位置函数，加入合适的适应度函数使其具有自适应性，从而进一步提升算法性能并运用到主题爬虫中将是下一步的工作。

参 考 文 献

[1] CHO J, GARCIA-MOLINA H, PAGE L. Efficient crawlingthrough URL ordering[J]. Computer Networks and ISDN Systems, 1998, 30(1):161-172.

[2] KLEINBERG J M. Authoritative sources in a hyperlinked environment[J]. Journal of the ACM (JACM), 1999, 46(5):604-632.

[3] HERSEOVICI M, JACOV M, MAREK Y S. The Shark-search algorithm an application: Tailored Web site mapping[J]. Computer Networks and ISDN Systems, 1998, 23(1):41-58.

[4] 杨小平，丁浩，黄都培. 基于向量空间模型的中文信息检索技术研究[J]. 计算机工程与运用，2003，15：109-111.

[5] 邹永斌，陈兴蜀，王文贤. 基于贝叶斯分类器的主题爬虫研究[J]. 计算机应用研究，2009，26(9)：3418-3420，3439.

[6] 李璐，张国印，李正文. 基于 SVM 的主题爬虫技术研究[J]. 计算机科学，2015，42(2)：118-122.

[7] 张莉婧，曾庆涛，李业丽，等. 面向图书主题的爬虫算法研究[J]. 计算机科学，2017，44(11)：460-463

[8] 陈黎，李志易，琚生根，等. 基于 SVM 预测的金融主题爬虫[J]. 四川大学学报(自然科学版)，2010，47(3)：493-497.

[9] MIRJALILI S, MIRJALILI S M, LEWIS A. Grey wolf optimization[J]. Advances in Engineering Software, 2014, 69(7):46-61.

[10] 魏政磊，赵辉，韩邦杰，等. 具有自适应搜索策略的灰狼优化算法[J]. 计算机科学，2017，44(3)：259-263.

[11] 刘国靖，康丽，罗长寿，等. 基于遗传算法的主题爬虫策略[J]. 计算机应用，2007，27(12)：172-176.

[12] 张海亮，袁道华. 基于遗传算法的主题爬虫[J]. 计算机技术与发展，2012，22(8)，48-52.

[13] 郭振洲，刘然，拱长青，等. 基于灰狼算法的改进研究[J]. 计算机应用研究，2017，34(12)：3603-3606.

[14] 荆文鹏，王育坚，董伟伟. 自适应遗传算法在主题爬虫搜索略中的应用研究[J]. 计算机科学，2016，43(8)：254-257.