Journal of Computer Applications
计算机应用,2013,33(3):796-799,805
ISSN 1001-9081
CODEN JYIIDU
2013-03-01
http://www.joca.cn

文章编号:1001-9081(2013)03-0796-04
doi:10.3724/SP.J.1087.2013.00796

基于混沌理论的动态种群萤火虫算法

冯艳红[1*],刘建芹[2],贺毅朝[1]
(1.石家庄经济学院 信息工程学院,石家庄 050031; 2.石家庄信息工程职业学院 国际教育部,石家庄 050035)
(*通信作者电子邮箱 qinfyh@163.com)

摘 要:针对萤火虫算法在全局寻优搜索中收敛速度慢、求解精度低,易陷入局部极值区域等缺陷,提出一种基于混沌理论的动态种群萤火虫算法。首先,该算法采用立方映射产生的混沌序列对萤火虫位置进行初始化,为全局搜索的多样性奠定基础;其次,通过对种群的动态监测,每当算法满足预设条件时,基于混沌序列生成部分新的个体,以提高算法的收敛速度;最后,对每一代产生的全局最优解,适时采用高斯扰动进行变异操作,使算法更具有跳出局部极小的能力。通过对6个复杂Benchmark函数进行测试,实验结果表明,该算法提高了全局搜索能力、收敛速度和解的精度。

关键词:萤火虫算法;混沌;立方映射;函数优化
中图分类号:TP301.6 **文献标志码**:A

Chaos-based dynamic population firefly algorithm

FENG Yanhong[1*], LIU Jianqin[2], HE Yichao[1]
(1. *School of Information Engineering, Shijiazhuang University of Economics, Shijiazhuang Hebei* 050031, *China*;
2. *Department of International Education, Shijiazhuang Information Engineering Vocational College, Shijiazhuang Hebei* 050035, *China*)

Abstract: The Firefly Algorithm (FA) has a few disadvantages in the global searching, including slow convergence speed, low solving precision and high possibility of being trapped in local optimum. A FA based on chaotic dynamic population was proposed. Firstly, chaotic sequence generated by cube map was used to initiate individual position, which strengthened the diversity of global searching; secondly, through dynamic monitoring of population, whenever the algorithm meets the preset condition, the new population individuals were generated using chaotic sequences, thus effectively improving convergence speed; thirdly, a Gaussian disturbance would be given on the global optimum of each generation, thus the algorithm could effectively jump out of local minima. Based on six complex test functions, the test results show that chaos-based dynamic population FA improves the capacity of global searching optimal solution, convergence speed and computational precision of solution.

Key words: Firefly Algorithm (FA); chaos; cube mapping; function optimization

0 引言

萤火虫算法(Firefly Algorithm,FA)是剑桥学者Yang于2008年提出的一种新颖的进化算法,其思想源于自然界中萤火虫的发光行为,即通过萤火虫总是朝向更亮的区域飞去实现进化。目前,该算法已引起国内外学者的广泛关注,成为计算智能研究领域的一个新的研究热点。随着对FA研究的不断深入,目前该算法已在数值优化、工程技术、资源管理等领域得到应用并取得了良好效果[1-7]。

尽管该算法已经在诸多领域得以应用,但是算法本身存在着对于初始解分布的依赖性、后期收敛速度慢、易于停滞、早熟和求解精度低等缺陷。由于FA提出时间相对较短,只有少数学者对其进行改进。Lukasik等[2]在2009年对FA进行了改进,并对算法的参数进行研究;虽然改进后的FA提高了求解精度,但求解速度较慢。Apostolopoulos等[5]在2011年应用FA成功求解多目标优化问题中的经济排放负荷调度问题,拓展了FA的应用领域。Chai-Ead等[6]应用FA求解带有噪声的非线性函数优化问题,并简化了算法参数的设置。刘长平和叶春明在2011年9月将FA应用于组合优化问题中的置换流水线调度问题[7],开启了应用FA求解组合优化问题的研究。

最近,各种混沌优化算法[8,10-12]不断被提出,这些算法的基本思想是把变量从混沌空间变换到解空间,然后利用混沌变量具有遍历性、随机性和规律性的特点进行搜索,混沌优化方法具有全局渐进收敛、易跳出局部极小点和收敛速度快等特点,非常适于与进化算法相结合。受此启发,本文提出一种基于混沌理论的动态种群萤火虫算法(Chaos-based Dynamic Population Firefly Algorithm, CDPFA),该算法不仅在对萤火虫位置初始化过程中使用混沌序列,而且在算法迭代过程中使用混沌策略周期性地向种群中添加新的萤火虫个体,有效地避免了算法的早熟现象和种群多样性较低等缺点,进一步提高了算法的寻优精度和求解速度。

1 基本萤火虫算法

萤火虫算法是模拟自然界中萤火虫的发光行为构造出的随机优化算法,算法舍弃了萤火虫发光的一些生物学意义,只

收稿日期:2012-09-18;**修回日期**:2012-10-25。 **基金项目**:河北省高等学校科学技术研究项目(Z2011143);河北省科学技术研究与发展计划项目(11213593);河北省自然科学基金资助项目(F2012208016);河北省高等学校科学研究项目(Q2012153)。

作者简介:冯艳红(1978-),女,河北卢龙人,讲师,硕士,主要研究方向:人工智能; 刘建芹(1966-),女,河北灵寿人,副教授,主要研究方向:智能算法; 贺毅朝(1969-),男,河北晋州人,教授,CCF会员,主要研究方向:智能计算、算法理论、计算机密码学。

利用其发光特性来根据其搜索区域寻找伙伴,并向位置较优的萤火虫移动,从而实现位置进化。对于 FA,通常作如下三个假设:1)萤火虫不分雌雄,发光较强的萤火虫可以吸引其他发光较弱的萤火虫。2)每个萤火虫的吸引度 β 正比于它的发光强度 I;对于任意两只发光的萤火虫,发光较弱的萤火虫会朝向发光较强的萤火虫移动,并且吸引度和发光强度随着两只萤火虫之间距离 R 的变大而变小;最亮的萤火虫(即 I 值最大的萤火虫)是随机飞行的。3)萤火虫的发光强度 I 与目标函数值有关。即对于 $\min f(\boldsymbol{X})$,$\boldsymbol{X}=(x_1,x_2,\cdots,x_d)$,其中 $x_i\in[L_i,U_i]$,$1\leqslant i\leqslant d$,且 $I(\boldsymbol{X})\propto -f(\boldsymbol{X})$。下面给出基本萤火虫算法(FA)的算法描述。

步骤 1　初始化。

在可行域中随机放置 m 个萤火虫,并赋予每个萤火虫的最大吸引度为 β_0,吸收系数为 γ,随机步长大小为 α,迭代次数为 $MaxT$。

步骤 2　确定萤火虫 i 与萤火虫 j 之间的距离:

$$R_j=\|\boldsymbol{X}_i-\boldsymbol{X}_j\|=\sqrt{\sum_{k=1}^{d}(x_{i,k}-x_{j,k})^2} \tag{1}$$

其中 $\|\cdot\|$ 为欧氏范数。

步骤 3　更新萤火虫 i 的吸引度。

在 FA 中,每只萤火虫均拥有各自的视线范围,萤火虫的吸引度 β 定义为距离 $R_j=d(\boldsymbol{X}_i,\boldsymbol{X}_j)$ 的单调递减函数。由于萤火虫发出的荧光在介质中传播时会减弱,定义 γ 为荧光吸收率。

$$\beta(\gamma)=\beta_0*\exp(-\gamma R_j) \tag{2}$$

其中:β_0 表示 $R_j=0$ 时的吸引度,$\beta_0\in[0,1]$,$\gamma\in[0,10]$。

步骤 4　更新萤火虫 i 的位置。

萤火虫 i 被更亮的萤火虫 j 吸引而发生位置移动:

$$\boldsymbol{X}_i(t+1)=\boldsymbol{X}_i(t)+\beta_0*\exp(-\gamma R_j)*(\boldsymbol{X}_j(t)-\boldsymbol{X}_i(t))+\alpha*(r_1-0.5)*\boldsymbol{X}_M \tag{3}$$

式(3)的第一部分反映萤火虫当前位置的影响,起到了平衡全局和局部搜索的能力;第二部分反映群体信息的影响,即萤火虫之间的信息共享;第三部分设置随机步长,避免萤火虫陷入局部最优。其中,$\boldsymbol{X}_i(t)$ 是第 i 个萤火虫个体在第 t 代时的位置向量,$\alpha\in[0,1]$,$r_1\sim U(0,1)$。若 $r_1<0.5$,则 $\boldsymbol{X}_M$ 取值 $\boldsymbol{X}_{\max}-\boldsymbol{X}_i(t)$;否则,$\boldsymbol{X}_M$ 取值 $\boldsymbol{X}_i(t)-\boldsymbol{X}_{\min}$。其中,$\boldsymbol{X}_{\max}$ 代表目标函数搜索空间的上限,$\boldsymbol{X}_{\min}$ 代表目标函数搜索空间的下限。

步骤 5　最亮的萤火虫随机飞行。

$$\boldsymbol{X}_B(t+1)=\boldsymbol{X}_B(t)+\alpha(r_2-0.5)*\boldsymbol{X}_M \tag{4}$$

其中:$\boldsymbol{X}_B(t)$ 是种群进化到第 t 代的全局最优位置,r_2 为区间(0,1)内的随机数,即 $r_2\sim U(0,1)$。若 $r_2<0.5$,则 $\boldsymbol{X}_M$ 取值 $\boldsymbol{X}_{\max}-\boldsymbol{X}_B(t)$;否则,$\boldsymbol{X}_M$ 取值 $\boldsymbol{X}_B(t)-\boldsymbol{X}_{\min}$。

步骤 6　如果达到给定的精度或者迭代次数或者运行时间,则停止;否则,$t=t+1$,转步骤 2。

2　基于混沌的动态种群萤火虫算法

2.1　混沌序列初始化萤火虫位置

作为一种新颖的优化技术,混沌优化方法在近十年得到了学术界和工程界的广泛关注,并在工程实践中得到了应用。对于数值优化问题,在利用混沌优化方法求解时,通过将搜索过程对应为混沌轨道的遍历过程,可使搜索过程具有避免陷入极小的能力,并最终获得全局最优解或者较好的满意解[8]。

目前,FA 几乎全部采用随机方式对萤火虫的位置进行初始化,这种方式有可能造成萤火虫分布位置的不均匀。注意到混沌具有随机性、规律性、有界性等特点,它对初值是极度敏感的,而且能够按其自身规律在一定范围内不重复地遍历所有状态,许多学者提出了嵌入混沌序列的多种改进算法[8,10-12]。因此,本文采用混沌序列初始化萤火虫位置,以加强种群的多样性,为进一步有效地进行全局搜索建立基础。本文并未采用许多混沌模型中最常用的 logistic 映射来初始化种群,而是使用立方映射[11],这是因为尽管 logistic 映射和立方映射产生的序列都是混沌的,但是立方映射产生的序列均匀性较 logistic 映射更好,这一点在文献[11]中已经证明。立方映射的表达式如下:

$$y(n+1)=4y(n)^3-3y(n);\ -1\leqslant y(n)\leqslant 1;\ n=0,1,2,\cdots \tag{5}$$

在文献[9]中,给出了利用立方映射产生混沌序列的优化过程所包括的两个关键步骤即:1)将混沌空间映射到优化问题的解空间;2)利用混沌动态特性来实现对解空间的搜索。根据这两个关键步骤,提出一种将混沌空间映射到优化问题解空间(Chaotic Space Mapping to Search Space, CMS)的搜索算法:

步骤 1　对于 D 维空间中的 M 个萤火虫个体,首先随机产生一个 D 维向量,作为第一个萤火虫个体,即 $\boldsymbol{Y}=(y_1,y_2,\cdots,y_d)$,其中 $y_i\in[-1,1]$,$1\leqslant i\leqslant d$。

步骤 2　将 $\boldsymbol{Y}$ 的每一维利用式(5)进行 $M-1$ 次迭代,这样就产生了其余的 $M-1$ 个萤火虫个体。

步骤 3　将产生的混沌变量按式(6)映射到解的搜索空间:

$$x_{id}=L_d+(1+y_{id})\frac{U_d-L_d}{2} \tag{6}$$

其中:U_d 和 L_d 分别表示搜索空间第 d 维的上下限,y_{id} 是利用式(5)产生的第 i 个萤火虫的第 d 维,x_{id} 即为第 i 个萤火虫在搜索空间第 d 维的坐标。

2.2　基于混沌理论的动态种群萤火虫算法

为了避免 FA 在迭代过程中出现停滞现象,即萤火虫过多地朝着某一固定方向飞行,从而增大算法陷于局部极值的可能。以下在 FA 的基础上,本节采用三种方式对算法进行改进。

方式 1　提出一种基于混沌序列初始化萤火虫位置算法(Chaotic Initialization Firefly Algorithm, CFA)。显然,CFA 的实现步骤与 FA 基本一致,仅仅在步骤 1 的初始化中,利用式(5)产生 m 个萤火虫个体,再将这 m 个萤火虫的位置应用式(6)映射到种群的搜索空间。而其余步骤同 FA。

方式 2　在 CFA 的基础上,对种群每一代全局最优个体的位置(用 $Gbest$ 表示)用高斯分布进行微小扰动,提出一种带高斯扰动的混沌萤火虫算法(Chaotic Firefly Algorithm with Gaussian Perturbation, GCFA)。

$$NGbest=Gbest*(1+Gaussian(\sigma)) \tag{7}$$

其中 $NGbest$ 为扰动后的位置。对新的全局最优位置按式(8)进行更新:

$$Gbest^{t+1}=\begin{cases}NGbest^t, & f(NGbest^t)<f(Gbest^t)\\ Gbest^t, & \text{其他}\end{cases} \tag{8}$$

由式(4)可知,最亮的萤火虫随机飞行,通过对 $Gbest$ 做微小扰动可以帮助其跳出局部极值区域,可有效提高整个算法的性能。

方式 3　在 CFA 的基础上，提出一种基于混沌理论的动态种群萤火虫算法（Chaos-based Dynamic Population Firefly Algorithm，CDPFA）。设置一阈值 N_p，采用 CFA 首先进行全局搜索；算法每迭代 N_p 次则根据当前最优解进行局部搜索，即采用式(9)所示的立方映射混沌模型，生成 ps 个新个体，采用随机方式替换掉原种群中相应数目个体。

$$\begin{cases} y_{11} = Gbest * \mathrm{rand}(ps, dim) \\ y_{i+1,j} = 4y_{i,j}^3 - 3y_{i,j} \end{cases} \tag{9}$$

其中：$y_{i,j}$ 为新生成的萤火虫个体 i 的位置，第一个萤火虫位置以随机数形式产生，ps 为产生的新个体的数目，dim 为维数。

为了保持种群的多样性，加强搜索的分散性，CDPFA 采用随机方式用新生成的部分个体替换掉原种群中的 ps 个个体。由式(9)可知，新的萤火虫个体散布在全局最优位置的周围，有效地增加了对原有种群的牵引，平衡了在全局最优位置处的全局搜索和局部探索，大大提高了算法的收敛速度，并且使算法有更多的机会搜索到精度更高的解。最后，对每一代产生的全局最优解再根据式(7)进行高斯扰动，根据式(8)获得新的全局最优解。

综上所述，CDPFA 的具体流程如下：

步骤 1　初始化每个萤火虫的最大吸引度 β_0，吸收系数 γ，随机步长大小 α，迭代次数 $MaxT$。

步骤 2　采用 CMS 对萤火虫的位置进行初始化。

步骤 3　根据式(1)确定萤火虫之间的距离，式(2)更新萤火虫吸引度，式(3)更新萤火虫的位置，式(4)确定最好的萤火虫随机飞行。

步骤 4　若迭代次数等于 N_p，则进入混沌搜索状态

1）采用式(9)产生 ps 个新个体；

2）采用随机方式替换掉原种群中的 ps 个个体；

3）对于每个萤火虫个体，计算其新位置对应的目标值 $f(y_n)$；

4）记录全局最优值。

步骤 5　对全局最优值采用式(7)做高斯扰动，采用式(8)获得新的全局最优值。

步骤 6　若算法停止准则满足，则停止；否则转入步骤 3。

步骤 7　输出全局最优值 $Gbest$，算法结束。

3　数值实验与结果分析

3.1　环境与参数

对于 CFA、GCFA、CDPFA 与 FA 四种算法，利用表 1 中的 6 个经典复杂 Benchmark 测试函数进行仿真实验比较。仿真硬件环境为 HP Intel CPU T2050 1.60 GHz，512 MB RAM，操作系统为 Windows XP，利用VC++6.0 编程实现。

各算法的参数设置为：种群规模 $n = 40$，随机步长参数 $\alpha = 0.01$，吸引度 $\beta_0 = 0.7$，荧光吸收率 $\gamma = 1.0$，高斯变异参数 $\sigma = 0.01$，周期更新代数 $N_p = 20$，新个体个数 $ps = 10$。由于每种算法一次迭代的时间各异，因此本文采用设定相同运行时间的方法对以上四种算法的优劣进行比较，即设定每种算法的运行时间均为 $Time = 5$ s，比较各算法在运行时间 $Time$ 后的计算结果。

3.2　实验结果分析

表 2 中给出了四种算法对每个函数进行 20 次独立实验的计算结果，其中 $Best$ 为 20 次的最优解，$Worst$ 为最差解，$Average$ 为 20 次求得解的数学期望，$Std.\ Dev.$ 为相应的标准方差。为了进一步比较 FA 与较好的改进算法 CDPFA 的收敛趋势，在图 1～6 中给出分别利用两种算法求解 Benchmark 函数 $f_1 \sim f_6$ 各 20 次的平均进化曲线。

由以上实验结果可知：

1）由表 2 中的计算结果可以看出，三种改进算法对于 Benchmark 函数 $f_1 \sim f_6$，无论是平均最优适应度值还是标准差均优于原始的 FA。其中 CFA 仅仅采用立方映射产生混沌序列，虽然保证了萤火虫个体初始位置在整个搜索空间的均匀分布，但是方差略有减小，效果并不明显。GCFA 不仅仅采用立方映射产生混沌序列对萤火虫位置进行初始化，而且通过高斯变异对全局最优值做微小扰动，帮助其跳出局部极值区域，提高了算法收敛速度。对于函数 f_1、f_2、f_5，均值以及标准差都存在数量级别的提升，函数 f_4、f_6 均值以及标准差与原始 FA 相当，仍然处于一个数量级别，这说明了算法的有效性。CDPFA 的改进效果明显优于以上两种策略。这证明本文提出的 CDPFA 在充分利用混沌序列的遍历性和随机性的同时，为萤火虫发现全局最优解位置提供重要的信息，致使萤火虫快速接近全局最优解，而且通过对最优解采用高斯扰动进行变异操作，一定程度上使算法更具有跳出局部极小的能力。

2）由图 1 的平均进化曲线可以看出，CDPFA 对应的目标函数下降曲线比 FA 对应的下降曲线具有更快的下降速度和更好的下降程度。这再次表明，CDPFA 比 FA 在函数全局优化方面具有更好的有效性，收敛速度更快。

表 1　6 个典型的复杂 Benchmark 函数 $f_1 \sim f_6$

函　数	维数	搜索范围	理论值
$f_1(x) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	30	$[-100,100]^d$	0
$f_2(x) = \max_i\{\lvert x_i\rvert, 1 \leqslant i \leqslant n\}$	30	$[-100,100]^d$	0
$f_3(x) = \sum_{i=1}^{n-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$	30	$[-50,50]^n$	0
$f_4(x) = \sum_{i=1}^{n-1}(x_i^2 + x_{i+1}^2)^{0.25}[\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1]$	30	$[-100,100]^d$	0
$f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	$[-32,32]^d$	0
$f_6(x) = \sum_{i=1}^{n} ix_i^2$	30	$[-5.12,5.12]^d$	0

表 2　测试函数 f_1 ~ f_6 对于四种算法的实验结果比较

算法	f_1				f_2			
	Best	*Worst*	*Average*	*Std. Dev.*	*Best*	*Worst*	*Average*	*Std. Dev.*
FA	3.024E-03	25.019	2.268	5.391	1.190E-04	5.278E-03	1.092E-03	1.141E-03
CFA	3.865E-07	7.588	1.223	1.994	4.521E-05	1.124E-03	5.252E-04	2.826E-04
GCFA	7.708E-06	0.1862	3.605E-02	5.286E-02	1.010E-06	3.541E-04	1.277E-04	1.169E-04
CDPFA	8.194E-08	6.184E-04	7.017E-05	1.339E-04	9.251E-10	1.842E-04	4.194E-05	5.337E-05
算法	f_3				f_4			
	Best	*Worst*	*Average*	*Std. Dev.*	*Best*	*Worst*	*Average*	*Std. Dev.*
FA	1.995E-06	1.369E-03	3.040E-04	4.014E-04	0.4582	2.9963	1.8981	0.6702
CFA	2.244E-08	1.196E-03	3.093E-04	3.714E-04	0.4393	2.2713	1.0824	0.4719
GCFA	1.047E-04	8.562E-01	9.687E-02	1.932E-01	9.948E-04	1.726	0.634	0.529
CDPFA	1.138E-08	1.189E-05	1.896E-06	2.625E-06	4.203E-07	5.726E-04	1.735E-04	1.789E-04
算法	f_5				f_6			
	Best	*Worst*	*Average*	*Std. Dev.*	*Best*	*Worst*	*Average*	*Std. Dev.*
FA	1.253E-04	5.763E-03	1.906E-03	1.538E-03	1.638E-11	1.770E-07	1.797E-08	3.760E-08
CFA	5.042E-05	2.448E-03	8.706E-04	6.832E-04	1.602E-11	1.044E-07	1.753E-08	2.594E-08
GCFA	8.439E-09	4.768E-04	1.234E-04	1.101E-04	3.599E-11	2.771E-07	3.268E-08	6.809E-08
CDPFA	1.639E-08	1.376E-05	3.820E-06	4.014E-06	2.776E-14	7.194E-08	8.187E-09	1.577E-08

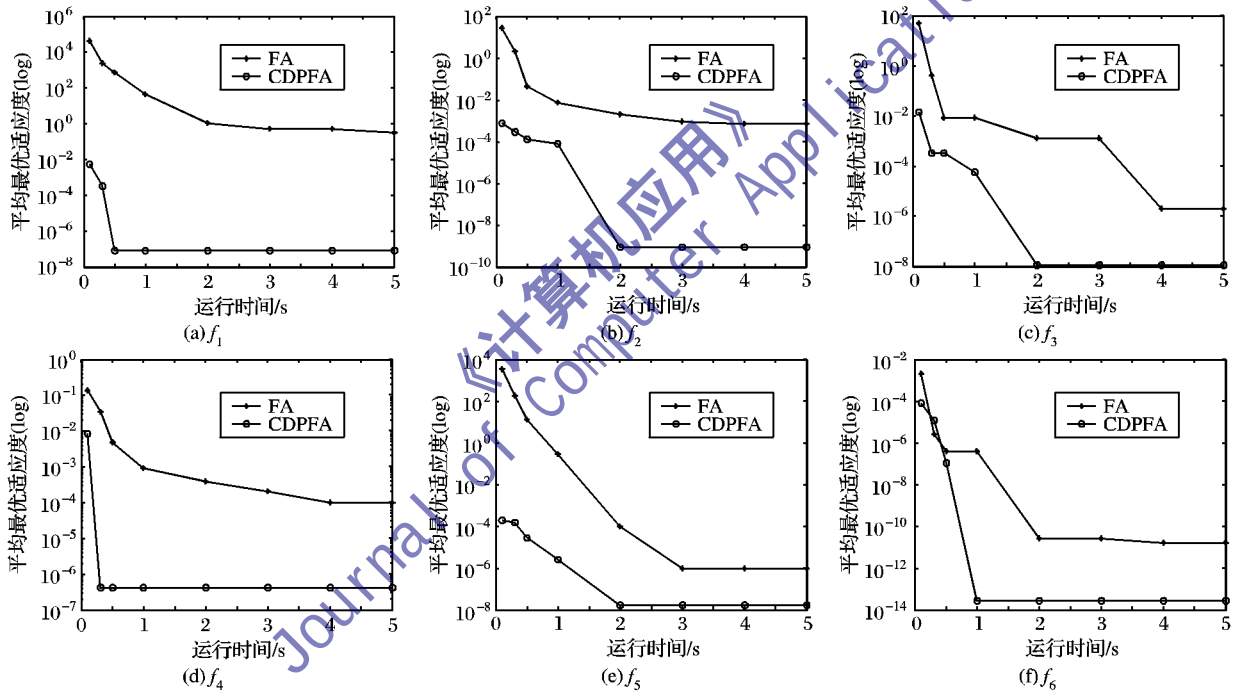


图 1　f_1 ~ f_6 的平均进化曲线

4　结语

针对 FA 易陷入早熟、求解精度不高等缺陷,本文采用三种方式对算法进行改进,从实验结果看出,CDPFA 无论在求解精度上还是收敛速度均优于 CFA 和 GCFA。由于 CDPFA 引入混沌序列对萤火虫位置初始化,有效地防止萤火虫位置分布的不均匀,且算法每隔一定代数,利用混沌特征产生新种群,很好地保证了萤火虫在整个迭代过程中的多样性,使得萤火虫不会过多朝向某一固定区域聚集,减少了寻优时陷入局部最优的可能。此外,新的萤火虫个体散布在全局最优位置的周围,有效地平衡了在全局最优位置处的全局搜索和局部挖掘,使得算法加速收敛且有更大的机会搜索到精度更优的位置。由于 FA 是新近提出的,有关算法参数的研究还较少,应用领域也相对较窄,今后在此方面值得进一步深入的研究。

参考文献:

[1] YANG X S. Firefly algorithms for multimodal optimization[C]// SAGA'09: Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications. Berlin: Springer-Verlag, 2009: 169-178.

[2] LUKASIK S, ZAK S. Firefly algorithm for continuous constrained optimization tasks[C]// Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems, LNCS 5796. Berlin: Springer, 2009: 97-100.

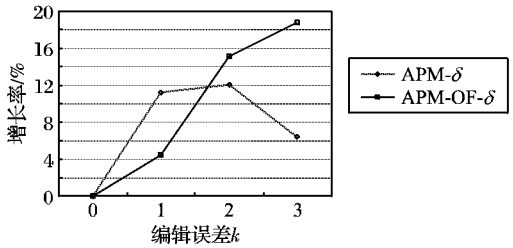
[3] YANG X S. Firefly algorithm, stochastic test functions and design optimisation[J]. International Journal of Bio-Inspired Computation, 2010, 2(2): 78-84.

[4] YANG X S. Firefly algorithm, Lévy flights and global optimization [M]. Berlin: Springer, 2010: 209-218.

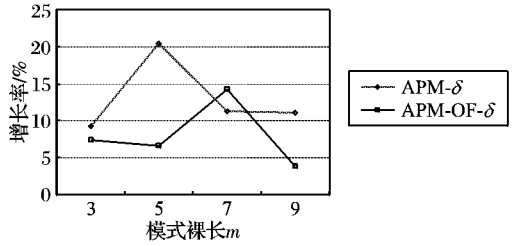
[5] APOSTOLOPOULOS T, VLACHOS A. Application of the firefly algorithm for solving the economic emissions load dispatch problem [J]. International Journal of Combinatorics, 2011, 2011: Article ID 523806.

(下转第 805 页)

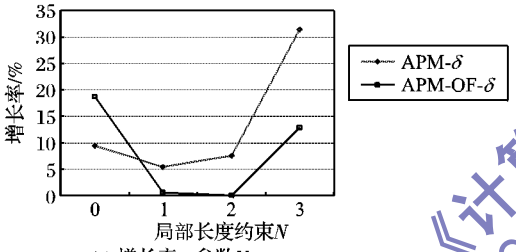
MAPM 算法调用 APM-OF 算法(插入、替换和删除操作),而 OneoffMining 算法调用 Quicksearch 算法[13](仅精确匹配),故 MAPM 算法在模式近似挖掘方面要明显优于 OneoffMining 算法。令最小支持度 $min\_sup = 0.013 * |T|$,$|T|$ 为随机文本串的长度,分别为1000,2000,3000,4000,间隔约束为[9,11],实验结果如图6所示,MAPM 算法找到的频繁模式个数约为 OneoffMining 算法的2.07倍。



(a) 增长率δ-参数k



(b) 增长率δ-参数m



(c) 增长率δ-参数N

图5　增长率δ变化曲线

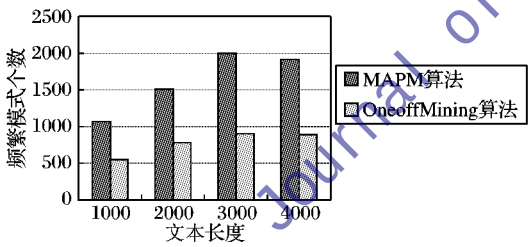


图6　MAPM 算法与 OneoffMining 算法对比

5　结语

针对 APMWL 问题,本文提出的 APM 和 APM-OF 算法与现有算法相比所处理的问题更加灵活和复杂。在真实生物数据上的实验结果显示算法在解的平均增长率上分别达到8.34%和12.37%,反映了其有效性。同时,本文分析了算法求解APMWL问题的主要因素,发现当编辑误差k值较大,模式P中字符(非通配符)的个数m适中,局部长度约束下限N_i很小或很大时解增长率最为明显,可分别达到31.43%和18.78%。此外,本文将 APM-OF 算法推广至模式挖掘中,与同类算法 OneoffMining 相比,挖掘出的频繁模式个数为其2.07倍,这也意味着可以挖掘出更多的有效信息。以后我们将试图对算法时间及空间性能进行改进。如何在线输出质量更高的解(即编辑误差较小,解的总个数更多),也是我们未来的研究方向。

参考文献:

[1] CHANG Y I, CHEN J R, HSU M T. A Hash Trie filter method for approximate string matching in genomic databases[J]. Applied Intelligence, 2010, 33(1):21 – 38.

[2] BHUKYA R, SOMAYAJULU D V L N. 2-jump DNA search multiple pattern matching algorithm[J]. International Journal of Computer Science Issues, 2011, 8(3):320 – 329.

[3] 龚才春, 黄玉兰, 许洪波, 等. 基于多重索引模型的大规模词典近似匹配算法[J]. 计算机研究与发展, 2008, 45(10):1776 – 1781.

[4] 范立新. 改进的中文近似字符串匹配算法[J]. 计算机工程与应用, 2006, 42(34):172 – 207.

[5] 樊爱京, 杨照峰. 用于网络入侵检测的模式匹配新方法[J]. 计算机应用, 2011, 31(11):2961 – 2985.

[6] QU Z Y, HUANG X B. The improving pattern matching algorithm of intrusion detection[C]// 2011 International Conference on Advanced in Control Engineering and Information Science. Oxford: Elsevier, 2011:2841 – 2846.

[7] ALTSCHUL S F, BOGUSKI M S, FISH W, *et al*. Issues in searching molecular sequence databases[J]. Nature Genetics, 1994(6): 119 – 128.

[8] WU S, MANBER U, MYERS E W. A subquadratic algorithm for approximate limited expression matching[J]. Algorithmica, 1996, 15(1):50 – 67.

[9] NAVARRO G. Regular expression searching over Ziv-Lempel compressed text[C]// Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching. Berlin: Springer, 2001:1 – 17.

[10] FISCHER M J, PATERSON M S. String matching and other products[C]// Proceedings of the 7th SIAM-AMS Complexity of Computation. New York: American Mathematical Society, 1974: 113 – 125.

[11] CHEN G, WU X D, ZHU X Q, *et al*. Efficient string matching with wildcards and length constraints[J]. Knowledge and Information Systems, 2006, 10(4):399 – 419.

[12] HE D, WU X D, ZHU X Q. SAIL-APPROX: an efficient on-line algorithm for approximate pattern matching with wildcards and length constraints[C]// IEEE International Conference on Bioinformatics and Biomedicine. Washington, DC: IEEE Computer Society, 2007: 151 – 158.

[13] HUANG Y M, WU X D, HU X G, *et al*. Mining frequent patterns and one-off condition[C]// Proceedings of the 12th IEEE International Conference on Computational Science and Engineering. Washington, DC: IEEE Computer Society, 2009: 180 – 186.

(上接第 799 页)

[6] CHAI-EAD N, AUNGKULANON P, LUANGPAIBOON P. Bees and firefly algorithms for noisy non-linear optimisation problems [C]// Proceedings of the International MultiConference of Engineers and Computer Scientists. Hongkong: [s. n.], 2011, 2.

[7] 刘长平, 叶春明. 一种新颖的仿生群智能优化算法: 萤火虫算法[J]. 计算机应用研究, 2011, 28(9):3295 – 3297.

[8] 胥小波, 郑康锋, 李丹, 等. 新的混沌粒子群优化算法[J]. 通信学报, 2012, 33(1):24 – 37.

[9] 王凌, 刘波. 微粒群优化与调度算法[M]. 北京: 清华大学出版社, 2008:38 – 39.

[10] 王翔, 李志勇, 许国艺, 等. 基于混沌局部搜索算子的人工蜂群算法[J]. 计算机应用, 2012, 32(4):1033 – 1036.

[11] 周燕, 刘培玉, 赵静, 等. 基于自适应惯性权重的混沌粒子群算法[J]. 山东大学学报: 理学版, 2012, 47(30):1 – 6.

[12] 黄凯, 周永权. 带交尾行为的混沌人工萤火虫优化算法[J]. 计算机科学, 2012, 39(3):231 – 234.