基于斥力的引力搜索算法

王奇琪 孙根云 王振杰 张爱竹 陈晓琳 黄丙湖
(中国石油大学(华东)地球科学与技术学院 青岛 266580)

摘 要 针对引力搜索算法(Gravitational Search Algorithm,GSA)收敛速度较快、易陷入局部最优的缺点,提出一种加入斥力的引力搜索算法 RFGSA(Repulsion Force based Gravitational Search Algorithm)。该算法在引力搜索算法中引入斥力,即将一部分引力变为斥力,从而增加种群的多样性,有利于寻找全局最优。对10个基准测试函数进行优化的结果表明:该算法的收敛结果明显优于遗传算法、粒子群算法及原始的引力搜索算法。

关键词 引力搜索算法(GSA),斥力,多样性,基准测试函数
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.9.046

Repulsion Force Based Gravitational Search Algorithm

WANG Qi-qi SUN Gen-yun WANG Zhen-jie ZHANG Ai-zhu CHEN Xiao-lin HUANG Bing-hu
(School of Geosciences, China University of Petroleum, Qingdao 266580, China)

Abstract To overcome the shortage of gravitational search algorithm(GSA), such as high convergence speed and premature convergence, this paper presented a repulsion force based GSA(RFGSA). In RFGSA, repulsion force is introduced to GSA, which means that a part of attraction force is changed to repulsive force. Therefore, the diversity of the population is increased and thus the search ability of GSA is improved. To demonstrate the validity of RFGSA, 10 benchmark functions were tested. The compared results indicate the significant superiority of the proposed algorithm.

Keywords Gravitational search algorithm(GSA), Repulsion force, Diversity, Benchmark function

1 引言

在解决具有高维搜索空间的优化问题时,由于搜索空间随着维度的增加呈指数级增加,因此很难使用精确解法(如穷举搜索)来解决这类复杂的高维优化问题[1]。在过去的几十年里,科学家们受自然现象的启发,提出了许多适合解决复杂计算问题的启发式优化算法,比如遗传算法[2]、模拟退火算法[3]、蚁群搜索算法[4]、粒子群优化算法[5]等。这些算法可以很好地解决某些特定情况下的优化问题,尤其对于高维优化问题具有独特的优势,因而受到研究者的关注。

引力搜索算法(Gravitational Search Algorithm,GSA)[1]是 Rashedi 等人在2009年提出的一种元启发式算法。该算法受万有引力定律的启发,通过群体中粒子之间的相互作用实现最优化。现有的研究表明,GSA 的寻优精度和收敛速度都要优于粒子群算法[6](Particle Swarm Optimization,PSO)和遗传算法[7](Genetic Algorithm,GA)等优化算法,并且具有结构简单、易于实现、参数设置少和全局优化能力强等特点,在很多优化问题中都得到了成功应用[8-11]。但是 GSA 算法在优化过程中存在早熟收敛、易陷入局部最优等问题[12]。为了解决这些问题,许多研究者提出了一些解决方法,比如对 GSA 中粒子的记忆性进行改进[11,12]等。

本文针对 GSA 早熟收敛的缺点,提出一种加入斥力的引力搜索算法 RFGSA,该算法通过计算粒子和质心之间的距离,在模糊规则的指导下,自适应地将一部分引力变成斥力,从而增加种群的多样性,有利于寻找全局最优。对10个基准测试函数进行优化的结果表明,该算法的收敛结果明显优于原始的引力搜索算法,提高了 GSA 算法的收敛精度,避免了早熟收敛。

2 引力搜索算法

在 GSA 中,粒子之间都是相互吸引的,相互之间的作用力使种群中粒子朝着质量大的粒子的方向移动,如图1所示。每个粒子有4个特征:位置、惯性质量、施力粒子、受力粒子。粒子的位置就是问题的解。

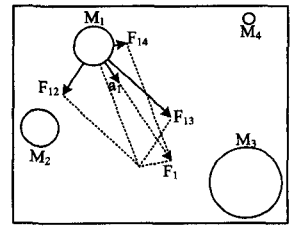


图1 粒子运动图

引力搜索算法中粒子的初始位置和初始速度都是随机生成的。

到稿日期:2014-09-15 返修日期:2014-12-08 本文受国家自然科学基金(41471353)资助。
王奇琪(1990—),女,硕士,主要研究方向为智能优化算法、遥感图像处理;孙根云(1979—),男,博士,副教授,主要研究方向为遥感图像处理、智能优化算法,E-mail:genyunsun@163.com。

根据式(1)和式(2)，可计算出每个粒子的惯性质量 $M_i(t)$：

$$m_i(t)=\frac{fit_i(t)-worst(t)}{best(t)-worst(t)} \tag{1}$$

$$M_i(t)=\frac{m_i(t)}{\sum_{j=1}^{N}m_j(t)} \tag{2}$$

其中，$i=1,2,\cdots,N$，N 为粒子数目；$m_i(t)$ 为计算粒子质量的中间变量；$fit_i(t)$ 是粒子 i 在 t 时刻的适应值；$worst(t)$ 和 $best(t)$ 分别是指在 t 时刻整个粒子群的最坏的适应值和最好的适应值。

在搜索目标函数最小值问题时，最坏和最好的适应值分别为：

$$worst(t)=\max_{j\in\{1,\cdots,N\}}fit_j(t) \tag{3}$$

$$best(t)=\min_{j\in\{1,\cdots,N\}}fit_j(t) \tag{4}$$

在搜索目标函数最大值问题时，最坏和最好的适应值分别为：

$$worst(t)=\min_{j\in\{1,\cdots,N\}}fit_j(t) \tag{5}$$

$$best(t)=\max_{j\in\{1,\cdots,N\}}fit_j(t) \tag{6}$$

然后，使用变换后的万有引力公式，可以计算各个粒子在每一维空间上相互之间的引力，在第 d 维空间上粒子 i 和粒子 j 之间的引力为：

$$F_{ij}^d(t)=G(t)\frac{M_{pi}(t)\times M_{aj}(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t)-x_i^d(t)) \tag{7}$$

式中，M_{aj} 和 M_{pi} 分别代表粒子 j 的主动引力质量和粒子 i 的被动引力质量，$G(t)$ 是在时间 t 处的万有引力常数值，ε 是一个小常数，$R_{ij}(t)$ 是粒子 i 和 j 间的欧氏距离：

$$R_{ij}(t)=\|X_i(t),X_j(t)\|_2 \tag{8}$$

万有引力常数 G 是一个初始值 G_0 和时间 t 的函数，变化的 G 能够更好地控制搜索过程：

$$G(t)=G(G_0,t) \tag{9}$$

根据牛顿运动定律，t 时刻粒子 i 在第 d 维空间上所受引力是搜索空间中其他所有的粒子作用力之和。GSA 中，为了增加算法的随机特性，引力叠加时使用 $rand_j$ 随机函数，如式(10)所示：

$$F_i^d(t)=\sum_{j\in Kbest,j\neq i}rand_jF_{ij}^d(t) \tag{10}$$

此时粒子 i 的惯性质量为 $M_{ii}(t)$，那么其加速度计算公式为式(11)：

$$a_i^d(t)=\frac{F_i^d(t)}{M_{ii}(t)} \tag{11}$$

最后，粒子 i 在 $t+1$ 时刻的速度和位置的更新公式为：

$$v_i^d(t+1)=rand_i\times v_i^d(t)+a_i^d(t) \tag{12}$$

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1) \tag{13}$$

3 基于斥力的引力搜索算法

在原始的 GSA 算法中，根据粒子的移动图(见图 1)可知，粒子在运动过程中受到种群中所有粒子的吸引，包括不利于找到全局最优的粒子的吸引，而且在收敛过程中速度过快，在迭代后期多样性损失较严重，易陷入局部最优。针对这一问题，为了获得较好的搜索结果，在 GSA 搜索过程中引入斥力，即将一部分引力自适应地变为斥力，从而增加种群的多样性，以有利于找到全局最优。

3.1 算法原理

3.1.1 斥力的引入

GSA 算法中，粒子之间通过引力相互吸引，粒子运动方式单一。为了增加粒子运动的多样性，引入斥力，可以在适当的时候改变粒子的作用力方向，延缓或者改变粒子的运动方向，从而使粒子能够探索更多的未知区域，以加强全局搜索能力。

在种群中引入斥力，需要解决两个问题：(1)对哪些粒子引入斥力；(2)斥力的大小如何确定。本文通过引入群质心 X_{cen}[13] 和半径 r 两个参数完成这两个任务。

首先，计算粒子 i 到群质心 X_{cen} 的距离 $d(x_{cen},x_i)$：

$$d(x_{cen},x_i)=|x_i-X_{cen}| \tag{14}$$

其中，群质心 X_{cen} 为：

$$X_{cen}=\frac{\sum_{i=0}^{N}x_im_i}{\sum_{i=0}^{N}m_i} \tag{15}$$

其中，N 为粒子的个数，x_i 表示粒子 i 的坐标，m_i 表示粒子 i 的质量(本算法中此处的质量 m_i 等于粒子 i 的适应值)。

然后，比较 $d(x_{cen},x_i)$ 和半径 r 的大小，若 $d(x_{cen},x_i)<r$，则半径内的粒子对粒子 i 就表现为斥力。落在半径 r 中的粒子越多，斥力越大。

引入斥力后粒子之间的作用力如式(16)所示。

$$F_{ij}^d(t)=A\times G(t)\frac{M_{pi}(t)\times M_{aj}(t)}{R_{ij}(t)+\varepsilon}(x_j^d(t)-x_i^d(t)) \tag{16}$$

当 A 取值为 1 时，粒子之间表现为引力；当 A 取值为 -1 时，粒子之间表现为斥力。

3.1.2 半径大小的模糊控制策略

如 3.1.1 节所述，半径 r 的大小直接决定施加斥力的粒子数目，从而影响种群中斥力的大小。为了更好地发挥斥力的作用，需要根据种群进化水平自适应地调整半径大小，所以本文引入度量种群进化水平的参数 CM[14]，在最小化问题中，CM 计算公式如下：

$$CM=\frac{fit^{ave}(t-1)-fit^{ave}(t)}{fit^{ave}(t)} \tag{17}$$

CM 为负值，说明 $t-1$ 次迭代结果比 t 次好，表明粒子正朝着不利于找到全局最优值的方向运动，此时应取较大的半径，所以施加斥力的粒子数增加，斥力增大，有利于搜索更广泛的区域，找到全局最优；CM 为正值，说明 t 次迭代结果比 $t-1$ 次好，值越大说明 t 次的迭代结果越好，此时应取较小的半径，斥力也相应地较小，有利于粒子缩小搜索范围，更快地完成寻优过程。

具体地说，在算法搜索的初级阶段，粒子的分布范围较广，较大的半径才能够保证对足够的粒子施加斥力，较大的斥力才会对种群产生作用；在迭代后期，粒子已经收敛到了一定的范围，此时半径太大就会使得所有的粒子都是斥力，进而使得粒子来回震荡，难以收敛。因此，在迭代过程中，根据粒子的集散程度确定半径 r，既可以增加种群的多样性，又能够使得粒子收敛到全局最佳值。

所以，本文根据种群进化水平，设计了一个模糊控制器来完成半径大小的自适应调整，进而控制斥力的大小，防止算法陷入局部最优和过早收敛。半径 r 的模糊控制见表 1。

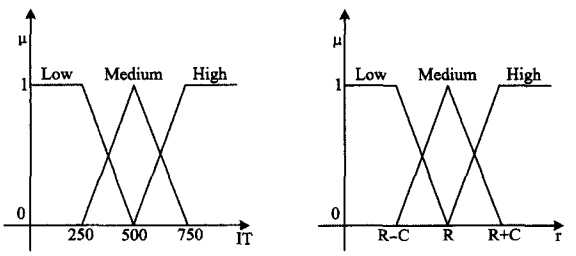
表 1　控制半径 r 的模糊规则

Rule		IT	CM		半径 r	斥力
1		Low	正		High	增大
2		Medium	正		Medium	减小
3	IF	High	负	Then	Medium	增大
4		High	正		Low	减小

如表 1 所列，在控制器中输入变量为迭代次数 IT 和 CM，输出变量为半径 r，如图 2 所示。其中，IT 为当前迭代次数。$IT\in[1,1000]$，CM 的值为正或者负，半径 $r\in[R-2C, R+2C]$，其中 $C=R/8$，在每次迭代时 R（所有粒子到质心的平均距离）的值都会重新计算，因此半径 r 的大小是自适应调整的。R 的计算公式如下：

$$R=\sum_{i=1}^{N}\frac{d(x_{cen},x_i)}{N} \tag{18}$$

其中，$d(x_{cen},x_i)$ 为粒子 i 到质心的距离，N 为种群粒子数。



(a)隶属函数的输入值 IT（当前迭代次数）　　(b)输出值半径 r

图 2

在表 1 中，第一条模糊规则可以描述为：当迭代次数较少，并且 CM 为正值时，增大粒子半径。

当迭代次数较少，并且 CM 为正值时，说明粒子朝着有利于寻找最优值的方向运动，是一种易陷入局部最优而过早收敛的迹象，因此应增加粒子半径，使得半径中产生斥力的粒子增多，改变粒子的运动方向，从而有利于寻找全局最优值。

第二条模糊规则可以描述为：当迭代进行一半左右时，并且 CM 为正值时，减小粒子半径。

迭代中期，CM 为正值时，表明粒子正朝着有利于寻找全局最佳的方向运动，因此应减小半径，使得种群中产生斥力的粒子减少，有助于粒子朝着全局最优值的方向运动。

第三条模糊规则可以描述为：当迭代次数较大，并且 CM 为负值时，增大粒子半径。

迭代后期，CM 为负值时，说明在迭代后期粒子还没有找到全局最佳值，而且还是朝着不利于寻找最优值的方向运动，因此，应该增大粒子半径，增加斥力对当前运动粒子的作用，增加种群的多样性，但是在迭代后期粒子较密集，半径的设置不宜过大，因此选择 Medium。

第四条模糊规则可以描述为：当迭代次数较大，并且 CM 为正值时，减小粒子半径。

迭代后期，CM 为正值时，说明粒子朝着有利于寻找全局最优值的方向运动，应减小半径 r，减小斥力对粒子运动方向的影响，使得粒子能够朝着全局最优值的方向运动，有助于快速地收敛到全局最佳。

通过上述模糊规则的控制，既能够避免陷入局部最优而过早收敛，又能够保持种群的多样性，有利于寻找全局最优值。

3.2　算法流程

改进的 GSA 算法如下：

步骤 1　确定搜索空间；

步骤 2　初始化种群数目 N，最大迭代次数 max_it，并随机初始化粒子的位置和速度；

步骤 3　对每个粒子，根据目标函数计算它的适应值；

步骤 4　更新引力系数函数 $G(t)$、$gbest(t)$、$gworst(t)$ 惯性质量 $M_i(t)$；

步骤 5　计算 CM，通过模糊规则得到半径 r 的大小；

步骤 6　计算各粒子到质心的距离，并和 r 进行比较，确定系数 A 的值；

步骤 7　计算每个粒子不同方向上的力的总和；

步骤 8　计算粒子加速度和速度；

步骤 9　更新每个粒子的位置；

步骤 10　循环迭代，直至达到迭代次数或满足要求精度为止；

步骤 11　结束循环，输出结果。

RFGSA 进行迭代的流程如图 3 所示。

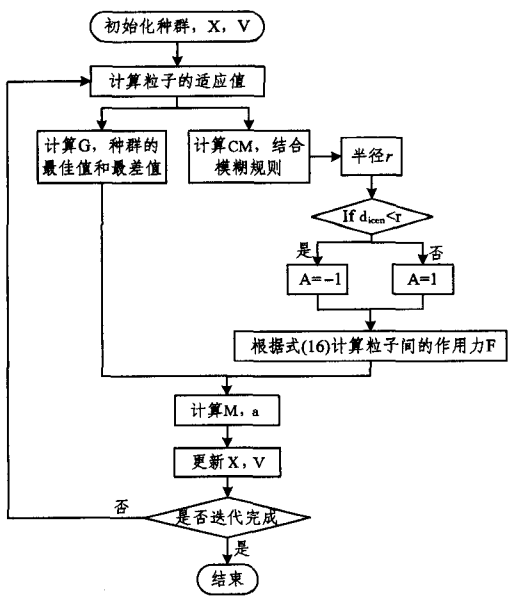


图 3　RFGSA 流程图

4　仿真实验及分析

本文在 Windows 7 系统上，使用 Matlab 2010 版本进行仿真实验。为了测试改进算法的性能，选用了 10 个基准测试函数进行实验，寻找测试函数的最小值。4.1 节介绍测试函数，4.2 节对实验结果进行比较分析。

4.1　测试函数

测试实验所用到的基准测试函数如表 2 所列，其中，n 代表函数的维数，S 是 R^n 的子集。表 2 中测试函数最小值都为

0，除了 F_7 以外，它们的最优位置 X_{opt} 都为$[0]^n$，F_7 的最优位置 X_{opt} 为$[1]^n$。

表 2　测试函数

测试函数	S
单峰测试函数	
$F_1(X)=\sum_{i=1}^{n}x_i^2$	$[-100,100]^n$
$F_2(X)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$	$[-100,100]^n$
$F_3(X)=\sum_{i=1}^{n}([x_i+0.5])^2$	$[-100,100]^n$
$F_4(X)=\sum_{i=1}^{n}ix_i^4+random(0,1)$	$[-128,128]^n$
多峰测试函数	
$F_5(X)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]^n$
$F_6(X)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^n$
$F_7(X)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}\mu(x_i,10,100,4)$	
$y_i=1+\frac{x_i+1}{4}$	$[-50,50]^n$
$\mu(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a<x_i<a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	
$F_8(X)=\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	$[-5,5]^4$
$F_9(X)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	$[0,10]^4$
$F_{10}(X)=-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	$[0,10]^4$

4.2　实验结果比较及分析

本实验将基于斥力的引力搜索算法(RFGSA)和遗传算法(GA)、粒子群算法(PSO)、原始引力搜索算法(GSA)对基准函数进行优化的结果进行比较分析。在所有情况下，粒子的个数设为 50($N=50$)，最大迭代次数设置为 1000($max\_it=1000$)，前 7 个测试函数的维数取 $n=30$，后 3 个测试函数的维数如表 2 所列。遗传算法参数设置为：交叉概率为 0.8，变异概率为 0.2，保留比例为 0.5。粒子群算法参数设置为：$c1=c2=2$，惯性权重从 0.9 线性递减为 0.2。

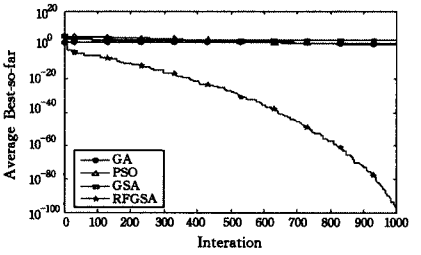
在 GSA 和 RFGSA 中，G_0 设置为 100，α 为 20，T 是总的迭代次数，G 的计算公式如式(19)所示：

$$G(t)=G_0e^{-\alpha\frac{t}{T}} \tag{19}$$

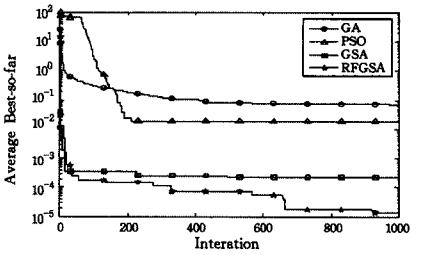
采用 GA、PSO、GSA 和 RFGSA 分别对表 2 中的测试函数进行优化，实验结果及分析如下。

4.2.1　单峰测试函数

表 3 是 GA、PSO、GSA 和 RFGSA 分别对表 2 中的单峰测试函数运行 30 次得到的结果。其中，各函数在各个指标上取得的最优值加粗显示。从表 3 所列的实验数据中可以看出：对于 4 个高维的单峰测试函数，RFGSA 的搜索结果明显好于 GA、PSO 和 GSA 的搜索结果。另外，对于单峰测试函数来说，搜索速度是检测算法性能的重要指标[1]。从图 4 的收敛图可以看到：RFGSA 在保证找到最好的全局最优的情况下，搜索速度较对比算法更快。



(a)GA、PSO、GSA 和 RFGSA 对单峰函数 F_2 优化的过程曲线



(b)GA、PSO、GSA 和 RFGSA 对单峰函数 F_4 优化的过程曲线

图 4

表 3　单峰测试函数最小值搜索结果

		GA	PSO	GSA	RFGSA
F_1	Average best-so-far	0.1837	0.0288	2.34×10^{-17}	$\mathbf{1.06\times10^{-19}}$
	Median best-so-far	0.0360	0.0292	2.42×10^{-17}	$\mathbf{1.04\times10^{-19}}$
	Average mean fitness	0.1915	0.0288	3.60×10^{-17}	$\mathbf{2.52\times10^{-19}}$
F_2	Average best-so-far	0.3392	7.6982	2.50×10^{2}	$\mathbf{2.83\times10^{-94}}$
	Median best-so-far	0.0734	6.3326	2.33×10^{2}	$\mathbf{2.10\times10^{-96}}$
	Average mean fitness	0.3432	7.6982	2.50×10^{2}	$\mathbf{5.42\times10^{-84}}$
F_3	Average best-so-far	179	0	0	**0**
	Median best-so-far	166	0	0	**0**
	Average mean fitness	180	0	0	**0**
F_4	Average best-so-far	0.0709	0.0015	0.0122	$\mathbf{1.41\times10^{-5}}$
	Median best-so-far	0.0798	0.0010	0.0104	$\mathbf{1.18\times10^{-5}}$
	Average mean fitness	0.5765	0.4861	0.5689	$\mathbf{1.43\times10^{-5}}$

4.2.2　多峰测试函数

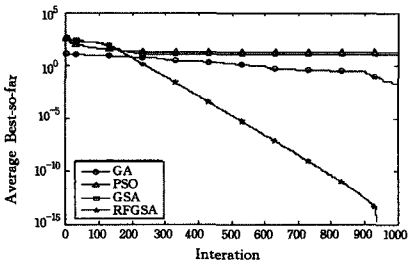
在表 2 的多峰测试函数中，F_5-F_7 为多峰高维测试函数，F_8-F_{10} 为固定维度多峰测试函数。

对于多峰测试函数，存在较多的局部最优解，因此优化难度较大，能够得到全局最优解是最重要的。在本实验中将表 2 中的多峰测试函数运行 30 次，得到的最优化果如表 4 所列，其中粗体表示最优值。

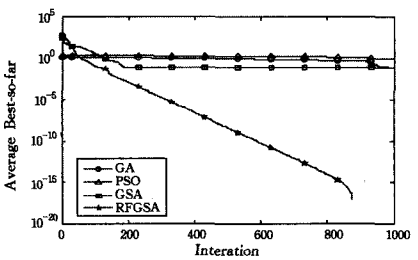
表 4 中的多峰测试函数的优化结果显示，RFGSA 的优化结果均优于 GA、PSO 和原始 GSA，其中相比 GSA 的优化结果，RFGSA 将 F_7 的优化精度提高了 16 个数量级，尤其对于 F_5 和 F_6，RFGSA 能够在迭代后期跳出局部最优，找到全局最优值 0，收敛曲线如图 5 所示。

表 4　多峰测试函数最小值搜索结果

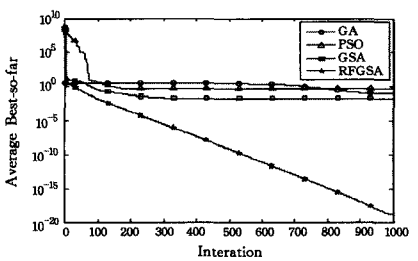
		GA	PSO	GSA	RFGSA
F_5	Average best-so-far	2.28	20.64	12.44	0
	Median best-so-far	3.27	20.23	12.44	0
	Average mean fitness	2.31	20.64	12.44	1.70×10^{-15}
F_6	Average best-so-far	0.3919	0.5423	0.0840	0
	Median best-so-far	0.3110	0.6315	0.0963	0
	Average mean fitness	0.4092	0.5669	0.0840	0
F_7	Average best-so-far	0.0241	0.6218	2.80×10^{-4}	6.62×10^{-20}
	Median best-so-far	0.0024	0.4974	2.67×10^{-4}	6.54×10^{-20}
	Average mean fitness	0.0243	0.6218	3.11×10^{-4}	2.75×10^{-19}
F_8	Average best-so-far	0.0054	0.1088	0.0011	3.18×10^{-4}
	Median best-so-far	0.0034	0.0891	0.0013	3.15×10^{-4}
	Average mean fitness	0.0080	0.1087	0.0011	3.28×10^{-4}
F_9	Average best-so-far	−5.9463	−7.1546	−6.4180	−10.1532
	Median best-so-far	−5.0552	−10.1532	−6.4180	−10.1532
	Average mean fitness	−4.4191	−7.1546	−6.4180	−10.1532
F_{10}	Average best-so-far	−7.3686	−7.9812	−10.5364	−10.5364
	Median best-so-far	−10.5364	−10.5364	−10.5364	−10.5364
	Average mean fitness	−7.3686	−7.9812	−10.5364	−10.5364



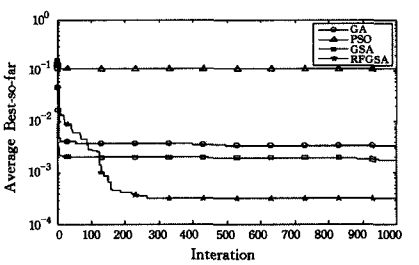
(a)GA、PSO、GSA、RFGSA 对多峰函数 F_5 进行优化的过程曲线



(b)GA、PSO、GSA、RFGSA 对多峰函数 F_6 进行优化的过程曲线



(c)GA、PSO、GSA、RFGSA 对多峰函数 F_7 进行优化的过程曲线



(d)GA、PSO、GSA、RFGSA 对多峰函数 F_8 进行优化的过程曲线

图 5

结束语　在引力搜索算法(GSA)中，所有的粒子都是相互吸引相互靠近的，最后达到一个平衡状态，也就是说粒子在寻找最优值的过程中，是不断地通过其他粒子的吸引来调整位置的，直到找到最优解。但是在这个运动过程中，可能会由于引力较大，导致粒子在收敛初期种群多样性丧失，陷入局部最优而过早收敛。为了提高种群的寻优能力，本文引入斥力，即将一部分粒子的吸引力变成排斥力，从而能够保持种群多样性，有利于寻找全局最优值。实验结果表明：引入斥力的引力搜索算法对函数的优化效果明显优于遗传算法、粒子群优化算法及原始引力搜索算法。

参 考 文 献

[1] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm [J]. Information Sciences, 2009, 179(13): 2232-2248

[2] Tang K S, Man K F, Kwong S, et al. Genetic algorithms and their applications [J]. IEEE Signal Processing Magazine, 1996, 13(6): 22-37

[3] Kirkpatrick S, Gelatto C D, Vecchi M P. Optimization by simulated annealing [J]. Science, 1983, 220: 671-680

[4] Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperating agents [J]. IEEE Transactions on Systems, Man, and Cybernetics: Part B, 1996, 26(1): 29-41

[5] Karakuzu J, Eberhart R C. Particle swarm optimization [C]// Proceedings of IEEE International Conference on Neural Networks, 1995: 1942-1948

[6] 刘建华，张永晖，周理，等. 一种权重递增的粒子群算法[J]. 计算机科学，2014，41(3)：59-65
Liu Jian-hua, Zhang Yong-hui, Zhou Li, et al. Partide Swarm Optimization with Weight Increasing [J]. Computer Science, 2014, 41(3): 59-65

[7] 席裕庚，柴天佑，挥为民. 遗传算法综述[J]. 控制理论与应用，1996，13(6)：697-708
Xi Yu-geng, Chai Tian-you, Hui Wei-min. Survey on Genetic Algorithm [J]. Coontrol Theory and Applications, 1996, 13(6): 697-708

[8] Liu Y, Ma L. Gravitational search algorithm for location problem [J]. Computer Engineering and Applications, 2012, 48(27): 42-44

[9] 牛培峰，肖兴军，李国强，等. 基于万有引力搜索算法的电厂锅炉 NO_x 排放模型的参数优化[J]. 动力工程学报，2013，33(2)：100-106
Niu Pei-feng, Xiao Xing-jun, Li Guo-qiang, et al. Parameter Op-

timization for NO_x Emission Model of power Plant Boilers Based on Gravitational Search Algorithm [J]. Journal of Chinese Society of Power Engineering, 2013, 33(2): 100-106

[10] 李沛,段海滨. 基于改进万有引力搜索算法的无人机航路规划[J]. 中国科学,2012,42(10):1130-1136
Li P, Duan H B. Path planning of unmanned aerial vehicle based on improved gravitational search algorithm[J]. Sci China Tech Sci, 2012, 42(10): 1130-1136

[11] Jiang S, Ji Z, Shen Y. A novel hybrid particle swarm optimization and gravitational search algorithm for solving economic emission load dispatch problems with various practical constraints [J]. Electrical Power and Energy Systems, 2014, 55: 628-644

[12] 李春龙,戴娟,潘丰. 引力搜索算法中粒子记忆性改进的研究[J]. 计算机应用,2012,32(10):2732-2735
Li C-L, Dai J, Pan F. Analysis on improvement of particle memory in gravitational search algorithm[J]. Journal of Computer Applications, 2012, 32(10): 2732-2735

[13] 徐星,李元香,姜大志,等. 一种基于分子动理论的改进粒子群优化算法[J]. 系统仿真学报,2009,21(7):1904-1907
Xu X, Li Y X, Jiang D Z, et al. Improved Particle Swarm Optimization Algorithm Based on Theory of Molecular Motion[J]. Journal of System Simulation, 2009, 21(7): 1904-1907

[14] Rashedi E. Fuzzy Gravitational Search Algorithm [C]// 2012 2nd International Conference on Computer and Knowledge Engineering(ICCKE). 2012: 18-19

(上接第 239 页)

机内存的限制,算法将无法执行。有的学者为了解决这个问题,采用了局部比对的方法,但这种方法很难得到全局的最优解。因此,这个问题仍面临着更大的挑战,需要更进一步的深入探讨和研究。

(3)探索方法之间的互补性

将进一步使用更多的基因数据和拼接算法做实验,后期将加入 Meraculous、IDBA-UD 等先进的算法,并试图从实验中找出拼接算法之间互补性的规律和结论,为选用哪几种拼接方法进行融合提供指导和推荐。

(4) 对 CGDNA 进行加速

在运行耗时上,CGDNA 优势并不明显。主要的耗时操作为构建索引和读长映射两个步骤,对其进行加速的工作迫在眉睫,目前正在使用 Hadoop 构建大规模并行框架来优化 CGDNA,以减少 CGDNA 的运行时间并提高运行效率。

结束语 基因序列拼接是全基因组测序的重要一环,其中基因组从头拼接难度远远大于重测序,也更具挑战性。在深入分析当前基因测序方法的特性以及拼接数据高通量、短序列的特点之后,结合当前序列拼接问题的研究现状,本文提出通过构建簇图的方式解决当前基因序列拼接的问题。

本文提出的 CGDNA 算法根据多个算法生成的重叠群之间的相关性,寻找潜在可拼接的重叠群,然后通过序列比对验证匹配区域的确切位置,从而确定可继续拼接的重叠群集合。通过构建簇图,将较难的序列拼接问题转化为在簇图上寻找路径的问题。

实验结果表明,本文提出的 CGDNA 算法可以显著提高单个拼接算法的性能,在最长的 scaffold 和 scaffold N50 这两项指标上提高幅度均超过 50%。由于 CGDNA 算法可兼容任意多个拼接算法,我们相信使用更多的基本拼接算法将进一步提高 CGDNA 算法的性能。

参 考 文 献

[1] Medvedev P, Georgiou K, Myers G, et al. Computability of Models for Sequence Assmbly[M]// Algorithms in Bioinformatics. Springer-Verlag Berlin Heidelberg, 2007: 289-301

[2] Drmanac R, et al. Human genome sequencing using unchained base reads on self-assembling DNA nanoarrays [J]. Science, 2010, 327(5961): 78-81

[3] Harris T D, et al. Single-molecule DNA sequencing of a viral genome[J]. Science, 2008, 320(5872): 106-109

[4] Margulies M, et al. Genome sequencing inmicrofabricated high-density picolitre reactors[J]. Nature, 2005, 437(7075): 376-380

[5] McKernan K J, et al. Sequence and structural variation in a human genome uncovered by short-read, massively parallel ligation sequencing using two-base encoding[J]. Genome Res, 2009, 19(9): 1527-1541

[6] Medvedev P, et al. Paired de Bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers[M]// Research in Computational Molecular Biology. Springer, 2011: 238-251

[7] Hernandez D, Francois P, Farinelli L, et al. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer[J]. Genome Res. , 2008, 18(5): 802-809

[8] Miller J R, Koren S, Sutton G. Assembly algorithms for next-generation sequencing data[J]. Genomics, 2010, 95(6): 315-327

[9] Warren R L, Sutton G G, Jones S J, et al. Assembling millions of short DNA sequences using SSAKE[J]. Bioinformatics, 2007, 23(4): 500-501

[10] Dohm J C, Lottaz C, Borodina T, et al. A fast and highly accurate short-read assembly algorithm for de novo genomic sequencing[J]. Genome Res, 2007, 17(11): 1697-1706

[11] Jeck W R, Reinhardt J A, Baltrus D A, et al. Extending assembly of short DNA sequences to handle error[J]. Bioinformatics, 2007, 23(21): 2942-2944

[12] http://linux1. softberry. com/berry. phtml? topic=OligoZip

[13] Bresler M, Sheehan S, Chan A H, et al. Telescoper: De novo Assembly of Highly Repetitive Regions[J]. Bioinformatics, 2012, 28(18): 311-317

[14] Chaisson M J P, et al. De novo fragment assembly with short mate-paired reads: does the read length matter? [J]. Genome Res, 2009, 19(2): 336-346

[15] MacCallum I, et al. ALLPATHS 2: small genomes assembled accurately and with high continuity from short paired reads[J]. Genome Biol, 2009, 10(10): R103

[16] Simpson J T, et al. ABySS: a parallel assembler for short-read sequence data[J]. Genome Res, 2009, 19(6): 1117-1123

[17] Zerbino D R, Birney E. Velvet: algorithms for de novo short-read assembly using de Bruijn graphs[J]. Genome Res, 2008, 18(5): 821-829

[18] Li R, et al. De novo assembly of human genomes with massively parallel shortread sequencing[J]. Genome Res, 2010, 20(2): 265-272

[19] Earl D A, et al. Assemblathon 1: a competitive assessment of de novo short-read assembly methods[J]. Genome Res, 2011, 21(12): 2224-2241

[20] Miller J, Koren S, Sutton G. Assembly algorithms for next-generation sequencing data[J]. Genomics, 2010, 95(6): 315-27