基于动态分数阶和 Alpha 稳定分布的粒子群优化算法

吕太之[1] 李 卓[2]
(南京理工大学计算机科学与技术学院 南京 210094)[1]
(加州大学默塞德分校工程学院 加州默塞德 95340)[2]

摘 要 针对传统粒子群优化算法(PSO)收敛速度慢及容易陷入局部极小化的问题,提出了一种改进的粒子群优化算法。新算法结合分数阶微分具有的记忆特性,使得粒子的更新融入了轨迹信息,提高了算法的收敛速度。使用 Alpha 稳定分布代替均匀分布使得粒子在一定概率条件下可以逃逸局部极小点,提高了粒子的全局搜索能力。仿真结果表明,算法不仅在单模态函数下具有更快的收敛速度和更有效的全局搜索能力,在复杂的具有欺骗性的多模态函数下也取得较理想的实验结果,证实了动态分数阶和 Alpha 稳定分布可以有效地提高粒子群优化算法的性能。

关键词 粒子群,优化算法,分数阶,Alpha 分布
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.07.051

Novel Particle Swarm Optimization Algorithm Based on Fractional Calculus and Alpha-stable Distribution

LV Tai-zhi[1] LI Zhuo[2]
(College of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China)[1]
(School of Engineering, University of California, Merced 95340, America)[2]

Abstract For the traditional particle swarm optimization(PSO) algorithm converges slowly and it is easy to fall into local minimum point, an improved PSO algorithm was proposed. The new algorithm combines memory character of fractional differential, reflects the historical information of particles' movement and therefore improves the optimization process. Using Alpha-stable distribution instead of uniform distribution to generate random value can make the particle to escape from local minima in a certain probability and therefore there is more effective global search capability in new algorithm. Simulation results show that there is not only faster convergence speed and more effective global search capability under the single function in new algorithm, but also more satisfactory results under a complex and deceptive function. It is confirmed that the Alpha-stable distribution and fractional calculus can improve the performance of the PSO algorithm.

Keywords Particle swarm, Optimization algorithm, Fractional calculus, Alpha-stable distribution

1995 年 Kennedy 和 Eberbar 提出了一种基于模拟鸟群和鱼群社会行为的搜索算法——粒子群优化算法(Particle Swarm Optimization PSO)[1]。由于 PSO 算法简单,容易实现,具有很强的通用性,因此其应用领域非常广泛,涵盖了生物医学、金融、工程设计、电子、自动控制、机器人、信号处理、图像处理等多个领域[2,3]。PSO 算法在寻优过程中受粒子的个体最优和全局最优位置影响,存在着早熟收敛和收敛较慢的问题。为了避免算法早熟收敛,有的研究者提出了融合模拟退火算法[4]以及遗传算法"变异"操作[5]来增强全局搜索性能。文献[6]提出了基于扩散机制的杂交 PSO 算法,以有效改善粒子群优化算法的收敛速度。显然这些算法提高了 PSO 算法的全局寻优能力,但是很难在提高搜索速度和提高全局搜索能力之间达到平衡。文献[7]首次提出将分数阶融入到 PSO 算法中,使得粒子在更新过程中融入了轨迹信息,提高了算法的收敛速度。为了提高 PSO 算法的收敛速度和全局搜索能力,本文提出了基于动态分数阶和 Alpha 稳定分布的粒子群优化(Dynamic fractional calculus and alpha-stable distribution, DFC&ASD, PSO)算法。新算法在文献[7]方法上加以改进,将固定分数阶的阶数改成根据粒子状态和最优粒子轨迹信息动态调整的阶数,同时引入 Alpha 稳定分布产生的随机数代替均匀分布的随机数,使得粒子具备逃逸局部极小值的能力,提高了算法的全局搜索能力。为了验证算法的有效性,引入了 12 个经典函数问题(包括简单和复杂函数、单模态和多模态函数)进行数值分析,并与基本 PSO、文献[5]、文献[7]算法进行对比分析,结果显示改进算法收敛速度更快,搜索结果更有效。本文选择的测试函数与文献[8]中 3 个单模态函数和 4 个多模态函数相同。在该文献中,作者将文中提出的 APSO 方法和其他 7 个 PSO 算法(GPSO, LPSO,

到稿日期:2013-09-16 返修日期:2013-12-08 本文受江苏高校科研成果产业化推进项目(JHZD2012-21),第二届江苏省高校优秀中青年教师和校长境外研修项目资助。
吕太之(1979-),男,博士生,高级工程师,主要研究方向为人工智能、机器人自主导航,E-mail:lvtaizhi@163.com;李 卓(1986-),博士生,主要研究方向为分数阶控制。

VPSO, FIPS, HPSO-TVC, DMS-PSO, CLPSO)进行了对比。本文提出的算法与 APSO 方法相比，测试结果各有优劣，但对比其他 7 种 PSO 算法还是有很大的改进。

1 标准粒子群优化算法

考虑 N 维全局优化问题 $\min\{f(x);x\in\Omega\subset R^n\}$，PSO 算法使用一组集合 $S=\{X_1,X_2,X_3,\cdots,X_M\}$ 来求解该问题。这组集合称为种群(swarm)，集合中每一个元素称为粒子(particle)，集合的长度 M 称为种群的规模(size)。种群中每个粒子根据自己已经到达过的最优位置和整个种群到达过的最优位置来改变自己的位置和速度。经过若干次迭代，这些粒子按照一定的速度来寻找全局最优解。粒子群在 t 时刻的状态描述如下：

粒子群：$S_t=\{X_{t,1},X_{t,2},X_{t,3},\cdots,X_{t,M}\}$

第 i 个粒子的位置：$X_{t,i}=\{x_{t,i1},x_{t,i2},x_{t,i3},\cdots,x_{t,in}\}$

第 i 个粒子的速度：$V_{t,i}=\{v_{t,i1},v_{t,i2},v_{t,i3},\cdots,v_{t,in}\}$

第 i 个粒子的个体最优位置：

$pBest_{t,i}=\{x_{t,i1},x_{t,i2},x_{t,i3},\cdots,x_{t,in}\}$

全局最优位置：$gBest_t=\{x_{t,1},x_{t,2},x_{t,3},\cdots,x_{t,n}\}$

$t+1$ 时刻粒子的位置和速度通过式(1)和式(2)来更新：

$$V_{t+1,i}=\underbrace{w\times v_{t,i}}_{\text{惯性部分(}momentum\text{)}}+\underbrace{C_1\times rand()\times(pBest_{t,i}-X_{t,i})}_{\text{认知部分(}Cognitive\ Component\text{)}}+\underbrace{C_2\times rand()\times(gBest_t-X_{t,i})}_{\text{社会部分(}Social\ Component\text{)}}\tag{1}$$

$$X_{t+1,i}=X_{t,i}+V_{t+1,i}\tag{2}$$

惯性部分为粒子先前行为的惯性，其中 w 为惯性权重。认知部分表示粒子本身的思考，社会部分表示粒子间的信息共享与相互合作。C_1，C_2 为加速常数，$rand()$ 为基于[0,1]均匀分布的随机函数。

2 分数阶微分和 Alpha 稳定分布

2.1 分数阶微分

分数阶微积分的定义有多种，Grünwald-Letnikov[9] 是应用最为广泛的一种，其定义如下。

$$aD_a^t=\lim_{h->0}\frac{1}{h^\alpha}\times\sum_{k=0}^{[\frac{t-a}{h}]}(-1)^k\times\binom{\alpha}{k}\times f(t-k\times h)\tag{3}$$

$$\binom{\alpha}{k}=\frac{\Gamma(\alpha+1)}{\Gamma(k+1)\times\Gamma(\alpha-k+1)}\tag{4}$$

其中，aD_a^t 表示函数 $f(t)$ 的 α 阶微积分，α 为负是分数阶积分，为正则是分数阶微分。如果时间 t 是离散的，其近似实现如下：

$$aD_a^t=\frac{1}{T^\alpha}\times\sum_{k=0}^{r}(-1)^k\times\binom{\alpha}{k}\times f(t-k\times T)\tag{5}$$

其中，T 是采样区间，r 是截断次数。

2.2 Alpha 稳定分布

Alpha 稳定分布又称分形分布，其分布通过尺度因子、特性指数、移位参数和偏度参数来表示，其概率密度函数可以用特征函数的连续傅立叶变换来定义[10]。

$$f(x;\alpha,\beta,\sigma,\mu)=\frac{1}{2\pi}\times\int_{-\infty}^{\infty}\exp[it\mu-|\sigma t|^\alpha(1-i\beta\mathrm{sgn}(t))\Phi]e^{-itx}dt\tag{6}$$

其中，sng(t)是 t 的符号，Φ 表示为

$$\Phi=\begin{cases}\tan(\pi\alpha/2), & \alpha\neq1\\ -(2/\pi)\log|t|, & \alpha=1\end{cases}\tag{7}$$

3 DFC&ASD PSO 算法

3.1 基于 Alpha 稳定分布的随机函数

DFC&ASD PSO 算法使用 Alpha 稳定分布代替式(1)中的随机函数。新算法采用文献[11,12]中的方法来生成随机数。随机数生成流程如下：

首先生成两个独立的随机变量 V 和 W。V 是一个在 $[-\frac{\pi}{2},\frac{\pi}{2}]$ 区间上均匀分布的变量，W 是一个均值为 1 的指数分布变量。

然后根据 V 和 W 生成服从 $S(\alpha,\beta,1,0)$ 的随机变量 X。

$$X=\begin{cases}S_{\alpha,\beta}\times\dfrac{\sin(\alpha(V+B_{\alpha,\beta}))}{(\cos(V))^{\frac{1}{\alpha}}}\times\left(\dfrac{\cos(V-\alpha(V+B_{\alpha,\beta}))}{W}\right)^{\frac{1-\alpha}{\alpha}}, & \alpha\neq1\\ \dfrac{2}{\pi}\left[\left(\dfrac{\pi}{2}+\beta V\right)\tan(V)-\beta\log\left(\dfrac{W\cos(V)}{\frac{\pi}{2}+\beta}\right)\right], & \alpha=1\end{cases}\tag{8}$$

$S_{\alpha,\beta}$ 和 $B_{\alpha,\beta}$ 定义如下：

$$S_{\alpha,\beta}=[1+\beta^2\tan^2(\frac{\pi\alpha}{2})]^{\frac{1}{2\alpha}}\tag{9}$$

$$B_{\alpha,\beta}=\frac{\arctan(\beta\tan(\frac{\pi\alpha}{2}))}{\alpha}\tag{10}$$

最后根据随机变量 X 生成服从 $S(\alpha,\beta,\sigma,\mu)$ 的随机变量 Y。

$$Y=\begin{cases}\sigma X+\mu, & \alpha\neq1\\ \sigma X+\dfrac{2}{\pi}\beta\sigma\log\sigma+\mu, & \alpha=1\end{cases}\tag{11}$$

本文采用 $\alpha=1.5$ 的 Lévy 分布，这种分布介于高斯分布 $\alpha=2.0$ 和柯西分布 $\alpha=1.0$ 之间。该分布可不同于标准算法中 [0,1]的均匀分布，基于该分布的随机函数在一定的概率条件 下可以获得较大的值，从而使得粒子有机会逃逸局部极小点，扩大其搜索范围。

3.2 基于动态分数阶的速度计算

DFC&ASD PSO 算法使用分数阶微分来更新粒子速度。新算法中惯性权重设置为 1，随机函数采用 3.1 节的方法，式(1)可以改写为下面的形式。

$$V_{t+1,i}-V_{t,i}=C_1\times stbrand()\times(pBest_{t,i}-X_{t,i})+C_2\times stbrand()\times(gBest_t-X_{t,i})\tag{12}$$

左边的 $V_{t+1,i}-V_{t,i}$ 是一阶差分，由于在 PSO 算法中，粒子飞行是离散时间的，其最小间隔 $T=1$，其差分形式如下：

$$aD_a^t=C_1\times stbrand()\times(pBest_{t,i}-X_{t,i})+C_2\times stbrand()\times(gBest_t-X_{t,i})\tag{13}$$

根据式(4)和伽玛函数递推公式，差分公式前 5 项的近似表达式整理如下。

$$V_{t+1,i}=a\times V_{t,i}+\frac{1}{2}\times a\times V_{t-1,i}+\frac{1}{6}\times a\times(1-\alpha)\times V_{t-2,i}+\frac{1}{24}\times\alpha\times(1-\alpha)\times(2-\alpha)\times V_{t-3,i}+\frac{1}{120}\times\alpha\times(1-\alpha)\times(2-\alpha)\times(3-\alpha)\times V_{t-4,i}+C_1\times stbrand()\times(pBest_{t,i}-X_{t,i})+C_2\times stbrand()\times(gBest_t-X_{t,i})\tag{14}$$

为了使粒子具有扩展搜索空间的能力，阶数 α 根据粒子

的状态和最优粒子的轨迹信息动态调整。阶数 α 的初始值为 0.5，上下限为[0.4,0.8]，每隔 20 步迭代调整一次 α，调整过程如下：

1)计算每个粒子与其他粒子间距离的总和。

$$d_i=\frac{1}{M-1}\sum_{j=1,j+i}^{M}\sqrt{\sum_{k=1}^{N}(x_{i,k}-x_{j,k})^2} \tag{15}$$

2)参照式(15)计算最优粒子和各个粒子距离的总和 d_{gbest}。

3)根据粒子最优轨迹信息确定 β 值。如果粒子最优位置 20 步没有发生变化，β 值加 0.1；否则 β 值为 0。

4)计算分数阶微分的阶数。

$$\alpha=\frac{0.55}{1+1.5e^{-2.6\times|\frac{d_{gbest}-d_{\min}}{d_{\max}-d_{\min}}|}}+\beta \tag{16}$$

5)根据上下限调整阶数。

3.3 算法详细流程

DFC&ASD PSO 算法的实现步骤如下：

1)初始化

完成各类参数的设定。设定粒子群规模 M 和最大迭代次数、搜索空间的上下限、学习因子 C_1 和 C_2；随机生成 M 个初始粒子和初始速度；个体最优位置为当前粒子的位置；全局最优就是个体最优中最好的。

2)粒子的评价

计算每个粒子的适应度值。若该值优于当前粒子个体最优值，将该值设置为新的个体最优值。根据各个粒子的个体最优值找出全局最优值。

3)速度更新

使用 3.1 节的方法生成随机数，利用 3.2 节介绍的方法更新粒子的速度。

4)粒子更新

根据式(2)更新粒子当前的位置。如果粒子的位置超出了搜索空间，将其设置为搜索空间的边界值。

5)检验是否符合结束条件

判断是否达到终止条件，若达到，则记录全局最优值及粒子位置；否则转向步骤 2)。

4 实验仿真

为了验证改进算法的性能，本文选择了 PSO 及其改进算法[5,7]进行对比试验，引入了 PSO 和遗传算法中经常使用的 12 个函数问题[13,14]进行数值分析。对于基本 PSO 和文献[5,7]的算法，惯性权重在[0.4,0.95]之间随代数线性递减，DFC&ASD PSO 算法惯性权重设置为 1，分数阶微分的阶数在[0.4,0.8]之间动态变化。所有算法中加速常数 $C_1=2$，$C_2=2$。实验函数的种群规模设置为 20，每次运行迭代 6000 次。函数评价次数为 1.2×10^5，搜索中如果全局最优值每相隔 100 次迭代后变化小于 1e−160 则停止搜索。每个函数独立搜索运行 30 次。

表 1 给出了 6 个单模态函数，该类函数都是连续的，在区间上呈现凸状。Sphere 函数是一个简单的平方函数，只有一个极小点 $f(0,0,0)=0$。Axis parallel hyper-ellipsoid，Tablet 和 Schwefel 1.2 函数是 Sphere 函数的变形，增加了各维函数之间的相互作用。Rosenbrock 虽然只有一个全局极小点 $f(1,1)=0$，但它却是病态的，在极值周围梯度极小，难以求得全局极小值。

表 2 给出了 6 个典型的非线性多模态函数 f_7-f_{12}，大量的局部极小点被认为是遗传算法很难处理的复杂多模态问题。

表 1 f_1-f_6 单模态函数

测试函数	维度	搜索空间	全局最优	函数名称
$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]^n$	0	Sphere
$f_2(x)=\sum_{i=1}^{n}i\times x_i^2$	30	$[-5.12,5.12]^n$	0	Axis parallel hyper-ellipsoid
$f_3(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$	30	$[-100,100]^n$	0	Quadric
$f_4(x)=\sum_{i=1}^{n-1}100\times(x_{i+1}-x_i^2)^2+(1-x_i)^2$	30	$[-10,10]^n$	0	Rosenbrock
$f_5(x)=\sum_{i=1}^{n}\lvert x_i\rvert^{(i+1)}$	30	$[-1,1]^n$	0	Sum of different power
$f_6(x)=10^5\times x_1^2+\sum_{i=2}^{n}x_i^2$	30	$[-100,100]^n$	0	Tablet

表 2 f_7-f_{12} 多模态函数

测试函数	维度	搜索空间	全局最优	函数名称
$f_7(x)=\sum_{i=1}^{n}-x_i\times\sin(\sqrt{\lvert x_i\rvert})$	30	$[-500,500]^n$	−12569.5	Schwefel
$f_8(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	$[-600,600]^n$	0	Griewank
$f_9(x)=10\times n+\sum_{i=1}^{n}(x_i^2-10\times\cos(2\times\pi\times x_i))$	30	$[-5.12,5.12]^n$	0	Rastrigin
$f_{10}(x)=-20\times e^{-0.2\times\sqrt{\frac{\sum_{i=1}^{n}x_i^2}{n}}}-e^{\frac{\sum_{i=1}^{n}\cos(2\times\pi\times x_i)}{n}}+a+e^1$	30	$[-32,32]^n$	0	Ackley
$f_{11}(x)=-\sum_{i=1}^{n}\sin(x_i)\times(\sin(\frac{i\times x_i^2}{\pi}))^{20}$	30	$[0,\pi]^n$		Michalewicz
$f_{12}(x)=\sum_{i=1}^{n-1}(x_i^2+x_{i+1}^2)^{0.25}\times[\sin(50\times((x_i^2+x_{i+1}^2)^{0.1})+1.0]$	30	$[-100,100]^n$	0	Schaffer f7

图 1 和图 2 显示了单模态 Quadric 和多模态 Ackley 函数全局最优值收敛曲线。DFC&ASD PSO 算法能够始终保持较快的收敛速度，持续有效地搜索全局最优点。从图 3 中可以看出 DFC&ASD 的收敛速度要远远快于其他函数，搜索终止于第 1462 次迭代，平均次数为 1584。PSO 算法的平均迭代次数为 5671，文献[5]算法的迭代次数为 1946，文献[7]为 3899 次。对于大部分多模态函数，DFC&ASD 的实际执行次数少于 6000 次，如 Rastrigin 函数的 1622 次，Michalewicz 的 3201 次。

表 3 显示 6 个单模态函数的对比实验结果。从算法的均值和方差可以看出，DFC&ASD PSO 算法要优于其他 3 种算法。对于复杂单模态的 Rosenbrock 函数，它是一个经典复杂优化问题，它的全局最优点位于一个平滑、狭长的抛物线山谷里面，因此找到全局最优点的机会微乎其微。虽然算法在

Rosenbrock 函数找到的最优结果是 7.5054e－8，相比标准 PSO 算法的 4.1936、文献[5]的 0.0124、文献[7]的 0.9478 要好得多，也有 10%的概率找到 1e－5 以下的最优值，但是该函数由于很难辨别方向，难以像其他单模函数那样找到更接近于 0 的值。对比文献[8]的 3 个单模态测试函数，Rosenbrock 函数和 Quadric 函数优于 APSO 算法，Sphere 函数不如 APSO 算法。DFC&ASD PSO 的搜索结果远优于其他 7 种算法。

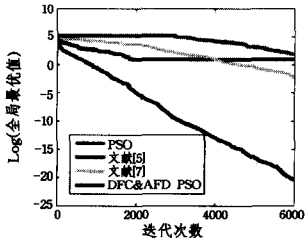

图 1　30 维 Quadric 函数收敛曲线

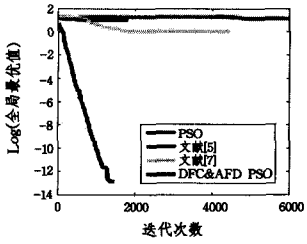

图 2　30 维 Ackley 函数收敛曲线

表 4 显示 6 个多模态函数的对比实验结果。从算法的均值和方差可以看出，DFC&ASD PSO 算法要优于其他 3 种算法。对于 30 维的 Griewank 函数，DFC&ASD PSO 算法有 11 次(35%)找到全局最优值 0，文献[7]的方法有 6 次(12%)找到全局最优值 0，基本 PSO 有 4 次，文献[5]算法没有找到全局最优值，可以看出其全局搜索能力显著优于 PSO 及其它改进算法，能够有效逃出全局最小点。DFC&ASD PSO 算法虽然加快了收敛速度但同时也提高了陷入局部极小点的可能性。动态调整阶数和使用 Alpha 稳定分布，不仅使得算法有一定概率逃逸局部极小点，也提高了算法的收敛速度。从图 2 和表 4 可以看出 DFC&ASD PSO 算法比文献[7]的算法有更快的收敛速度和更好的搜索效果。对比文献[8]中的 4 个多模态函数，在 Schwefel 函数下两个算法基本一致，对于 Griewank 函数，DFC&ASD PSO 优于 APSO 算法，Ackley 算法略低于 APSO 算法，Rastrigin 算法 APSO 的搜索结果均值是 5.8×10^{-15}，差距较大。除了 CLPSO 算法，在 Rastrigin 函数下的测试结果表明，DFC&ASD PSO 均优于其他 8 种 PSO 算法。

表 3　DFC&ASD PSO 和其他 PSO 算法在单模态函数下的实验对比

算法 \ 函数		f_1	f_2	f_3	f_4	f_5	f_6
PSO 算法	均值	1.0489e－34	2.5929e－26	70.4512	65.0803	9.8391e－35	1.4647e－24
	标准差	4.4784e－34	9.2668e－26	100.811	41.5124	5.3891e－34	2.4820e－24
文献[5]算法	均值	1.9827e－38	2.4149e－04	47.2532	52.85921	9.1823e－40	2.3532e－34
	标准差	1.0526e－37	1.9324e－04	29.9711	30.1952	5.9336e－44	3.1846e－32
文献[7]算法	均值	5.2376e－45	3.1235e－49	3.0653e－02	39.4322	1.2124e－01	1.4362e－42
	标准差	2.1163e－43	1.3829e－48	7.6793e－02	37.0615	2.7106e－01	3.3343e－42
DFC&ASD	均值	9.9113e－104	2.6418e－103	5.1666e－12	1.1459	2.7277e－164	2.5803e－100
PSO 算法	标准差	5.4286e－103	1.0399e－103	2.5095e－12	2.1638	1.177e－166	1.4133e－99

表 4　DFC&ASD PSO 和其他 PSO 算法在多模态函数下的实验对比

算法 \ 函数		f_7	f_8	f_9	f_{10}	f_{11}	f_{12}
PSO	均值	－7708.40	1.7099e－02	44.3751	9.0059	－8.2561	76.3290
	标准差	861.3414	2.9557e－02	7.9472	6.6778	1.0989	12.8546
文献[5]	均值	－12002.30	4.9051e－01	58.9224	4.4905	－6.2957	65.9342
	标准差	569.4132	5.7931e－01	9.6579	3.8359	18.2864	6.1247
文献[7]	均值	－8534.302	2.8496e－02	52.1351	9.8883e－01	－16.7878	73.5297
	标准差	717.0181	2.4851e－02	21.6914	3.7007e－01	5.0783	7.9845
DFC&ASD	均值	－12569.4937	1.3211e－02	1.1939	9.0594e－11	－27.52043	39.4445
PSO 算法	标准差	2.1417e－15	1.6793e－02	1.3302	2.8221e－11	1.4768	7.5143

结束语　在标准粒子群优化算法的基础上，DFC&ASD PSO 算法使用动态分数阶和 Alpha 稳定分布计算粒子速度，将分数阶融入到 PSO 算法中，充分利用了粒子群在寻优过程中的历史信息。动态的惯性权重有利于种群搜索，同样动态分数阶的阶数也有利于提高种群搜索能力。使用 Alpha 稳定分布代替标准算法中的均匀分布产生随机数，使得粒子具备逃逸局部最小值的能力。实验证明，算法在大部分函数中收敛速度都很快，获得了较好的理想的结果。但是对于部分函数问题仍然存在不能逃出局部极小值点的问题，同时该算法仅仅提高了个体粒子的搜索性能，没有充分利用种群间的相互信息。所以，进一步工作主要体现在以下几个方面：

(1)研究种群间的相互信息，提高算法全局搜索能力。

(2)根据算法所在的不同的搜索阶段，动态调整 Alpha 稳定分布的参数。

(3)在粒子陷入局部极小点的时候，是否有更有效的方式逃逸。

参 考 文 献

[1] Poli, Riccardo, Kennedy J, et al. Particle swarm optimization [J]. Swarm intelligence, 2007, 1: 33-57

[2] Poli, Riccardo. Analysis of the publications on the applications of particle swarm optimization [J]. Journal of Artificial Evolution and Applications, 2008, 3: 1-10

[3] 赵玉静. 改进的粒子群优化算法及应用[D]. 广州：华南理工大学，2011

[4] 焦巍，刘光斌，张艳红. 求解约束优化的模拟退火 PSO 算法[J]. 系统工程与电子技术，2010，32(7)：1532-1538

[5] Xie B, Chen S, Liu F. Biclustering of Gene Expression Data Using PSO-GA Hybrid[C]// The 1st International Conference on Bioinformatics and Biomedical Engineering(ICBBE 2007). 2007: 302-305

(下转第 289 页)

[10] Breiman L. Random Forests [J]. Machine Learning, 2001, 45 (1):5-32

[11] Zhang D, Chen S, Zhou Z-H, et al. Constraint projections for ensemble learning[C]// Fox D, Gomes C P, eds. Proceedings of the 23rd AAAI Conference on Artificial Intelligence. Chicago, IL, 2008:758-763

[12] Hecherman D. Bayesian Networks for Data Mining [J]. Data Mining and Knowledge Discovery, 1997, 1(1):79-119

[13] Lin H-T, Li L. Support Vector Machinery for Infinite Ensemble Learning[J]. Journal of Machine Learning Research, 2008, 9: 285-312

[14] Quinlan J R. C4. 5: Programs for Machine Learning [M]. Morgan-kaufmann Publisher, San Mateo, CA, 1993

[15] Breiman L, Friedman J H, Olshen R, et al. Classification and Regression Trees[M]. London: Chapman and & Hall, 1993

[16] Ho T K. The Random Subspace Method for Constructing Decision Forests[J]. IEEE Trans. Pattern Analysis and Machine Intelligence, 1998, 20(8):832-844

[17] Schapire R E, Freund Y, Bartlett P, et al. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods [J]. Annals of Statistics, 1998, 26(5):1651-1686

[18] Tan P, Steinbach M, Kumar V. 数据挖掘导论[M]. 范明, 范宏建, 等译. 北京: 人民邮电出版社, 2008

[19] Rodríguez J J, Kuncheva L I, Alonso C J. An Experimental Study on Rotation Forest Ensembles[C]// MCS 2007. LNCS 4472, 2007:459-468

[20] Rudin C, Schapire R E, Daubechies I. Open Problem: Does AdaBoost Always Cycle?[J]. Journal of Machine Learning Research, 2012, 23:1-4

[21] http://nlp.zzu.edu.cn/LySpoon.asp

(上接第 245 页)

搜索长路径，由于两方搜索是交替进行的，如果一方搜索结束，另一方搜索则很可能已经找到最短路的从起点开始(或到终点)的子路。因此，BiBFS 对于长路径查询有高正确率。

3. 结果分析

实验结果表明，对于大规模图中的长路径查询，双向优先搜索有较高的运行效率和正确率。在路网中，一般来说，起始点和终止点的地理位置越远，其最短路径越可能是长路径。启动查询时，可先检查起止点的地理位置，若相距较远，则可采用双向广度优先搜索；反之，则采用单向广度优先搜索。

结束语 最短路径查询是图研究中的一个经典问题。目前大部分研究假设图中每条边只有一种权值。然而，有些应用需要考虑图中每条边有多种权值。多权值路网中，最短路的子路不一定也是最短路，使得单权值路网中的大部分求解最短路算法不适用。本文提出了一种双向广度优先搜索方法及剪枝策略，用以求解多权值路网中的最短路径近似解。实验表明，该算法适用于长路径搜索。与单向广度优先搜索相比，该算法有更快的运行效率；与基于 Dijkstra 算法的贪心算法相比，该算法有更高的准确率。在路网中查询最短路径时，若起始点和终止点的地理位置相距较远，则适合采用双向广度优先搜索求解。

参 考 文 献

[1] Geisberger R, Sanders P, Schultes D, et al. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks[C]// WEA. 2008:319-333

[2] Abraham I, Fiat A, Goldberg A V, et al. Highway dimension, shortest paths, and provably efficient algorithms[C]// SODA. 2010:782-793

[3] Bast H, Funke S, Matijecvic D. Transit: ultrafast shortest-path queries with linear time preprocessing[C]// Proc. of the 9th DIMACS Implementaion Challenge. 2006:175-192

[4] 林澜, 闫春钢, 蒋昌俊, 等. 动态网络最短路问题的复杂性与近似算法[J]. 计算机学报, 2007(4):608-604

[5] 王树西, 吴政学. 改进的 Dijkstra 最短路径算法及其应用研究[J]. 计算机科学, 2012, 39(5):223-228

[6] 黄贵玲, 高西全, 靳松杰, 等. 基于蚁群算法的最短路径问题的研究和应用[J]. 计算机工程与应用, 2007, 43(13):233-235

[7] 戴树贵, 孙强, 潘荫荣. 带限制条件的多权最短路径近似算法[J]. 计算机工程, 2003, 29(7):88-91

[8] Yang Ya-jun, J Xu-yu, Gao Hong, et al. Finding the optimalpath overmulti-cost graphs[C]// CIKM'12. ACM, New York, NY, USA, 2012:2124-2128

[9] Dijkstra E W. A note on two problems in connection with graphs [J]. Numerical Mathematics, 1959(1):269-271

[10] Pohl I. Bi-direcitonal search[J]. Machine Intelligence, 1971(6): 128-140

(上接第 249 页)

[6] 徐星, 吴昱. 基于扩散机制的杂交粒子群优化算法[J]. 计算机应用研究, 2011, 11:4156-4159

[7] Pires, Solteiro E J, et al. Particle swarm optimization with fractional-order velocity[J]. Nonlinear Dynamics, 2010, 61(1/2): 295-301

[8] Zhan Zhi-hui, et al. Adaptive particle swarm optimization [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2009, 39(6):1362-1381

[9] Pan, Indranil, Das S. Brief Introduction to Computational Intelligence Paradigms for Fractional Calculus Researchers[M]// Intelligent Fractional Order Systems and Control. Springer Berlin Heidelberg, 2013:63-85

[10] 杨伟超. Alpha 稳定分布噪声下通信信号调制识别研究[D]. 哈尔滨: 哈尔滨工程大学, 2012

[11] Weron A, Weron R. Computer simulation of Lévy α-stable variables and processes[M]. Springer Berlin Heidelberg, 1995: 379-392

[12] 吕晓蕊. Alpha 稳定分布随机变量的产生[J]. 计算机与数字工程, 2012, 40(3):32-34

[13] GEATbx: Example Functions (single and multi-objective functions)[EB/OL]. http://www.geatbx.com/docu/ fcnindex-01.html#P86_3059

[14] Xin Yao, Liu Yong, Lin Guang-ming. Evolutionary programming made faster[J]. IEEE Transactions on Evolutionary Computation, 1999, 3(2):82-102