基于三角范数的引力搜索算法分析

徐 遥 安亚静 王士同
(江南大学数字媒体学院 无锡 214122)

摘 要 分析了由Esmat Rashedi提出的引力搜索算法(GSA)之后,对万有引力公式进行变换,用三角范数的其他算子代替万有引力公式中两个粒子惯性质量之间的乘法算子。分析不同三角范数算子的二维图像的特征之后,选择了三角范数中的5个算子进行实验。实验结果表明,对于具有一定三维图像特征的测试函数,使用相应三角范数算子的引力搜索算法对其全局搜索的能力相对地好于使用其它三角范数算子的改进引力搜索算法。

关键词 引力搜索算法,万有引力公式,三角范数算子,二维的图像特征
中图法分类号 TP391 **文献标识码** A

Analyzing of Gravitational Search Algorithm Based on Triangular Norms

XU Yao AN Ya-jing WANG Shi-tong
(School of Digital Media, Jiangnan University, Wuxi 214122, China)

Abstract To transform universal gravitational formula after analyzing Gravitational Search Algorithm(GSA) Esmat Rashedi proposed, the product operator between two particles's inertia quality in the universal gravitation formula ws substituted by other triangular norms operator. Five triangular norms operators were chose for experience after analyzing feature of two-dimensional image to different triangular norms operators. Results show: to a test function having certain three-dimensional image feature, ability to find the global optimum using GSA of corresponding triangular norms operator is relatively better than GSA of others triangular norms operators.

Keywords Gravitational search algorithm, Universal gravitational formula, Triangular norms operator, Feature of two-dimensional image

1 引言

引力搜索算法(GSA)是一种群体智能优化算法,是2008年由Esmat Rashedi等人受到万有引力公式的启发而提出来的。文献[1]中使用了23个经典测试函数对GAS进行测试,和其它的经典优化算法(由Eberhart博士和Kennedy博士提出的PSO算法[2]和Formato提出的CFO算法[3])相比较,对于大部分测试函数来说,GSA算法的收敛速度更快,而且最后搜索结果的精度也更好。

和PSO算法一样,引力搜索算法的初始位置和初始速度都是随机生成的。首先根据式(1)和式(2),我们计算出每个粒子的惯性质量 $M_i(t)$:

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

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

式中,$fit_i(t)$是t时刻粒子i的适应值,$best(t)$和$worst(t)$是分别在t时刻粒子群的最好适应值和最坏适应值。

对于求最小值问题,$best(t)$和$worst(t)$的定义如下:

$$best(t)=\min_{i\in\{1,\cdots,N\}} fit_i(t) \tag{3}$$

$$worst(t)=\max_{i\in\{1,\cdots,N\}} fit_i(t) \tag{4}$$

对于求最大值问题,$best(t)$和$worst(t)$的定义如下:

$$best(t)=\max_{i\in\{1,\cdots,N\}} fit_i(t) \tag{5}$$

$$worst(t)=\min_{i\in\{1,\cdots,N\}} fit_i(t) \tag{6}$$

其次,变换万有引力公式后,可以计算各个粒子在每一维空间上相互之间的引力。比如在第d维空间上粒子i和粒子j之间的引力$F_{ij}^d(t)$计算如下:

$$F_{ij}^d(t)=G(t)\frac{M_i(t)\times M_j(t)}{R_{ij}(t)+\varepsilon}\cdot(x_j^d(t)-x_i^d(t)) \tag{7}$$

$$G(t)=G_0 e^{-\alpha\frac{t}{T}} \tag{8}$$

式中,$R_{ij}(t)$表示粒子i到粒子j之间的欧式距离,$G(t)$表示引力常数。

然后再根据运动规律计算出各个粒子在每一维空间上t时刻的加速度。比如粒子i在第d维空间上的加速度$a_i^d(t)$计算如下:

$$a_i^d(t)=\frac{\sum_{i=1,j\neq i}^{N}\text{rand}\cdot F_{ij}^d(t)}{M_i(t)} \tag{9}$$

最后,下一时刻每个粒子的位置和加速度的更新公式如下:

到稿日期:2010-12-10 返修日期:2011-03-23 本文受国家自然科学基金(60773206,60704047)资助。

徐 遥(1985—),男,硕士,主要研究方向为人工智能与模式识别,E-mail:xuy2030@126.com;安亚静(1986—),女,硕士生,主要研究方向为人工智能与模式识别;王士同(1964—),男,博士生导师,主要研究方向为人工智能与模式识别、生物信息学。

$$v_i^d(t+1)=\text{rand}\times v_i^d(t)+a_i^d(t) \tag{10}$$

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1) \tag{11}$$

根据以上原理步骤，不断地迭代循环更新粒子群中每个粒子的位置和加速度，直到全局最小值 $best(t)$ 达到一定的精度或迭代达到规定的次数，跳出循环，$best(t)$ 就是所求的最优值。

2 基于三角范数的引力模糊集的构造

在本文中，使用三角范数其它不同的算子替换万有引力公式中的乘法算子，以此建立一个模糊集。第 2.1 节介绍了几种三角范数算子及其性质。第 2.2 节分析用各三角范数算子替换后的引力搜索算法。最后分析和解决使用有些三角范数算子时所遇到的加速度会无限大的问题。

2.1 三角范数的性质及其算子的选择

由文献[4-7]可知，三角范数是一个在单位区间[0,1]上的二元运算 $T(a,b):[0,1]\times[0,1]\to[0,1]$，它满足以下条件：

(1)交换律：$T(a,b)=T(b,a)$；

(2)单调性：如果 $b\leqslant c$，则 $T(a,b)\leqslant T(a,c)$；

(3)结合律：$T(T(a,b),c)=T(a,T(b,c))$；

(4)边界条件：$T(1,a)=a, T(0,a)=0$。

称满足以上 4 个条件者为 T-范数。

对于满足上面前 3 个条件并且满足以下条件者称为 T-余范数：

$T'(a,0)=a$，并且 $T'(a,1)=1$

在本文中，首先选取 3 个 T-范数算子，分别如下：

- $T_Z(a,b)=\min(a,b)$
- $T_P(a,b)=a\cdot b$
- $T_L(a,b)=\max(a+b-1,0)$

根据文献[4]中的定义 2，对于任意两个 T-范数算子 T_1 和 T_2，如果对于 $\forall x,y$，且有 $(x,y)\in[0,1]^2$，则有 $T_1(x,y)\leqslant T_2(x,y)$，那么可以认为在同等条件下 T_1 比 T_2 弱。对于以上 3 个 T 范数算子 T_Z、T_P 和 T_L，可以得出，在同等条件下，满足 $T_L\leqslant T_P\leqslant T_Z$，即 T_L、T_P 和 T_Z 3 个三角范数算子是越来越强。为了增加实验的对比性，又选取了另外两个算子：T-范数算子 T_{HD} 和 T-余范数算子 S_Z：

- $T_{HD}(a,b)=\begin{cases}0 & ,a=b=0\\ \dfrac{a\cdot b}{a+b-a\cdot b} & ,\text{otherwise}\end{cases}$
- $S_Z(a,b)=\max(a,b)$

在同等条件下，不难得出算子 T_{HD} 比算子 T_Z 要弱，但比算子 T_P 要强。算子 S_Z 虽然是 T-余范数算子，但在这里，我们只是从算子的强弱性上考虑，因此在同等条件下，算子 S_Z 比算子 T_Z 要强。在所选择的 5 个算子中，算子 S_Z 是最强的，算子 T_L 是最弱的。

2.2 三角范数各算子对引力搜索算法的替换与分析

根据文献[8,9]中的牛顿万有引力定律可知，任何一个物体都通过一种力吸引着其它所有的物体，这种力的大小与两个物体的惯性质量的乘积成正比，与物体之间的欧氏距离的大小成反比。这个力的计算公式如下：

$$\vec{f}_{1,2}=G\cdot\frac{m_1\cdot m_2}{\|\vec{r}\|^2}\cdot\frac{\vec{r}}{\|\vec{r}\|} \tag{12}$$

式中，G 为引力常数，m_1 和 m_2 为粒子的惯性质量，$\vec{r}$ 为粒子之间的距离矢量。

在本文中，由于所有粒子的惯性质量大小均在[0,1]之间，因此式(12)可以改写成如下形式：

$$\vec{f}_{1,2}=G\cdot\frac{T_P(m_1,m_2)}{\|\vec{r}\|^2}\cdot\frac{\vec{r}}{\|\vec{r}\|} \tag{13}$$

用其它三角范数算子 T 替换式(13)中的三角范数的乘积算子，得

$$\vec{f}_{1,2}=G\cdot\frac{T(m_1,m_2)}{\|\vec{r}\|^2}\cdot\frac{\vec{r}}{\|\vec{r}\|} \tag{14}$$

根据式(14)，可以把式(7)改写成如下形式：

$$F_{ij}^d(t)=G(t)\cdot\frac{T(M_i(t),M_j(t))}{R_{ij}(t)+\varepsilon}\cdot(x_j^d(t)-x_i^d(t)) \tag{15}$$

从式(15)可以看出，对于某个粒子，在 t 时刻，它的惯性质量与它和其它粒子之间的欧式距离以及引力常数都是一定不变的，因此它所受到的所有其它粒子对它的合力只与三角范数算子 $T(M_i(t),M_j(t))$ 的大小成正比。因此，只要知道三角范数算子的一些性质，那么几乎就知道了用一个三角范数算子替换后的引力搜索算法对一个函数搜索能力的好坏。

在某一个时刻，由于受力粒子的惯性质量不变，对于一个三角范数 $T(x,y)$，假设 y 为受力粒子的惯性质量，等于常数 a，x 为作用粒子的惯性质量，则 $T(a,x)=T(x,y)$。因此只要知道作用粒子的惯性质量 x 对三角范数 $T(x,y)$ 大小的影响，也就知道了作用粒子的惯性质量 x 对受力粒子引力大小的影响，所以有必要分析各三角范数算子的性质。为了便于分析，首先画出本文中所使用的各个三角范数 $T(a,x)$ 的二维图像，其中 $a=0.5$，如图 1 所示。

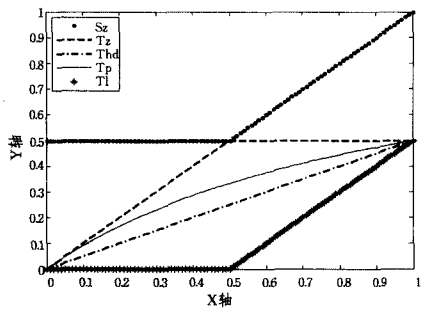

图 1 各算子二维特征图像

从图 1 可以看出，在三角范数各算子中，越弱的算子，其函数值相对也就越小；相反，越强的算子，其函数值相对也就越大。对于式(9)中的粒子加速度 $a_i^d(t)$，在引力搜索算法中，可以认为它是引力搜索算法的步长，它的大小决定了引力搜索算法的收敛速度和全局搜索的能力。从总体上说，加速度 $a_i^d(t)$ 的值越大，引力搜索算法相对越容易跳出局部最优值，但它的收敛速度相对比较慢，搜索到的结果精度相对也较差；相反，加速度 $a_i^d(t)$ 的值越小，引力搜索算法相对越难跳出局部最优值，但它的收敛速度相对比较快，搜索到的结果精度相对较高。但通过下面的实验结果表明，加速度 $a_i^d(t)$ 的值不能够太大，也不能够太小，否则改进后的引力搜索算法不收敛。由于受力粒子的受力大小和三角范数算子的函数值成正比，根据式(9)可知，三角范数算子 T 的函数值越大，粒子的加速度 $a_i^d(t)$ 的值相对也越大；三角范数算子 T 的函数值越小，粒

子的加速度 $a_i^d(t)$ 的值相对也越小。因此，对于本文所用到的5个三角范数算子 S_Z、T_Z、T_{HD}、T_P 和 T_L，用它们所替换的相对应的引力搜索算法跳出局部最优值的能力逐渐减小，收敛速度相对逐渐变快，搜索到的结果精度相对地也较高。但通过下面的实验表明，三角范数算子 S_Z 和 T_L 相对应替换的引力搜索算法对函数搜索全局最优值时，几乎不收敛。也就是说，在选取三角范数算子时，算子 T 不能够比 S_Z 更强，也不能够比 T_L 更弱。

2.3 加速度无限大问题的分析与解决

根据2.1节定义的三角范数可知，在本文中粒子的惯性质量的大小范围应该在[0,1]之间。假设粒子2受到来自粒子1的引力 $\vec{f}_{1,2}$ 作用，可以通过式(14)求出 $\vec{f}_{1,2}$。再根据运动定律可以求出粒子在引力 $\vec{f}_{1,2}$ 的作用下的运动加速度 $\vec{a}_2$：

$$\vec{a}_2=\frac{\vec{r}_{1,2}}{m_2} \tag{16}$$

文献[1]中使用的三角范数算子为乘积算子 T_P。假设粒子1、2的惯性质量分别为 m_1 和 m_2，根据式(14)和式(16)有

$$\begin{aligned}\vec{a}_2&=G\cdot\frac{T_P(m_1,m_2)}{\|\vec{r}\|^2}\cdot\frac{\vec{r}}{\|\vec{r}\|}\cdot\frac{1}{m_2}\\&=G\cdot\frac{m_1\times m_2}{\|\vec{r}\|^2}\cdot\frac{\vec{r}}{\|\vec{r}\|}\cdot\frac{1}{m_2}\\&=G\cdot\frac{m_1}{\|\vec{r}\|^2}\cdot\frac{\vec{r}}{\|\vec{r}\|}\end{aligned} \tag{17}$$

从上式可以看出，粒子2的惯性质量 m_2 在求加速度的过程中被约去了，因此不管粒子2的惯性质量 m_2 是否等于零，都不会出现粒子2的加速度等于无限大的问题。但我们在使用其它的三角范数算子 S_Z、T_Z 和 T_L 时，粒子2的惯性质量不能够约去，因此当粒子2的惯性质量等于零时，就会出现粒子2的加速度等于无限大的问题，显然本文不允许这种情况发生。

为了解决上述问题，我们变换一下式(4)和式(6)。对于求最小值问题时，把式(4)改为：

$$worst(t)=(\max_{j\in\{1,\cdots,N\}}fit_j(t))+\varepsilon \tag{18}$$

式中，ε 是一个很小的常数。把式(18)带入式(1)和式(2)，可知每个粒子的惯性质量的大小在区间[0,1]上。

对于求最大值问题时，把式(6)改为

$$worst(t)=(\min_{j\in\{1,\cdots,N\}}fit_j(t))-\varepsilon \tag{19}$$

式中，ε 是一个很小的常数。把式(19)带入式(1)和式(2)，可知每个粒子的惯性质量的大小在区间[0,1]上。这样就很好地解决了上述问题。

3 实验仿真与结果分析

本文的实验在Windows XP系统上，使用Matlab R2009a版本仿真。本文的实验都是寻找测试函数的最小值。为了叙述方便，把用三角范数算子 S_Z、T_Z、T_{HD}、T_P 和 T_L 所替换对应的引力搜索算法分别简称为SZ-GSA、TZ-GSA、THD-GSA、TP-GSA 和 TL-GSA，而 TP-GSA 即为本文原算法GSA。在3.1节，我们使用单峰函数进行测试，3.2节使用多峰函数进行测试。最后对一个多峰函数在不同密度的峰数情况下进行测试，以证实测试函数的峰值密度对各个算法搜索效果的影响。

3.1 单峰函数的测试与分析

在表1中，测试函数全是单峰函数，S 是 R^n 的子集，n 代表函数的维数，所有测试函数的最小值都为零。在本文中，函数的维数 $n=10$，对每个函数分别用5个三角范数算子替换相对应的引力搜索算法进行搜索，运行时每个算法分别迭代500次。实验结果如表2所列。表2中 Mean 和 Std 分别表示每个算法分别运行30次时最优值的平均值和标准差。

表1 测试函数

测试函数	S
$F_1(X)=\sum_{i=1}^{n}x_i^2$	$[-100,100]^n$
$F_2(X)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$	$[-100,100]^n$
$F_3(X)=\max\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	$[-100,100]^n$
$F_4(X)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-30,30]^n$
$F_5(X)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]^n$
$F_6(X)=\sum_{i=1}^{n}ix_i^4+\mathrm{random}[0,1)$	$[-1.28,1.28]^n$
$F_7(X)=\sum_{i=1}^{n}x_i^2+(\sum_{i=1}^{n}0.5ix_i)^2+(\sum_{i=1}^{n}0.5ix_i)^4$	$[-10,10]^n$

表2 仿真结果

		S_Z	T_Z	T_{HD}	T_P	T_L
F_1	Mean	347.4009	7.1918×10^{-10}	6.9065×10^{-10}	6.8068×10^{-10}	1.3937×10^{4}
	Std	1.0041×10^{3}	3.8175×10^{-10}	3.5421×10^{-10}	3.6102×10^{-10}	3.4267×10^{3}
F_2	Mean	8.4321×10^{3}	550.6594	561.4946	11.0039	1.6513×10^{4}
	Std	3.3845×10^{3}	378.5130	405.2270	9.7435	4.2552×10^{3}
F_3	Mean	10.2120	3.1384×10^{-7}	2.0149×10^{-7}	8.6645×10^{-9}	65.6240
	Std	18.8205	5.9906×10^{-8}	3.1414×10^{-8}	1.7886×10^{-9}	4.8426
F_4	Mean	1.3883×10^{7}	148.4185	374.1176	23.6853	2.8853×10^{7}
	Std	1.2306×10^{7}	268.4405	555.1399	53.5364	1.4507×10^{7}
F_5	Mean	7.7987×10^{3}	6.4290×10^{-10}	6.0955×10^{-10}	5.5607×10^{-10}	1.4052×10^{4}
	Std	3.8313×10^{3}	2.6863×10^{-10}	3.2376×10^{-10}	2.6070×10^{-10}	3.5841×10^{3}
F_6	Mean	0.0598	0.0046	0.0055	0.0052	5.0739
	Std	0.0339	0.0033	0.0045	0.0036	2.0905
F_7	Mean	345.4363	345.4363	312.0855	25.1028	354.6346
	Std	277.0855	277.0855	180.6964	11.4806	322.7101

从表2可以看出，函数 F_1、F_2、F_3、F_4、F_5 和 F_7 用 GSA 所搜索到的最优值要好于其它的算法。而且从图2可以看出，TP-GSA 对函数搜索的收敛性更远远好于其它的算法；对于 TZ-GSA 和 THD-GSA，除了函数 F_6 用 TZ-GSA 搜索的结果要好于 GSA 之外，其它函数的搜索结果都差不多。因为在式(9)中存在一个随机数 $rand$，所以在某一时刻 t，粒子 i 的步长 $a_i^d(t)$ 是一个小范围内的随机值，因此两个搜索能力相近的算法，不一定就能说明这两个算法搜索能力的好坏。并且从图2可以看出，TZ-GSA 和 THD-GSA 的收敛性也差不多，而 SZ-GSA 和 TL-GSA 对表1中的函数几乎不收敛。

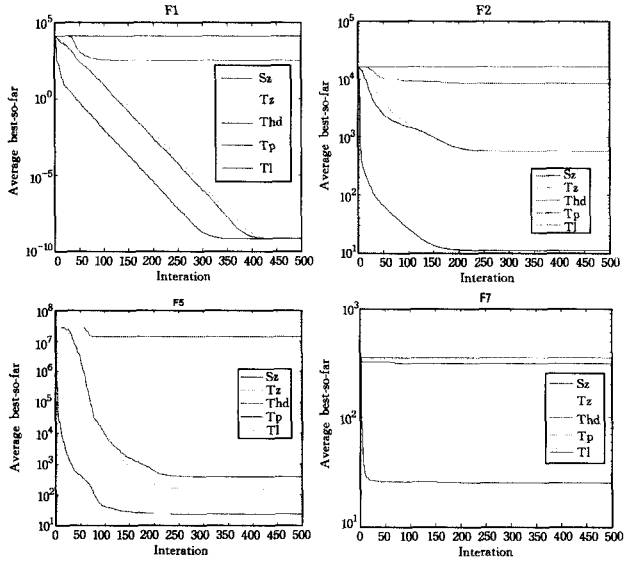

图 2　实验仿真图像

通过以上分析可以看出，对于单峰函数，不管从收敛性方面还是从最后结果好坏角度去考虑，都应该用 GSA 进行搜索；其次，仅仅从最后搜索到的结果精度来看，也可以用 TZ-GSA 和 THD-GSA 进行搜索；但从收敛性的角度看，最好还是用 GSA 搜索。SZ-GSA 和 TL-GSA 对表 1 中的函数几乎不收敛。从 2.1 节分析的三角范数性质知道，三角范数算子 S_Z 和 T_L 分别是本文中所用到的 5 个三角范数算子中最强的和最弱的，因此对于单峰函数，可以不用考虑用强于 S_Z 和弱于 T_L 的三角范数算子替换的引力搜索算法来搜索单峰函数的最优值。

3.2　多峰函数的测试与分析

表 3 中全是多峰函数，其中函数 F_8、F_{10} 和 F_{11} 是低维多峰函数，其它为高维多峰函数。S 是 R^n 的子集，$f*$ 表示相对应函数的最小值，n 代表函数的维数，在这里 $n=10$。实验结果如表 4 所列。表 4 中 Mean 表示运行 30 次最优值的平均值，Std 表示运行 30 次时最优值的标准差。

表 3　测试函数

测试函数	S	$f*$
$F_8(X)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	$[-10,10]^2$	0
$F_9(X)=(x_1-1)^2+\sum_{i=2}^{n} i(2x_i^2-x_{i-1})^2$	$[-10,10]^n$	0
$F_{10}(X)=(1+(x_1+x_2+1)^2(19-14x_1+13x_1^2-14x^2+6x_1x_2+3x_2^2))*(30+(2x_1-3x_2)^2(18-32x_1+12x_1^2-48x_2-36x_1x_2+27x_2^2))$	$[-2,2]^2$	3
$F_{11}(X)=-\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$	$[-100,100]^2$	−1
$F_{12}(X)=20+e-20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}}-e^{-\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)}$	$[-32,32]^n$	0
$F_{13}(X)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^n$	0
$F_{14}(X)=10n+\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i))$	$[-5.12,5.12]^n$	0
$F_{15}(X)=-\sum_{i=1}^{n}(x_i\sin\sqrt{\lvert x_i\rvert})$	$[-500,500]^n$	−418.98n

为了更好地分析实验结果，首先分析表 3 中每个函数的三维图像的特征。图 3 列出了部分函数的三维图像。对于函数 F_8、F_9 和 F_{10}，它们的最小值点周围的点的一阶导数的值相对比较小，周围很平坦。典型的三维图像如图 3 中的函数 F_8 的三维图像所示。对于函数 F_{11} 和 F_{12}，它们的最小值点周围的点的一阶导数的值相对比较大，而且它们的最小值和局部最小值都分布在一个相对比较小的范围内，峰值分布密度比较大，近似于锥形。典型的三维图像如图 3 中函数 F_{12} 的三维图像所示。对于函数 F_{13}、F_{14} 和 F_{15}，它们最小值点周围的点的一阶导数值相对比较大，而且它们的最小值和局部最小值都相对均匀地分散在比较大的范围内。典型的三维图像如图 3 中函数 F_{13} 和 F_{15} 的三维图像所示。

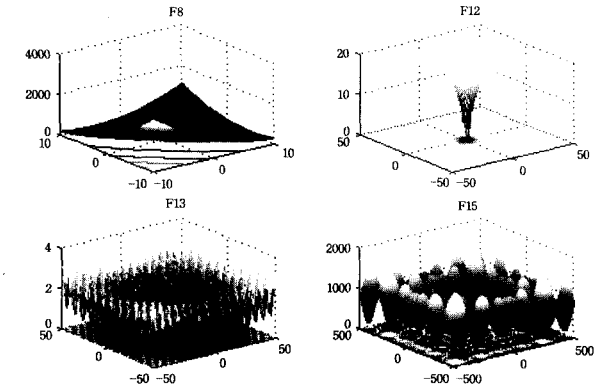

图 3　部分测试函数的三维图像

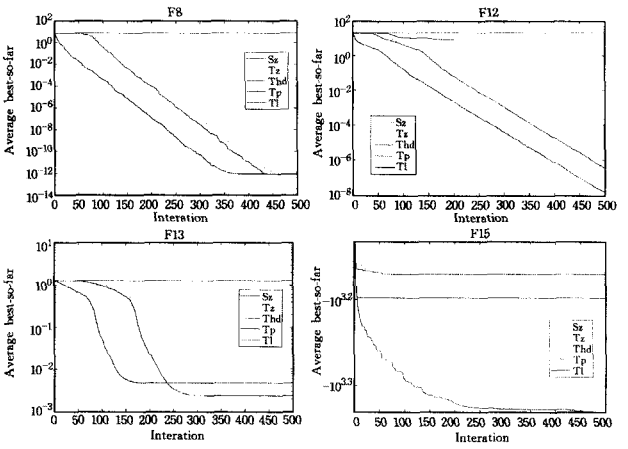

图 4　实验仿真图像

从表 4 的实验结果可以看出，对于函数 F_8、F_9、F_{10}、F_{11} 和 F_{12} 搜索到的结果和函数的收敛性还是用 GSA 算法比较好。而对于函数 F_{13}、F_{14} 和 F_{15}，TZ-GSA 和 THD-GSA 搜索的结果和收敛性相对比较好。特别值得注意的是 SZ-GSA 对函数 F_{15} 的搜索，它对该函数的搜索结果明显好于其它算法。从图 4 也可以看出，SZ-GSA 对函数 F_{15} 的收敛性也都远远好于其它的算法。由 2.1 节可知，三角范数算子 S_Z 是本文用到的 5 个算子中最强的，根据式(9)可知，它相对的加速度也比较大，因此它所替换的引力搜索算法很容易跳出局部最优值。但从函数 F_{13} 和 F_{14} 的搜索结果又可以看出，越强的三角范数算子替换的引力搜索算法搜索到的结果不一定越好。对此可以认为，当迭代一定次数而陷入局部最优值后，需要局部搜索。这时和单峰函数一样，步长越小，搜索到的结果越好。因此对于和函数 F_{15} 的二维图像分布相似的函数，搜索最优值的算法的选择应该从能否从局部最优值跳出和搜索精度两方面综合考虑来选一个合适步长的算法。从图 3 可以看出，函数 F_{13} 的局部最优值峰值的分布比 F_{15} 的局部最优值峰值分布的密度要大得多，因此对于函数 F_{15} 应该使用比较强的三角范数算子替换的引力搜索算法搜索，因为步长相对比较长

的算法更加容易跳出局部最优值。也就是说，函数局部最优值分布密度小的，应该用比较强的三角范数算子替换的引力搜索算法；函数局部最优值分布密度大的，应该用比较弱的三角范数算子替换的引力搜索算法。

表 4 仿真结果

		SZ	TZ	THD	TP	TL
F_8	Mean	7.2240	9.4310×10^{-13}	8.3343×10^{-13}	9.1037×10^{-13}	7.2240
	Std	5.8237	1.4625×10^{-12}	1.7449×10^{-12}	1.5467×10^{-12}	5.8237
F_9	Mean	1.0078×10^{4}	0.7309	1.4029	0.7074	8.6200×10^{4}
	Std	7.0881×10^{3}	0.1534	1.9988	0.0780	4.3790×10^{4}
F_{10}	Mean	40.2958	3.0000	3.0000	3.0000	40.2958
	Std	34.7973	0	0	0	34.7973
F_{11}	Mean	−0.1169	−0.2010	−0.2686	−0.5484	-1.1334×10^{-11}
	Std	0.2462	0.2731	0.3001	0.3948	5.9455×10^{-11}
F_{12}	Mean	7.7117	5.9618	3.5713	1.6148	19.7468
	Std	1.2057	1.0109×10^{-7}	5.8243×10^{-8}	2.7294×10^{-9}	0.8581
F_{13}	Mean	1.2770	0.0034	0.0024	0.0048	1.2770
	Std	0.0648	0.0047	0.0048	0.0070	0.0648
F_{14}	Mean	112.3180	3.3165	3.5819	3.5155	112.3180
	Std	12.4121	1.4151	1.6022	1.9331	12.4121
F_{15}	Mean	-2.1419×10^{3}	-1.5836×10^{3}	-1.4811×10^{3}	-1.5778×10^{3}	-1.3576×10^{3}
	Std	246.1643	213.2992	240.2957	211.4813	250.5114

4 仿真结果

为了论证上述结论，这里用函数 F_{13} 做实验。设该函数在这里的搜索空间为 $S=[-5,5]$，函数的维数 $n=10$。对表 3 中函数 F_{13} 的每个变量 x_i 乘以一个常数 k，得到下面函数 F_{16}。当 k 越大，函数 F_{13} 在一定范围内的局部最小值越多，也就是局部峰值密度越大。表 5 为不同 k 值的情况下不同三角范数算子替换的引力搜索算法搜索到的对应结果。

$$F_{16}(X)=\sum_{i=1}^{n}\frac{(k\cdot x_i)^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{k\cdot x_i}{\sqrt{i}}\right)+1$$

表 5 仿真结果

K		SZ	TZ	THD	TP	TL
0.1	Mean	0.0393	1.5267	1.6073	1.4044	0.0393
	Std	0.0092	1.2902	1.7273	7.5874	0.0092
1.5	Mean	0.2204	0.0035	0.0022	0.0200	0.9715
	Std	0.2327	0.0050	0.0038	0.0561	0.0467
2.0	Mean	0.9415	0.0029	0.0031	0.0046	0.6410
	Std	0.0986	0.0059	0.0055	0.0085	0.9415
3.0	Mean	1.0627	0.0019	0.0025	0.0039	1.0627
	Std	0.0447	0.0052	0.0049	0.0083	0.0447
4.0	Mean	1.0989	0.0014	0.0032	0.0020	1.0989
	Std	0.0952	0.0038	0.0069	0.0048	0.0952
5.0	Mean	1.2040	0.0010	0.0023	0.0016	1.2040
	Std	0.0756	0.0039	0.0050	0.0039	0.0756
6.0	Mean	1.3058	0.0031	0.0033	0.0048	1.3058
	Std	0.0691	0.0052	0.0069	0.0071	0.0691
7.0	Mean	1.4599	0.0014	0.0039	0.0044	1.4599
	Std	0.1075	0.0038	0.0052	0.0068	0.1075
8.0	Mean	1.5685	0.0026	0.0033	0.0025	1.5285
	Std	0.1194	0.0053	0.0053	0.0048	0.1194
9.0	Mean	1.6994	0.0017	0.0024	0.0030	1.6994
	Std	0.1576	0.0048	0.0046	0.0056	0.1576
10.0	Mean	1.8887	0.0044	0.0024	0.0021	1.8887
	Std	0.1566	0.0061	0.0042	0.0043	0.1566

当 $k=0.1$ 时，函数 F_{16} 只有一个最小值，即为单峰函数，表 5 中相对应的结果符合 5 个算法的搜索结果。当 k 等于 2.0、3.0、4.0、5.0、6.0、7.0 和 9.0 时，F_{16} 为多峰函数，而且各个峰值之间都存在一定的距离，这时需要步长比较大的算法搜索最优值；当 k 等于 6.0、7.0 和 9.0 时，F_{16} 为多峰函数，而且各个峰值之间都存在一定的距离，这时需要步长比较大的算法搜索最优值。从表 5 中的结果可以看出，这时 TZ-GSA 搜索到的结果最好，这几乎符合上节的结论。当 k 等于 1.5 和 9.0 时，THD-GSA 搜索结果最好，这是因为当 k 等于 1.5 时，函数 F_{16} 为多峰函数，而且它们的各峰值之间的距离足够大，以致于本文所提到的 5 个算法的步长都没有足够大从而跳不出局部最优值。因此在陷入最优值后，相当于搜索单峰函数，这时算法的步长越短，搜索到的效果越好；而当 k 等于 9.0 时，函数 F_{16} 也为多峰函数，这时步长越小，反而搜索到的效果可能越好，这是因为这时峰值之间的距离比较小，以致于本文提到的 5 个算法步长都足够长而很容易跳出局部最优值。这时算法的步长越小，反而搜索到的效果也有可能越好，这与多峰函数且峰值密度很大的情况相似。当 k 等于 10.0 时，TP-GSA 搜索的结果最好，这种情况可以和 k 等于 9.0 时的情况一样理解。

结束语 本文在引力搜索算法基础上根据三角范数的性质而变换出了其他几种引力搜索算法。针对测试函数二维图像特征，应该使用相应的三角范数算子替换出的引力搜索算法搜索它的最优值。由第 3 节可知，对于多峰函数，且它们的峰值分布均匀，密度不是太大也不是太小的情况下，应该尽量使用比较强的三角范数算子，这样搜索最优值时，它的搜索能力和收敛速度都比较好。而对于其它特征的测试函数，应该使用原来的算法，即 TP-GSA。我们还可以得出结论：对于强于 S_Z 和弱于 T_L 的三角范数算子，可以不用考虑替换。

参 考 文 献

[1] Rashedi E, Nezamabadi-Pour H, Saryazdi S. GSA: A Gravitational Search Algorit-hm[J]. Information Sciences, 2009, 179(13): 2232- 2248

[2] Kennedy J, Eberhart R C. Particle swarm optimi-zation[C]// IEEE International Conference on NeuralNetwork. Perth, Australia, 1995, 4: 1942-1948

[3] Formato R A. Studies in Computational Intelli-gence[M]. Heidelberg, Berlin; ElisabethRakus-Anders-son, 2008: 221-238

[4] Klement E P, Mesiar R, Pap E. Triangular Norms[M]. Kluwer

Academic Publishers,2000

[5] Alsina C,Frank M J,Schweizer B. Associative Functions:Triangular Norms and Copulas[M]. WorldScientific Publishing Company,2006

[6] 罗敏霞,何华灿. 基于幂零泛与运算模型的命题模糊逻辑[J]. 计算机科学,2004,31(8):97-99

[7] 陈丹,何华灿,王晖. 基于连续可控 T 范数的模糊控制方法研究[J]. 控制理论与应用,2001,18(5):717-721

[8] Schutz B. Gravity from the Ground Up[M]. Cambridge University Press,2003

[9] Holliday D,Resnick R,Walker J. Fundamentalsof physics[M]. John Wiley and Sons,1993

(上接第 207 页)

价和优劣比较,根据 4.2 节的灰熵绝对关联分析评价模型流程图进行综合评价,具体步骤如下所示。

表 1 嵌入式计算机比较序列与参考序列的无量纲数据

指标	x_0	x_1	x_2	x_3
U_{11}	1.0000	1.0000	1.0000	1.0000
U_{12}	1.4189	2.5803	0.9784	3.4400
U_{13}	1.4354	2.2802	1.1149	4.4800
U_{21}	1.4849	2.6103	1.1263	3.6000
U_{22}	1.5344	2.6103	1.1149	5.3600
U_{23}	1.3859	2.3702	1.0125	4.3200
U_{24}	1.3694	2.2802	1.0011	4.4800
U_{31}	1.6169	2.9403	1.1149	6.2400
U_{32}	1.4354	2.5803	1.0125	5.1200
U_{33}	1.4354	2.6103	1.0011	4.4800
U_{41}	1.1219	1.3201	0.9898	0.9600
U_{42}	1.2704	1.9502	1.0125	2.5600
U_{43}	1.2209	2.0102	0.8874	3.6000
U_{44}	1.2869	2.3402	0.8874	4.4800
U_{51}	1.4519	2.6703	0.9898	6.0800
U_{52}	1.4519	2.3702	1.1149	5.3600
U_{53}	0.5494	0.9991	0.3788	2.6640
U_{54}	1.0889	1.6802	0.8646	3.6000
U_{61}	1.5014	2.6403	1.0694	6.2400
U_{62}	1.3694	2.2802	1.0125	4.4800

表 2 比较序列的灰色绝对关联系数

k	r_{01}	r_{02}	r_{03}
1	0.4627	0.6942	0.3310
2	0.7595	0.8929	0.4942
3	0.7809	0.9633	0.5183
4	0.9528	0.9426	0.3689
5	0.9161	0.9559	0.5287
6	0.9315	0.9949	0.8500
7	0.7079	0.8821	0.3980
8	0.8485	0.9267	0.5159
9	0.9709	0.9887	0.6098
10	0.5059	0.7679	0.2377
11	0.6749	0.8883	0.4079
12	0.9013	0.9297	0.4786
13	0.7911	0.9381	0.5513
14	0.8583	0.9411	0.4107
15	0.7692	0.8888	0.5814
16	0.6809	0.8573	0.3580
17	0.8760	0.9490	0.7161
18	0.6462	0.8280	0.3098
19	0.8143	0.9301	0.3805

步骤 1 建立层次结构模型

根据嵌入式计算机性能参数指标和能够适应未来环境发展的需求,构建了 3 层的评价指标体系,如图 1 所示。

步骤 2 确定参考序列和比较序列

参考序列 x_0 是通过统计方法从众多性能较好的嵌入式计算机中得到的相关指标数据而构造的,比较序列 x_1,x_2,x_3 分别是由 A、B、C 3 种嵌入式计算机根据相关指标得到的数据,并采用初值像对序列 $x_i,i=0,1,2,3$ 进行无量纲处理,处理结果如表 1 所列。

步骤 3 计算比较序列的灰色绝对关联系数 $r_{0i}(k),(i=1,2,3)$。

根据式(6),得到比较序列的灰色绝对关联系数如表 2 所列。

步骤 4 计算灰熵绝对关联度

根据式(2)和式(3),计算比较序列的灰绝对关联熵 $H(p_i)$分别为:

$H(p_1)=2.9278,H(p_2)=2.9410,H(p_3)=2.9004$

则根据式(4),得到灰熵绝对关联度 $E_r(x_i)$分别为:

$E_r(x_1)=0.9943,E_r(x_2)=0.9988,E_r(x_3)=0.9850$

步骤 5 进行综合评价

将灰熵绝对关联度 $E_r(x_i),i=1,2,3$ 按由大到小的顺序进行排列,显然有 $E_r(x_2)>E_r(x_1)>E_r(x_3)$。

这表明 A、B、C 3 种嵌入式计算机中,B 型嵌入式计算机的综合性能最优,其次是 A 型嵌入式计算机,最后为 C 型嵌入式计算机。评价结果与实际使用中的性能表现相符。

结束语 本文将灰熵理论与灰色绝对关联分析理论相结合,提出了灰熵绝对关联分析评价模型,该模型有效地解决了传统灰熵关联分析的不唯一性,提高了嵌入式计算机性能评估的准确性和有效性,为今后嵌入式计算机综合性能的评价提供了有价值的参考。

参 考 文 献

[1] Kahrman C,Cevik S,Ates N Y,et al. Fuzzy multi-criteria evaluation of industrial robotic systems[C]// Omputer & Industrial Engineering. 2007,52:414-433

[2] 邓聚龙. 灰色系统基本方法[M]. 武汉:华中理工大学出版社,2006

[3] 吕锋,刘翔,等. 七种灰色系统关联度的比较研究[J]. 武汉工业大学学报,2000,22(2):41-44

[4] 张岐山,郭喜江,邓聚龙. 灰关联熵分析法[J]. 系统工程理论与实践,1996(8):7-11

[5] 郭齐胜,董志明,单家元. 系统建模[M]. 北京:国防工业出版社,2006

[6] 邓聚龙. 灰色系统理论教程[M]. 武汉:华中理工大学出版社,1990:3-84

[7] 郑子华,陈家祯,陈利永. 基于灰色绝度关联度的边缘检测算法[J]. 福建师范大学学报,2004,20(4):20-23

[8] 梅振国. 灰色绝对关联度及其计算方法[J]. 系统工程,1993,10(5):43-44