基于子群极值和Sharing重分布的粒子群优化算法

龚 燕 张 浩
(绵阳电业局 绵阳 621000)

摘 要 为提高粒子群优化算法在优化问题中的效率,提出了粒子群优化算法(ESPSO)。其基本思想是分多子群搜索和Sharing函数重分布。主要工作包括:(1)将粒子群分成多个子群,各自搜索解空间;(2)信息共享机制中引入子群极值,使粒子更新能参考其他粒子的信息;(3)使用Sharing对陷入局部最优的粒子进行重分布。在4个基准函数上的优化实验表明,新方法比经典的IPPSO粒子群算法在达到目标精度的成功率上提高了64%~93%。

关键词 粒子群算法,子群,极值,Sharing

Particle Swarm Optimization Algorithm Based on Extreme Value of Sub-swarm and Sharing Redistribution

GONG Yan ZHANG Hao
(Mianyang Electric Power Bureau, Mianyang 621000, China)

Abstract To improve the efficiency of Particle Swarm Optimization, this paper proposed a novel Particle Swarm Optimization algorithm(ESPSO). The basic idea is Sub-Swarm mechanism and Sharing Redistribution. The main contributions include, (1) Divides whole Swarm into n sub-Swarm; Each sub-Swarm search solution Independently; (2) Introduces extreme value of Sub-Swarm strategies to enable particle interaction; (3) Introduces Sharing Function to redistribute some Particle. The experiments on four benchmark functions show that the new algorithm increases success rate by 64%~93% compared with IPPSO.

Keywords PSO, Sub-swarm, Extreme value, Sharing

粒子群优化算法(PSO)由R. Eberhart和J. Kennedy等提出,能够有效解决复杂优化任务[1,2],已被广泛应用于函数优化、神经网络训练、模糊系统控制以及其它应用领域[3]。但PSO在接近最优解时容易出现早收敛[4],导致搜索精度不高。

为解决上述问题,本文提出了子群和重分布的策略,用以提高粒子的多样性,同时扩大搜索范围,以解决算法早收敛的情况。

1 传统概念和算法

PSO算法根据对环境的适应度将群体中的个体移动到好的区域,将个体看作D维搜索空间中的无体积的粒子,在搜索空间中以一定的速度飞行[5]。

每个粒子在解空间中移动,各个粒子记录下自己曾搜索到的最优点并记录下所有粒子搜索到的全局最优点,粒子根据自身最优点及全局最优点来更新自己的速度和位置。粒子群优化算法就是通过不断地对极值点的更新而实现的智能算法。

在D维空间,第i个粒子被描述为$X_i=(x_{i1},x_{i2},\cdots,x_{iD})$;每个粒子迄今所经历的个体最优位置,描述为$P_i=(p_{i1},p_{i2},\cdots,p_{iD})$;整个粒子群搜索到的最优位置为$P_g=(p_{g1},p_{g2},\cdots,p_{gD})$。加上惯性权重因子[7],粒子状态更新策略为:

$$v_{id}^{(t+1)}=wv_{id}^{(t)}+c_1r_1(p_{id}-x_{id}^{(t)})+c_2r_2(p_{gd}-x_{id}^{(t)}) \quad (1)$$

$$x_{id}^{(t+1)}=x_{id}^{(t)}+v_{id}^{(t+1)} \quad (2)$$

式中,w为惯性权重;c_1,c_2为学习因子,一般取2;r_1,r_2为分布在(0,1)之间的随机数;$v_{id}^{(t)}$表示第i个粒子第t次迭代时在第d维上的速度;p_{id}表示第i个粒子的历史最优位置在第d维上的坐标;p_{gd}表示整个粒子群中最优粒子在第d维上的坐标。

2 子群极值和Sharing重分布

2.1 基于子群极值的算法

IPPSO[6]算法中,主进程选出主群中全局最佳粒子并广播给子进程,迫使子群体进行全局最优演化。本文提出了基于子群极值的粒子群算法(extreme value of Sub-swarm, EPSO),其采用的模型与IPPSO一样,都是基于岛屿模型的。

该算法周期性地将子群内的最佳粒子发给主进程,主进程则从中选出全局最佳粒子并广播给各子群,但各子群并不用全局最佳粒子来替换子群内最佳粒子,只是向全局最佳粒子方向移动,同时保留自己的群内最佳粒子。亦即各粒子的更新策略中的速度更新增加了一部分,得到了新的更新公式:

$$v_{id}=wv_{id}+u_1c_1r_1(p_{id}-x_{id})+u_2c_2r_2(p_{gd}-x_{id})+c_3r_3(P_{gmaxd}-x_{id}) \quad (3)$$

到稿日期:2010-01-12 返修日期:2010-04-03 本文受川电科技([2010]45号)资助。

龚 燕(1981-),女,硕士生,主要研究方向为数据库理论与信息系统,E-mail:gongyan33@163.com;张 浩(1975-),男,主要研究方向为计算机网络与信息化管理。

式中，c_1，c_2，c_3 是学习因子，r_1，r_2，r_3 是(0～1)之间的随机数，p_{gd} 是各子群的群内最佳粒子，p_{gmaxd} 是全局最佳粒子。速度更新由 4 部分组成，第三部分是区域学习，第四部分是向全局最佳粒子学习。粒子保留自己的群内最佳粒子，使得搜索多样化，子群不丢失寻优方向，同时向全局最佳粒子方向移动。

新算法引入了两个参数：影响因子 u_1，u_2，用来刻画个体极值和子群内极值对粒子速度更新的影响。

在速度更新方式中，学习因子由 2 个变成了 3 个。一个新的关键问题就产生了，即必须确定 u_1，u_2，c_1，c_2，c_3 的值，以保证算法的收敛性。当 c_1，c_2，c_3 全取常数 2 时，没有破坏公式的平衡性，并有利于确保新算法的收敛性；而 u_1，u_2 的具体取值由实验决定(具体见后面实验)。

算法 1 基于子群极值(EPSO)的优化算法

```
1. while(maximum iterations is not attained)
2. {
3.    Calculate fitness_value();//计算适应度
4.    Particle Fly();//根据式(2)，式(3)更新粒子的速度和位置
5.    if(The time of transference has arrived) // 迁徙周期到的时候
6.    {
7.      Attain best_fitness();//子进程将最优适应度传送给主进程
8.      Select best_solution();//选择最优的解
9.      Distribute the_best_solution();//将最优解广播到子进程
10.   }
11. }
```

2.2 基于 Sharing 函数重分布的策略

为提高粒子群算法的局部搜索能力，提高搜索精度并避免陷入局部最优，我们提出了基于 Sharing 函数重分布的优化算法 SPSO。基本思想是：当粒子种群陷入局部最优或到暂时停滞状态时，保留部分群体最优粒子进行局部搜索，并将其它粒子在 Sharing 函数的作用下重新分布到搜索空间中进行全局搜索，同时阻止重新初始化的粒子立刻被同一局部最优点吸引。新方法的要点在于不损失全局最优信息的前提下避免再度陷入同一局部最优。算法 2 描述了这一过程。

算法 2 基于 Sharing 函数的粒子群优化算法(SPSO)

```
1. if(particles plunge local trap or iterations stagnate)
2. Particle_Redistribute();//调用基于 Sharing 函数的重分布函数，保
   留群体最优的同时重分布其余粒子，避免陷入同一局部最优
3. else ParticleFly();//粒子继续飞翔
```

Sharing 函数由 Goldberg 于 1987 年提出[7]，它通过共项函数调整群体中各个个体的适应度。在部分粒子的重新分布过程中，算法能够根据调整后的新适应度适当地将粒子重新分布，避免进入同一局部最优。Sharing 函数的定义如下。

定义 1(Sharing 函数)

$$\text{sharing}(d(i,j))=\begin{cases}1-\dfrac{d(i,j)}{d_{radius}}, & d(i,j)<d_{radius}\\ 0, & d(i,j)\geqslant d_{radius}\end{cases} \tag{4}$$

式中，d_{radius} 为粒子邻域的半径，其大小由使用者给定；d_{ij} 为粒子 i 与局部最优点 j 之间的欧几里德距离。

重新分布的粒子的坐标通过以下定义的更新坐标函数计算。

定义 2(基于 Sharing 函数的新坐标函数)

$$X_{inew}^d=\frac{X_i^d}{1-\text{sharing}(x)} \tag{5}$$

式中，$X_i^d=(X_i^1,X_i^2,\cdots,X_i^d)$ 为粒子 i 重分布初始化时的坐标。

定理 1 基于 Sharing 函数的重新分布能避免粒子群陷入同一局部最优。

证明：设某次搜索过程中找到的局部最优点坐标为 $X_l=(X_l^1,X_l^2,\cdots,X_l^d)$，最优点的邻域半径为 R_l，重分布后的粒子坐标为 $X_i=(X_i^1,X_i^2,\cdots,X_i^d)$。若其中某一粒子 p 的坐标为 $X_p=(X_p^1,X_p^2,\cdots,X_p^d)$，并且粒子 l，p 之间的欧几里德距离 $d(p,l)<R_l$ 且 $d(p,l)\neq 0$，将式(4)代入式(5)，可得

$$X_{pnew}^d=(R_l/\ d(p,l))X_p^d$$

令 $R_l/\ d(p,l)=k$，把 k 定义为脱离因子，则 $Xd_{pnew}=kX_p^d$，即粒子 p 重分布时的坐标在脱离因子的作用下会远离上次的局部最优点。如果经过一次脱离因子作用后粒子仍在局部最优点的邻域内，则重复迭代以上过程，直到粒子脱离局部最优点。若 $d(p,l)=0$，即粒子 p 重分布时的坐标正好是局部最优点，则使用一个使用者自己决定大小的常数 C 作用于粒子 p，脱离局部最优点。所以，定理得证。

设重新初始化后的粒子 1、2 和 3 的分布如图 1 所示。$d(1,i)=0.2d_{radius}$，根据式(4)、式(5)，则 $X_{1new}^d=5\times X_1^d$；同理，如果 $d(2,i)=0.8d_{radius}$，则对于粒子 2，$X_{2new}^d=1.25\times X_2^d$；当 $d(3,i)>d_{radius}$ 时，对于粒子 3，$X_{3new}^d=X_3^d$。

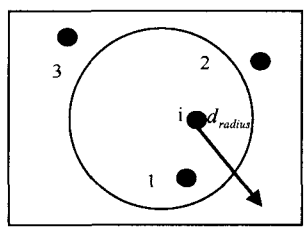

图 1

重新初始化后的粒子越靠近上一次的局部最优，则根据 Sharing 函数计算出的脱离因子就越大，就好像局部最优点对重新初始化后的粒子有排斥力。这样就能避免粒子群再次陷入同一局部最优。

2.3 基于子群极值算法和 Sharing 函数重分布

基于子群极值算法的 EPSO 能在一定程度上隔离各个子群的影响，能提高多样性。但是同样可能陷入局部最优。而基于 Sharing 函数重分布能在保留部分最优粒子信息的同时，对其他粒子重分布，可以有效地改善早熟情况，因此将两者结合。基于子群极值和 Sharing 函数重分布的粒子群算法如算法 3 所示。

算法 3 基于子群极值和 Sharing 函数重分布的并行粒子群算法(ESPSO)

```
1. while(maximum iterations is not attained)
2. {
3.    Calculate fitness_value();//计算适应度
4.    if(particles plunge local trap or iterations stagnate)
5.    Particle_Redistribute();//调用基于 Sharing 函数的重分布函
      数，保留群体最优的同时重分布其余粒子，以避免陷入同一局部
      最优
6.    else ParticleFly();//粒子继续飞翔
7.    if(The time of transference has arrived) // 迁徙周期到的时候
8.    {
9.      Attain best_fitness();//子群将最优适应度传送给主进程
10.     Select best_solution();//选择最优的解
```

```
11.     Distribute the_best_solution();//将最优解广播到子群
12.     }
13. }
```

3 实验

采用4个无约束优化标准测试函数，即Sphere，Rosenbrock，Griewank，Rastrigrin，维数都为10，分别采用PSO，IPPSO，ESPSO进行实验，并进行比较。这4个函数如表1所列。

表1 标准测试函数

名称	表达式	最大速度	搜索空间	维数(d)	最小值位置
Sphere	$\sum_{i=1}^{d} x_i^2$	$(0.9)^d$	$(-100,100)^d$	10	$(0)^d$
Rosenbrock	$\sum_{i=1}^{d-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$(1.1)^d$	$(-100,100)^d$	10	$(1)^d$
Griewank	$\frac{1}{4000}\sum_{i=1}^{d} x_i^2-\prod_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}})+1$	$(6.6)^d$	$(-600,600)^d$	10	$(0)^d$
Rastrigrin	$\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i^2)+10]$	$(0.06)^d$	$(-5.12,5.12)^d$	10	$(0)^d$

为了提高可比性，3种算法取同样的参数，惯性权重 w 的取值都为1，学习因子 c_1，c_2 都为2，粒子总数都为80，演化代数为1000。IPPSO，ESPSO将粒子分成8个子群，每个子群10个粒子，各子群都是每隔20代向主进程传送群内最佳粒子。

3.1 ESPSO中的参数 u_1，u_2

首先需要通过实验来确定ESPSO中的影响因子 u_1，u_2 要取的固定常数值。

比较 u_1，u_2 不同的取值，ESPSO运行第一个函数得到的解的精度如表2所列。适应度误差限为 10^{-2}，即对该函数运行一段时间后，若该时刻的全局最好适应值和该函数的最优值误差绝对值小于 10^{-2}，就认为该算法在此刻已经成功收敛。记录此刻的迭代次数，比较各组参数成功收敛的速度，如表3所列。每组数据进行50次实验。

表2 使用Sphere函数测试不同影响因子在第1000代得到的结果

u_1	u_2	能找到的最好结果	能找到的最差结果
0	1	0.425e-1	3.5
0.1	0.9	0.125e-1	0.654e-1
0.2	0.8	0.865e-2	0.545e-1
0.3	0.7	0.442e-2	0.981e-2
0.4	0.6	0.153e-2	0.654e-2
0.5	0.5	0.854e-3	0.58e-2
0.6	0.4	0.782e-3	0.84e-2
0.7	0.3	0.452e-2	0.873e-2
0.8	0.2	0.786e-2	0.673e-1
0.9	0.1	0.583e-1	4.90
1	0	0.675e-1	4.12

表2表示，随着影响因子 u_1 的增大、u_2 的减少，第1000代所得的解的精度的变化趋势是先增大、后减少，且中间位置左右解的精度最高。特殊情况下，当 $u_1=0$，$u_2=1$ 时，即粒子只按照子群内极值和全局极值搜索，所以精度低。当 $u_1=1$，$u_2=0$ 时，就是标准的PSO算法，解的精度也比较低，明显不如影响因子在中间附近取值的时候。

表3表示随着影响因子 u_1 的增大、u_2 的减小，Sphere函数达到误差限的平均迭代次数是先减小后增大的。也就是说该函数的收敛速度是先快后慢的，在中间位置该函数的收敛速度是最快的。同样在特殊情况下，当 $u_1=0$，$u_2=1$ 时，即粒子只是按照子群内极值和全局极值这两个极值搜索，对于Sphere单峰函数，所得到的收敛速度最慢。当 $u_1=1$，$u_2=0$，即标准的PSO算法，收敛速度也比较慢，远远比不上影响因子取中间的时候。

表3 使用Sphere函数测试不同影响因子达到预定精度的速度

u_1	u_2	最快迭代次数	最慢迭代次数	平均迭代次数
0	1	874	998	936
0.1	0.9	768	896	805
0.2	0.8	687	789	756
0.3	0.7	678	724	695
0.4	0.6	606	682	654
0.5	0.5	624	674	650
0.6	0.4	627	697	652
0.7	0.3	645	708	678
0.8	0.2	678	724	708
0.9	0.1	698	778	736
1	0	699	789	745

由表2和表3综合考虑所得的最优解的精度和收敛速度，发现当影响因子取中间值时，算法性能最好，即 $u_1=0.5$，$u_2=0.5$，明显优于标准PSO算法。

下面就用这种固定参数来检测其他3个测试函数是否有比较好的性能。

3.2 3种算法的收敛精度

对3种算法重复运行50次，结果如表4所列。总的来说，IPPSO比PSO的精度有所提高。而ESPSO所达到目标解的成功率比IPPSO有大幅度提高，提高了64%到93%。同时，ESPSO所得解的精度比PSO提高了78%到93%。

表4 PSO，IPPSO，ESPSO 3种算法实验结果比较

目标函数	要求达到的精度	平均迭代次数			成功率		
		PSO	IPPSO	ESPSO	PSO	IPPSO	ESPSO
Sphere	1.0e-1	987	896	1030	7.0%	13.0%	100.0%
Rosenbrock	1.0	1456	1394	1191	3.0%	6.0%	78.0%
Griewank	1.0e-1	1235	1056	994	21.0%	35.0%	99.0%
Rastrigrin	1.0	954	1098	768	9.0%	7.0%	100.0%

结束语 不同的信息共享方式，就会导致不同的算法。ESPSO通过让子群共享信息，扩展了子群的多样性，有效抑制了早熟，收敛效果明显比IPPSO和PSO好，且目标精度的成功率也有很大的提高。

参考文献

[1] Parsopoulos K E, Vrahatis M N. On the computation of all global minimizes through particle swarm optimization [J]. IEEE Transaction Evolutionary Computation, 2004, 8(3): 211-224

[2] 赵勇，岳继光. 一种新的求解复杂函数优化问题的并行粒子群算法[J]. 计算机工程与应用，2005，16

[3] Kazutomi S, Tanaka H. Interval Evaluation in the Analytic Hierarchy Process by Possibility Analysis [J]. Computational Intelligence, 2001, 17(3): 567-579

[4] Clere M. Stagnation Analysis in Particle Swarm Optimization or What Happens When Nothing HaPPens[J]. 2006(1)

[5] 谢晓锋，张文俊，杨之廉. 微粒群算法综述[J]. 控制与决策，2003，18(2)：129-134

[6] 黄芳，樊晓平. 基于岛屿群体模型的并行粒子群优化算[J]. 控制与决策，2006，21(2)：175-179

[7] Goldberg D E, Richardson J. Genetic Algorithm with Sharing for Multimodal Function Optimization[C]// Proceedings of the Second International Conference on Genetic Algorithms. 1987: 42-49