·博士论坛·

粒子群优化算法

李爱国[1,2]　覃　征[1]　鲍复民[1]　贺升平[1]
[1](西安交通大学计算机系,西安 710049)
[2](西安科技学院计算机系,西安 710054)
E-mail:liag@xust.sn.cn

摘　要　粒子群优化(PSO)算法是一类随机全局优化技术,PSO 算法通过粒子间的相互作用发现复杂搜索空间中的最优区域。PSO 的优势在于简单容易实现而又功能强大。PSO 已成为国际演化计算界研究的热点。该文介绍了基本的 PSO 算法、若干类改进的 PSO 算法及其应用,并讨论将来可能的研究内容。

关键词　粒子群　优化　演化计算　群智能

文章编号 1002-8331-(2002)21-0001-03　**文献标识码** A　**中图分类号** TP301.6

Particle Swarm Optimization Algorithms

Li Aiguo[1,2]　Qin Zheng[1]　Bao Fumin[1]　He Shengping[1]
[1](Department of Computer Science,Xi'an Jiaotong University,Xi'an 710049)
[2](Department of Computer Science,Xi'an University of Science and Technology,Xi'an 710054)

Abstract: Particle swarm optimizers are a stochastic global optimization technique.The particle swarm algorithms find optimal regions of complex search spaces through the interaction of individuals in a population of particles.Particle swarm optimizers have become the hotspot of evolutionary computation because of its excellent performance and simple for implement.In this paper,classical particle swarm optimization algorithm and its several variants and some applications of the algorithms are introduced,future research issues are also discussed.

Keywords: Particle Swarm,Optimization,evolutionary computation,Swarm Intelligence

1　引言

粒子群优化(Particle Swarm Optimizer,PSO)算法是一种基于群智能(Swarm Intelligence)方法的演化计算(evolutionary computation)技术。PSO 同遗传算法类似,是一种基于群体(population)的优化工具。系统初始化为一组随机解,通过迭代搜寻最优值。但是并没有遗传算法用的交叉(crossover)以及变异(mutation)操作,而是粒子(潜在的解)在解空间追随最优的粒子进行搜索。与遗传算法比较,PSO 的优势在于简单容易实现同时又有深刻的智能背景,既适合科学研究,又特别适合工程应用。因此,PSO 一提出,立刻引起了演化计算等领域的学者们的广泛关注,并在短短的几年时间里出现大量的研究成果,形成了一个研究热点。目前国内对 PSO 的研究还很少,希望该文能起到抛砖引玉的作用。

PSO 最早是由 kennedy 和 Eberhart 于 1995 年提出的[1,2]。受到人工生命(Artificial Life)的研究结果启发,PSO 的基本概念源于对鸟群捕食行为的研究。设想这样一个场景:一群鸟在随机搜寻食物。在这个区域里只有一块食物。所有的鸟都不知道食物在那里。但是他们知道当前的位置离食物还有多远。那么找到食物的最优策略是什么呢。最简单有效的就是搜寻目前离食物最近的鸟的周围区域。PSO 从这种模型中得到启示并用于解决优化问题。PSO 中,每个优化问题的潜在解都是搜索空间中的一只鸟,称之为"粒子"。所有的粒子都有一个由被优化的函数决定的适应值(fitness value),每个粒子还有一个速度决定他们飞翔的方向和距离。然后粒子们就追随当前的最优粒子在解空间中搜索。PSO 初始化为一群随机粒子(随机解)。然后通过迭代找到最优解。在每一次迭代中,粒子通过跟踪两个"极值"来更新自己。第一个就是粒子本身所找到的最优解。这个解称为个体极值。另一个极值是整个种群目前找到的最优解。这个极值是全局极值。另外也可以不用整个种群而只是用其中一部分作为粒子的邻居,那么在所有邻居中的极值就是局部极值。

由于认识到 PSO 在函数优化等领域所蕴含的广阔的应用前景,在 Eberhart 和 kennedy 之后很多学者都进行了这方面的研究。目前,已提出了多种 PSO 改进算法,并且 PSO 已广泛应用于函数优化,神经网络训练,模式分类、模糊系统控制以及其他的应用领域。

下面将首先介绍基本 PSO 算法以及几种典型的改进算法;然后简要分析了目前 PSO 算法的应用情况。最后讨论目前

基金项目:陕西省科学技术发展计划"十五"攻关资助项目(编号:2000K08-G12)
作者简介:李爱国,博士生,目前研究兴趣:信息融合,数据挖掘。覃征,教授,博士后,博士生导师,目前研究兴趣:复杂环境下自适应信息处理、计算机系统集成与电子商务、计算机网络应用、分布式并行信息处理。

PSO 研究的局限和将来可能的研究内容。

2 PSO 算法及其改进算法

自 PSO 算法提出以来,引起了人们的极大兴趣,因此提出了各种各样的改进算法,有力地推动了 PSO 算法的研究和应用。这一节介绍基本的 PSO 算法,以及几种有代表性的 PSO 改进算法。

2.1 基本 PSO 算法

假设在一个 D 维的目标搜索空间中,有 m 个粒子组成一个群落,其中第 i 个粒子表示为一个 D 维的向量 $\vec{x}_i=(x_{i1},x_{i2},\cdots,x_{iD})$,$i=1,2,\cdots,m$,即第 i 个粒子在 D 维的搜索空间中的位置是 $\vec{x}_i$。换言之,每个粒子的位置就是一个潜在的解。将 $\vec{x}_i$ 带入一个目标函数就可以计算出其适应值,根据适应值的大小衡量 $\vec{x}_i$ 的优劣。第 i 个粒子的'飞翔'速度也是一个 D 维的向量,记为 $\vec{v}_i=(v_{i1},v_{i2},\cdots,v_{iD})$。记第 i 个粒子迄今为止搜索到的最优位置为 $\vec{p}_i=(p_{i1},p_{i2},\cdots,p_{iD})$,整个粒子群迄今为止搜索到的最优位置为 $\vec{p}_g=(p_{g1},p_{g2},\cdots,p_{gD})$。

kennedy 和 Eberhart 最早提出的 PSO 算法[1]采用下列公式对粒子操作:

$$v_{id}=v_{id}+c_1r_1(p_{id}-x_{id})+c_2r_2(p_{gd}-x_{gd}) \tag{1}$$

$$x_{id}=x_{id}+v_{id} \tag{2}$$

其中,$i=1,2,\cdots,m$,$d=1,2,\cdots,D$;学习因子 c_1 和 c_2 是非负常数;r_1 和 r_2 是介于[0,1]之间的随机数。$v_{id}\in[-v_{\max},v_{\max}]$;$v_{\max}$ 是常数,由用户设定。

迭代中止条件根据具体问题一般选为最大迭代次数或(和)粒子群迄今为止搜索到的最优位置满足预定最小适应阈值。

因为 $\vec{p}_g$ 是整个粒子群的最优位置,因此上述 PSO 算法也被称为全局版 PSO。也可以把第 i 个粒子的邻居们搜索到的最优位置作为 $\vec{p}_g$,则上述方法又被称为局部版 PSO[2]。全局版 PSO 收敛速度快,但有时会陷入局部最优。局部版 PSO 收敛速度慢一点,但相对的不易陷入局部最优。

文献[3]对(1)式作了如下的改动:

$$v_{id}=wv_{id}+c_1r_1(p_{id}-x_{id})+c_2r_2(p_{gd}-x_{gd}) \tag{3}$$

其中 w 是非负数,称为惯性因子。

而文献[4]对(2)式作了以下改动:

$$x_{id}=x_{id}+av_{id} \tag{4}$$

其中 a 称为约束因子,目的是控制速度的权重。

目前,也有许多学者把方程(3),(4)视作基本的 PSO 算法。基本 PSO 算法需要用户确定的参数并不多,而且操作简单,故使用比较方便。但是它的缺点是易陷入局部极小点,搜索精度不高,因此研究者们对其作了各种各样的改进。

2.2 自适应 PSO 算法

2.2.1 自适应 PSO 算法

文献[3]研究了惯性因子 w 对优化性能的影响,发现较大的 w 值有利于跳出局部极小点,而较小的 w 值有利于算法收敛,因此提出了自适应调整 w 的策略,即随着迭代的进行,线性地减小 w 的值。这种方法的进一步发展是用模糊规则动态地修改 w 的值[5],即构造一个 2 输入、1 输出的模糊推理机来动态地修改惯性因子 w。模糊推理机的两个输入分别是当前 w 值,以及规范化的当前最好性能演化(the normalized current best performance evaluation,NCBPE);而输出是 w 的增量。CBPE 是 PSO 算法迄今为止发现的最好候选解的性能测度。CBPE 可以有不同的定义方法,但是一般 CBPE 可以定义为最好候选解的适应值。NCBPE 用下式计算

$$NCBPE=\frac{CBPE-CBPE_{\min}}{CBPE_{\max}-CBPE_{\min}} \tag{5}$$

其中 $CBPE_{\max}$ 和 $CBPE_{\min}$ 分别是 CBPE 可能取值的上下限。

模糊推理机的输入、输出的论域定义为 3 个模糊集合:LOW,MEDIUM 和 HIGH,相应的模糊隶属度函数分别是 *leftTriangle*,*Triangle*,*rightTriangle*(隶属度函数的图形如图 1)。模糊推理机定义了 9 条规则进行模糊推理,决定当前 w 的增量。

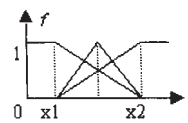

图 1 隶属度函数

类似地,约束因子对 a 优化性能的影响也引起了人们的注意,并提出了自适应改变 a 的值从而改善优化性能的策略[4]。

这一类自适应 PSO 算法对许多问题都能取得满意的结果。通过自适应调整全局系数,兼顾搜索效率和搜索精度,是一类有效的算法。但是对许多复杂的非线性优化问题,试图通过自适应调整一个全局系数提高搜索精度的余地是有限的。

2.2.2 杂交 PSO 算法

借鉴遗传算法的思想,文献[6]最早提出了杂交 PSO 算法的概念。文献[7]进一步提出具有繁殖和子群的 HPSO 算法。粒子群中的粒子被赋予一个杂交概率,这个杂交概率是用户确定的,与粒子的适应值无关。在每次迭代中,依据杂交概率选取指定数量的粒子放入一个池中。池中的粒子随机地两两杂交,产生同样数目的孩子粒子,并用孩子粒子代替父母粒子,以保持种群的粒子数目不变。孩子粒子的位置由父母粒子的位置的算术加权和计算,即

$$child_1(\vec{x})=\vec{p}*parent_1(\vec{x})+(1-\vec{p})*parent_2(\vec{x}) \tag{6}$$

$$child_2(\vec{x})=\vec{p}*parent_2(\vec{x})+(1-\vec{p})*parent_1(\vec{x}) \tag{7}$$

其中 $\vec{x}$ 是 D 维的位置向量,而 $child_k(\vec{x})$ 和 $parent_k(\vec{x})$,$k=1,2$ 分别指明是孩子粒子还是父母粒子的位置;$\vec{p}$ 是 D 维均匀分布的随机数向量,$\vec{p}$ 的每个分量都在 [0,1]取值。

孩子粒子的速度分别由下面的公式得到

$$child_1(\vec{v})=\frac{parent_1(\vec{v})+parent_2(\vec{v})}{\left|parent_1(\vec{v})+parent_2(\vec{v})\right|}\left|parent_1(\vec{v})\right| \tag{8}$$

$$child_2(\vec{v})=\frac{parent_1(\vec{v})+parent_2(\vec{v})}{\left|parent_1(\vec{v})+parent_2(\vec{v})\right|}\left|parent_2(\vec{v})\right| \tag{9}$$

其中 $\vec{v}$ 是 D 维的速度向量,而 $child_k(\vec{v})$ 和 $parent_k(\vec{v})$,$k=1,2$ 分别指明是孩子粒子还是父母粒子的速度。

对局部版的 PSO 算法而言,相当于在一个种群中划分了若干个子群,因此,杂交操作既可以在同一子群内部进行,也可

以选择在不同的子群之间进行。

实验结果显示[6,8]，杂交 PSO 算法的收敛速度比较快，搜索精度也相对比较高，对一类非线性优化问题可以得到满意的结果。不过引入了较多的待调整参数，对使用者的经验有一定要求。

2.2.3 协同 PSO 算法

文献[8,9]提出了一种协同 PSO 算法，其基本思想是用 K 个相互独立的粒子群分别在 D 维的目标搜索空间中的不同维度方向上进行搜索。具体做法是：选定划分因子(split factor)K 和粒子群的粒子数 M，将输入的 D 维向量(粒子的位置及速度向量)划分到 K 个粒子群。前 $D\bmod K$ 个粒子群，其粒子的位置及速度向量都是 D/K 维的；而后 $K-(D\bmod K)$ 个粒子群，其粒子的位置及速度向量也是 D/K 维的。在每一步迭代中，这 K 个的粒子群相互独立地进行状态更新，粒子群之间不共享信息。计算适应值时，将每个粒子群中最优粒子的位置向量拼接起来，组成 D 维向量并代入适应函数计算适应值。例如，考虑有 24 个自变量的优化问题，即 D=24。如果选取 M=10，K=5，那么每个粒子的位置及速度向量的维数就是 D/K=5。

这种协同 PSO 算法有明显的“启动延迟”(start-up delay)现象[7]，在迭代初期，适应值下降缓慢，换言之，收敛速度慢。文献[7]的实验结果显示粒子群数目越大，收敛越慢。不过这种协同 PSO 算法因为实际上采用的是局部学习策略，因此比基本的 PSO 算法更易跳出局部极小点，达到较高的收敛精度。

2.2.4 离散 PSO 算法

基本的 PSO 算法是在连续域中搜索一个数值函数的最小值的有力工具。文献[10]对此作了扩展，提出了一种离散二进制版的 PSO 算法。而文献[11]推广了这一工作，研究了离散版的 PSO 算法，并将其应用于旅行商问题(TSP)的求解，取得了较好的结果。离散 PSO 算法扩展了基本 PSO 算法的应用领域，尤其是让人看到了在一类组合优化问题中的应用前景。

3 PSO 应用

PSO 算法一提出就吸引了广泛的注意，各种关于 PSO 算法应用研究的成果不断涌现，有力地推动了 PSO 研究。PSO 算法的应用领域可以划分为：函数优化、神经网络训练、工业系统优化与控制以及其他遗传算法的应用领域等。

许多实际问题都可以归结为函数优化问题，PSO 算法用于一般的函数优化问题并不令人意外。值得注意的是将 PSO 算法用于各种复杂的优化问题也已经取得了一些进展，例如，文献[11]将离散 PSO 算法用于求解 TSP 问题；文献[12],[13]研究了 PSO 算法在噪声和动态环境下的优化问题；文献[14]将 PSO 算法用于多目标优化(multiobjective problems，MO)问题等，都取得了令人感兴趣的结果。

文献[8]将 PSO 算法用于训练积单元神经网络(product unit neural networks)进行模式分类，取得了很好的效果，显示 PSO 算法是一种很有希望的训练神经网络的手段。文献[15]尝试将 PSO 算法用于训练模糊前向神经网络进行模式分类，从而从网络的输出中抽取规则。也取得了满意的效果。

文献[16,17]分别将 PSO 算法用于实际系统的模拟和控制问题，也都得到了很好的效果。

总之，PSO 算法的应用十分广泛，它有着比较好的发展前景，值得大家做进一步的研究。

4 进一步的问题

PSO 的研究尚处于初期，还有许多问题值得研究。笔者认为，以下几个问题值得关注：

(1)PSO 的理论基础研究还很贫乏，研究者们还不能对 PSO 的工作机理给出恰当的数学解释。文献[18]对此问题作了一些有益的探索，但还是远远不够的，还需要就此问题做大量深入的研究。

(2)由于实际问题的多样性和复杂性，尽管目前已提出了许多改进的 PSO 算法，但是研究新算法的动力并未减弱，还有许多工作可做。例如，虽然已有学者研究了用离散 PSO 算法求解 TSP 问题的方法[11]，但是所得到的结果还是很初步的，所用算例比文献[19]所用算例的复杂度要小得多。

(3)开拓新的 PSO 算法的应用领域是一项有价值的工作，因为 PSO 算法的生命力在于工程应用。

(收稿日期：2002 年 7 月)

参考文献

1.Kennedy J，Eberhart R.Particle Swarm Optimization[C].In：IEEE Int'l Conf on Neural Networks，Perth，Australia，1995：1942~1948

2.Eberhart R，Kennedy J.A New Optimizer Using Particle Swarm Theory[C].In：Proc of the Sixth International Symposium on Micro Machine and Human Science，Nagoya，Japan，1995：39~43

3.Shi Y，Eberhart R.A modified particle swarm optimizer[C].In：IEEE World Congress on Computational Intelligence，1998：69~73

4.Clerc M.The Swarm and the Queen：Towards a Deterministic and Adaptive Particle Swarm Optimization[C].In：Proc of the Congress of Evolutionary Computation，1999：1951~1957

5.Shi Y，Eberhart R C.Fuzzy Adaptive Particle Swarm Optimization[C].In：Proc Congress on Evolutionary Computation，Seoul，Korea，2001

6.Angeline P J.Evolutionary Optimization Versus Particle Swarm Optimization：Philosophy and Performance Differences[C].In：Evolutionary Programming VII，1998：601~610

7.Lovbjerg M，Rasmussen T K，Krink T.Hybrid Particle Swarm Optimiser With Breeding and Subpopulations[C].In：Proc of the third Genetic and Evolutionary Computation Conference，2001

8.Van den Bergh F，Engelbrecht A P.Training Product Unit Networks Using Cooperative Particle Swarm Optimizers[C].In：Proc of the third Genetic and Evolutionary Computation Conference(GECCO)，San Francisco，USA，2001

9.Van den Bergh F，Engelbrecht A P.Effects of Swarm Size on Cooperative Particle Swarm Optimizers[C].In：Proc of the GECCO，San Francisco，USA，2001

10.Kennedy J，Eberhart R.A Discrete Binary Version of the Particle Swarm Algorithm[C].IEEE Int'l Conf on Computational Cybernetics and Simulation，1997：4104~4108

11.Clerc M.Discrete Particle Swarm Optimization Illustrated by the Traveling Salesman Problem.http：//www.mauriceclerc.net，2000

12.Carlisle A，Dozier G.Adapting Particle Swarm Optimization to Dynamic Environments[C].In：Proc of Int'l Conf on Artificial Intelligence，Las Vegas，Nevada，USA，2000：429~434

13.Carlisle A，Dozier G.Tracking changing extrema with particle swarm optimizer[R].Technical Report CSSE01-08，Auburn University，2001

14.Parsopoulos K E，Vrahatis M N.Particle Swarm Optimization Method in Muliobjective Problems[C].In：Proc of the SAC，2002

(下转 17 页)

选择一个最小元冲突函数值等于 0 的问题的原因是因为它是一个用来评估性能的较好的例子。

应该注意到 Potts 方法能够为证据找到一个优化的划分，不一定获得最佳划分。每条获得的证据的平均的元冲突比每一类中证据的两两冲突小，就可以认为是一个较好的结果。

4.4 原型抽取过程

通过聚类处理过程，每条证据都被放到它所属于的子集中。但有一些证据可能属于几个不同的集合。这样的证据项用途不大，不能作为信息原型。

首先要找到一个度量子集可信度的方法。一条证据不可能属于一个可信度为 0 的子集，它必须和子集有所关联。即，一条证据属于某个子集而不是其它子集的程度，主要和它在该子集的作用的重要性有关。

当证据 e_q 在 X_j 被使用，e_q 的可信度 a_j 的计算公式如下：

$$a_j=\frac{[Pls(e_q\in x_j)]^2}{\sum_K Pls(e_q\in x_k)} \tag{11}$$

每条证据都可以作为它的最大信用子集的潜在的原型。

可以看出对于 e_q 可信度 a_j 的最大取值等于 m $(e_q\notin X_j)$对所有 j 的最小取值。

这样就得出了一个简单的判别规则：每一条证据 e_q 并且为所有 j 能找到 $m(e_q\notin X_j)$最小值，则可以认为 e_q 是 X_j 的潜在的原型。

$m(e_q\notin X_j)$的值可以通过观察把一条证据从一个子集移到另一个子集所引起的聚类中的冲突的变化得到。

如果一条证据 e_q 在 X_j 被从子集中移出，X_j 中的冲突 c_j 减少到 c_j^*。$c_j-c_j^*$ 作为 e_q 不属于 X_j 的证据。得出：

$$m(e_q\notin X_j)=\frac{c_j-c_j^*}{1-c_j^*} \tag{12}$$

无论如何，必须利用可信度来为子集确定其原型。通过公式(12)的计算，求出每个子集中所有潜在原型的可信度，并对其进行排序，找出最大的一个，作为该子集的原型。

这样，就选择了 N 个对子集来说是最高可信度的原型。它们能够极好地代表每个子集的特征。将来的分类就是把新得到的证据与它们比较，把新证据归到最合适的子集中。

第二个决策规则描述如下：

(1)从子集潜在的 N 个原型中选择最高可信度的一个，作为它的当前原型。

(2)忽略该子集其它的潜在原型。

通过应用第一个决策规则对所有的证据和第二个规则对所有的每个子集，对每个子集都能找到一个 N 个潜在的原型。最后，对每个子集中所有的原型，结合成一个新的概率值，这样每个子集将只包含一条证据，将用它来实现快速分类处理。

4.5 快速分类处理

至此，已经得到了每个子集的所有原型，因此能够对将来得到的证据进行快速分类。如果证据 e_q 对每个子集都不兼容，将不把证据 e_q 分到任何子集 X_j。对 e_q 来说，如果它最适合的子集所带来的冲突比最低极限高，则抛弃这条证据。否则，就把该证据分类到 $m(e_q\notin X_j)$最小的子集中。

如果在每个聚类中，原型的最大数目是确定的，那么分类可以在有限的时间内完成。并不依赖原先聚类过程中的证据的总数量。

4.6 信息融合处理

新的情报通过预处理分类并被放到最合适的子集，便可以与以前的情报进行融合。由于任何一个子集中的情报不依赖于其它子集中的情报，所以每个子集可以独立地进行各自的信息融合过程，这些融合过程可以并行进行。

随时间的变化，情报可能失去了其时效性，这样可能会在信息融合过程中发现不必要的冲突增强。此时，应该对当前的情报进行重新划分和分类。保证信息融合的实时性。

5 结论

通过上述一连串的处理过程，能够对不同主题的情报作出及时的处理。并且可以对实时获取的情报通过预处理在信息融合前进行快速分类。这种处理方法利用了智能信息处理技术，可以高效地完成任务，具有较高的实用价值。通过对算法的测试，达到了预期结果。今后的工作是，结合具体情报处理和信息融合任务要求，完成一个综合情报处理平台。

(收稿日期：2002 年 7 月)

参考文献

1.Johan Schubert.Managing Inconsistent Intelligence[C].In：The third international conference of Information Fusion.Sunnyvale，CA，2000

2.P Smets，R kennes.The transferable belief model[J].Artificial Intelligence，1994

3.康耀红编著.信息融合技术[M].西安：西安电子科技大学出版社，1997

4.范明，孟晓峰编译.数据挖掘概念与技术[M].北京：机械工业出版社，2001

(上接 3 页)

15.He Z，Wei C，Yang L et al.Extraction Rules from Fuzzy Neural Network by Particle Swarm Optimization[C].In：IEEE Int'l Conf On Evolutionary Computation，Anchorage，Alaska，USA，1998

16.Brandstatter B，Baumgartner U.Particle Swarm Optimization-Mass-Spring System Analogon[J].IEEE Trans on Magnetics，2002；38(2)：997~1000

17.Ciuprina G，Ioan D，Munteanu I.Use of Intelligent-Particle Swarm Optimization in Electromagnetics[J].IEEE Trans on Magnetics，2002；38(2)：1037~1040

18.Clerc M，Kennedy J.The Particle Swarm-Explosion，Stability，and Convergence in A Multidimensional Complex Space[J].IEEE Trans on Evolutionary Computation，2002；6(1)：58~73

19.吴斌，史忠植.一种基于蚁群算法的 TSP 问题分段求解算法[J].计算机学报，2001；24(12)：1328~1333