一种改进粒子群优化算法

朱玉平
(南京邮电大学,江苏 南京 210003)

摘 要:为了提高粒子群优化算法的性能,提出了一种惯性权值调整的改进粒子群优化算法,该算法的惯性权值满足不同粒子对全局和局部搜索能力的不同需求,每次迭代后根据适应度值对惯性权值做相应的调整。对4个典型的测试函数进行仿真表明,该算法比标准粒子群优化算法有更好的收敛性和更快的收敛速度,改善了优化性能。

关键词:粒子群优化算法;适应度;惯性权值
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2008)11-0106-03

An Algorithm of Modified Particle Swarm Optimization

ZHU Yu-ping
(Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: To enhance the performance of the particle swarm optimization, the individual inertia weight adjustment particle swarm optimization is proposed. Each particle has an individual inertia weight, which can provide the different global and local searching performances for particles. The inertia weight will be adjusted by the fitness of adaptive degree. Experimental results show that the new algorithm can greatly improve the global convergence ability and enhance the rate of convergence.

Key words: particle swarm optimization; adaptive degree; inertia weight

0 引 言

粒子群优化算法PSO(Particle Swarm Optimization)是由Eberhart和Kennedy[1,2]引入了一种新的基于种群的优化算法。其最初的出发点是为了模拟鸟群、鱼群、蜂群等生物群体的社会性行为。与遗传算法和蚁群算法相类似,它也是基于群智能的进化计算技术(evolutionary computation),该算法近年来已引起许多研究人员的关注,已广泛应用于函数优化、神经网络训练、模糊系统控制等诸多领域。文中针对粒子群优化算法存在易于陷入局部最优及早熟收敛,并且对多峰值函数的搜索效果不理想等缺点,提出了一种对惯性权值动态改进的粒子群优化算法。

1 PSO基本原理

粒子群优化算法模拟鸟群的捕食行为,一群鸟在随机搜索食物,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,那么找到食物最简单有效的办法就是搜寻目前离食物最近的鸟的周围区域。PSO算法模拟鸟集群飞行觅食的行为,通过鸟之间的集体协作使群体达到最优。在PSO系统中,每个备选解被称为一个"粒子"(particle),多个粒子共存、合作寻优(近似鸟群寻找食物),每个粒子根据它自身的"经验"和相邻粒子群的最佳"经验"在问题空间中向更好的位置"飞行",搜索最优解。PSO算法本质上就是一种基于群体和适应度的优化算法。粒子群中每一个粒子代表了系统的一个潜在解,每个粒子用3个指标来表征:位置、速度、适应度。

在PSO中,每个粒子都认为是D维空间内的一个点。第i个粒子(或称种群成员)可以表示为$X_I=(x_{i1},x_{i2},\cdots,x_{iD})$;第$i$个粒子以前最好的位置(对应的目标函数或者适应度值最小)被保存下来并表示为$P_I(p_{i1},p_{i2},\cdots,p_{iD})$,在$P_I$向量中最好的粒子成员下标用$g$表示(全局最优)。第$i$个粒子位置变化的快慢用$V_I=(v_{i1},v_{i2},\cdots,v_{iD})$表示。

粒子群优化算法的基本递推公式[3,4]如下:

$$v_{id}=w*v_{id}+c_1*\text{rand}()*(p_{id}-x_{id})+c_2*\text{Rand}()*(p_{gd}-x_{id}) \quad (1)$$

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

其中,c_1、c_2为两个正常数,rand()和Rand()是在[0,1]范围内的随机数,w为惯性权值。

收稿日期:2008-05-31
作者简介:朱玉平(1974-),男,江苏靖江人,硕士,讲师,主要研究方向为计算机在通信中的应用、智能优化算法等。

从图 1 中可以更为直观地看出 PSO 算法的优点：在二维空间 (x,y) 上以一定原始速度运动的粒子 1 和粒子 2，在以前最好位置 $pbest_1$ 和 $pbest_2$ 以及全局最好位置 gbest 的共同作用下，将偏离原来的运动方向而向着与最好位置接近的方向运动。通过一次次的运动搜索，最终摆脱一个个局部最优点，而收敛到全局唯一最优点。

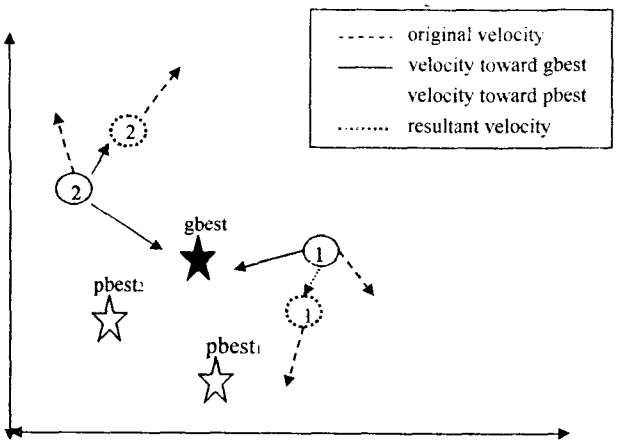

图 1　PSO 算法基本原理示意图

2　粒子群优化算法的改进

虽然基本的 PSO 具有算法收敛速度快和简便易行等优点，但是在算法中所有的粒子采用统一的惯性权值，忽视了粒子的独特性特点，降低了种群的多样性，易于早熟收敛、搜索精度不高，易于陷入局部最优解。基于模糊系统的惯性因子的动态调整 W 值由最大惯性权值 W_{max} 线性减少到最小惯性权值 W_{min}，可明显改善算法的收敛性能[5,6]；但它也有缺点，迭代初期局部搜索能力较弱，即使初始粒子已接近全局最优解，却无法搜索到，而在迭代后期，全局搜索能力较弱，而易于陷入局部最优，即使在同一迭代时期，处于不同位置的粒子对全局和局部搜索的要求也不一样。因此，文中提出的 MPSO 算法在粒子群惯性权值 W 总体减少的同时，针对每个粒子所处的不同位置惯性权值 W 再做相应的微调。研究实验表明：w 较大时侧重于全局搜索、w 较小时侧重于局部搜索；在其它参数都相同的条件下，若能找到全局最优解，则 w 较小时迭代次数少、运算量低。这是因为 w 较大时花费了不少的运算量在新的无关区域上进行了搜索。假定目标是求最小化问题，MPSO 算法的惯性权值 W 动态按如下式变化：

$$w_i^t = W_{max}(W_{max} - W_{min})t/T_{max} \tag{3}$$

$$x_i^t = F_g^t/F_i^t \tag{4}$$

$$f(x_i^t) = |1 - x_i^t| \tag{5}$$

$$w_i^t = (W_{max} - W_{min})f(x_i^t) + w_i^t \tag{6}$$

t 为当前迭代次数；w_i^t 为粒子 i 第 t 次迭代时的惯性权值；T_{max} 为算法总迭代次数；F_g^t 和 F_i^t 分别是截止到第 t 次迭代时种群搜索到的全局最小值和粒子 i 第 t 次迭代时的适应度值。W_{max} 为最大惯性权值；W_{min} 为最小惯性权值（根据研究实验取 $W_{min} = 0.3$，$W_{max} = 1.2$ 时搜索效率较高）。C_1 和 C_2 是学习因子，通常 $C_1 = C_2 = 2$。

MPSO 算法步骤如下：

步骤 1：初始化粒子种群维数、粒子规模（成员数）、速度、迭代次数等参数。

步骤 2：计算每个粒子的适应度值。

步骤 3：比较以前最优位置对应的适应度值与当前目标函数值。如果当前值更好（如更小），则用当前 x_{id} 取代 p_{id}；否则，p_{id} 保持不变。

步骤 4：经过一次迭代以后，重新计算下标 g，以确定新的全局最优点 p_{gd}。

步骤 5：利用式(1) 更新速度并限制其最大范围、利用式(2) 更新粒子位置并限制其范围；根据式(6) 调整惯性权值 W。

步骤 6：检查是否满足 PSO 算法终止条件，若否，转至步骤 2，进行下一次搜索；若是，则求出最优解。

3　算法仿真实验

为了检测 MPSO 的性能选择了 Sphere 函数、Rosenbrock 函数、Rastirgrin 函数、Griewank 函数 4 个函数进行模拟（结果见表 1～表 4），算法中粒子种群规模为 40，最大迭代次数取 1000，在达到最大迭代次数或者位置误差的平方和 $e = \sum_{d}\sum_{i=1}^{40}(p_{id} - x_{id})^2 < 0.01$ 的情况下停止搜索，$W_{min} = 0.3$，$W_{max} = 1.2$，$C_1 = C_2 = 2$，以检测改进后算法的优化效率。

表 1　Sphere 函数的实验结果

算法	维数	优化结果的均值
PSO	20	0.000161
	30	0.000226
MPSO	20	0.000061
	30	0.000083

表 2　Rosenbrock 函数的实验结果

算法	维数	优化结果的均值
PSO	20	0.856201
	30	9.176811
MPSO	20	0.073446
	30	0.212931

Sphere 函数：$f_1 = \sum_{i=1}^{n} x_i^2, x_i \in [-5.12, 5.12]$

Rosenbrock 函数：$f_2 = \sum_{i=1}^{n-1}(100(x_{i-1} - x_i^2)^2 + (x_i - 1)^2)$，$x_i \in [-2.048, 2.048]$

Rastirgrin 函数：$f_3 = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$，$x_i \in [-5.12, 5.12]$

Griewank 函数：$f_4 = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$，$x_i \in [-600, 600]$

表 3 Rastirgrin 函数的实验结果

算法	维数	优化结果的均值
PSO	20	1.103487
	30	6.025851
MPSO	20	0.132085
	30	1.828539

表 4 Griewank 函数的实验结果

算法	维数	优化结果的均值
PSO	20	0.030128
	30	0.081736
MPSO	20	0.008361
	30	0.015027

从表 1～表 4 的对 4 个函数的实验结果可以看出，MPSO 比标准 PSO 有着更好的搜索性能。

4 结束语

标准 PSO 算法由于粒子采用统一的惯性权值，忽视了粒子的独特性特点，种群的多样性不好，易于早熟收敛、搜索精度不高，易于陷入局部最优解。惯性权值 w 直接影响 PSO 算法的局部和全局搜索能力，处于不同位置的粒子对全局和局部搜索的要求也不一样，因此改进的 MPSO 算法的惯性权值 w 做动态调整，并且在粒子群惯性权值 W 总体减少的同时，针对每个粒子所处的不同位置惯性权值 W 再做相应的微调；改进后的新算法跳出局部最优，同时加快了收敛速度，具有较强的全局搜索能力，显著提高了优化性能。

参考文献：

[1] Kennedy J, Eberhart R C. Particle swarm optimization[C]// Proceeding of 1995 IEEE International Conference on Neural Networks. New York, NY, USA: IEEE, 1995: 1942 - 1948.

[2] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory[C]//Proceedings of the Sixth International Symposium on Micro Machine and Human Science. New York, NY, USA: IEEE, 1995: 39 - 43.

[3] Shi Y H, Eberhart R. Parameter selection in particle swarm optimization[C]//Proc of the 7th Annual Conf on Evolutionary Programming. Washington D C: [s. n.], 1998: 591 - 600.

[4] Shi Y, Eberhart R C. A modified particle swarm optimizer [C]//Proceedings of 1998 IEEE International Conference on Evolutionary Computation. New York, NY, USA: IEEE, 1998: 69 - 73.

[5] Shi Y, Eberhart R C. Empirical study of particle swarm optimization[C]//Proceedings of the Congresson Evolutionary Computation. Piscataway, NJ: IEEE Service Center, 1999: 1945 - 1950.

[6] Bergh F, Engelbrecht A P. A Cooperative Approach to Particle Swarm Optimization[J]. IEEE Trans. on Evolutionary Computation, 2004, 8(3): 225 - 239.

(上接第 105 页)

次化，简单明了；文本输入方法具有类似于高级程序设计语言的抽象能力，便于多人配合工作。需要注意的是两种方法下输出数据的高位和低位次序正好相反，在实际使用时，根据需要可作适当的变换。

3 结束语

这种使用集成电路芯片 74LS166 和 VHDL 语言通过现代电子系统设计方法来实现串行 CRC 编码运算中移位寄存器的方案，既增强了对数字逻辑电路的理解，又在系统灵活性和系统可移植性等方面有了一定的提高。实际结果表明，两种方法都能够完全达到预定的设计要求，有一定的实用与借鉴价值。后续工作中，CRC 译码器设计及数据的串行/并行转换模块设计值得去做进一步探讨。

参考文献：

[1] 曹雪虹，张宗橙. 信息论与编码[M]. 北京：清华大学出版社，2004.

[2] Ramabadran T V, Gaitonde S S. A Tutorial on CRC Computations[J]. Micro IEEE, 1988, 8(4): 62 - 75.

[3] Derby J H. High - speed CRC computation using state - space transformations[C]// IEEE Global Telecommunications Conference GLOBECOM'01. San Antonio, TX, United States: [s. n.], 2001: 166 - 170.

[4] 蒋安平. 循环冗余校验码(CRC)的硬件并行实现[J]. 微电子学与计算机，2007, 24(2): 107 - 112.

[5] 潘 松，黄继业. EDA 技术实用教程[M]. 北京：科学出版社，2002.