

# 具有趋向向量及迁移特征的协同 PSO 算法

邵增珍<sup>1,2</sup>, 王洪国<sup>1</sup>, 刘 弘<sup>1,2</sup>, 赵学臣<sup>1,2</sup>

(1. 山东师范大学信息科学与工程学院, 济南 250014; 2. 山东省分布式计算机软件新技术重点实验室, 济南 250014)

**摘 要:** 为提高 PSO 算法的搜索能力, 提出一种协同粒子群算法 CPSO-ADS。引入种群分布熵及群落差异度评价, 用以有效初始化群落。给出趋向向量修正粒子的位置向量, 提高算法收敛速度。运用占优子空间概念, 通过评价子空间搜索价值确定种群的迁移方向。实验结果表明, 该算法搜索性能稳定, 能以大概率收敛到全局最优。

**关键词:** 种群分布熵; 趋向向量; 占优子空间; 协同进化; 粒子群优化算法

## Cooperative PSO Algorithm with Appulsive Vector and Migration Character

SHAO Zeng-zhen<sup>1,2</sup>, WANG Hong-guo<sup>1</sup>, LIU Hong<sup>1,2</sup>, ZHAO Xue-chen<sup>1,2</sup>

(1. Institute of Information Science and Engineering, Shandong Normal University, Jinan 250014, China;

2. Shandong Provincial Key Laboratory for Distributed Computer Software Novel Technology, Jinan 250014, China)

**【Abstract】** This paper proposes a novel cooperative Particle Swarm Optimization(PSO) algorithm(CPSO-ADS) to improve the search ability of PSO algorithm. To initialize the cluster effectively, population scatter entropy strategy and cluster differential degree strategy are introduced. To improve the convergence rate, it amends the position vector of a particle by producing an appulsive vector. And to ascertain the migration direction of a population, it proposes the concept of dominant subspace to evaluate the value of the special subspace. Experimental result shows that algorithm has stable search ability and can converge to the global optimum with large probability.

**【Key words】** population scatter entropy; appulsive vector; dominant subspace; co-evolution; Particle Swarm Optimization(PSO) algorithm

DOI: 10.3969/j.issn.1000-3428.2011.21.063

### 1 概述

粒子群优化(Particle Swarm Optimization, PSO)算法<sup>[1]</sup>于1995年由Eberhart等人提出,它是一种基于种群搜索的进化计算技术,目前该算法已广泛应用于多个领域,并取得较好效果。PSO算法原理简单,但存在易陷于局部最优等问题。国内外学者从多方面改进该算法以提高其性能。文献[2]通过调整PSO公式中的参数实现了粒子自我学习及外向学习的平衡;文献[3-4]研究了粒子分布性对算法的影响;文献[5]将PSO算法同其他算法混合,取长补短,形成混合粒子群算法;文献[6]提出两分群交换粒子群优化算法;文献[7]根据生物遗传学规律,提出了双倍体差分进化粒子群算法,并将其用于车辆路径问题的求解。1964年, Ehrlich P R首次提出“协同进化”<sup>[8]</sup>的概念,但未给出确切定义。将协同理论引入进化算法的目的是为了弥补传统单种群进化算法的缺陷。通过构造群落内多种群之间的竞争、合作关系,提高算法的性能,以实现各种优化目的。

基于文献[9]中的 PSO\_TVAC 算法,本文提出一种新的协同粒子群算法 CPSO ADS。针对标准 PSO 算法容易陷于局部最优的不足,提出基于分布熵及种群差异度的群落初始化调整策略与基于趋向向量的粒子位置向量修正策略,以及基于占优子空间的迁移策略。CPSO ADS 算法中多个种群协同进化,共同完成优化任务。

### 2 协同粒子群算法 CPSO ADS

#### 2.1 基于分布熵的种群初始化调整策略

种群在初始化过程中,需要保证个体分布的随机性。本

文提出对搜索空间进行有效划分的“等分维度法”,实际上是对搜索空间的每一维都进行划分,每个划分称为子段,种群的分布熵取各维分布熵的加权和。设问题搜索空间  $\Omega \subset R^n$ , 空间中个体总数为  $N$ 。

**定义 1** 设每个子段中个体分布数目为  $N_{ij}$ ,  $i=1,2,\dots,n$ ,  $j=1,2,\dots,K$ , 称:

$$S_i = -\sum_{j=1}^K \frac{N_{ij}}{N} \lg \frac{N_{ij}}{N} \quad (1)$$

为种群在第  $i$  维的种群分布熵, 称:

$$S = (\alpha_1 S_1 + \alpha_2 S_2 + \dots + \alpha_n S_n) = \sum_{i=1}^n \alpha_i S_i \quad (2)$$

为种群在空间  $\Omega$  内的分布熵。其中,  $K$  为常数,表示搜索空间中某维的子段数目。

在式(2)中,  $\alpha_i \in [0,1]$ ,  $\alpha_i \in R^+$ , 且  $\sum \alpha_i = 1$ 。特别地,可取  $\alpha_i = 1/n$ , 此时式(2)简化为:

$$S = \frac{1}{n} \sum_{i=1}^n S_i \quad (3)$$

为保证粒子分散性,要求初始化种群时其分布熵取得较大值。设  $S_0 < 1$  为熵阈值,系统可根据情况对初始种群进行简

**基金项目:** 国家自然科学基金资助项目(60970004);山东省科技攻关计划基金资助项目(2009GG10001008);济南市高校院所自主创新基金资助项目(200906001)

**作者简介:** 邵增珍(1976—),男,副教授、博士研究生、CCF 会员,主研方向:智能计算;王洪国、刘 弘,教授、博士生导师;赵学臣,硕士研究生

**收稿日期:** 2011-04-11 **E-mail:** shaozenghen@163.com

单扩散调整, 将粒子均匀分布在具有最小分布熵的第  $i$  维的各子段中, 从而使得  $S_i=1$ 。由于种群在第  $i$  维上调整分布时会影响其他维粒子的分布状态, 种群分布熵  $S$  需重新计算。

## 2.2 具有趋向向量粒子位置的修正

对粒子  $r$ , 文献[9]中有关微粒速度向量的计算公式为:

$$v_r(t+1) = \omega \times v_r(t) + c_1 \times rand1() \times (p_r - x_r) + c_2 \times rand2() \times (p_g - x_r) \quad (4)$$

其中,  $\omega=0.5+rand3()/2$ ;  $c_1=(c_{1f}-c_{1r})t/\max T+c_{1r}$ ;  $c_2=(c_{2f}-c_{2r})t/\max T+c_{2r}$ 。修正粒子位置计算公式同文献[1]。rand1()、rand2()和rand3()是独立随机函数, 取值范围为[0,1]。 $\omega$ 的平均取值为0.729,  $c_1$ 的取值从2.5降至0.5,  $c_2$ 的取值从0.5增至2.5<sup>[9]</sup>。

为修正粒子的位置, 在位置计算公式中增加一个趋向向量。当第  $t$  次迭代时, 随机选择第  $i$  维, 按均匀分布原则在每子段内随机生成一个静态粒子(该粒子不属于种群, 且速度为0)。计算该子段内所有静态粒子的平均适应度作为该子段的适应度, 记  $f_{ik(t)}$ ,  $k=1,2,\dots,K$ 。假设  $f_{im(t)}=\max\{f_{ik(t)}|k=1,2,\dots,K\}$  为具有最高适应度的子段(称为吸引子空间), 从统计意义上看, 该子段所定义的子空间中包含全局最优点的概率也最大, 故有必要增强所有粒子飞向该区域的概率。称吸引子空间中适应值最高的静态粒子为吸引粒子, 其位置向量记为  $x_a$ 。对每个粒子  $r$ , 构建其指向吸引粒子的标准方向向量  $D_r(t+1)=std(x_a-x_r(t))$ ,  $std()$ 的作用是将向量单位化, 使得  $|D_r(t+1)|=1$ 。称向量  $D_r(t+1)$  为粒子  $r$  在第  $t+1$  次迭代时的趋向向量。基于此, 对粒子  $r$  的位置变化公式修改为:

$$x_r(t+1) = x_r(t) + v_r(t+1) + \gamma |v_r(t+1)| \cdot D_r(t+1) \quad (5)$$

式(5)增加了  $\gamma |v_r(t+1)| \cdot D_r(t+1)$  作为启发项。 $\gamma$  称为趋向向量影响因子, 同迭代次数  $t$  呈正相关性, 用于调整启发项的影响能力, 一般可设  $\gamma \in [1,3]$ 。启发项具有较强的导向性, 是增大粒子选择正确飞行方向概率的重要因素。实验结果表明, 趋向向量在很小计算代价的基础上提高粒子搜索的目的性, 能有效加快收敛速度。吸引粒子的启发作用如图1所示。

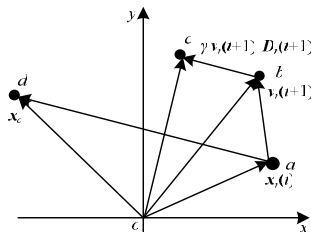


图1 粒子启发作用示意图

在图1中, 假设圆点  $a$  为粒子  $r$  在第  $t$  次迭代时所处位置, 圆点  $d$  为吸引粒子位置。标准 PSO 算法中, 经第  $t+1$  次迭代时速度向量的修正, 粒子  $r$  的位置将偏移到圆点  $b$ 。在吸引粒子的吸引作用下, 系统产生趋向向量  $D_r(t+1)$ , 该向量的正相关向量  $\gamma |v_r(t+1)| \cdot D_r(t+1)$  对位置  $b$  又增加了一个飞向位置  $d$  的拉动作用, 使得粒子  $r$  在第  $t+1$  次迭代时位置偏移到圆点  $c$  位置。一般来说, 位置  $c$  比位置  $b$  更接近吸引粒子。影响因子  $\gamma$  越大, 粒子飞向吸引粒子的趋势越明显, 因此, 系统运行后期增大影响因子是有必要的。

## 2.3 基于子空间搜索价值评价的种群迁移策略

种群从开始进化到当前状态, 所有曾落在某子段  $k$  中的粒子的数目, 称为子空间的历史搜索密度, 用  $H\rho_k$  表示。历史搜索密度记录, 实际上是算法迭代过程中的搜索痕迹。研

究发现, 搜索价值较大的子段需满足: (1)历史搜索密度较小; (2)平均适应度较大。基于此, 定义子段  $k$  的搜索价值评价函数为:

$$Eval(k) = f(H\rho_k, \overline{EF}_k) = \Gamma(1 - \frac{H\rho_k}{\max_j H\rho_j}) + (1-\Gamma) \frac{1}{H\rho_k} \sum_{p=1}^{H\rho_k} fitness(p) \quad (6)$$

其中,  $\overline{EF}_k$  表示子段  $k$  中所有粒子的平均适应度;  $p$  表示粒子; 参数  $\Gamma \in (0, 1)$  称为调节系数, 用于确定历史搜索密度和子段的平均适应度对搜索价值的贡献, 其取值需根据具体问题设定为常量, 或设定其同迭代次数具有某种相关性, 具体如下:

当算法初次陷于局部最优时, 大多数粒子聚集的区域的搜索密度较大, 该区域是算法需要跳出的区域, 则可设此时的调节系数  $\Gamma$  较小; 随着程序的持续运行, 当算法执行后期再次停滞时, 此时大多数粒子聚集的区域很可能是全局最优区域, 则可设置较大取值的  $\Gamma$ 。总之,  $\Gamma$  的取值在算法执行后期随着迭代次数的增加而增加, 尤其是当检测到算法处于停滞状态时,  $\Gamma$  的变化幅度将加大。

在第  $t$  次迭代时, 系统随机选择某一维  $i$ ,  $i=1,2,\dots,n$ , 利用式(6)计算各子空间搜索价值, 最大值对应的子空间称为占优子空间, 记为  $DS$ 。为加快群体逃逸速度, 本文采用迁移策略, 选择  $DS$  中最优的  $q$  个个体直接取代群体中最差的  $q$  个个体, 粒子速度变为最小  $V_{\min}$ , 方向随机, 种群进入下一次迭代继续进化。

## 2.4 多种群粒子群算法初始化及协同进化过程

### 2.4.1 基于群落差异度的群落初始化调整策略

多个种群以及种群生存的生境构成群落。从全局考虑, 种群差异性越大, 个体对搜索空间的覆盖率就越高, 算法得到全局最优解的概率就越大。以二维搜索区域为例, 考虑如图2所示情形。假设矩形区域为搜索空间  $\Omega$  的一个子空间  $\delta$ , 圆形和小正方形各代表2个种群(设为  $Pop_p$  和  $Pop_q$ ) 在  $\delta$  上的分布情况。 $m_p$  和  $m_q$  分别代表2种群分布在  $\delta$  中的个体的平均位置, 称为中心点。

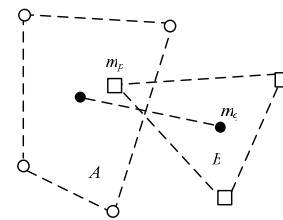


图2 子空间内个体分布及中心点

**定义2(种群子空间距离)** 将2个中心点之间的欧氏距离  $\Delta dis_{pq}^{\delta} = |m_q - m_p|$  称为种群  $Pop_p$ 、 $Pop_q$  在子空间  $\delta$  上的距离。

**定义3(种群距离)** 假设在某一维上将空间  $\Omega$  分为  $K$  个子空间(子段), 称:

$$\overline{Edis}_{pq} = \frac{1}{K} \sum_{k=1}^K \Delta dis_{pq}^k \quad (7)$$

为种群  $Pop_p$ 、 $Pop_q$  在空间  $\Omega$  上的距离。

从统计意义上看, 距离越大, 种群  $Pop_p$ 、 $Pop_q$  在空间  $\Omega$  上的差别越大, 对  $\Omega$  的覆盖性就越好。当  $K=1$  时, 种群子空间距离同种群距离相等; 当  $K>1$  时, 种群距离是种群子空间距离的数学期望。需要说明的是, 此处的子空间是从  $n$  维空间中随机选择其中一维, 并对之  $K$  等分而得到的子段划分。

显然, 当  $K>1$  时的种群距离更能体现种群在搜索空间上分布的差异性。

**定义 4**(群落距离矩阵) 以  $Q$  个种群在空间  $\Omega$  上的种群距离为元素构造的矩阵  $A$  :

$$A = \begin{bmatrix} 0 & \overline{Edis}_{12} & \overline{Edis}_{13} & \dots & \overline{Edis}_{1Q} \\ \overline{Edis}_{21} & 0 & \overline{Edis}_{23} & \dots & \overline{Edis}_{2Q} \\ \overline{Edis}_{31} & \overline{Edis}_{32} & \dots & \dots & \overline{Edis}_{3Q} \\ \dots & \dots & \dots & \dots & \dots \\ \overline{Edis}_{Q1} & \overline{Edis}_{Q2} & \overline{Edis}_{Q3} & \dots & 0 \end{bmatrix}$$

称为群落的距离矩阵。

为方便比较, 将矩阵  $A$  的最大元素设定为 1, 其余元素同比例缩放, 得到群落的标准距离矩阵  $SA$ 。假设  $SA$  中的元素用  $Edis_{pq}$  表示, 该矩阵有如下特点: (1)  $Edis_{pp}=0$ ; (2)  $Edis_{pq}=Edis_{qp}$ 。其中,  $p, q=1, 2, \dots, Q$ 。

**定义 5**(群落差异度) 在  $SA$  上, 称和值  $Cdiff = \sum_{p>q} Edis_{pq}$  为群落的差异度。

$Cdiff$  的取值范围为  $[0, Q(Q-1)/2]$ , 其值越大, 说明种群之间的差别越大。当  $Cdiff \geq Q(Q-1)/5$  时, 可认为各种群的分布是可接受的, 否则将对群落进行扩散性调整。调整步骤为:

**Step1** 计算群落距离标准矩阵  $SA$  每行的和值  $\sum_{q=1}^Q Edis_{uq}$ ,  $u=1, 2, \dots, Q$ , 找出其最小值所对应的行号, 假设为第  $p$  行。

**Step2** 对种群  $Pop_p$  执行扩散操作。

**Step3** 计算群落差异度  $Cdiff$ , 如果  $Cdiff < Q(Q-1)/5$ , 则转 Step2, 否则结束。

#### 2.4.2 合作式群落协同进化过程

群落初始化完成以后, 系统进入协同进化过程。为实现种群间信息交互并减少通信成本, 本文建立了种群之间环形拓扑结构。 $Pop_i$  同  $Pop_{i-1}$ 、 $Pop_{i+1}$  建立邻居关系, 所有种群完成一次迭代后,  $Pop_i$  将向  $Pop_{i-1}$  和  $Pop_{i+1}$  发送本种群的当前最优解、当前占优子空间等信息( $Pop_Q$  将向  $Pop_1$ 、 $Pop_{Q-1}$  发送以上信息)。任何种群既是信息发送者, 同时又是信息接收者。接收到信息后, 种群通过同自身信息对比, 取最优者作为自身信息, 指导下一轮的进化过程。例如, 当  $Pop_i$  发现  $Pop_{i+1}$  搜索到的占优子空间具有更大的优势, 它将直接利用该占优子空间指导其自身进化。

显然, 种群邻居之间的通信实现了种群之间的相互学习, 进而促进了多种群间的协同进化过程。同单种群相比, 多种群协同进化可在较大程度上避免陷入局部最优, 能以较大概率指导系统搜寻到全局最优解。

## 2.5 复杂性讨论

假设群落中各种群数量总和为  $N$ , 种群数为  $Q$ , 搜索空间为  $\Omega \subset R^n$ , 其每一维被分割成的子段数目为  $K$ 。CPSO-ADS 算法在标准 PSO 算法的基础上, 增加了若干优化操作。其中, 种群分布熵的计算复杂度为  $O((K \times n \times N) + O(K \times n)) = O((K \times n \times (N+1))) \approx O(K \times n \times N)$ ; 如果种群需要扩散操作, 则其计算复杂度为  $O(K \times N)$ ; 占优子空间的计算复杂度较为复杂, 达到了  $O(t \times k \times n \times N)$ , 其中,  $t$  为迭代次数, 随着  $t$  的增加, 占优子空间的计算量逐渐增加。群落初始化的计算包括 2 个部分: (1) 计算粒子平均位置, 其复杂度为  $O(n \times N)$ ; (2) 计算群落的距离矩阵, 复杂度为  $O(Q(Q+1) \times K \times n/2)$ , 则群落初始化的计算复杂度为  $O(n \times N) + O(Q(Q+1) \times K \times n/2)$ 。如群落需要调整, 其计算机复杂度为  $O(Q^2 + Q) + O(K \times N) + O(Q(Q+1) \times K \times n/2)$ 。可见, 该算法为多项式算法。

## 3 实验结果与分析

为验证该算法的有效性, 对 PSO\_TVAC 算法、HPSO\_TVAC 算法<sup>[9]</sup>和本文提出的 CPSO-ADS 算法进行实验验证与结果对比分析。所有实验硬件环境均为 Intel PIV2.8 双核, 内存 2 GB, 软件环境 VS2008 及 Matlab7.0。

### 3.1 函数测试

本文选择实值函数对算法的性能进行测试。表 1 列出了用于实验的 5 个测试函数。与文献[9]相同, 以上函数在运行时  $V_{max}$  取值分别是 100、100、10、600、100。除函数  $f_6$  外, 其余函数  $f_1 \sim f_4$  均为多维函数, 具体实验时维数  $n$  取 20、30。所有函数的全局最小值均为 0。

表 1 Benchmark 函数

函数名称	函数形式	搜索区域	最优值
Sphere function	$f_1(x) = \sum_{i=1}^n x_i^2$	$(-100, 100)^n$	0
Rosenbrock function	$f_2(x) = \sum_{i=1}^n [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$(-100, 100)^n$	0
Rastrigrin function	$f_3(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$(-10, 10)^n$	0
Griewank function	$f_4(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1$	$(-600, 600)^n$	0
Schaffer's $f_6$	$f_6(x) = 0.5 - \frac{(\sin(\sqrt{x^2 + y^2}) - 0.5)}{(1.0 + 0.001(x^2 + y^2))^2}$	$(-100, 100)^2$	0

### 3.2 算法性能对比实验

设种群数目  $Q=5$ , 粒子总数  $N=100$ , 最大迭代次数  $G_{max}$  根据函数难度适当调整, 精度控制为  $10^{-4}$ , 每个算法执行 50 次。执行函数  $f_4$  的寻优过程时,  $K=10$ , 其余函数均设置  $K=5$ 。实验数据见表 2。表中黑体部分代表最优结果。

表 2 算法性能对比

函数	维数	$G_{max}$	平均函数值/(标准差)			收敛到最优值的次数/(平均迭代次数)		
			PSO-TVAC	HPSO-TVAC	CPSO-ADS	PSO-TVAC	HPSO-TVAC	CPSO-ADS
$f_1$	20	2 000	0.009 7/(0.076 2)	0.008 1/(0.064)	<b>0.004 6/(0.004 9)</b>	47/(589)	49/(551)	<b>49/(389)</b>
	30	3 000	0.012 2/(0.011 8)	<b>0.012 9/(0.004 9)</b>	0.008 1/(0.003 7)	48/(911)	<b>48/(899)</b>	49/(903)
$f_2$	20	4 000	18.491 2/(47.316 4)	14.917 1/(11.148 4)	<b>13.000 7/(9.111 4)</b>	11/(3 017)	17/(2 916)	<b>21/(2 889)</b>
	30	5 000	19.834 0/(38.331 5)	13.460 1/(9.024 3)	<b>13.423 7/(8.120 9)</b>	6/(4 675)	15/(4 659)	<b>27/(3 503)</b>
$f_3$	20	4 000	16.147 3/(9.041 9)	0.032 8/(2.184 2)	<b>0.031 7/(2.006 1)</b>	5/(3 099)	<b>41/(2 512)</b>	48/(2 490)
	30	5 000	28.994 0/(6.953 1)	<b>0.061 4/(0.090 7)</b>	0.067 9/(1.621 2)	0	<b>48/(3 221)</b>	43/(3 199)
$f_4$	20	4 000	0.015 8/(0.018 3)	0.017 2/(0.021 9)	<b>0.014 3/(0.010 6)</b>	17/(3 174)	29/(1 654)	<b>39/(1 539)</b>
	30	5 000	0.020 2/(0.033 7)	0.023 4/(0.049 3)	<b>0.019 1/(0.014 5)</b>	27/(3 178)	31/(2 364)	<b>41/(2 499)</b>
$f_6$	2	1 000	0.013 8/(0.002 8)	<b>0.012 1/(0.001 6)</b>	0.013 3/(0.001 8)	48/(323)	24/(697)	<b>49/(315)</b>

从表 2 平均函数值可以看出, 算法 HPSO-TVAC 和算法 CPSO-ADS 都表现出了优于算法 PSO-TVAC 的性能, 而且 CPSO-ADS 算法表现更为稳定。从各算法收敛到最优解的次数及 50 次实验中收敛到最优解的平均代数进行比较分析方

面看, CPSO-ADS 算法能在较少的迭代次数就收敛到最优解, 而且 50 次实验中成功收敛到最优解的概率明显占据优势。可见, CPSO-ADS 算法从全局寻

(下转第 193 页)



