文章编号: 1001-0920(2010)01-0020-05

一种高速收敛粒子群优化算法

朱海梅, 吴永萍
(扬州大学 物理科学与技术学院, 江苏 扬州 225002)

摘 要: 针对粒子群优化算法早熟问题,提出一种克服早熟的高速收敛粒子群算法. 该算法首先采用混沌序列初始化粒子位置,以增强搜索多样性;其次,在算法中嵌入有效判断早熟停滞的方法,一旦检索到早熟迹象,便随机地选择最优解任意一维的分量值,用一个随机值取代它,以扰乱粒子的当前搜索轨迹,使其跳出局部最优. 大量仿真实验表明,大多数连续函数的寻优过程只需用几个粒子、迭代几十次便能完成,可实现全局寻优过程的高速收敛.

关键词: 粒子群优化; 连续函数优化; 早熟; 高速收敛; 混沌序列
中图分类号: TP18 **文献标识码**: A

A PSO algorithm with high speed convergence

ZHU Hai-mei, WU Yong-ping
(Physics Science and Technology College, Yangzhou University, Yangzhou 225002, China. Correspondent: ZHU Hai-mei, E-mail: littlezhu80@hotmail.com)

Abstract: For the problem that particle swarm optimization (PSO) algorithm often suffers from being trapped in local optima so as to be premature convergence, an improved PSO with high-speed convergence is proposed to efficiently control premature stagnation. Firstly, chaotic sequence is used to initiate individual position, which strengthens the diversity of searching. Furthermore, an effective method that identifies premature stagnation is embedded to PSO, so once premature stagnation happens, a randomized solution, as a substitute for current optimum, is used to change the current searching locus so that particles can go out of the local optima. By using the two measures, the searching process can converge to the global optimum with high speed. Abundant simulation experiments demonstrate that the algorithm proposed in this paper only needs several particles and iterates a few times to be able to obtain the global optimum for most continuous function optimization problems. The convergence speed and searching ability are quite outstanding and satisfactory.

Key words: Particle swarm optimization (PSO); Continuous function optimization; Premature; High-speed convergence; Chaotic sequence

1 引 言

粒子群优化算法(PSO)[1]自 1995 年提出以来,已引起了有关研究者的广泛关注,并在连续优化领域取得了很多成果[2-15]. 然而,基本的 PSO 算法的性能很大程度上依赖于初始参数,并且经常容易陷于局部最优值而导致典型的早熟收敛问题[3]. 为了克服这一问题,相关学者已提出了很多改进算法,最有代表性的有以下两类:第 1 类是在标准 PSO 算法中引入惯性权重 w 并使其线性递减[4-6],从而使算法在开始阶段以大步长搜索. 随着搜索的深入,使搜索步长逐渐减小,从而使搜索更精细. 围绕对惯性权重 w 的改进有很多策略,诸如 w 线性递增[7]、非线性递减[8]、自适应变化[9,10]、按指数变化的策略等等[11]. 这类算法主要的改进在于不同搜索时期,其搜索的精细程度不同,从而可提高收敛速度和优化精度. 与标准 PSO 算法相比,能加快收敛速度,但无法克服算法早熟问题. 第 2 类代表性改进算法是混合算法[12-15],诸如全局搜索用 PSO 算法,用混沌搜索进行局部优化[12] 或用蚂蚁算法进行局部搜索[13] 等等. 这类算法吸收了不同算法的优点,两种算法取长补短,对改进算法性能具有明显的效果. 但是仍解决不了 PSO 算法本身固有的早熟等问题. 同时,作

收稿日期: 2009-02-20; **修回日期**: 2009-05-06.
基金项目: 国家自然科学基金项目(60673102).
作者简介: 朱海梅(1973—),女,江苏扬州人,讲师,硕士,从事智能控制、仿生算法的研究; 吴永萍(1972—),女,江苏扬州人,讲师,博士生,从事复杂网络的研究.

者也用这两类改进算法的代表成果对连续函数进行优化实验,结果表明:在许多连续函数的优化过程中,仍存在大量的无效迭代,即陷于局部最优值后难以自拔,往往经过几十次的迭代都不能跳出局部最优值,甚至一直陷入其中,从而大大影响算法的收敛速度和寻优精度. 作者认为:这些改进算法仍存在问题的根本原因在于,算法没有足够加强搜索的多样性,缺少跳出局部最优的机制.

为了进一步提高算法的收敛速度和寻优精度,针对上述改进算法存在的问题,本文提出了一种克服早熟停滞的高速收敛的改进粒子群算法. 首先,根据混沌序列具有随机性和遍历性的特征,采用混沌序列初始化各个粒子的位置,可使粒子较均匀地、不重复地遍布整个解空间,为全局搜索多样性奠定了坚实的基础;其次,在粒子群算法中嵌入能有效地判断早熟停滞的方法,一旦检索到早熟迹象,便用一个随机值取代当前最优解以便扰乱粒子的当前搜索轨迹,使其跳出局部最优. 这样,可大大减少无效迭代的次数,从而实现全局寻优过程的高速收敛. 大量的仿真实验表明:在大多数连续函数优化问题中,本文算法只需要几个粒子、迭代几十次便能实现全局寻优,收敛速度和寻优精度都优于国内外一些较新的改进粒子群算法,效果令人满意.

2　相关问题描述

2.1　求解问题描述

一个连续域全局优化问题,记 $C=(S,\Omega,f)$,其中搜索空间 S 为连续域变量 $X_i=(x_{i1},x_{i2},\cdots,x_{iD})$ 的有限集,$i=1,2,\cdots,m$;D 为空间维数. 有限空间的解为 $X_g=(x_{g1},x_{g2},\cdots,x_{gD})$. 为简便,以下将下标 i 省略. Ω为变量的约束集,若为空,则 C为无约束问题模型,否则为有约束问题. f 为优化函数,分别用 $f_{\min}$ 和 $f_{\max}$ 表示取函数最小值和最大值,本文以 $f_{\min}$ 为例,方法可推广到 $f_{\max}$. 根据以上描述,对于连续优化问题可表示为 $\min f(X)$,$X=(x_1,x_2,\cdots,x_D)$,简记为 $f_{\min}(X)$. s.t. $\min_i \le X_i \le \max_i$,$i=1,2,\cdots,D$. $[\min_i,\max_i]$ 为第 i 分量的取值区间. 记 $\forall P_j$,$j=1,2,\cdots,m$ 为任意一个粒子,m 为粒子总数,t 时刻,P_j 在解空间的位置为 $X_j(t)\in S$. 对于 $\forall X_j\in S_i$ 都有 $f(X_g)\le f(X_j)$,则 X_g 为全局最优解. 连续域空间函数优化问题的目标则是求得至少一个 X_g. 为了与粒子群习惯用的符号一致,以下记 X_g 为 P_g.

2.2　标准 PSO 算法问题简析

设第 i 个粒子在解空间 S 的位置为 $X_i=(x_{i1},x_{i2},\cdots,x_{iD})$,其中 D 表示问题的维度. 相应地,第 i 个粒子的速度可表示为 $V_i=(v_{i1},v_{i2},\cdots,v_{iD})$. 为了与一般位置相区别,第 i 个粒子的当前最优位置用 $P_i=(y_{i1},y_{i2},\cdots,y_{iD})$ 表示. 所有粒子到目前为止找到的全局最优位置可表示为 $P_g=(y_{g1},y_{g2},\cdots,y_{gD})$. 在找到个体极值和全局极值后,粒子根据下式:

$$\begin{cases} v_{i,d}(t+1)= \\ wv_{id}(t)+c_1r_1[y_{id}(t)-x_{id}(t)]+ \\ c_2r_2[y_{gd}(t)-x_{id}(t)], \\ x_{i,d}(t+1)=x_{id}(t)+v_{i,d}(t+1) \end{cases} \tag{1}$$

更新自己的速度和位置. 式中:$d\in[1,D]$;$0<w\le 1$ 称作惯性权重;c_1 和 c_2 被称作学习因子,通常 $c_1=c_2=2$;r_1,r_2 表示 $(0,1)$ 之间的随机数. 在算法运行过程中,粒子的个体极值和粒子群的全局极值都不断更新. 算法运行结束时,输出全局极值 P_g.

从式(1)不难看出,每个粒子不仅受它自己找到的当前最优解的吸引,而且还受全局最优解的吸引. 如果这两个最优解都是局部最优值,粒子将被这两个值吸引而很快重复相同的搜索轨迹. 由于式(1)没有使算法跳出局部最优的机制,搜索将由于粒子早熟而停滞. 另外,从式(1)还可看出,惯性权重仅能改变粒子的搜索步长,不能改变其运行方向,从而不能克服早熟问题. 可见,如何加强搜索多样性、如何使算法跳出局部最优是提高算法收敛速度和寻优精度的重要措施. 本文从这两个方面入手,提出了克服早熟停滞的高速 PSO 算法.

3　克服早熟停滞的高速 PSO 算法

3.1　粒子位置分布初始化

目前的粒子群优化算法的初始化大多采用随机分布的策略,难以保证初始粒子群有较好的遍历性. 考虑混沌序列具有混沌运动的特点,从而有较好的随机性和遍历性. 因此本文利用混沌序列这一特点进行粒子群的初始化分布,可大大加强算法的搜索多样性,为找到更优解和加快收敛奠定坚实的基础.

首先,利用如下 Logistic 映射产生混沌序列:

$$L(i+1)=\mu\cdot L(i)(1-L(i)). \tag{2}$$

其中:μ是一个常数$(\mu\in[3.56,4.0])$,用来控制系统处于混沌状态的程度;$L(i)\in(0,1)$,$i=1,2,\cdots,D$.

然后,对于 m 个粒子处于 D 维空间中,首先产生 m 个随机初值 $L_1(1),L_2(1),\cdots,L_k(1),\cdots,L_m(1)$. 把该混沌序列中的 m 个初值代入式(2)经过 D 次迭代运算,将产生 m 条运动轨迹. 从 m 条混沌运动轨迹中取 D 个迭代值,代入下式:

$$x_{k,i}=L_k(i)(\max_i-\min_i)k/m+\min_i,$$
$$k=1,2,\cdots,m;\ i=1,2,\cdots,D, \tag{3}$$

即可计算 $x_{k,i}$. 其中:$x_{k,i}$ 表示第 k 个粒子第 i 维的坐

标;$L_k(i)$ 为第 k 个粒子的随机初始值 $L_k(1)$ 运用式(2)经过 i 次迭代运算后的值;$\max_i$,$\min_i$ 分别为第 i 维的上下限.

利用式(3)计算得到所有 $x_{k,i}$ 组成 m 行 D 列的矩阵如下:

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \cdots & \cdots & \ddots & \cdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,D} \end{bmatrix}. \tag{4}$$

式中每个行向量都代表一个粒子的初始位置.

3.2 判断并克服早熟停滞的方法

在粒子群的每次迭代过程中,当得到的最优解在连续的 MAX 次迭代过程中都无变化时,便可认为算法有停滞的可能,表明粒子群根据现有的运动轨迹已经或者即将陷于局部最优解. MAX 的值可根据求解的问题规模预先指定,也可根据实验的结果选取适合特定函数优化的值. MAX 的取值越大,表明判断早熟停滞的标准越宽松. 这样,用一个停滞代数计数器记录到目前为止停滞的代数 SG,只要得到的最优解与上次相比不变时,就将其加 1,否则将其清 0. 当 SG 达到上限值 MAX 时,则说明算法可能停滞,即在 MAX 次的迭代中,粒子没有能力打破"僵局",跳出局部最优值. 此时,为改变粒子的运行轨迹,将当前最好解的任意一维修改成一个随机值,即

$$P_g = (y_{g1}, y_{g2}, y_{rk}, \cdots, y_{gD}). \tag{5}$$

式中:$y_{rk} \in [\min_k, \max_k]$ 和 $k \in [1, D]$ 都是随机产生的;P_g 是所有粒子群目前找到的全局最优位置.

该方法对目前的全局最优解进行微调,从而对式(1)中位置更新中的后一项增加了随机的扰动,亦即改变了粒子的位置更新策略,从而使得粒子的运行轨迹改变,赋予其打破"僵局"的能力,使其跳出局部最优解. 通过这种判断停滞和增加随机扰动的方法,可有效地减少无效迭代,从而大大提高算法的收敛速度和提高优化结果的精度.

3.3 克服早熟停滞的高速粒子群算法步骤

综合 3.1 和 3.2 节的基本原理,克服早熟停滞的高速收敛的粒子群算法步骤如下:

Step1: 设置最大迭代代数为 M,停滞判定代数 MAX,停滞代数计数器 SG = 0,迭代代数计数器 $j = 0$,$c_1 = c_2 = 2$,$r_1 = \text{rand}() \%100/100$,$r_2 = \text{rand}() \%100/100$,rand( ) 是一个随机函数,$r_1, r_2 \in [0,1]$;惯性权重 $w = w_0$,w_0 为其初值.

Step2: 采用 3.1 节所述的混沌初始化方法,对 m 个粒子的位置进行初始化 $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$,$i = 1,2,\cdots,m$;并随机初始化 V_i. 然后用适应值函数对每个粒子的初始位置进行评价,即计算出各粒子位置的适应值 f_i. 令 $P_i = X_i$,P_g 为最优的 f_i 对应的 X_i.

Step3: 根据式(1)更新 X_i 和 V_i,其中惯性权重 w 按线性递减计算,即 $w = 0.9 - 0.5 * j/M$,式中 j 表示第 j 次迭代,M 为算法迭代的总次数.

Step4: 重新用适应值函数对 X_i 进行评价,得到 f_i,$i = 1,2,\cdots,m$.

Step5: 对于 $\forall i < m$,如果 $f_i < f_{\text{best}\,i}$,则 $P_i = X_i$,$f_{\text{best}\,i}$ 表示第 i 个粒子到目前为止得到的最优函数值.

Step6: 令 X_q 是由 m 个粒子在本次迭代中找到的最优解位置,P_g 为到目前为止所有粒子得到的全局最优解位置,如果有 $f(X_q) < f(P_g)$,则令 $P_g = X_q$,SG = 0;否则,SG 加 1. 如果 SG = MAX,则采用 3.2 节所述的方法,随机产生一个维度 k,并随机产生一个随机值 $y_{rk} \in [\min_k, \max_k]$,用 y_{rk} 替换 P_g 中的 y_{gk},并令 SG = 0.

Step7: $j = j + 1$,如果 $j < M$,则返回 Step3;否则,迭代结束,输出 P_g 及对应的 $f(P_g)$ 即为最优解.

4 数值仿真

为了验证本文提出算法的效果,作者进行了大量的计算机数值实验,包括与一些新近改进的粒子群优化算法等的实验比较. 实验仿真软件 VC6.0. 为了与文献[12,16]比较,随机选用了其中的 7 个基准测试函数,这些测试函数见表 1.

4.1 对本文算法的实验测试

为了测试本文提出改进方法的效果,用表 1 所示的测试函数对本文算法进行测试,方法是在惯性权重线性递减的改进 PSO[6](简称 WPSO)的基础上,仅将粒子用混沌序列初始化,记录实验效果;再在 WPSO 基础上,仅进行停滞判别并进行随机扰动,记录实验结果;最后记录将两种改进措施都加入后的实验结果. 实验结果见表 2. 在实验中,取 $c_1 = c_2 = 2.0$,粒子群大小 m 取 10,采用线性递减的惯性权重 w,由初始的 0.9 减为最后的 0.4,在增加随机扰动时,MAX = 10. 表 2 中的寻优值是算法得到的最优解,评价次数是各算法在找到相应的最优解时需要的评价次数. 从表 2 可以看出,两种改进策略对改进算法收敛速度和寻优精度都有较好的效果. 一方面,用混沌序列初始化粒子能使粒子的初始位置遍布解空间,为粒子的搜索多样性奠定了基础,从而减少了寻优需要的进化代数;另一方面,算法增加了停滞判别,使算法能跳出局部最优,有效去除无效迭代,从而加速收敛. 此外,从时间复杂度分析,减少粒

表 1　基准测试函数

Symbol	Function(F)	Interval of x_i
GP	$f(X) = (1 + (x_1 + x_2 + 1)^2 \times (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)) \times (30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	$[-2,2]$
BR	$f_B(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$x_1:[-5,10]$ $x_2:[0,15]$
RA	$f_R(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$	$[-1,1]$
SH	$f_S(x) = \left\{\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right\}\left\{\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right\}$	$[-10,10]$
RS	$f_{RS} = \sum_{j=1}^{n-1}[100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2]$	$[-5,10]$
B2	$f_{B2}(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	$[-100,100]$
ES	$f_{ES}(x) = -\cos x_1\cos x_2\exp(-[(x_1 - \pi)^2 + (x_2 - \pi)^2])$	$[-100,100]$

表 2　两种改进措施及其综合实验结果

测试函数	WPSO		仅用混沌初始化		仅加随机扰动		综合改进		理论值
	寻优值	评价次数	寻优值	评价次数	寻优值	评价次数	寻优值	评价次数	
GP	4.62020	700	3.00000	56	3.00000	70	3.00000	46	3.00000
BR	0.49600	650	0.39791	87	0.39800	40	0.39800	31	0.39800
RA	-1.97020	1000	-2.00000	140	-2.00000	134	-2.00000	116	-2.00000
SH	-180.32650	480	-186.73020	200	-186.62820	209	-186.73090	138	-186.73090

表 3　固定的评价次数下本文算法与其他算法的优化结果比较

测试函数	本文算法	CPSO	WPSO	理论值
GP	3.00000	3.00000	4.62020	3.00000
BR	0.39800	0.39790	0.49600	0.39800
RA	-2.00000	-1.99400	-1.97020	-2.00000
SH	-186.73090	-186.72740	-180.32650	-186.73090

表 4　固定的精度要求下本文算法与其他算法的结果比较

测试函数	精度要求	平均评价次数		平均误差	
		本文算法	NM-PSO	本文算法	NM-PSO
GP	1e-3	168	217	2.4153e-10	0.00003
BR	1e-3	100	151	0.00002	0.00003
RS_2	1e-3	161	339	1.7740e-12	0.00003
SH	1e-2	132	400	3.6497e-6	0.00002
B2	1e-2	139	240	5.0000e-15	0.00003
ES	1e-3	135	165	6.4650e-12	0.00004

子数和进化次数可以显著减小时间复杂度,提高计算性能.

4.2　与其他算法的比较

为了展示本文算法的先进性,与一些有代表性的新近改进的 PSO 算法进行了实验比较. 首先,为了与 CPO 算法[12] 的结果进行比较,固定评价的总次数为 2000. 表 3 分别给出了 3 种算法的优化结果比较. 在 CPSO 算法与采用线性递减惯性权重因子的 WPSO 算法中, $c_1 = c_2 = 2.0$;粒子群大小 m 为 20,都采用线性递减的惯性权重 w,由初始的 1.2 减为最后的 0.2;而本文算法中 $c_1 = c_2 = 2.0$,粒子群大小 m 仅取 10,也采用线性递减的惯性权重 w,由初始的 0.9 减为最后的 0.4, MAX = 10. 表 3 的结果表明,本文算法的寻优结果最理想. 其次,为了与 NM-PSO 算法[16] 的结果进行比较,选择与其相同的精度要求,分别测试它们达到该精度所需的评价次数,并记录得到的最优值与理论值的差别,即误差. 分别运行 10 次,取平均值,结果示于表 4. 从表 4 看,本文算法达到较优精度时评价次数较少. 从寻优精度看,在较少的迭代评价次数下,本文算法的优化结果和理论值的误差最小,对表中的测试函数找到最优解的误差多数为 1e-10 以下,求解精度非常高.

为了更加直观地反映本文算法的收敛及优化性能,本文给出了各算法对相关测试函数的收敛曲线图,如图 1 ～ 图 4 所示. 图中的 CPSO 和 WPSO 收敛曲线由文献[12] 的数据得到. 由图可知,本文提出

的粒子群算法收敛曲线的下降速度比 CPSO 和 WPSO 要快得多,目标函数值也下降到更低的水平,因此可以得出结论:本文算法的优化能力要强于 CPSO 和 WPSO.

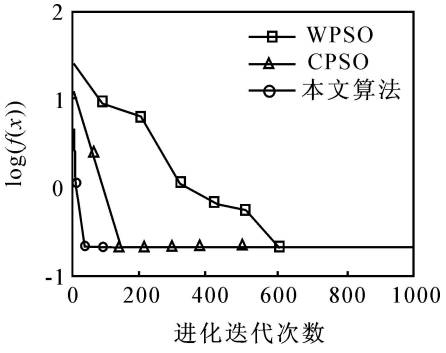

图 1 f_{GP} 函数的收敛性能比较

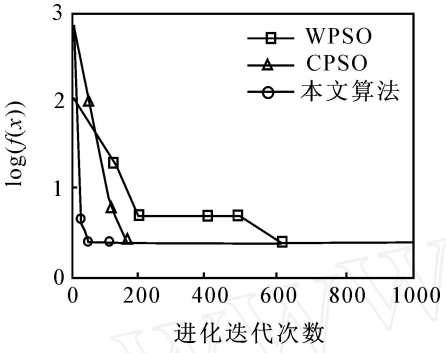

图 2 f_{BR} 函数的收敛性能比较

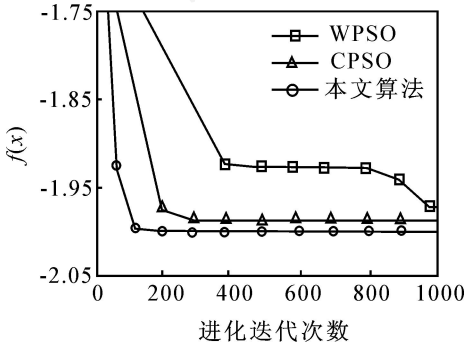

图 3 f_{RA} 函数的收敛性能比较

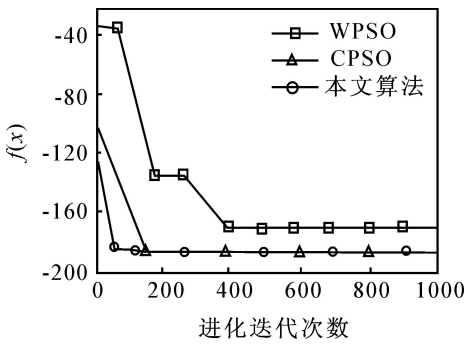

图 4 f_{SH} 函数的收敛性能比较

5 结 论

针对粒子群算法常见的早熟停滞问题,本文利用混沌运动具有较好的随机性和遍历性的特点,提出了采用混沌序列初始化粒子的位置,为粒子群的多样性搜索奠定了坚实的基础;同时在算法中增加对早熟停滞的判断,用随机值的扰动给早熟的局部最优解一个扰动,使得粒子的运行轨迹改变,跳出局部最优解,有效地减少无效迭代,从而大大提高算法的收敛速度和优化结果的精度. 仿真实验结果表明:本文算法只需要很少的粒子数,很少的迭代次数便能完全达到测试函数的理论优化结果,充分说明了本文算法的高速和精确寻优性能.

参考文献(References)

[1] Kennedy J, Eberhart R. Particle swarm optimization [C]. Proc of IEEE Int Conf on Piscataway, 1995: 1942-1948.

[2] Eberhart R C, Shi Y. Particle swarm optimization[C]. Proc of Congress on Evolutionary Computation. Seoul, 2001: 81-88.

[3] Angeline PJ. Evolutionary optimization versus particle swarm optimization [C]. Evolutionary Programming VII. London: Springer, 1998: 601-610.

[4] Shi Y, Eberhart R. Parameter selection in particle swarm optimization [C]. Proc of 7th Annual Conf on Evolution Computation. Berlin, 1998: 591-601.

[5] Shi Y, Eberhart R. Empirical study of particle swarm optimization [C]. Proc of the 1999 Congress on Evolution Computation. Berlin, 1999: 1945-1950.

[6] Kennedy J, Eberhart R, Shi Y. Swarm Intelligence [M]. San Francisco: Morgan Kaufmann Publishers, 2001.

[7] Zheng Y L, Ma L H. Convergence analysis and parameter selection in particle swarm optimization[C]. Proc of the Second Int Conf on Machine Learning and Cybernetics. Xi'an, 2003: 1802-1807.

[8] Lei W, Zhao C X. The research of PSO algorithms with non-linear time-decreasing inertia[C]. Proc of the Int Conf on Intelligent Control and Automation. Chongqing, 2008: 4002-4005.

[9] Chen D, Wang G F, Chen Z Y. The inertia weight self-adapting in PSO[C]. Proc of the 7th World Congress on Intelligent Control and Automation. Chongqing, 2008: 5313-5316.

[10] Feng C S, Cong S, Feng X Y. A new adaptive inertia weight strategy in particle swarm optimization [C]. Proc of the IEEE Congress on Evolutionary Computation. Singapore, 2007: 4186-4190.

[11] Chen G M, Huang X B, Jia J Y, et al. Natural exponential inertia weight strategy in particle swarm optimization[C]. Proc of the 6th World Congress on Intelligent Control and Automation. Dalian, 2006: 3672-3675.

(下转第 30 页)

化网络运行成本,最大程度地弥补节点失效造成的损失,为供应链管理中的应急管理研究提供了一条有效的途径.

参考文献(References)

[1] 刘希龙, 季建华. 基于应急供应链的弹性供应网络设计研究[J]. 控制与决策, 2002, 22(11): 1224-1227.
(Liu X L, Ji J H. Resilient supply network design based on contingency supply[J]. Control and Decision, 2002, 22(11): 1224-1227.)

[2] 肖为群. 基于资源外取的供应链弹性研究[J]. 物流技术, 2008, 27(1): 77-80.
(Xiao W Q. Research on SC flexibility based on outsourcing[J]. Logistics Technology, 2008, 27(1): 77-80.)

[3] 李怡娜, 徐学军. 弹性需求下可控提前期供应链库存优化的信用期机制[J]. 运筹与管理, 2008, 17(2): 32-37.
(Li Y N, Xu X J. Research on credit period mechanism for optimizing supply ohain inventory with controllable lead time and elastic demand[J]. Operations Research and Management Science, 2008, 17(2): 32-37.)

[4] Christopher. Creating resilient supply chain[J]. Int J of Operations of Production Management, 2003, 24(6): 589-601.

[5] Rice J B, Caniato F. Building a secure and resilient supply network[J]. Supply Chain Management Review, 2003, 7(5): 22-30.

[6] Haywood M, Peck H. Improving the management of supply china vulnerability in UK aerospace manufacturing[C]. Proc of the 1st EUROMA/ POMs Conf. London, 2003: 121-130.

[7] Muckstadt J A, Murray D H. Guidelines for collaborative supply chain system design and operation [J]. Information Systems Frontiers: Special Issue on Supply Chain Management. London, 2007, 3(4): 427-453.

[8] Kishalay Mitra, Ravindra D. Towards resilient supply chains: Uncertainty analysis using fuzzy mathematical programming[J]. Chemical Engineering Research and Design, 2009, 15(3): 172-184.

[9] 赵林度. 基于细胞弹性模型的供应链弹性研究[J]. 物流技术, 2009, 28(1): 101-104.
(Zhao L D. Analysis on supply chain resilience based on cell resilience model[J]. Logistics Technology, 2009, 28(1): 101-104.)

[10] Tomlin Brian T. On the value of mitigation and contingency strategies for managing supply chain disruption risks[J]. Management Science, 2006, 52(5): 639-657.

[11] Qi X T, Bard J, Yu G. Supply chain coordination with demand disruptions[J]. Omega, 2004, 32(4): 301-312.

[12] Christian Kuhnert, Dirk Helbing. Scaling laws in urban supply networks[J]. Physica A, 2006, 363(1): 89-95.

[13] Marco Laumanns, Erjen Lefeber. Robust optimal control of material flows in demand-driven supply networks[J]. Physica A, 2006, 363(1): 24-31.

(上接第 24 页)

[12] Liu B, Wang L, Jin Y H, et al. Improved particle swarm optimization combined with chaos[J]. Chaos Solitons & Fractals, 2005, 25: 1261-1271.

[13] Shelokar P S, Jayaraman V K, Kulkarni B D. Particle swarm and ant colony algorithms hybridized for improved continuous optimization [J]. Applied Mathematics and Computation, 2007, 188: 129-142.

[14] Esmin A A A, Lambert-Torres G, Alvarenga G B. Hybrid evolutionary algorithm based on PSO and GA mutation[C]. Proc of the Sixth Int Conf on Hybrid Intelligent Systems. Auckland, 2006: 57-59.

[15] Zhang T, Zhang C M, Zhang Y J, et al. A mixed PSO algorithm for the VRPSPD[C]. 2008 Chinese Control and Decision Conf. Yantai, 2008: 4017-4021.

[16] Fan S K S, Liang Y C, Zahara E. Hybrid simplex search and particle swarm optimization for the weight [C]. Proc of the 7th World Congress on Engineering Optimization. Novosibirsk, 2004: 401-418.