文章编号: 1001-0920(2008)11-1238-05

基于拥挤度与变异的动态微粒群多目标优化算法

王 辉, 钱 锋
(华东理工大学 a. 化学工程联合国家重点实验室, b. 信息科学与工程学院, 上海 200237)

摘 要: 提出一种动态微粒群多目标优化算法(DCMOPSO),算法中的惯性权重和加速因子动态变化以增强算法的全局搜索能力,并采用拥挤度的方法对外部档案进行维护以增加非劣解的多样性. 在维护过程中,从外部档案中按拥挤度为每个微粒选择全局最好位置,同时使用变异操作避免算法早熟. 通过几个典型的多目标测试函数对DCMOPSO 算法的性能进行了测试,并与多目标优化算法 MOPSO 和 NSGA-Ⅱ进行对比. 结果表明, DCMOPSO 算法具有良好的搜索性能.

关键词: 微粒群优化; 多目标优化; 动态变化; 拥挤度; 变异操作
中图分类号: TP18　　**文献标识码**: A

Improved PSO-based multi-objective optimization by crowding with mutation and particle swarm optimization dynamic changing

WANG Hui, QIAN Feng
(a. State Key Laboratory of Chemical Engineering, b. School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China. Correspondent: WANG Hui, E-mail: zgwangh@163.com)

Abstract: A dynamic changing multi-objection particle swarm optimization (DCMOPSO) algorithm is proposed, in which inertia weight and acceleration coefficients are dynamic changing to explore the search space more efficiently. Crowding distance is used to maintain the external archive and mutation operator mechanism is also adopted to maintain the diversity of non-dominated solutions. The global best location for every particle is selected in the procedure of external archive maintenance. Some benchmark functions are tested for comparing the performance of DCMOPSO with MOPSO and NSGA-Ⅱ. The results show the feasibility of DCMOPSO.

Key words: Particle swarm optimization; Multi-objective optimization; Dynamic changing; Crowding; Mutation

1 引　　言

现实生活中有许多优化问题包含多个优化目标,并且各个目标需同时优化,但多个目标往往存在冲突,难于优化. 传统的多目标优化方法是,通过加权方法将多目标问题转化为单目标优化问题,但许多问题需要先验知识,因此该方法对许多多目标问题难以有效处理. 基于种群的进化算法具有隐含的并行特征,可在单轮模拟过程中得到多个解,因此非常适合求解多目标优化问题. Schaffer 于 1985 年首次用进化算法求解多目标优化问题(VEGA). 近几年来,又有许多进化算法解决多目标优化问题,主要有非劣分类遗传算法(NSGA),改进型非劣分类遗传算法(NSGA-Ⅱ),Pareto 存档进化策略(PAES),强度 Pareto 进化算法(SPEA),改进强度进化算法(SPEA2)以及蚁群多目标优化算法[1]等.

微粒群算法(PSO)由 Kennedy 与 Eberhart[2]于 1995 年提出,是一种模拟群体行为的智能优化方法,具有快速收敛、易实现的特点,在许多优化问题中得到成功应用[3]. 随着对微粒群算法研究的深入,微粒群算法由解决单目标优化问题,逐渐拓展到解决多目标优化问题.

在微粒群多目标优化算法中, Coello 与

收稿日期: 2007-08-29; **修回日期**: 2007-12-06.
基金项目: 国家杰出青年科学基金项目(60625302); 国家 973 计划项目(2002CB3122000); 国家 863 计划项目(20060104Z1081); 国家自然科学基金项目(60704028); 国家科技支撑计划项目(2007BAF22B05); 上海市科委重大基础研究项目(05DJ14002); 上海市基础研究重点项目(07JC14016).
作者简介: 王辉(1972—),男,河南开封人,博士生,从事计算智能、智能控制理论与应用的研究; 钱锋(1961—),男,江苏扬中人,教授,博士生导师,从事工业过程先进控制、优化与故障诊断等研究.

Lechunga[4]将微粒群搜索过程中得到的全局最优解存储到一个档案,而个体的局部最优存储到另外一个档案. Hu 与 Eberhart[5]提出,通过搜索过程中计算不同微粒之间的距离选择本地最优个体,并对每个目标函数进行逐次优化,但该算法对目标函数的求解顺序较为敏感. Fieldsend 与 Singh[6]采用精英归档方法将搜索过程中得到的非劣解存储到精英档案中,并在速度公式中加入扰动项影响微粒的飞行行为、提高解的多样性. 由于增加的扰动项具有随机性,使算法对于不同的多目标优化问题难以得到理想的效果. Coello 与 Pulido[7,8]提出了采用变异机制保持非劣解多样性的多目标微粒群算法等.

为了提高微粒群多目标算法的收敛性,增加非劣解的多样性,本文提出一种动态微粒群多目标优化算法(DCMOPSO). 在该算法中,通过惯性权重与加速因子动态变化提高微粒群的搜索性能,采用拥挤度与变异操作增加非劣解的多样性,并用约束处理机制解决约束优化问题.

2　多目标优化问题

定义 1　一般地,含有 n 个变量 s 个目标的多目标优化问题可表示如下:

$$\min\ y = f(x) = (f_1(x), f_2(x), \cdots, f_n(x)).$$
$$\text{s.t. } g_i(x) \leq 0.$$

其中

$$x = (x_1, x_2, \cdots, x_n) \in X,$$
$$y = (y_1, y_2, \cdots, y_s) \in Y.$$

这里:决策向量 $x \in R^n$,目标向量 $y \in R^s$,$f_i(x)\ (i = 1,2,\cdots,n)$ 是目标函数,$g_i(x) \leq 0\ (i = 1,2,\cdots,k)$ 是约束条件.

定义 2　非劣最优解(Pareto 最优解):若 x^* 是搜索空间中的一点,则 x^* 为非劣最优解,当且仅当不存在另外一点使 $f(x) < f(x^*)$ 且 $f_i(x) < f_i(x^*)$ 成立.

定义 3　Pareto 支配向量和非支配向量对于目标函数的一个矢量 $f(x^*)$,若其为非支配的,当且仅当不存在另外一个矢量 $f(x)$ 使 $f(x) \leq f(x^*)$ 且至少一个 $f_i(x) < f_i(x^*)$ 成立,则称 $f(x^*)$ 为非支配的;否则为支配的.

定义 4　Pareto 前端 X^* 为 A 中非支配解的集合,即 $X^* = \{x_1^*, x_2^*, \cdots, x_n^*\}$,则集合 X^* 为 Pareto 最优集,其中所有非支配解组成 Pareto 前端.

3　基于拥挤度与变异操作的动态微粒群多目标优化算法

因为 Pareto 最优集的元素有无穷个,所以有限规模的微粒群不可能获得整个 Pareto 最优集,而获得一个能代表整个 Pareto 最优集的有限子集是可行的,这也是 DCMOPSO 算法求解的目的.

DCMOPSO 算法采用外部档案存储种群产生的非劣解,通过非劣解拥挤度的大小对外部档案进行维护获得更多分布均匀的非劣解,使其能更大程度地逼近整个 Pareto 前端.

在微粒群进化过程中,DCMOPSO 算法从外部档案中拥挤度大的部分非劣解中随机选择个体,作为微粒个体全局最好位置. 当外部档案已满时,通过微粒群产生的新非劣解对拥挤度小的非劣解进行替代.

3.1　惯性权重与加速因子动态变化

微粒群算法中,惯性因子表明微粒的历史速度信息对当前速度的影响. 当其较大时可加大微粒群的搜索空间,提高搜索的全局性能;较小时微粒的局部搜索性能得到改善.

在微粒群多目标优化中,惯性因子通常设置为常数(如 0.4,0.5)或取一定范围内的随机值(如 0.5 + Rand(0,1)/2). 固定设置的惯性因子往往针对特定的优化问题有效,在优化函数变化时,需对其值大小不断调整. 为了适应不同的优化问题,使微粒更有效地搜索全局最优,DCMOPSO 惯性权重随着迭代的进行动态下降,有

$$w(t) = (w_f - w_i) \times \frac{(\mathrm{ITERMAX} - t)}{\mathrm{ITERMAX}} + w_i. \quad (1)$$

其中:t 为当前迭代次数,ITERMAX 为最大迭代次数. 惯性权重初始值 w_i 较大,有利于在较大的空间内搜索. 随着搜索的进行,算法按线性规律下降到较小的终值 w_f,并对目标进行细致的搜索.

PSO 算法中,加速因子用来平衡个体和群体认知能力,通常取常数. 在 DCMOPSO 中,为了更好地平衡微粒的自身认知能力和全局认知能力,加速因子随进化的进行动态变化[9],表达式为

$$c_1(t) = (c_{1f} - c_{1i}) \times \frac{(\mathrm{ITERMAX} - t)}{\mathrm{ITERMAX}} + c_{1i}, \quad (2)$$

$$c_2(t) = (c_{2f} - c_{2i}) \times \frac{(\mathrm{ITERMAX} - t)}{\mathrm{ITERMAX}} + c_{2i}. \quad (3)$$

其中:c_{1i} 和 c_{2i} 分别为 c_1 和 c_2 的初值;相应地,c_{1f} 和 c_{2f} 分别为 c_1 和 c_2 的终值. 开始时认知部分所占比重较大,微粒在搜索空间内进行大范围地搜索. 随着搜索的进行,社会认知部分占主导地位,微粒逐渐向全局最优搜索.

3.2　拥挤度计算与变异操作

在 DCMOPSO 算法中,当外部档案中非劣解的数目没有达到规定规模时,将产生的非劣解直接存储到外部档案;当外部档案容量满时,通过度量个体

拥挤度的方法删减外部档案中的解[10],并采用变异操作来避免算法早熟[7].

3.2.1 拥挤度计算

外部档案中各个解间的疏密程度可通过解的拥挤度即个体的拥挤距离来判断. 图 1 为计算点 i 拥挤距离的示意图,点 i 所在立方体为不包含其他点所在的最大立方体. 点 i 的拥挤距离为所在立方体对应不同目标函数相邻点的边长(即相邻非劣解适应值之差)之和. 拥挤距离越小表明解分布的越密集,解的多样性越小;反之,表明解分布的越稀疏,解的多样性越大. 在外部归档对非劣解删减时,拥挤距离大的解被保留,拥挤距离小的解则被删除. 这样,外部档案中的解就具有多样性,从而获得均匀的 Pareto 前端.

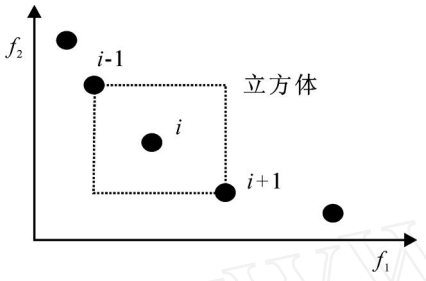

图 1 拥挤距离计算

外部档案 A 中非劣解的拥挤距离计算流程如下:

Step1: 设档案 A 包含 l 个非劣解,$1 \le i \le l$.

Step2: 初始化 A 中每个非劣解的拥挤距离,Crowddist(i) = 0.

Step3: For f = 1 to s

(f 为目标函数,假设共有 s 个目标函数)

A = sortFit(A, f);

(对档案 A 中的非劣解按对应目标向量的适应值进行排序)

max = 2;

For i = 2 to l - 1

Crowddist(i) = Crowddist(i) + (archiveFit$(i+1).f$ - archiveFit$(i-1).f$);

(archiveFit$(i).f$ 表示档案 A 中第 i 个个体对第 f 个目标函数的适应值)

if Crowddist(max) < Crowddist(i) (Crowddist(max) 为最大拥挤距离)

max = i;

End

End for Crowddist(1) = Crowddist(1) + Crowddist(max)

Crowddist(l) = Crowddist(l) + Crowddist(max)

(档案 A 中处于边界的非劣解的拥挤距离为最大,总是被选中)

End for.

3.2.2 变异操作

微粒群算法具有快速收敛的特点,但在多目标算法中,过快的收敛速度反而可能导致算法早熟、陷入局部最优,收敛到错误的 Pareto 前端. 因此,DCMOPSO 在算法初始阶段采用变异操作,使所有的微粒在整个搜索空间进行搜索. 随着搜索的进行,进行变异的微粒数将逐渐下降,搜索到一定阶段后,算法停止变异操作[7],从而避免算法早熟.

3.3 DCMOPSO 算法描述

Step1: 算法参数设定:设置种群规模 m,变异概率为 MUT,算法搜索的最大代数 ITERMAX,令进化代数 t = 1.

Step2: 随机产生初始种群 P,计算 P 中每个粒子所对应的目标向量,将非劣解存储到外部档案 A 中.

Step3: 计算档案 A 中每个非劣解的拥挤度,将 A 中的非劣解按拥挤度从大到小排列.

Step4: For i = 1 to m

1) 从外部档案 A 中拥挤度大的部分非劣解中随机选择个体,将其位置设置为 GBEST.

2) 微粒速度更新:

$$v(i) = w(t) \times v(i) + c_1(t) R_1 (\mathrm{PBEST}(i) - P(i)) + c_2(t) R_2 (A(\mathrm{GBEST}) - P(i)).$$

其中:$1 \le i \le m$;PBEST 为个体最优位置;$w(t)$,$c_1(t)$ 与 $c_2(t)$ 分别采用式(1) ~ (3) 计算得到.

3) 微粒位置更新:

$$P(i) = P(i) + v(i).$$

4) 若 t < ITERMAX * MUT,则对 $P(i)$ 进行变异操作.

5) 计算 P 中每个粒子所对应的目标向量.

End for.

Step5: 若 P 新产生的非劣解不受外部档案 A 中解的支配,则将其插入到 A 中,删除 A 中所有受新非劣解支配的解. 若档案 A 已满,则用新的非劣解替代随机从档案 A 中拥挤度小的部分非劣解的某一个体.

Step6: 当 $P(i)$ 在微粒历史位置中占优时,更新 P 中个体最优,即 PBEST(i) = $P(i)$.

Step7: $t = t + 1$, 若 $t <$ ITERMAX, 则转 Step3; 否则, 算法结束.

3.4 约束处理机制

针对多目标优化中的约束问题, DCMOPSO 算法采用 NSGA-Ⅱ算法的约束处理机制[10] 对两个解进行比较: 1) 若两个解中一个为可行解, 另外一个为非可行解, 则选可行解; 2) 若两个解均为可行解, 选非支配解; 3) 若两个都是非可行解, 则计算它们违反约束条件的次数, 选违反约束条件最少的解.

4 实验结果与讨论

DCMOPSO 算法与文献[7] 中提出的多目标微粒群算法 MOPSO 及 NSGA-Ⅱ算法进行对比研究. 3 种算法的群体规模均设为 100, 外部档案规模为 100, 迭代 500 次. 对于 NSGA-Ⅱ算法采用实数值编码, 交叉概率为 0.8, 采用锦标赛选择机制, 变异率为 $1/n$(n 为优化多目标问题的变量数目); DCMOPSO 与 MOPSO 的变异概率为 0.5; 对于 MOPSO 自适应网格单元数目设置为 30. 在 DCMOPSO 中, 随机选择外部档案中拥挤度最大的 10 % 非劣解中的某一个作为微粒个体的全局最优位置; 当外部档案 A 已满, 需对档案中的解删减时, 从拥挤度最小的 10 % 的非劣解中随机选择个体用新产生的非劣解进行替换. MOPSO 的惯性权重设置为 0.4, 加速因子均设置为 2.0; DCMOPSO 的惯性因子与加速因子设置为: $w_i = 0.4$, $w_f = 0.9$, $c_{1i} = 2$, $c_{1f} = 1$, $c_{2i} = 1$, $c_{2f} = 2$. 对每个测试函数, 各个算法分别进行 30 次独立实验.

4.1 测试函数

第 1 个测试函数由 Deb 提出, 其中 $g(x_2)$ 有许多全局和局部最优值, 即

$$\min f_1(x_1, x_2) = x_1,$$
$$\min f_2(x) = g(x_2)/x_1.$$

其中

$$g(x_2) = 2.0 - \exp\left\{-\left(\frac{x_2 - 0.2}{0.004}\right)^2\right\} - 0.8 * \exp\left\{-\left(\frac{x_2 - 0.6}{0.4}\right)^2\right\},$$
$$0.1 \leq x_1, x_2 \leq 1.0.$$

第 2 个测试函数由 Kita 提出, 有 3 个约束条件, 即

$$\max f_1(x_1, x_2) = -x_1^2 + x_2,$$
$$\max f_2(x_1, x_2) = \frac{1}{2}x_1 + x_2 + 1,$$
$$\text{s.t. } 0 \geq \frac{1}{6}x_1 + x_2 - \frac{13}{2},$$
$$0 \geq \frac{1}{2}x_1 + x_2 - \frac{15}{2},$$
$$0 \geq 5x_1 + x_2 - 30,$$

其中 $x_1, x_2 \geq 0$.

第 3 个测试函数由 Kursawe 提出, 有 3 个离散曲线组成, 即

$$\min F(f_1(x), f_2(x)).$$

其中

$$f_1(x) = \sum_{i=1}^{n}\left(-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right),$$
$$f_2(x) = \sum_{i=1}^{n}\left(|x_i|^{0.8} + 5\sin(x_i)^3\right),$$

这里 $-5 \leq x_1, x_2, x_3 \leq 5$.

4.2 算法性能评价标准

多目标优化算法通常通过算法的收敛性与非劣解的多样性衡量算法的性能, 本文采用以下两个标准对算法性能进行对比.

1) 收敛性. 算法的收敛性可通过算法所得的非劣解与 Pareto 最优解之间的距离 GD 衡量[11], 有

$$GD = \frac{1}{n}\sqrt{\sum_{i=1}^{n} d_i^2}.$$

其中: n 为算法所得非劣解的个数, d_i 为第 i 个解到 Pareto 最优解集的最小距离. 若 GD 为 0, 则表示所得非劣解均属于 Pareto 最优解集; 若 GD 为其他值, 则表示所求非劣解与 Pareto 最优值有所偏离. 因此, 该指标反映了算法所得的优化解集与 Pareto 最优解集的逼近程度, GD 值越小, 逼近程度越好.

2) 分散性. 算法所求非劣解的多样性可通过非劣解的分布情况衡量, 即通过非劣解的间隔距离 SP[12] 度量, 即

$$SP = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{D} - d_i)^2},$$

其中

$$d_i = \min\left(\left|f_1^i(\vec{x}) - f_1^j(\vec{x})\right| + \left|f_2^i(\vec{x}) - f_2^j(\vec{x})\right|\right).$$

$i, j = 1, 2, \cdots, n$; $\bar{D}$ 是所有 d_i 均值; n 是迄今为止发现的非劣解的数目. 若 SP = 0, 则表示非劣解前端上所有解呈均匀分布, 因此 SP 值越小, 非劣解前端分布越均匀.

4.3 DCMOPSO 算法复杂度分析

DCMOPSO 算法的计算复杂度主要由目标函数计算、拥挤距离计算和群体之间非劣解比较以及档案非劣解之间的比较组成. 对于含有 M 个目标函数 N 个解的计算而言, 计算目标函数的复杂度为 $O(MN)$; 拥挤距离计算的复杂度在于每个要优化的目标函数的排序问题. 若有 K 个解在档案中进行排序, 则拥挤距离的计算复杂度为 $O(MK\log K)$; 若档

案大小与微粒群体规模一样均为 N,则算法的计算复杂度为 $O(MN^2)$. 因此,DCMOPSO的计算复杂度为 $O(MN^2)$.

4.4 测试结果与讨论

图 2 ~ 图 4 为 DCMOPSO,MOPSO 与 NSGA-Ⅱ对 3 个测试函数所生成的 Pareto 前端与理论 Pareto 前端的对比图. 表 1 与表 2 分别为 3 种算法关于 3 个测试函数的 GD 值与间隔度量 SP 值.

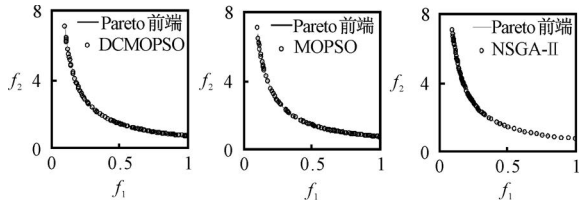

图 2 3 种算法关于 Deb 函数的 Pareto 曲线

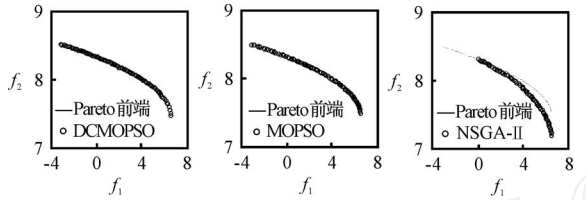

图 3 3 种算法关于 Kita 函数的 Pareto 曲线

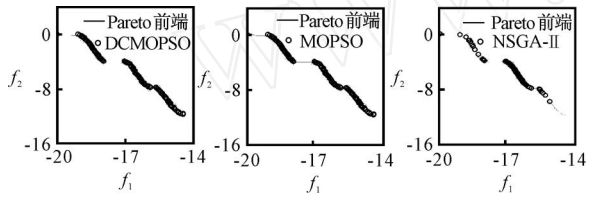

图 4 3 种算法关于 Kursawe 函数的 Pareto 函数

由图 2 可以看出,对于 Deb 函数,3 种算法中 DCMOPSO 与 MOPSO 的解都能完全覆盖 Pareto 前端,NSGA-Ⅱ的解却不能完全覆盖 Pareto 前端. 由表 1 与表 2 可以看出,MOPSO 具有较好的收敛性(GD 值较小),但 MOPSO 的解分布性较差;相反,NSGA-Ⅱ的解分布较均匀,但其收敛性较差. 与 MOPSO 及 NSGA-Ⅱ相比,DCMOPSO 具有较好的收敛性,并且其解的分布也更均匀.

表 1 3 个测试函数的 GD 值

测试函数	GD	DCMOPSO	MOPSO	NSGA-Ⅱ
Deb	平均值	0.03016	0.03201	0.04568
	中值	0.00042	0.00052	0.00087
	标准差	0.05263	0.05982	0.07962
Kita	平均值	0.03008	0.03528	0.08567
	中值	0.00052	0.00782	0.01264
	标准差	0.09245	0.10458	0.17924
Kursawe	平均值	0.00535	0.00845	0.02962
	中值	0.00521	0.00838	0.01868
	标准差	0.00048	0.00051	0.02812

表 2 3 个测试函数的间隔度量 SP 值

测试函数	SP	DCMOPSO	MOPSO	NSGA-Ⅱ
Deb	平均值	0.034271	0.08361	0.03745
	中值	0.029245	0.05522	0.03556
	标准差	0.008305	0.11821	0.00924
Kita	平均值	0.058730	0.10945	0.09849
	中值	0.037536	0.06756	0.02717
	标准差	0.067795	0.10143	0.32738
Kursawe	平均值	0.032877	0.09462	0.03614
	中值	0.030244	0.10396	0.03607
	标准差	0.009025	0.01575	0.01091

对于 Kita 函数,NSGA-Ⅱ的 SP 值较小,表明其解分布较均匀,但 GD 值较大. 由图 3 可以看出,NSGA-Ⅱ的解不能完全覆盖 Pareto 前端,且偏离 Pareto 前端较远,因此 NSGA-Ⅱ的解较差. MOPSO 能完全覆盖理论的 Pareto 前端,并且 GD 值较小、算法具有较好的收敛性,但其 SP 值较大,表明解分布的均匀性较差. DCMOPSO 所获得的解能完全覆盖 Pareto 前端,其 GD 值与 SP 值都较小,因此 DCMOPSO 算法解的分布性、算法的收敛性都更好.

对于 Kursawe 函数,通过表 1,表 2 及图 4 可知,NSGA-Ⅱ的 SP 值也较小,解的分布较均匀,但其 GD 值较大,说明其解偏离理论的 Pareto 前端较远,收敛性差,NSGA-Ⅱ所求解不能完全覆盖 Pareto 前端. MOPSO 的解能完全覆盖 Pareto 前端,且偏离理论 Pareto 前端较小,但其解的分布性较差. DCMOPSO 能完全覆盖 Pareto 前端,其收敛性与解的分布性最好.

通过对 3 个典型函数的优化实验可知,与 MOPSO,NSGA-Ⅱ相比,DCMOPSO 能更好地收敛到非劣解前端,且所得非劣解的分布更均匀.

5 结 论

本文提出一种动态多目标微粒群算法 DCMOPSO. 在该算法中,惯性权重与加速因子动态变化能更好地平衡算法的全局和局部搜索能力,使算法在多目标优化问题的空间进行更有效地搜索,同时算法对不同优化问题的适应能力得到提高. DCMOPSO 算法采用拥挤度机制对外部档案进行维护,增加了非劣解的多样性,在维护过程中从外部档案中按拥挤度为每个微粒选择全局最好位置. DCMOPSO 算法中的变异操作机制避免了算法早熟. 将其应用于 3 个常用的多目标测试函数,并与多目标优化算法 MOPSO,NSGA-Ⅱ进行了对比和分析,结果表明,DCMOPSO 算法具有良好的性能.

(下转第 1248 页)

method for hiding data in VQ encoded images[C]. Proc of 2004 Int Symposium on Intelligent Multimedia, Video and Speech Processing. Hong Kong: IEEE, 2004: 358-361.

[9] Du W C, Hsu W J. Adaptive data hiding based on VQ compressed images[J]. IEE Proc Visual Image Signal Processing, 2003, 150(4): 233-238.

[10] Chang C C, Lin P Y. A compression-based data hiding scheme using vector quantization and principle component analysis[C]. Proc of 2004 Int Conf on Cyberworlds. Tokyo: IEEE, 2004: 369-375.

[11] Chang C C, Chen G M, Lin M H. Information hiding based on search-order coding for VQ indices[J]. Pattern Recognition Letters, 2004, 25(11): 1253-1261.

[12] Chang C C, Tai W L, Lin M H. A reversible data hiding scheme with modified side match vector quantization[C]. Proc of the 19th Int Conf on Advanced Information Networking and Application. Washington: IEEE, 2005: 947-952.

[13] Chang C C, Lu T C. Reversible index-domain information hiding scheme based on side-match vector quantization[J]. J of Systems and Software, 2006, 79(8): 1120-1129.

[14] Cachin C. An information-theoretic model for steganography[J]. Information and Computation, 2004, 192(1): 41-56.

[15] Provos N. Defending against statistical steganalysis[C]. Proc of the 10th USENIX Security Symposium. Washington: USENIX, 2001: 323-336.

[16] Solanki K, Sullivan K, Madhow U, et al. Statistical restoration for robust and secure steganography[C]. Proc of IEEE Int Conf on Image Processing. Genova: IEEE, 2005: 1118-1121.

[17] Eggers J J, Bauml R, Girod B. A communications approach to image steganography[C]. Proc of SPIE Security and Watermarking of Multimedia Contents IV. San Jose: SPIE, 2002, 4675: 26-37.

[18] Sallee P. Model-based steganography[C]. Lecture Notes in Computer Science: IWDW2003. Seoul: Springer, 2004, 2939: 154-167.

[19] Zhang X P, Wang S Z, Zhang K W. Steganography with least histogram abnormality[C]. Lecture Notes in Computer Science: MMM-ACNS2003. Petersburg: Springer, 2003, 2776: 395-406.

(上接第 1242 页)

参考文献(References)

[1] 张勇德, 黄莎白. 多目标优化问题的蚁群算法研究[J]. 控制与决策, 2005, 20(2): 170-173.
(Zhang Y D, Huang S B. On ant colony algorithm for solving multiobjective optimization problems[J]. Control and Decision, 2005, 20(2): 170-173.)

[2] Kennedy J, Eberhart R C. Particle swarm optimization[C]. Proc IEEE Int Conf on Neural Networks. Piscataway: IEEE Service Center, 1995: 1942-1948.

[3] 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.
(Xie X F, Zhang W J, Yang Z L. Overview of particle swarm optimization[J]. Control and Decision, 2003, 18(2): 129-134.)

[4] Coello C A C, Lechuga M S. MOPSO: A proposal for multiple objective particle swarm optimization[C]. Proc IEEE Int Conf on Evolutionary Computation. Piscataway: IEEE Service Center, 2002, 2: 1051-1056.

[5] Hu X, Eberhart R. Multiobjective optimization using dynamic neighborhood particle swarm optimization[C]. Proc IEEE Int Conf on Evolutionary Computation. Honolulu, 2002, 2: 1677-1681.

[6] Fieldsend J E, Singh S. A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence[C]. Proc UK Workshop on Computational Intelligence. Birmingham, 2002: 37-44.

[7] Coello C A C, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 256-279.

[8] Reyes-Sierra M, Coello C A C. Multi-objective particle swarm optimizers: A survey of the state-of-the-Art[J]. Int J of Computational Intelligence Research, 2006, 2(3): 287-308.

[9] Ratnaweera A, Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 240-255.

[10] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-Ⅱ[J]. IEEE Trans on Evolutionary Computation, 2002, 6(2): 182-197.

[11] Van Veldhuizen D A, Lamont G B. Multiobjective evolutionary algorithm research: A history and analysis[R]. Ohio: Air Force Institute of Technology, 1998.

[12] Schott J. Fault tolerant design using single and multicriteria genetic algorithm optimization[D]. Cambridge: Massachusetts Institute of Technology, 1995.