文章编号: 1001-0920 (2015) 08-1417-06 DOI: 10.13195/j.kzyjc.2014.0709

自适应动态重组多目标粒子群优化算法

倪红梅[a,b], 刘永建[a], 李盼池[b]

(东北石油大学 a. 提高油气采收率教育部重点实验室, b. 计算机与信息技术学院, 黑龙江 大庆 163318)

摘 要: 提出一种自适应动态重组粒子群优化算法. 该算法采用凝聚的层次聚类算法, 将种群分成若干个子群体, 用一个精英集对非支配解进行存储; 根据贡献度和多样性, 对各子群体的粒子和整个种群进行自适应动态重组; 同时引入扰动算子对精英集存储的非支配解进行扰动, 实现对精英集进行动态调整. 利用具有不同特点的测试函数进行验证并与同类算法相比较, 结果表明, 所提出的算法可加快收敛速度, 提高种群的可进化能力.

关键词: 多目标优化; 动态重组; 粒子群优化; 精英集

中图分类号: TP18 **文献标志码:** A

Adaptive dynamic reconfiguration multi-objective particle swarm optimization algorithm

NI Hong-mei[a,b], *LIU Yong-jian*[a], *LI Pan-chi*[b]

(a. MOE Key Laboratory of Enhanced Oil Recovery, b. School of Computer and Information Technology, Northeast Petroleum University, Daqing 163318, China. Correspondent: NI Hong-mei, E-mail: nhm257@163.com)

Abstract: The adaptive dynamic reconfiguration multi-objective particle swarm optimization algorithm is proposed, which uses the agglomerate hierarchical clustering algorithm to divide the population into several groups, and stores the non-dominated solutions with an elite set. According to the contribution and diversity, the particles of various subgroups and the entire population are reconfigured adaptively and dynamically. At the same time, a perturbed operator is introduced to disturb non-dominated solutions set stored in the elitist set, achieving dynamically adjustment to the elite set. Test functions with different characteristics are used to verify the proposed algorithm which is compared with other similar algorithms. The results show that the proposed algorithm can speed up the convergence speed and increase the evolving ability of the population.

Keywords: multi-objective optimization; dynamic reconfiguration; PSO; elitist set

0 引 言

多目标优化问题 (MOP) 广泛存在于科学研究与工程实践. 因为MOP不存在唯一的全局最优解, 所以多目标优化算法的目的是寻找一组Pareto最优解的集合. 由于群体智能算法可以在每次迭代中得到多个Pareto最优解, 非常适合求解MOP. 粒子群优化 (PSO) 算法是一种新兴的群体智能算法, 以其实现简单、参数设置少、高效并行搜索等优点已被广泛地应用于各类复杂优化问题的求解[1-3]. 经验显示, PSO算法不但在求解单目标优化问题上表现十分出色, 而且也是解决多目标优化问题的有效方法, 但在求解过程中极易陷入局部最优解而导致典型的早熟收敛.

为了克服上述缺点, 提高算法的收敛速度, 一些研究者提出了许多改进算法. 文献 [4] 应用非约束的存档方案, 利用一种支配树的数据结构来选择每个群体成员的全局最优个体, 同时引入了一个扰动算子以保持多样性; 文献 [5] 采用自适应网格机制的外部种群, 依据种群迭代次数来设置变异尺度, 实现粒子的变异; 文献 [6] 将小生境技术引入MOPSO算法, 通过增加独立优化的子群体来提高算法的效率; 文献 [7] 提出了多子群体进化算法, 利用非支配集中的个体构成一个全局最优区域来指导整个粒子群的进化, 并且通过子群体间的信息交换保证解的多样性和分布的均匀性; 文献 [8] 提出了一种动态多

收稿日期: 2014-05-09; **修回日期:** 2014-11-14.

基金项目: 国家科技重大专项课题 (2011ZX05012-003); 国家自然科学基金项目 (61170132); 黑龙江省教育厅科学技术研究项目 (12521058).

作者简介: 倪红梅 (1975−), 女, 副教授, 博士生, 从事进化计算、优化设计的研究; 刘永建 (1955−), 男, 教授, 博士生导师, 从事采油采气化学理论与工程等研究.

群MOPSO算法, 为了保证算法的收敛性和分布性, 在整个迭代过程中子群体数量是自适应分配的.

本文提出一种自适应动态重组粒子群优化(ADRMOPSO)算法, 主要作了如下改进: 1) 依据每个子群体的贡献度和多样性大小, 确定各子群体中粒子重组的个数和幅度, 借鉴差分算法[9]的思想进行粒子自适应动态重组, 引导粒子跳出局部最优, 从而提高可进化能力; 2) 采用凝聚的层次聚类算法对整个种群进行自适应动态重组, 通过重组增加各子群体的信息交换, 加快收敛速度; 3) 引入随机扰动算子, 对精英集进行随机扰动, 可实现对Pareto解集的动态调整, 加快Pareto解集的搜索和逼近速度. 通过与经典算法NSGA-II[10]、PESA-II[11]、MOPSO[5]、AMOPSO[12]和PSO-IMOCA[13]比较, 所得结果表明了ADRMOPSO算法求解MOP的有效性.

1 PSO算法

在PSO算法[14]中, 搜索空间中的每个粒子即为解空间的一个解. 在每次迭代中, 粒子通过动态跟踪个体极值和全局极值来更新其速度和位置, 具体更新公式如下:

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 (p_i^t - x_i^t) + c_2 r_2 (p_g^t - x_i^t). \tag{1}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}. \tag{2}$$

其中: v_i^t 表示粒子 i 在第 t 代的速度矢量, 且 $v_i \in [v_{i\min}, v_{i\max}]$; x_i^t 表示粒子 i 在第 t 代的位置矢量, 且 $x_i \in [x_{i\min}, x_{i\max}]$; p_i^t 表示粒子 i 得到的历史最优解; p_g^t 表示整个粒子群内所搜索到的历史最优解; ω 为惯性权重; c_1 和 c_2 为学习因子; r_1 和 r_2 为均匀分布在[0,1]之间的随机数; $i = 1, 2, \cdots, N$, N 表示粒子种群的规模.

2 ADRMOPSO算法

在ADRMOPSO算法的实现过程中, 应用了贡献度和多样性指标, 下面给出贡献度和多样性的定义.

设种群Pop共有 m 个粒子, 分为 n 个子群体, 即 $\text{Pop} = C_1, C_2, \cdots, C_n$, 每个子群体分别有 $m_i (i = 1, 2, \cdots, N)$ 个粒子, 粒子维数为 D.

1) 贡献度.

假设在第 t 次迭代中, 若从第 i 个子群体加入到精英集ES中的非支配解有 s_i 个, 则第 i 个子群体的贡献度

$$\eta_i = s_i / m_i, \tag{3}$$

整个种群的贡献度

$$\phi = \frac{1}{m} \sum_{i=1}^{n} s_i. \tag{4}$$

2) 多样性.

设搜索空间的最长对角线为 $|L|$, x_{ij}^k 为子群体 C_k 中第 i 个粒子在第 j 维的分量, $\overline{x}_j^k$ 为子群体 C_k 中所有粒子在第 j 维的均值, x_{ij} 为整个种群第 i 个粒子在第 j 维的分量, $\overline{x}_j$ 为整个种群所有粒子在第 j 维的均值, 则子群体 C_k 的多样性和整个种群的多样性分别表示为

$$d_k = \frac{1}{m_k \times |L|} \times \sum_{i=1}^{m_k} \sqrt{\sum_{j=1}^{D} (x_{ij}^k - \overline{x}_j^k)^2}, \tag{5}$$

$$\sigma = \frac{1}{m \times |L|} \times \sum_{i=1}^{m} \sqrt{\sum_{j=1}^{D} (x_{ij} - \overline{x}_j)^2}. \tag{6}$$

2.1 粒子重组算子

在设计粒子重组算子时, 依据每个子群体的贡献度 η_i 和多样性 d_i, 利用差分算法思想实现粒子的重组. 首先, 利用下式确定粒子重组的个数 N 和幅度 F:

$$N = (1 - \eta_i) \times C_i, \tag{7}$$

$$F = (1 - d_i)/3; \tag{8}$$

然后, 在 C_i 中随机选择1个粒子 X_i, 随机在其他两个不同的子群体中各随机选择1个粒子 X_{r1} 和 X_{r2}, 计算 X_i 重组后粒子 X_i', 即

$$X_i' = X_i + F(X_{r1} - X_{r2}); \tag{9}$$

最后比较 X_i' 与 X_i, 保留非支配解, 若二者相互不支配, 则随机保留其中一个.

由式(7)和(8)可以看出: 贡献度 η_i 越大, 重组的粒子个数越少, 多样性 d_i 越大, 重组的幅度越小, 重组后的新粒子 X_i' 与原粒子 X_i 变化的程度也就较小, 从而实现算法的局部搜索; 反之, 贡献度 η_i 越小, 重组的粒子个数越多, 多样性 d_i 越小, 重组的幅度也越大, 重组后的新粒子 X_i' 与原粒子 X_i 变化的程度也就较大, 可以帮助跳出局部最优, 提高全局搜索能力.

算子1 粒子重组算子.

Step 1: 初始化粒子群中粒子重组触发标志变量 $\text{Particle\_flag}[i] = \text{False}$, $i = 1, 2, \cdots, n$;

Step 2: 计算子群体 C_i 的贡献度 η_i 和多样性 d_i, $i = 1, 2, \cdots, n$;

Step 3: 若 η_i 小于子群体贡献度阈值 α 或者 d_i 小于子群体多样性阈值 β, 则 $\text{Particle\_flag}[i] = \text{True}$, $i = 1, 2, \cdots, n$;

Step 4: 若 $\text{Particle\_flag}[i] = \text{True}$, $i = 1, 2, \cdots, n$, 则执行以下操作:

Step 4.1: 由式(7)和(8)计算粒子重组的个数 N 和幅度 F;

Step 4.2: 令 $\text{num} = 1$;

Step 4.3: 在第 C_i 个子群体中随机取出1个粒子, 然后随机在其他两个不同的子群体中各随机选择1个粒子, 由式(9)进行重组;

Step 4.4: 比较 X_i' 与 X_i, 保留非支配解, 若二者相互不支配, 则随机保留其中一个.

Step 4.5: $\text{num} = \text{num} + 1$, 若 $\text{num} < N$, 则转到 Step 4.3.

2.2 种群重组算子

种群重组算子是将所有子群体的粒子又重新合并成一个大种群, 然后根据凝聚的层次聚类算法再重新进行分组, 产生各个子群体, 从而实现子群体随着进化自适应地动态变化. 由于每个粒子有机会分到不同子群体中和不同的粒子进行进化, 这样可增加各子群体间的信息交换, 提高算法的寻优能力.

采用凝聚的层次聚类方法[15]实现种群重组, 需要建立相似度矩阵 M, 即

$$M = \begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(N,1) & d(N,2) & d(N,3) & \cdots & 0 \end{bmatrix}, \quad (10)$$

其中

$$d(i,j) = \sqrt{\sum_{d=1}^{D}(x_i^d - x_j^d)^2}. \quad (11)$$

这里: $d(i,j)$ 为个体 i 与 j 之间的相异度, 并有 $d(i,j) = d(j,i)$, $d(i,i) = 0$; D 为粒子维数; N 为粒子的个体数.

算子 2　种群重组算子.

Step 1: 初始化种群重组触发标志变量 $\text{Pop\_flag} = \text{false}$;

Step 2: 计算种群的贡献度 ϕ 和多样性 σ;

Step 3: 若 ϕ 小于种群贡献度阈值 λ, 并且 σ 小于种群多样性阈值 θ, 则 $\text{Pop\_flag} = \text{True}$;

Step 4: 若 $\text{Pop\_flag} = \text{True}$, 则转到 Step 5, 否则结束算法;

Step 5: 建立聚类表 G, 种群中的每个粒子分别当作一个子类, 即 $G[i] = \text{pop}[i], i = 1, 2, \cdots, m$;

Step 6: 依据式 (11), 计算聚类表 G 中各子类之间距离, 得到初始化距离矩阵 M;

Step 7: 从距离矩阵 M 中, 找出距离最近的两个子类 s 和 t 进行合并, 即 $t = t + s$, 从聚类表 G 中删除子类 s;

Step 8: 重新计算距离矩阵 M, 重复 Step 7, 直到所有子类中的个体多余 1 个为止;

Step 9: 输出聚类表 G, 即为种群重组结果.

2.3 精英集更新策略

本文中的精英集是用来保存整个种群在整个搜索过程中产生的非支配解. 在每一个新的非支配解加到精英集后, 若精英集中的个体数目超过指定规模, 则对精英集中的个体进行修剪操作. 本文借鉴文献 [10] 所采用的基于拥挤距离的个体选择策略完成精英集的修剪操作. 该策略为了保持解群体的分布性和多样性, 每次删除拥挤距离最小的个体. 同时, 在精英集中引入随机扰动算子, 对每次迭代得到的精英集中的非支配解进行随机扰动, 比较扰动前后的解, 保留非支配解, 若二者相互不支配, 则随机保留其中一个.

在精英集中引入随机扰动算子, 可得更优或更有潜力的个体, 这样不但可增强全局搜索能力, 而且能以较快的速度收敛到理想的 Pareto 最优解集, 随机扰动可表示为

$$X_i' = U(0,1) \times X_i, \quad (12)$$

其中 $U(0,1)$ 为均匀分布在 [0,1] 之间的随机数.

2.4 整个算法流程

Step 1: 设置粒子群 Pop 的基本参数, 初始位置作为每个个体的历史最优位置 P_{best};

Step 2: 利用凝聚的层次聚类算法将种群 Pop 分组, 产生 n 个子群体 $\text{Pop} = C_1, C_2, \cdots, C_n$;

Step 3: 计算子群体 C_i 中每个粒子的目标向量, 找出其中的非支配解, 放入非支配解集 NDS 中, 再将 NDS 中的解与精英集 ES 中的解进行比较, 加入新的非支配解到精英集 ES 中, 删除被支配的解, 记录加入解的个数;

Step 4: 每个子群体 C_i 执行粒子重组算子, $i = 1, 2, \cdots, n$;

Step 5: 执行种群重组算子;

Step 6: 执行精英集更新策略;

Step 7: 采用轮盘赌方法为每个粒子在精英集中选择一个粒子, 作为该粒子的全局极值 G_{best}, 更新粒子的个体极值 P_{best}, 根据式 (1) 和 (2) 更新所有粒子的速度和位置;

Step 8: $t = t + 1$;

Step 9: 如果 $t < \text{Gen}$, 则转到 Step 3;

Step 10: 输出精英集 ES 作为最终的 Pareto 支配集.

3 性能验证

3.1 测试函数和指标

为了验证 ADRMOPSO 算法的有效性, 本文选用 SCH、FON、ZDT 1 ~ ZDT 4[10]、DLTZ 2 和 DLTZ 7[16]8 个有代表性的标准测试函数. 在实验中采用文献 [10] 提出的收敛度 γ 和多样度 Δ 来比较各种算法的性能. 这两项指标的定义如下.

1) 收敛度

$$\gamma = \frac{1}{|A|}\sum_{i=1}^{|A|}\min_{1\leqslant j\leqslant|p^*|}(|a_i - p_j|). \tag{13}$$

其中: $p^* = (p_1, p_2, \cdots, p_{|p_*|})$ 为真实的非劣解集, $A = (a_1, a_2, \cdots, a_{|A|})$ 为算法得到非劣解集.

2) 多样度

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1}|d_i - \overline{d}|}{d_f + d_l + (N-1)\overline{d}}. \tag{14}$$

其中: d_i 为Pareto解集中相邻两个解之间的距离; $\overline{d}$ 为所有距离 d_i 的均值; d_f 和 d_l 分别为Pareto解集的边界解与极端解之间的距离.

3.2 参数设置

实验结果与NSGA-II[10]、PESA-II[11]、MOPSO[5]、AMOPSO[12]和PSO-IMOCA[13]五种算法进行了比较. 这5种算法的参数设置如下: NSGA-II算法和PESA-II算法的种群规模为100, 最大迭代次数为250; MOPSO算法的种群规模为50, 最大迭代次数为500, 惯性权值 $\omega = 0.729\,8$, 学习因子 $c_1 = c_2 = 1.496\,18$; AMOPSO算法和PSO-IMOCA算法的种群规模为50, 最大迭代次数为500; 函数评价次数均为25 000次, 输出的Pareto解集大小均为100. NSGA-II算法和PESA-II算法中 $P_c = 0.8$, $P_m = 1/n$, 其中, n 代表多目标优化问题决策变量维数.

本文ADRMOPSO算法的参数经验地选取为: 种群规模 $N = 200$, 最大迭代次数 $\mathrm{Gen} = 125$, 惯性权重 $\omega = 0.729\,8$; 学习因子 $c_1 = c_2 = 1.496\,18$, 子群体贡献度阈值 $\alpha = 0.3$, 子群体多样性阈值 $\beta = 0.2$, 种群贡献度阈值 $\lambda = 0.05$, 种群多样性阈值 $\theta = 0.15$, 算法重复运行30次. 算法采用Matlab编程, 仿真实验在CPU为4 G的PC机上进行.

3.3 实验结果与分析

实验比较了ADRMOPSO与NSGA-II、PESA-II、MOPSO、AMOPSO和PSO-IMOCA这6种算法对多目标函数的求解性能. 图1为在8个测试函数上应用ADRMOPSO算法所求得的Pareto最优前沿与理想Pareto最优前沿的对比. 表1和表2为6种算法对应的收敛度 γ 和多样性 Δ 的比较. 其中: M 为平均值, VAR为方差, "–"为参考文献中无该项统计结果, NSGA-II, PESA-II, MOPSO, AMOPSO, PSO-IMOCA的实验结果来自于文献[12]和[13].

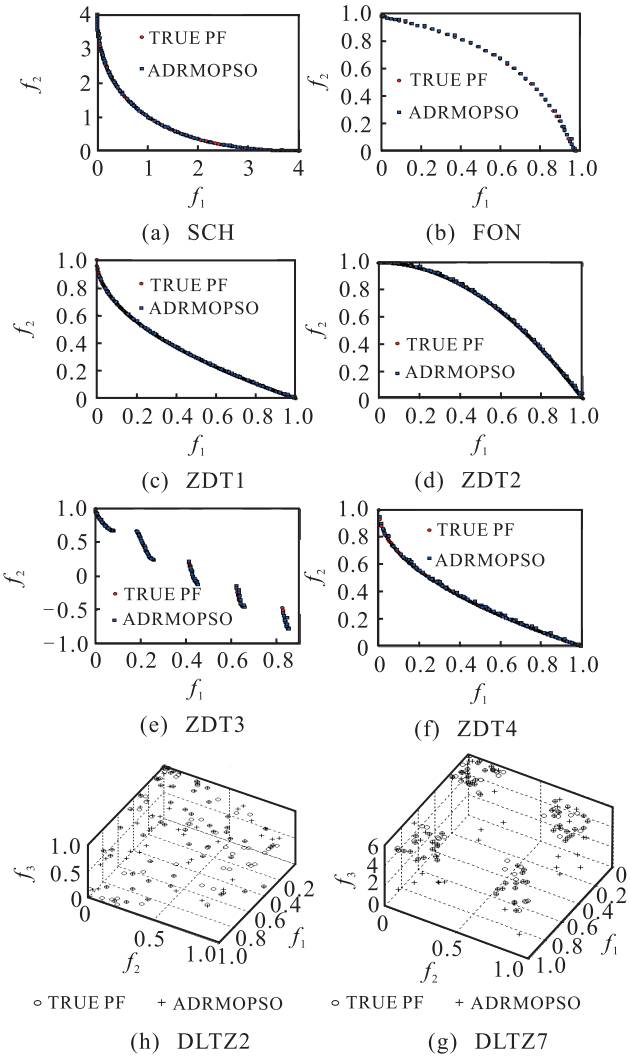

图1 ADRMOPSO在8个测试函数上的Pareto前沿

图1中, 针对2目标函数SCH, FON, 以及ZDT1～ZDT4测试函数, ADRMOPSO算法求解的Pareto最优解集的分布接近于理想Pareto最优前沿, 并且所得解

表1 收敛度 γ 的比较

算法	指标	SCH	FON	ZDT1	ZDT2	ZDT3	ZDT4	DLTZ2	DLTZ7
NSGA-II	M	0.003 39	0.001 93	0.033 48	0.072 39	0.114 50	0.513 05	0.727 75	0.049 76
	VAR	0.000 00	0.000 00	0.004 75	0.031 68	0.007 94	0.118 46	0.204 00	0.000 00
PESA-II	M	0.016 87	0.020 02	0.001 05	0.000 74	0.007 89	9.982 54	0.032 19	0.051 39
	VAR	0.000 00	0.000 00	0.000 00	0.000 00	0.000 11	20.134 0	0.000 00	0.000 00
MOPSO	M	0.011 48	0.001 22	0.001 33	0.000 89	0.004 18	7.374 29	0.827 99	0.049 86
	VAR	0.000 00	0.000 00	0.000 00	0.000 00	0.000 00	5.482 86	0.011 33	0.000 00
AMOPSO	M	0.008 00	0.001 20	0.000 99	0.000 74	0.003 91	0.403 11	0.020 24	0.023 06
	VAR	0.000 00	0.000 00	0.000 00	0.000 00	0.000 00	0.012 59	0.000 00	0.000 00
PSO-IMOCA	M	0.003 10	0.001 70	0.001 10	0.000 79	0.001 30	—	—	—
	VAR	0.000 00	0.000 00	0.000 00	0.000 00	0.000 00	—	—	—
ADRMOPSO	M	0.000 76	0.000 62	0.000 94	0.000 57	0.003 89	0.430 50	0.012 59	0.009 73
	VAR	0.000 00	0.000 00	0.000 00	0.000 00	0.000 00	0.109 42	0.000 00	0.000 00

表 2　多样 Δ 的比较

算法	指标	SCH	FON	ZDT1	ZDT2	ZDT3	ZDT4	DLTZ2	DLTZ7
NSGA-II	M	0.477 89	0.378 06	0.390 31	0.430 77	0.738 54	0.702 61	0.975 99	0.891 49
	VAR	0.003 47	0.000 64	0.001 87	0.004 72	0.019 71	0.064 65	0.007 38	0.000 52
PESA-II	M	0.650 25	0.852 17	0.848 16	0.892 92	1.227 31	1.011 36	0.748 08	0.747 91
	VAR	0.012 98	0.000 32	0.002 87	0.005 74	0.029 25	0.000 72	0.000 93	0.001 06
MOPSO	M	0.760 97	0.849 43	0.681 32	0.639 22	0.831 95	0.961 94	0.748 08	0.873 75
	VAR	0.016 43	0.000 16	0.013 35	0.001 14	0.008 92	0.001 14	0.000 93	0.081 86
AMOPSO	M	0.320 74	0.724 22	0.318 26	0.319 96	0.531 54	0.650 60	0.672 73	0.732 01
	VAR	0.000 23	0.000 03	0.000 60	0.000 68	0.000 36	0.003 76	0.000 87	0.001 34
PSO-IMOCA	M	0.024 90	0.019 60	0.023 50	0.023 30	0.016 0	—	—	—
	VAR	0.000 00	0.000 00	0.000 00	0.000 00	0.000 00	—	—	—
ADRMOPSO	M	0.234 45	0.201 94	0.276 25	0.298 62	0.479 52	0.512 47	0.401 95	0.386 57
	VAR	0.004 29	0.000 37	0.001 04	0.001 36	0.006 17	0.041 82	0.000 64	0.000 39

分布均匀, 具有良好的多样性. 尽管ZDT3测试函数的真实Pareto前沿为非连续函数, ZDT4测试函数是一个多峰函数, 但由于ADRMOPSO算法采用了自适应动态重组和精英集更新策略, 实现了Pareto解集的动态调整, 增加了解的多样性, 加快了收敛速度, 从而较好地完成了Pareto解集的搜索和逼近, 提高了算法的可进化能力.

DLTZ2和DLTZ7测试函数是3目标函数, 决策变量个数分别是12和22. DTLZ2的Pareto最优边界是第1象限内的单位球面, 且3个目标函数值均满足 $\sum_{i=1}^{3}(f_i)^2 = 1$. 从图1中可见, ADRMOPSO算法所获得的Pareto解分布均匀, 具有良好的多样性; DTLZ7具有一组不连续Pareto最优边界; Pareto最优边界离散地分布在4个区域中, ADRMOPSO算法使最优个体在不同Pareto最优边界保持较好的分布度.

从表1和表2解的收敛度和多样度可以看出, ADRMOPSO算法对于大部分测试函数均优于其他算法. 从实验数据可以看出, 自适应动态重组和精英集更新策略的引入, 能够引导非支配解快速接近Pareto最优解, 这样极大地提高了算法的收敛性和分布性, 尤其表现在ZDT4问题上. 由于ZDT4是一个多峰函数, 多重局部Pareto前端导致很多算法都难以收敛到真实解, PESA-II和MOPSO算法的平均收敛度分别为9.982 54和7.374 29, 收敛性能较差, 但ADRMOPSO算法的平均收敛度为0.430 50, 比这两种算法降低了很多, 且收敛性能较好. 针对ZDT4问题, 在多样性方面, ADRMOPSO算法与NSGA-II、PESA-II、MOPSO和AMOPSO四种算法相比, 平均多样度分别减少了0.190 14, 0.498 89, 0.449 47和0.138 13. 针对3目标DLTZ2和DLTZ7测试函数, 从表1和表2解的收敛度和多样度可见, ADRMOPSO算法的均值和方差均优于其他算法, 故该算法在求解这8个测试函数时, 整体来说优于其他5种算法, Pareto最优解集更接近真实Pareto前沿, 具有良好的收敛性和分布性. 从实验结果看, ADRMOPSO算法在3目标函数上比在2目标函数上具有更好的优化性能, 这表明ADRMOPSO算法更适合于高维多目标优化问题.

3.4　参数敏感性分析

由于种群规模和迭代次数参数的设置不同, 对ADRMOPSO算法的性能有很大影响, 下面对这两个参数进行敏感性分析. 以ZDT1～ZDT4测试函数为例, 首先固定迭代次数 $\mathrm{Gen} = 100$, 确定种群规模 N 的值. 表3给出了不同种群规模对寻优效果的影响.

表 3　不同种群规模对寻优效果的影响

N	指标	ZDT1	ZDT2	ZDT3	ZDT4
20	γ	0.533 98	0.706 49	2.588 30	5.162 93
	Δ	0.725 24	0.827 95	1.411 07	1.549 48
80	γ	0.031 93	0.204 86	0.775 31	2.829 57
	Δ	0.534 22	0.527 59	0.633 92	0.937 40
150	γ	0.002 75	0.001 46	0.008 93	0.709 61
	Δ	0.418 49	0.467 02	0.579 52	0.774 74
200	γ	0.001 06	0.000 93	0.005 24	0.493 90
	Δ	0.276 25	0.298 62	0.479 52	0.512 47
300	γ	0.000 85	0.000 63	0.004 96	0.428 10
	Δ	0.263 79	0.295 17	0.458 72	0.524 87

从表3可以看出, 当种群规模 N 过小时, 收敛度和多样度的值都非常大, 算法的收敛性和分布性都较差. 当种群规模 N 由20增加到80时, 收敛度和多样度的值都降低了很多, 算法的寻优能力提高了很多. 由此可知, 种群规模 N 的大小影响着种群和粒子的重组效果, 种群规模越大, 重组效果越好, 但会影响算法运行速度. 从实验中发现, 当 $N = 200$ 时, 每个函数的优化结果都是较好的.

表 4　不同迭代次数对寻优效果的影响

Gen	指标	ZDT1	ZDT2	ZDT3	ZDT4
50	γ	0.012 47	0.101 98	0.383 34	1.076 23
	Δ	0.469 27	0.495 03	0.573 61	0.824 87
100	γ	0.001 06	0.000 93	0.005 24	0.493 90
	Δ	0.276 25	0.298 62	0.479 52	0.512 47
125	γ	0.000 94	0.000 57	0.003 89	0.430 50
	Δ	0.000 00	0.000 00	0.000 00	0.109 42
150	γ	0.000 83	0.000 52	0.003 61	0.421 17
	Δ	0.000 00	0.000 00	0.000 00	0.096 45
200	γ	0.000 79	0.000 46	0.003 44	0.413 81
	Δ	0.000 00	0.000 00	0.000 00	0.095 92

固定种群规模N为200, 确定迭代次数Gen的值. 表4给出了不同迭代次数对寻优效果的影响. 从表4可以看出, 迭代次数大于100后, 收敛度和多样度的值随着迭代次数的增加变化不大, 这说明此时迭代次数对寻优效果影响不大.

4 结 论

本文提出了一种自适应动态重组多目标粒子群优化算法. 该算法依据贡献度和多样性, 自适应地动态触发粒子重组算子和种群重组算子. 在设计粒子重组算子时, 利用差分算法的思想完成各粒子的重组, 可以帮助跳出局部最优, 提高全局搜索能力. 在设计种群重组算子时, 采用凝聚的层次聚类算法进行分组, 产生各个子群体, 使粒子有机会到不同的子群体中与不同的粒子进行进化, 提高了种群的多样性和分布性. 同时, 为了提高搜索精度, 加快收敛效率, 引入随机扰动算子对精英集进行更新, 从而指导算法快速收敛到Pareto最优解, 有效地解决了收敛性问题.

从实验结果可以看出, ADRMOPSO算法与其他5种有效的多目标优化算法相比, 不同程度地提高了种群的收敛性和多样性. ADRMOPSO算法在求得Pareto最优解集的逼近性、均匀性和宽广性上具有明显优势, 可提高进化能力.

参考文献(References)

[1] 倪红梅, 刘永建, 范英才, 等. 基于改进粒子群算法的蒸汽驱注采方案优化[J]. 石油学报, 2014, 35(1): 114-117.
(Ni H M, Liu Y J, Fan Y C, et al. Injection and production project optimization of steam flooding based on improved particle swarm optimization algorithm[J]. Acta Petrolei Sinica, 2014, 35(1): 114-117.)

[2] 王维刚, 刘占生. 多目标粒子群优化的支持向量机及其在齿轮故障诊断中的应用[J]. 振动工程学报, 2013, 26(5): 743-750.
(Wang W G, Liu Z S. Support vector machine optimized by multi-objective particle swarm and application in gear fault diagnosis[J]. J of Vibration Engineering, 2013, 26(5): 743-750.)

[3] 黄艳, 臧传治, 于海斌. 基于改进粒子群优化的无线传感器网络定位算法[J]. 控制与决策, 2012, 27(1): 156-160.
(Huang Y, Zang C Z, Yu H B. Localization method based on modified particle swarm optimization for wireless sensor networks[J]. Control and Decision, 2012, 27(1): 156-160.)

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

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

[6] Brits R, Engelbrecht A P, Bergh F. Locating multiple optima using particle swarm optimization[J]. Applied Mathematics and Computation, 2007, 189(2): 1859-1883.

[7] 张利彪, 周春光, 刘小华, 等. 求解多目标优化问题的一种多子群体进化算法[J]. 控制与决策, 2007, 22(11): 1313-1320.
(Zhang L B, Zhou C G, Liu X H, et al. A multiple subswarms evolutionary algorithm for multi-objective optimization problems[J]. Control and Decision, 2007, 22(11): 1313-1320.)

[8] Yen G G, Leong W F. Dynamic multiple swarms in multiobjective particle swarm optimization[J]. IEEE Trans on Systems, Man and Cybernetics, Part A: Systems and Humans, 2009, 39(4): 890-911.

[9] 刘波, 王凌, 金以慧. 差分进化算法研究进展[J]. 控制与决策, 2007, 22(7): 721-729.
(Liu B, Wang L, Jin Y H. Advances in differential evolution[J]. Control and Decision, 2007, 22(7): 721-729.)

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

[11] Corne D W, Jerram N R, Knowles J D, et al. PESA-II: Region-based selection in evolutionary multiobjective optimization[C]. Proc of the Genetic and Evolutionary Computing Conf. San Francisco: Morgan Kaufmann, 2001: 283-290.

[12] Tripathi P K, Bandyopadhyay S. Adaptive multi-objective particle swarm optimization algorithm[C]. Proc of IEEE Congress on Evolutionary Computation. Piscataway, 2007: 2281-2288.

[13] 吴亚丽, 徐丽青. 一种基于粒子群算法的改进多目标文化算法[J]. 控制与决策, 2012, 27(8): 1127-1132.
(Wu Y L, Xu L Q. An improved multi-objective cultural algorithm based on particle swarm optimization[J]. Control and Decision, 2012, 27(8): 1127-1132.)

[14] Kennedy J, Eberhart R. Particle swarm optimization, neural networks[C]. Proc of IEEE Int Conf on Neural Networks. Perth, 1995: 1942-1948.

[15] 郑金华, 史忠植, 谢勇. 基于聚类的快速多目标遗传算法[J]. 计算机研究与发展, 2004, 41(7): 1081-1087.
(Zheng J H, Shi Z Z, Xie Y. A fast multi objective genetic algorithm based on clustering[J]. J of Computer Research and Development, 2004, 41(7): 1081-1087.)

[16] Deb K, Thiele L, Laumanns M, et al. Scalable multi-objective optimization test problems[C]. Proc of the Congress on Evolutionary Computation. Piscataway: IEEE Service Center, 2002: 825-830.

(责任编辑: 滕 蓉)