文章编号: 1001-0920 (2015) 09-1653-07 **DOI:** 10.13195/j.kzyjc.2014.0991

寻求“理想”解的改进多目标粒子群优化算法

周 黎[a], 周承恩[b], 李海滨[b]

(内蒙古工业大学 a. 管理学院，b. 理学院，呼和浩特 010051)

摘 要: 如何在众多非劣解中为决策者推荐一个合理的方案是使用多目标粒子群算法 (MOPSO) 所面临的问题. 为此, 将逼近理想解的排序方法 (TOPSIS 策略) 引入到算法中. 为了提高求解精度和均匀性, 还提出了基于 Pbest 的变异策略和改进的 k 邻近距离策略. 测试结论显示, 仅使用 TOPSIS 策略确定 Gbest 的算法, 求解精度虽好, 但均匀性较差, 而包含所有改进策略的算法在精度和均匀性方面都更优, 并且能够按照 TOPSIS 方法在非劣解集中找到一个适合向决策者推荐的“理想”方案.

关键词: 多目标优化；粒子群优化；TOPSIS 策略；变异策略；k 邻近距离

中图分类号: TP202.7 **文献标志码:** A

Improved multi-objective particle swarm optimization algorithm that can give “ideal” solution

ZHOU Li[a], *ZHOU Cheng-en*[b], *LI Hai-bin*[b]

(a. College of Management，b. College of Science，Inner Mongolia University of Technology，Huhhot 010051，China. Correspondent：ZHOU Cheng-en，E-mail：zce@imut.edu.cn)

Abstract: One problem of using multi-objective particle swarm optimization algorithm(MOPSO) is how to find a recommendable solution within the non-inferior solution set for decision-makers. Therefore, the technique for order preference by similarity to ideal solution(TOPSIS) strategy is introduced into the MOPSO algorithm. In order to improve the accuracy and uniformity of solutions, two other strategies are also proposed: the mutation operator according to Pbest, and the modified method to calculate k neighbor distance. The results show that the solution uniformity of the algorithm which chooses Gbest using the TOPSIS method only is not satisfied, while the algorithm that uses all improvement strategies has better performance. An ideal solution can be found for decision-makers according to the TOPSIS strategy.

Keywords: multi-objective optimization；particle swarm optimization；TOPSIS；mutation strategy；k neighbor distance

0 引 言

多目标粒子群算法 (MOPSO) 实现简单, 收敛速度快, 已广泛应用于多目标优化领域. 最初的 MOPSO 策略是通过降低目标空间的维数将多目标优化问题转变成为单目标问题[1-2], 目前更多 MOPSO 是基于帕雷托支配的方法, 种群中的每个粒子都需要帕雷托最优解集中的一个解作为全局最优粒子 (Gbest). Alvarez-Benitez 等[3]采用了最简单的方式, 在所有非支配解中随机选择一个作为 Gbest. 这种做法会增加解的多样性, 但是由于更新过程中破坏了系统的“记忆”而降低了收敛的速度. 为了克服这个问题, 在确定 Gbest 时, 一种方法是使用拥挤距离 (CD) 指标, 另一种方法是对找到的解空间进行超立方体划分形成“网格”(grids)[4-5]. 网格策略、拥挤距离以及变异算子等各种策略的使用都有效地改进了 MOPSO 的性能[6].

MOPSO 改进策略的目的有以下几种: 使种群更快地靠近前沿面, 提高解精度; 提高解空间搜索能力, 避免种群陷入局部最优; 保证解分布的多样性和均匀性. 使用一种改进策略提高某个性能指标时可能会同时损害其他方面的性能, 为此以同时提高算法多种性能为目的的各种混合策略[7-8]被提出.

与所有进化算法一样, MOPSO 最终找到的非劣解集包含很多解, 这对于决策者没有实际意义. 因此, 如何在最终的众多非劣解中为决策者推荐一个较为理想的决策方案是应用 MOPSO 所面临的一个难题. 为此, Ching-Shih[9]把 MOPSO 和逼近理想解的排序方

收稿日期: 2014-06-21；**修回日期:** 2014-12-14.

基金项目: 国家自然科学基金项目(11262014)；内蒙古自然科学基金项目(2014MS0709).

作者简介: 周黎(1971−), 女, 副教授, 博士, 从事决策理论、信息管理的研究；周承恩(1972−), 男, 副教授, 博士, 从事计算力学、分子动力学的研究.

法(TOPSIS)结合使用, 首先使用MOPSO找到多目标库存规划的非劣解集, 然后使用TOPSIS方法确定一个最优解, 以便为决策提供依据, 这是两种方法的分阶段使用.

粒子群算法的求解过程是找到很多离散的解, 根据这些解确定相应的理想解和负理想解, 这为使用TOPSIS方法提供了基础. 因此, 本文的研究思路是把TOPSIS策略直接应用到MOPSO的求解过程中. 可以使用该策略确定每一步的Gbest, 以便引导种群更快地靠近帕雷托前沿, 同时还可以依据该策略为决策者提供一个最优方案. 本文还提出了其他改进策略以全面改善算法性能.

1 标准多目标粒子群算法

1.1 多目标问题

多目标决策问题的目标多于一个, 目标间存在矛盾性和不可公度性. 多目标决策问题可以表示为

$$\begin{aligned}&\min\{f_1(x), f_2(x), \cdots, f_m(x)\};\\&\text{s.t. } x \in X.\end{aligned} \tag{1}$$

其中 $x = (x_1, x_2, \cdots, x_n)^{\mathrm{T}}$. 因为最大化问题容易转换成为最小化问题, 这里设所有的目标都是求最小化.

1.2 标准粒子群算法

粒子群算法由Eberhart等[10]于1995年提出.1998年Shi等[11]将惯性权重引入进化方程, 以改善基本粒子群算法的收敛性能.

设目标函数 $f(x)$ 是求最小化目标, 种群包含 N 个粒子, 如果种群中的每个粒子 x_i 的位置表示一个潜在的解; x_i 在运动过程中所经历的最好位置记为 Pbest_i; 全部粒子经历过的最好位置记为Gbest; $v_{ij}(t)$ 表示在第 t 次迭代后第 i 个粒子的第 j 维的速度. 则标准粒子群算法的速度更新和位置更新公式为

$$\begin{aligned}&v_{ij}(t+1) =\\&\omega v_{ij}(t) + c_1 r_{1j}(t)[\text{pbest}_{ij}(t) - x_{ij}(t)] +\\&c_2 r_{2j}(t)[\text{gbest}_j(t) - x_{ij}(t)],\end{aligned} \tag{2}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \tag{3}$$

其中: ω 为惯性权重; c_1、c_2 为在 [0, 2] 间取值的常数; r_1、r_2 为两个相互独立的随机数.

目前常用的MOPSO算法都使用外部存档机制存放每一步寻找到的非劣解, 构成精英解集. 当精英解集的规模超出限制时, 使用拥挤度策略或者网格策略剔除多余的粒子.

1.3 TOPSIS方法

TOPSIS方法是解决多属性问题的一种方法, 它借助于理想解和负理想解给方案集 X 中各方案排序. 理想解 x^* 是方案集中并不存在的虚拟的最佳方案, 它的每个属性值都是该属性的最好值. 负理想解 x^0 是虚拟的最差方案, 它的每个属性值都是该属性的最差值. 在 n 维空间中, 将方案集中的每个方案与理想解和负理想解的距离进行比较, 既靠近理想解又远离负理想解的方案就是最佳方案.

2 改进多目标粒子群算法

2.1 改进策略

在常用的精英解集(外部存档机制)、拥挤度和变异算子等策略的基础上, 本文针对MOPSO算法提出以下改进策略.

2.1.1 基于Pbest的动态可行域变异策略

如果某个粒子的Pbest长时间没有进入到精英解集, 则这个粒子可能陷入了局部极值点, 而这个极值点不是非劣解, 此时需要对该粒子采用变异策略, 引导其跳出局部最优. 本文提出动态可行域的变异策略, 即随机产生新粒子的可行域范围, 根据精英解集的粒子进行动态调整.

可以认为在所有已经找到的粒子中, 精英解集中的粒子比非精英解粒子更靠近帕雷托前沿. 为了增加解的收敛性, 根据精英解集中的粒子计算每个自变量的均值和标准差, 进而调整可行域的取值范围.

设第 j 个自变量原取值范围是 $[x_j^{\min}, x_j^{\max}]$, 根据第 t 期精英解集内所有粒子计算的第 j 个自变量的均值是 $\bar{x}_j(t)$, 标准差是 $\sigma_j(t)$, 则调整随机产生的新粒子的第 j 个自变量的取值范围为

$$[x_j^{\min}(t), x_j^{\max}(t)] = [\bar{x}_j(t) - 3\sigma_j(t), \bar{x}_j(t) + 3\sigma_j(t)]. \tag{4}$$

实际上, 该策略是利用精英解引导新粒子的产生, 新的可行域范围小于原可行域范围, 使粒子有机会更快地靠近帕雷托前沿.

2.1.2 改进的 k 邻近距离的拥挤度计算

粒子间拥挤度的度量方法有多种, 其中NSGA-II算法采用的排挤距离只适合于目标数量较少的时候, 当目标数超过2个时, 在维护分布性方面会变得困难[12]. 一般粒子群改进策略中用到的拥挤程度的度量是采用"k 邻近"距离. 首先计算每两个个体在目标空间的欧氏距离, 个体 a 与个体 b 之间的距离计算公式为

$$d(a,b) = \sqrt{\sum_{j=1}^{m} (p[a]\cdot f_j - p[b]\cdot f_j)^2}. \tag{5}$$

其中: $d(a,b)$ 表示个体 a 与个体 b 的欧氏距离, m 表示目标数, $p[a]\cdot f_j$ 和 $p[b]\cdot f_j$ 分别表示个体 a 和个体 b 的第 j 个目标的目标值.

对于精英解集中的个体 a, 所有其他个体 b 与个体 a 的距离都依据式(5)计算得出, 并把这些距离按升序排列. 其中排序第 k 个点的距离称为 a 的 k 邻近

距离, 记为σ_a^k, k 的取值由下式确定:

$$k = \mathrm{INT}(\sqrt{N+\bar{N}}), \tag{6}$$

其中 N 和 $\bar{N}$ 分别表示基本种群和精英解集的规模. 则个体 a 的密度函数为

$$D(a) = \frac{1}{\sigma_a^k + 2}. \tag{7}$$

这个密度函数的策略与 NSGA-II 中的排挤距离类似, 当目标数超出 2 个时则难以维护解的分布性.

重新定义粒子 a 的 k 邻近距离: k 邻近距离是距离 a 最近的 k 个点的距离的倒数和的倒数, 表示为

$$\mathrm{DISTK}_a = \Big(\sum_{i=1}^{k} \frac{1}{\mathrm{dis}(i,a)}\Big)^{-1}, \tag{8}$$

其中

$$\mathrm{dis}(i,a) = \sqrt{\sum_{j=1}^{m} (f_j(i) - f_j(a))^2}. \tag{9}$$

这里: m 表示目标数, $f_j(i)$ 表示第 i 个粒子的第 j 个目标值. DISTK 值越小, 表示粒子附近越“拥挤”. 与式 (7) 相比, 式 (8) 更全面地考虑了粒子 a 附近的全部 k 个粒子的“ 拥挤”程度, 而且这 k 个点对 DISTK 的影响随着距离的不同而不同, 距离越近的点影响越大, 而不是仅考虑了第 k 个点与 a 的拥挤程度, 这样更有利于维护解分布的均匀性.

2.1.3 使用 TOPSIS 方法确定 Gbest

设有 m 个目标, 所有粒子的目标值 $y = (f_1(x), f_2(x), \cdots, f_m(x))$ 都是已经标准化处理的目标值, 所有目标都是求最小值. 令 $f_j^*\,(j = 1, 2, \cdots, m)$ 表示当前所有粒子进化过程中获得的第 j 个目标的最小值, f_j^0 表示所有粒子进化过程中获得的第 j 个目标的最大值, 则正理想解 X^* 对应的目标向量是 $Y^* = (f_1^*, f_2^*, \cdots, f_m^*)$, 负理想解 X^0 对应的目标向量是 $Y^0 = (f_1^0, f_2^0, \cdots, f_m^0)$, 正理想解和负理想解都是虚拟的点, 实际并不存在.

令精英解集中第 i 个粒子与正理想解的距离为

$$\mathrm{dis}(i, X^*) = \sqrt{\sum_{j=1}^{m} (f_j(i) - f_j^*)^2}, \tag{10}$$

与负理想解的距离为

$$\mathrm{dis}(i, X^0) = \sqrt{\sum_{j=1}^{m} (f_j(i) - f_j^0)^2}, \tag{11}$$

第 i 个粒子的综合评价指数为

$$\mathrm{TOP}_i = \mathrm{dis}(i, X^0)/(\mathrm{dis}(i, X^0) + \mathrm{dis}(i, X^*)). \tag{12}$$

在精英解集中, 取综合评价指数 TOP 最大的粒子确定为 Gbest.

根据拥挤度确定 Gbest 是为了使解的分布更均匀. 与此不同, 使用 TOPSIS 方法确定 Gbest 是为了让粒子群更快地靠近帕雷托前沿, 这个 Gbest 点也可以看成在所有已找到的解中的“理想点”. 如果在算法中最后一步是根据 TOPSIS 策略确定 Gbest 点, 则该点可以看成是为决策者提供的一个最佳方案. 当决策者没有明确的决策偏好时, 这个 Gbest 点就成为一个合适的可推荐方案. 因此 TOPSIS 策略在算法中起到两个重要作用: 一是引导粒子群更快靠近帕雷托前沿; 另一个是提供一个最佳方案.

2.2 改进多目标粒子群算法的步骤

以上改进策略分别针对算法中不同的环节, 下面是改进算法的一般步骤. 设 t 表示迭代次数.

Step 1: $t = 0$, 初始化算法参数, 产生初始种群、初始 Pbest 和初始速度.

Step 2: 生成初始精英解集. 如果解集的规模超过设定, 则删除 DISTK 值最小的粒子. 选择 DISTK 值最大的粒子作为初始 Gbest. 每个粒子的 DISTK 值由式 (6)、(8) 和 (9) 计算获得.

Step 3: $t = t + 1$, 更新种群中粒子的速度和位置.

Step 4: 使用文献 [4] 提出的变异算子进行变异. 当 t 超过 50 时, 判断第 i 个粒子的 Pbest 是否进入过精英解集, 如果没有, 则对该粒子实施基于动态可行域的变异策略, 按照式 (4) 产生新的可行域, 并在其中随机产生新粒子.

Step 5: 更新 Pbest 和精英解集. 如果解集的规模超过设定, 则删除 DISTK 值最小的粒子.

Step 6: Gbest 的更新.

1) 若仅使用 TOPSIS 策略, 则按式 (10) ∼ (12) 计算精英解集所有粒子的综合评价指数 TOP, 取 TOP 值最大的粒子作为 Gbest 点.

2) 若仅使用拥挤度策略, 则取精英解集中 DISTK 值最大的粒子作为 Gbest 点.

3) 若混合使用 TOPSIS 策略和拥挤度策略, 则:

当 $t = 2n+1, n = 0, 1, \cdots$ 时, 取精英解集中 DISTK 最大的粒子作为 Gbest;

当 $t = 2n, n = 1, 2, \cdots$ 时, 采用 TOPSIS 策略, 取 TOP 值最大的粒子作为 Gbest.

Step 7: 重复 Step 3 ∼ Step 6, 直至符合停机条件, 输出精英解集和最后 Gbest 点.

2.3 改进算法复杂度分析

假设粒子群的规模为 N, 目标个数为 M, 自变量个数为 n, 则在使用基于 Pbest 的变异策略时, 需要记载下每一个粒子 Pbest 是否进入过精英解集, 其计算复杂度为 $O(N)$. 使用改进的 k 邻近距离公式计算拥挤距离时, 其排序比较的过程没有发生本质变化, 复杂度和已有的拥挤度策略一样, 仍然是 $O(MN^2)$; 使用 TOPSIS 策略确定 Gbest 时, 需要每个粒子与已有的正负理想解比较, 进而确定新的正负理想解, 其复杂度是 $O(MN)$, 这样总的复杂度是 $O(MN^2)$.

3 对改进粒子群算法的仿真测试

3.1 对照算法选取与改进算法标识

每个改进策略的有效性需要通过测试进行验证. 因为一般拥挤距离和变异算子的有效性已经得到证实[6], 所以选取的对照算法包含一般拥挤距离和变异算子策略, 其中拥挤距离由式(5)～(7)得到, 使用文献[4]提出的变异算子. 为了方便, 用“CMOPSO”表示对照算法; 用“REMOPSO”表示不包含TOPSIS策略仅使用拥挤度策略确定Gbest的改进算法; 用“TOPMOPSO”表示不使用拥挤度策略仅使用TOPSIS策略确定Gbest的改进算法; 用“THYMOPSO”表示混合使用TOPSIS策略和拥挤度策略确定Gbest的改进算法. 几种改进的算法都使用基于Pbest的动态可行域变异策略, 使用改进的k临近距离计算DISTK. 关于每种算法包含的策略和测试目的见表1.

表1 各算法包含的策略与测试目的

策略目的	对照算法CMOPSO	改进算法		
		REMOPSO	TOPMOPSO	THYMOPSO
采用策略	文献[4]的变异算子; 一般k邻近距离确定拥挤度; 拥挤度策略确定Gbest;	文献[4]的变异算子; 改进策略1; 改进策略2; 拥挤度策略确定Gbest;	文献[4]的变异算子; 改进策略1; 改进策略2; 改进策略3确定Gbest;	文献[4]的变异算子; 改进策略1; 改进策略2; 混合使用改进策略3和 拥挤度策略确定Gbest;
测试目的	在GD值、SP值和运算时间方面为改进策略提供对照数据	与CMOPSO比较, 测试改进策略1对GD值的影响, 能否跳出局部最优; 测试改进策略2对SP值的影响.	与CMOPSO、REMOPSO比较, 测试改进策略3对GD值和SP值的影响	与CMOPSO、REMOPSO、TOPMOPSO比较, 测试使用混合方法确定Gbest对GD和SP值的影响

表1中, 改进策略1表示2.1.1节提出的基于Pbest的动态可行域变异策略; 改进策略2表示2.1.2节提出的改进的k临近距离的拥挤度计算; 改进策略3表示2.1.3节提出的使用TOPSIS方法确定Gbest; 关于测度指标GD和SP见3.3节.

3.2 测试函数的选取

为了能较为全面地测试各策略的效果, 选择下面几个测试函数.

1) ZDT1, 由Zitzler[13]提出. 函数形式为

$$\begin{aligned}&\min\ (f_1(x), f_2(x)),\\ &f_1(x) = x_1,\\ &f_2(x) = g(x)\cdot h(f_1(x), g(x)),\\ &g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i,\\ &h(f_1, g) = 1 - \sqrt{f_1/g},\\ &\text{s.t. } 0 \leqslant x_i \leqslant 1,\ i = 1, 2, \cdots, n,\end{aligned} \tag{13}$$

其中$n = 30$. 该函数理论的帕雷托前沿是

$$f_2 = 1 - \sqrt{f_1},\ 0 \leqslant x_1 \leqslant 1,\ x_i = 0,\ i = 2, 3, \cdots, n.$$

用这个函数测试算法寻找连续前沿面的能力.

2) DTLZ1, 由Deb[14]提出. 设自变量个数为n, 目标数为m, 函数形式为

$$\begin{aligned}&\min\ f(x),\\ &f_1(x) = (1 + g(X_M))\cdot\frac{1}{2}\cdot x_1 x_2 \cdots x_{m-1},\\ &f_2(x) = (1 + g(X_M))\cdot\frac{1}{2}\cdot x_1 x_2 \cdots (1 - x_{m-1}),\\ &\quad\vdots\\ &f_m(x) = (1 + g(X_M))\cdot\frac{1}{2}\cdot (1 - x_1),\\ &g(X_M) = 100\Big[|X_M| + \sum (x_i - 0.5)^2 - \\ &\qquad\cos(20\pi(x_i - 0.5))\Big],\\ &\text{s.t. } 0 \leqslant x_i \leqslant 1.\end{aligned} \tag{14}$$

其中: $g(X_M)$是差异函数, $M = n-m+1$是难度因子, X_M是最后M个决策变量. 该函数具有许多可控数量的局部最优. 理论非劣解$x_i = 0.5$, 理论帕雷托前沿是一个线性超平面: $\sum_{k=1}^{m} f_k^* = 0.5$. 本文研究中, 令$n = 7, m = 3$. 用此函数测试算法跳出局部最优, 寻找全局最优的能力.

3) KUR, 由Kursawe[15]提出, 是不连续凹函数. 函数形式为

$$\begin{aligned}&\min\ (f_1(x), f_2(x)),\\ &f_1(x) = \sum_{i=1}^{2}(-10\mathrm{e}^{-0.2\sqrt{x_i^2 + x_{i+1}^2}}),\\ &f_2(x) = \sum_{i=1}^{3}(|x_i|^{0.8} + 5\sin(x_i)^3),\\ &\text{s.t. } -5 \leqslant x_1, x_2, x_3 \leqslant 5.\end{aligned} \tag{15}$$

本文用此函数测试算法寻找非连续前沿面的能力, 但是这个函数没有理论帕雷托前沿. 为了能够使用后面提到的测度指标, 分别用对照算法和所有改进算法计算KUR函数. 计算过程中迭代次数取50 000, 基本种群和精英解集规模都取100, 将所有算法得到的全部解再按照帕雷托支配原则进行比较挑选, 最终获得的解构成非支配解集合, 将该非支配解集合作为近似帕雷托前沿.

3.3 测度指标

1) 收敛性度量. 采用 Van 等[16]提议的当代距离指标 (GD)

$$\mathrm{GD}=\frac{1}{h}\Big(\sum_{i=1}^{h}d_i^2\Big)^{\frac{1}{2}},\tag{16}$$

其中 h 表示精英解中粒子的个数. 该值越小越好.

2) 均匀性度量. 使用 Schott[17]提出的间隔指标 (SP)

$$\mathrm{SP}=\sqrt{\frac{1}{h-1}\sum_{i=1}^{h}(\bar{d}-d_i)^2}.\tag{17}$$

SP 值越小, 表明解集的均匀性越好.

3.4 测试结果分析

DTLZ1 运行迭代步数为 4 000 步, ZDT1、KUR 迭代步数为 200 步, 基本粒子种群数和精英解集规模都为 100.

3.4.1 测试结果图示

首先借助图形对各个算法效果进行直观比较. 见图 1～图 4.

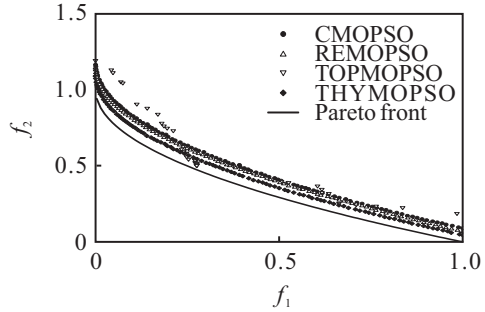

图 1　求解 ZDT1 结果

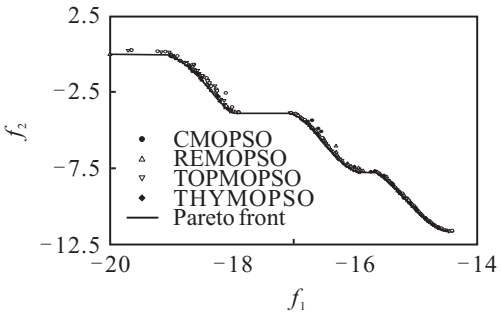

图 2　求解 KUR 结果

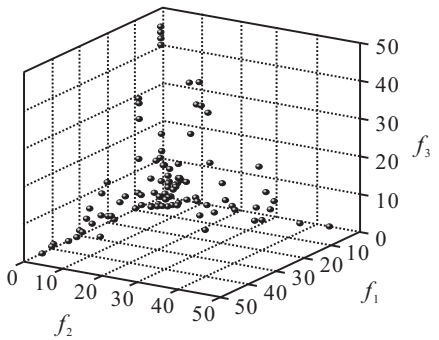

图 3　使用 CMOPSO 求解 DTLZ1 的结果

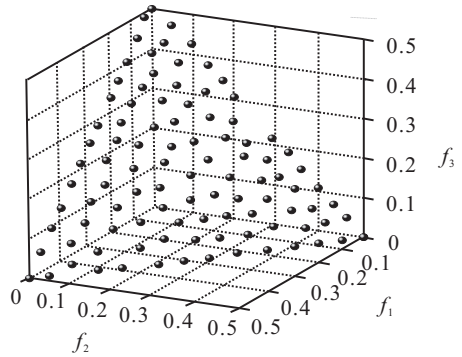

图 4　使用 THYMOPSO 求解 DTLZ1 的结果

图 1 是多个算法求解 ZDT1 函数的结果. 距离前沿面最远的是 CMOPSO, 求解最不均匀的是 TOPMOPSO, 但是距离真实前沿面最近的解也是由 TOPMOPSO 算法找到的, 可见 TOPSIS 策略在引导粒子群更快地靠近前沿面方面是有效的. 图 2 显示, 几种算法对于 KUR 函数的效果几乎相同. 因为 DTLZ1 函数是 3 个目标, 将多个算法运算结果放进同一图中比较是困难的. 图 3 显示的是对照算法 CMOPSO 求解 DTLZ1 函数一次的结果, 图 4 显示的是改进算法 THYMOPSO 求解 DTLZ1 函数一次的结果, 两个图比较说明同样运算步数时, 改进算法 THYMOPSO 获得的解质量明显好于对照算法 CMOPSO 获得的解质量.

3.4.2 测试结果的统计分析

图示只能显示每个算法一次的计算结果, 为了进一步明确改进策略的效果, 使用每个算法对每个测试函数分别测试 200 次, 计算每一次求解的 GD 值和 SP 值, 并进行统计分析. 见表 2～表 4.

表 2　各算法的 GD 值

测试函数		CMOPSO	REMOPSO	TOPMOPSO	THYMOPSO
ZDT1	max	0.266 97	0.178 64	0.182 32	0.129 45
	min	0.029 46	0.018 78	0.008 24	0.015 19
	Mean	0.077 15	0.041 76	0.031 98	0.025 63
DTLZ1	max	15.487 2	6.076 58	2.356 29	2.665 40
	min	0.009 07	0.005 79	0.011 20	0.003 99
	mean	3.468 51	1.434 23	0.500 71	0.325 38
KUR	max	0.005 25	0.003 57	0.005 76	0.004 68
	min	0.000 75	0.000 21	0.000 15	0.000 43
	mean	0.002 39	0.001 91	0.001 73	0.001 84

表 3　各算法的 SP 值

测试函数		CMOPSO	REMOPSO	TOPMOPSO	THYMOPSO
ZDT1	max	0.015 60	0.006 42	0.213 64	0.005 82
	min	0.007 18	0.001 12	0.013 32	0.000 79
	mean	0.010 54	0.003 76	0.063 20	0.003 00
DTLZ1	max	4.121 41	0.195 64	20.375 61	0.075 19
	min	0.006 94	0.003 82	0.096 50	0.003 25
	mean	0.489 69	0.077 65	1.998 02	0.031 45
KUR	max	0.004 66	0.005 55	0.159 82	0.005 52
	min	0.001 17	0.000 47	0.019 08	0.000 84
	mean	0.002 34	0.002 01	0.040 17	0.002 21

表 4 各算法使用的时间 s

测试函数		CMOPSO	REMOPSO	TOPMOPSO	THYMOPSO
ZDT1	max	0.065	0.072	0.066	0.072
	min	0.047	0.046	0.050	0.050
	mean	0.052	0.054	0.057	0.054
DTLZ1	max	8.514	8.956	9.071	9.164
	min	6.950	7.323	7.915	7.931
	mean	8.275	8.532	8.738	8.886
KUR	max	0.020	0.022	0.023	0.023
	min	0.011	0.013	0.013	0.014
	mean	0.013	0.014	0.014	0.016

通过对表 2～表 4 中的数据进行分析, 可以得到以下结论:

1) ZDT1 是连续函数, DTLZ1 是有很多局部最优的函数, 这两个函数的测试结论基本一致. 对照算法 CMOPSO 的 GD 值都是最差的, 改进算法 REMOPSO 的 SP 值和 GD 值都比 CMOPSO 好, 这说明基于 Pbest 的动态可行域变异策略有助于摆脱局部极值, 找到全局最优, 提高收敛度; 改进的 k 临近距离计算拥挤度的策略利于提高解分布的均匀性.

与 REMOPSO 比较, 含有用 TOPSIS 确定 Gbest 策略的算法 TOPMOPSO 和 THYMOPSO 的 GD 值都更小, 说明用 TOPSIS 方法确定 Gbest 可以引导粒子群更快地靠近前沿面.

所有算法中, TOPMOPSO 的 SP 值最大, 可见仅使用 TOPSIS 方法确定 Gbest 不利于解分布的均匀性. 对照算法 CMOPSO 的 SP 值好于 TOPMOPSO, 劣于其他算法, 说明使用改进的 k 临近距离计算拥挤度, 并以此确定 Gbest, 有助于提高解分布的均匀性, 可以弥补单独使用 TOPSIS 方法确定 Gbest 的策略产生的不足.

THYMOPSO 算法的 GD 值和 SP 值都是最好的, 说明混合使用拥挤度方法和 TOPSIS 方法确定 Gbest 的效果比单独使用其中一种方法更优, 同时也说明几种策略的综合应用能够做到优势互补, 尤其适用于类似 DTLZ1 这样有多个局部最优的多目标规划问题.

2) KUR 函数的测试结果显示了几个改进算法的 GD 值略优于对照算法. 在 SP 值方面, TOPMOPSO 算法的表现最差, 其余算法相差不多, 这说明对于类似于 KUR 函数这样的不连续前沿面的多目标规划问题, 本文提出的几种改进策略起到的作用有限.

表 4 显示了在相同运算环境和条件设置时, 几种算法运行花费时间差别不大, 这说明改进策略产生的额外时间消耗不大.

3.4.3 利用改进算法可以获得“理想”点

TOPMOPSO 算法和 THYMOPSO 算法的最后一步都可以得到一个基于 TOPSIS 策略的 Gbest 点, 这个点可以看成是一个“理想点”. 每种算法针对每个函数测试了 200 次, 每次都会得到一个理想点. 下面以 THYMOPSO 一次测试的结果为例.

1) ZDT1 函数获得的理想点为

$$X = (0.327\,594, 8.81\,\text{e-}07, 3.66\,\text{e-}13, 1.69\,\text{e-}06, 1.94\,\text{e-}13, 1.17\,\text{e-}10, 2.34\,\text{e-}14, 1.17\,\text{e-}07, 7.37\,\text{e-}12, 4.49\,\text{e-}08, 2.88\,\text{e-}06, 3.00\,\text{e-}07, 1.58\,\text{e-}08, 4.34\,\text{e-}07, 1.75\,\text{e-}13, 9.29\,\text{e-}13, 2.66\,\text{e-}15, 8.39\,\text{e-}09, 3.35\,\text{e-}07, 3.33\,\text{e-}06, 1.35\,\text{e-}13, 1.38\,\text{e-}06, 4.23\,\text{e-}06, 1.44\,\text{e-}12, 2.23\,\text{e-}08, 4.64\,\text{e-}11, 1.18\,\text{e-}11, 0, 9.84\,\text{e-}08, 1.13\,\text{e-}05);$$

目标值为

$$Y = (0.327\,594, 0.427\,648).$$

2) DTLZ1 函数的理想点为

$$X = (0.649\,742, 0.504\,555, 0.500\,006, 0.500\,026, 0.500\,034, 0.499\,842, 0.499\,926);$$

目标值为

$$Y = (0.164\,964, 0.161\,986, 0.176\,25).$$

3) KUR 函数获得的理想点为

$$X = (-1.121\,24, -1.081\,24, -1.135\,78);$$

目标值为

$$Y = (-14.631\,2, -11.406\,4).$$

结果显示, 这 3 个函数得到的理想点在各目标之间实现了一种较好的“均衡”. 对应到具体决策问题, 如果决策者没有特殊决策偏好, 则可以将相应理想点作为一个较合适推荐的方案.

4 结 论

本文提出了 3 种多目标粒子群算法的改进策略: 基于 Pbest 的动态可行域变异策略; 使用改进的 k 临近距离计算拥挤度; 使用 TOPSIS 方法确定 Gbest.

测试结果显示: 基于 Pbest 的变异策略有助于跳出局部极值点, 提高了算法收敛性; 改进的 k 邻近距离策略使算法解分布的均匀性更好; 仅使用 TOPSIS 方法确定 Gbest 的策略虽然能够使粒子群较快地靠近帕雷托前沿面, 但是最终解集的均匀性不理想.

包含所有改进策略的 THYMOPSO 算法不但在运算效果上优于对照算法和其他算法, 还可以为决策者提供一个“理想”方案. 因为包含 TOPSIS 策略的改进算法在最后一步获得的 Gbest 点是按照 TOPSIS 方法获得的目标值, 对应的自变量取值是按照 TOPSIS

准则获得的规划方案. 根据本文设定的迭代次数和精英解集的粒子数, 正、负理想解是根据几万或几十万个离散的方案确定的, 而最后的Gbest点就是在非劣解集中按照与正理想解和负理想解的距离找到的最“理想”的点, 适合成为向决策者推荐的解. 即, 通过使用包含本文提出的所有改进策略的多目标粒子群算法对多目标问题求解, 既可以为决策者提供很多非劣解构成的方案集合, 由决策者按照偏好进行选择, 又可以在决策者没有明确偏好时, 为其提供一个按照TOPSIS准则确定的最优方案.

参考文献(References)

[1] Baumgartner U, Magele C, Renhart W. Pareto optimality and particle swarm optimization[J]. IEEE Trans on Magnetics, 2004, 40(2): 1172-1175.

[2] Hu X, Eberhart R. Multi objective optimization using dynamic neighborhood particle swarm optimization[C]. Proc of the 2002 World on Congress on Computational Intelligence. Piscataway: IEEE Press, 2002: 1677-1681.

[3] Alvarez-Benitez J E, Everson R M, Fieldsend J E. A MOPSO algorithm based exclusively on Pareto dominance concepts[J]. Lecture Notes in Computer Science, 2005, 3410: 459-473.

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

[5] Knowles J D, Corne D W. Approximating the non-dominated front using the Pareto archived evolution strategy[J]. Evolutionary Computation, 2000, 8(2): 149-172.

[6] Moslemi H, Zandieh M. Comparisons of some improving strategies on MOPSO for multi-objective (r, Q) inventory system [J]. Expert Systems with Applications, 2011, 38(10): 12051-12057

[7] 胡广浩, 毛志忠, 何大阔. 基于两阶段领导的多目标粒子群优化算法[J]. 控制与决策, 2010, 25(3): 404-410.
(Hu G H, Mao Z Z, He D K. Multi-objective PSO optimization algorithm based on two stages-guided[J]. Control and Decision, 2010, 25(3): 404-410.)

[8] 吴亚丽, 徐丽青. 基于差分演化的改进多目标粒子群优化算法[J]. 系统仿真学报, 2011, 23(10): 2211-2215.
(Wu Y L, Xu L Q. Improved multi-objective particle swarm optimization based on differential evolution[J]. J of System Simulation, 2011, 23(10): 2211-2215.)

[9] Ching-Shih Tsou. Multi-objective inventory planning using MOPSO and TOPSIS[J]. Expert Systems with Applications, 2008, 35(1/2): 136-142.

[10] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory[C]. The 6th Int Symposium on Micro Machine and Human Science. Nagoya: IEEE, 1995: 39-43.

[11] Shi Y, Eberhart R C. A modified particle swarm optimize[C]. The IEEE Int Conf on Evolutionary Computation. Piscataway: IEEE Press, 1998: 69-73.

[12] Deb K, Mohan M, Mishra S. Evaluating the ε-Domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions[J]. Evolutionary Computation, 2005, 13(4): 501-525.

[13] Zitzler E, Deb K, Thiele L. Comparison of multi objective evolutionary algorithms: Empirical results[J]. Evolutionary Computation, 2000, 8(2): 173-195.

[14] Deb K. Multi-objective genetic algorithms: Problem difficulties and construction of test problems[J]. Evolutionary Computation, 1999, 7(3): 205-230.

[15] Kursawe Frank. A variant of evolution strategies for vector optimization[C]. Parallel Problem Solving from Nature. Berlin: Springer-Verlag, 1991: 193-197.

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

[17] Schott J. Fault tolerant design using single and multi criteria genetic algorithm optimization[D]. Cambridge: Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1995: 76-77.

(责任编辑：齐　霁)