文章编号: 1001-0920(2011)04-0540-08

基于 QPSO 和拥挤距离排序的多目标量子粒子群优化算法

施 展, 陈庆伟

(南京理工大学 自动化学院，南京 210094)

摘 要: 为了提高多目标优化算法的收敛性、分布性和减少算法的计算代价, 提出一种基于量子行为特性的粒子群优化 (QPSO) 和拥挤距离排序的多目标量子粒子群优化 (MOQPSO-CD) 算法. MOQPSO-CD 利用 QPSO 快速接近真实的 Pareto 最优解, 同时引入高斯变异算子以增强解的多样性. 采用拥挤距离排序的方法对外部存储器中最优解进行更新和维护, 使得从中选择的具有全局最优的领导粒子能够引导粒子群最终找到真实的 Pareto 最优解. 仿真结果表明, MOQPSO-CD 具有更好的收敛性和更均匀的分布性.

关键词: 多目标优化；量子行为特性粒子群优化；拥挤距离；Pareto 最优解

中图分类号: TP18 **文献标识码:** A

Multi-objective quantum-behaved particle swarm optimization algorithm based on QPSO and crowding distance sorting

SHI Zhan, *CHEN Qing-wei*

(School of Automation，Nanjing University of Science and Technology，Nanjing 210094，China. Correspondent: SHI Zhan，E-mail：z_shi2006@163.com)

Abstract: For improving the convergence and distribution together with less computation cost of multi-objective optimization algorithm, a multi-objective quantum-behaved particle swarm optimization based on QPSO and crowding distance sorting(MOQPSO-CD) algorithm is proposed. MOQPSO-CD makes full use of QPSO to approximate the true Pareto optimal solutions quickly, and Gaussian mutation operator is introduced to enhance the diversity of solution. MOQPSO-CD updates and maintains the archived optimal solutions based on crowding distance sorting technique, whose purpose is making the leader particles with global optimal ability guide the particle swarm finding the true Pareto optimal solutions finally. Simulation results show that MOQPSO-CD has better convergence and distribution.

Key words: multi-objective optimization；quantum-behaved particle swarm optimization；crowding distance；Pareto optimal solution

1 引 言

在很多实际优化问题中, 需要同时对多个目标进行优化, 而这些目标之间往往是相互矛盾的. 例如机器人路径规划中路径的长度、平滑性和安全性之间是相互冲突的. 多目标优化问题通常不存在唯一的最优解, 使得优化问题的每个目标同时最优, 因此需要对优化的目标进行协调和折中处理. 多目标进化算法是解决多目标优化问题的有效方法之一[1], 而多目标粒子群优化算法由于其良好的收敛性、简便的计算性和设置参数少的特点, 引起众多学者的广泛关注和研究, 近年来取得了一定的研究成果[2-11].

自 2002 年 Coello 和 Lechuga[3] 正式提出多目标粒子群优化算法以来, 人们借鉴多目标进化算法改进多目标粒子群优化算法的不足, 涌现出许多改进的算法[2], 如采用拥挤距离排序技术[4], 聚类技术[5], 小生境技术[6]和自适应网格法[7-8]等. 所有的改进算法都是为了解决两个关键问题: 1) 如何找到真实的 Pareto 解集; 2) 如何维护 Pareto 解的多样性.

经典粒子群优化 (PSO) 算法[12]存在局部搜索精度不高的缺陷. 与 PSO 相比, 量子行为特性的粒子群优化 (QPSO) 算法[13]的收敛速度更快, 寻优能力更强, 控制参数更少. QPSO 可以改进 PSO 局部搜索精度不

收稿日期: 2010-01-28；**修回日期:** 2010-04-21.

基金项目: 国家自然科学基金项目(60975075)；教育部高等学校博士学科点基金项目(20070288022)；江苏省自然科学基金项目(BK2008404).

作者简介: 施展(1984−), 男, 博士生, 从事智能控制、多目标优化理论与算法的研究；陈庆伟(1963−), 男, 教授, 博士生导师, 从事智能控制、网络控制系统等研究.

高的问题, 但目前 QPSO 绝大部分用于求解单目标优化问题[13], 而基于QPSO的多目标优化算法的研究成果极少[11,14]. 因为利用 QPSO 求解多目标优化问题时, 收敛速度快也表示算法容易早熟收敛, 过早地丧失解的多样性, 因此需要引入多样性维护机制来保持算法所找到的 Pareto 解的多样性, 从而最终找到优化问题的真实的 Pareto 最优解. 在维护解的多样性方面, 聚类技术的计算复杂度较高, 小生境技术的参数难以选定, 而自适应网格法的计算代价低且无需设置额外的参数, 但是当网格中包含的粒子数都等于 1 时, 全局最优粒子的选择将退化为随机搜索. 拥挤距离排序技术虽然计算复杂度较高, 但是全面地反映了粒子的密度信息和拥挤程度, 有利于裁剪冗余度大的粒子, 保留其他具有全局寻优能力的粒子, 从而更好地维护解的多样性, 这一点可以从文献 [4] 中得到佐证. 虽然基于拥挤距离排序的多目标粒子群优化 (MOPSO-CD) 算法[4]所需时间比基于自适应网格的多目标粒子群优化 (MOPSO) 算法[7]长, 但是在找到的 Pareto 最优解的性能上并不差于 MOPSO, 且在解的分布性上优于 MOPSO.

为了能够更快地找到多目标优化问题的真实 Pareto 最优解, 同时使得所求解在 Pareto 前沿上分布更加宽广和均匀, 本文提出一种新型的多目标优化方法 —— 基于 QPSO 和拥挤距离排序的多目标量子粒子群优化算法, 利用 QPSO 的寻优优势快速地接近真实的 Pareto 最优解, 同时引入高斯变异算子以增强搜索解的多样性. 采用拥挤距离排序的方法维护 Pareto 最优解的多样性, 使得从中选择的领导粒子能够引导粒子群最终找到真实的 Pareto 最优解.

2　多目标优化问题描述

下面给出多目标优化问题中几个常用的概念[15].

定义 1 (多目标优化问题)　不失一般性, 假设求解多目标最小化问题 (最大化问题可以通过取相反数或倒数转化为最小化问题), 则多目标优化问题的数学模型可描述如下:

$$\begin{cases} \text{Min } \boldsymbol{y} = \boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})). \\ \text{s.t. } g_i(\boldsymbol{x}) \leqslant 0,\ i = 1, 2, \cdots, h; \\ \quad \boldsymbol{x} \in \boldsymbol{X} \in \boldsymbol{R}^n,\ \boldsymbol{y} \in \boldsymbol{Y} \in \boldsymbol{R}^m. \end{cases} \tag{1}$$

其中: $\boldsymbol{x}$ 为决策向量, $\boldsymbol{y}$ 为目标向量; $\boldsymbol{X}$ 为决策向量 $\boldsymbol{x}$ 形成的决策空间, $\boldsymbol{Y}$ 为目标向量 $\boldsymbol{y}$ 形成的目标空间; f 为将 $\boldsymbol{x}$ 映射至目标向量空间的优化函数; $g_i(\boldsymbol{x}) \leqslant 0$ $(i = 1, 2, \cdots, h)$ 为 $\boldsymbol{x}$ 需满足的 h 个约束条件.

定义 2 (Pareto 支配)　设 $\boldsymbol{X}_f$ 为多目标优化问题的可行解集, $F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x}))$ 为目标向量, $\boldsymbol{x}_k \in \boldsymbol{X}_f$, $\boldsymbol{x}_l \in \boldsymbol{X}_f$. 称 $\boldsymbol{x}_k$ Pareto 支配 $\boldsymbol{x}_l$ (简称支配, 记作 $\boldsymbol{x}_k \prec \boldsymbol{x}_l$) 当且仅当

$$\begin{aligned} &\forall\ i \in 1, 2, \cdots, m:\ f_i(\boldsymbol{x}_k) \leqslant f_i(\boldsymbol{x}_l), \text{ and} \\ &\exists\ j \in 1, 2, \cdots, m:\ f_j(\boldsymbol{x}_k) < f_j(\boldsymbol{x}_l). \end{aligned} \tag{2}$$

定义 3 (Pareto 最优)　如果在某一集合中不存在任何其他解 $\boldsymbol{x}'$ Pareto 支配 $\boldsymbol{x}$, 则称 $\boldsymbol{x}$ 为该集合中的 Pareto 非支配解 (简称非支配解); 如果 $\boldsymbol{x}$ 为多目标优化问题整个可行解集中的 Pareto 非支配解, 则称 $\boldsymbol{x}$ 为该问题的 Pareto 最优解.

定义 4 (Pareto 最优前沿)　一个多目标优化问题所有 Pareto 最优解对应的目标向量集合, 构成了该问题的 Pareto 最优前沿.

根据定义 1 ~ 定义 4 可知: 类似于单目标优化中的最优解在多目标优化问题中是不存在的, 只存在 Pareto 最优解; 多目标优化问题通常具有很多个 Pareto 最优解, 而多目标优化问题的 Pareto 最优解仅是其一个可以接受的"满意解"; 提高 Pareto 最优解在任何一个目标上的性能, 都必然会导致它在其他至少一个目标上的性能降低.

3　量子行为特性的粒子群优化算法

文献 [13] 将量子行为的一些特性引入 PSO 的更新策略中, 基于量子 δ 势阱模型提出一种量子行为特性的粒子群优化 (QPSO) 算法. 与 PSO 相比, QPSO 仅有一个位移更新公式而没有速度更新公式. 其位移更新方程为

$$x_{i,j}(t+1) = p_{i,j} \pm \alpha |C_j(t) - x_{i,j}(t)| \ln\Big[\frac{1}{u_{i,j}(t)}\Big], \tag{3}$$

$$p_{i,j}(t) = \varphi_j(t) y_{i,j}(t) + [1 - \varphi_j(t)] \widehat{y}_j(t), \tag{4}$$

$$C(t) = (C_1(t), C_2(t), \cdots, C_N(t)) = \frac{1}{M}\sum_{i=1}^{M} y_i(t) = \Big(\frac{1}{M}\sum_{i=1}^{M} y_{i,1}(t), \cdots, \frac{1}{M}\sum_{i=1}^{M} y_{i,N}(t)\Big). \tag{5}$$

其中: $i (i = 1, 2, \cdots, M)$ 表示第 i 个粒子, M 为群体规模; $j (j = 1, 2, \cdots, N)$ 表示粒子的第 j 维, N 为搜索空间维数; t 为进化代数; $u_{i,j}(t)$ 和 $\varphi_j(t)$ 均为 [0,1] 区间上均匀分布的随机数; $\boldsymbol{x}_i(t)$, $\boldsymbol{y}_i(t)$ 和 $\boldsymbol{p}_i(t)$ 分别表示在进化代数为 t 时粒子 i 的当前位置、个体最好粒子位置和吸引子位置; $\widehat{\boldsymbol{y}}(t)$ 表示进化代数为 t 时群体最好位置; $\boldsymbol{C}(t)$ 表示粒子在进化代数为 t 时平均最好位置, 定义为所有粒子个体最好位置的平均; α 称为扩张–收缩因子, 是算法除群体规模和迭代次数以外的唯一的参数. 在 QPSO中, 单个粒子的收敛性将受 α 取值的影响, 文献 [13] 通过仿真实验指出, 单个粒子收敛到吸引子的充分必要条件是 $\alpha < 1.782$. 本文

α的取值随着进化代数的增加, 将逐渐从1.0减小到0.5, 利用α的动态变化来平衡探索与利用之间的关系.

4 一种新型多目标量子粒子群优化算法

4.1 算法原理

无论采用何种多目标优化方法, 总是希望多目标优化问题的最优解能够达到两种理想状态: 1) 收敛性好: 希望所求的Pareto最优解集与真实的Pareto最优解集在目标空间上尽可能地接近; 2) 分布均匀: 希望所求的Pareto最优解集均匀分布, 且分布范围尽可能的宽广. 但是这两种状态也是相互冲突的, 因为多样性的维护过程将降低收敛速度, 甚至引起所找到的Pareto最优解的退化.

QPSO以量子行为特性的粒子搜索解空间, 利用粒子群特有的记忆功能使其可以动态地跟踪当前搜索情况并调整搜索策略, 具有较强的全局搜索能力和鲁棒性. 因此, 本文采用QPSO作为构建新型多目标优化方法的基准算法. 利用QPSO求解多目标优化问题时, 收敛速度快也表示算法容易早熟收敛, 过早地丧失解的多样性, 所以QPSO在整个解空间搜索的同时, 为了增强种群的多样性, 引入变异算子来提高在寻优过程中解的多样性. 与此同时, 需要引入多样性维护机制以保持算法所找到的Pareto解的多样性, 从而最终找到优化问题的真实Pareto前沿.

在多目标粒子群优化方法中, 领导粒子的选择事关搜索到最终解的优劣, 因为领导粒子是从个体最好粒子中选择, 它代表一个群体的认知能力, 引领着群体的搜索方向. 本文设置一个外部存储器来存储搜索过程中找到的Pareto非劣解, 以期快速地接近真实的Pareto前沿. 借鉴第2代非支配排序遗传算法(NSGA-II)[16]的拥挤距离排序技术, 对外部存储器中的非劣解进行维护和更新操作, 以期从中选择的领导粒子具有全局搜索到真实Pareto最优解的能力. 据此, 本文基于QPSO和拥挤距离排序的多样性维护策略, 提出一种新型的量子行为特性的多目标量子粒子群优化(MOQPSO-CD)算法.

4.2 算法实施步骤

根据MOQPSO - CD原理, 本文设计MOQPSO - CD实现的具体步骤如下:

Step 1: 设置算法基本参数, 随机初始化粒子群中的所有粒子.

Step 2: 根据初始粒子群初始化外部存储器中的粒子.

Step 3: 对粒子群中的每个粒子选择各自的领导粒子, 根据QPSO位移更新方程更新粒子的位置. 判断是否满足变异条件. 若满足, 则进行高斯变异操作; 否则, 执行下一步.

Step 4: 评价更新后粒子的优劣, 并根据Pareto支配关系更新个体最好粒子和外部存储器最优粒子集. 如果更新后的粒子支配当前个体最好粒子或外部存储器最优粒子, 则将新粒子作为其个体最好粒子, 或存入外部存储器并删除被其支配的最优粒子; 如果两者互不支配, 则随机选择两者之一作为个体最好粒子, 或存入外部存储器并转Step 5; 否则舍弃更新后的粒子并转Step 6.

Step 5: 当外部存储器中粒子数已达最大容量时, 采用拥挤距离排序策略更新外部存储器中的粒子.

Step 6: 判断当前进化代数是否达到最大进化代数. 若达到, 则输出外部存储器中所有粒子作为最终解; 否则转Step 3.

4.3 MOQPSO-CD的具体实现

4.3.1 领导粒子的选择机制

为了维护搜索过程中解的多样性, 同时产生更好的分布性, 通过每一代对外部存储器最优粒子进行更新和维护操作, 实现外部存储器中粒子是当前为止产生的最优粒子; 而领导粒子是从外部存储器最优粒子中根据拥挤距离值选择的, 因此最优粒子的更新使得每一代的领导粒子并不完全相同, 从而实现领导粒子参与粒子的更新过程. 每个粒子的领导粒子的选择根据外部存储器中最优粒子的拥挤距离值确定. 首先需要计算外部存储器中所有最优粒子的拥挤距离值, 然后根据各自的拥挤距离值从大到小对所有最优粒子进行排序. 拥挤距离值越大, 说明该粒子越均匀地分布在目标空间, 被选择作为领导粒子的概率越大. 当被选为领导粒子的拥挤距离值有两个或两个以上相等时, 随机选择与之对应的最优粒子中的一个作为领导粒子; 否则, 直接被选为领导粒子.

4.3.2 高斯变异算子

尽管QPSO的寻优能力很强, 为了增加所求解的多样性, 本文引入变异算子来提高MOQPSO-CD在寻优过程中搜索解的多样性. 具体是以一种分段作用的方式对粒子群进行变异, 即在寻优的开始阶段作用于所有粒子; 在寻优的中期仅作用于部分粒子; 而在寻优的后期不再变异, 使得粒子能够在最终解区域进行精细搜索. 这样可以实现寻优粒子的开发和利用之间的折中. 文献[4]和[7]均采用随机的变异方式. 因为无规律的随机变异方式在进化过程可能出现退化现象, 而高斯变异具有很强的局部搜索能力, 有利于提高算法局部搜索精度, 所以本文采用具有高斯分布特性的随机方式进行变异.

4.3.3 外部存储器更新策略

对外部存储器的粒子进行更新时采用拥挤距离排序的方法. 拥挤距离排序方法首次出现在 NSGA-II 中, 它描述了某个最优解周围分布的最优解的密度. 文献 [4] 也采用该方法维护外部存储器中最优粒子的多样性. 外部存储器的更新需要满足以下两个条件中的任何一个: 1) 当产生的新粒子支配外部存储器的某个或某些粒子时; 2) 当外部存储器里的粒子数已达到最大容量时. 下面对拥挤距离计算方法作一简要说明, 具体实现可参考文献[16].

首先需要在每一个优化目标上按照目标函数值对外部存储器中的所有最优粒子作一个升序排列; 然后分别确定所求最优粒子在每一个优化目标空间上与之最近的最优粒子, 并计算这两个最近的最优粒子在各个优化目标空间上的距离; 最后将这些距离求和即得到所求最优粒子的拥挤距离. 为了更形象地说明拥挤距离的计算方法, 本文以两个目标的优化问题为例, 则第 i 个粒子的拥挤距离值 i_{dist} ($i_{\text{dist}} = i_{\text{dist}}^{F_1} + i_{\text{dist}}^{F_2}$) 计算过程如图 1 所示.

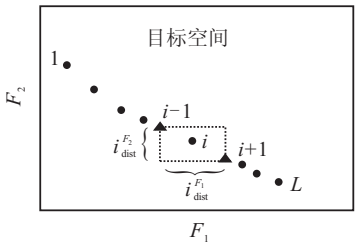


图 1　第 i 个粒子的拥挤距离值计算过程

当加入新粒子之前外部存储器的粒子数已达到最大容量时, 需要对加入新粒子后的外部存储器进行裁剪操作, 以保证粒子数在最大容量内; 否则, 仅执行添加新粒子的操作. 外部存储器中粒子裁剪操作步骤如下:

Step 1: 计算外部存储器中的所有最优粒子的拥挤距离值, 将所有的最优粒子根据各自的拥挤距离值作一个降序排列.

Step 2: 根据 Step 1 确定的拥挤距离值降序排列关系. 如果最小的拥挤距离值有两个或两个以上时, 则随机选择其中的一个最优粒子并移出外部存储器; 否则, 直接移出外部存储器.

Step 3: 判断外部存储器的容量是否超出最大值. 若超出, 则转 Step 1; 否则, 结束裁剪操作. 外部存储器采用拥挤距离排序法的裁剪操作进行自适应地动态更新与维护, 以便于后续领导粒子的选择.

4.3.4 算法的计算复杂度

设量子粒子群的规模为 N, 优化的目标个数为 n, 外部存储器的容量为 M, 则初始化粒子群并计算各粒子的目标函数值的计算复杂度为 $O(nN)$, 构造非支配解集的计算复杂度为 $O(nN)$, 拥挤距离排序的计算复杂度为 $O(nN^2)$, 从 N 个非支配解中选取 M 个非支配解的计算复杂度为 $O(nMN\log N)$. 因此在每一代运行中, 算法的计算复杂度为 $\text{CC} = n(O(N)+O(N)+O(N^2)+O(MN\log N))$. 于是 MOQPSO-CD 算法总的计算复杂度为 $G_{\max}\times$ CC.

5 仿真实验与结果分析

5.1 仿真实验设计

文献 [4] 研究表明, 虽然 MOPSO-CD 在计算时间上比 MOPSO[7] 长, 但算法的性能并不劣于 MOPSO, 特别是在 Pareto 最优解的分布性方面; 另外文献 [7] 的研究表明, MOPSO 的性能优于 NSGA-II 和 PAES. 因此为了更全面、更客观地评价所提算法的性能, 结合已有研究成果, 本文选取 MOPSO-CD[4], VEQPSO[11,14] 和 MOQPSO-CD 在标准测试函数 ZDT1, ZDT2, ZDT3, ZDT4, DTLZ1 和 DTLZ2 上进行对比分析[15,17]. 不同的标准测试函数具有不同的特点, 因此通过选择某些具有代表性的测试函数来验证多目标优化方法的寻优能力和算法的有效性. 本文所选的 6 个测试函数具有一定的代表性, 如非凸性、不连续性以及欺骗性. ZDT1, ZDT2, ZDT3 和 ZDT4 均为 2 目标的最小化问题. 其中: ZDT1 具有凸 Pareto 前沿, ZDT2 具有非凸 Pareto 前沿, ZDT3 具有 5 段不连续的凸 Pareto 前沿, ZDT4 具有 21^9 局部最优值来干扰全局 Pareto 前沿的搜索. DTLZ1 和 DTLZ2 的优化目标个数均取 3. DTLZ1 具有 7 个决策变量, 其真实 Pareto 前沿是一个具有 11^7-1 个局部 Pareto 最优的线性超平面, 且 3 个目标值均满足 $\sum_{i=1}^{3} f_i = 0.5$. DTLZ2 具有 12 个决策变量, 其真实 Pareto 前沿在第 1 象限的八分之一单位球面, 且 3 个目标值均满足 $\sum_{i=1}^{3} f_i^2 = 1$.

根据文献 [18] 的分析, 为了更全面、客观地评价不同算法的性能, 本文除了选取所求解的收敛性 (GD), 分布性 (SP), 正确率 (ER)[7] 和各算法运算时间 (time/s) 这 4 个性能指标外, 还增加了支配覆盖率 $C(X_1, X_2)$ 和超体积覆盖率 $H(X_1, X_2)$ 两个性能指标[19] 进行评价, 最终得出不同算法之间的优劣关系. 对于每一个测试函数, 每种算法均独立运行 30 次, 然后对所得结果作一个统计平均. 为了使所得结果具有可比性, 计算机为 Pentium 4, 2.80 GHz, 512 MB, DDR 内存. 3 种算法均在 VC++6.0 环境下运行.

5.2 仿真结果与分析

根据设计的仿真实验, MOPSO-CD, VEQPSO和MOQPSO-CD参数设置如表1所示, VEQPSO算法中种群的个数与优化的目标个数一致. 为了评价的公平性, 3种算法的粒子数(VEQPSO算法为每个子种群的粒子数)均取100, 进化代数均为200, 外部存储器容量均为100.

表 1 不同算法的参数设置

算 法	粒子个数	外部存储器容量	迭代次数	种群个数	变异概率
MOPSO-CD	100	100	200	1	0.5
VEQPSO	100	100	200	2或3	0
MOQPSO-CD	100	100	200	1	0.1

某一次在ZDT1, ZDT2, ZDT3, ZDT4, DTLZ1和DTLZ2测试函数上所求的Pareto最优前沿分别如图2和图3所示. GD, SP, ER, Time, $C(X_1, X_2)$和$H(X_1, X_2)$的性能指标值均为30次运行结果的统计平均值, 分别见表2～表5.

在评价一个多目标优化算法是否优于其他算法时, 最关键的指标是该算法能否找到真实的Pareto前沿. 在保证能够全部或绝大部分找到真实的Pareto前沿的条件下, 对所求解的收敛性和分布性进行比较, 然后对算法消耗时间等其他指标进行比较, 最终得出算法的评价结果. 本文对不同算法的评价将遵循这一原则, 即ER的优先级最高, GD和SP次之, time最低. 其实ER值与GD值是相互关联的, ER的

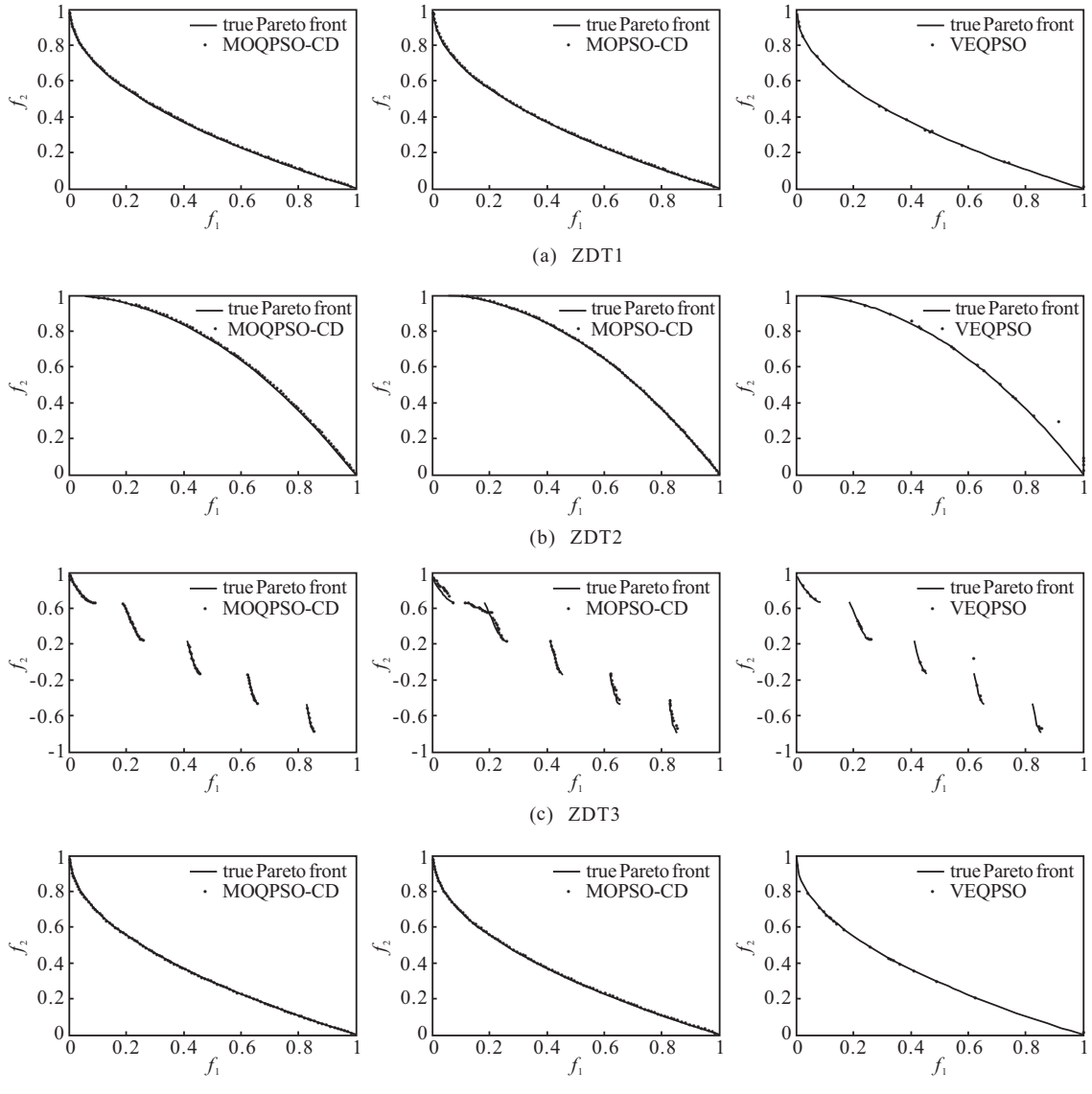
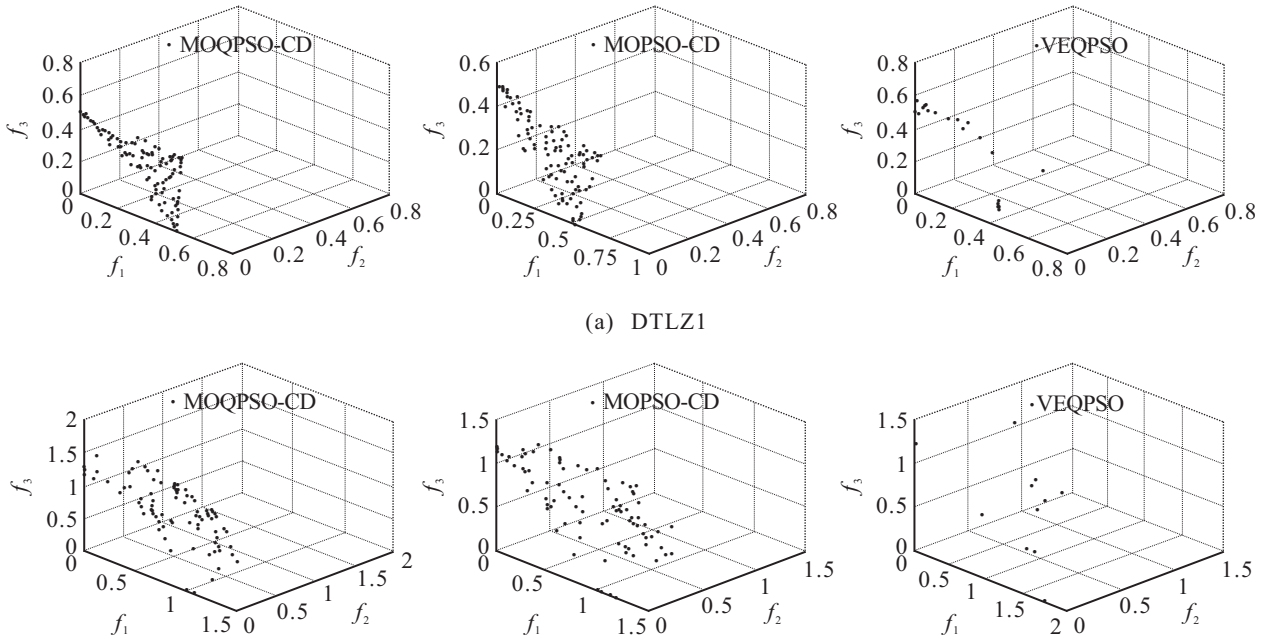


图 2 3种算法分别在ZDT1～ZDT4上所求的Pareto前沿



(a) DTLZ1

(b) DTLZ2

图 3　3 种算法分别在 DTLZ1 和 DTLZ2 上所求的 Pareto 前沿

表 2　MOQPSO-CD, MOPSO-CD 和 VEQPSO (分别用 A, B 和 C 表示) 在 ZDT 和 DTLZ 系列测试函数上性能指标值

测试函数	GD			SP			ER			time/s		
	A	B	C	A	B	C	A	B	C	A	B	C
ZDT1	0.000 851	**0.000 766**	0.011 137	**0.006 928**	0.007 214	0.233 727	0.080 000	**0.060 000**	0.428 571	18.328 000	25.359 000	**0.390 000**
ZDT2	0.000 624	**0.000 547**	0.004 641	**0.006 807**	0.007 630	0.195 610	**0.010 000**	**0.010 000**	0.625 000	17.266 000	24.687 000	0.359 000
ZDT3	**0.001 658**	0.003 275	0.004 394	**0.009 550**	0.060 090	0.102 894	**0.000 409**	0.777 778	0.666 667	1.625 000	1.203 000	**0.453 000**
ZDT4	0.000 886	**0.000 810**	0.007 499	0.007 476	**0.006 296**	0.314 877	0.110 000	**0.090 000**	0.600 000	1.313 000	8.578 000	**0.234 000**
DTLZ1	**0.000 823**	0.001 142	2.516 432	0.020 041	**0.018 880**	0.897 421	**0.070 000**	0.160 000	1.000 000	9.953 000	11.750 000	**0.265 000**
DTLZ2	**0.013 266**	0.021 781	1.378 249	**0.073 072**	0.079 595	0.481 882	**0.465 517**	0.655 056	1.000 000	0.812 000	0.625 000	**0.171 000**

表 3　MOQPSO-CD 和 MOPSO-CD 所需时间对比

算　法	ZDT1	ZDT2	ZDT3	ZDT4	DTLZ1	DTLZ2
MOPSO-CD	25.359	24.687	1.203	8.578	11.750	0.625
MOQPSO-CD	18.328	17.266	1.625	1.313	9.953	0.812
后前对比/%	**−27.73** ↓	**−30.06** ↓	35.08 ↑	**−84.69** ↓	**−15.29** ↓	29.92 ↑

表 4　MOQPSO-CD, MOPSO-CD 和 VEQPSO (分别用 A, B 和 C 表示) 在 *C*(*, *) 性能指标上的比较

测试函数	C(A, B)	C(B, A)	C(A, C)	C(C, A)
ZDT1	0.000 000	0.140 000	**0.125 000**	**0.030 000**
ZDT2	0.000 000	0.380 000	**0.666 667**	**0.040 000**
ZDT3	**0.566 667**	**0.351 064**	**0.200 000**	**0.040 000**
ZDT4	0.010 000	0.051 020	**0.250 000**	**0.010 204**
DTLZ1	**0.101 010**	**0.000 000**	**1.000 000**	**0.000 000**
DTLZ2	**0.328 767**	**0.160 494**	**0.500 000**	**0.041 096**

表 5　MOQPSO-CD, MOPSO-CD 和 VEQPSO (分别用 A, B 和 C 表示) 在 *H*(*, *) 性能指标上的比较

测试函数	H(A, B)	H(B, A)	H(A, C)	H(C, A)
ZDT1	0.001 416	0.002 589	**0.101 570**	**0.000 266**
ZDT2	**0.004 037**	**0.000 631**	**0.157 715**	**0.006 912**
ZDT3	**0.014 645**	**0.001 687**	**0.157 770**	**0.089 233**
ZDT4	0.001 856	0.001 920	**0.172 390**	**0.000 521**
DTLZ1	**0.001 991**	**0.001 630**	**0.816 357**	**0.000 000**
DTLZ2	**0.075 847**	**0.049 756**	**0.644 085**	**0.000 000**

值越小, GD 的值也越小, 它们从不同的角度反映算法的收敛性. GD, SP, ER 和 time 都是越小越好. 表 2 和表 3 中粗体部分表示在某一性能指标下的最优值. $C(X_1, X_2)$ 和 $H(X_1, X_2)$ 需要比较 X_1 和 X_2 不同次序时相对大小, 都是越大越好. 表 4 和表 5 中的粗体部分表示前者大于后者, 即前一算法相对较优.

通过对比分析表 2 中数据并结合图 2 和图 3 可知: 从解的收敛性(GD 值)方面比较, 对于 ZDT1, ZDT2 和 ZDT4 而言, MOQPSO-CD 和 MOPSO-CD 都能找到对应测试函数的真实 Pareto 前沿, 两者之间

没有多少区别; 但是在ZDT3, DTLZ1和DTLZ2上, MOQPSO-CD能够找到MOPSO-CD所找不到的真实Pareto前沿, 所以MOQPSO-CD要优于MOPSO-CD. 尽管VEQPSO在ZDT1和ZDT4上能找到测试函数的真实Pareto最优解, 但所求解只是真实Pareto最优解的一部分; 在ZDT2和ZDT3上除了找到部分真实的Pareto最优解外, 还找到了一些局部最优解和较差的解; 而对于DTLZ1和DTLZ2而言, 几乎找不到测试函数的真实Pareto最优解, 导致VEQPSO的ER值较大. 因此, VEQPSO的性能远不如MOQPSO-CD和MOPSO-CD, 这一点可从表2中VEQPSO的ER和GD值得到佐证. 在解的分布性(SP值)方面, MOQPSO-CD除在ZDT4和DTLZ1上稍逊于MOPSO-CD外, 在其他4个测试函数上均优于MOPSO-CD; 同时, MOQPSO-CD和MOPSO-CD均优于VEQPSO. 在运算时间(time值)方面, VEQPSO时间最短, MOQPSO-CD次之, MOPSO-CD最长(除ZDT3和DTLZ2外). MOPSO-CD和MOQPSO-CD所需时间对比分析见表3.

从表4中$C(*,*)$的相对大小可知: MOQPSO-CD在ZDT3, DTLZ1和DTLZ2上均优于MOPSO-CD, 但在ZDT1, ZDT2和ZDT4上则劣于MOQPSO-CD; MOQPSO-CD在所有测试函数上均优于VEQPSO.

从表5中$H(*,*)$的相对大小可知: MOQPSO-CD除在ZDT1和ZDT4上劣于MOPSO-CD外, 在其他4个测试函数上均优于MOPSO-CD; MOQPSO-CD在所有测试函数上均优于VEQPSO.

关于算法参数对算法性能的影响, 本文仅从仿真结果进行比较分析和说明. 首先MOQPSO-CD算法是基于QPSO构造的, 而文献[13]通过仿真实验指出: 只要QPSO算法除群体规模和迭代次数以外的唯一参数——扩张收缩因子α满足$\alpha < 1.782$, 则单个粒子将收敛于全局最优解, 因此本文不对该参数进行讨论.

对于其他参数, 如粒子群的规模和进化代数对算法性能的影响: 当粒子群规模为50, 进化代数为200时所得结果基本接近, 但不如粒子群规模为100, 进化代数为200时所得结果. 随着粒子群规模从50增加到100, 所得结果越来越接近, 说明本文所提MOQPSO-CD在粒子群规模较小时也具备较强的寻优能力. 对于进化代数的选取, 则是兼顾了MOPSO-CD和VEQPSO算法的性能. 因为当进化代数较小时, 这两种算法没有充分利用其寻优能力, 导致算法性能不佳. 为了与已有结果进行比较(特别是文献[4])以及算法比较的公平性, 本文选取粒子群规模为100, 进化代数为200. 变异概率则根据算法取不同变异概率所得结果的优劣进行选择. 如果变异概率大, 则导致搜索空间范围较大, 虽然能够提高算法所求解的分布性, 但是延长了算法的收敛速度; 如果变异概率太小, 则导致搜索空间范围较小, 虽然能够提高收敛速度, 但同时容易丧失多样性, 从而陷入局部最优. 综合考虑分布性和收敛速度, 本文所选的变异概率能够较好地平衡这两方面的矛盾.

综上所述, 与MOPSO-CD相比, 尽管MOQPSO-CD在个别测试函数上的分布性不如MOPSO-CD, 但是MOQPSO-CD能够找到所有测试函数的绝大部分真实Pareto前沿, 且所求解的分布性均匀; 而MOPSO-CD在一些测试函数上会陷入局部最优, 没有找到真实的Pareto前沿. 另外, MOQPSO-CD所需时间比MOPSO-CD更短. 据此得出这3种算法之间的优劣关系是: MOQPSO-CD的性能最优, MOPSO-CD次之, VEQPSO的性能最差.

6 结 论

通过理论分析及仿真结果, 本文认为VEQPSO的性能差主要有两方面的原因: 1) 算法采用每个优化目标利用一个子种群来寻优, 然后通过在某一时刻交互各子种群的最优解, 导致每个种群只追求所优化目标的最优解, 而将能够较好均衡不同优化目标的折中解淘汰掉; 2) VEQPSO没有额外提高搜索解的多样性的机制, 使得搜索过程过早地丧失了解的多样性. 本文所提出的基于QPSO和拥挤距离排序的多目标量子粒子群优化(MOQPSO-CD)算法解决了VEQPSO存在的问题. 仿真结果表明, MOQPSO-CD所求解的收敛性能和分布性能得到了进一步提高, 整体性能优于MOPSO-CD, 更优于VEQPSO, 因此MOQPSO-CD是一种整体性能更优、鲁棒性更强的多目标优化方法. 另外还表明了QPSO应用于求解多目标优化问题的可行性, 为多目标优化理论与算法的应用提供了一种新的思路.

参考文献(References)

[1] Deb K. Multi-objective optimization using evolutionary algorithms[M]. Chichester: Wiley, 2001.

[2] Sierra M R, 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.

[3] Coello C A C, Lechuga M S. MOPSO: A proposal for multiple objective particle swarm optimization[C]. IEEE Congress on Evolutionary Computation. Piscataway: IEEE Press, 2002: 1051-1056.

[4] Raquel C R, Jr P C N. An effective use of crowding distance in multiobjective particle swarm optimization[C].

Proc of the 2005 Workshops on Genetic and Evolutionary Computation. Washington: ACM Press, 2005: 257-264.

[5] Pulido G T, Coello C A C. Using clustering technique to improve the performance of a multi-objective particle swarm optimizer[J]. Lecture Notes on Computer Science, 2004, 3102: 225-237.

[6] Lechuga M S, Rowe J. Particle swarm optimization and fitness sharing to solve multi-objective optimization problems[C]. IEEE Congress on Evolutionary Computation. Edinburgh: IEEE Press, 2005: 1204-1211.

[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] 杨俊杰, 周建中, 方仍存, 等. 基于自适应网格的多目标粒子群优化算法[J]. 系统仿真学报, 2008, 20(21): 5843-5847.
(Yang J J, Zhou J Z, Fang R C, et al. Multi-objective particle swarm optimization based on adaptive grid algorithms[J]. J of System Simulation, 2008, 20(21): 5843-5847.)

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

[10] Li Z Y, Xu K, Liu S B, et al. Quantum multi-objective evolutionary algorithm with particle swarm optimization method[C]. The 4th Int Conf on Natural Computation. Ji'nan, 2008: 672-676.

[11] Omkar S N, Khandelwal R, Ananth T V S, et al. Quantum behaved particle swarm optimization(QPSO) for multi-objective design optimization of composite structures[J]. Expert Systems with Applications, 2009, 36(8): 11312-11322.

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

[13] 孙俊. 量子行为粒子群优化算法研究[D]. 无锡: 江南大学信息工程学院, 2009.
(Sun J. Particle swarm optimization with particles having quantum behavior[D]. Wuxi: School of Information Technology, Southern Yangtze University, 2009.)

[14] 沈佳宁. 基于QPSO算法求解多目标优化问题及其应用[D]. 无锡: 江南大学信息工程学院, 2008.
(Shen J N. Solving multi-objective problem based on QPSO algorithm[D]. Wuxi: School of Information Technology, Southern Yangtze University, 2008.)

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

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

[17] Deb K, Thiele L, Laumanns M, et al. Scalable test problems for evolutionary multi-objective optimization[R]. Switzerland: Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology Zurich, 2001: 1-27.

[18] Zitzler E, Thiele L, Laumanns M, et al. Performance assessment of multiobjective optimizers: An analysis and review[J]. IEEE Trans on Evolutionary Computation, 2003, 7(2): 117-132.

[19] Zitzler E. Evolutionary algorithms for multiobjective optimization methods and applications[D]. Switzerland: Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology Zurich, 1999.

(上接第539页)

[8] Zhu Xia, Li Xiaoping, Wang Qian. Objective increment based meta heuristic for total flow time minimization in no-wait flowshops[J]. J of Southeast University(English Edition), 2008, 24(2): 168-173.

[9] 王磊, 黄文奇. 求解工件车间调度问题的一种新的邻域搜索算法[J]. 计算机学报, 2005, 28(5): 809-815.
(Wang L, Huang W Q. A new local search algorithm for job shop scheduling problem[J]. Chinese J of Computers, 2005, 28(5): 809-815.)

[10] Rajendran C, Ziegler H. An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs[J]. European J of Operations Research, 1997, 103(1): 129-138.