doi:10.6041/j.issn.1000-1298.2015.04.047

基于分解机制的多目标蝙蝠算法*

王亚辉[1]　贾晨辉[2]　赵仁鹏[1]

(1.华北水利水电大学机械学院,郑州 450011;2.河南科技大学机电工程学院,洛阳 471023)

摘要:在分析蝙蝠算法性能基础上,将蝙蝠算法融入分解机制,提出了一种基于分解机制的多目标蝙蝠算法。为了进一步提高算法的多样性,将差分进化策略引入算法中。对 14 个具有复杂 Pareto 前沿的多目标优化问题(LZ-09 系列和 ZDT 系列)测试不同邻域规模对算法性能的影响,结果表明新算法的邻域规模为 20 时性能最优;将其与 MOEA/D-DE 和 NSGA-II 算法进行对比分析,结果显示该算法的分布性、收敛性和多样性均优于另外两种算法。为了验证其求解含有约束问题的性能,将其应用于滑动轴承多目标优化设计问题中,获得的 Pareto 前沿分布均匀,表明算法具有工程实用性,是求解复杂高维多目标问题的有效方法。

关键词:蝙蝠算法　分解机制　差分进化　滑动轴承　多目标优化

中图分类号:TP301.6　　**文献标识码**:A　　**文章编号**:1000-1298(2015)04-0316-09

Multi-objective Bat Algorithm Based on Decomposition

Wang Yahui[1]　Jia Chenhui[2]　Zhao Renpeng[1]

(1. *College of Mechanical Engineering*, *North China University of Water Resources and Electric*, *Zhengzhou* 450011, *China*
2. *College of Mechatronics Engineering*, *Henan University of Science and Technology*, *Luoyang* 471023, *China*)

Abstract: The bat algorithm was integrated into decomposition mechanism on the basis of its evaluation and a multi-objective bat algorithm based on decomposition (MOBA/D) was proposed. In order to improve the algorithm diversity, the differential evolutionary strategy was introduced into MOBA/D. The performances of MOBA/D on 14 multi-objective optimization problems were tested, which included family benchmark functions of LZ-09 and ZDT with different neighborhood scales effect on the performance of the algorithm. The result indicated that MOBA/D had the best performance with neighborhood size of 20. Compared with MOEA/D-DE and NSGA-II, the simulation results showed that MOBA/D can obtain a more uniform distribution of Pareto solution set and better convergence as well as diversity than those of state-of-the-art multi-objective metaheuristics. For further performance analysis of MOBA/D on constraint problem, the optimization design of sliding bearing was solved to demonstrate the feasibility and effectiveness. The good performance on convergence and diversity of the obtained Pareto set demonstrated that MOBA/D was suitable for engineering practice, which was an effective way for solving complex and high dimensional multi-objective optimization problems.

Key words: Bat algorithm　Decomposition mechanism　Differential evolution　Sliding bearing　Multi-objective optimization

引言

在科学研究与工程实践中,存在着大量的多目标优化问题(Multi-objective optimization problem, MOP),如结构优化、车间调度、方案优选等。在过去近 10 年中,国际上出现了许多典型的多目标进化算法(Multi-objective evolutionary algorithm, MOEA),如 NSGA-II[1]、SPEA2[2]、MOCell[3]、HypE[4]、

收稿日期:2014-12-18　修回日期:2015-01-23

*国家自然科学基金资助项目(51475142)

作者简介:王亚辉,副教授,主要从事先进制造技术研究,E-mail: wangyahui@ncwu.edu.cn

MOEA/D[5]等。文献[6]将这些算法按照其所采用的基本思想分为：①基于 Pareto 占优关系的 MOEA 算法。②基于性能指标的 MOEA 算法。③基于分解机制的 MOEA 算法。基于分解机制的 MOEA 算法是将 MOP 分解为一系列子问题来求解，并将这些子问题有效地组织起来求解而获得 Pareto 解集的一个逼近，由于算法只运行一次，所以能获得较高的计算效率。基于分解机制的 MOEA 算法框架能容纳各类单目标优化方法和局部搜索方法，可以在启发式算法与传统数学规划方法之间建立起联系。

2010 年 Yang 提出了新型启发式算法，蝙蝠算法(Bat algorithm，BA)。研究表明[7-8]，蝙蝠算法参数少、收敛速度、收敛精度高，相比于粒子群算法、遗传算法和和声算法等，具有更大的潜能。为了发挥蝙蝠算法的特点，将其与分解机制相结合，并应用于多目标优化问题求解，本文提出一种基于分解机制的多目标蝙蝠算法(Multi-objective bat algorithm based on decomposition，MOBA/D)。为了验证其求解含有约束问题的性能，将其应用于动压滑动轴承优化设计中。

1　多目标基本概念

对于多目标优化问题，其至少有 2 个目标且相互冲突，任何一个目标性能的提升必然会引起至少一个目标性能的下降。因此，求解多目标优化问题的目的不在于找到单个最优解，而是一组相互折中的解，称之为 Pareto 最优解集(Pareto solution，PS)[9]，其对应的目标函数值称之为 Pareto 前端(Pareto front，PF)。不失一般性，一个具有 n 个决策变量 m 个目标的多目标问题可以定义为

$$\min F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})) \quad (1)$$
$$\text{subject to } \boldsymbol{x} \in \Omega$$

其中　$\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$

式中　$\boldsymbol{x}$——决策向量　　Ω——决策空间

Pareto 的相关定义如下：

Pareto 支配：假设 $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ 和 $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$ 是决策空间内的两个可行解向量，如果有 $\forall i \in \{1,2,\cdots,m\}$ 使得 $f_i(\boldsymbol{x}) \leqslant f_i(\boldsymbol{y})$ 且 $\exists j \in \{1,2,\cdots,m\}$ 使得 $f_j(\boldsymbol{x}) < f_j(\boldsymbol{y})$，则称 $\boldsymbol{x}$ 支配 $\boldsymbol{y}$，记作 $\boldsymbol{x} \prec \boldsymbol{y}$。

Pareto 最优解集：在决策空间 Ω 内，$\neg\ \exists \boldsymbol{y} \prec \boldsymbol{x}$，则 $\boldsymbol{x}$ 为决策空间内的一个 Pareto 最优解，Pareto 最优解集为决策空间 Ω 内所有 Pareto 最优解的集合，即 $PS = \{x \mid \neg\ \exists \boldsymbol{y} \in \Omega : \boldsymbol{y} \prec \boldsymbol{x}\}$。

Pareto 前端：Pareto 最优解集在目标空间内的映射称之为 Pareto 前端，即 $PF = \{F(\boldsymbol{x}) \mid \boldsymbol{x} \in PS\}$。

2　基于分解机制的多目标蝙蝠算法

2.1　蝙蝠算法

蝙蝠拥有强大的回声定位能力，在飞行过程中通过发射不同频率与响度的声波不断改变自身的位置，从而捕获猎物。在蝙蝠算法中，使用一些理想化的规则来模拟这一过程：蝙蝠以速度 v、位置 x 和脉冲频率 f 在空间随机飞行，通过判断与猎物的接近程度来调整脉冲频率，蝙蝠的位置用来模拟优化问题中的决策向量，频率的改善用来模拟搜索的步长，用捕捉到的猎物来模拟优化问题中的最优解。

实际仿真中，在一个 d 维的搜索空间中，蝙蝠在时刻 t 的位移和速度为

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \quad (2)$$
$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*) f_i \quad (3)$$
$$x_i^t = x_i^{t-1} + v_i^t \quad (4)$$

式中　f_i——搜索频率

$f_{\min}$、$f_{\max}$——最小、最大搜索频率

β——[0,1]内均匀分布的随机变量

x^*——当前全局最优值

在搜索过程中可以通过不断改变频率来调整搜索步长。对于局部搜索，是通过对当前最优解产生扰动来实现的，即对当前最优解进行随机移动而产生，即

$$x_{new} = x_{old} + \varepsilon A^t \quad (5)$$

式中　ε——[-1,1]的随机数

A^t——所有蝙蝠在同一时刻的平均音量

式(3)、(4)的更新方式与粒子群算法有些类似，因此某种程度上，蝙蝠算法可视为标准粒子群和局部搜索的结合。

在现实中，蝙蝠找到猎物时会通过降低音量提高脉冲频率来进行进一步搜索。在 BA 中，全局搜索和局部寻优之间的平衡是通过改变音量 A 和脉冲发生率 r_i 来实现的，即

$$A_i^{t+1} = \alpha A_i^t \quad (6)$$
$$r_i^{t+1} = r_i^0 [1 - \exp(-\gamma t)] \quad (7)$$

式中　α、γ——常量

在迭代过程中，当 $0 < \alpha < 1$，$\gamma > 0$ 时，$A_i^t \to 0$，$r_i^t \to r_i^0$。

蝙蝠算法的伪代码如下：

```
Proc steps Up( )
Initialize population x_i (i = 1,2,…,n) and v_i
Define pulse frequency f_i at x_i
While ! Stop Condition( )
for i←1 to popSize
```

x_{new}←exploration(x_i, x^*) // 全局搜索,式(2)~(4)
if (rand 1 > r_i)
  x_{new}←exploitation(x^*) // 局部寻优,式(5)
end if
if (rand 2 < A_i && $f(x_{new}) < f(x^*)$)
  $x_i = x_{new}$ // 替换操作
  Increase r_i and reduce A_i // 更新操作,式(6)~(7)
  end if
end for
  x^*←Rank(pop) // 找出最优解 x^*
End while

2.2 分解机制

分解机制[5,10]是 Zhang 在 MOEA/D-DE 中提出的一种求解多目标问题的策略,其将 MOP 分解为一系列的子问题,然后利用单目标进化算法来求解每个子问题,其中每个子问题的目标值是各个目标分量的聚合函数,每个子问题对应一个权重向量,通过权重向量来获取每个子问题的邻居,每个子问题的优化被限定为只能通过与其邻域内的子问题来完成进化操作。

在 MOEA/D-DE 中,常用的分解方法有权重向量法[11]、切比雪夫法[11]以及边界插入法[11-13],其中以切比雪夫法应用最为广泛,其分解机制为

$$\min g^{te}(\boldsymbol{x}|\boldsymbol{\lambda},\boldsymbol{z}^*) = \max_{1\leqslant i\leqslant m}\{\boldsymbol{\lambda}_i|f_i(\boldsymbol{x}) - z_i^*|\} \quad (8)$$
$$\text{subject to } \boldsymbol{x}\in\Omega$$

其中,$\boldsymbol{z}^* = (z_1^*, z_2^*, \cdots, z_m^*)$为最优参考点,通常很难获取精确的最优参考点,一般利用 $z_i^* = \min\{f_i(\boldsymbol{x})|\boldsymbol{x}\in\Omega\}$,$i = 1,2,\cdots,m$ 来作为最优参考点的近似参考点。对于 MOP 中的每一个 Pareto 最优解 x^*,一定存在一个对应的权重向量 $\boldsymbol{\lambda}$,使得 x^* 对应单目标问题 $g^{te}(\boldsymbol{x}|\boldsymbol{\lambda},\boldsymbol{z}^*)$ 的一个最优解。反之,单目标问题 $g^{te}(\boldsymbol{x}|\boldsymbol{\lambda},\boldsymbol{z}^*)$ 的每个最优解也对应 MOP 的一个 Pareto 最优解,因此可以通过改变权重向量来获得 Pareto 最优解集[14]。

设种群规模为 N,目标个数为 m,$\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2, \cdots, \boldsymbol{\lambda}^N\}$为一个权重向量集合,其中 $\boldsymbol{\lambda}^i = \{\lambda_1^i, \lambda_2^i, \cdots, \lambda_m^i\}$且满足 $\sum_{j=1}^{m}\lambda_j^i = 1$,则第 i 个子问题的表达式为

$$g^{te}(\boldsymbol{x}|\boldsymbol{\lambda}^i,\boldsymbol{z}^*) = \max_{1\leqslant j\leqslant m}\{\lambda_j^i|f_j(\boldsymbol{x}) - z_j^*|\} \quad (9)$$

每个子问题对应一个权重向量,子问题的邻居是通过计算每个权重向量与其欧氏距离最小的 T 个权重向量来确定的。每一代种群由每个子问题的当前最优解构成,对每个子问题的进化操作被限制在邻居内进行。在每一代 t,采用切比雪夫机制的 MOEA/D-DE 保存的个体信息方式有:①组成群体的 N 个点:$x^1, x^2, \cdots, x^N \in \Omega$,其中 x^i 为子问题 i 的当前最优解。②$FV^1, FV^2, \cdots, FV^N$,其中 $FV^i = F(x^i)$,$i = 1,2,\cdots,N$。③$\boldsymbol{z} = (z_1, z_2, \cdots, z_m)$,其中 z_i 为目标函数 f_i 目前为止所找到的最优值。

2.3 MOBA/D 算法流程

将蝙蝠算法中融入切比雪夫分解机制,并使用差分进化策略对种群进行扰动,增强算法多样性,算法 MOBA/D 的基本流程如下:

算法输入:①多目标优化问题。②停止准则。③N:MOBA/D 所分解的子问题个数。④$\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2, \cdots, \boldsymbol{\lambda}^N$:分布均匀的 N 个权重向量。⑤T:权重向量的邻域规模。

步骤 1 初始化

(1)设置 EP = ∅。

(2)计算任意两个权重向量的欧氏距离,为每个权重向量选出最近的 T 个向量作为它的邻居。设 $B(i) = \{i_1, i_2, \cdots, i_T\}$,$i = 1,2,\cdots,N$,其中 $\boldsymbol{\lambda}^{i_1}, \boldsymbol{\lambda}^{i_2}, \cdots, \boldsymbol{\lambda}^{i_T}$为距离 $\boldsymbol{\lambda}^i$ 最近的 T 个权重向量。

(3)初始化种群 $x^1, x^2, \cdots, x^N$,设 $FV^i = F(x^i)$,$i = 1,2,\cdots,N$。

(4)采用基于问题的特点方法初始化 $\boldsymbol{z} = (z_1, z_2, \cdots, z_m)$。

步骤 2 蝙蝠算法更新操作

(1)全局搜索。从 $B(i)$ 中随机选出两个邻居 x^k, x^l,使用 BA 中式(2)~(4)生成一个新解。

(2)局部寻优。如果随机数 rand 1 > r_i,选择邻域中 g^{te} 值最小的解,根据 BA 中式(5)生成一个新解 x_{new}。

(3)更新子问题。如果 rand 2 < A_i 且 $g^{te}(x_{new}|\lambda^i) < g^{te}(x^*|\boldsymbol{\lambda}^i)$,则接受新解并更新子问题。方法如下:若 $g^{te}(x_{new}|\lambda^j, z) \leqslant g^{te}(x^j|\lambda^j, z)$,$j \in B(i)$,则 $x^j = x_{new}$,$FV^j = F(x_{new})$。

(4)更新参考点。若 $z_j < f_j(x_{new})$,则 $z_j = f_j(x_{new})$,$j = 1,2,\cdots,m$。

(5)更新 A_i 和 r_i。

步骤 3 差分进化[15]操作

(1)从邻域中随机选择两个解,应用 DE 算子生成新解 y。

(2)利用新解 y 更新邻域内的子问题。

步骤 4 停止判断:若满足停止条件,则停止算法并输出结果,否则返回步骤 2。

2.4 算法效率分析

MOEA/D-DE 和 MOBA/D 的计算时间主要来自步骤 2 的更新操作,NSGA-II 的计算时间主要取决于快速非支配排序操作。在每一代操作中,3 种

算法均维持着大小为 N 的种群规模。在 MOEA/D－DE 中，当利用 DE 算子产生新解后，需要更新参考点 z，同时在大小为 T 的邻居中更新子问题，因此在每一代中 MOEA/D－DE 的时间复杂度为 $O(mNT)$。在 MOBA/D 中，当利用 BA 算法产生新解后，也需要更新参考点与子问题，还要进一步利用差分操作来进行扰动，因此其每次迭代的时间复杂度为 $O(2mNT)$。因为 NSGA－II 在进行快速非支配排序时需要比较每两个解的支配关系，因此在每一代中 NSGA－II 的时间复杂度为 $O(mN^2)$。由于邻居规模 T 小于种群规模 N，因此 MOBA/D 的复杂度稍高于 MOEA/D－DE，但是低于 NSGA－II。

3　实验仿真与分析

为了充分验证算法的性能，采用 14 个测试函数：ZDT[16] 系列和具有复杂 PS 的 LZ－09[10] 系列，其中 LZ－09 测试函数 F6 具有 3 个目标函数，其他问题为 2 个目标函数。对于 ZDT 系列函数：ZDT1 的 Pareto 前端为凸曲面；ZDT2 为非凸曲面；ZDT3 为非连续函数；ZDT4 含有 21^9 个局部 Pareto 前端；ZDT6 的 PS 具有非均匀的分布。对于 LZ 测试函数：F6 和 F9 为非凸 Pareto 前端；其他问题为凸 Pareto 前端；F7 和 F8 为多峰问题。

3.1　算法性能度量

由于采用的测试函数均可得到其理论最优值，本文采用 Epsilon[17]、Spread[18] 和 IGD[19] 作为评估指标，其中 Epsilon 指标为收敛性指标，Spread 指标为多样性指标，IGD 为综合型指标。

(1) Epsilon(ε) 指标

假设 Pareto 前端集合为 A，那么 Epsilon 指标是移除 A 中每个个体所需要的最小距离的一个尺度。其具体形式是：假设 $\boldsymbol{Z}^1=(Z_1^1,Z_2^1,\cdots,Z_n^1)$，$\boldsymbol{Z}^2=(Z_1^2,Z_2^2,\cdots,Z_n^2)$，其中 n 是问题的目标维数，则

$$\varepsilon(A)=\inf\{\varepsilon\in R\mid\forall z^2\in PF,\exists z^1\in A:z^1\prec_\varepsilon z^2\} \tag{10}$$

其中，当且仅当 $\forall 1\leqslant i\leqslant n:z_i^1<\varepsilon+z_i^2$，则 $z^1\prec_\varepsilon z^2$ 成立。

(2) Spread 指标

Deb 提出的分布指标是衡量所得的 Pareto 前端解集的分布情况。其计算公式为

$$\Delta=\frac{d_f+d_l+\sum_{i=1}^{n-1}\mid d_i-\bar{d}\mid}{d_f+d_l+(n-1)\bar{d}} \tag{11}$$

式中　d_i——所得 Pareto 前端上每两个连续解点的欧氏距离

$\bar{d}$——欧氏距离的平均距离

d_f、d_l——所得 Pareto 前端的边界点与 Pareto 最优边界点的欧氏距离

n——所得 Pareto 前端个体的数目

对于分布均匀的解来说，该指标取 0。因此，该指标的值越小，表明分布程度越均匀。

(3) IGD 指标

IGD 为收敛性评价方法 GD 的反转，它计算 Pareto 面上均匀点到非支配解集上最小距离的平均值，其评价函数为

$$IGD\triangleq 1/N_{ture}\sqrt{\sum_{i=1}^{N_{ture}}d_i^2} \tag{12}$$

式中　d_i——真实 Pareto 前端中的向量与目标空间中每个向量之间的最短距离

IGD 越小，说明收敛性越好。IGD 不仅能评价解集收敛性，也能评价均匀性和广泛性。

3.2　MOBA/D 的邻域规模分析

算法中，邻域规模 T 对算法的收敛速度和多样性都有一定影响[20]，如图 1 所示。为了研究不同的邻域规模 T 的算法性能，分别取 $\boldsymbol{T}=[10,15,20,25,30]$ 进行分析。

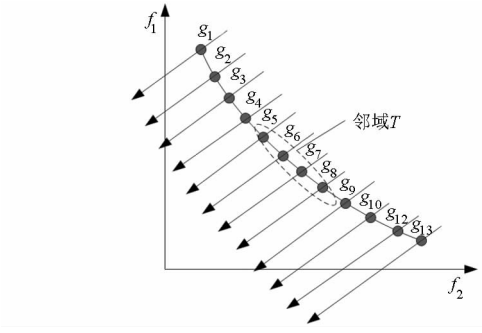

图 1　邻域示意图

Fig. 1　Illustration of neighborhood

设定 2 目标函数的种群大小为 300，3 目标为 500，最大进化代数为 1 500 代，$A=0.6$，$r=0.6$，$\alpha=0.95$，$\gamma=0.05$，$C=1$，$F=0.5$，邻域搜索概率 $\delta=0.9$，子问题更新数目 $n_r=2$ 或 3。对上述 14 个测试函数各独自运行 30 次，采用综合指标 IGD 进行评价。表 1 为 MOBA/D 中不同 T 的运算结果。从表 1 中可见：ZDT1～ZDT3 函数中，$T=10$ 的算法性能最好，且算法的整体性能基本上呈现随 T 增大而效果变差；ZDT4 函数，$T=20$ 的性能最佳；ZDT6 函数，T 的大小对算法性能影响较小。对于 LZ－09 系列函数，$T=20$ 时算法性能最佳。采用 Friedman 进行整体统计分析，如图 2 所示。由图 2 可知，$T=20$ 时算法的整体性能最佳。因此，通过分析可知，邻域规模为 20 时算法的性能最优。

3.3　算法对比实验

进一步比较 MOBA/D、NSGA－II[1] 和 MOEA/D－DE[10] 算法，3 种算法的参数设置见表 2。对每个

表 1 T 为 10、15、20、25 和 30 时 MOBA/D 所得结果 IGD 度量均值与均方差

Tab. 1 Mean values and mean square error of IGD for MOBA/D when T were 10, 15, 20, 25 and 30

测试	$T=10$		$T=15$		$T=20$		$T=25$		$T=30$	
函数	均值	方差	均值	方差	均值	方差	均值	方差	均值	方差
ZDT1	5.74×10^{-5}*	1.6×10^{-6}	5.77×10^{-5}	1.9×10^{-6}	5.92×10^{-5}	2.4×10^{-6}	6.01×10^{-5}	3.0×10^{-6}	6.01×10^{-5}	2.2×10^{-6}
ZDT2	4.71×10^{-5}*	4.3×10^{-7}	4.77×10^{-5}	8.1×10^{-7}	4.82×10^{-5}	1.3×10^{-6}	4.87×10^{-5}	1.1×10^{-6}	4.98×10^{-5}	2.6×10^{-6}
ZDT3	8.87×10^{-5}*	6.7×10^{-7}	8.91×10^{-5}	1.4×10^{-6}	8.97×10^{-5}	1.2×10^{-6}	8.91×10^{-5}	9.8×10^{-7}	8.87×10^{-5}	1.1×10^{-6}
ZDT4	6.14×10^{-5}	2.2×10^{-6}	6.15×10^{-5}	1.8×10^{-6}	6.03×10^{-5}*	1.5×10^{-6}	6.13×10^{-5}	1.8×10^{-6}	6.12×10^{-5}	1.6×10^{-6}
ZDT6	4.63×10^{-5}	9.8×10^{-9}	4.63×10^{-5}*	5.4×10^{-9}	4.63×10^{-5}	1.0×10^{-8}	4.63×10^{-5}	1.1×10^{-8}	4.63×10^{-5}	9.0×10^{-9}
LZ-09_F1	8.04×10^{-5}	1.9×10^{-5}	7.36×10^{-5}	2.5×10^{-6}	7.24×10^{-5}*	3.9×10^{-7}	7.25×10^{-5}	4.1×10^{-7}	7.24×10^{-5}	4.8×10^{-7}
LZ-09_F2	1.68×10^{-4}	6.9×10^{-5}	1.31×10^{-4}*	1.7×10^{-5}	1.35×10^{-4}	1.0×10^{-5}	1.42×10^{-4}	1.2×10^{-5}	1.54×10^{-4}	1.2×10^{-5}
LZ-09_F3	2.63×10^{-4}	1.4×10^{-4}	2.10×10^{-4}	1.2×10^{-4}	1.96×10^{-4}*	4.9×10^{-5}	2.22×10^{-4}	6.6×10^{-5}	2.79×10^{-4}	1.1×10^{-4}
LZ-09_F4	2.14×10^{-4}	9.2×10^{-5}	1.93×10^{-4}	1.1×10^{-4}	1.76×10^{-4}*	6.0×10^{-5}	1.95×10^{-4}	6.5×10^{-5}	2.46×10^{-4}	9.3×10^{-5}
LZ-09_F5	8.60×10^{-4}	2.8×10^{-4}	8.35×10^{-4}*	2.2×10^{-4}	9.77×10^{-4}	2.3×10^{-4}	8.79×10^{-4}	1.6×10^{-4}	9.27×10^{-4}	2.6×10^{-4}
LZ-09_F6	1.15×10^{-3}	3.2×10^{-5}	1.13×10^{-3}	2.9×10^{-5}	1.10×10^{-3}*	2.3×10^{-5}	1.10×10^{-3}	2.0×10^{-5}	1.11×10^{-3}	3.8×10^{-5}
LZ-09_F7	4.33×10^{-3}	2.3×10^{-3}	2.24×10^{-3}	2.0×10^{-3}	1.67×10^{-3}	1.1×10^{-3}	1.04×10^{-3}	7.7×10^{-4}	8.64×10^{-4}*	5.7×10^{-4}
LZ-09_F8	6.54×10^{-3}	2.3×10^{-3}	5.61×10^{-3}	1.3×10^{-3}	4.51×10^{-3}	1.1×10^{-3}	3.66×10^{-3}*	1.4×10^{-3}	4.21×10^{-3}	1.6×10^{-3}
LZ-09_F9	6.55×10^{-4}	3.2×10^{-4}	2.51×10^{-4}	1.1×10^{-4}	2.21×10^{-4}*	1.1×10^{-4}	2.27×10^{-4}	8.7×10^{-5}	2.90×10^{-4}	5.0×10^{-5}

注：* 为最优值，下同。

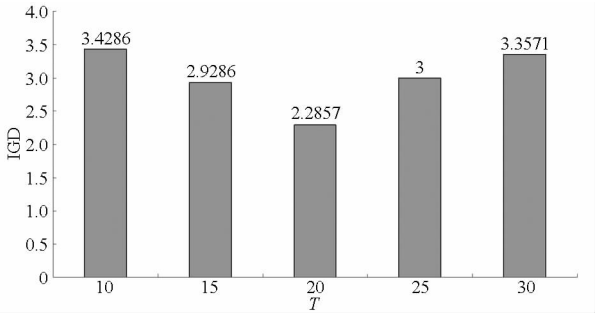

图 2 IGD 值的 Friedman 排名柱状图

Fig. 2 Histogram of Friedman number of IGD value

测试函数均独立运行 30 次，然后统计指标 Epsilon、Spread 和 IGD 的均值和标准差，所得结果如表 3～5 所示。

由表 3 可知，MOBA/D 算法获得了 10 个最优值，MOEA/D-DE 获得了 3 个最优值，NSGA-II 算法获得了一个最优值。对于 F1～F9 函数，除了 F7 函数差于 MOEA/D-DE 以外，均优于另外 2 种算法，且在数量级远好于 NSGA-II 算法。对于 ZDT 函数，MOBA/D 算法在 ZDT1～ZDT2 函数上收敛性

表 2 算法参数设置

Tab. 2 Parameter settings of algorithms

算法	算法参数设置	共同参数
NSGA-II	SBX[21] 交叉概率 $P_c=0.9$；多项式变异概率 $P_m=1/V_{ar}$，V_{ar} 为变量长度；	种群大小
MOEA/D-DE	$C=1$，$F=0.5$，邻域大小 $T=20$，邻域搜索概率 $\delta=0.9$，子问题更新数目 $n_r=2$ 或 3；	$N_P=\begin{cases}300 & (N=2)\\500 & (N=3)\end{cases}$
MOBA/D	$A=0.6$，$r=0.6$，$\alpha=0.95$，$\gamma=0.05$，$C=1$，$F=0.5$，邻域大小 $T=20$，邻域搜索概率 $\delta=0.9$，子问题更新数目 $n_r=2$ 或 3。	最大迭代次数 $G=1\,500$

表 3 Epsilon 平均值及标准差

Tab. 3 Mean and standard deviation of Epsilon

测试函数	MOBA/D		MOEA/D-DE		NSGA-II	
	平均值	标准差	平均值	标准差	平均值	标准差
ZDT1	3.04×10^{-3}*	6.5×10^{-4}	3.21×10^{-3}	4.0×10^{-4}	4.65×10^{-3}	7.6×10^{-4}
ZDT2	2.47×10^{-3}*	1.6×10^{-4}	2.57×10^{-3}	2.3×10^{-4}	4.40×10^{-3}	6.2×10^{-4}
ZDT3	5.32×10^{-3}	1.2×10^{-4}	5.27×10^{-3}	1.2×10^{-4}	3.31×10^{-3}*	6.9×10^{-4}
ZDT4	2.94×10^{-3}	1.8×10^{-4}	2.92×10^{-3}*	1.1×10^{-4}	4.28×10^{-3}	6.4×10^{-4}
ZDT6	1.64×10^{-3}	1.9×10^{-6}	1.64×10^{-3}*	2.3×10^{-6}	4.70×10^{-3}	7.5×10^{-4}
LZ-09_F1	2.47×10^{-3}*	3.6×10^{-4}	2.53×10^{-3}	5.1×10^{-4}	5.68×10^{-3}	4.1×10^{-4}
LZ-09_F2	7.10×10^{-3}*	1.3×10^{-3}	8.94×10^{-3}	3.4×10^{-3}	0.173	4.4×10^{-2}
LZ-09_F3	2.05×10^{-2}*	1.2×10^{-2}	7.91×10^{-2}	8.8×10^{-2}	0.111	2.7×10^{-2}
LZ-09_F4	1.15×10^{-2}*	6.7×10^{-3}	3.48×10^{-2}	1.6×10^{-2}	0.148	2.5×10^{-2}
LZ-09_F5	6.39×10^{-2}*	1.5×10^{-2}	6.86×10^{-2}	3.1×10^{-2}	9.50×10^{-2}	2.2×10^{-2}
LZ-09_F6	8.62×10^{-2}*	1.5×10^{-2}	9.35×10^{-2}	1.4×10^{-2}	0.228	2.8×10^{-2}
LZ-09_F7	0.111	8.3×10^{-2}	7.62×10^{-2}*	7.1×10^{-2}	0.247	6.3×10^{-2}
LZ-09_F8	0.244*	8.2×10^{-2}	0.265	7.0×10^{-2}	0.254	5.8×10^{-2}
LZ-09_F9	1.48×10^{-2}*	6.8×10^{-3}	1.80×10^{-2}	1.2×10^{-2}	0.189	2.9×10^{-2}

表 4　Spread 平均值及标准差

Tab. 4　Mean and standard deviation of Spread

测试函数	MOBA/D		MOEA/D－DE		NSGA－II	
	平均值	标准差	平均值	标准差	平均值	标准差
ZDT1	0.285*	2.4×10^{-3}	0.286	3.8×10^{-3}	0.374	1.9×10^{-2}
ZDT2	0.141*	7.4×10^{-3}	0.143	8.8×10^{-3}	0.377	1.5×10^{-2}
ZDT3	1.03	1.6×10^{-3}	1.030	1.9×10^{-3}	0.791*	6.7×10^{-3}
ZDT4	0.286*	2.3×10^{-3}	0.286	2.5×10^{-3}	0.390	1.7×10^{-2}
ZDT6	0.150*	8.8×10^{-5}	0.150	1.8×10^{-4}	0.631	2.6×10^{-2}
LZ－09_F1	0.287*	2.4×10^{-3}	0.295	3.6×10^{-2}	0.493	6.8×10^{-2}
LZ－09_F2	0.384*	6.2×10^{-2}	0.404	1.1×10^{-1}	1.530	1.0×10^{-1}
LZ－09_F3	0.395*	7.3×10^{-2}	0.459	1.1×10^{-1}	0.787	7.9×10^{-2}
LZ－09_F4	0.387*	5.8×10^{-2}	0.565	1.8×10^{-1}	0.647	6.1×10^{-2}
LZ－09_F5	0.531*	6.0×10^{-2}	0.548	6.3×10^{-2}	0.714	4.7×10^{-2}
LZ－09_F6	0.796	1.7×10^{-2}	0.795*	1.4×10^{-2}	0.841	3.9×10^{-2}
LZ－09_F7	1.010	3.0×10^{-1}	0.932*	3.5×10^{-1}	1.530	1.3×10^{-1}
LZ－09_F8	1.27*	1.8×10^{-1}	1.310	8.0×10^{-2}	1.520	8.3×10^{-2}
LZ－09_F9	0.247*	4.9×10^{-2}	0.277	9.5×10^{-2}	1.700	1.2×10^{-1}

表 5　IGD 平均值及标准差

Tab. 5　Mean and standard deviation of IGD

测试函数	MOBA/D		MOEA/D－DE		NSGA－II	
	平均值	标准差	平均值	标准差	平均值	标准差
ZDT1	5.84×10^{-5}*	2.1×10^{-6}	6.05×10^{-5}	2.8×10^{-6}	6.04×10^{-5}	1.7×10^{-6}
ZDT2	4.86×10^{-5}*	1.3×10^{-6}	4.94×10^{-5}	1.8×10^{-6}	6.20×10^{-5}	2.5×10^{-6}
ZDT3	8.93×10^{-5}	1.6×10^{-6}	8.91×10^{-5}	1.3×10^{-6}	4.29×10^{-5}*	1.6×10^{-6}
ZDT4	6.07×10^{-5}	1.9×10^{-6}	6.06×10^{-5}	1.6×10^{-6}	6.02×10^{-5}*	1.3×10^{-6}
ZDT6	4.63×10^{-5}	9.3×10^{-9}	4.63×10^{-5}*	7.8×10^{-9}	7.44×10^{-5}	2.7×10^{-6}
LZ－09_F1	7.25×10^{-5}*	6.4×10^{-7}	7.44×10^{-5}	9.6×10^{-6}	1.29×10^{-4}	3.2×10^{-6}
LZ－09_F2	1.33×10^{-4}*	9.5×10^{-6}	1.49×10^{-4}	2.9×10^{-5}	4.88×10^{-3}	1.4×10^{-3}
LZ－09_F3	2.39×10^{-4}*	1.1×10^{-4}	1.80×10^{-3}	2.4×10^{-3}	2.43×10^{-3}	8.8×10^{-4}
LZ－09_F4	1.65×10^{-4}*	6.3×10^{-5}	3.93×10^{-4}	1.9×10^{-4}	2.85×10^{-3}	8.9×10^{-4}
LZ－09_F5	8.36×10^{-4}*	1.9×10^{-4}	9.39×10^{-4}	5.5×10^{-4}	1.80×10^{-3}	5.3×10^{-4}
LZ－09_F6	1.12×10^{-3}*	3.7×10^{-5}	1.15×10^{-3}	6.2×10^{-5}	3.09×10^{-3}	3.2×10^{-4}
LZ－09_F7	1.49×10^{-3}	1.3×10^{-3}	9.57×10^{-4}*	1.0×10^{-3}	7.48×10^{-3}	2.4×10^{-3}
LZ－09_F8	5.23×10^{-3}*	2.3×10^{-3}	5.77×10^{-3}	1.9×10^{-3}	6.95×10^{-3}	1.4×10^{-3}
LZ－09_F9	2.06×10^{-4}*	5.8×10^{-5}	2.37×10^{-4}	1.7×10^{-4}	5.79×10^{-3}	1.7×10^{-3}

较好，NSGA－II 算法在 ZDT3 函数上收敛性占优，在 ZDT4 和 ZDT6 函数上，MOEA/D－DE 收敛性略微高于 MOBA/D 算法。

由表 4 可知，14 个测试函数中 MOBA/D 算法获得了 11 个最优值；MOEA/D－DE 获得 2 个最优值；NSGA－II 算法获得 1 个最优值。数据分析表明，MOBA/D 在求解 ZDT 系列和 LZ－09 系列函数时多样性能总体最佳；在求解 ZDT3 函数时，NSGA－II 算法的多样性最好，MOBA/D 算法其次；对于 F6 和 F7 函数，MOEA/D－DE 性能较好。

由表 5 可知，在 IGD 指标上 MOBA/D 算法获得 10 个最优值，MOEA/D－DE 获得 2 个最优值，NSGA－II 获得 2 个最优值。对于 ZDT 系列函数，MOBA/D 算法在 ZDT1～ZDT2 上获得较好的分布性，NSGA－II 在 ZDT3～ZDT4 上性能较优，MOEA/D－DE 在求解 ZDT6 时性能较好，且 3 种算法的性能相差较小。对于 F1～F9 函数，MOBA/D 算法占有绝对的优势，仅在 F7 函数上差于 MOEA/D－DE 算法。在 F3 函数求解上，MOBA/D 算法所获得的均值约等于 MOEA/D－DE 和 NSGA－II 算法的 0.1 倍。在 F1～F9 函数中 MOBA/D 均远胜于 NSGA－II 算法。

综合以上分析可知，MOBA/D 与 MOEA/D－DE、NSGA－II 相比，具有较强的竞争力。为了直观地显示出 MOBA/D 算法的性能，给予 F6 函数的盒图和 Pareto 前端图，如图 3 和图 4 所示，其中图 4 为从 30 次运行结果中取每个算法得到的较好结果。由图 3 和图 4 获得的结果与表 3～5 的结果一致。

4　工程实例

流体动压滑动轴承是表面完全被油膜分开的一

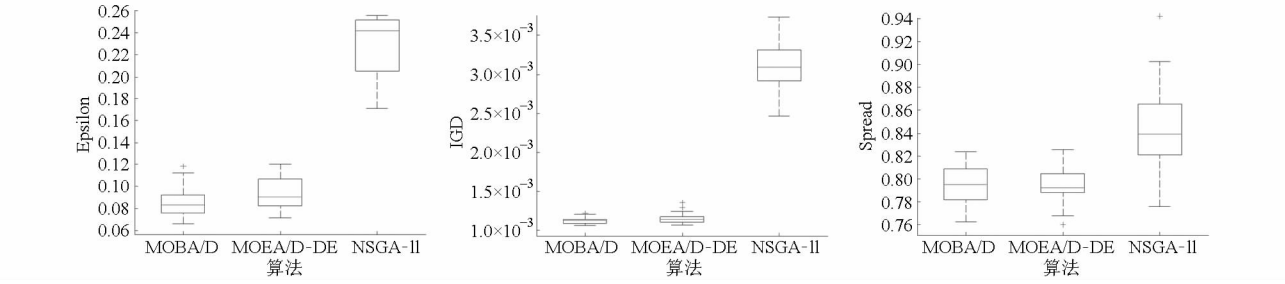

图 3 3 种算法在 LZ-09_F6 函数的性能盒图

Fig. 3 Boxplot of algorithms on LZ-09_F6

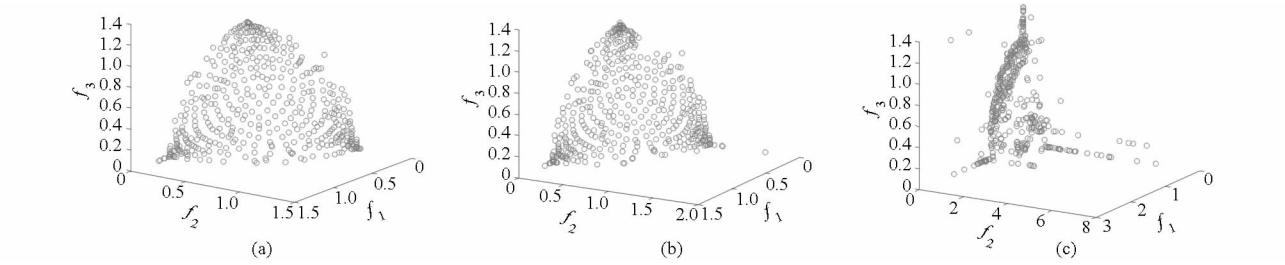

图 4 3 种算法在 F6 函数获得的 Pareto 前端图

Fig. 4 Obtained Pareto front on F6 of three algorithms

(a) MOBA/D (b) MOEA/D-DE (c) NSGA-II

种滑动轴承,本文以轴承承载能力最大,发热量最小,摩擦因数最小为优化目标,设计变量为宽径比 B/d,相对间隙 ψ,润滑油动力粘度 η,故动压滑动轴承多目标优化数学模型[22-24]为

目标函数

$$\begin{cases} \max f_1 = C_p = \dfrac{Fx_2^2}{\omega d^2 x_1 x_3} \\ \min f_2 = pv = \dfrac{Fv}{x_1 d^2} \\ \min f_3 = \dfrac{\pi \omega d^2 x_1 x_3}{Fx_2} + 0.55 x_2 x_1^{-1.5} \end{cases} \tag{13}$$

约束条件

$$\begin{cases} g_1(X) = k(R_{z1} + R_{z2}) - 55\dfrac{nd^3 x_3 x_1^2}{Fx_2(1+x_1)} \\ g_2(X) = x_1 - (B/d)_{\max} \\ g_3(X) = (B/d)_{\max} - x_1 \\ g_4(X) = \dfrac{F}{x_1 d^2} - p_{\max} \\ g_5(X) = p_{\min} - \dfrac{F}{x_1 d^2} \\ g_6(X) = x_2 - \psi_{\max} \\ g_7(X) = \psi_{\min} - x_2 \\ g_8(X) = x_3 - \eta_{\max} \\ g_9(X) = \eta_{\min} - x_3 \end{cases} \tag{14}$$

式中 C_p——承载量系数

F——轴承工作载荷

ω——轴颈的角速度

p——轴承平均比压

d——轴承直径

v——轴颈的圆周速度

k——考虑几何形状误差、安装误差和轴颈变形的安全系数,一般取 $k=2\sim3$

R_{z1}、R_{z2}——轴颈和轴承孔的表面粗糙度

以设计某齿轮箱的流体动压润滑径向滑动轴承为例,已知工作载荷为 35 000 N,轴承直径 $d=100$ mm,轴的转速为 $n=1\,000$ r/min,决策变量 x_1 取值范围[0.5,1.2],x_2 取值范围[0.001,0.002],x_3 取值范围[0.02,0.04]。

为了验证 MOBA/D 算法求解含有约束问题的能力,将其应用于滑动轴承优化设计问题。该优化问题是一个 3 目标函数的约束优化问题,本文采用如下方法处理约束[25],当在 2 个比较个体中,一个个体为可行解,另外一个个体为不可行解时,选择可行解;当 2 个比较个体均为可行解时,选择非支配的个体占优;当 2 个比较个体均为不可行解时,选择违反约束条件程度小的个体。设置算法的参数为:种群 N_p 设为 500,$A=0.6$,$r=0.6$,$\alpha=0.95$,$\gamma=0.05$,$C=1$,$F=0.5$,$T=20$,$\delta=0.9$,$n_r=3$。

图 5 为 MOBA/D 算法获得的 Pareto 前端曲线。图 6 为承载量系数与液体摩擦因数之间的关系曲线,图 7 为液体摩擦因数和发热量之间的关系曲线。MOBA/D 算法获得的 Pareto 前端解集同文献[21]相当,说明算法在求解含有约束的实际工程问题时具有实用性。

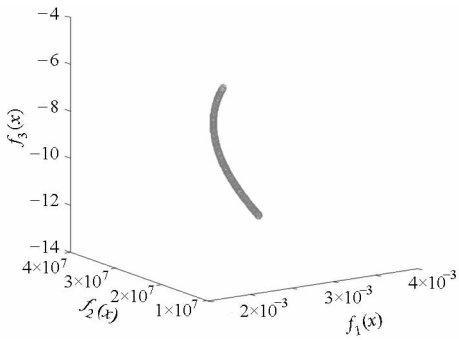

图 5　Pareto 曲线

Fig. 5　Pareto curve

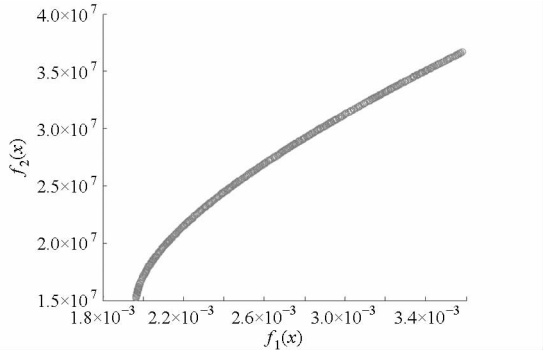

图 6　承载量系数与液体摩擦因数之间的关系曲线

Fig. 6　Relation curve between f_1 and f_2

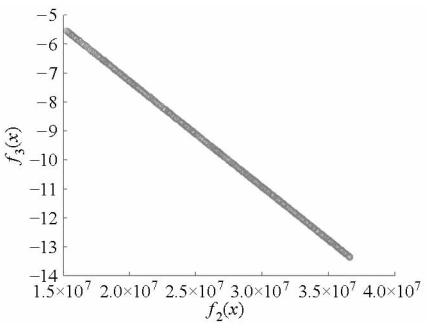

图 7　液体摩擦因数与发热量之间的关系曲线

Fig. 7　Relation curve between f_2 and f_3

5　结论

(1)在分析蝙蝠算法的基础上,将蝙蝠算法融入切比雪夫分解技术中,并采用差分进化策略对种群进行扰动,提出了一种基于分解机制的多目标蝙蝠算法(MOBA/D)。通过 ZDT 系列和 LZ－09 系列测试函数进行实验仿真,结果表明邻域规模为 20 时算法的性能最优。将 MOBA/D 算法与 NSGA－II 和 MOEA/D－DE 算法,在 14 个基准测试函数上进行对比分析,结果表明新算法在 ZDT 系列上与另外 2 种算法性能相当,但在 LZ－09 系列函数上占有绝对的优势,说明 MOBA/D 算法在解决复杂的 PS 问题上有较好的多样性和收敛性。

(2) 为了分析 MOBA/D 算法求解含有约束问题的能力,将其应用于滑动轴承多目标优化设计问题中,结果表明该算法获得的 Pareto 前端较均匀,说明该算法具有求解含有约束问题的能力和工程实用性。

参　考　文　献

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

2　Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm for multi-objective optimization [C]// Proceedings of the Evolutionary Methods for Design, Optimization and Control. Athens: International Center for Numerical Methods in Engineering, 2002:95－100.

3　Nebro J, Durillo J, Luna F, et al. MOCell: a cellular genetic algorithm for multi-objective optimization [J]. International Journal of Intelligent Systems, 2009, 24(7): 726－746.

4　Bader J, Zitzler E. HypE: an algorithm for fast hypervolume-based many-objective optimization [J]. Evolutionary Computation, 2011, 19(1):45－76.

5　Zhang Q, Li H. MOEA/D: a multi-objective evolutionary algorithm based on decomposition [J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 712－731.

6　周爱民,张青富,张桂戌. 一种基于混合高斯模型的多目标进化算法[J]. 软件学报,2014,25(5):913－928.
Zhou Aimin, Zhang Qingfu, Zhang Guixu. Multi-objective evolutionary algorithm based on mixture Gaussian models [J]. Journal of Software, 2014, 25(5):913－928. (in Chinese)

7　Yang Xinshe. A new meta-heuristic bat-inspired algorithm [M]// Cruz C, González J R, Krasnogor N, et al. Nature inspired cooperative strategies for optimization (NICSO 2010). New York: Springer,2010:65－74.

8　Yang X S, Gandomi A H. Bat algorithm: a novel approach for global engineering optimization [J]. Engineering Computations, 2012, 29(5):464－483.

9　Deb K. Multi-objective optimization using evolutionary algorithms [M]. New York: John Wiley & Sons, 2001.

10　Li H, Zhang Q. Multi-objective optimization problems with complicated Pareto sets, MOEA/D and NSGA－II [J]. IEEE Transactions on Evolutionary Computation, 2009, 13(2): 284－302.

11　Miettinen K. Nonlinear multi-objective optimization [M]. Norwell, MA: Kluwer, 1999.

12　Das I, Dennis J E. Normal-boundary intersection: a new method for generating Pareto optimal points in multi-criteria optimization problems [J]. SIAM Journal on Optimization, 1998, 8(3):631－657.

13 Messac A, Ismail-Yahaya A, Mattson C. The normalized constraint method for generating the Pareto frontier [J]. Structural and Multidisciplinary Optimization, 2003, 25(2):86 – 98.
14 谭艳艳. 几种改进的分解类多目标进化算法及其应用[D]. 西安:西安电子科技大学,2013.
Tan Yanyan. Several modified decomposition-based multi-objective evolutionary algorithms and their applications [D]. Xi'an: Xidian University,2013. (in Chinese)
15 Price K, Storn R, Lampinen J. Differential evolution: a practical approach for global optimization [M]. Berline: Springer Verlag, 2005.
16 Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results [J]. Evolutionary Computation, 2000, 8(2):173 – 195.
17 Fonseca C M, Knowles J D, Thiele L, et al. A tutorial on the performance assessment of stochastic multiobjective optimizers[C]// Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005), 2005, 216: 240.
18 Corne D W, Jerran N R, Knowles J D, et al. PESA-II: region-based selection in evolutionary multi-objective optimization[C]// Proceedings of the Genetic and Evolutionary Computation Conference, 2001:283 – 290.
19 Zitzler E, Thiele L, Laumanns M, et al. Performance assessment of multi-objective optimizers: an analysis and review [J]. IEEE Transactions on Evolutionary Computation, 2003, 7(2):117 – 132.
20 Ishibuchi H, Akedo N, Nojima Y. Relation between neighborhood size and MOEA/D performance on many-objective problems [C]// Evolutionary Multi-Criterion Optimization. Berlin, Heidelberg: Springer, 2013: 459 – 474.
21 Deb K, Agrawal R B. Simulated binary crossover for continuous search space [J]. Complex Systems, 1995, 9(2):115 – 148.
22 卢青波,张学良,温淑花,等 基于差异演化算法的动压滑动轴承多目标优化[J]. 农业机械学报,2013,44(3):230 – 236.
Lu Qingbo, Zhang Xueliang, Wen Shuhua, et al. Multi-objective optimization of hydrodynamic sliding bearing based on differential evolution algorithm [J]. Transactions of the Chinese Society for Agricultural Machinery, 2013, 44(3):230 – 236. (in Chinese)
23 张鄂,蒙娟,贾焕如. 液体动压径向滑动轴承的概率多目标优化设计[J]. 机械科学与技术,2001,20(2):224 – 226.
Zhang E, Meng Juan, Jia Huanru. The multi-aim probability optimal optimization of hydrodynamic sliding bearing [J]. Mechanical Science and Technology, 2001, 20(2):224 – 226. (in Chinese)
24 王增胜,刘保国,吴磊,等. 齿轮箱流体动压滑动轴承的多目标优化[J]. 机械传动,2007,31(5):74 – 75.
Wang Zengsheng, Liu Baoguo, Wu Lei, et al. Multi-objective optimization of hydrodynamic sliding bearing used in gearbox [J]. Journal of Mechanical Transmission, 2007, 31(5):74 – 75. (in Chinese)
25 Deb K. An efficient constraint handling method for genetic algorithms [J]. Computation Methods in Applied Mechanics and Engineering, 2000, 86(2 – 4):311 – 338.

(上接第 315 页)

13 王勇,蔡自兴,周育人,等. 约束优化进化算法[J]. 软件学报,2009, 20(1):11 – 29.
Wang Yong, Cai Zixing, Zhou Yuren, et al. Constrained optimization evolutionary algorithms [J]. Journal of Software, 2009, 20(1):11 – 29. (in Chinese)
14 Deb K. An efficient constraint handling method for genetic algorithms [J]. Computation Methods in Applied Mechanics and Engineering, 2000, 86(2 – 4):311 – 338.
15 Knowles J D, Corne D W. Approximating the nondominated front using the Pareto archived evolution strategy [J]. Evolutionary Computation, 2000, 8(2):149 – 172.
16 Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: empirical results [J]. Evolutionary Computation, 2000,8(2):173 – 195.
17 Huband S, Hingston P, Barone L, et al. A review of multiobjective test problems and a scalable test problem toolkit [J]. IEEE Transactions on Evolutionary Computation, 2006, 10(5):477 – 506.
18 Pholdee N, Bureerat S. Hybrid real-code population-based incremental learning and approximate gradients for multi-objective truss design [J]. Engineering Optimization, 2014, 46(8):1032 – 1051.
19 Gandomi A H, Talatahari S, Yang X S, et al. Design optimization of truss structures using cuckoo search algorithm [J]. Structural Design of Tall and Special Buildings, 2013,22(17):1330 – 1349.
20 唐和生,胡长远,薛松涛. 桁架结构多目标优化的免疫克隆选择算法[J]. 湖南大学学报:自然科学版,2013,40(5):18 – 23.
Tang Hesheng, Hu Changyuan, Xue Songtao. Immune clone selection algorithm for truss structure multi-objective optimization [J]. Journal of Hunan University: Natural Sciences, 2013,40(5):18 – 23. (in Chinese)
21 张卓群,李宏男. 基于蚁群算法的桁架结构布局离散变量优化方法[J]. 计算力学学报,2014,30(3):336 – 342.
Zhang Zhuoqun, Li Hongnan. The method of truss structure layout discrete variable optimization based on ant colony optimization [J]. Chinese Journal of Computational Mechanics,2014,30(3):336 – 342. (in Chinese)