DOI：10.7652/xjtuxb201903010

一种采用改进交叉熵的多目标优化问题求解方法

赵舵，唐启超，余志斌
（西南交通大学电气工程学院，610031，成都）

摘要：针对传统交叉熵算法不能解决多目标优化问题，采用单目标交叉熵优化算法提出了改进多目标交叉熵优化(Multi-Objective Cross Entropy Optimization，MOCEO)算法。首先，采用个体选择机制来保留进化过程中的优良个体，通过精英保留策略提取优良个体分布信息以不断修正算法正态分布概率模型参数；其次，引入进化方向在正态分布群体采样过程中，引导所产生新个体在解空间中的分布使得种群朝着性能提高的方向进化；最后，为了避免陷入局部最优点在参数平滑操作过程中，定义了调节系数随机调整正态分布概率模型参数。ZDT和DTLZ系列多目标问题的测试结果表明，与经典多目标优化算法NSGA-II、SPEA2、MOEAD、PAES相比，MOCEO在超体积和反转世代距离性能指标以及进化速度等方面较好，是一种收敛速度快、寻优能力强、鲁棒性高的算法。为验证MOCEO在工程实际中的效果，将其应用于某型高速列车悬挂系统横向平稳控制系统的参数优化中，仿真结果表明：相比于NSGA-II算法，使用MOCEO优化调整控制系统参数后，车体横向平稳性指标提高4.16%，横向加速度峰值减小10.34%，横向振动加速度在1～2 Hz人体敏感频率范围内有一定改善，列车具有更好的横向平稳性能。

关键词：多目标优化；进化算法；交叉熵优化算法；横向平稳性
中图分类号：TP18 **文献标志码**：A **文章编号**：0253-987X(2019)03-0066-09

A Solution to Multi-Objective Optimization Problem with Improved Cross Entropy Optimization

ZHAO Duo，TANG Qichao，YU Zhibin
(School of Electrical Engineering，Southwest Jiaotong University，Chengdu 610031，China)

Abstract：Due to the traditional cross entropy algorithm is unable to solve multi-objective optimization problem，an improved multi-objective cross entropy optimization (MOCEO) algorithm is proposed based on the single target cross entropy optimization algorithm. The individual selection mechanism is adopted to retain the elite individuals in the evolution process，and the distribution information is extracted via the elite retention strategy to constantly correct the parameters of the normal distribution probability model of the algorithm. Then the evolution direction in the traditional normal distribution population is introduced. During the sampling process，the new individuals are guided to the spatial distribution of the solution so that the population evolves towards the improvement of performance. To avoid the algorithm falling into local optimum，the adjustment coefficient is defined in the process of the traditional parameter

收稿日期：2018-05-04。 作者简介：赵舵(1975—)，男，副教授，硕士生导师；唐启超(通信作者)，男，硕士生。 基金项目：国家自然科学基金资助项目(U1730105)；国家科技支撑计划资助项目(2015BAG14B01-05)；四川省科技厅资助项目(17ZDYF1517)。
网络出版时间：2018-12-10 网络出版地址：http://kns.cnki.net/kcms/detail/61.1069.T.20181207.1459.002.html

smoothing operation. Adjusting the parameters of the normal distribution probability model, premature convergence of the algorithm is effectively avoided. This paper implements the MOCEO and compares it in terms of the hyper volume, the inverted generational distance performance indications and the evolutionary speed with the NSGA-II, SPEA2, MOEAD, PAES algorithms using the ZDT and DTLZ test functions. The results indicate that MOCEO is superior to the other algorithms with fast convergence, strong searching ability and high robustness. An optimization for parameters of horizontal stability control system of a high-speed train suspension system verifies the effect of MOCEO. Compared with the NSGA-II algorithm, the lateral stability index of the train body is increased by 4.16% and the peak value of lateral acceleration is reduced by 10.34% by adjusting the MOCEO optimization parameters of the control system. The lateral vibration acceleration of the vehicle body is improved in the frequency range of 1-2 Hz, to which human body is sensitive, and the train achieves better lateral stability.

Keywords: multi-objective optimization; evolutionary algorithm; cross entropy optimization algorithm; lateral stability

交叉熵(Cross Entropy,CE)优化算法是一类新型的启发式随机优化算法，在优化过程中无需优化问题的梯度信息，仅根据适应度函数进行寻优，具有计算复杂度低、鲁棒性强并能以较大概率求得全局最优解的特点[1-4]。目前，有关 CE 优化算法的研究已经取得了一定进展，并解决了许多优化问题。在单目标优化问题的求解方面：文献[5]介绍了基本 CE 优化算法的原理及改进，并讨论了其在组合优化和机器学习方面的应用；文献[6]在 CE 优化算法的开发和迭代过程中做了改进，提高了收敛速度，并将其应用于连续变量的逆问题求解；文献[7]将 CE 优化算法应用于比例-积分-微分(PID)控制器的设计，并与基于遗传算法的 PID 控制器设计方法进行比较，结果表明 CE 优化算法具有较低的计算复杂度。随着研究的发展，CE 优化算法逐步被拓展到多目标优化问题的求解中：文献[8]将广义分解的方法和 CE 优化算法结合，提出了 MACE-gD 算法，并与 MOEA/D 和 RM-MEDA 算法进行了性能比较；文献[9]将模糊 c 均值聚类算法与 CE 优化算法相结合，提出了改进交叉熵优化算法，并应用于解决多目标不等间距阵列综合问题；文献[10]结合了分布估计算法和 CE 优化算法的优点，提出了一种改进的 CE 优化算法，并应用于合金微钻孔加工工艺参数的多目标优化问题中，结果表明加工效率得到了有效提高。

高速和舒适是世界铁路发展的主流，因此对于高速铁道车辆横向运行平稳性的要求越来越高，被动悬挂系统逐渐难以满足使用要求，而主动悬挂和半主动悬挂控制是改善列车横向运行平稳性的有效方法[11-12]。结合我国实际，采用半主动悬挂控制系统是目前最佳的控制方法[13]，半主动悬挂控制系统的参数优化对于列车横向平稳性的改善至关重要。

本文提出了采用 Pareto 精英个体保留排序策略以及概率分布渐进变化的多目标交叉熵优化(Multi-Objective Cross Entropy Optimization，MOCEO)算法，并将其与 NSGAII[14]、SPEA2[15]、MOEAD[16]以及 PAES[17]算法一同应用于经典多目标优化问题求解分析，超体积和反转世代距离指标表明了本文算法求解多目标问题的有效性。最后，将 MOCEO 应用于 17 自由度列车横向半主动控制系统的参数优化，结果表明，MOCEO 具有更快的收敛速度，优化后的列车系统具有更好的横向运行平稳性。

1　交叉熵优化算法的基本思想

CE 优化算法源于对小概率事件的估计，后来被拓展到求解最优化问题[18]。

考虑一般最大化问题，若 S 为有限状态集 χ 的实值性能函数，γ^* 为其最大值，则二者关系为 $S(x^*)=\gamma^*=\max\limits_{x\in\chi}S(x)$，式中 x 为统计样本。定义集合 χ 上的指示函数族为 $\{I_{\{S(x)\geqslant\gamma\}},\gamma\in\mathbf{R}\}$，概率密度函数族为 $\{f(x,u),u\in\mathbf{R}\}$，则有如下估计问题

$$l(\gamma)=P_u(S(x)\geqslant\gamma)=\int\limits_{x\in\chi}I_{\{S(x)\geqslant\gamma\}}f(x,u)=E_u(I_{\{S(x)\geqslant\gamma\}}) \tag{1}$$

式中：P_u 是条件 $\{S(x)\geqslant\gamma\}$ 的概率；E_u 是相应的期望算子。估计概率 $l(\gamma)$ 的最直接的方法是：产生 N 个

随机样本 $X_1,X_2,\cdots,X_N$，然后使用 $\frac{1}{N}\sum_{i=1}^{N}I_{\{S(X_i)\geqslant\gamma\}}$ 对 $l(\gamma)$进行无偏估计。当 $l(\gamma)$很小时，需要很大的样本数 N 才能得到满意的精度，为此需要采用重要度采样技术，即从一个重要度采样密度函数 $g(x)$中产生随机样本。因此，对 l 的估计就变为

$$\hat{l}(\gamma)=\frac{1}{N}\sum_{i=1}^{N}I_{\{S(X_i)\geqslant\gamma\}}\frac{f(X_i,u)}{g(X_i)} \tag{2}$$

式中 X_i 是由 $g(x)$生成的样本。可以看出，$\hat{l}(\gamma)$的估计取决于 $g(x)$的选择。最优的重要采样密度函数为

$$g^*(x)=\frac{I_{\{S(X_i)\geqslant\gamma\}}f(x,u)}{l} \tag{3}$$

由于 $g^*(x)$和未知的 $l(\gamma)$有关，需在$\{f(\cdot,u)\}$函数族内选择一个合适的 $g^*(x)$，即选择一个参数 u 使得 $g^*(x)$和$\{f(\cdot,u)\}$的距离(即 CE)最小。两个概率密度函数 $h(x)$和 $h'(x)$之间的 CE 定义为

$$D(h,h')=\int h(x)\ln\frac{h(x)}{h'(x)}\mathrm{d}x \tag{4}$$

使得 $g^*(x)$和 $f(x,u)$的 CE 最小的等价式为

$$\begin{aligned}&\min D(g^*(x),f(x,u))=\\&\min\int g^*(x)\ln\frac{g^*(x)}{f(x,u)}\mathrm{d}x=\\&\min\left(\int g^*(x)\ln g^*(x)\mathrm{d}x-\int g^*(x)\ln f(x,u)\mathrm{d}x\right)\end{aligned} \tag{5}$$

将式(3)代入式(5)，可以得到

$$\max_u D(u)=\max_u\left(E_u I_{\{S(X)\geqslant\gamma\}}\ln f(X,u)\right) \tag{6}$$

传统 CE 优化算法主要包括以下步骤[19]：

Step1 初始化，定义均值 μ_0 和方差 σ_0^2，令精英样本数 $N_e=\lfloor\rho N\rfloor$(ρ 是精英采样率，$\lfloor\cdot\rfloor$表示向上取整)，令迭代次数 $t=1$；

Step2 采样，根据正态分布概率密度函数 $\mathcal{N}(\mu_{t-1},\sigma_{t-1}^2)$生成 $x_1,x_2,\cdots,x_N$；

Step3 选择，计算适应度函数 $S(x_i)$，$i=1,2,\cdots,N$，对 $S(x_i)$进行排序得到 $S_{(1)}\leqslant S_{(2)}\leqslant\cdots\leqslant S_{(N)}$，选取适应度值大的 N_e 个个体，作为采集的精英样本集合；

Step4 更新参数，对于 $j=1,2,\cdots,N_e$，计算新的均值 μ_t 和方差 σ_t^2，即

$$\left.\begin{aligned}\mu_t&=\sum_{j=1}^{N_e}x_j/N_e\\\sigma_t^2&=\sum_{j=1}^{N_e}(x_j-\mu_t)^2/N_e\end{aligned}\right\} \tag{7}$$

Step5 平滑处理，平滑系数 $\alpha\in[0,1]$，更新 μ_t 和 σ_t 为

$$\left.\begin{aligned}\mu_t&=\alpha\mu_t+(1-\alpha)\mu_{t-1}\\\sigma_t&=\alpha\sigma_t+(1-\alpha)\sigma_{t-1}\end{aligned}\right\} \tag{8}$$

Step6 结果判断，如果 $\max\{\sigma_t\}<\varepsilon$($\varepsilon$ 为方差限制边界，令 $\varepsilon=10^{-6}$)，则程序结束返回参数 μ_t，否则，令 $t\leftarrow t+1$，并转到 Step2。

2 多目标交叉熵优化算法

MOCEO 算法以 CE 优化算法为基础，下面详细描述 MOCEO 算法的组成部分。

2.1 初始化

不失一般性，考虑一个包含 n 个变量、m 个目标的多目标最小化问题[20]，具体为

$$\left.\begin{aligned}&\text{寻找}\quad \boldsymbol{x}=[x_1,x_2,\cdots,x_n]\\&\text{最小化}\quad \boldsymbol{y}=F(\boldsymbol{x})=[f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\cdots,f_m(\boldsymbol{x})]\end{aligned}\right\} \tag{9}$$

式中：$x_i\in[x_{\min,i},x_{\max,i}]$，$i=1,2,\cdots,n$；$n$ 为决策变量的维数；m 为目标函数的个数。$x_{\min}=[x_{\min,1},\cdots,x_{\min,n}]$，$x_{\max}=[x_{\max,1},\cdots,x_{\max,n}]$分别是决策变量的上限和下限。

随机选取一个均值向量 $\boldsymbol{\mu}=[\mu_1,\cdots,\mu_n]$，$\mu_i\in[x_{\min,i},x_{\max,i}]$，以及一个方差向量 $\boldsymbol{\sigma}^2=[\sigma_1^2,\sigma_2^2,\cdots,\sigma_n{}^2]$，$\sigma_i{}^2=\theta(x_{\max,i}-x_{\min,i})$，$\theta$ 为控制系数，$\theta>1$，本文令 $\theta=10$。按照正态分布 $\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma}^2)$，初始种群 $\boldsymbol{P}^0=[\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_k]'$，$k=1,2,\cdots,0.3N_c$，式中 N_c 为种群大小，$\boldsymbol{p}_k=[x_{k,1},x_{k,2},\cdots,x_{k,n}]$。

2.2 快速非支配排序及适应度值的分配

生成初始种群 $\boldsymbol{P}^0$ 后，算法的主循环开始。在每次迭代过程中，首先分别计算第 t_c 代种群 $\boldsymbol{P}^{t_c}$ 中每个个体的适应度函数值，随后根据非支配排序算法确定非支配个体，并将其用于后续算法的参数更新以及个体选择。

采用外部存档机制，定义外部存档集合 $\boldsymbol{A}^{t_c}\in\mathbf{R}^{N_c\times n}$存放种群中的非支配解(或 Pareto 近似解)，定义集合 $\boldsymbol{P}'^{t_c}\in\mathbf{R}^{M\times n}$存放支配解，其中：种群大小 N_c 即为集合 $\boldsymbol{A}^{t_c}$ 的最大容量；M 是支配解的数量，它的值在每次迭代中都不同，$M\leqslant N_c$。

采用文献[14]中的快速非支配排序方法，将所有非支配解保留在集合 $\boldsymbol{A}^{t_c}$ 中，将所有支配解保存在集合 $\boldsymbol{P}'^{t_c}$ 中，由上文可知 $\boldsymbol{P}^{t_c}=\boldsymbol{A}^{t_c}\cup\boldsymbol{P}'^{t_c}$。当$|\boldsymbol{A}^{t_c}|\leqslant N_c$(本文用$|\cdot|$表示集合中解的数量)时，采用非支配序作为适应度值；当$|\boldsymbol{A}^{t_c}|>N$ 时，将相同非支配序的个体进行超体积贡献度计算，优先选取超体积贡献度更大的个体存入 $\boldsymbol{A}^{t_c}$，采用非支配序和超体积

贡献度同时作为适应度值。超体积又称勒贝格测度，是对一系列解集的一种度量，反映了在目标空间中，这些解所共同支配的空间的大小[21]。

2.3　参数更新

由于集合 $\boldsymbol{A}^{t_c}$ 中存放的个体构成了当代 Pareto 前沿面，因此需要提取这些最优个体的分布特征，并用于更新正态分布概率模型 $\mathcal{N}(\boldsymbol{\mu}_j^{t_c},\boldsymbol{\sigma}^{2t_c})$ 的参数均值向量 $\boldsymbol{\mu}_j^{t_c}$ 和方差向量 $\boldsymbol{\sigma}^{2t_c}$，以便在下一次进化中采样生成新的种群，$j=1,2,\cdots,|\boldsymbol{A}^{t_c-1}|$。

2.3.1　均值向量 $\boldsymbol{\mu}_j^t$ 的更新　在 MOCEO 算法的第 t_c 次进化中，选取集合 $\boldsymbol{A}^{t_c-1}$ 中每个个体 $\boldsymbol{x}_j$ 作为产生下一代种群个体的正态分布函数的均值向量 $\boldsymbol{\mu}_j^{t_c}$，即 $\boldsymbol{\mu}_j^{t_c}=\boldsymbol{x}_j$，式中 $\boldsymbol{x}_j\in\boldsymbol{A}^{t_c-1}$，$j=1,2,\cdots,|\boldsymbol{A}^{t_c-1}|$。

2.3.2　方差向量 $\boldsymbol{\sigma}^{2t}$ 的更新　对集合 $\boldsymbol{A}^{t_c-1}$ 每一维的变量值分别取均值，得到

$$\bar{\mu}_i^{t_c}=\sum_{j=1}^{N'}x_{j,i}^{t_c}/N' \tag{10}$$

式中 $N'=|\boldsymbol{A}^{t_c-1}|$。对应于均值 $\bar{\mu}_i^{t_c}$，可以求得集合 $\boldsymbol{A}^{t_c-1}$ 中每个维度的平均方差 $\bar{\sigma}_i^{2t_c}$，定义为

$$\bar{\sigma}_i^{2t_c}=\sum_{j=1}^{N'}(x_{j,i}^{t_c}-\bar{\mu}_i^{t_c})^2/N' \tag{11}$$

定义向量 $\bar{\boldsymbol{\sigma}}^{2t_c}=[\bar{\sigma}_1^{2t_c},\bar{\sigma}_2^{2t_c},\cdots,\bar{\sigma}_n^{2t_c}]$。

和传统的 CE 优化算法相同，MOCEO 也使用了平滑操作来提升参数的稳定性。若平滑系数 $\varphi\in[0,1]$，则新的方差向量 $\boldsymbol{\sigma}^{2t_c}$ 定义为

$$\boldsymbol{\sigma}^{2t_c}=\varphi\bar{\boldsymbol{\sigma}}^{2t_c}+(1-\varphi)\boldsymbol{\sigma}^{2t_c-1} \tag{12}$$

为了保证算法更好的跳出局部收敛，取随机数 $r\in[0,1]$，如果 $r<0.1$，则令 $\boldsymbol{\sigma}^{2t_c}=\beta\boldsymbol{\sigma}^{2t_c}$，$\beta$ 为调节系数，$\beta>1$，本文取 $\beta=100$。

2.4　进化方向更新

为了提高算法收敛到前沿面的速度，算法定义了方向向量，以采样个体综合算法收敛过程信息，引导算法加快寻优速度。本文通过比较进化过程中 10 代个体之间的差异，得到个体进化的方向信息，用以加速种群的进化过程，具体步骤如下：

Step1　每隔 10 代记录均值向量 $\bar{\boldsymbol{\mu}}^{t_c-10}$ 和 $\bar{\boldsymbol{\mu}}^{t_c}$；

Step2　定义方向向量 $\boldsymbol{d}^{t_c}=[d_1^{t_c},d_2^{t_c},\cdots,d_n^{t_c}]$，式中

$$d_i^{t_c}=\bar{\mu}_i^{t_c}-\bar{\mu}_i^{t_c-10}=\begin{cases}1, & \bar{\mu}_i^{t_c}\leqslant\bar{\mu}_i^{t_c-10}\\ -1, & \bar{\mu}_i^{t_c}>\bar{\mu}_i^{t_c-10}\end{cases} \tag{13}$$

Step3　使用 $\boldsymbol{d}^{t_c}$ 指导子代的生成，进化 10 代之后，返回 Step1。

2.5　子代的生成

更新了均值向量 $\boldsymbol{\mu}^{t_c}$ 和方差向量 $\boldsymbol{\sigma}^{2t_c}$ 以后，按照新的正态分布模型，结合方向向量 $\boldsymbol{d}^{t_c}$，通过采样产生下一代种群 $\boldsymbol{Q}^{t_c}$。具体步骤如下。

Step1　如果 $\sigma_i^{t_c}<\delta$，则转到 Step3 否则转到 Step2，为提高算法的局部搜索精度，取 $\delta=0.01$。

Step2　使用均值 $\mu_i^{t_c}$ 和方差 $\sigma_i^{2t_c}$ 按照正态分布 $\mathcal{N}(\mu_i^{t_c},\sigma_i^{2t_c})$ 生成 N_1 个个体，集合 $\boldsymbol{A}^{t_c-1}$ 中每个个体产生 $\lceil N_1/N'\rceil$（$\lceil\cdot\rceil$ 表示向下取整）个子代，剩下的 $N_1-\lceil N_1/N'\rceil N'$ 个子代由集合 $\boldsymbol{A}^{t_c-1}$ 中的个体随机生成，N_1 的计算公式为

$$N_1=\begin{cases}0.3N_c, & n_e\leqslant\eta E_{\max}\\ N_c, & n_e>\eta E_{\max}\end{cases} \tag{14}$$

式中：n_e 是适应度函数的计算次数；η 为比例系数，经实验，$\eta=0.48$ 时效果最好；$E_{\max}$ 为最大计算次数。

Step3　使用均值 $\bar{\mu}_i^{t_c}$ 和方差 $\sigma_i^{2t_c}$ 产生 N_1 个子代，具体公式为

$$\left.\begin{aligned}\boldsymbol{x}_l&=\mathcal{N}(\bar{\boldsymbol{\mu}}_i^{t_c},\boldsymbol{\sigma}^{2t_c})\\ x_{l,i}&=\begin{cases}-|\,x_{l,i}\,|, & d_i^{t_c}=1\\ |\,x_{l,i}\,|, & d_i^{t_c}=-1\end{cases}\end{aligned}\right\} \tag{15}$$

Step4　得到子代 $\boldsymbol{Q}^{t_c}=[\boldsymbol{q}_1^{t_c},\boldsymbol{q}_2^{t_c},\cdots,\boldsymbol{q}_{N_1}^{t_c}]'$，式中 $\boldsymbol{q}_l^{t_c}=[x_{l,1},x_{l,2},\cdots,x_{l,n}]$，$l=1,2,\cdots,N_1$。

2.6　个体选择

在 MOCEO 迭代初期，有 $|\boldsymbol{A}^{t_c}|\leqslant N_1$，随着算法的不断收敛，落在 Pareto 前沿面上的个体越来越多，最终导致出现 $|\boldsymbol{A}^{t_c}|>N_1$ 的情况，根据算法的要求需要删除集合 $\boldsymbol{A}^{t_c}$ 中的一些个体，使得 $|\boldsymbol{A}^{t_c}|=N_1$。对于不同层间的个体，优先选择非支配序低的个体存入 $\boldsymbol{A}^{t_c}$。对于同一层间个体的优劣度的比较，采用快速排序[21]的机制来判断计算同一层每个个体对于整个种群的超体积贡献度并排序，选取贡献度较大的个体存入 $\boldsymbol{A}^{t_c}$，直到 $|\boldsymbol{A}^{t_c}|=N_1$。由于快速超体积排序算法的复杂度会随着 N_c 的增大而呈指数级增长，为了降低算法的复杂度，减少运算时间，在算法运行前期取 $N_1=0.3N_c$，同时算法的迭代次数增加，算法的局部搜索精度提高。

2.7　结束条件

一般地，当达到算法设置的 $E_{\max}$ 时，程序停止运行，输出保存在集合 $\boldsymbol{A}^{t_c}$ 中的个体作为最优解集。

3　多目标交叉熵优化算法的测试与分析

为了验证算法的有效性和鲁棒性，分别针对二维 ZDT[21] 系列和三维 DTLZ[21] 系列目标测试函数，选用 MOCEO、NSGA-II、SPEA2、MOEAD、PAES 算法独立运算 30 次，计算超体积[22] 和反转

世代距离[16]性能指标并进行对比分析。使用 Java 编程语言在 jMetal 框架[23]中实现上述优化算法，硬件环境为 Intel i7@2.6 GHz CPU，8 GB 内存。

在进行仿真计算时，各算法的最大计算次数均设置为 25 000。在 MOCEO 算法中，当适应度函数计算次数 $n_e \leqslant 12\ 000$ 时，设置种群大小 $N_c=100$，$N_1=0.3N_c=30$，当 $12\ 000<n_e\leqslant 25\ 000$ 时，设置 $N_1=N_c=100$。其他算法种群大小均设置为 100。

3.1 不同优化算法性能指标的比较

反转世代距离指标是表征进化计算获得的近似 Pareto 面和真实 Pareto 面之间的距离指标，数值越小表明近似 Pareto 最优解越接近真实 Pareto 最优解。优劣序是不同算法在同种测试函数下的性能指标的均值和方差综合排序的结果，数值越小表明算法越优秀。表 1 和表 2 分别给出了 5 种算法分别对 ZDT 系列以及 DTLZ 系列测试函数独立运算 30 次所求得的超体积和反转世代距离性能指标的比较结果(深色单元格中的数据表示最优值，浅色单元格中的数据表示次优值)。从表 1 可以看出，在 5 种算法对 7 种测试函数的比较中，MOCEO 取得了 5 个最优值和 2 个次优值，NSGA-II、MOEAD 以及 PAES 分别只取得了 1 个次优值，SPEA2 取得了 2 个最优值和 2 个次优值。从表 2 可以看出，MOCEO 取得了 4 个最优值和 2 个次优值。综上可知：MOCEO 在超体积和反转世代距离指标上明显优于其他 4 种算法，即 MOCEO 得到的 Pareto 面具有更好的贴近度和分散性；MOCEO 在 30 次计算得到的超体积值方差很小，即 MOCEO 具有更好的鲁棒性。

表 1 不同算法对 ZDT 测试函数以及 DTLZ 测试函数优化结果超体积值的比较

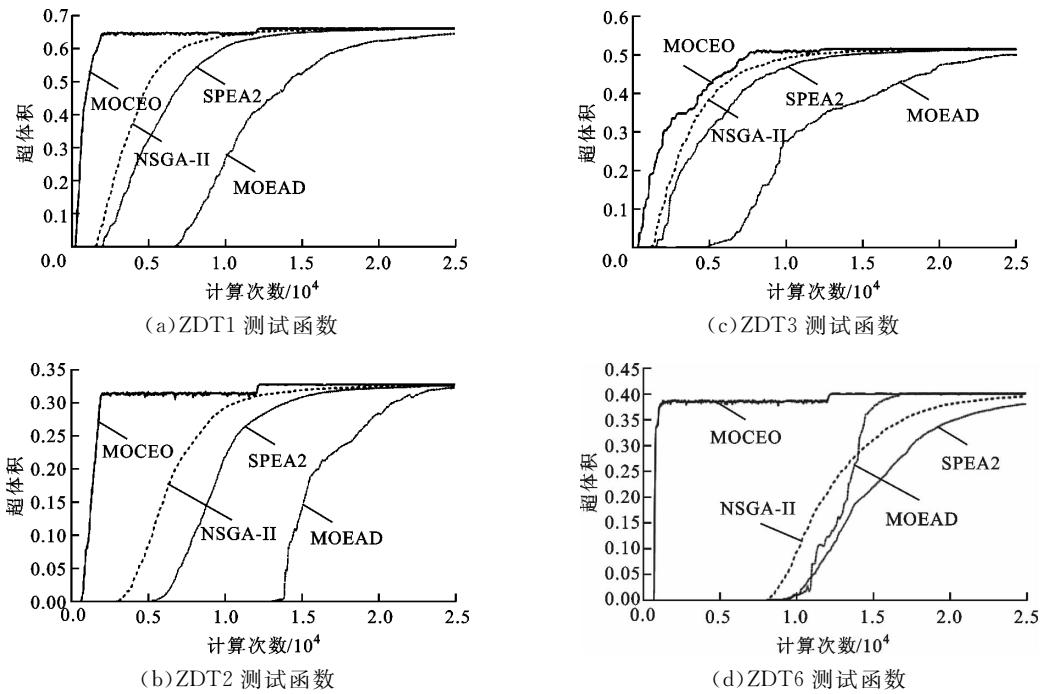
测试函数	参数	超体积				
		MOCEO	NSGA-II	SPEA2	MOEAD	PAES
ZDT1	均值	0.660 5	0.660 3	0.661 0	0.638 9	0.653 5
	方差	5.697×10^{-8}	1.065×10^{-7}	1.039×10^{-7}	2.380×10^{-4}	6.677×10^{-5}
	优劣序	2	3	1	5	4
ZDT2	均值	0.327 2	0.326 7	0.327 0	0.317 0	0.323 6
	方差	5.140×10^{-8}	1.090×10^{-7}	6.310×10^{-7}	3.910×10^{-5}	9.470×10^{-6}
	优劣序	1	3	2	5	4
ZDT3	均值	0.515 3	0.515 1	0.515 0	0.471 8	0.481 7
	方差	4.020×10^{-7}	1.840×10^{-8}	3.970×10^{-7}	6.590×10^{-4}	2.561×10^{-3}
	优劣序	1	2	3	5	4
ZDT6	均值	0.402 0	0.397 0	0.395 4	0.401 8	0.383 6
	方差	1.090×10^{-7}	3.150×10^{-6}	4.320×10^{-6}	7.040×10^{-7}	1.619×10^{-3}
	优劣序	1	3	4	2	5
DTLZ2	均值	0.430 8	0.403 0	0.451 9	0.370 9	0.196 0
	方差	2.300×10^{-6}	2.810×10^{-5}	2.250×10^{-6}	2.370×10^{-4}	5.390×10^{-3}
	优劣序	2	3	1	4	5
DTLZ5	均值	0.094 62	0.092 46	0.092 71	0.084 92	0.091 47
	方差	5.330×10^{-8}	4.960×10^{-8}	2.210×10^{-8}	1.450×10^{-6}	3.560×10^{-7}
	优劣序	1	3	2	5	4
DTLZ6	均值	0.092 72	0.039 15	0.060 79	0.074 59	0.087 44
	方差	2.060×10^{-8}	3.810×10^{-5}	2.670×10^{-6}	2.070×10^{-5}	1.270×10^{-4}
	优劣序	1	5	4	3	2
平均优劣序		1.286	3.143	2.429	4.143	4.000

3.2 不同优化算法收敛速度的比较

图 1 是 MOCEO、NSGA-II、SPEA2 以及 MOEAD 共 4 种算法得到的 ZDT 系列二维测试函数的超体积随计算次数的变化曲线，可以看出：对于 ZDT1、ZDT2，MOCEO 在 2 500 次时收敛到最优，NSGA-II 和 SPEA2 在 15 000 次左右收敛到最优，MOEAD 在 25 000 次之前没有完全收敛；对于 ZDT3，MOCEO 在 7 500 次时收敛到最优，NSGA-II

表 2　不同算法对 ZDT 测试函数以及 DTLZ 测试函数优化结果反转世代距离的比较

测试函数	参数	反转世代距离				
		MOCEO	NSGA-II	SPEA2	MOEAD	PAES
ZDT1	均值	1.20×10^{-4}	1.42×10^{-4}	1.22×10^{-4}	1.39×10^{-3}	3.88×10^{-4}
	方差	5.15×10^{-11}	4.91×10^{-11}	3.70×10^{-11}	4.23×10^{-7}	1.85×10^{-7}
	优劣序	1	3	2	5	4
ZDT2	均值	1.23×10^{-4}	1.37×10^{-4}	1.46×10^{-4}	3.83×10^{-4}	2.51×10^{-4}
	方差	8.85×10^{-11}	3.31×10^{-11}	1.54×10^{-8}	2.67×10^{-8}	1.97×10^{-8}
	优劣序	1	2	3	5	4
ZDT3	均值	8.73×10^{-5}	1.01×10^{-4}	8.26×10^{-5}	2.19×10^{-3}	2.11×10^{-3}
	方差	1.12×10^{-10}	1.93×10^{-11}	2.31×10^{-12}	1.25×10^{-6}	3.49×10^{-6}
	优劣序	2	3	1	5	4
ZDT6	均值	1.13×10^{-4}	2.29×10^{-4}	2.48×10^{-4}	1.60×10^{-5}	4.87×10^{-4}
	方差	8.25×10^{-11}	3.44×10^{-10}	1.02×10^{-9}	7.26×10^{-11}	1.27×10^{-6}
	优劣序	2	3	4	1	5
DTLZ2	均值	1.33×10^{-3}	1.61×10^{-3}	1.41×10^{-3}	1.54×10^{-3}	6.99×10^{-3}
	方差	1.46×10^{-9}	6.07×10^{-9}	2.11×10^{-10}	6.69×10^{-8}	3.30×10^{-6}
	优劣序	1	4	2	3	5
DTLZ5	均值	2.27×10^{-4}	1.93×10^{-4}	1.39×10^{-4}	4.05×10^{-4}	2.49×10^{-4}
	方差	1.77×10^{-10}	1.47×10^{-10}	7.33×10^{-12}	1.55×10^{-9}	5.43×10^{-10}
	优劣序	3	2	1	5	4
DTLZ6	均值	1.55×10^{-4}	5.26×10^{-3}	2.29×10^{-3}	1.62×10^{-4}	8.75×10^{-4}
	方差	1.88×10^{-10}	3.26×10^{-6}	1.46×10^{-7}	6.39×10^{-10}	2.47×10^{-6}
	优劣序	1	5	4	2	3
平均优劣序		1.571	3.143	2.429	3.714	4.143



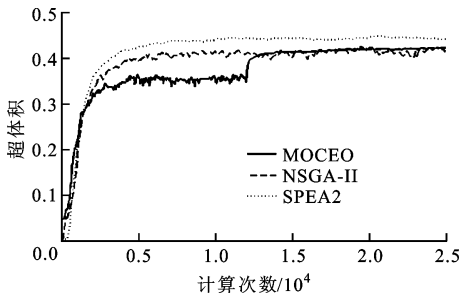
(a)ZDT1 测试函数　(c)ZDT3 测试函数

(b)ZDT2 测试函数　(d)ZDT6 测试函数

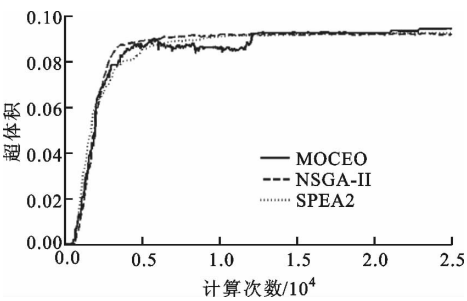
图 1　4 种算法在 4 种 ZDT 测试函数下超体积变化情况

和 SPEA2 在 12 500 次左右收敛到最优，MOEAD 没有完全收敛；对于 ZDT6，MOCEO 在 1 250 次时收敛到最优，NSGA-II 和 SPEA2 在 22 500 次左右收敛到最优，MOEAD 在 17 500 次左右收敛到最优。由于 MOCEO 算法在计算次数为 12 000 次时种群大小由 30 转变为 100，所以超体积在图中 12 000 次的位置会有一个跳变。

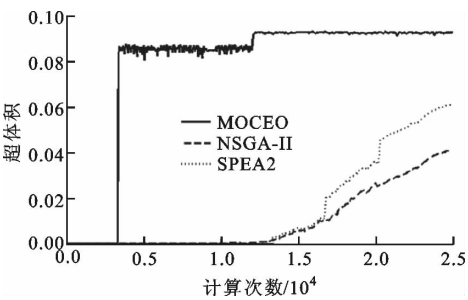
图 2 给出了 MOCEO、NSGA-II 和 SPEA2 共 3 种算法在 DTLZ2、DTLZ5 和 DTLZ6 这 3 种三维测试函数下超体积随计算次数的变化曲线，可以看出：对于 DTLZ2 和 DTLZ5，3 种算法的收敛速度基本一致；对于 DTLZ6，MOCEO 在 3 500 次时基本收敛到最优值，NSGA-II 和 SPEA2 基本没有收敛。MOCEO 由于种群数量的变化，超体积会在 12 000 次时发生跳变。



(a)DTLZ2 测试函数



(b)DTLZ5 测试函数



(c)DTLZ6 测试函数

图 2 3 种算法在 3 种 DTLZ 测试函数下超体积变化情况

综上所述，MOCEO 在 ZDT1、ZDT2、ZDT3、ZDT6 及 DTLZ6 这 5 种测试函数下较其他算法明显收敛速度更快，在 DTLZ2 和 DTLZ5 这 2 种测试函数下收敛速度与其他算法基本持平。

4 列车悬挂半主动控制系统参数优化

为了使高速列车具有更好的横向平稳性，主要采用半主动和主动悬挂的控制方法，控制器的选取及优化尤为重要。

采用文献[11]中的高速列车 17 自由度横向振动模型，以水平不平顺、方向不平顺及速度 v 为模型输入，车体横向振动合成加速度为模型输出，设置车辆运行速度 $v=300$ km/h。采用 MOCEO 算法和 NSGA-II 算法分别对模型中的广义预测控制器[24]进行优化，选择预测时域长度 P 和控制时域长度 P' 作为决策变量，选择列车在直线轨道上轮对横移量的均方根 f_1、车体横向加速度的均方根 f_2 以及列车横向平稳性指标[25] f_3 作为 3 个优化目标。设置种群大小 $N_d=50$，迭代次数 $g=100$，决策变量 $P\in(0,300]$，$P'\in(0,300]$，P 和 P' 均为正整数。

4.1 控制器参数优化结果

两种优化算法对控制器进行优化后，种群个体在空间中的分布如图 3 所示可以看出，MOCEO 算法得到的个体解可以支配 NSGA-II 得到的个体解，即 MOCEO 算法具有更好的优化效果。对于 MOCEO 算法，选取图中下边界上的点对应参数作为控制器参数的优化结果，此时 $P=13$，$P'=2$，平稳性指标 f_3 为 1.357 8。对于 NSGA-II 算法，选取图中下边界上的点对应参数作为控制器参数的优化结果，此时 $P=10$，$P'=5$，平稳性指标 f_3 为 1.416 7。可见，通过 MOCEO 算法的优化，平稳性指标由 1.416 7 减小到了 1.357 8，平稳性提高了 4.16%。

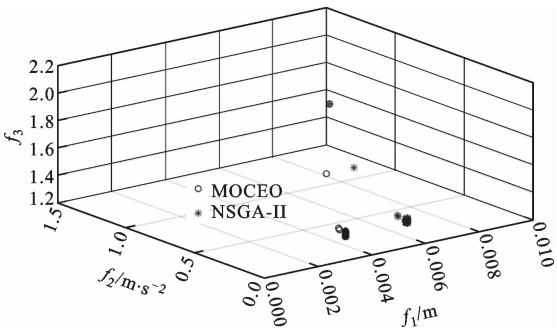
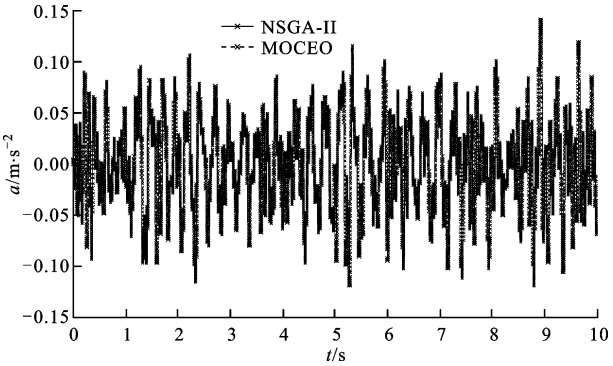


图 3 两种优化算法的解的空间分布图

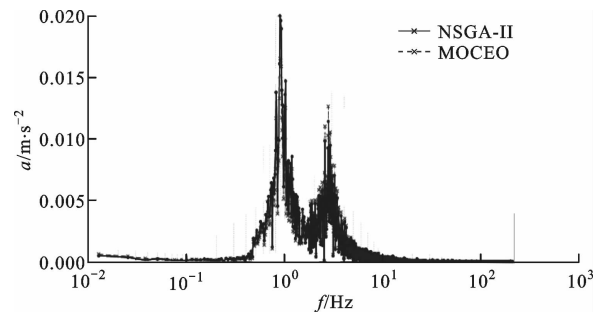
4.2 两种算法优化后悬挂系统的比较

将 MOCEO 和 NSGA-II 得到的控制参数分别代入仿真模型，得到车体横向振动加速度的时域、频谱、功率谱密度，如图 4 所示，a 表示加速度，t 表示

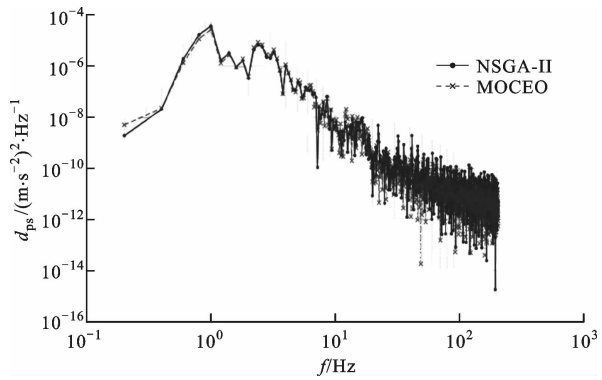
时间，f 表示频率，d_{ps} 表示功率密度。由图 4a 看出，车体横向加速度峰值由 0.145 m/s^2 减小到 0.13 m/s^2，下降了 10.34%。可见采用 MOCEO 算法优化后列车的横向平稳性得到较大改善，旅客乘坐舒适性提高。由图 4b 和图 5c 可以看出，车体横向振动加速度主要集中在 0.5～5 Hz，而人体对横向振动最敏感频率范围为 1～2 Hz，相比于 NSGA-II 算法，采用 MOCEO 算法优化后，在该频率范围内车体横向振动加速度明显改善。



(a)时谱



(b)频谱



(c)功率谱密度

图 4　车体横向加速度时域、频域、功率谱密度

5　结　论

(1)针对多目标优化问题，引入进化方向，设置新的子代生成方式，提出一种改进的 CE 优化算法 MOCEO。ZDT 系列和三维 DTLZ 系列目标测试函数的测试结果证明，MOCEO 是一种快速、有效的多目标优化算法。

(2)将 MOCEO 应用于某型高速列车横向稳定控制系统的参数优化，仿真结果表明，车体横向加速度峰值以及平稳性指标均得到改善，较 NSGA-II 算法取得了更优的横向平稳性和舒适性。

(3)算法种群大小 N_c 和平滑操作系数 α 是 MOCEO 的主要参数，对算法性能有着重要影响。采用动态种群大小策略，即在进化不同阶段分别设置不同种群的大小，可以使算法更好地平衡探索和获得更优的开采性能。

(4)今后一方面将研究种群规模自适应调整变化策略对算法性能的影响，另一方面将研究平滑系数 α 的设置方式及不同的概率模型对算法性能的影响。

参考文献：

[1] CELESTE F, DAMBREVILLE F, LE CADRE J P. Optimal path planning using cross-entropy method [C] // 2006 9th International Conference on Information Fusion. Piscataway, NJ, USA: IEEE, 2006: 1-8.

[2] JALOCON C L C, NERVES A C. Renewable energy portfolio planning using the cross entropy method [C] // Proceedings of 2013 IEEE PES Asia-Pacific Power and Energy Engineering Conference. Piscataway, NJ, USA: IEEE, 2013: 1-5.

[3] CASERTA M, RICO E Q, URIBE A M. A cross entropy algorithm for the Knapsack problem with setups [J]. Computers & Operations Research, 2008, 35(1): 241-252.

[4] HABER R E, DEL TORO R M, GAJATE A. Optimal fuzzy control system using the cross-entropy method: a case study of a drilling process [J]. Information Sciences, 2010, 180(14): 2777-2792.

[5] BOER P T D, KROESE D P, MANNOR S, et al. A tutorial on the cross-entropy method [J]. Annals of Operations Research, 2005, 134(1): 19-67.

[6] AN S, YANG S, HO S L, et al. An improved cross-entropy method applied to inverse problems [J]. IEEE Transactions on Magnetics, 2012, 48(2): 327-330.

[7] LI J, CHAI T Y, GONG J K. Design of PID controller using cross entropy method [J]. Control and Decision, 2011, 26(5): 794-796.

[8] GIAGKIOZIS I, PURSHOUSE R C, FLEMING P J. Generalized decomposition and cross entropy methods

for many-objective optimization [J]. Information Sciences, 2014, 282: 363-387.

[9] 边莉，车向前，张少卿. 基于改进交叉熵算法多目标不等间距阵列综合 [J]. 上海交通大学学报，2014，48(3)：372-376.
BIAN Li, CHE Xiangqian, ZHANG Shaoqing. Multi-objective unequally-spaced array synthesis based on modified cross entropy algorithm [J]. Journal of Shanghai Jiaotong University, 2014, 48(3): 372-376.

[10] BERUVIDES G, QUIZA R, HABER R E. Multi-objective optimization based on an improved cross-entropy method: a case study of a micro-scale manufacturing process [J]. Information Sciences, 2016, 334: 161-173.

[11] 陈春俊. 高速列车横向主动/半主动悬挂控制研究 [D]. 成都：西南交通大学，2006：49-65.

[12] 孟宏. 提速机车车辆横向运动稳定性研究及应用 [D]. 成都：西南交通大学，2009：10-20.

[13] 周洪涛，杨绍普，朱红霞. 基于模糊控制的高速机车横向半主动控制研究 [J]. 振动与冲击，2011，30(9)：145-149.
ZHOU Hongtao, YANG Shaopu, ZHU Hongxia. Fuzzy control strategy applied in lateral semi-active control of the high-speed locomotive [J]. Journal of Vibration and Shock, 2011, 30(9): 145-149.

[14] DEB K, PRATAP A, AGAREAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[15] ZIZTLER E, LAUMANNS M, THIELE L. SPEA2: improving the strength Pareto evolutionary algorithm for multi-objective optimization: TIK-Report 103 [R]. Zurich, Switzerland: Swiss Federal Institute of Technology, 2001: 1-21.

[16] 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.

[17] KNOWLES J D, CORNE D W. Approximating the nondominated front using the Pareto archived evolution strategy [J]. Evolutionary Computation, 2000, 8(2): 149-172.

[18] RUBINSTEIN R Y. Optimization of computer simulation models with rare events [J]. European Journal of Operational Research, 1997, 99(1): 89-112.

[19] KROESE D P, POROTSKY S, RUBINSTEIN R Y. The cross-entropy method for continuous multi-extremal optimization [J]. Methodology and Computing in Applied Probability, 2006, 8(3): 383-407.

[20] 邓涛，林椿松，李亚南，等. 采用 NSGA-II 算法的混合动力能量管理控制多目标优化方法 [J]. 西安交通大学学报，2015，49(10)：143-150.
DENG Tao, LIN Chunsong, LI Yanan, et al. A multi-objective optimization method for energy management control of hybrid electric vehicles using NSGA-II algorithm [J]. Journal of Xi'an Jiaotong University, 2015, 49(10): 143-150.

[21] ZITZLER E, DEB K, THIELE L. Comparison of multi-objective evolutionary algorithms: empirical results [J]. Evolutionary Computation, 2000, 8(2): 173-195.

[22] RUSSO L M S, FRANCISCO A P. Quick hypervolume [J]. IEEE Transactions on Evolutionary Computation, 2014, 18(4): 481-502.

[23] DURILLO J J, NEBRO A J. jMetal: a Java framework for multi-objective optimization [J]. Advances in Engineering Software, 2011, 42(10): 760-771.

[24] 李中奇，杨振村，杨辉，等. 高速列车双自适应广义预测控制方法 [J]. 中国铁道科学，2015，36(6)：120-127.
LI Zhongqi, YANG Zhencun, YANG Hui. Generalized predictive control with dual adaptation method of high speed train [J]. China Railway Science, 2015, 36(6): 120-127.

[25] 杨世杰，陈建政. Matlab 在机车车辆动力学试验数据处理中的应用 [J]. 中国测试技术，2006，32(3)：26-28.
YANG Shijie, CHEN Jianzheng. Data processing in the dynamic test of rail vehicle based on Matlab [J]. China Measurement & Test, 2006, 32(3): 26-28.

（编辑 陶晴）