第34卷 第13期 计 算 机 工 程 2008年7月
Vol.34 No.13 Computer Engineering June 2008
·人工智能及识别技术· 文章编号：1000—3428(2008)13—0193—02 文献标识码：A 中图分类号：TP301.6

# 基于改进DE算法的难约束优化问题的求解

贺毅朝[1]，王熙照[2]
(1. 石家庄经济学院信息工程学院，石家庄 050031；2. 河北大学数学与计算机学院，保定 071002)

**摘 要：**基于指数函数的性质，提出简易罚函数法(SPFM)，用于有效求解难约束优化问题(COP)，并屏蔽选取罚因子的困难性。将SPFM和差分演化相结合，给出一种求解难COP的改进差分演化算法(MDE)。利用MDE求解Bump问题可以得出该问题的多个新的最优解，证明MDE在求解难COP时的高效性。

**关键词：**差分演化；约束优化；罚函数法；Bump问题

# Solution of Hard Constrained Optimization Problem Based on Modified Differential Evolution Algorithm

HE Yi-chao[1], WANG Xi-zhao[2]
(1. Information Engineering School, Shijiazhuang University of Economics, Shijiazhuang 050031;
2. College of Mathematic and Computer, Hebei University, Baoding 071002)

**【Abstract】** For solving hard Constrained Optimization Problem(COP), this paper proposes a new method named Simple Penalty Function Method (SPFM) based on properties of exponent function. SPFM avoids the difficulty of choosing the penalty factors. Modified Differential Evolution algorithm(MDE) is advanced, which combines SPFM with Differential Evolution(DE). By using MDE to solve Bump problem, more better optimization solutions gained by MDE shows that MDE is effective.

**【Key words】** Differential Evolution(DE); constrained optimization; Penalty Function Method(PFM); Bump problem

在科学研究、工程技术和资源分配等领域中，大量实际问题最终归结为约束优化问题(Constrained Optimization Problem, COP)[1]，如何有效求解此类问题是演化计算领域中一个重要的研究方向。目前，求解COP的算法有多种，如遗传算法(GA)[1]和粒子群算法[2]。差分演化(Differential Evolution, DE)[3-4]是由Rainer和Price于1997年提出的一种高效演化算法，在首届IEEE演化大赛中表现超群，引起国内外学者的广泛关注。目前，DE已应用于多目标优化、神经网络训练和系统控制等领域[4]。

## 1 背景知识

### 1.1 差分演化算法

差分演化[3-4]是一种基于实数编码的高效演化算法，在算法的每一代演化迭代过程中利用差异算子和选择算子进行演化，差异算子基于3个随机选取的差异个体向量进行交叉重组来产生一个中间种群，选择算子基于择优选取原则从中间种群和父代种群中选择最优个体生成子代种群。DE的一般原理[4]如下：

设目标函数为min$f(X)$，搜索空间$S=\{X|X=(x_1,x_2,\cdots,x_d)\wedge L_i\le x_i\le U_i,\ 1\le i\le d\}\subseteq R^d$，$R$为实数集。在DE中，$X_i(t)=(x_{i1}(t),x_{i2}(t),\cdots,x_{id}(t))\in S$表示第$t$代种群中的第$i$个个体，$V_i(t)=(v_{i1}(t),v_{i2}(t),\cdots,v_{id}(t))\in S$表示第$t$代中间种群的第$i$个个体，种群规模设为$s$，则差异算子(Differential Operator, DO)定义为

$$v_{ij}(t+1)=\begin{cases}x_{p_1j}(t)+\alpha(x_{p_2j}(t)-x_{p_3j}(t)) & r\le CR\text{或}\ j=R(i)\\ x_{ij}(t) & r>CR\text{或}\ j\ne R(i)\end{cases}\quad(1)$$

其中，$p_1\ne p_2\ne p_3\ne i$, $1\le i\le s$,$1\le j\le d$；$\alpha<1$，称为缩放因子；$r$是随机数且$r\sim(0,1)$；$CR$是一个常数因子，称为交叉概率，且$CR\in[0,1]$；$R(i)$是$[1,n]$上的一个正随机整数。

选择算子(Selection Operator, SO)定义为

$$X_i(t+1)=\begin{cases}V_i(t+1) & f(V_i(t+1))<f(X_i(t))\\ X_i(t) & \text{否则}\end{cases}\quad(2)$$

若以$X_{\text{best}}$表示DE演化过程中的全局最优个体，则DE算法描述如下：

**算法1** 差分演化算法

(1)随机生成初始种群$P(0)=\{X_i(0)|1\le i\le s\text{且}X_i(0)\in S\}$。

(2)置$t=0$，最大迭代次数置$N_{\max}$。

(3)由差异算子(即式(1))生成中间种群$\{V_i(t+1)|\ V_i(t+1)\in S\text{且}\ 1\le i\le s\ \}$。

(4)由选择算子(即式(2))生成种群$P(t+1)=\{X_i(t+1)|\ X_i(t+1)\in S\text{且}\ 1\le i\le s\ \}$。

(5)计算全局最优个体$X_{\text{best}}=\min\{X_i(t+1)|X_i(t+1)\in P(t+1)\text{且}\ 1\le i\le s\ \}$。

(6)当$t<N_{\max}$时，置$t=t+1$并转步骤(3)；否则输出$(X_{\text{best}}, f(X_{\text{best}}))$，算法结束。

### 1.2 约束优化问题及其常见处理方法

约束优化问题[1]是一种非线性规划问题，若以最小化问题描述，则COP的一般形式为

**基金项目：**国家自然科学基金资助重点项目(60473045)
**作者简介：**贺毅朝(1969－)，男，副教授、硕士，主研方向：智能计算，计算机密码学，计算复杂性理论；王熙照，教授、博士、博士生导师
**收稿日期：**2007-08-20 **E-mail：**heyichao@sjzue.edu.cn

$$\min f(X),\ X\in D$$
$$\text{s.t.}\begin{cases} g_i(X)\le 0 & i=1,2,\cdots,q \\ h_k(X)=0 & k=q+1,q+2\cdots,m \end{cases} \tag{3}$$

其中，$D=\prod_{i=1}^{n}[a_i,b_i]\subseteq R^n$；$X=(x_1,x_2,\cdots,x_n)$且$x_i\in[a_i,\ b_i]$；$S=\{X|X\in D\wedge g_i(X)\le 0\wedge h_k(X)=0\}$称为可行解区域；$f: D\to R$称为目标函数；$g_i(X)$是$q$个不等式约束条件；$h_k(X)$是$m$-$q$个等式约束条件。

COP是一种求解较为困难的优化问题，当问题的维数很高且目标函数和约束条件为非线性时，更难求解，目前尚无普遍有效的求解方法。利用EA求解COP的关键在于对约束条件的处理，目前常见方法有 4 类[5]：抛弃不可行解法，基于智能编码方法，修补方法和罚函数法。罚函数法是最常用的处理方法，具有较好的通用性。在利用罚函数法求解COP时，一般定义如下新目标函数[1]：

$$F(X)=\begin{cases} f(X) & \text{若}X\in S \\ f(X)+penalty(X) & \text{否则} \end{cases} \tag{4}$$

其中，$penalty(X),\ X\in D$ 为一个惩罚函数。由此可将式(3)转化为求解 min$F(X)$的全局最小优化问题。

## 2　求解难 COP 的改进差分演化算法

在应用罚函数法求解COP时，关键是$penalty(X)$中罚因子的适当选取，根据罚因子选取方法的不同，罚函数法又分为：静态罚函数法、动态罚函数法、退火罚函数法、分化罚函数法和DCPM方法等 5 种[1,5]。但无论哪一种罚函数法，罚因子的选取都是困难的，往往要依赖使用者的经验或通过大量的尝试才能确定，这为此方法的使用增加了许多难度。DCPM方法[5]虽然不使用罚因子，但仍需要设置 2 个限制因子$\varepsilon$和$\varepsilon'$，而$\varepsilon$和$\varepsilon'$的选取存在诸多不便。下面基于函数$e^x$的一般性质，提出一种不含任何待定因子的简单有效的改进方法：简易罚函数法(Simple Penalty Function Method, SPFM)。

**定义 1** 对于式(3)描述的COP，定义函数$\Phi: D\to R$，且对于任意$X=(x_1,x_2,\cdots,x_n)\in D$有

$$\Phi(X)=\sum_{i=1}^{q}G_i(X)+\sum_{k=q+1}^{m}H_k(X)$$

其中，$i=1,2,\cdots,q$, $k=q+1, q+2,\cdots,m$。

$$G_i(X)=\begin{cases} 1/2 & g_i(X)\le 0 \\ 1/(1+e^{g_i(X)}) & \text{否则} \end{cases}$$

$$H_k(X)=\begin{cases} 1/2 & h_k(X)=0 \\ 1/(1+e^{|h_k(X)|}) & \text{否则} \end{cases}$$

称函数$\Phi(X)$的值为个体$X$的满足约束度。显然，当个体$X\in S$时，$\Phi(X)=m/2$；当$X\notin S$时，$0<\Phi(X)<m/2$，并且$X$偏离$S$越远，$\Phi(X)$越小。为了比较$X_1$与$X_2$的优劣，还需要定义一个$Prior$函数。

**定义 2** 对于最小约束优化问题，设$X_1$与$X_2$为 2 个不同的个体，定义函数

$$Prior(X_1,X_2)=\begin{cases} 1 & \Phi(X_1)=\Phi(X_2)\text{且}f(X_1)<f(X_2) \\ & \text{或}\Phi(X_1)>\Phi(X_2) \\ 0 & \text{其他} \end{cases} \tag{5}$$

若$Prior(X_1, X_2)=1$，表示$X_1$优于$X_2$；若$Prior(X_1, X_2)=0$，表示$X_1$劣于$X_2$。

由于$G_i(X)$和$H_k(X)$的计算只与$g_i(X)$和$h_k(X)$相关，不依赖于任何因子，因此$Prior(X_1, X_2)$的计算也不依赖于任何因子。这样，利用$Prior$函数判定 2 个体的优劣，完全克服了前面所述方法对于罚因子或因子的依赖性，为处理约束条件提供了一种新方法。为将SPFM和DE相结合，必须利用$Prior$函数修改DE中选择算子的定义，以判定两个体的优劣。

**定义 3** 设$X_i(t+1)$和$X_i(t)$分别为第$t+1$ 代和第$t$代种群中第$i$个个体，$V_i(t+1)$是第$t+1$ 代中间种群的第$i$个中间个体，则改进选择算子定义如下：

$$X_i(t+1)=\begin{cases} V_i(t+1) & Prior(V_i(t+1),(X_i(t))=1 \\ X_i(t) & \text{否则} \end{cases} \tag{6}$$

显然，利用改进的选择算子可以方便地对个体进行比较，将其与 DE 结合，得到如下求解难 COP 的改进差分演化算法：

**算法 2** 改进差分演化算法

(1)随机生成初始种群$P(0)=\{X_i(0)|1\le i\le s\text{且}X_i(0)\in D\}$。

(2)置$t=0$，最大迭代次数置$N_{\max}$。

(3)由差异算子(式(1))生成中间种群$\{V_i(t+1)|1\le i\le s\text{且}V_i(t+1)\in D\}$。

(4)由改进选择算子(式(6))生成种群：

$P(t+1)=\{X_i(t+1)|1\le i\le s\text{且}X_i(t+1)\in D\}$

(5)计算全局最优个体：

$X_{\text{best}}=\min\{X_i(t+1)|X_i(t+1)\in P(t+1)\text{且 }1\le i\le s\}$

(6)当$t<N_{\max}$时，置$t=t+1$，转步骤(4)；否则输出($X_{\text{best}}$, $f(X_{\text{best}})$)，算法结束。

## 3　实例计算与比较

为了验证算法 2 的有效性和可行性，以下利用MDE求解著名的Bump问题[1,6]。运行环境为DELL Pentium4 CPU 1.69 GHz微机，用C++语言编程实现。在求解Bump问题时，MDE的参数设置为：缩放因子$\alpha=0.5$，交叉概率$CR=0.3$，$N_{\max}=10\ 000$，并且重复计算 20 次。

Bump 问题可以描述为

$$\min f(X)=-\left|\frac{\sum_{i=1}^{n}\cos^4(x_i)-2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n}ix_i^2}}\right|$$
$$\text{s.t.}\begin{cases} \sum_{i=1}^{n}x_i\le 7.5n \\ \prod_{i=1}^{n}x_i\ge 0.75 \end{cases}$$

其中，$X=(x_1,x_2,\cdots,x_n)$且 $0\le x_i\le 10$，$i=1,2,\cdots,n$。

由于Bump问题是具有“三超”(超非线性、超多峰、超高维)的难约束优化问题，因此成为国际上衡量EA优劣的通用的Benchmark函数[6]。目前，Bump问题的最优解未知，当维数$n=20$ 时，已知的最好解为-0.803 619[6]。下面利用MDE求解 20 维数Bump问题，在表 1 中列出了所得到的多个优于-0.803 619 的解。

**表 1　MDE 算法求得优于-0.803 619 的部分最好解**

| min $f(X)$ | min $f(X)$对应的点的坐标$X=(x_1,x_2,\cdots,x_{20})$ |
|---|---|
| -0.803 619 104 060 | 3.162 469 958 730, 3.128 335 267 582, 3.094 793 638 677, 3.061 447 887 202, 3.027 925 002 215, 2.993 826 855 054, 2.958 665 459 642, 2.921 852 913 576, 0.494 822 211 204, 0.488 367 835 872, 0.482 314 361 608, 0.476 647 297 794, 0.471 301 171 105, 0.466 226 806 094, 0.461 419 679 851, 0.456 837 485 573, 0.452 455 178 940, 0.448 257 911 583, 0.444 247 797 135, 0.440 383 469 863 |
| -0.803 619 104 044 | 3.162 470 459 833, 3.128 332 721 786, 3.094 796 029 979, 3.061 450 020 694, 3.027 929 126 303, 2.993 817 352 013, 2.958 667 841 357, 2.921 831 755 504, 0.494 823 678 721, 0.488 365 210 252, 0.482 320 795 485, 0.476 653 393 627, 0.471 294 004 413, 0.466 226 356 085, 0.461 412 651 523, 0.456 829 118 147, 0.452 460 420 829, 0.448 262 860 591, 0.444 245 064 409, 0.440 390 939 608 |
| -0.803 619 103 787 | 3.162 470 209 904, 3.128 318 452 844, 3.094 807 563 115, 3.061 439 147 103, 3.027 922 960 160, 2.993 834 942 114, 2.958 662 768 310, 2.921 829 125 301, 0.494 827 032 090, 0.488 369 705 412, 0.482 299 831 622, 0.476 646 118 897, 0.471 307 327 317, 0.466 222 577 155, 0.461 416 000 148, 0.456 809 371 906, 0.452 455 271 322, 0.448 288 941 478, 0.444 261 411 897, 0.440 381 337 414 |

将MDE与FPDCGA算法[5]、PSODE算法[6]进行比较，表 2 给出了它们的比较结果。显然，MDE的结果最优，而且从方差来看，其鲁棒性也非常好。

（下转第 217 页）