软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(7):1688−1699 [doi: 10.13328/j.cnki.jos.004623]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于内部罚函数的进化算法求解约束优化问题*

崔承刚, 杨晓飞

(中国科学院 上海高等研究院,上海 201210)

通讯作者: 崔承刚, E-mail: nanfeicui@sina.com

摘 要: 为解决现有约束处理方法可行解的适应度函数不包含约束条件的问题,提出了一种内部罚函数候选解筛选规则.该候选解筛选规则分别对可行解和不可行解采用内部罚函数和约束违反度进行筛选,从而达到平衡最小化目标函数和满足约束条件的目的.以进化策略算法为基础,给出了基于内部罚函数候选解筛选规则的进化算法的一个实现.进一步地,从理论和实验角度分别验证了内部罚函数候选解筛选规则的有效性:以(1+1)进化算法为例,从进化成功率方面验证了内部罚函数候选解筛选规则的理论有效性;通过 13 个测试问题的数值实验,从进化成功率、候选解后代是可行解的比例、进化步长和收敛速度方面验证了内部罚函数候选解筛选规则的实验有效性.

关键词: 约束优化问题;进化算法;内部罚函数筛选规则;进化策略

中图法分类号: TP18

中文引用格式: 崔承刚,杨晓飞.基于内部罚函数的进化算法求解约束优化问题.软件学报,2015,26(7):1688−1699. http://www.jos.org.cn/1000-9825/4623.htm

英文引用格式: Cui CG, Yang XF. Interior penalty rule based evolutionary algorithm for constrained optimization. Ruan Jian Xue Bao/Journal of Software, 2015,26(7):1688−1699 (in Chinese). http://www.jos.org.cn/1000-9825/4623.htm

Interior Penalty Rule Based Evolutionary Algorithm for Constrained Optimization

CUI Cheng-Gang, YANG Xiao-Fei

(Shanghai Advanced Research Institute, The Chinese Academy of Sciences, Shanghai 201210, China)

Abstract: In order to combine constraints into the evaluation of feasible solutions, a set of interior penalty rules is proposed to improve the efficiency of evolutionary algorithms in solving the constrained optimization problems. In these rules, interior penalty functions are used to evaluate feasible solutions and constraint violations are used to evaluate infeasible solutions. In addition, an interior penalty rule based evolution strategy algorithm is derived to solve constrained optimization problems. The theory validity of these rules is analyzed based on the successful rate of an (1+1) evolutionary algorithm. The proposed approach is tested with 13 benchmark problems. The results indicate that the presented approach is competitive with two existing state-of-the-art techniques.

Key words: constrained optimization; evolutionary algorithm; interior penalty function rule; evolution strategy

进化算法是求解约束优化问题的主要启发式搜索方法之一[1].由于不采用约束优化问题的任何信息,仅通过估计产生解的质量来指导搜索过程,被认为是“盲目型启发式”算法[1].为了提高进化算法求解约束优化问题的效率,研究人员提出了很多将约束条件结合到进化算法的方法[2].罚函数法是将约束条件结合到进化算法适应度函数求解约束优化问题的常用方法[3],该类方法是一种通用的约束条件处理方法,它通过候选解的约束条件违反程度对候选解进行惩罚.基于可行解优于不可行解启发式规则的约束处理方法是另外一种将约束条件结合到进化算法的方法[4,5],该类方法假设可行解的适应度函数值优于不可行解的适应度函数值.因此,该方法使得不可行解难以进入进化算法的种群,这样,就使得该方法在求解最优解位于可行域边界的约束优化问题时

* 基金项目: 住房和城乡建设部科学技术项目(2013-K8-25); 中国科学院知识创新工程重要方向项目(KGCX2-EW-321); 国家国际科技合作项目(2010DFB13040); 国家科技支撑计划(2012BAH43F03)

收稿时间: 2013-04-02; 修改时间: 2013-11-28; 定稿时间: 2014-04-09

可能存在问题.基于多目标优化方法求解约束优化问题近些年受到了极大的重视[6],该方法根据将约束优化问题转换为多目标优化问题,并采用多目标优化方法来求解问题.上述方法具有一个共同的特点:即,可行解只通过目标函数值进行评价.这样,当进化算法采用上述方法时,由于约束条件并不包含在可行解的适应度函数中,这就使得种群中候选解产生可行解的概率较低,从而使得进化算法早熟收敛.这意味着需要付出额外的代价产生可行解,从而降低了搜索过程的效率.

为了解决进化算法中可行解的适应度函数不包含约束条件的问题,本文在简单候选解筛选规则[5]的基础上,提出了一种内部罚函数候选解筛选规则.该筛选规则通过内部罚函数对靠近约束条件边界的可行解进行惩罚,从而使得进化算法能够平衡最小化目标函数和满足约束条件这两个冲突的目标;通过约束违反度评价不可行解,从而使进化算法能够迅速找到可行解.为了使进化算法有效地利用这种筛选规则,本文采用对数形式的内部罚函数使其迅速收敛;采用基于积极约束条件辨识的动态惩罚因子使罚函数值迅速减小;采用自适应松弛方法将等式约束条件转换为不等式约束条件.本文提出的方法是一种通用的计算框架,可适用于各种类型的进化算法.文中以多成员进化策略算法[7]为基础,给出了一个采用该候选解筛选规则的进化算法的实例,称为内部罚函数进化策略算法.最后,本文以(1+1)进化算法为例,根据候选解的进化成功率,分析了该筛选规则的理论有效性.通过与现有约束优化问题求解算法的实验比较,从进化成功率、候选解后代是可行解的比例、进化步长和收敛速度这 4 个方面验证了本文方法的有效性.

1 基于内部罚函数候选解筛选规则的进化算法

1.1 问题描述

约束优化问题的一般形式可表达如下:

$$\min f(\boldsymbol{x}) \tag{1}$$

其满足以下 m 个约束条件:

$$g_j(\boldsymbol{x})\leqslant 0, j=1,\ldots,q \tag{2}$$

$$h_j(\boldsymbol{x})=0, j=q+1,\ldots,m \tag{3}$$

其中,$\boldsymbol{x}=[x_1,\ldots,x_n]^T$ 为决策变量,$f(x)$为目标函数,q 为不等式约束条件个数,$m-q$ 为等式约束条件个数.如果在候选解 $\boldsymbol{x}$ 处 $g_j(\boldsymbol{x})=0$,则约束条件 $g_j(\boldsymbol{x})\leqslant 0$ 称为候选解 $\boldsymbol{x}$ 的积极约束条件.

1.2 内部罚函数[8]

内部罚函数法在传统非线性优化方法中被广泛应用,主要原理是:在求解优化问题最优解的搜索过程中,平衡最小化目标函数和满足约束条件这两个互相冲突的要求.其特点是:将构造的惩罚函数定义于可行域内,并在可行域内求惩罚函数的最优解.即,求解无约束优化问题时的搜索过程总是保持在可行域内部.

对于给定约束优化问题(1)~问题(3),内部罚函数的基本表达式如下:

$$\phi(x,r(t))=f(x)+r(t)B(x) \tag{4}$$

其中,$B(x)$为惩罚项,且当候选解 x 趋向于可行域边界时,$B(x)$趋近于无穷大;$r(t)$为惩罚因子,是单调递减的正数序列,即,$r(0)>r(1)>\ldots>r(t)>r(t+1)>\ldots>0$,且 $\lim_{t\to\infty}r(t)=0$.

随着惩罚因子 $r(t)$逐渐减小,可通过求解以下无约束优化问题(5)得到约束优化问题(1)~问题(3)的最优解:

$$\lim_{t\to\infty}\min\phi(x,r(t)) \tag{5}$$

当 $t\to\infty$时,$\lim_{t\to\infty}r(t)=0$.因此,函数$\phi(x,r(\infty))$的取值等于 $f(x)$.

1.3 内部罚函数候选解筛选规则

当采用进化算法求解约束优化问题(1)~问题(3)时,为了有效地平衡求解约束优化问题过程中最小化目标函数和满足约束条件这两个冲突的要求,本文根据 Deb[4]提出的候选解的简单候选解筛选规则设计了内部罚函数候选解筛选规则,具体做法是:

(1) 当两个候选解都是可行解时,选择内部罚函数值小的候选解作为优胜者;

(2) 当两个候选解之一是可行解时,选择可行的候选解作为优胜者;

(3) 当两个候选解都是不可行解时,选择约束违反度低的候选解作为优胜者.

上述候选解筛选规则具有以下特性:

(1) 由于内部罚函数的定义域为约束优化问题的可行域,因此,内部罚函数不能应用于不可行域.这样,采用该方法的进化算法就无法处理不可行解.规则(2)采用可行解优于不可行解的启发式规则,这就使得筛选规则能够分别对不可行解和可行解采用不同的筛选方法;

(2) 由于规则(3)仅根据约束违反度评价不可行解而忽略了目标函数,这样,通过与精英保留策略的结合,该规则能够保证进化算法找到一个可行解.因此,该规则对强约束优化问题非常有效[9],这就使筛选规则可以应用到没有初始可行解的强约束优化问题当中;

(3) 内部罚函数对靠近可行域边界的可行解进行惩罚.这样,距离可行域边界近的可行解的内部罚函数值较大.由于内部罚函数候选解筛选规则偏向于内部罚函数值小的候选解,即,偏向于选择距离可行域边界较远的可行解,因此在进化算法选择算子的作用下,筛选规则可使进化算法避免违反约束条件.

下面给出筛选规则的内部罚函数的形式、惩罚因子和等式约束条件的处理方法等实现细节.

1.3.1 内部罚函数的形式

在传统数学规划方法中,对数罚函数法具有超线性收敛特性[10].因此,为了获得较好的收敛特性,本文在公式(4)定义的内部罚函数中采用如下对数形式的惩罚项 $B(x)$:

$$B(x)=-\sum_{i=1}^{m}\ln(-v_i(x)) \tag{6}$$

其中,$v_i(x)$是规范化后的约束函数 $g_i(x)$,定义如下:

$$v_i(x)=\frac{g_i(x)}{|\min g_i(x)|} \tag{7}$$

其中,$\min g_i(x)$表示进化算法搜索过程中约束函数 $g_i(x)$在候选解 x 处取得的最小值.

1.3.2 惩罚因子更新策略

内部罚函数针对最优解的积极约束条件进行惩罚[11].由于最优解的积极约束条件难以判断,因此,常规的进化算法很少采用内部罚函数法[11].但是,最优解的积极约束条件具有以下性质:在可行域内,如果目标函数沿搜索方向 d 单调减少(或单调增加),则沿搜索方向 d 单调增加(或单调减少)的约束条件中至少包含一个最优解的积极约束条件[12].根据最优解的积极约束条件的上述性质,可以在最优解未知的条件下,通过分析目标函数和约束函数之间的相关性判别哪些约束条件可能是最优解的积极约束条件.本文将这类可能的最优解积极约束条件称为最优解的候选积极约束条件,并将除此之外的约束条件称为最优解的非候选积极约束条件.

辨识最优解候选积极约束条件的具体步骤为:采用进化算法种群中所有个体的约束条件函数值和目标函数值作为样本,每隔一定迭代次数计算样本的 Spearman 秩相关性系数[13].如果约束条件函数与目标函数的 Spearman 秩相关性系数为负数,则该约束条件为最优解的候选积极约束条件.

由于最优解的候选积极约束条件描述了在最优解处可能起到积极作用的约束条件,因此与非候选积极约束条件相比,最优解的候选积极约束条件对避免搜索过程违反约束条件的作用更大.故,本文对不同的约束条件采用不同的惩罚因子,并将公式(4)的内部罚函数扩展为

$$\phi(x,\boldsymbol{r}(t))=f(x)-\sum_{i=1}^{m}r_i(t)\ln(-v_i(x)) \tag{8}$$

其中,$\boldsymbol{r}(t)=\{r_i(t)\}_{i=1,2,\ldots,m}$.$r_i(t)$为约束条件 $g_i(\boldsymbol{x})\leqslant 0$ 的惩罚因子,它采用以下方式进行更新:对最优解的候选积极约束条件采用较大的乘积因子δ_1 更新惩罚因子;对最优解的非候选积极约束条件采用较小的乘积因子δ_2 更新惩罚因子.即:

$$r_i(t+1)=\begin{cases}\delta_1 r_i(t), & \chi_{g_i}\leqslant 0\\ \delta_2 r_i(t), & \text{其他}\end{cases} \tag{9}$$

其中,χ_{g_i} 为约束条件函数 $g_i(\boldsymbol{x})$与目标函数的 Spearman 秩相关性系数,$0<\delta_2<\delta_1<1$.

上述惩罚因子更新方法采用较大的乘积因子更新最优解候选积极约束条件的惩罚因子,使得最优解候选积极约束条件的惩罚因子值下降较慢,从而可避免违反该类约束条件.

1.3.3 等式约束条件处理

进化算法求解包含等式约束条件的约束优化问题时,通常采用以下方法将等式约束条件松弛为不等式约束条件:

$$g_j=|h_j|-\varepsilon_j\leqslant 0 \tag{10}$$

其中,$\varepsilon_j\geqslant 0$ 为等式约束条件的容忍值,且为一个很小的正数.上述松弛方法采用很小的ε_j 值,因此,松弛后的约束优化问题可行域占搜索域的比例通常比较小.当进化算法采用内部罚函数候选解筛选规则时,规则(1)仅可以应用到可行解,这必将极大地降低内部罚函数候选解筛选规则的作用.

为了提高规则(1)的应用范围,本文采用自适应松弛法来处理等式约束条件[14].ε_j 的具体松弛公式如下:

$$\begin{aligned}&\text{If } (R_n(t)\leqslant R_l) \text{ Then } \varepsilon_j(t+1)=\beta_l\varepsilon_j(t)\\ &\text{If } (R_n(t)\geqslant R_u) \text{ Then } \varepsilon_j(t+1)=\beta_u\varepsilon_j(t)\end{aligned} \tag{11}$$

其中,$R_n(t)$为当代种群中满足松弛后约束优化问题可行解的比例,$0\leqslant R_l\leqslant R_u\leqslant 1, 0\leqslant\beta_l\leqslant 1\leqslant\beta_u$,$\varepsilon_j(0)$等于初始种群中候选解约束违反度 $G(x)$的最大值.

由于等式约束条件的函数值都接近ε_j,因此不再对等式约束条件函数 $g_j(x)$进行规范化.

1.4 基于进化策略算法的实现

进化策略算法(evolution strategy,简称 ES 算法)是一类采用实数编码在连续空间中进行随机搜索的进化算法,广泛应用于求解各种优化问题并具有良好的理论基础[15].本文在求解约束优化问题的多成员进化策略算法(simple multimembered evolution srategy,简称 SMES)[7]的基础上,利用内部罚函数候选解筛选规则替换该算法采用的简单候选解筛选规则,从而将内部罚函数候选解筛选规则应用到进化策略算法中,以验证该筛选规则的有效性.该算法称为内部罚函数进化策略算法(interior penalty based evolution strategy,简称 IPES).

在 IPES 算法中,个体可以看作(x_i,σ_i)的集合,其中,i 为种群中的任意个体,x_i 为 n 维决策变量,σ_i 为 n 维步长变量.初始种群通过 n 维均匀分布在搜索空间中产生.离散交叉和中间交叉的混杂交叉算子、参数变异算子和保留不可行解的多样性策略被用于提高进化策略算法的搜索能力,其中,变异算子不仅应用于决策变量,也应用于步长.具体计算公式如下:

$$\sigma_i'=\sigma_i\cdot\exp(\tau'\cdot N(0,1)+\tau\cdot N_i(0,1)) \tag{12}$$

$$x_i'=x_i+\sigma_i'\cdot N_i(0,1) \tag{13}$$

其中,τ和τ'为“学习率”,并根据 Schwefel[16]提出的方法进行计算:$\tau=(\sqrt{2\sqrt{n}})^{-1}$ 和 $\tau'=(\sqrt{2n})^{-1}$;$N_i(x,y)$是以 x 为均值、以 y 为方差的实数高斯随机分布.

为了引入内部罚函数候选解筛选规则,本文将该算法的候选解评价部分加以修改,将可行解和不可行解的适应度函数值分开进行计算.其中,可行解的适应度函数值根据内部罚函数进行计算,不可行解的适应度函数值根据约束违反度进行计算.然后,根据内部罚函数候选解筛选规则选择μ个父代个体和λ个子代个体中最好的μ个个体保留到下一代中.

IPES 算法的具体计算步骤如下:

(1) 根据决策变量的取值范围,产生包含μ个个体的初始种群,设 t=0,每个个体对应一个向量(x_i,σ_i);

(2) 计算初始种群μ个个体的适应度函数值.其中,根据约束违反度计算不可行解的适应度函数值,根据内部罚函数计算可行解的适应度函数值;

(3) 通过重组算子产生λ个个体;

(4) 根据公式(12)和公式(13)对λ个个体进行变异,产生λ个子代个体;

(5) 计算产生的λ个子代个体的适应度函数值.其中,根据约束违反度计算不可行解的适应度函数值,根据内部罚函数计算可行解的适应度函数值;

(6) 将父代个体和子代个体组成一个(μ+λ)的种群,然后按照内部罚函数候选解筛选规则选择最好的μ个个体作为下一代的父代个体;

(7) 当满足 t%p=0(其中,%为取余运算)时,计算约束条件与目标函数的 Spearman 秩相关性系数,并根据公式(9)进行惩罚因子更新;

(8) 如果满足停止准则,算法终止;否则 t=t+1,回到步骤(3).

通过上述算法步骤可知:内部罚函数候选解筛选规则仅修改了进化策略算法的候选解适应度函数估计方法和候选解筛选规则,并不修改进化策略算法的交叉、变异和选择等基本遗传算子.因此,内部罚函数候选解筛选规则不仅可以应用于进化策略算法,也可以应用于其他进化算法.

2 理论有效性分析

Karmer[17]通过实验和理论分析指出:当求解最优点位于可行域边界的约束优化问题时,由于采用目标函数作为可行解适应度函数的进化算法进化成功率很低,因此,进化算法将会早熟收敛.内部罚函数筛选规则以简单筛选规则为基础,为了验证该规则的有效性,本文分析了采用简单筛选规则的步长自适应(1+1)进化算法(简称(1+1)SFEA 算法)和采用内部罚函数筛选规则的步长自适应(1+1)进化算法(简称(1+1)IPEA 算法)在积极约束条件边界的进化成功率和早熟收敛特性.其中,进化成功率表示进化算法中后代个体替代父代个体的概率(即产生更好解的概率)[17].

2.1 进化过程

考虑马尔可夫过程模型(x_t,σ_t),(1+1)SFEA 算法的后代 x_{t+1} 和步长σ_{t+1} 通过公式(14)和公式(15)生成.

$$x_{t+1}=\begin{cases}x_t+\sigma_t Z_t, & \text{if } f(x_t+\sigma_t Z_t)<f(x_t) \text{ and } g_i(x_t+\sigma_t Z_t)\leqslant 0\\ x_t, & \text{otherwise}\end{cases} \tag{14}$$

$$\sigma_{t+1}=\begin{cases}\gamma\sigma_t, & \text{if } f(x_t+\sigma_t Z_t)<f(x_t) \text{ and } g_i(x_t+\sigma_t Z_t)\leqslant 0\\ \gamma^{-1}\sigma_t, & \text{otherwise}\end{cases} \tag{15}$$

(1+1)IPEA 算法后代 x_{t+1} 和步长σ_{t+1} 通过公式(16)和公式(17)生成.

$$x_{t+1}=\begin{cases}x_t+\sigma_t Z_t, & \text{if } \phi(x_t+\sigma_t Z_t)<\phi(x_t) \text{ and } g_i(x_t+\sigma_t Z_t)\leqslant 0\\ x_t, & \text{otherwise}\end{cases} \tag{16}$$

$$\sigma_{t+1}=\begin{cases}\gamma\sigma_t, & \text{if } \phi(x_t+\sigma_t Z_t)<\phi(x_t) \text{ and } g_i(x_t+\sigma_t Z_t)\leqslant 0\\ \gamma^{-1}\sigma_t, & \text{otherwise}\end{cases} \tag{17}$$

其中,γ>1 表示变异参数;随机向量 Z_t,$t\geqslant 0$,独立且均匀分布.

假设变异产生的新个体在以σ_t 为半径、x_t 为圆心的圆上(称变异产生的候选解组成的圆为变异生成圆).当进化策略算法产生一个成功变异时,步长σ_t 增加;否则,步长σ_t 减小.其中,成功变异表示产生的新个体优于父代个体的变异.以该圆形模型分析这两种算法的进化成功率,其中,进化算法从可行域内部开始进化,且进化成功率的大小等于由成功变异的候选解组成的弧的弧长 l=θr 与变异生成圆的周长 2πr 的比值.

$$p_s=\theta/2\pi \tag{18}$$

2.2 典型情况分析

考虑一个包含一个线性约束条件和一个线性目标函数的约束优化问题.

给定候选解 $\boldsymbol{x}$.设过点 $\boldsymbol{x}$ 的简单筛选规则的可行解适应度值的等值线与约束条件 $g(x)\leqslant 0$ 边界的夹角为α_f,内部罚函数筛选规则的可行解适应度函数的等值线与约束条件 $g(x)\leqslant 0$ 边界的夹角为α_ϕ.根据公式(9)可知:内部罚函数筛选规则的可行解适应度函数值与简单筛选规则的可行解适应度值之间的差值为

$$\varphi(x)=-\sum_{i=1}^{m}r_i(t)\ln(-v_i(x)),$$

且$\varphi(x)$与候选解到可行域边界的距离成正比.因此,简单筛选规则的可行解适应度值的等值线与约束条件$g(x)\leqslant 0$ 边界的夹角小于内部罚函数筛选规则的可行解适应度函数的等值线与约束条件 $g(x)\leqslant 0$ 边界的夹角,即,$\alpha_\phi>\alpha_f$.

根据 Karmer[17]的分析可知,可行解在约束条件边界的变异情况可分为如下 3 种.

(1) $\sigma_t<d_t$,即,候选解的变异生成圆与约束条件边界相离,也就是说,候选解的变异生成圆不受约束条件的影响;

(2) $\sigma_t>d_t,\sigma_t>s$,即,候选解的变异生成圆与适应度函数等值线在可行域外相割;

(3) $\sigma_t>d_t,\sigma_t<s$,即,候选解的变异生成圆与适应度函数等值线在可行域内相割.

其中,d_t 表示候选解到约束条件边界的距离,α为过点 x 的适应度函数等值线与约束条件边界的夹角,$s=d_t/\sin\alpha$是适应度函数等值线在候选解与约束条件边界之间的线段的长度.

由于情况(1)的变异不受约束条件的影响,因此我们不考虑这种情况.

对于情况(2)、情况(3),由于α_f和α_ϕ不相等,因此 s_f与 s_ϕ也不相等,且 $s_\phi<s_f$.所以,本文仅分析(1+1)SFEA 算法和(1+1)IPEA 算法在步长σ_t 相等和 d_t 相等且$\sigma_t>d_t$ 的情况下的进化成功率.图 1 给出了(1+1)SFEA 算法和(1+1)IPEA 算法在步长σ_t 相等和 d_t 相等且$\sigma_t>d_t$ 情况下(1+1)SFEA 算法和(1+1)IPEA 算法变异的 3 种典型情况.其中,$g(x)=0$ 表示约束优化问题的约束条件边界,$g(x)=0$ 上面的区域为可行域,下面的区域为不可行域;x 为约束优化问题的任意可行解,虚线为过点x的目标函数的等值线,点划线为过点x的内部罚函数的等值线;α_f为过点x的目标函数的等值线与约束条件边界的夹角,α_ϕ为过点 x 的内部罚函数的等值线与约束条件边界的夹角,且$\alpha_\phi>\alpha_f$.为了便于阅读,我们用σ代替σ_t,d 代替 d_t.

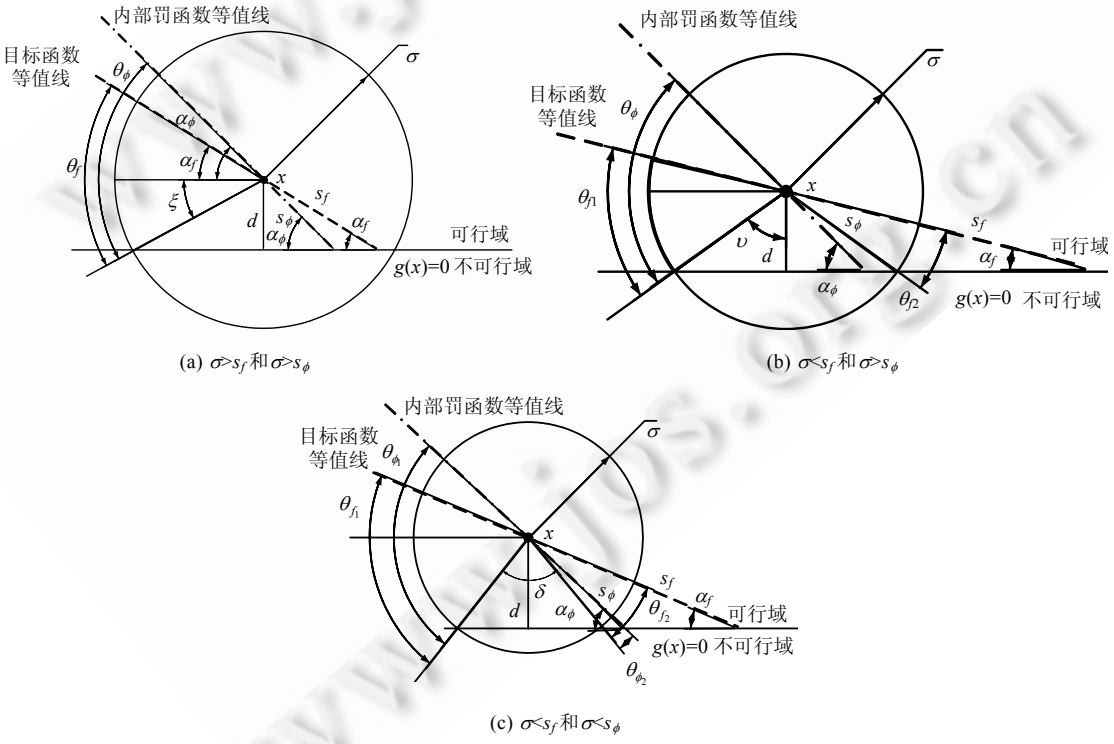

Fig.1 Success rates of (1+1)SFES and (1+1)IPES

图 1 (1+1)SFES 算法和(1+1)IPES 算法的进化成功率示意图

2.2.1 $\sigma>s_f$和$\sigma>s_\phi$

图1(a)给出了$\sigma>s_f$和$\sigma>s_\phi$时,即,候选解的变异生成圆与目标函数等值线和内部罚函数等值线都在可行域内相割时,(1+1)SFEA 算法和(1+1)IPEA 算法的进化成功率示意图.其中,采用简单筛选规则的候选解成功变异的后代组成的圆弧为θ_f,采用内部筛选规则的候选解成功变异的后代组成的圆弧为θ_ϕ.借助角ξ,可以得到$p_{s,f}=\dfrac{\alpha_f+\xi}{2\pi}$和$p_{s,\phi}=\dfrac{\alpha_\phi+\xi}{2\pi}$,其中,$\xi=\arcsin\dfrac{d}{\sigma}$.由于$\alpha_\phi>\alpha_f$,因此,$p_{s,\phi}>p_{s,f}$,且:

$$p_{s,\phi}-p_{s,f}=\frac{\alpha_\phi+\xi}{2\pi}-\frac{\alpha_f+\xi}{2\pi}=\frac{\alpha_\phi-\alpha_f}{2\pi} \tag{19}$$

2.2.2 $\sigma<s_f$和$\sigma>s_\phi$

图1(b)给出了$\sigma<s_f$和$\sigma>s_\phi$时,即,候选解的变异生成圆与目标函数等值线在可行域内相割、与内部罚函数等值线在可行域外相割时,(1+1)SFEA 算法和(1+1)IPEA 算法的进化成功率示意图.其中,采用简单筛选规则的候选解成功变异的后代组成的圆弧由θ_{f_1}和θ_{f_2}组成,采用内部筛选规则的候选解成功变异的后代组成的圆弧为θ_ϕ.借助角υ,可以得到$p_{s,f}=\dfrac{1}{2}-\dfrac{2\upsilon}{2\pi}$和$p_{s,\phi}=\dfrac{1}{2}-\dfrac{\upsilon+\frac{\pi}{2}-\alpha_\phi}{2\pi}$,其中,$\upsilon=\arccos\dfrac{d}{\sigma}$.

由于$\sigma>s_\phi$,因此,$\cos\upsilon=\dfrac{d}{\sigma}<\dfrac{d}{s_\phi}=\sin\alpha_\phi=\cos\left(\dfrac{\pi}{2}-\alpha_\phi\right)$,所以$p_{s,\phi}>p_{s,f}$,且:

$$p_{s,\phi}-p_{s,f}=\left(\frac{1}{2}-\frac{\upsilon+\frac{\pi}{2}-\alpha_\phi}{2\pi}\right)-\left(\frac{1}{2}-\frac{2\upsilon}{2\pi}\right)=\frac{\upsilon+\alpha_\phi-\pi/2}{2\pi}=\frac{\arccos(d/\sigma)+\alpha_\phi}{2\pi}-\frac{1}{4} \tag{20}$$

2.2.3 $\sigma<s_f$和$\sigma<s_\phi$

图1(c)给出了$\sigma<s_f$和$\sigma<s_\phi$时,即,候选解的变异生成圆与目标函数等值线和内部罚函数等值线都在可行域外相割时,(1+1)SFEA 算法和(1+1)IPEA 算法的进化成功率示意图.其中,采用简单筛选规则的候选解成功变异的后代组成的圆弧为θ_{f_1}和θ_{f_2},采用内部筛选规则的候选解成功变异的后代组成的圆弧为θ_{ϕ_1}和θ_{ϕ_2}.借助角δ可得$p_{s,f}=p_{s,\phi}=\dfrac{1}{2}-\dfrac{\delta}{2\pi}$,其中,$\delta=\arccos\dfrac{d}{\sigma}$.因此,这时(1+1)SFES 算法和(1+1)IPES 算法的进化成功率相同.

通过上述 3 种典型情况的分析可知:对于适应度等值线和约束条件可局部线性化的约束优化问题,当候选解靠近可行域边界时,采用内部罚函数筛选规则的(1+1)进化算法的进化成功率大于采用简单筛选规则的(1+1)进化算法的进化成功率,从而使进化算法能够避免早熟收敛.但是,该方法对一般性约束优化问题以及相对其他约束处理方法的理论有效性而言,还需要进一步深入加以研究.

3 算法测试

3.1 测试问题与算法参数

为了测试本文所研究的算法对各类约束优化问题的求解性能,本文对常见的 13 个测试问题[18]进行了算法测试.IPES 算法中的进化策略算法的参数均采用 Coello[7]中的设置:父代种群个数μ=100,子代种群个数λ=300,最大迭代次数为 800,算法停止时整个计算过程中计算适应度函数的总次数(称为适应度函数估计次数)为240 000.步长的初始值通过以下公式进行计算:

$$\sigma_i(0)=0.4\times\left(\frac{\Delta x_i}{\sqrt{n}}\right) \tag{21}$$

其中,$\Delta x_i=\overline{x}_i-\underline{x}_i$,$n$ 表示决策变量个数.内部罚函数候选解筛选规则的参数设置如下:初始惩罚因子为 $r_i(0)$=1;约束条件乘积因子δ_1=0.9 和δ_2=0.7;惩罚因子更新间隔迭代次数 p=10;等式约束条件处理参数为 R_l=0.25,R_u=0.75,β_l=0.618 和β_u=1.382.

3.2 与现有约束优化问题求解算法的比较

本文对IPES算法与现有的基于随机排序的约束进化策略算法(SR)[18]和SMES算法[7]求解以上测试问题的数值结果进行了比较.其中,SR 算法参数采用文献[18]中的设置和 SMES 算法的参数采用文献[7]中的设置.表 1 分别给出了 3 种算法求解以上测试问题的最好值、平均值和最差值的数值结果.对于测试问题 g02,g06,g07,g09 和 g10,IPES 算法在最好值、平均值和最差值等性能指标方面都优于 SR 算法和 SMES 算法;对于测试问题 g03, g04,g08,g11 和 g12,IPES 算法、SR 算法和 SMES 算法每次运行都能找到已知的最好值;对于测试问题 g01,虽然 IPES 算法未能找到已知的最好值,但是 IPES 算法也能够找到非常接近已知最好值的结果;对于测试问题 g05 和 g13,虽然 IPES 算法的求解结果略差于 SR 算法,但是 IPES 算法的适应度函数估计次数为 240 000,而 SR 算法的适应度函数估计次数为 350 000,因此,IPES 算法的计算代价小于 SR 算法.从上述结果可以看出,IPES 算法具有良好的有效性和鲁棒性.

Table 1 Comparison of the solutions found by our IPES against the SR, the SMES

表 1 IPES 算法与 SR 算法和 SMES 算法求解测试问题的比较

Prob	Optimal	Stat	Methods		
			SR	SMES	IPES
g01	−15.000	Best	**−15.000**	**−15.000**	−14.999
		Mean	**−15.000**	**−15.000**	−14.999
		Worst	**−15.000**	**−15.000**	−14.999
g02	−0.803 619	Best	−0.803 515	−0.803 601	−0.803 607
		Mean	−0.781 975	−0.785 238	−0.792 771
		Worst	−0.726 288	−0.751 322	−0.769 198
g03	1.000	Best	**1.000**	**1.000**	**1.000**
		Mean	**1.000**	**1.000**	**1.000**
		Worst	**1.000**	**1.000**	**1.000**
g04	−30 665.539	Best	**−30 665.539**	**−30 665.539**	**−30 665.539**
		Mean	**−30 665.539**	**−30 665.539**	**−30 665.539**
		Worst	**−30 665.539**	**−30 665.539**	**−30 665.539**
g05	5 126.497	Best	**5 126.497**	5 126.599	**5 126.498**
		Mean	5 128.881	5 174.492	5 139.003
		Worst	5 142.472	5 304.167	5 197.991
g06	−6 961.814	Best	**−6 961.814**	**−6 961.814**	**−6 961.814**
		Mean	−6 875.940	−6 961.284	**−6 961.814**
		Worst	−6 350.262	−6 952.482	**−6 961.814**
g07	24.306	Best	24.307	24.327	24.307
		Mean	24.374	24.475	24.316
		Worst	24.642	24.843	24.333
g08	−0.095 825	Best	**−0.095 825**	**−0.095 825**	**−0.095 825**
		Mean	**−0.095 825**	**−0.095 825**	**−0.095 825**
		Worst	**−0.095 825**	**−0.095 825**	**−0.095 825**
g09	680.630	Best	**680.630**	680.632	**680.630**
		Mean	680.656	680.643	**680.630**
		Worst	680.763	680.719	**680.630**
g10	7 049.248	Best	7 054.316	7 051.903	7 051.341
		Mean	7 559.192	7 253.047	7 210.360
		Worst	8 835.655	7 638.366	7 376.721
g11	0.75	Best	**0.75**	**0.75**	**0.75**
		Mean	**0.75**	**0.75**	**0.75**
		Worst	**0.75**	**0.75**	**0.75**
g12	1.000	Best	**1.000**	**1.000**	**1.000**
		Mean	**1.000**	**1.000**	**1.000**
		Worst	**1.000**	**1.000**	**1.000**
g13	0.053 950	Best	0.053 957	0.053 986	0.053 950
		Mean	0.067 543	0.166 385	0.146 26
		Worst	0.216 915	0.468 294	0.453 029

3.3 算法动态特性

为了进一步分析 IPES 算法的性能,本文以测试问题中 IPES 算法和 SMES 算法求解数值结果不同的 8 个测

试问题(g01,g02,g05,g06,g07,g09,g10 和 g13)为例,从进化成功率、候选解的后代个体是可行解的比例、进化步长和收敛速度这 4 个方面比较了 IPES 算法和 SMES 算法的动态性能.

3.3.1 进化成功率

为了进一步验证内部罚函数候选解筛选规则的有效性,本文对 IPES 算法和 SMES 算法求解上述测试问题的种群平均进化成功率进行了比较.图 2 给出了两种算法求解上述测试问题 30 次独立运行的种群平均进化成功率的比较结果,其中,横坐标为适应度函数估计次数,纵坐标为种群平均进化成功率.从图 2 可知:对于测试问题 g01,g05,g06,g07,g09,g10 和 g13,在具有相同的适应度函数估计次数情况下,IPES 算法的种群平均进化成功率均大于 SMES 算法的种群平均进化成功率.这意味着,进化策略算法采用内部罚函数候选解筛选规则求解约束优化问题能够提高算法的进化成功率.

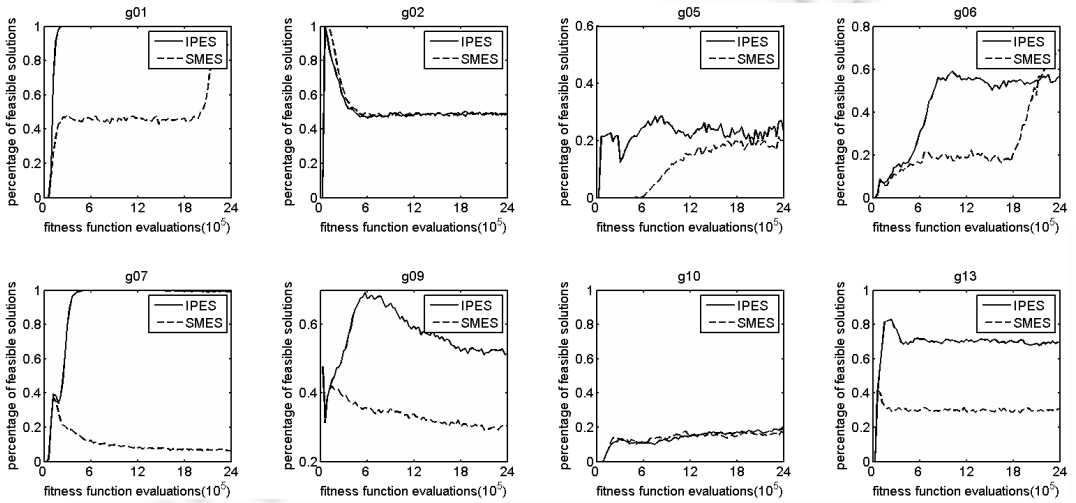

Fig.2 Comparison of the feasible rates of the IPES and the SMES

图 2 IPES 算法和 SMES 算法的候选解的后代个体是可行解的平均比例比较

由于测试问题 g02 的可行域占搜索域的比例较大(ρ量度为 99.9973%),在具有相同的适应度函数估计次数情况下,IPES 算法和 SMES 算法的种群平均进化成功率相近,且都大于 50%.这说明当约束优化问题可行域占搜索域的比例较大时,由于约束条件对候选解的进化影响较小,因此,采用基于内部罚函数候选解筛选规则的进化算法和采用简单候选解筛选规则的进化算法的进化成功率相同.

3.3.2 候选解的后代个体是可行解的比例

根据第 2 节的分析可知:采用内部罚函数筛选规则能够提高进化算法的进化成功率的主要因素是,内部罚函数提高了候选解的后代个体是可行解的概率.为了验证该结论,本文对 IPES 算法和 SMES 算法每次迭代时候选解的后代个体是可行解的比例进行了比较.图 3 给出了两种算法求解上述测试问题 30 次独立运行时候选解的后代个体是可行解的平均比例,其中,横坐标为适应度函数估计次数,纵坐标为候选解的后代个体是可行解的平均比例.由图 3 可知:对于测试问题 g01,g05,g06,g07,g09 和 g13,在具有相同的适应度函数估计次数情况下,IPES 算法候选解的后代个体是可行解的平均比例大于 SMES 算法候选解的后代个体是可行解的平均比例.

由于测试问题 g02 的可行域占搜索域的比例较大(ρ量度为 99.9973%),在具有相同的适应度函数估计次数情况下,IPES 算法与 SMES 算法候选解的后代个体是可行解的平均比例相近,都接近 50%.这说明,当约束优化问题可行域占搜索域的比例较大时,采用内部罚函数候选解筛选规则的进化策略算法和采用简单候选解筛选规则的进化策略算法候选解的后代个体是可行解的概率相似.测试问题 g10 与 g02 的结果相似,在具有相同的迭代次数情况下,IPES 算法和 SMES 算法候选解的后代个体是可行解的平均比例也相近.这是因为该问题的搜索域较大,进化策略算法很难找到可行解.

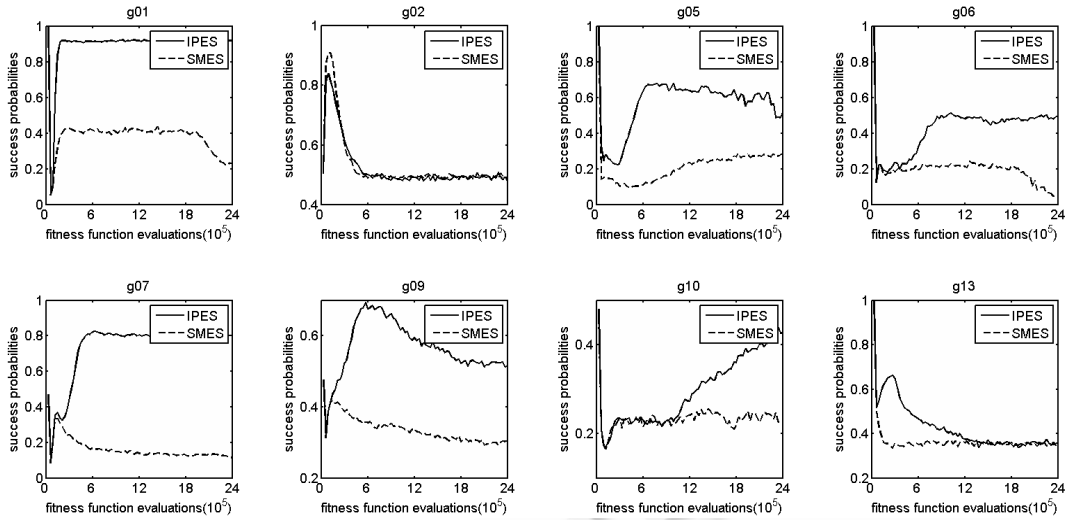

Fig.3 Comparison of the success rates of the IPES and the SMES

图 3 IPES 算法和 SMES 算法的种群平均进化成功率比较

3.3.3 进化步长

根据第 2 节的分析可知:内部罚函数筛选规则将提高进化算法的进化成功率,从而将提高候选解的进化步长进化速度,并使得进化算法避免早熟收敛.本文对 IPES 算法和 SMES 算法每次迭代时候选解的归一化进化步长进行比较.图 4 给出了两种算法求解上述测试问题 30 次独立运行时候选解的平均进化步长,其中,横坐标为适应度函数估计次数,纵坐标为候选解的平均进化步长.由图 4 可知:对于测试问题 g01,g06 和 g13,在具有相同的适应度函数估计次数情况下,IPES 算法和 SMES 算法两者的进化步长近似相等;对于测试问题 g02,g05,g07,g09 和 g10,在具有相同的适应度函数估计次数情况下,IPES 算法的进化步长小于 SMES 算法;对于测试问题 g05,g07,g09 和 g10,SMES 算法在进化过程中,当进化到一定代数后,SMES 算法的进化步长基本不再进化.因此可以看出,SMES 算法在求解这些问题时发生了早熟收敛.与此同时,对于这些问题,IPES 算法的进化步长持续减小,并未发生早熟收敛.因此,这也验证了内部罚函数筛选规则能够提高进化步长进化并避免进化算法早熟收敛的结论.

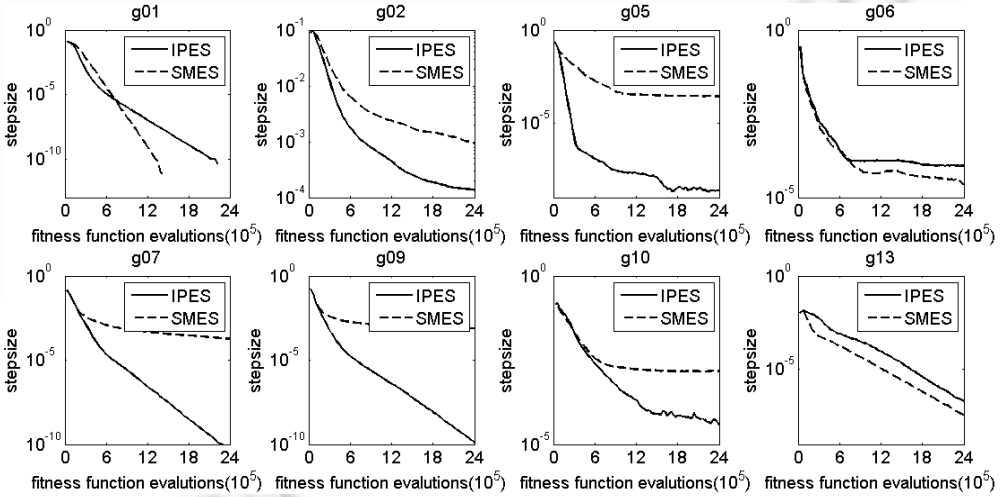

Fig.4 Comparison of the mean stepsizes of the IPES and the SMES

图 4 IPES 算法和 SMES 算法的种群平均进化平均步长比较

3.3.4 收敛速度

为了进一步比较两种算法的性能,本文对 IPES 算法和 SMES 算法的收敛速度进行了比较.图 4 给出了两种算法收敛速度的比较结果,其中,横坐标为适应度函数估计次数,纵坐标为当前种群最好值与问题最优解的差值.从图 5 可知:对于测试问题 g02,g05,g06,g07,g09 和 g13,IPES 算法的精确性均优于 SMES 算法.在搜索初期(适应度函数估计次数小于 30 000),IPES 和 SMES 这两种算法都具有较快的收敛速度;在搜索后期(适应度函数估计次数大于 30 000),SMES 算法陷入了局部极小点,而 IPES 算法仍然在继续搜索.因此,若给定足够的适应度函数估计次数,则 IPES 算法能够找到问题的最优解或者近似最优解.对于测试问题 g10,由于该问题的搜索域较大,因此,两种算法在最大适应度函数估计次数内找到的最好值相近.

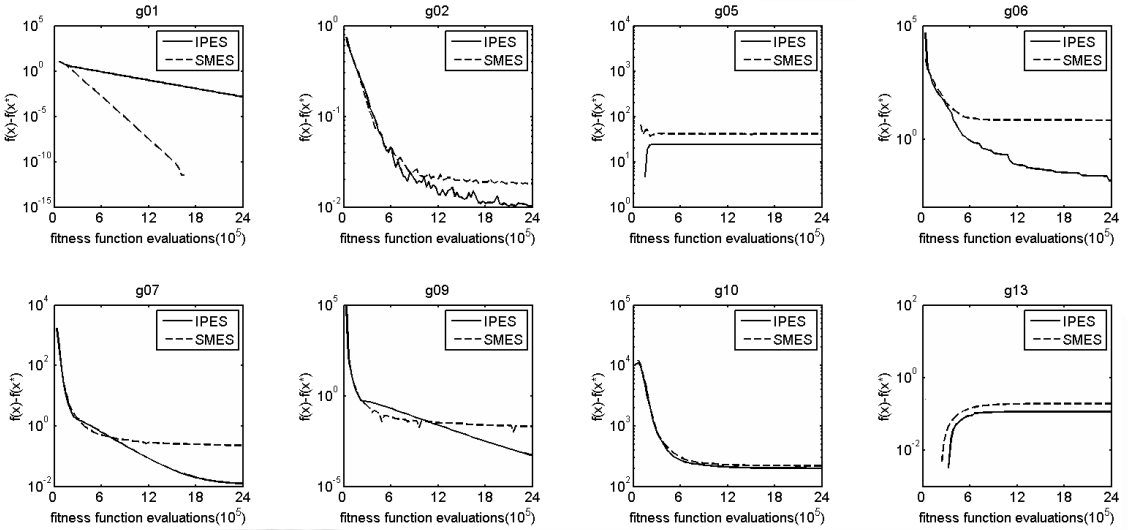

Fig.5 Comparison of the convergence of the IPES and the SMES

图 5 IPES 算法和 SMES 算法的收敛速度比较

通过上述两种算法 4 方面性能的比较可知:采用内部罚函数候选解筛选规则能够增大进化算法候选解的后代个体是可行解的概率,提高进化算法的进化成功率,进而提高进化算法进化步长的进化概率.由于内部罚函数候选解筛选规则能够提高进化算法的进化成功率,这使得进化算法避免了早熟收敛,并最终使进化策略算法找到更好的结果.因此,采用内部罚函数候选解筛选规则能够显著提高进化算法求解约束优化问题的性能.

4 结 语

为了解决可行解的适应度函数不考虑约束条件的问题,提出了一种内部罚函数候选解筛选规则.该筛选规则分别采用内部罚函数和约束违反度来筛选可行解和不可行解,从而达到平衡最小化目标函数和满足约束条件的目的.本文通过对数内部罚函数、动态惩因子更新策略和自适应松弛等式约束条件处理方法,使得该候选解筛选规则能够有效地应用到进化算法中,并以多成员进化策略算法为例,给出了采用基于内部罚函数候选解筛选规则的进化算法的实现.进一步地,本文根据进化成功率分析了采用内部罚函数候选解筛选规则的理论有效性.最后,通过与多成员进化策略算法在进化成功率、候选解的后代个体是可行解的比例、进化步长和收敛速度这 4 个方面的比较,从实验角度验证了采用基于内部罚函数候选解筛选规则的进化算法求解约束优化问题的有效性.

References:

[1] Mezura-Montes E, Coello CAC. Constraint-Handling in nature-inspired numerical optimization: Past, present and future. Swarm and Evolutionary Computation, 2011,1(4):173−194. [doi: 10.1016/j.swevo.2011.10.001]

[2] Wang Y, Cai ZX, Zhou YR, Xiao CX. Constrained optimization evolutionary algorithms. Ruan Jian Xue Bao/Journal of Software, 2009,20(1):11−29 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3363.htm

[3] Tessema B, Yen G. An adaptive penalty formulation for constrained evolutionary optimization. IEEE Trans. on Systems, Man and Cybernetics—Part A: Systems and Humans, 2009,39(3):565−578. [doi: 10.1109/TSMCA.2009.2013333]

[4] Deb K. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 2000,186(2-4):311−338. [doi: 10.1016/S0045-7825(99)00389-8]

[5] Cui CG, Li YJ, Wu TJ. A relative feasibility degree based approach for constrained optimization problems. Journal of Zhejiang University-Science C—Computers & Electronics, 2010,11(4):249−260. [doi: 10.1631/jzus.C0910072]

[6] Wang Y, Cai ZX. Combining multiobjective optimization with differential evolution to solve constrained optimization problems. IEEE Trans. on Evolutionary Computation, 2012,16(1):117−134. [doi: 10.1109/TEVC.2010.2093582]

[7] Mezura-Montes E, Coello CAC. A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Trans. on Evolutionary Computation, 2005,9(1):1−17. [doi: 10.1109/TEVC.2004.836819]

[8] Rao SS. Engineering Optimization: Theory and Practice. 5th ed., Hoboken: Wiley Interscience, 2009. 432−442.

[9] Venkatraman S, Yen GG. A generic framework for constrained optimization using genetic algorithms. IEEE Trans. on Evolutionary Computation, 2005,9(4):424−435. [doi: 10.1109/TEVC.2005.846817]

[10] Wright M. The interior-point revolution in optimization: History, recent developments, and lasting consequences. American Mathematical Society, 2005,42(1):39−56.

[11] Back T, *et al*. Handbook of Evolutionary Computation. Bristol: Institute of Physics Publishing, 1997. 351−356.

[12] Papalambros PY, Wilde DJ. Principles of Optimal Design: Modeling and Computation. 2nd ed., Cambridge: Cambridge University Press, 2000. 310−314.

[13] Sprent P, Smeeton N. Applied Nonparametric Statistical Methods. 3rd ed., New York: Taylor & Francis US, 2007. 249−252.

[14] Xie XF, Zhang WJ, Bi DC. Handling equality constraints by adaptive relaxing rule for swarm algorithms. In: Garrison W, ed. Proc. of the IEEE Congress on Evolutionary Computation. Oregon: IEEE, 2004. 2012−2016. [doi: 10.1109/CEC.2004.1331143]

[15] Beyer H, Schwefel H. Evolution strategies—A comprehensive introduction. Natural Computing, 2002,1(1):3−52. [doi: 10.1023/A:1015059928466]

[16] Schwefel H. Evolution and Optimum Seeking. New York: John Wiley & Sons, 1995. 105−158.

[17] Kramer O. Premature convergence in constrained continuous search spaces. In: Rudolph G, ed. Proc. of the Parallel Problem Solving from Nature PPSN X. Berlin, Heidelberg: Springer-Verlag, 2008. 62−71. [doi: 10.1007/978-3-540-87700-4_7]

[18] Runarsson TP, Yao X. Stochastic ranking for constrained evolutionary optimization. IEEE Trans. on Evolutionary Computation, 2000,4(3):284−294. [doi: 10.1109/4235.873238]

附中文参考文献:

[2] 王勇,蔡自兴,周育人,肖赤心.约束优化进化算法.软件学报,2009,20(1):11−29. http://www.jos.org.cn/1000-9825/3363.htm

崔承刚(1981－),男,山东济南人,博士,助理研究员,主要研究领域为进化计算,约束优化问题求解,能源优化调度.

杨晓飞(1975－),男,高级工程师,主要研究领域为管理科学,电子信息技术.