ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.1, January 2009, pp.11−29
doi: 10.3724/SP.J.1001.2009.03363

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

约束优化进化算法*

王 勇[1+], 蔡自兴[1], 周育人[2], 肖赤心[1,3]

[1](中南大学 信息科学与工程学院,湖南 长沙 410083)

[2](华南理工大学 计算机科学与工程学院,广东 广州 516040)

[3](湘潭大学 信息工程学院,湖南 湘潭 411105)

Constrained Optimization Evolutionary Algorithms

WANG Yong[1+], CAI Zi-Xing[1], ZHOU Yu-Ren[2], XIAO Chi-Xin[1,3]

[1](School of Information Science and Engineering, Central South University, Changsha 410083, China)

[2](School of Computer Science and Engineering, South China University of Technology, Guangzhou 516040, China)

[3](School of Information Engineering, Xiangtan University, Xiangtan 411105, China)

+ Corresponding author: E-mail: ywang@csu.edu.cn

Wang Y, Cai ZX, Zhou YR, Xiao CX. Constrained optimization evolutionary algorithms. *Journal of Software*, 2009,20(1):11−29. http://www.jos.org.cn/1000-9825/3363.htm

Abstract: Constrained optimization problems (COPs) are mathematical programming problems frequently encountered in the disciplines of science and engineering application. Solving COPs has become an important research area of evolutionary computation in recent years. In this paper, the state-of-the-art of constrained optimization evolutionary algorithms (COEAs) is surveyed from two basic aspects of COEAs (i.e., constraint-handling techniques and evolutionary algorithms). In addition, this paper discusses some important issues of COEAs. More specifically, several typical algorithms are analyzed in detail. Based on the analyses, it concluded that to obtain competitive results, a proper constraint-handling technique needs to be considered in conjunction with an appropriate search algorithm. Finally, the open research issues in this field are also pointed out.

Key words: evolutionary algorithm; constraint-handling technique; constrained optimization; multi-objective optimization; constrained optimization evolutionary algorithms

摘 要: 约束优化问题是科学和工程应用领域经常会遇到的一类数学规划问题.近年来,约束优化问题求解已成为进化计算研究的一个重要方向.从约束优化进化算法=约束处理技术+进化算法的研究框架出发,从约束处理技术和进化算法两个基本方面对约束优化进化算法的研究及进展进行了综述.此外,对约束优化进化算法中的一些重要问题进行了探讨.最后进行了各种算法的比较性总结,深入分析了目前约束优化进化算法中亟待解决的问题,并指出了值得进一步研究的方向.

关键词: 进化算法;约束处理技术;约束优化;多目标优化;约束优化进化算法

* Supported by the National Natural Science Foundation of China under Grant Nos.60805027, 60234030, 60673062, 90820302 (国家自然科学基金); the Academician Foundation Project of Hu'nan Province of China under Grant No.06IJY3035 (湖南省院士基金); the Graduate Degree Thesis Innovation Foundation of Central South University of China under Grant No.1373-74334000016 (中南大学研究生学位论文创新基金)

Received 2007-12-02; Accepted 2008-04-15

中图法分类号: TP18 文献标识码: A

约束优化问题(constrained optimization problems,简称 COPs)是一类广泛存在于实际工程中但又较难求解的问题,因而对其研究具有十分重要的理论和实际意义.目前,求解约束优化问题的算法有很多,按照性质大体可分为两类:确定性的算法和随机性的算法.确定性的算法通常是基于梯度的搜索方法,如投影梯度法、简约梯度法、各类外点及内点惩罚函数法、Lagrangian 法和序列二次规划法等.这些方法存在的主要问题是,求解需要设置很好的初值点并需要函数的梯度信息,它们对于不可导、可行域不连通、甚至根本没有显式数学表达式等问题无能为力,而且求得的多为局部最优解.随机性的算法主要包括进化算法(evolutionary algorithms,简称 EAs)、模拟退火(simulated annealing,简称 SA)算法、禁忌搜索(tabu search,简称 TS)等.进化算法是一种模拟自然进化过程的全局优化方法,它借用了达尔文"物竞天择、适者生存"的生物进化观点,通过选择、交叉、变异等机制来提高个体的适应性.模拟退火算法利用统计力学中物质退火过程与优化问题求解的相似性,采用 Metropolis 接受准则并适当控制温度的下降过程实现模拟退火,从而达到求解全局优化问题的目的.禁忌搜索是对局部邻域搜索的一种扩展,是一种全局逐步寻优的启发式搜索方法.它通过设置禁忌表和禁忌对象来避免迂回搜索,并采用藐视准则来放松禁忌策略.与确定性的优化方法相比,进化算法是一种具有有向随机性的智能优化方法,更适合于求解约束优化问题.此外,与随机性的优化方法(如模拟退火算法、禁忌搜索等)相比,进化算法是一种基于群体的搜索技术,具有鲁棒性强、搜索效率高、不易陷入局部最优等特点.

近 10 年来,利用进化算法求解约束优化问题已有许多学者进行了广泛的研究,并且提出了大量的约束优化进化算法(constrained optimization evolutionary algorithms,简称 COEAs)[1,2].特别值得一提的是,2006 年~2008 年,进化计算国际大会(IEEE Congress on Evolutionary Computation)每年均为约束优化举办了 Special Session.

进化算法在约束优化问题中的成功应用取决于以下几个主要因素:

(1) 进化算法从一个群体即多个点而不是一个点开始搜索,这使得它能够以较大概率找到全局最优解;

(2) 进化算法对所优化问题的特征不敏感;

(3) 进化算法很容易执行和使用.

本文介绍了约束优化进化算法的最新研究成果.从约束处理技术和进化算法两个基本方面出发,对约束优化进化算法的研究及其进展进行了综述.此外,本文对约束优化进化算法中的若干重要问题进行了探讨.最后进行了实验比较,并指出了值得进一步研究的方向.

1 约束优化问题及其相关定义

不失一般性,一个约束优化问题可描述为如下形式:

$$\begin{aligned} &\text{minimize} && f(\vec{x})\ \ \vec{x}=(x_1,x_2,...,x_n)\in\Re^n \\ &\text{subject to} && g_j(\vec{x})\le 0,\ j=1,\ldots,l \\ &&& h_j(\vec{x})=0,\ j=l+1,\ldots,p \end{aligned} \qquad \text{(问题(1))}$$

这里, $\vec{x}\in\Omega\subseteq S$ 为决策向量,Ω为可行域,S 为决策空间.一般地,S 为 $\Re^n$ 中的 n 维长方体:$l(i)\le x_i\le u(i),l(i),u(i)$ 为常数,$i=1,\ldots,n$. $f(\vec{x})$, $g_j(\vec{x})$, $h_j(\vec{x})$ 均为 $\Re^n$ 上的 n 元函数, $f(\vec{x})$ 为目标函数, $g_j(\vec{x})\le 0$ 为第 j 个不等式约束条件, $h_j(\vec{x})=0$ 为第 j 个等式约束条件.l 表示不等式约束条件的个数,$p-l$ 表示等式约束条件的个数.

定义 **1**. Ω为问题(1)的可行域(feasible region)当且仅当

$$\Omega=\{\vec{x}\in S\mid g_j(\vec{x})\le 0,\ j=1,...,l;h_j(\vec{x})=0,\ j=l+1,...,p\} \tag{1}$$

Ω 在 S 中的补集为问题(1)的不可行域.可行域中的解称为可行解,不可行域中的解称为不可行解.图 1 给出了搜索空间 S 及其可行域Ω的示意图.如果任意一个不等式约束条件满足 $g_j(\vec{x})=0$ ($j\in\{1,\ldots,l\}$),则称 $g_j(\vec{x})$ 在 $\vec{x}$ 处活跃(active).显然,所有的等式约束条件 $h_j(\vec{x})$ ($j=l+1,\ldots,p$)对于可行域Ω中的任意点均活跃.

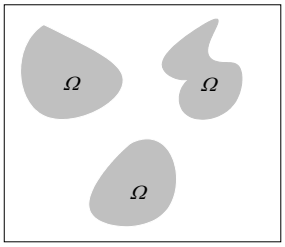


Fig.1 Search space S and its feasible region Ω

图 1 搜索空间 S 及其可行域Ω

因为本文中涉及的一些约束处理方法是基于多目标优化技术的,所以下面给出了多目标优化问题的相关描述和多目标优化中的 4 个重要定义.不失一般性,考虑以下具有 n 个决策变量和 m 个目标函数的多目标优化问题(multi-objective optimization problems,简称 MOPs):

$$\min \quad \vec{y} = \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_m(\vec{x})) \qquad \text{(问题(2))}$$

其中,$\vec{x} = (x_1, x_2, ..., x_n) \in X \subset \Re^n$ 为决策向量,X 为决策空间,$\vec{y} \in Y \subset \Re^m$ 为目标向量,Y 为目标空间.

定义 **2**(**Pareto** 优超(**Pareto dominance**)). 决策向量 $\vec{x}_u \in X$ Pareto 优超决策向量 $\vec{x}_v \in X$,记为 $\vec{x}_u \prec \vec{x}_v$,当且仅当:

1) $\forall i \in \{1,\ldots,m\}$满足 $f_i(\vec{x}_u) \le f_i(\vec{x}_v)$;

2) $\exists j \in \{1,\ldots,m\}$满足 $f_j(\vec{x}_u) < f_j(\vec{x}_v)$.

此时,也称决策向量 $\vec{x}_v$ Pareto 劣于(dominated by)决策向量 $\vec{x}_u$.若决策向量 $\vec{x}_u$ 与决策向量 $\vec{x}_v$ 不存在 Pareto 优超关系,则称它们非劣(non-dominated).

定义 **3**(**Pareto** 最优解(**Pareto optimality**)). 决策向量 $\vec{x}_u \in X$ 称为 X 上的 Pareto 最优解,当且仅当 $\neg\exists \vec{x}_v \in X$ 使得 $\vec{x}_v \prec \vec{x}_u$.

定义 **4**(**Pareto** 最优解集(**Pareto optimal set**)). 对于给定的多目标优化问题 $\vec{f}(\vec{x})$, Pareto 最优解集(ρ^*)定义为 $\rho^* = \{\vec{x}_u \in X \mid \neg\exists \vec{x}_v \in X, \vec{x}_v \prec \vec{x}_u\}$.

Pareto 最优解集中的个体也称为非劣个体.

定义 **5**(**Pareto** 前沿(**Pareto front**)). 对于给定的多目标优化问题 $\vec{f}(\vec{x})$ 和 Pareto 最优解集(ρ^*),Pareto 前沿(ρf^*)定义为 $\rho f^* = \{\vec{u} = \vec{f}(\vec{x}_u) \mid \vec{x}_u \in \rho^*\}$.

显然,Pareto 前沿是 Pareto 最优解集在目标空间中的像.

2 基于进化算法的约束处理技术

进化算法已被广泛应用于求解优化问题.然而,其性能的好坏主要取决于两个因素:

1) 进化算法的随机性能;

2) 如何将优化问题的目标函数转换为适应值函数,因为适应值函数可以使搜索向着合理的区域进行.

当优化问题具有约束条件时,将目标函数转换为适应值函数变得非常困难.这是由于此时适应值函数不仅要评价一个解的好坏,还应描述其与搜索空间中可行域的接近程度.

当优化问题具有许多线性和非线性、等式和不等式约束条件时,其求解过程将变得更加复杂.值得注意的是,进化算法是一种无约束的搜索技术,因为它缺乏明确的约束处理机制,这促使研究者开发不同的方法来处理约束条件.一般来说,在进化算法中结合约束处理技术会给算法带来一些额外的参数,这些参数的选取通常由使用者决定.正因为如此,设计具有较好性能的约束处理技术显得尤为重要.

一般来说,基于进化算法的约束处理技术将等式约束条件转换为如下的不等式约束条件来处理,即

$$|h(\vec{x})| - \delta \le 0 \qquad (2)$$

其中,δ为等式约束条件的容忍值,一般取较小的正数.将等式约束条件转换为不等式约束条件处理后,问题(1)将包含 p 个不等式约束条件.

通常,群体中的个体 $\vec{x}$ 违反第 j 个约束条件的程度可表示为

$$G_j(\vec{x})=\begin{cases}\max\{0,g_j(\vec{x})\}, & 1\le j\le l\\ \max\{0,|h_j(\vec{x})|-\delta\}, & l+1\le j\le p\end{cases} \tag{3}$$

则

$$G(\vec{x})=\sum_{j=1}^{p}G_j(\vec{x}) \tag{4}$$

表示个体 $\vec{x}$ 违反问题(1)中所有约束条件的程度,也反映了个体 $\vec{x}$ 在群体中的不可行性.

值得注意的是,由于约束条件之间的特征差异,某些约束条件可能对个体的约束违反程度 $G(\vec{x})$ 起着决定性的作用,此时,可通过标准化来平等地对待每个约束条件.在标准化过程中,首先找出群体中的个体违反每个约束条件的最大值 $G_j^{\max}(j\in\{1,...,p\})$:

$$G_j^{\max}=\max_{i=1,\cdots,N}(G_j(\vec{x}_i)),\quad j\in\{1,...,p\} \tag{5}$$

其中,N为群体规模,即群体中所包含的个体数.利用这些 $G_j^{\max}$ 可以标准化个体 $\vec{x}_i$ 对每个约束条件的违反值,最后个体 $\vec{x}_i$ 的标准化约束违反程度 $G_{nor}(\vec{x}_i)$ 定义为该个体的每个约束违反标准值的平均值:

$$G_{nor}(\vec{x}_i)=\frac{\sum_{j=1}^{p}G_j(\vec{x}_i)\big/G_j^{\max}}{p},\quad i\in\{1,...,N\} \tag{6}$$

根据近年来基于进化算法的约束处理技术的研究趋势,本文将它们划分为以下 3 类[3]:1) 惩罚函数法;2) 多目标法;3) 其他方法.

2.1 惩罚函数法

惩罚函数法因为执行简单而得到了广泛的应用,其主要思想是,通过对目标函数 $f(\vec{x})$ 增加惩罚项 $p(\vec{x})$ 来构造惩罚适应值函数 $fitness(\vec{x})$,将约束优化问题转换为无约束优化问题进行处理.

惩罚项的构造通常基于个体违反约束条件的程度 $G(\vec{x})$.同时,惩罚项的形式决定了惩罚函数法的类型,例如若惩罚项中的惩罚系数不依赖于进化代数,则这类方法称为静态惩罚函数法.文献[4]按如下方式构造惩罚适应值函数:

$$fitness(\vec{x})=f(\vec{x})+\sum_{j=1}^{p}r_{k,j}G_j^2(\vec{x}) \tag{7}$$

其中,$r_{k,j}(k=1,\ldots,q;j=1,\ldots,p)$为惩罚系数,$q$ 为用户对每个约束条件定义的约束违反水平数.若惩罚项中的惩罚系数随着进化代数的改变而改变,则这类方法称为动态惩罚函数法.文献[5]提出了如下的惩罚适应值函数:

$$fitness(\vec{x})=f(\vec{x})+(Ct)^{\alpha}\sum_{j=1}^{p}G_j^{\beta}(\vec{x}) \tag{8}$$

其中,t 是进化代数,C,α,β是需要调整的参数.

Le Riche 等人[6]设计了一种隔离遗传算法,它具有两个惩罚系数,这两个惩罚系数旨在过大与过小的惩罚之间实现平衡.在进化过程中,该方法首先随机产生规模为 $2m$ 的初始群体.接着通过两个惩罚系数构造两个惩罚适应值函数,群体中的每个个体分别通过两个惩罚适应值函数进行评价,这样得到两个惩罚适应值列表.然后对两个列表中的惩罚适应值分别排序,最后根据两个排序后的列表,从规模为 $2m$ 的群体中选出最好的 m 个个体构成下一代群体.

死惩罚法[7]是最简单但最严厉的惩罚函数法,它总是拒绝不可行解,不利用可行解提供的任何信息.在死惩罚法中,不可行解的惩罚适应值定义为 0.这样当初始群体不包含可行解时,进化过程将会停滞,因为群体中的所有个体具有相同的惩罚适应值,此时需要重新生成初始群体.死惩罚法仅适合于可行域为凸或可行域占搜索空间比例较大的约束优化问题.

Huang 等人[8]提出了一种协同进化法,该方法采用两个群体.第 1 个群体中的个体表示惩罚系数集,第 2 个群体中的个体表示问题的解.利用第 1 个群体中的惩罚系数可以进化第 2 个群体中的解,同时,第 2 个群体中的个体可以用来调整第 1 个群体中的惩罚系数.通过协同地进化这两个群体,迭代结束时可以得到满意的解和合理

的惩罚系数.

一般来说,自适应惩罚函数法[9,10]具有较好的优化效果,因为它能利用搜索过程中的反馈信息动态地调节参数.Rasheed[11]提出了一种自适应惩罚函数法.该方法在初始阶段具有较小的惩罚系数,这样可以保证群体对搜索空间充分采样.在进化过程中,该方法根据群体状态自适应地决定增加或减少惩罚系数.最近,在文献[12]的基础上,Farmani 和 Wright[13]提出了一种自适应适应值表示法,该方法将惩罚分为两个阶段进行:第 1 个惩罚阶段使得群体中最差的不可行解具有比群体中最好解更高或相等的惩罚适应值,第 2 个惩罚阶段使得群体中最差的不可行解的惩罚适应值等于群体中具有最大目标函数值的解的惩罚适应值.

对于惩罚函数法,具有以下定理.

定理 **1**[14]. 令 $\{s_t\}_1^\infty$ 为一个非负、严格单调递增趋于无穷大的序列,定义以下函数:

$$L(s,\vec{x}) = f(\vec{x}) + sG(\vec{x}) \tag{9}$$

其中,s 为惩罚系数.令 $\vec{x}_t$ 使得 $L(s,\vec{x})$ 取最小值,则序列 $\{\vec{x}_t\}_1^\infty$ 的极限即为问题(1)的最优解.

上述定理说明,当 $s\to\infty$时, $L(s,\vec{x})$ 的最小值与 $f(\vec{x})$ 的最小值等价.

虽然惩罚函数法是进化算法求解约束优化问题时最常用的方法,但其仍存在着一定的缺陷.其中最为主要的缺陷是惩罚系数的合理设置十分复杂,往往需要多次实验来不断地进行调整.惩罚系数决定着对不可行解的惩罚程度,过大或过小的惩罚程度可能给进化算法的求解过程带来困难.如果惩罚程度过大,群体将以较快的速度进入可行域,此时忽略了对不可行域的勘探和开采,这样对于最优解位于可行域边界或可行域不连通的约束优化问题求解便会出现困难.另一方面,如果惩罚程度过小,个体的惩罚适应值主要由目标函数决定,此时,群体可能在不可行域产生滞留现象,这样,群体将很难进入可行域,甚至可能收敛于不可行解.

Richardson 等人[15]对如何构造惩罚函数提出了以下导向性的准则:

1) 基于个体违反约束条件程度构造的惩罚函数比基于个体违反约束条件个数构造的惩罚函数具有更好的性能;
2) 对于具有较少约束条件和较少可行解的约束优化问题,如果仅仅基于个体违反约束条件个数来构造惩罚函数,则不可能找到最优解;
3) 好的惩罚函数应该从两个量出发来构造:最大完备花费(maximum completion cost)和期望完备花费(expected completion cost).完备花费是指个体违反约束条件的程度.
4) 惩罚应该接近期望完备花费,但是不能频繁地低于它.总之,惩罚越精确,得到的解的质量越好.

在惩罚函数法中,个体的惩罚适应值由目标函数和惩罚项同时决定,所以在计算个体惩罚适应值时,目标函数和惩罚项具有一定的支配关系.Runarsson 和 Yao[16]认为,惩罚函数法试图在目标函数和惩罚项中找到一个好的平衡(trade-off).在文献[16]中,他们将惩罚函数法归结为选取一个合理的惩罚系数 r_g,并且系统地分析了在评价个体时,r_g 如何影响目标函数和惩罚项之间的支配关系.事实上,对于每个群体,均存在某个区间$[r_1,r_2]$,使得当 $r_g<r_1$ 时,个体惩罚适应值的比较完全由目标函数 $f(\vec{x})$ 决定;当 $r_g>r_2$ 时,个体惩罚适应值的比较完全由惩罚项 $p(\vec{x})$ 决定;当 $r_g\in[r_1,r_2]$时,个体惩罚适应值的比较由目标函数 $f(\vec{x})$ 和惩罚项 $p(\vec{x})$ 共同决定.值得注意的是,参数 r_1 和 r_2 的选取与具体的群体有关,因而是依赖于问题的.

同时,Yu 等人[17]指出,即使最具动态性的惩罚函数法,鉴于无约束全局最优解与具有约束条件时的全局最优解相距很远的问题,其优化效果也不可能很好.

2.2 多目标法

由于惩罚函数法存在着一些缺陷,近年来,研究者提出将约束优化问题转换为多目标优化问题来处理.本文将基于上述思想的方法分为两类:区分可行解与不可行解法和多目标优化法.

2.2.1 区分可行解与不可行解法

区分可行解与不可行解法通常将约束优化问题转换为具有两个目标的多目标优化问题.一般来说,其中一个目标为原问题的目标函数 $f(\vec{x})$,另一个目标为个体违反约束条件的程度 $G(\vec{x})$. 这类方法的主要特点是,在群

体进化过程中对可行解与不可行解区别对待.区分可行解与不可行解法与惩罚函数法的本质区别在于,后者在评价个体时同时考虑个体的目标函数值和约束违反程度,因而需要通过惩罚系数使目标函数值和约束违反程度具有相同的阶(order),然而,前者有针对性地利用目标函数值或约束违反程度来比较个体.以下介绍几种典型的算法.

Powell 和 Skolnick[18]将可行解的适应值映射到区间(−∞,1),将不可行解的适应值映射到区间(1,+∞),使得可行解总是优于不可行解.Powell 和 Skolnick 使用如下的个体评估方式:

$$ftiness(\vec{x}) = \begin{cases} f(\vec{x}), & \text{if feasible} \\ 1 + rG(\vec{x}), & \text{otherwise} \end{cases} \tag{10}$$

其中, $f(\vec{x})$ 被缩放到区间(−∞,1), $G(\vec{x})$ 被缩放到区间(1,+∞),r 为常数.

Deb[19]提出了一种联赛选择算子(也就是每次比较成对的个体),并采用以下准则比较个体:

1) 当在两个比较的个体中,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;

2) 当两个比较的个体均为可行解时,选择目标函数值小的个体;

3) 当两个比较的个体均为不可行解时,选择违反约束条件程度小的个体.

上述比较准则的主要缺陷是难以发挥不可行解的作用,特别是当群体中的绝大部分个体均为可行解时,不可行解将很难进入群体.为了保持群体的多样性,该文还提出了一种简单的小生态技术.

Jiménez 和 Verdegay[20]提出了一种类似于多目标优化中使用的 min-max 表示方法.该方法中的个体比较准则类似于 Deb[19]所提出的个体比较准则:

1) 当一个个体为可行解,另外一个个体为不可行解时,可行解总是优于不可行解;

2) 当两个个体均为可行解时,目标函数值小的个体占优;

3) 当两个个体均为不可行解时,个体的比较基于最大的约束违反程度 $\max_{j=1,\ldots,p} G_j(\vec{x})$, 具有最小的最大约束违反程度的个体占优.

在文献[19]的基础上,Mezura-Montes 和 Coello Coello[21]提出了一种简单的多样性操作.该操作以一定的概率(例如 0.03)使得群体中最好的不可行解可以继续生存.值得注意的是,这类多样性机制具有十分重要的作用,特别是当全局最优解位于可行域边界上时.

林丹等人[22]针对很多约束优化问题的最优解位于可行域边界的特点,提出了一种自适应保持群体中不可行解比例的策略,并将其与文献[19]中的个体比较准则结合起来,得到了一个新的个体比较准则:

1) 当两个个体 $\vec{x}_1$ 和 $\vec{x}_2$ 都可行时,比较它们的目标函数值,目标函数值小的个体占优;

2) 当两个个体 $\vec{x}_1$ 和 $\vec{x}_2$ 都不可行时,比较它们违反约束条件的程度,违反约束条件程度小的个体占优;

3) 当 $\vec{x}_1$ 可行而 $\vec{x}_2$ 不可行时,如果 $G(\vec{x}_2) < \varepsilon$, 比较它们的目标函数值,目标函数值小的个体占优;否则, $\vec{x}_1$ 占优.

为了将不可行解的比例保持在一个固定的水平,文献[22]对ε进行了自适应的调整.

Runarsson 和 Yao[16]提出了随机排序法,它是目前最为经典的基于进化算法的约束处理技术.该方法采用参数 p_f 表示在不可行域中仅使用目标函数比较个体的概率,也就是说,当比较两个相邻的个体时,若两个个体都是可行解,则比较它们目标函数的概率是 1,否则,参数 p_f 将决定是否采用目标函数或约束违反程度来比较个体.实验结果表明,当 p_f=0.45 时,随机排序法可以产生很好的优化效果.值得注意的是,p_f=0.45 意味着个体之间的比较更多地依赖于约束违反程度.最近,两位作者结合差异进化对该算法进行了改进[23].

需要说明的是,文献[19]中的个体比较准则可视为随机排序法的一个特例.因为当 p_f=0 时,随机排序法中的个体比较准则与文献[19]中的个体比较准则完全等价.

Takahama 和 Sakai[24]提出了α约束法.该方法采用约束满足水平 $\mu(\vec{x})$ 来表示个体 $\vec{x}$ 满足约束条件的程度.为了定义个体 $\vec{x}$ 的约束满足水平 $\mu(\vec{x})$,首先计算个体 $\vec{x}$ 对每个约束条件的满足水平,接着再将这些满足水平组合起来.例如,对于个体 $\vec{x}$,根据关于 $g_i(\vec{x})$ 和 $h_i(\vec{x})$ 的分段线性函数,每个约束条件可转换为如下的满足水平:

$$\mu_{g_i}(\vec{x})=\begin{cases}1, & \text{if} \quad g_i(x)\le 0\\ 1-g_i(\vec{x})/b_i, & \text{if} \quad 0\le g_i(x)\le b_i\\ 0, & \text{otherwise}\end{cases} \tag{11}$$

$$\mu_{h_j}(\vec{x})=\begin{cases}1-|h_j(\vec{x})|/b_j, & \text{if} \quad |h_j(\vec{x})|\le b_j\\ 0, & \text{otherwise}\end{cases} \tag{12}$$

其中,b_i 和 b_j 为正常数.通过组合满足水平 $\mu_{g_i}(\vec{x})$ 和 $\mu_{h_j}(\vec{x})$,个体 $\vec{x}$ 的约束满足水平 $\mu(\vec{x})$ 定义为

$$\mu(\vec{x})=\min_{i,j}\{\mu_{g_i}(\vec{x}),\mu_{h_j}(\vec{x})\} \tag{13}$$

在定义个体的约束满足水平后,文献[24]采用α水平比较(记为$<_\alpha$)来比较个体的优劣.令 f_1,f_2 和μ_1,μ_2 分别表示个体 $\vec{x}_1$ 和 $\vec{x}_2$ 的目标函数值和约束满足水平,α水平比较定义如下:

$$(f_1,\mu_1)<_\alpha(f_2,\mu_2)\Leftrightarrow\begin{cases}f_1<f_2, & \text{if} \quad \mu_1,\mu_2\ge\alpha\\ f_1<f_2, & \text{if} \quad \mu_1=\mu_2\\ \mu_1>\mu_2, & \text{otherwise}\end{cases} \tag{14}$$

其中 $0\le\alpha\le1$.在文献[24]中,α的取值由分段函数控制.通过采用α水平比较替换通常的比较准则,α水平法可将求解无约束问题的算法转换为求解约束优化问题的算法.

2.2.2 多目标优化法

多目标优化法近年来受到了极大的关注,其主要特点是:

1) 将约束优化问题转换为多目标优化问题,

2) 利用多目标优化技术来处理转换后的问题.

在将约束优化问题转换为多目标优化问题时,通常存在着两种方式.第 1 种方式将约束优化问题转换为具有两个目标的多目标优化问题.在第 2 种方式中,约束优化问题的目标函数和约束条件分别作为不同的目标看待.对于第 1 种方式,第 1 个目标为原问题的目标函数 $f(\vec{x})$,第 2 个目标为个体违反约束条件的程度 $G(\vec{x})$.令 $\boldsymbol{f}(\vec{x})=(f(\vec{x}),G(\vec{x}))$,此时可以利用多目标优化技术对 $\boldsymbol{f}(\vec{x})$ 进行求解.对于第 2 种方式,转换后的多目标优化问题将具有 $p+1$ 个目标,其中 p 为原问题的约束条件个数.这样就得到了一个新的待优化的向量 $\boldsymbol{F}(\vec{x})=(f(\vec{x}),f_1(\vec{x}),...,f_p(\vec{x}))$,其中 $f_1(\vec{x}),...,f_p(\vec{x})$ 为原问题的约束条件,此时可利用多目标优化技术对 $\boldsymbol{F}(\vec{x})$ 进行求解.

一般来说,以下 3 种多目标优化技术经常运用于处理转换后的问题:

1) 使用 Pareto 优超作为一种选择准则;

2) 使用 Pareto 排序(ranking)来定义个体适应值;

3) 将群体划分为若干个子群体,子群体的评估或者基于目标函数,或者基于某个约束条件.这种机制称为基于群体的方法.

下面介绍几种典型的算法.

Surry和Radcliffe[25]提出了COMOGA(constrained optimization by multi-objective genetic algorithm)方法.在该方法中,约束优化问题被视作约束满足问题或无约束优化问题来处理.当约束优化问题被视为约束满足问题时,目标函数被忽略,此时,个体之间的比较由Pareto排序决定,Pareto排序基于约束违反来定义.当约束优化问题被视为无约束优化问题时,约束条件被忽略,此时,个体之间的比较由目标函数决定.受向量评估遗传算法[26]的启发,文献[25]采用参数p_{cost}来决定基于目标函数选择个体的概率.为了避免因参数p_{cost}固定而引发的一些问题,该方法通过对群体中的可行解设置目标比例τ来动态地调节p_{cost}.例如,假设群体中可行解的目标比例为τ=0.1,若当前代群体中的可行解比例没有靠近0.1,则应相应地调节p_{cost}.

Camponogara 和 Talukdar[27]从 Pareto 集合中计算个体的改进方向,Pareto 集合由目标函数和约束违反程度共同定义.如图 2 所示,考虑两个 Pareto 集合 S_i 与 S_j 和两个个体 $\vec{x}_i$ 与 $\vec{x}_j$,其中 $i<j$, $\vec{x}_i\in S_i$, $\vec{x}_j\in S_j$, $\vec{x}_i\prec\vec{x}_j$.通过这两个个体,可以得到如下的改进方向:

$$d=(\vec{x}_i-\vec{x}_j)/|\vec{x}_i-\vec{x}_j| \tag{15}$$

根据此改进方向进行线性搜索,可望找到一个更好的解 $\vec{x}$,使得 $\vec{x}$ 同时 Pareto 优超 $\vec{x}_i$ 和 $\vec{x}_j$.

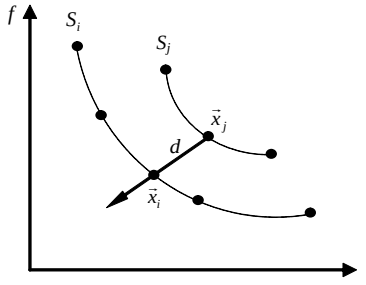


Fig.2 Search direction obtained by individual $\vec{x}_i \in S_i$ and $\vec{x}_j \in S_j$

图 2 从个体 $\vec{x}_i \in S_i$ 和 $\vec{x}_j \in S_j$ 得到的搜索方向

Coello Coello[28]提出了一种基于 Pareto 排序过程[29]的方法来定义个体的等级(rank).对于群体中的每个个体 $\vec{x}_i$ ($i=1,...,N$),该方法通过公式(16)计算个体的等级:

$$rank(\vec{x}_i)=count(\vec{x}_i)+1 \tag{16}$$

其中,$count(\vec{x}_i)$ 根据以下准则来计算:初始化 $count(\vec{x}_i)$ =0,将 $\vec{x}_i$ 与群体中的其他个体 $\vec{x}_j$ ($j=1,...,N; j\neq i$)逐一进行比较:

1) 如果 $\vec{x}_i$ 和 $\vec{x}_j$ 都为可行解,则 $count(\vec{x}_i)$ 保持不变;
2) 如果 $\vec{x}_i$ 为不可行解,$\vec{x}_j$ 为可行解,则 $count(\vec{x}_i)=count(\vec{x}_i)+1$;
3) 如果 $\vec{x}_i$ 和 $\vec{x}_j$ 均为不可行解,但 $\vec{x}_i$ 比 $\vec{x}_j$ 违反更多的约束条件,则 $count(\vec{x}_i)=count(\vec{x}_i)+1$;
4) 如果 $\vec{x}_i$ 和 $\vec{x}_j$ 均为不可行解,且它们违反相同个数的约束条件,但 $\vec{x}_i$ 比 $\vec{x}_j$ 具有更大的约束违反程度,则 $count(\vec{x}_i)=count(\vec{x}_i)+1$.

然后,对个体的等级按公式(17)进行变换:

$$rank(\vec{x}_i)=\begin{cases} fitness(\vec{x}_i), & \text{if } \vec{x}_i \text{ is feasible} \\ 1/rank(\vec{x}_1), & \text{otherwise} \end{cases} \tag{17}$$

值得注意的是,$fitness(\vec{x}_i)$ 需要经过一定的处理,以使得可行解的等级总是高于不可行解的等级.这样与不可行解相比,可行解更容易进入下一代种群.

周育人等人[30]采用 Pareto 强度值对个体进行排序选优.Pareto 强度值定义如下:

定义 **6**(**Pareto** 强度值). 设 $\vec{x}_i$ 为群体 P 中的一个个体,用 $S(\vec{x}_i)$表示群体中 Pareto 劣于 $\vec{x}_i$ 的个体总数,称为 $\vec{x}_i$ 的强度值,即 $S(\vec{x}_i)$=#{ $\vec{x}_j \mid \vec{x}_j \in P$ 且 $\vec{x}_i \prec \vec{x}_j$ }.其中,#表示集合的基数.

Pareto 强度值反映了个体在群体 P 中的强弱程度.Pareto 强度值越大,表示群体中 Pareto 劣于该个体的个体越多,则该个体越优秀.该方法在比较个体时,首先比较个体的 Pareto 强度值,Pareto 强度值大的个体为优,若个体之间具有相等的 Pareto 强度值,则比较它们违反约束条件的程度,违反约束条件程度小的个体为优.通过上述比较方法可以对群体中的个体进行排序.每次遗传操作完成之后,该算法选择 Pareto 强度值最大的个体和违反约束条件程度最小的个体同时进入下一代种群.

Cai 和 Wang 提出了 CW 算法[31].CW 算法首先找出子代群体中所有的非劣个体,然后随机选择一个非劣个体,并用该非劣个体随机替换掉父代群体中的一个劣于个体(如果该劣于个体存在).此外,该算法还提出了一种不可行解存档和替换机制,旨在引导群体快速地向可行域逼近.值得注意的是,该算法在不需要将等式约束条件转换为不等式约束条件的情况下,也能搜索到全局最优解.

Venkatraman 和 Yen[32]提出了一个通用的框架求解约束优化问题,该框架包括两个阶段.第 1 个阶段将约束优化问题作为约束满足问题进行处理,其目标为至少找到一个可行解.为了实现这个目标,群体基于约束违反程度进行排序.若群体中出现可行解则转入第 2 个阶段,此时,其目标为找到全局最优解.第 2 个阶段同时考虑目标函数和约束违反程度,并对群体进行非劣排序[33].此外,第 2 个阶段还采用小生态策略来保持群体的多样性.

Wang 等人[34]认为约束优化问题求解的本质在于如何有效地均衡目标函数和约束违反程度,提出了一种自适应均衡模型.该模型将群体进化分为 3 种情形,即 1) 群体中仅包含可行解;2) 群体由可行解与不可行解联合组成;3) 群体中仅包含不可行解,并针对每种进化情形设计了不同的个体比较和选择准则.当群体处于第 1 种情形时,提出了一种分层的非劣个体选择机制,其目的在于引导群体从不同的方向朝可行域逼近;当群体处于第 2

种情形时,提出了一种自适应转换个体目标函数值的方法,其目的在于自适应地调节群体中可行解与不可行解的比例;当群体处于第 3 种情形时,个体之间的比较和选择主要依赖于目标函数值.

Coello Coello[35]基于向量评估遗传算法[26]求解约束优化问题.在每一次迭代中,群体被划分为 p+1 个具有相等规模的子群体,其中 p 为约束条件的个数.该方法中,其中一个子群体将目标函数作为适应值函数来评价个体.此外,其余的子群体将相应的约束条件作为适应值函数来评价个体,其目的在于每个子群体试图找到对应于相应约束条件的可行解,通过联合这些子群体,该方法可以找到对应于所有约束条件的可行解.对于将约束条件作为适应值函数的子群体,该方法基于如下准则来定义个体的适应值:如果 $g_j(\vec{x})<0$,则适应值定义为 $g_j(\vec{x})$;否则,如果 v≠0,则适应值定义为–v;否则,适应值定义为 $f(\vec{x})$.其中, $g_j(\vec{x})$ 为对应于第 j+1 个子群体的约束条件,v 是指个体违反约束条件的个数.该算法的主要缺陷是子群体的个数会随着约束条件的个数呈线性增加.此外,如何确定每个子群体的规模也有待深入研究.

Coello Coello 和 Mezura-Montes[36]提出了一种类似于 Pareto 小生态遗传算法[37]的约束处理技术.Pareto 小生态遗传算法是一种多目标优化方法,它采用基于 Pareto 优超的联赛选择机制.然而与文献[37]不同,该方法不使用小生态技术,而是通过选择概率 S_r 来控制群体的多样性.选择概率 S_r 指定了群体中按以下 4 个准则进行比较的个体比例:1) 当两个个体均为可行解时,比较它们的目标函数值,目标函数值小的个体占优;2) 可行解总是优于不可行解;3) 当两个个体均为不可行解时,若一个个体劣于,另外一个个体非劣,则非劣个体占优;4) 当两个个体均为不可行解且同时非劣或劣于时,违反约束条件程度小的个体占优.判断某个个体劣于或非劣,是通过将该个体与群体中预先设定的一个样本集进行比较而得到的.群体中的其余个体通过纯概率方法进行选择.

Ray 和 Liew[38]通过模仿社会行为来处理约束优化问题.该方法提出了一种社会-文明(society-civilization)模型.文明由若干个社会组成,每个社会拥有自己的“领导”,这些“领导”用于引导其邻居进化.此外,“领导”可以从一个社会迁移到另外一个社会,这促进了对搜索空间中新的领域的勘探.同时,该方法还提出了一种新颖的“领导”辨识机制:1) 如果群体中没有可行解,则“领导”为约束等级为 1 的个体,其中,约束等级为 1 表示该个体非劣;2) 如果群体中的所有个体均为可行解,则“领导”为目标等级小于群体平均目标等级的个体.

Aguirre等人[39]提出了一种IS-PAES(inverted-shrinkable Pareto archived evolutionary strategy)法,该方法基于Pareto存档进化策略[40].IS-PAES采用自适应网格存储进化过程中发现的最好解,同时自适应网格也是一种群体多样性维持技术.当产生一个新的个体后,Pareto优超用于比较该个体与自适应网格中个体的优劣.此外,IS-PAES还提出了一种缩减区域技术.在进化过程中,随着对包围在可行域周围个体信息的开发与利用,搜索区域将不断地被缩减.

为了评估多目标优化法的优化性能,Mezura-Montes 和 Coello Coello[41]对其中 4 种较好的方法进行了广泛的实验研究.实验结果表明,Pareto 优超选择准则能够比 Pareto 排序和基于群体的方法得到更好的结果.同时,Mezura-Montes 和 Coello Coello 还指出,必须采用额外的机制(如多样性技术)来改进这类方法的有效性.

2.3 其他算法

除了上述两类约束处理技术以外,研究者还提出了许多不同的思想.与上述两类方法相比,因为这些思想所形成的约束处理技术或者不具备较好的优化性能,或者通用性欠佳,所以本文称它们为其他算法,并选取其中较具代表性的几种算法进行说明.

1991 年,Michalewicz 和 Janikow[42]提出了 GENOCOP(genetic algorithm for numerical optimization for constrained problems)算法,该算法通过投影操作将可行解映射到可行域边界.GENOCOP 算法仅适合于线性的约束条件并且需要初始可行解.为了克服这些缺陷,1994 年,Michalewicz 和 Attia[43]提出了一种混合优化方法 GENOCOPII 来处理通用的非线性规划问题.后来,Michalewicz 和 Nazhiyath[44]通过修补不可行解并结合协同进化提出了一种新的算法 GENOCOPIII.

同态映射法[45]通过在可行域与 n 维立方体$[-1,1]^n$ 之间建立映射关系,将约束优化问题转化为无约束优化问题求解.图 3 解释了二维平面中的任意凸可行域 F 通过 T 映射到正方形$[-1,1]^2$,其中, $\vec{r}_0$ 点映射到 0 点, $\vec{x}_0$ 点映射到 $\vec{y}_0$ 点.同态映射法的优点是,不需要特别的操作来保持解的可行性,不需要评价不可行解,可以与任何类型

的进化算法进行结合;缺点是,当可行域非凸时,算法的实现较为复杂,而且算法需要初始可行解.

增广 Lagrangian 法可将约束优化问题转化为 min-max 问题进行求解.增广 Lagrangian 函数通常定义为

$$L_A(\vec{x},\vec{\mu},\vec{\lambda},\rho)=f(\vec{x})+\sum_{i=1}^{l}p_i(\vec{x},\mu_i,\rho)+\vec{\lambda}^T h(\vec{x})+\rho\sum_{i=l+1}^{p}h_i^2(\vec{x}) \tag{18}$$

其中,ρ 为惩罚常量,$\vec{\mu}$ 为 $l\times 1$ 的乘子向量,$\vec{\lambda}$ 为$(p-l)\times 1$ 的乘子向量,p_i 由公式(19)定义:

$$p_i(\vec{x},\mu_i,\rho)=\begin{cases}\mu_i g_i(\vec{x})+\rho g_i^2(\vec{x}), & \text{if} \quad g_i(\vec{x})\ge -\mu_i/2\rho\\ -\mu_i^2/4\rho, & \text{if} \quad g_i(\vec{x})<-\mu_i/2\rho\end{cases} \tag{19}$$

通过增广 Lagrangian 函数,约束优化问题可转化为如下 min-max 问题:

$$\min_{\vec{x}}\max_{\vec{\mu},\lambda}L(\vec{x},\vec{\mu},\vec{\lambda},\rho) \tag{20}$$

$$\text{Subject to} \quad \mu_i\ge 0, \quad i=1,...,l \tag{21}$$

$$\lambda_i\ge 0, \quad i=l+1,...,p \tag{22}$$

事实上,当问题(1)的最优解 $\vec{x}^*$ 与 min-max 问题中的 $(\vec{\mu}^*,\vec{\lambda}^*)$ 满足 Kuhn-Tucker 条件时,$(\vec{x}^*,\vec{\mu}^*,\vec{\lambda}^*)$ 称为增广 Lagrangian 函数的鞍点,即

$$L_A(\vec{x}^*,\vec{\mu},\vec{\lambda},\rho)\le L_A(\vec{x}^*,\vec{\mu}^*,\vec{\lambda}^*,\rho)\le L_A(\vec{x},\vec{\mu}^*,\vec{\lambda}^*,\rho) \tag{23}$$

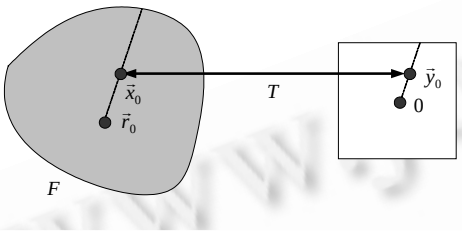


Fig.3 A mapping T from a convex feasible region F to a square $[-1,1]^2$

图 3 凸可行域 F 通过 T 映射到正方形$[-1,1]^2$

当 min-max 问题具有鞍点时,称为 zero-sum 博弈问题.将约束优化问题通过增广 Lagrangian 法转化为 min-max 问题后的主要任务在于求解 zero-sum 博弈问题的鞍点.公式(23)表明,在 $(\vec{\mu}^*,\vec{\lambda}^*)$ 已知的情况下,$\vec{x}^*$ 为 $L_A(\vec{x},\vec{\mu}^*,\vec{\lambda}^*,\rho)$ 的无约束最优解,这样 $\vec{x}^*$ 可在整个决策空间 S 中搜索且不需要考虑约束条件.

Kim 和 Myung[46]提出了一种两阶段的进化规划方法.该方法在第 2 个阶段采用增广 Lagrangian 函数作为适应值函数来评价个体.同时,Lagrangian 乘子向量 $(\vec{\mu},\vec{\lambda})$ (记 $\vec{\theta}=(\vec{\mu},\vec{\lambda})$)采用确定性的增广 Lagrangian 法中广泛使用的更新策略来进行更新.Tahk 和 Sun[47]提出了一种协同增广 Lagrangian 方法.该方法通过进化具有相反目标的两个群体来求解 zero-sum 博弈问题的鞍点,第 1 个群体 P_1 由决策向量 $\vec{x}$ 组成,第 2 个群体 P_2 由 Lagrangian 乘子向量 $\vec{\theta}$ 组成.当进化第 1 个群体 P_1 中的决策向量 $\vec{x}$ 时,第 2 个群体 P_2 中的 Lagrangian 乘子向量 $\vec{\theta}$ 保持不变;当进化第 2 个群体 P_2 中的 Lagrangian 乘子向量 $\vec{\theta}$ 时,第 1 个群体 P_1 中的决策向量 $\vec{x}$ 保持不变,这两个群体协同地进化,并通过适应值函数相互影响.对于第 1 个群体中的个体 $\vec{x}$,适应值函数定义为

$$fitness(\vec{x})=\max_{\vec{\mu},\vec{\lambda}\in P_2}L_A(\vec{x},\vec{\mu},\vec{\lambda},\rho) \tag{24}$$

对于第 2 个群体中的个体 $\vec{\theta}$,适应值函数定义为

$$fitness(\vec{\theta})=\min_{\vec{x}\in P_1}L_A(\vec{x},\vec{\mu},\vec{\lambda},\rho) \tag{25}$$

近期,Krohling 和 Coelho[48]也采用类似方法来处理约束优化问题.

3 进化算法

传统的进化算法包括 3 个主要分支:遗传算法(genetic algorithm)、进化策略(evolutionary strategy)和进化规划(evolutionary programming).

作为进化算法的一个重要部分,近年来,进化策略在处理约束优化问题时受到了极大的重视.对此,有以下两个原因:1) 具有理论背景支持进化策略收敛;2) 进化策略的自适应机制对其处理约束搜索空间有一定的帮助[49,50].实验结果表明,在相同的约束处理技术下,采用进化策略时算法的整体性能明显优于遗传算法[51].通过

融合 3 个简单的个体比较准则,Mezura-Montes 和 Coello Coello[49]对几种不同的进化策略(包括(μ+1)-ES、(μ^+,λ)-ES 和相关变异(μ^+,λ)-ES)进行了实验比较.结果表明,对于具有高维搜索空间、非线性等式约束条件的测试函数,利用进化策略不能得到很好的结果.同时还应注意到,进化策略本身存在着一个缺陷,即在进化策略中交叉操作往往被忽略.但就约束优化问题而言,可行解与不可行解在一些重要的区域进行交叉,对于找到全局最优解很有帮助,特别是当最优解位于可行域边界上时.尽管在进化策略中也可以使用交叉操作(如中间交叉、离散交叉等)[21],但其搜索能力十分有限.一般来说,进化规划较少用于处理约束优化问题.

近年来,粒子群优化(partial swarm optimization)[52]、差异进化(differential evolution)[53]、文化算法(cultural algorithm)[54]等也常用于处理约束优化问题.粒子群优化是由 Eberhart 和 Kennedy[55]于 1995 年提出来的一种基于种群搜索的自适应进化计算技术.粒子群优化随机产生一个初始种群并赋予每个粒子一个随机速度,在飞行过程中,粒子的速度通过跟踪两个极值来加以动态调整,第 1 个极值是粒子本身找到的最好解,第 2 个极值是种群找到的最好解.差异进化由 Storn 和 Price[56]于 1997 年提出,它是一种简单、高效的进化算法.差异进化采用浮点数编码,它通过联合父代个体和群体中其他个体来产生子代个体.当子代个体优于父代个体时,便替换掉父代个体.Reynolds[57]于 1994 年提出了文化算法,其主要思想是通过提取进化过程中的领域知识来改进算法的有效性.文化算法框架由群体空间(population space)和信念空间(belief space)两部分组成,如图 4 所示.群体空间中包含问题的解.信念空间是一个信息存储器,个体可以将其经验存储到信念空间中供其他个体学习.群体空间和信念空间通过通信协议(例如 acceptance 函数和 influence 函数)连结起来.

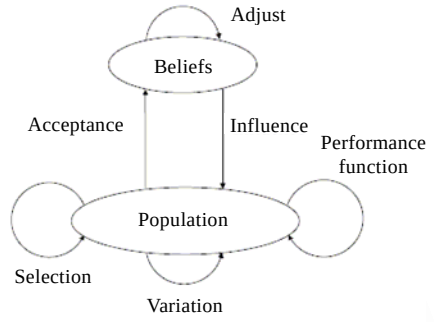


Fig.4 Framework of cultural algorithm

图 4 文化算法框架

4 对约束优化进化算法的几点讨论

4.1 $\boldsymbol{f}(\vec{x})$ 与一般多目标优化问题的区别

事实上,将约束优化问题转换为多目标优化问题来处理的主要目的是为了避免使用惩罚函数,因为惩罚函数中的参数设置较为复杂.虽然一个约束优化问题可以转换为一个双目标优化问题 $\boldsymbol{f}(\vec{x})$ 来处理(见第 2.2.2 节),但值得注意的是,$\boldsymbol{f}(\vec{x})$ 与一般的多目标优化问题仍然具有本质上的区别.这种区别在于,对一般的多目标优化问题而言,其求解目的是为了找到一个均匀分布的 Pareto 最优解集;而 $\boldsymbol{f}(\vec{x})$ 在可行域内又退化为一个单目标优化问题 $f(\vec{x})$ (此时 $G(\vec{x})=0$),这样,其最优解仍然为一个点,因而对 $\boldsymbol{f}(\vec{x})$ 而言,我们没有必要关心全局最优解的分布情况.由此可见,两者在寻优过程上并不完全等价.从第 2.2.2 节中的分析可以看出,现有的多目标优化法大都是对多目标进化算法(multi-objective evolutionary algorithms)的简单模仿和借鉴.

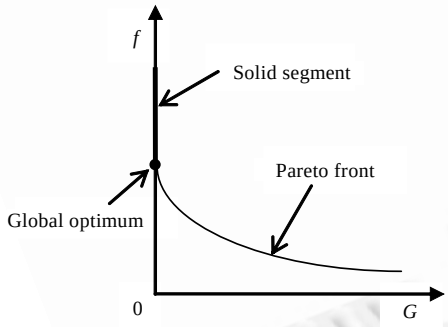


Fig.5 Graph representation for $\boldsymbol{f}(\vec{x})$

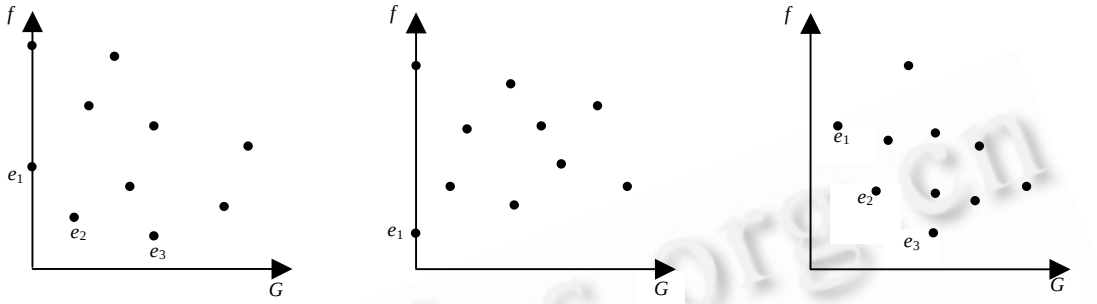
图 5 $\boldsymbol{f}(\vec{x})$ 的示意图

定义 7(全局最优解(**global optimum**)). $\vec{x}^* \in S$ 称为 $\boldsymbol{f}(\vec{x})$ 的全局最优解,当且仅当 $G(\vec{x}^*)=0$ 且 $\neg\exists\vec{x}$, 当 $G(\vec{x})=0$ 时,使得 $f(\vec{x}) \le f(\vec{x}^*)$.

基于第 1 节中提到的 4 个定义,一个对于 $\boldsymbol{f}(\vec{x})$ 的更为深刻的解释如图 5 所示.在图 5 中,Pareto 最优解集ρ^*映射到 Pareto 前沿ρf^*;可行域Ω 映射到实线段;全局最优解 x^*映射到 Pareto 前沿和实线段的交点;搜索空间 S 映射到 Pareto 前沿及 Pareto 前沿以上的区域.

4.2 非劣个体的重要意义

在多目标进化算法中,群体中的所有个体均被赋予等级,而且群体中的非劣个体具有相同的等级值.例如,在文献[33]中,非劣个体的等级值为 1,在文献[58]中,非劣个体的等级值为 0.在约束优化中,非劣个体在群体中具有重要的地位.图 6(a)解释了非劣个体在群体中所占的重要地位.例如,群体中有 3 个非劣个体,分别用 e_1,e_2 和 e_3 表示.显然,个体 e_1 表示最好的可行解,个体 e_2 表示约束违反程度最小的不可行解,个体 e_3 表示具有最小目标函数值的不可行解.这样,一个群体中最主要的信息均包含在非劣个体中.文献[30]认为个体 e_2 为最优个体,因为它具有最大的 Pareto 强度值;同时,文献[32]认为个体 e_1 为需要存档的精英个体,因为它为具有最小目标函数值的可行解.然而,个体 e_1 与个体 e_2 仅为非劣个体的一部分.



(a) Nondominated individuals in a population consist of a combination of feasible and infeasible solutions (individuals e_1, e_2 and e_3)

(a) 群体中的非劣个体由可行解和不可行解组成(个体 e_1,e_2 和 e_3)

(b) Nondominated individuals in a population consist of one feasible solution (individual e_1)

(b) 群体中的非劣个体由一个可行解组成(个体 e_1)

(c) Nondominated individuals in a population consist of infeasible solutions (individuals e_1, e_2 and e_3)

(c) 群体中的非劣个体由多个不可行解组成(个体 e_1,e_2 和 e_3)

Fig.6

图 6

非劣个体还具有以下几个重要的特征(仅考虑决策变量和目标函数一一映射的情况):

定理 **2**. 一个群体的非劣个体中最多包含一个可行解.

证明(反证法):假设有 $k(k>1)$个可行解包含于一个群体的非劣个体中,则在这 k 个个体中具有最小目标函数值 $f(\vec{x})$ 的个体必然 Pareto 优超这 k 个个体中的其他个体.这与这 k 个个体均为非劣个体相矛盾. □

定理 2 意味着,就整个搜索空间而言,非劣个体中的可行解即为全局最优解(如图 5 所示).

性质 **1**. 非劣个体可能由一个可行解(如图 6(b)所示)、不可行解(如图 6(c)所示)或部分可行解与不可行解(如图 6(a)所示)组成.

性质 **2**. 子代群体非劣个体中的可行解可以 Pareto 优超父代群体中的可行解或不可行解,但子代群体非劣个体中的不可行解只能 Pareto 优超父代群体中的不可行解.

性质 2 可由 Pareto 优超的定义直接得到.

定理 **3**. 若子代群体 M'中的非劣个体全部由不可行解组成,则子代群体 M'也全部由不可行解组成.

证明(反证法):假设 M'中的非劣个体全部由不可行解组成且 M'包含可行解.因为不可行解不能 Pareto 优超可行解,所以,此时 M'包含的非劣个体中也必然包含可行解,矛盾.因此,定理 2 成立. □

定理 **4**. 若子代群体 M'由不可行解组成,则 M'中违反约束条件最小的不可行解必为 M'的非劣个体中违反约束条件最小的不可行解.

证明:设 $\vec{x}'$ 为 M'中违反约束条件最小的不可行解,则 $\neg\exists\vec{x}''\in M'$,使得 $\vec{x}''\prec\vec{x}'$.因此, $\vec{x}'$ 必包含于 M'的非劣个体中,从而也必为 M'的非劣个体中违反约束条件最小的不可行解. □

4.3 Pareto优超关系与算法全局收敛性

若算法仅使用 Pareto 优超关系比较个体,则可能不具备全局收敛性.下面举例说明.

设所求解问题的搜索空间、可行域和目标函数如图 7 所示.从图 7 中可以看出,可行域占搜索空间的比例非常小.假设初始种群中不包含可行解,经过一定次数的迭代后,后代群体中可能会出现可行解.但是,根据 Pareto 优超关系,此时,可行解与群体中的所有不可行解均具有非劣关系(由图 7 中目标函数的特征可以看出).显然,在这种情况下,若仅采用 Pareto 优超关系比较个体,群体中不可能包含可行解.这样,算法就不具备全局收敛性.

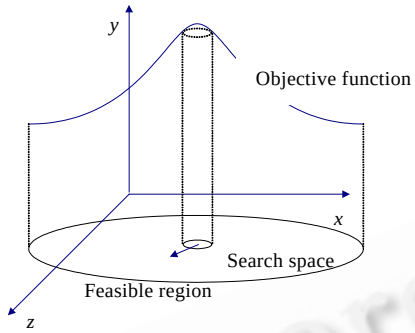


Fig.7 Schematic diagram of the search space, feasible region and objective function

图 7 搜索空间、可行域和目标函数示意图

4.4 随机排序法与多目标优化法

随机排序法[16]是目前约束优化进化算法中最为经典的一种算法.对于随机排序法,可采用如下准则分析其机理:给定两个个体 $\vec{x}_1$ 和 $\vec{x}_2$,考虑其间的 Pareto 优超性,1) 若其中一个个体 Pareto 优超另一个个体,则性能较优的个体占优;否则,意味着 $\vec{x}_1$ 与 $\vec{x}_2$ 存在 Pareto 非劣关系,则 2) 如果 $rand<p_f$,比较两个个体的目标函数值,目标函数值小的个体占优;如果 $rand\geq p_f$,比较两个个体违反约束条件的程度,违反约束条件程度小的个体占优.可以证明上述比较准则与随机排序法等价,但是,从上述比较准则可以更加清晰地理解随机排序法的运行机理:只有当两个个体 Pareto 非劣时,参数 p_f 才可以发挥其作用.可见,本质上随机排序法与多目标优化法具有一定的相关性.

5 约束优化进化算法性能比较与亟待解决的问题

5.1 约束优化进化算法性能比较

为了评估约束优化进化算法的全局优化性能,国际上一般采用 13 个标准(benchmark)测试函数($g01$~$g13$)来进行实验研究(测试函数见文献[16]).这些测试函数的主要特征在表 1 中给出.

Table 1 Characteristics of 13 benchmark test functions

表 **1** 13 个标准测试函数的特征

Function	n	Type of f	ρ (%)	LI	NE	NI	a
$g01$	13	Quadratic	0.000 3	9	0	0	6
$g02$	20	Nonlinear	99.996 5	1	0	1	1
$g03$	10	Nonlinear	0.000 0	0	1	0	1
$g04$	5	Quadratic	26.935 6	0	0	6	2
$g05$	4	Nonlinear	0.000 0	2	3	0	3
$g06$	2	Nonlinear	0.006 4	0	0	2	2
$g07$	10	Quadratic	0.000 3	3	0	5	6
$g08$	2	Nonlinear	0.864 0	0	0	2	0
$g09$	7	Nonlinear	0.525 6	0	0	4	2
$g10$	8	Linear	0.000 5	3	0	3	3
$g11$	2	Quadratic	0.000 0	0	1	0	1
$g12$	3	Quadratic	0.019 7	0	0	9^3	0
$g13$	5	Nonlinear	0.000 0	0	3	0	3

其中,n 为决策变量的个数,LI 为线性不等式约束条件的个数,NE 为非线性等式约束条件的个数,NI 为非线性不

等式约束条件的个数,a 表示在最优解处活跃的约束条件的个数,ρ表示可行域占整个搜索空间的比例,它通过以下的公式计算得到:

$$\rho=|\Omega|/|S| \tag{26}$$

其中,$|S|$表示从决策空间 S 中随机选出的个体数,$|\Omega|$表示选出的$|S|$个个体中的可行解的个数,一般$|S|$=1000000.

表 2 给出了 7 种优化性能较好的约束进化算法对 13 个标准测试函数的实验结果,表中结果均从原文中收集得到.

Table 2 Comparison of results provided by 7 algorithms on 13 benchmark test functions (*NA*=not available)

表 **2** 7 种算法对 13 个标准测试函数的实验结果比较(*NA* 表示原文中没有提供实验结果)

Function/ optimal	Status	Methods						
		SAFF[13]	SMES[21]	RY[16]	IRY[23]	αSimplex[24]	CDE[54]	CW[31]
g01/ −15.000	Best	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000
	Mean	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000
	Worst	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000
	st. dev	0	0	0.0E+00	1.3E−13	6.4E−06	2.0E−06	1.3E−14
g02/ −0.803619	Best	−0.802 97	−0.803 601	−0.803 515	−0.803 619	−0.803 619	−0.803 619	−0.803 619
	Mean	−0.790 10	−0.785 238	−0.781 975	−0.772 078	−0.784 187	−0.724 886	−0.803 220
	Worst	−0.760 43	−0.751 322	−0.726 288	−0.683 055	−0.754 259	−0.590 908	−0.792 608
	st. dev	1.2E−02	1.7E−02	2.0E−02	2.6E−02	1.3E−02	7.0E−02	2.0E−03
g03/ −1.000	Best	−1.000	−1.000	−1.000	−1.001	−1.001	−0.995	−1.000
	Mean	−1.000	−1.000	−1.000	−1.001	−1.001	−0.789	−1.000
	Worst	−1.000	−1.000	−1.000	−1.001	−1.001	−0.640	−1.000
	st. dev	7.5E−05	2.1E−04	1.9E−04	6.0E−09	8.5E−14	1.1E−01	2.8E−16
g04/ −30665.539	Best	−30 665.50	−30 665.539	−30 665.539	−30 665.539	−30 665.539	−30 665.539	−30 665.539
	Mean	−30 665.20	−30 665.539	−30 665.539	−30 665.539	−30 665.539	−30 665.539	−30 665.539
	Worst	−30 663.30	−30 665.539	−30 665.539	−30 665.539	−30 665.539	−30 665.539	−30 665.539
	st. dev	4.9E−01	0	2.0E−05	2.2E−11	4.2E−11	0	8.0E−12
g05/ 5126.498	Best	5 126.989	5 126.599	5 126.497	5 126.497	5 126.497	5 126.571	5 126.498
	Mean	5 432.080	5 174.492	5 128.881	5 126.497	5 126.497	5 207.411	5 126.498
	Worst	6 089.430	5 304.167	5 142.472	5 126.497	5 126.497	5 327.391	5 126.498
	st. dev	3.9E+03	5.0E+01	3.5E+00	6.2E−12	3.5E−11	6.9E+01	1.5E−12
g06/ −6961.814	Best	−6 961.800	−6 961.814	−6 961.814	−6 961.814	−6 961.814	−6 961.814	−6 961.814
	Mean	−6 961.800	−6 961.284	−6 875.940	−6 961.814	−6 961.814	−6 961.814	−6 961.814
	Worst	−6 961.800	−6 952.482	−6 350.262	−6 961.814	−6 961.814	−6 961.814	−6 961.814
	st. dev	0	1.9E+00	1.6E+02	6.4E−12	1.3E−10	0	1.8E−12
g07/ 24.306	Best	24.48	24.327	24.307	24.306	24.306	24.306	24.306
	Mean	26.58	24.475	24.374	24.306	24.306	24.306	24.306
	Worst	28.40	24.843	24.642	24.308	24.307	24.306	24.306
	st. dev	1.1E+00	1.3E−01	6.6E−02	2.7E−04	1.3E−04	1.0E−06	5.7E−12
g08/ −0.095825	Best	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825
	Mean	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825
	Worst	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825
	st. dev	0	0	2.6E−17	4.2E−17	3.8E−13	0	3.2E−17
g09/ 680.630	Best	680.64	680.632	680.630	680.630	680.630	680.630	680.630
	Mean	680.72	680.643	680.656	680.630	680.630	680.630	680.630
	Worst	680.87	680.719	680.763	680.630	680.630	680.630	680.630
	st. dev	5.9E−02	1.6E−02	3.4E−02	4.6E−13	2.9E−10	0	4.7E−13
g10/ 7049.248	Best	7 061.34	7 051.903	7 054.316	7 049.248	7 049.248	7 049.248	7 049.248
	Mean	7 627.89	7 253.047	7 559.192	7 049.249	7 049.248	7 049.248	7 049.248
	Worst	8 288.79	7 638.366	8 835.655	7 049.296	7 049.248	7 049.248	7 049.248
	st. dev	3.7E+02	1.4E+02	5.3E+02	4.9E−03	4.7E−06	1.7E−04	4.0E−09
g11/ 0.750	Best	0.750	0.75	0.750	0.750	0.750	0.750	0.750
	Mean	0.750	0.75	0.750	0.750	0.750	0.758	0.750
	Worst	0.750	0.75	0.750	0.750	0.750	0.796	0.750
	st. dev	0	1.5E−04	8.0E−05	1.8E−15	4.9E−16	1.7E−02	0.0E+00
g12/ −1.000	Best	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000
	Mean	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000
	Worst	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000	−1.000
	st. dev	0	0	0.0E+00	9.6E−10	3.9E−10	0	0.0E+00
g13/ 0.0539498	Best	NA	0.053 986	0.053 957	0.053 942	0.053 941 5	0.056 180	0.053 949 8
	Mean	NA	0.166 385	0.067 543	0.096 276	0.066 770 2	0.288 324	0.053 949 8
	Worst	NA	0.468 294	0.216 915	0.438 803	0.438 802 6	0.392 100	0.053 949 8
	st. dev	NA	1.8E−01	3.1E−02	1.2E−01	6.9E−02	1.7E−01	6.5E−17

表 2 中的性能指标包括最好结果(best)、平均结果(mean)、最差结果(worst)和解的标准方差(St.dev).此外,7 种约束优化进化算法分别为 SAFF(self-adaptive fitness formulation)[13],SMES(simple multimembered evolutionary strategy)[21],RY(Runarsson and Yao's algorithm)[16],IRY(improved Runarsson and Yao's algorithm)[23], aSimplex[24],CDE(cultured differential evolution)[54]和 CW(Cai and Wang's algorithm)[31].

值得注意的是,在 13 个测试函数中,测试函数 $g02$ 主要考察进化算法的搜索能力,因为其目标函数维数较高且拓扑结构非常复杂;测试函数 $g03$,$g05$,$g11$ 和 $g13$ 主要考察约束处理能力,因为它们具有等式约束条件;其他函数主要考察约束优化进化算法的综合能力.

通过表 2,可以从整体上对约束优化进化算法的性能有一个比较清晰的认识,这有助于更加全面地了解约束优化进化算法,并进一步发现和考虑其中一些亟待解决的问题.从表 2 中可以看出,SAFF,SMES 和 RY 在进化算法和约束处理两方面均存在缺陷;IRY 和 αSimplex 主要在进化算法方面存在缺陷;CDE 主要在约束处理方面存在缺陷;CW 具有最好的优化效果,这是由于它采用了高效的进化算法和有效的约束处理技术,并将这两部分有机地结合起来.上述实验结果充分说明,只有同时发挥约束处理技术和进化算法的作用,约束优化进化算法才能取得较好的效果.

5.2 亟待解决的问题

虽然约束优化进化算法已经取得了一定的成果,但是仍有很多问题需要解决,集中体现在以下几点:

(1) 处理等式约束条件.为了找到可行解,现有的算法大都将等式约束条件转换为不等式约束条件来处理(见公式(2)).值得注意的是,上述转换使得可行域的拓扑结构发生了变化,而且这种变化与 δ 的取值有关.显然,当可行域发生变化时,全局最优解也可能发生变化,因而上述转换对算法的性能和找到的解的精度存在着直接的影响.然而,如果没有上述转换或 δ 取值很小时,对于具有复杂等式约束条件的约束优化问题,进化算法很难找到可行解.为了减少上述转换给算法性能带来的影响,文献[21]随着进化代数的增加动态地减小 δ,文献[59]根据群体中可行解的百分比调节 δ,文献[60]提出了一种基于百分比的容忍值调节策略.但是,如何有效地设置参数 δ 仍是一个值得深入研究的问题.

(2) 理论性.一种理论的成熟取决于它的数学描述的完善程度,与无约束单目标进化算法的理论研究相比,约束优化进化算法的理论研究相对落后.虽然 Mezura-Montes[51]对约束优化进化算法的收敛性作了初步的尝试,但其模型过于简单,而且作者在证明过程中对约束条件并没有严格考虑,因而如何证明约束优化进化算法的收敛性需要深入研究.此外,研究约束优化进化算法的收敛速度也是该领域的一个重要课题[61].

(3) 混合性.对于无约束优化问题,近年来,混合算法(memetic algorithms)[62]通过融合局部搜索机制,成功地改进了进化算法的优化效率和有效性.这类算法的优点是搜索效率很高,而且特别适合于优化复杂多峰函数.对于约束优化问题,如何设计混合算法以提高约束优化进化算法的优化性能是一个值得广泛研究的问题[63].

(4) 减少计算开销.对于某些约束优化问题,计算目标函数值和约束违反需要很高的计算开销.因而,如何使目标函数和约束违反的评估变得高效是一个很有意义的研究课题.适应值近似[64]为该课题的研究提供了一条有效的途径.

(5) 缩减搜索区域.因为约束优化问题的全局最优解位于可行域边界或内部,所以在进化过程中,离可行域边界很远的搜索区域对发现最优解几乎没有帮助.因而可以根据群体的进化情形,对搜索区域进行缩减以减少不必要的搜索,提高算法的收敛速度.文献[65,66]提出了变范围的遗传算法求解约束优化问题,这两种方法自适应地转移和减小了搜索区域的大小.

(6) 分布估计算法.分布估计算法是当前进化计算领域研究的热点[67],它已被应用于求解无约束优化问题和多目标优化问题[68],并得到了较好的结果.然而其在约束优化问题中的应用还未见报道.在第 4.1 节中我们曾分析过,约束优化问题的全局最优解位于 Pareto 前沿与可行域的交界处.这样,在进化过程中,可以利用分布估计算法产生的群体不断地近似 Pareto 前沿,在进化结束时,Pareto 前沿与可行域的交点即为问题的最优解.目前,这方面的研究正在展开.

(7) 测试函数.约束优化进化算法研究的一个重要问题,就是如何定义实际约束优化问题的共同特性,并把

这些共同的特性作为算法所要解决的特殊问题.实际问题是丰富多彩的,但并不是每一个实际的约束优化问题都包括了约束优化问题的所有特征.因而,设计一些能够反映实际约束优化问题基本特征的标准测试函数,是该领域需要研究的一个重要问题.Michalewicz 等人[69]虽然设计了一个用于约束优化的测试函数产生器,但通过其设计出来的测试函数过于对称.

(8) 推广性.与单目标优化问题不同的是,多目标优化问题的最优解通常为一个集合,这种不同所带来的直接影响是个体间的比较准则和最优解分布的差异.研究这些差异及其所引发的一系列问题,可将约束处理技术推广到多目标约束优化问题中去.

6 结 论

本文根据约束优化进化算法=约束处理技术+进化算法的研究框架出发,对约束优化进化算法的研究进展进行了归纳并加以介绍.此外,还对约束优化进化算法中的若干问题进行了讨论,分析了约束优化进化算法亟待解决的问题,并展望了其未来可能的发展方向.如何求解约束优化问题是进化计算的一个重要研究领域,具有十分重要的理论和实际意义.因为该研究中涉及到的问题很多,本文不可能面面俱到,希望能够起到一个抛砖引玉的作用.

References:

[1] Michalewicz Z, Schoenauer M. Evolutionary algorithm for constrained parameter optimization problems. Evolutionary Computation, 1996,4(1):1−32.

[2] Coello Coello CA. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Computation Methods in Applied Mechanics and Engineering, 2002,191(11-12):1245−1287.

[3] Guo GQ, Wang Y. Constrained-Handling techniques based on evolutionary algorithms. Journal of Hu'nan Institute of Science and Technology (Natural Sciences), 2006,19(4):15−18 (in Chinese with English abstract).

[4] Homaifar A, Lai SHY, Qi X. Constrained optimization via genetic algorithms. Simulation, 1994,62(4):242−254.

[5] Joines J, Houck C. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with Gas. In: Michalewicz Z, Schaffer JD, Schwefel HP, Fogel DB, Kitano H, eds. Proc. of the 1st IEEE Conf. on Evolutionary Computation. Orlando: IEEE Press, 1994. 579−584.

[6] Le RRG, Knopf-Lenoir C, Haftka RT. A segregated genetic algorithm for constrained structural optimization. In: Eshelman LJ, ed. Proc. of the 6th Int'l Conf. on Genetic Algorithms. San Francisco: Morgan Kaufman Publishers, 1995. 558−565.

[7] Hoffmeister F, Sprave J. Problem-Independent handling of constraints by use of metric penalty functions. In: Fogel LJ, Angeline PJ, Back T, eds. Proc. of the 5th Annual Conf. on Evolutionary Programming (EP'96). San Diego: MIT Press, 1996. 289−294.

[8] Huang F, Wang L, He Q. An effective co-evolutionary differential evolution for constrained optimization. Applied Mathematics and computation, 2007,186(1):340−356.

[9] Hamida SB, Schoenauer M. ASCHEA: New results using adaptive segregational constraint handling. In: Fogel DB, *et al*., eds. Proc. of the Congress on Evolutionary Computation 2002 (CEC 2002). Piscataway: IEEE Service Center, 2002. 884−889.

[10] Coit DW, Smith AE, Tate DM. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. Informs Journal on Computing, 1996,8(2):173−182.

[11] Rasheed K. An adaptive penalty approach for constrained genetic algorithm optimization. In: Koza JR, *et al*., eds. Proc. of the 3rd Annual Conf. on Genetic Programming (GP'98)/Symp. on Genetic Algorithms (SGA'98). San Francisco: Morgan Kaufmann Publishers, 1998. 584−590.

[12] Wright JA, Farmani R. Genetic algorithm: A fitness formulation for constrained minimization. In: Spector L, *et al*., eds. Proc. of the Genetic and Evolutionary Computation Conf. San Francisco: Morgan Kaufmann Publishers, 2001. 725−732.

[13] Farmani R, Wright JA. Self-Adaptive fitness formulation for constrained optimization. IEEE Trans. on Evolutionary Computation, 2003,7(5):445−455.

[14] Luenberger DG. Introduction to Linear and Nonlinear Programming. Reading: Addison-Wesley, 1973.

[15] Richardson JT, Palmer MR, Liepins G, Hilliard M. Some guidelines for genetic algorithms with penalty functions. In: Schaffer JD, ed. Proc. of the 3rd Int'l Conf. on Genetic Algorithms. Morgan Kaufmann Publishers, 1989. 191−197.

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

[17] Yu JX, Yao X, Choi C, Gou G. Materialized view selection as constrained evolutionary optimization. IEEE Trans. on Systems, Man, and Cybernetics (C), 2003,33(4):458−467.

[18] Powell D, Skolnick MM. Using genetic algorithm in engineering design optimization with nonlinear constraint. In: Forrest S, ed. Proc. of the 5th Int'l Conf. Genetic Algorithms (ICGA'93). San Mateo: Morgan Kaufmann Publishers, 1993. 424−431.

[19] Deb K. An efficient constraint handling method for genetic algorithms. Computation Methods in Applied Mechanics and Engineering, 2000,86(2-4):311−338.

[20] Jiménez F, Verdegay JL. Evolutionary techniques for constrained optimization problems. In: Zimmermann HJ, ed. Proc. of the 7th European Congress Intelligence Techniques and Soft Computing (EUFIT'99). Berlin: Springer-Verlag, 1999.

[21] Mezura-Montes E, Coello Coello CA. A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Trans. on Evolutionary Computation, 2005,9(1):1−17.

[22] Lin D, Li MQ, Kou JS. A GA-based method for solving constrained optimization problems. Journal of Software, 2001,12(4): 628−632 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/12/628.htm

[23] Runarsson TP, Yao X. Search biases in constrained evolutionary optimization. IEEE Trans. on Systems, Man, Cybernetics (C), 2005,35(2):233−243.

[24] Takahama T, Sakai S. Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations. IEEE Trans. on Evolutionary Computation, 2005,9(5):437−451.

[25] Surry PD, Radcliffe NJ. The COMOGA method: Constrained optimization by multiobjective genetic algorithm. Control and Cybernetics, 1997,26(3):391−412.

[26] Schaffer JD. Multiple objective optimization with vector evaluated genetic algorithms. In: Grefenstette JJ, ed. Proc. of the 1st Int'l Conf. Genetic Algorithms and their Applications. L. Hillsdale: Erlbaum Associates Inc., 1985. 93−100.

[27] Camponogara E, Talukdar SN. A genetic algorithm for constrained and multiobjective optimization. In: Alander JT, ed. Proc. of the 3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA). Vaasa: University of Vaasa, 1997. 49−62.

[28] Coello Coello CA. Constraint handling using an evolutionary multiobjective optimization technique. Civil Engineering and Environmental Systems, 2000,17(4):319−346.

[29] Fonseca CM, Fleming PJ. Multiobjective optimization and multiple constraint handling with evolutionary algorithms-Part I: A unified formulation. IEEE Trans. on Systems, Man, and Cybernetics (A), 1998,28(1):26−37.

[30] Zhou YR, Li YX, Wang Y, Kang LS. A Pareto strength evolutionary algorithm for constrained optimization. Journal of Software, 2003,14(7):1243−1249 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1243.htm

[31] Cai Z, Wang Y. A multiobjective optimization based evolutionary algorithm for constrained optimization. IEEE Trans. on Evolutionary Computation, 2006,10(6):658−675.

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

[33] Deb K, Pratab A, Agrawal S, Meyarivan T. A fast and elitist nondominated sorting genetic algorithm for multi-objective optimization: NSGA II. IEEE Trans. on Evolutionary Computation, 2002,6(2):182−197.

[34] Wang Y, Cai Z, Zhou Y, Zeng W. An adaptive trade-off model for constrained evolutionary optimization. IEEE Trans. on Evolutionary Computation, 2008,12(1):80−92.

[35] Coello Coello CA. Treating constraints as objectives for single-objective evolutionary optimization. Engineering Optimization, 2000,32(3):275−308.

[36] Coello Coello CA, Mezura-Montes E. Constraint-Handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics, 2002,16(3):193−203.

[37] Horn J, Nafpliotis N, Goldberg D. A niched Pareto genetic algorithm for multiobjective optimization. In: Horn J, Nafpliotis N, Goldberg DE, eds. Proc. of the 1st IEEE Conf. Evolutionary Computation, Vol.1, IEEE World Congress on Computational Intelligence. Piscataway: IEEE Service Center, 1994. 82−87.

[38] Ray T, Liew KM. Society and civilization: An optimization algorithm based on the simulation of social behavior. IEEE Trans. on Evolutionary Computation, 2003,7(4):386−396.

[39] Aguirre AH, Rionda SB, Coello Coello CA, Lizáraga GL, Mezura-Montes E. Handling constraints using multiobjective optimization concepts. Int'l Journal for Numerical Methods in Engineering, 2004,59(15):1989−2017.

[40] Knowles JD, Corne DW. Approximating the nondominated front using the Pareto archived evolutionary strategy. Evolutionary Computation, 2000,8(2):149−172.

[41] Mezura-Montes E, Coello Coello CA. A numerical comparison of some multiobjective-based techniques to handle constraints in genetic algorithms. Technical Report, EVOCINV-01-2003, México: Evolutionary Computation Group (EVOCINV), Computer Science Section, Electrical Engineering Department, CINVESTAVIPN, 2003.

[42] Michalewicz Z, Janikow CZ. Handling constraints in genetic algorithm. In: Belew RK, Booker LB, eds. Proc. of the 4th Int'l Conf. on Genetic Algorithms (ICGA-91).Los Altos: Morgan Kaufmann Publishers, 1991. 151−157.

[43] Michalewicz Z, Attia NF. Evolutionary optimization of constrained problems. In: Sebald AV, Fogel LJ, eds. Proc. of the 3rd Annual Conf. on Evolutionary Programming. River Edge: World Scientific, 1994. 98−108.

[44] Michalewicz Z, Nazhiyath G. Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In: Fogel DB, ed. Proc. of the 2nd IEEE Int'l Conf. on Evolutionary Computation. Piscataway: IEEE Service Center, 1995. 647−651.

[45] Koziel S, Michalewicz Z. Evolutionary algorithm, homomorphous mappings, and constrained parameter optimization. Evolutionary Computation, 1999,7(1):19−44.

[46] Kim JH, Myung H. Evolutionary programming techniques for constrained optimization problems. IEEE Trans. on Evolutionary Computation, 1997,1(2):129−140.

[47] Tahk MJ, Sun BC. Coevolutionary augmented Lagrangian methods for constrained optimization. IEEE Trans. on Evolutionary Computation, 2000,4(2):114−124.

[48] Krohling RA, Coelho LS. Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. IEEE Trans. on Systems Man, and Cybernetics (B), 2006,36(6):1407−1416.

[49] Mezura-Montes E, Coello Coello CA. On the usefulness of the evolution strategies self-adaptation mechanism to handle constraints in global optimization. Technical Report, EVOCINV-01-2003, México: Evolutionary Computation Group (EVOCINV), Computer Science Section, Electrical Engineering Department, CINVESTAVIPN, 2003.

[50] Asselmeyer T, Ebeling W, Rosé H. Evolutionary strategies of optimization. Physical Review (E), 1997,59(1):1171−1180.

[51] Mezura-Montes E. Alternative techniques to handle constraints in evolutionary optimization [Ph.D. Thesis]. México: Computer Science Section, Electrical Engineering Department, CINVESTAVIPN, 2004.

[52] Pulido GT, Coello Coello CA. A constraint-handling mechanism for particle swarm optimization. In: Proc. of the 2004 Congress on Evolutionary Computation (CEC 2004), Vol.2. Portland: IEEE, 2004. 1396−1403.

[53] Xie XF, Zhang WJ, Zhang GR, Yang ZL. Empirical study of differential evolution. Control and Decision, 2004,19(1):49−52 (in Chinese with English abstract).

[54] Becerra RL, Coello Coello CA. Cultured differential evolution for constrained optimization. Computation Methods in Applied Mechanics and Engineering, 2006,195(33-36):4303−4322.

[55] Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. In: Proc. of the 6th Symp. on MicroMachine and Human Science. Piscataway: IEEE Service Center, 1995. 39−43.

[56] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997,11(4):341−359.

[57] Reynolds RG. An introduction to cultural algorithms. In: Fogel DD, ed. Proc. of the 3rd Annual Conf. on Evolutionary Programming. Cambridge: The MIT Press, 1994. 131−139.

[58] Zitzler E, Laumannns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou KC, *et al*., eds. Proc. of the EUROGEN 2001-Evolutionary Methods for Design, Optimization and Control with Barcelona, 2001. 95−100.

[59] Hinterding R. Constrained parameter optimization: Equality constraints. In: Proc. of the Congress on Evolutionary Computation (CEC 2001). Piscataway: IEEE Press, 2001. 687−692.

[60] Ho PY, Shimizu K. Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. Information Science, 2007,177(14):2985−3004.

[61] Zhou YR, He J. A runtime analysis of evolutionary algorithms for constrained optimization problems. IEEE Trans. on Evolutionary Computation, 2007,11(5):608−619.

[62] Lozano M, Herrera F, Krasnogor N, Molina D. Real-Coded memetic algorithms with crossover hill-climbing. Evolutionary Computation, 2004,12(3):273−302.

[63] Wang Y, Cai ZX, Guo GQ, Zhou YR. Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Trans. on Systems, Man, and Cybernetics (B), 2007,37(3):560−575.

[64] Jin Y. A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing, 2005,9(1):3−12.

[65] Amirjanov A. A changing range genetic algorithm. Int'l Journal for Numerical Methods in Engineering, 2004,61(15):2660−2674.

[66] Amirjanov A. The development of a changing range genetic algorithm. Computer Methods in Applied Mechanics and Engineering, 2006,195(19-22):2495−2508.

[67] Zhou SD, Sun ZX. A survey on estimation of distribution algorithms. Acta Automatica Sinica, 2007,33(2):113−224 (in Chinese with English abstract).

[68] Zhang Q, Zhou A, Jin Y. RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm. IEEE Trans. on Evolutionary Computation, 2008,12(1):41−63.

[69] Michalewicz Z, Deb K, Schmidt M, Stidsen T. Test-Case generator for constrained parameter optimization techniques. IEEE Trans. on Evolutionary Computation, 2000,4(3):197−215.

附中文参考文献:

[3] 郭观七,王勇.基于进化算法的约束处理技术.湖南理工学院学报(自然科学版),2006,19(4):15−18.

[22] 林丹,李敏强,寇纪淞.基于遗传算法求解约束优化问题的一种算法.软件学报,2001,12(4):628−632. http://www.jos.org.cn/1000-9825/12/628.htm

[30] 周育人,李元香,王勇,康立山.Pareto 强度值进化算法求解约束优化问题.软件学报,2003,14(7):1243−1249. http://www.jos.org.cn/1000-9825/14/1243.htm

[53] 谢晓锋,张文俊,张国瑞,杨之廉.差异演化的实验研究.控制与决策,2004,19(1):49−52.

[67] 周树德,孙增圻.分布估计算法综述.自动化学报,2007,33(2):113−224.

王勇(1980－),男,湖北天门人,博士生,讲师,主要研究领域为进化计算,约束优化,多目标优化.

周育人(1965－),男,博士,副教授,主要研究领域为演化计算,并行计算.

蔡自兴(1938－),男,教授,博士生导师,CCF高级会员,主要研究领域为人工智能,计算智能,智能控制.

肖赤心(1973－),男,博士生,讲师,主要研究领域为进化计算,神经网络.