一种求解约束优化问题的新算法*)

A New Evolutionary Algorithm for Constrained Optimization Problem

曾三友[1,2] 康立山[1] 丁立新[1]

(武汉大学软件工程国家重点实验室 武汉430072)[1](株洲工学院计算机系 株洲412008)[2]

Abstract A new evolutionary algorithm for constrained optimization problem is proposed in this study. In the algorithm, search space was gradually constracted to optimal solution while evolutionary algorithm was used repeatedly and incompletely, until the space constracted small enough. Then evolutionary algorithm ran fully. Real coding was used, and only crossover operators were adopted but two kinds of crossover operators were used in the new algorithm. The computation results dimonstrated that the new algorithm is superior to other methods in terms of solution quality, robustness and convergent speed.

Keywords Genetic algorithm, Functon optimization, Constrained function

演化算法基于达尔文的适者生存的原理，通过模拟大自然演化过程寻找问题的最优解。由于演化算法的全局性、灵活性、自适应性和稳健性，它特别适用于解象非线性、不可导和多峰等高难度优化问题。近年来，演化算法已经成功地解决了一些工程优化问题[1]。毫无疑问，演化计算是一类解决高难度优化问题最重要的办法之一。

本文提出一种新的演化算法可用于求解约束优化问题，它利用快速有效的不完全演化(见后面具体讨论)提供问题较优解的分布信息，通过该分布信息，定位最优解的可能范围，逐渐缩小搜索空间。在这种算法中，演化算法既用来定位最优解区域，实现搜索空间自动向最优解收缩，又用来最终求得最优解。它不仅可以求解无约束非线性函数优化问题，而且可以求解约束非线性函数优化问题。计算结果表明本文的算法在解的质量、稳定性和收敛速度等方面优于其它的演化算法。

1 约束优化问题的表示

不失一般性，约束优化问题描述为：

$$\min_x f(x) \tag{1}$$

$$\text{subject to:}\quad h_j(x)=0,\quad j=1,\cdots,m_e \tag{2}$$

$$g_j(x)\leqslant 0,\quad j=1,\cdots,m_i \tag{3}$$

$$x^l\leqslant x\leqslant x^u \tag{4}$$

其中，$x=(x_1,x_2,\cdots,x_n)$，$x^l=(x_1^l,x_2^l\cdots,x_n^l)$，$x^u=(x_1^u,x_2^u,\cdots,x_n^u)$

引进下列记号：

$S=\{x\mid x^l\leqslant x\leqslant x^u\}$叫做搜索空间，是一个n维长方体，S的边长$w=x^u-x^l$，中心$c=(x^l+x^u)/2$。

上述问题的目标函数、等式约束、不等式约束都是非线性的，这种形式称之为约束优化问题。

对于约束优化问题的求解，一般的办法是通过惩罚函数把原约束问题变成无约束问题。惩罚值适应函数为：

$$\Phi(x)=f(x)+\sum_{k=1}^{m_e}\alpha_k h_k^2(x)+\sum_{k=1}^{m_i}\beta_k(g_k(x))_+^2 \tag{5}$$

这里α_k和β_k是惩罚系数，$(g_k(x))_+=\max\{g_k,0\}$，本文针对惩罚函数问题求解。

2 新算法的设计思想

对于变量取值范围小、函数峰值不多，即使函数不可导甚至不连续，演化算法仍然能有效地求得最优解。若变量取值范围大、搜索空间高维、适应函数多峰，一般的演化算法在初始阶段，群体中的个体适应值改进较快，执行到一定代数后，群体中的个体分散在各个峰值的附近，此时遗传操作很难再改进后代的适应性，从而导致收敛速度变慢。针对这种情况，本文算法利用快速有效的不完全演化(见后面具体讨论)提供问题较优解的分布信息，并分配一个足够大的空间存放当前已经得到的较优解，我们称之为优解池，并把这个优解集合作为分析优解分布的样本空间。利用该样本空间，定

*)国家自然科学基金(60073043,70071042)。曾三友 博士生，现从事演化计算研究。

位最优解可能在哪个较小范围。怎样确定这个较小范围呢?为此我们作如下假设:某一点和其邻域的点有相似的适应值(这是一个比较宽松的假设,也是遗传算法有效的一个条件)。定位最优解较小范围方法如下:将优解池的个体分组,每个小组的个体在同一个较小的n维长方体里,我们认为最优解在具有较好平均适应值的小组(平均适应值大于或等于群体平均适应值的小组)对应的小n维长方体里。下一步,以这些较小的n维长方体作为搜索空间执行不完全演化,更新优解池里的解,再确定最优解在哪个更小的n维长方体里…如此迭代,直到搜索空间的边长小于允许误差,把优解池里的最优解作为问题的最优解。这样做的特点是:粗略定位最优解速度较快,但精确定位最优解速度较慢。考虑到此特点,在本算法设计中,我们让搜索空间收缩到适当大小时,停止继续收缩,转而彻底演化求得满足精度要求的最优解或者采用其他的快速算法求最优解,以提高求解速度。

2.1 本文算法的理论依据

本文算法的设计依据除基于演化算法的收敛性定理外,还基于如下定理1:

定理1 设f(x)是定义在闭n维长方体S上的连续函数,闭n维长方体序列$\{S_k\}$满足:

①$S_0\supset S_1\supset S_2\supset\cdots\supset S_k\cdots$,$S_0=S$;

②$\lim_{k\to\infty}R_k=0$,R_k是S_k的半径;

③S_k里包含f(x)在闭n维长方体S上的最优解,$k=1,2,\cdots$;

则存在唯一的$s^*\in S_k$,$k=1,2,\cdots$,且s^*是f(x)在闭n维长方体S上的最优解

证明:由条件③S_k里包含f(x)在闭n维长方体S上的最优解,$k=1,2,\cdots$,设$s_k\in S_k$是f(x)在S的一个最优解,根据条件①②并利用完备距离空间的区间套定理,存在唯一的$s^*\in S_k$,使得$\lim_{k\to\infty}s_k=s^*$,下面证明s^*是f(x)在闭n维长方体S上的最优解。反设s^*不是f(x)在闭n维长方体S上的最优解,记M为f(x)在闭n维长方体S上的最优值(不妨设为最大值),$m=f(s^*)$,根据$\lim_{k\to\infty}s_k=s^*$以及f(x)的连续性,可知当k充分大时,$|f(s_k)-f(s^*)|<\frac{M-m}{2}$,然而$f(s_k)-f(s^*)=M-m$,这是一个矛盾。这证明了$s^*$是f(x)在闭n维长方体S上的最优解。

在实际问题中,由于适应值函数不连续点很少、搜索空间自动收缩保持定理条件①②满足而且条件③也在概率意义下满足,因此,该定理具有一定指导意义,但是优化问题中的适应值函数终究不一定满足连续性条件、定理的条件③也只是在概率意义下满足、而且演化算法是一个随机算法,故本算法不能绝对保证每次都能找到最优解。

2.2 遗传算子

遗传算子(g)的一般定义:$s=g(p_1,p_2,\cdots,p_m)$,p_i($p_i\in P$)是群体P($P\subset S$)中的个体,$i=1,2,\cdots,m$,$s\in S$是由m个父体$p_1,p_2,\cdots,p_m$通过遗传操作产生的后代个体,S为搜索空间。

本算法只采用按如下方式定义的杂交算子:

$$\text{杂交算子1}:s=\sum_{i=1}^{n}a_i(p_i-c)+c$$

这里c为搜索空间S的中心,从群体P中随机选取p_i,随机产生a_i满足:$-0.5\leqslant a_i\leqslant 1.5$。

$$\text{杂交算子2}:s=\sum_{i=1}^{n}a_iP_i$$

这里从群体P中随机选取p_i、随机产生a_i满足:$-0.5\leqslant a_i\leqslant 1.5$且$\sum_{i=1}^{n}a_i=1$。

2.3 不完全演化

一般来说,对于变量取值范围小、函数峰值不多,即使函数不可导甚至不连续,演化算法仍然能有效地求得最优解。对于变量取值范围大,搜索空间高维、适应函数多峰、一般的演化算法在初始阶段,群体中的个体适应值改进较快,执行到一定代数后,群体中的个体分散在各个峰值的附近,此时遗传操作很难再改进后代的适应性,从而导致收敛速度变慢。当群体中的个体分散在各个峰值的附近时、终止演化算法,本文称这种演化为不完全演化。假设某一点和其邻域的点有相似的适应值,那么最优解附近的点有较好的适应值。假设已具有足够多的较优点信息(有最优解附近的点包含在信息里),则可以粗略定位最优解,缩小搜索空间。现在的问题是怎样获取足够多的较优点信息?不完全演化的一个优点是快速提供多个较优点,从而是提供较优点信息的好工具;另一个优点是随机性,多次进行不完全演化,每次可以获取不同的较优点。于是多次重复不完全演化,可以获取足够多的较优点信息。重复不完全演化多少次才能提供足够多的较优点信息呢?搜索空间越大,函数的峰越多,需要越多的较优解信息来定位最优解的范围。因此,不完全演化的重复次数强依赖于搜索空间的边长和适应值函数的峰的数目、但随搜索空间的收缩及较优点信息的逐渐积累,较少次数的不完全演化就可以提供足够的信息。

2.4 新算法描述

(1)初始化不完全演化重复次数r、初始化优解池better,当前搜索空间的边长w=问题搜索空间边长。

(2)分组:对优解池里的元素分组,使每个组里的元素在同一个边长为w的长方体里。

(3)选择平均值较好组的长方体作为最优解的可

能空间，依次让 s 取这些长方体，以 s 为搜索空间不完全演化 r 次，更新优解池 better。

(5)如果边长 w 足够小，转(6)；否则，缩短边长 w，减小 r，转(2)。

(6)以 s 为搜索空间，完全演化，或采用其它快速算法。

3 数值结果

为了检验算法的性能，本文选择了两个典型的机械优化问题各进行了50次计算。测试函数如下：

$F_1: f_1(x)=0.6224x_1x_3x_4+1.7781x_2x_3^2+3.1661x_3^2x_4+19.84x_3^2x_1$

subject to: $g_1(x)=0.0193x_3-x_1\leqslant 0$

$g_2(x)=0.00954x_3-x_2\leqslant 0$

$g_3(x)=750\times 1728-\pi x_3^2x_4-(4/3)\pi x_3^3\leqslant 0$

$g_4(x)=x_4-240\leqslant 0$

$F_2: f_2(x)=1.10471x_1^2x_2+0.04811x_3x_4(14.0+x_2)$

Subject to: $g_1(x)=\tau(x)-\tau_{max}$

$g_2(x)=\sigma(x)-\sigma_{max}$

$g_3(x)=x_1-x_4$

$g_4(x)=0.10471x_1^2+0.04811x_3x_4(14.0+x_2)$

$g_5(x)=0.125-x_1$

$g_6(x)=\delta(x)-\delta_{max}$

$g_7(x)=P-P_c(x)$

这里 $\tau(x)=\sqrt{(\tau')^2+2\tau'\tau''x_2/2R+(\tau'')^2}$

$\tau'=\frac{P}{\sqrt{2}x_1x_2}$，$\tau''=\frac{MR}{J}$，$M=P(L+\frac{x_2}{2})$

$R=\sqrt{\frac{x_2^2}{4}+(\frac{x_1+x_3}{2})^2}$

$J=2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12}+(\frac{x_1+x_3}{2})^2\right]\right\}$

$\sigma(x)=\frac{6PL}{x_4x_3^2}$，$\delta(x)=\frac{4PL^3}{Ex_3^3x_4}$

$P_c(x)=\frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}(1-\frac{x_3}{2L}\sqrt{\frac{E}{4G}})$

$P=6000$，$L=14$，$E=30\times 10^6$，$G=12\times 10^6$，

$\tau_{max}=13600$，$\sigma_{max}=30000$，$\delta_{max}=0.25$

表 1 各种算法对 $f_1(x)$ 的计算结果比较

	Cao[2]	Kannan[3]	GeneAS[4]	SAPGA[5]	This paper
x_1	1.000	1 125	0.9375	0.8125	0.727591963412
x_2	0.625	0 625	0.5000	0.4375	0.3596497937573
x_3	51.1598	58.291	48.3290	40.3239	37. 6990135988
x_4	90.7821	43.690	112.6790	200.00	239. 9999999999
g_1	-0.011921	-0.000016	-0.004750	-0.034324	-0.00000100095565
g_2	-0.136592	-0.068904	-0.038941	-0.052847	-0.0000012040252
g_3	-13584.58	-21.22010	-3652.8768	-27.105845	-0.12813537
g_4	-149 2179	-196.31	-127.321	-40.000	-0.00000000
$f_1(x)$	7108.6160	7198.0428	6410.3811	6288.7445	5804. 38767986

表 2 各种算法对 $f_2(x)$ 的计算结果比较

	Ragsdell[6]	SAPGA[5]	Deb[7]	SAPGA[5]	This paper
x_1	0.2455	0.2444	0.2489	0.2088	0.2057154703
x_2	6.1960	6.2189	6.1730	3.4205	3.47185980684
x_3	8.2730	8.2519	8.1789	8.9975	9.03591338946
x_4	0.2455	0.2444	0.2533	0.2100	0.20576200846
g_1	-5743.82652	-5743.502	-5758.603777	-0 337812	-2.4777203801
g_2	-4.715097	-4.015209	-255 5769	-353.902604	-0.0019141320
g_3	0.000000	0.000000	-0.004400	-0.00120	-0.00004653816
g_4	-3.020289	-3.022561	-2.982866	-3.411865	-3.43273877903
g_5	-0.120500	-0.119400	-0.123900	-0.08380	-0.08071547030
g_6	-0.234208	-0.234243	-0.234160	-0.235649	-0.23553918641
g_7	-3604.275	-3490.4694	-4465.270928	-363.232384	-2.52217228247
$f_2(x)$	2.38593732	2.38154338	2.43311600	1.74830941	1.725536372771

F_1是一个压力容器设计问题。设计参数是容器的各种尺寸，目标函数是熔制压力容器的总成本，约束条件保证容器符合美国机械工程师学会标准，这是一个约束优化问题。F_2是一个熔制梁设计问题，它也是一个约束优化问题。对2个例子各计算50次，参数设置如下：群体规模23，杂交父体数2，在不完全演化中，选择杂交算子1的概率为0.5，杂交算子2的概率为0.5；在彻底完全演化中，杂交算子1的概率为0.2，杂交算子2的概率为0.8，每次不完全演化的杂交数不超过1000，开始时，

(下转第116页)

RECOM 还将中间件的功能性属性(远程方法调用)与非功能性属性相分离。这些都得益于反射技术的运用。

反射技术在中间件中的应用研究才刚刚起步，Illinois 大学的 dynamicTAO 项目[9]，Lancaster 大学的 OpenORB 等项目[10]，在这方面进行了开创性的研究。但是 dynamicTAO 和 OpenORB 都是将 ORB 的某些内部实现开放出来。我们将绑定作为中间件的自表示，而每个绑定完全监控了分布式对象间通信的全过程，换句话说，也就是这种自表示完全地反映了中间件实现的各个方面。我们采用绑定具体化反射模型来设计中间件，从而推导出具有指导意义的设计思路，并实现了一个反射中间件的原型 RECOM。

进一步的工作包括：丰富 RECOM 的绑定生成器和反射层；将 RECOM 运用到一个实际应用中；将实现和配置反射层的代码(按反射技术的术语即元程序)从应用程序分离出来，仅在需要的时候才与应用程序发生关联，因为编写元程序需要专门的知识，可由专人完成；研究不同反射层之间的制约关系；等等。

参 考 文 献

1 Kiczales G. Towards a New Model of Abstraction in Software Engineering. In. Proc. of the IMSA'92 Workshop on Reflection and Meta-level Architecture, 1992

2 Maes P Concepts and Experiments in Computational Reflection. OOPSLA'87, Sigplan Notices, Oct. 1987

3 RM'2000 — Workshop on Reflective Middleware. 2000. Available at: http: // www. comp lancs. ac uk/computing/RM2000

4 Ferber J. Computational Reflection in Class-based Object Oriented Languages. OOPSLA'89, Sigplan Notices, Oct. 1989

5 Cazzola W. Evaluation of Object-Oriented Reflective Model. In. Proc. of ECOOP Workshop on Reflective Object-Oriented Programming and Systems (EWROOPS'98). Brussels, Belgium, Jul. 1998

6 Sun Microsystems. Java Core Reflection Available at: http. // Java. sun. com/products/jdk/1. 2/docs/guide/reflection/index. html

7 Hayton R. ANSA Team FlexiNet Architecture. Architecture Report, Citrix Systems (Cambridge) Limited, Feb. 1999

8 The CORBA Architecture and Specification, Reversion2. 2 Object Management Group, Inc. Feb. 1998

9 Roman M, Kon F, Compell R. Design and Implementation of Runtime Reflection in Communication Middleware: the Dynamic TAO Case. In: Proc. of the ICDCS'99 Workshop on Middleware Austin, Texas, May 1999

10 Costa F, Blair G. A Reflective Architecture for Middleware: Design and Implementation. In. Proc. of the ECOOP'99 Workshop for PHD Students in Object Oriented Systems, Lisbon, Jun. 1999

(上接第97页)

不完全演化重复次数在10～50之间，搜索空间收缩迭代过程中，搜索空间边长按等比缩小，不完全演化重复次数按等差减小。$f_1(x)$的50个结果中两个结果在6500～6000之间，其余的均小于6000，表1是各种算法对 $f_1(x)$的计算结果比较。$f_2(x)$的50个结果都小于1.9，表2是各种算法对 $f_2(x)$的计算结果比较。

结论 本文为解约束优化问题提出了一种新的演化算法。在这个方法里，演化算法不完全地重复执行提供问题较优解的分布信息，利用该分布信息，本算法定位最优解的可能范围，逐渐缩小搜索空间。在这种算法中，演化算法既用来定位最优解区域，实现搜索空间自动向最优解收缩，又用来最终求解最优解。2个测试问题的计算结果表明本文的算法在解的质量、稳定性和收敛速度等方面优于一般演化算法。

参 考 文 献

1 Michalewicz Z, et al. Evolutionary algorithms for constrained engineering problems. Computers & Industrial Engineering J., 1996, 30, 851～870

2 Cao Y J, Wu Q H. Mechanical Design Optimization by Mixed-Variabel Evolutionary Programming. In: Zbigniew Michalewicz, Xin Yao, eds. Proc. of the 1997 Int Conf. on Evolutionary Computation. Indianapolis Indiana: IEEE Sevice Center, 1997. 443～446

3 Dannan B D, Kramer S N. An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design. *Journal of Mechanical Design*, *1994*, *116*: *318～320*

4 Kalyanmoy Deb. DeneAS: A Robust Optimal Design Technique for Mechanical Component Design. In: Zbigniew Michalewicz, eds Evolutionary Algorithms in Engineering Application. Berlin: Springer-Verlag, 1997. 497～514

5 Coello C A. Self-Adaptive Penaltres for GA-based Optimization. In: Angelme P, ed. Proc. of the 1999 Congress on Evolutionary Computation Washington D. C. USA: IEEE Service Center, 1999, 3: 2159～2169

6 Ragsdell K M, Phillips D T. Optimal Designed Class of Welded Structures Using Geometric Programming. ASME Journal of Engineering for Industrial, 1976, 98(3), Series B. 1021～1025

7 Lalyanmoy Deb. Optimal Design of a Welded Beam by Genetic Algorithms AIAA Journal, 1991, 29(11): 2013～2018