计算机科学2002Vol.29№.6

遗传算法对约束优化问题的研究综述

Study Based on Genetic Glgorithms for Constrained Optimization

余 文 李人厚

(西安交通大学系统工程研究所 西安710049)

Abstract In all aspects of GA, including the constraints-handling technique, encoding, fitness function and genetic operator, etc, GA is able to provide an alternative optimization technique for constrained problems that are solved in a wide variety of domains. The penalty one is the most valid and flexible in the all techniques. The fundamental methods that can solve the nonlinear programming problems in the GA area are reviewed and discussed in this paper.

Keywords Genetic algorithms, Constraints-handling techniques, Nonlinear programming

1 引言

工程、数学等领域经常遇到大量的约束优化(或非线性规划)问题,需要对约束条件进行处理。目前,还没有一种通用的传统优化方法,能够处理各种类型的约束。相比,遗传算法(GA)在这一领域,比其它方法更有巨大优势和应用潜力。遗传算法的群体搜索策略和不依赖梯度信息的计算方式,使得它在处理约束优化问题时比传统搜索算法通用和有效[1]。许多处理约束优化问题的传统算法都可以直接或改进后而用于GA。此外,由于GA是一种随机算法,既可以在编码时或设计遗传算子时加以考虑,也可以在每一代通过修正算法使所产生后代个体满足约束条件。因此,GA比其它方法又有许多独特之处。然而,与处理无约束优化问题的大量文献对比,所见的用GA处理约束优化问题的文章还显得十分不足。粗略地,可以把GA处理约束的方法分为五类,它们是:空间限定法、拒绝法、修复法、修改算子法和惩罚法。

2 约束处理方法

空间限定法的基本思想是通过对染色体搜索空间的大小加以限制,使得搜索空间中表示一个个体的点与解空间中表示一个可行解的点一一对应。该方法经常能通过编码限定技术,保证染色体的有效性。在一些简单或规则的约束条件(如$a\leqslant x\leqslant b$之类)中,该方法可以产生较高的搜索效率。

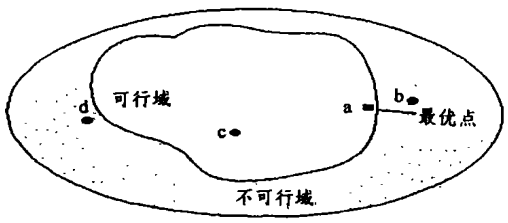

图1 解空间:可行域和不可行域

拒绝法是简单地抛弃进化过程中所出现的不可行染色体。对不可行的染色体,使其适应度降低为0。这样,不可行染色体将不会遗传到下一代种群。该方法最为简单,但缺乏有效性。在约束条件严格时,依赖遗传算子在搜索空间生成新个体的能力以及群体多样性将大大降低。拒绝法的另一缺点是无法利用不可行区域中进化个体的有用信息,只能期望遗传搜索从可行解的一侧趋近和得到最优解。

修复法是通过某种修复程序修复进化过程所出现的不可行染色体,使其变为可行的染色体。该方法多见于组合优化问题,原因是组合优化中修复程序相对容易产生。修复法的优点在于对个体编码、遗传运算等没有过多的附加要求,并可期望遗传搜索能够从可行解和不可行解的两侧最终趋近最优解。但修复法的缺点是对问题本身的依赖性,且以扩大搜索空间为代价。值得注意的是,对于某些问题,修复过程甚至能比原问题的求解更加复杂[2]。

修改算子法是一类处理约束的自然方法。它是针对性地设计适于问题表达的方式和专门的遗传算子,使可行解仍变化到可行解,即遗传算子在可行域上的操作是封闭的[4]。目前,在许多领域研究者们通过改进编码方式和遗传算子,成功地开发了多种非常独特的遗传算法,Michalewicz[5]等指出这种方法比惩罚法较为可靠,缺点是对问题本身的依赖性。

前面几种策略的共同优点是不会给下代产生不可行解。但在一些具有高度约束的问题中,可行区域可能只占解空间中较小部分或比例。这时,可行解相对较难产生。Glover和Greenberg建议的约束管理技术允许在搜索空间的不可行域中进行搜索,这比将搜索限制在可行域内的方法能更快地获得更好的最终解[4,6],惩罚法就是这类在遗传搜索中考虑不可行解的技术。

惩罚策略是GA求解约束优化问题中最常用的技术。通常,约束优化问题大都可以归结为具有等式和/或不等式约束前提下最优化某个目标函数的非线性规划问题,因此,我们以非线性规划问题为例来详细说明GA处理约束优化问题的实用技术,特别是惩罚策略。

3 惩罚法

3.1 非线性规划问题形式描述

非线性规划的形式描述如下:

$$\max \quad f(x) \tag{1}$$

$$\text{s.t.} \quad g_i(x)\leqslant 0,\ i=1,2,\cdots,m_1 \tag{2}$$

$$h_i(x)=0\ i=m+1,\cdots,m(=m_1+m_2) \tag{3}$$

余 文 博士生,硕士,现研究复杂系统智能控制理论和方法,遗传算法等。李人厚 博士生导师,研究复杂系统智能控制理论和方法,CSCW,优化算法等。

$$x \in X \tag{4}$$

这里 $f(x), g_i(x), h_i(x)$ 均为 n 维欧氏空间 E^n 上的 n 元函数。$f(x)$ 称为目标函数；$g_i(x) \leqslant 0$ 称为不等式约束；$h_i(x)=0$ 称为等式约束。$x=(x_1, x_2, \cdots, x_n)^T \in X$ 称为设计变量或决策变量，X 为 n 维欧氏空间 E^n 的子集。数学上，把 X 中满足所有约束的决策变量的集合：

$$R=\{x: g_i(x) \leqslant 0, i=1,2,\cdots,m_1; h_i(x)=0, i=m_1,\cdots,m_1+m_2; x \in X\} \tag{5}$$

称为问题的可行集，R 中的点称为可行解。可行集中的最优解称为全局最优解(对于一般的约束问题，可行集是指满足所有约束条件的决策变量的集合)。求解非线性规划的问题就是找到可行集中的最优解。

3.2 惩罚函数

惩罚法是 GA 求解非线性规划问题最常用的方法。不同于前面几种方法，它容忍群体中的个体可以在一定程度上违反给定的约束，但必须在个体的适应度计算上体现出约束的违反程度，并予以相应的惩罚。这种违反约束的程度由惩罚函数来确定。在 GA 中，惩罚法通过降低不可行解的适应度，以减小它们被选择的概率，其本质是利用惩罚技术，把约束问题转化为无约束的优化问题。

解空间通常包含可行域和不可行域两个部分。全局最优解位于可行域中的某个位置，由于事先难以对该点及区域的情况作出判定，因此对进化中出现的违反约束的个体，只能依据个体破坏约束的程度加以相应惩罚。通常，惩罚函数能够基于以下三种途径来得到：

(1)(非可行)个体到可行区域绝对距离的函数。例如，对于式(2)、(3)的约束，惩罚函数可以按下式来定义：

$$p(x)=-\theta(t)\cdot\left(\sum_{i=1}^{m_1}\alpha_i\cdot((g_i(x))^q+\sum_{j=m_1+1}^{m}\beta_j\cdot|h_j(x)|^q\right) \tag{6}$$

式中 q 是正整数，一般取1或2，$\theta(t)>0$ 称为惩罚因子，t 为进化代数。

(2)当前种群中所有非可行个体相对距离的函数。

(3)自适应惩罚项的函数。

大多数方法都采用第一种途径。当不可性区域相对较大时，为避免过重或过轻惩罚，可采用后两种途径。一个例子是(24)式中的惩罚。

惩罚策略的主要问题是如何确定适当的惩罚。需要谨慎地在过轻或过重惩罚之间找到平衡，才能有效地引导遗传搜索到达解空间最好的区域。

3.3 评价函数

惩罚函数确定后，还需要建立相应的评价函数。在一定条件下，评价函数的解等同于原问题中的解。适应度评价函数的构造通常采用加法和乘法两种途径。

加法途径是给原目标函数简单地添加上附加的惩罚项。即评价函数为：

$$eval(x)=f(x)+p(x) \tag{7}$$

这里 x 代表染色体，$f(x)(>0)$ 为目标函数，$p(x)$ 为惩罚项，它通常要满足：

$$\begin{cases} p(x)=0 & \text{当 } x \in R \\ p(x)<0 & \text{否则} \end{cases} \tag{8}$$

为了避免评价函数出现负值，可以对 p(x)作如下限制：

$$\max\{-p(x)\} \leqslant \min\{f(x)\} \tag{9}$$

很明显：对加法途径，当惩罚因子 θ 增加时，评价函数式(7)的解将收敛到原问题中的解。

乘法途径则把评价函数取为目标函数与罚函数的乘积。即：

$$eval(x)=f(x)\cdot p(x) \tag{10}$$

这时，惩罚项 $p(x)$ 要求满足：

$$p(x)=\begin{cases} p(x)=1 & \text{当 } x \in R \\ p(x)<1 & \text{否则} \end{cases} \tag{11}$$

惩罚法借用了某些传统优化技术中罚函数法的基本思想。它的优点是灵活、易于实现，只需稍微改变评价函数，便能较为有效地解决各类问题。另外，惩罚法通过保持一定数量的不可行解并利用它们的有用信息，使遗传群体从可行域和不可行域的两侧同时逼近到最优解，这能改善遗传算法的进化效果。

3.4 惩罚函数分类

惩罚策略是通过惩罚函数来体现惩罚的。惩罚策略基本分为两种类型：一种是定量惩罚，另一种是变量惩罚。

定量惩罚策略多用于简单约束问题，而在有复杂约束的优化问题中，采用变量惩罚技术往往能得到更有效的效果。变量惩罚一般包含两部分：可变惩罚率和违反约束的惩罚量。可变惩罚率可按下面两个因素来调节：违反约束程度和进化迭代次数。前者随违反约束的程度变得严重而增加惩罚压力，但不随进化过程而变，属于静态惩罚，最早由 A. S. Homaifart 提出。后者随着进化过程的进展而增加惩罚压力，属于动态惩罚，由 Jonies 和 Houck 提出[7]。

惩罚可以进一步依据是否与问题相关而分为：问题依赖的和问题独立的。大多数惩罚是问题依赖型的。

惩罚还可以依据是否含有参数划分为：含参数的和不含参数的。大多数惩罚技术是含参数的。含参数的惩罚几乎总是问题依赖型的。

除了上面的分类外，还可以依据惩罚函数的具体特点来分类。目前，惩罚函数依据其特点可分为如下几类：(1)静态惩罚。惩罚函数不随进化过程而变；(2)动态惩罚。惩罚函数随进化过程的进展而增加；(3)退火惩罚[8,10]。惩罚函数借鉴了模拟退火法的技术和策略；(4)自适应惩罚[11,12]。可利用搜索过程中反馈的信息自适应调节惩罚因子；(5)启发式惩罚[13]。启发式惩罚函数保证：任一可行解都优于非可行解；(6)双重惩罚。群体内实行双重的惩罚标准以图惩罚力度的稳健性；(7)逐次惩罚。该方法由 Schoenaur 提出，与其它方法大不相同，它使用了分享策略、翻转阈值及约束排序次序等概念，并一次只处理一个约束[14]。

按照惩罚函数的分类，介绍几种求解非线性规划问题遗传算法的构造惩罚函数的方法。

(1)Homaifar，Qi 和 Lai 的方法(静态惩罚) Homaifar 等考虑如下非线性规划问题：

$$\min \quad f(x) \tag{12}$$

$$\text{s.t.} \quad g_i(x) \leqslant 0, \ i=1,2,\cdots,m_1 \tag{13}$$

评价函数为

$$eval(x)=f(x)+p(x) \tag{14}$$

惩罚项 $p(x)$ 的表达式如下

$$p(x)=\begin{cases} 0 & \text{若 } x \text{ 可行} \\ \sum_{i=1}^{m_1} r_i\cdot g_i(x) & \text{其它} \end{cases} \tag{15}$$

其中 r_i 是约束 i 的可变惩罚系数。对于每个约束，又可分为几个违反级，按违反的级，r_i 相应变化。

(2)Joiness 和 Houck 的方法(动态惩罚) Joiness 和

Houck 考虑如下非线性规划问题

$$\min \quad f(x) \tag{16}$$

$$\text{s.t.} \quad g_i(x)\leqslant 0,\ i=1,2,\cdots,m_1 \tag{17}$$

$$h_i(x)=0\ i=m+1,\cdots,m(=m_1+m_2) \tag{18}$$

评价函数为

$$eval(x)=f(x)+(Ct)^{\alpha}\sum_{i=1}^{m}d_i^{\beta}(x) \tag{19}$$

这里，t 是迭代次数，α,β 是调节惩罚大小的参数。单个约束 $d_i(x)$ 的惩罚项和惩罚因子按下式计算：

$$d_i(x)=\begin{cases}0 & \text{若 } x \text{ 可行}\\ |g_i(x)| & i=1,2,\cdots,m_1\\ |h_i(x)| & i=m_1+1,\cdots,m=m_1+m_2\end{cases} \tag{20}$$

与 Homaifar 的方法相比，Joiness 和 Houck 的惩罚因子随进化过程而变，而 Homaifar 的方法中则随违反级而变。前者为动态惩罚，后者为静态惩罚。

(3)Gen 和 Cheng 的方法(自适应惩罚) Gen 和 Cheng 等考虑如下非线性规划问题：

$$\max \quad f(x) \tag{21}$$

$$\text{s.t.} \quad g_i(x)\leqslant b_i,\ i=1,2,\cdots,m \tag{22}$$

评价函数为

$$eval(x)=f(x)\cdot p(x) \tag{23}$$

惩罚项 $p(x)$ 的构造如下

$$p(x)=1-\frac{1}{m}\sum_{i=1}^{m}\left(\frac{\nabla b_i(x)}{\nabla b_i^{\max}}\right)^{\alpha} \tag{24}$$

这里：$$\nabla b_i(x)=\max\{0,g_i(x)-b_i\} \tag{25}$$

$$\nabla b_i^{\max}=\max\{\varepsilon,\nabla b_i(x)\mid x\in P(t)\} \tag{26}$$

其中 $\nabla b_i(x)$ 是约束 i 在 x 的违反量，α 是用来调节惩罚严厉性的参数。$\nabla b_i^{\max}$ 是当前种群 $P(t)$ 中约束 i 的最大违反量，ε 是避免为零的小正数。该惩罚方法可以通过逐代自动调节惩罚率，维持信息保留和不可行惩罚压力的平衡，以避免过惩罚。

值得注意：惩罚法中惩罚系数对解的质量有重要影响。当惩罚系数不适当时，算法可能收敛于不可行解；另一方面，当惩罚系数过大时，惩罚法等价于拒绝策略[20]。

4 编码和遗传运算

在非线性规划优化中，GA 大多采用连续表达技术来表达给定问题的解。这种编码通常为浮点编码[8]和实数编码[9]。连续编码在处理约束方面比二进制编码具有明显的优点，容易设计出适合一些约束的遗传运算。在连续编码中，每个染色体简单地为一个与解向量维数相同的实向量，如 $x=[x_1,\cdots,x_k,\cdots,x_n]$。

以最基本的单点交叉和变异为例，下述运算适用于一定的约束优化问题。

4.1 交叉运算

令双亲为 $x=[x_1,x_2,\cdots,x_n]$ 和 $y=[y_1,y_2,\cdots,y_n]$，若交叉点在第 k 位，下面给出一些较常用的交叉运算：

(1)简单交叉 模仿了二进制表达的交叉，其产生的后代为：

$x'=[x_1,x_2,\cdots,x_{k-1},y_k,\cdots,y_n]$ 和 $y'=[y_1,y_2,\cdots,y_{k-1},x_k,\cdots,x_n]$ (27)

(2)部分算术交叉 产生的后代为：

$$x'=[x_1,x_2,\cdots,x_{k-1},x_k\cdot\lambda+y_k\cdot(1-\lambda),\cdots,x_n\cdot\lambda+y_n(1-\lambda)] \tag{28}$$

$$y'=[y_1,y_2,\cdots,y_{k-1},y_k\cdot\lambda+x_k\cdot(1-\lambda),\cdots,y_n\cdot\lambda+x_n(1-\lambda)] \tag{29}$$

(3)单点算术交叉 产生的后代为：

$$x'=[x_1,x_2,\cdots,x_{k-1},x_k\cdot\lambda+y_k\cdot(1-\lambda),x_{k+1},\cdots,x_n] \tag{30}$$

$$y'=[y_1,y_2,\cdots,y_{k-1},y_k\cdot\lambda+x_k\cdot(1-\lambda),y_{k+1},\cdots,y_n] \tag{31}$$

(4)(整体)算术交叉 定义为两个向量的线性组合：

$$x'=\lambda_1\cdot x+\lambda_2\cdot y \text{ 和 } y'=\lambda_1\cdot y+\lambda_2\cdot x \tag{32}$$

按对乘子 λ_1,λ_2 限制的不同，可获得凸交叉[8]、仿射交叉和线性交叉[15]等三种交叉，其中凸交叉最为常用。它们的区别在下式中描述：

1)凸交叉：$\lambda_1+\lambda_2=1,\lambda_1>0,\lambda_2>0$ (33)

2)仿射交叉：$\lambda_1+\lambda_2=1$ (34)

3)线性交叉：$\lambda_1+\lambda_2\leqslant 2,\lambda_1>0,\lambda_2>0$ (35)

(5)基于方向的交叉 该算子最初由 A. Wright 在文[16]提出，其特点是利用了目标函数的值来确定搜索方向。另外，该交叉运算只产生一个后代 x'：

$$x'=r\cdot(x-y)+x \tag{36}$$

式中 r 是0和1之间的随机数，并假定 x 不比 y 坏。

此外，还有一些交叉运算，如几何交叉、球面交叉等。

4.2 变异运算

变异运算也有多种，如均匀变异、边界变异[17]和非均匀变异[18]和基于方向的变异。设个体 $x=[x_1,\cdots,x_k,\cdots,x_n]$，则前三种变异的区别仅表现在下式中变异基因 x'_k 的不同取值：

$$x'=[x_1,\cdots,x'_k,\cdots,x_n] \quad x'_k\in[x_k^L,x_k^U] \tag{37}$$

1)均匀变异：若变异基因 x'_k 在指定范围 $[x_k^L,x_k^U]$ 内随机选取。这里 x_k^U,x_k^L 分别取变异量 x_k 的上下界，它们与约束有关。

2)边界变异：若 x'_k 等概率地取用变异量的上界或下界，即 x_k^U 或 x_k^L。

3)非均匀变异或动态变异：若变异量 x'_k 按如下选取：

$$x'_k=x_k+\Delta(t,x_k^U-x_k) \text{ 或 } x'_k=x_k-\Delta(t,x_k-x_k^L) \tag{38}$$

$\Delta(t,y)$ 随代数 t 增加而趋于0，并有如下形式：

$$\Delta(t,y)=y\cdot r\cdot\left(1-\frac{t}{T}\right)^{b} \tag{39}$$

其中，r 是0和1之间的随机数，T 为最大代数，b 为参数。非均匀变异有时也叫动态变异。

4)基于方向的变异：该变异由 Gen 和 Liu 提出[19]，与前三种变异有较大的不同，基于方向的变异得到：

$$x'=x+r\cdot d \tag{40}$$

这里 d 为目标函数的近似梯度，有时，可随机选择一个自由方向来替代。

上述算子中，部分算术交叉、单点算术交叉和凸交叉对于可行域为凸集情况，运算是封闭的；基于方向的交叉使搜索朝着最有可能的方向探索；几何交叉在处理乘积约束方面具有优势，并与球面交叉算子一样经常被用于可行域边界的搜索策略。对变异算子而言，当最优解在可行域边界上或其附近时，边界变异算子较为有效。动态变异是为提高精度，增加微调而设计的；基于方向的变异类似于梯度搜索算子。

一些运算可能会导致不可行解。但在罚函数中，这是允许的。

评论 工程、数学等领域经常遇到大量的约束优化问题。解决这些问题的关键是如何处理给定的约束。GA 在约束优化领域，有巨大的优势和应用潜力。在 GA 处理约束的技术和

策略中，惩罚技术最为通用和有效。惩罚技术本质是通过惩罚不可行解，把约束问题转化为无约束的优化问题。

然而，如何设计惩罚函数以有效地惩罚非可行解，对问题的解决至关重要，目前尚缺少普遍意义的指导原则。另外，对特定的约束优化问题，需要谨慎地选择合适的遗传运算。本文系统地阐述了遗传算法处理约束的策略和方法，特别是它的惩罚策略和遗传操作技术，希望能为实际工程应用中设计和开发具有更高性能、适用约束优化的进化算法提供和开拓一些新的思路。

参 考 文 献

1 Gen M, Cheng Runwei. Genetic algorithms and engineering design. New York: Wiley-Interscience, 2000

2 Gen M, Cheng Runwei. Genetic algorithms and engineering optimization. New York : Wiley-Interscience, 2000

3 Herrera F, Verdegay J L. Genetic algorithms and soft computing. Heidelberg Physica-Verlag, 1996

4 Man K F, Tang K S, Man S K. Genetic algorithms : concepts and designs London; New York : Springer, 1999

5 Michalewicz Z, Dasgupta D, et al. Evolutionary algorithms for industrial engineering problems. International Journal of Computers & Industrial Engineering, 1996, 30(4)

6 Glover F, Greenberg H. New approaches for heuristic search: A bilateral linkage with artificial intelligence. European Journal of Operational Research, 1989, 39: 119～130

7 Tanese R. distributed genetic algorithms for function optimization: [Ph. D. Thesis]. University of Michigan , Ann Arbor, MI, 1989

8 Michalewicz Z. Genetic Algorithm + Data Structure = Evolution Programs. Springer-Verlag, New York , 1994

9 Davis L. Handbook of genetic algorithms, New York, Van Nostrand Reinhold, 1991

10 Michalewicz Z, Attia N. Evolutionary algorithms for constrained engineering problems. In: Proc. of the third annual conf. on Evolutionary Programming, 1994. 98～108

11 Bean J C M, Hadj-Alouane A B. A dual genetic algorithm for bounded integer programs: [Technical Report TR 92-53]. Department of industrial and operations engineering , the University of Michigan, 1992

12 Hadj-Alouane A B, Bean J C. A Genetic algorithm for multichoice integer programs: [Technical Report TR 92-50]. Department of industrial and operations engineering , the University of Michigan, 1992

13 Powell D, Skolnick M M . Using Genetic algorithm in engineering design optimization with nonlinear constraints. In: Proc. of the 5th Intl. conf. on Genetic algorithm, 1993. 424～430

14 Schoenauer M, Xanthakis S. Constrained GA optimization. In: Proc. of the 5th Intl. conf. on Genetic algorithm, 1993. 573～580

15 Cheng R, Gen M. Genetic algrithms for multi-row machine layout problem. Engineering Design and Automation, 1996(to appear)

16 Wright A H. Genetic algorithm for real parameter optimization, (in): [21]205-218, 1991

17 Michalewicz Z, et al. Evolutionary operations for continuous convex parameter spaces. In: Proc. of the third annual conf. on Evolutionary Programming, 1994. 84～97

18 Janilow C, Michalewicz Z. An experimental comparison of binary and floating point representations in genetic alogrithms. In: Proc. of the 4th Intl. conf. on Genetic algorithm, 1991. 31～36

19 Gen M, Liu B, Ida K. Evolution program for deterministic and stochastic optimizations. European Journal of Operational Research, 1996(in press)

20 Tamaki H, et al. A Comparison study of genetic coding for the traveling salesman problem. In: Proc. of the 1th IEEE Intl. conf. on Evolutionary computation(ICEC'94)

21 Rawline G J E. Foundations of Genetic Algorithm. Morgen Kaufmann, San Mateo, CA, 1991

（上接第95页）

·ECSVDP-DH (Elliptic Curve Secret Value Derivation Primitive, Diffie-Hellman version)：椭圆曲线 Diffie-Hellman 秘密值生成原语，用于实现 ECDH 算法的数学运算，即用对方的 ECDH 公钥和自己的 ECDH 私钥计算双方协商的秘密值。

·ECSP-DSA (Elliptic Curve Signature Primitive, DSA version)：椭圆曲线 DSA 签名原语，用于实现 ECDSA 签名的数学运算，即用自己的私钥和数据产生签名数据。

· ECVP-DSA (Elliptic Curve Verification Primitive, DSA version)：椭圆曲线 DSA 校验原语，用于实现 ECDSA 校验的数学运算，即用对方的公钥和数据对签名进行校验。

3.3.2 *模式* 在 WTLS 的椭圆曲线算法实现中需要的模式如下：

·ECKAS-DH (Elliptic Curve Key Agreement Scheme, Diffie-Hellman version)：椭圆曲线 Diffie-Hellman 密钥协商模式，用于完成 ECDH 算法在应用程序中的全部过程。该过程在我们的实现中定义如下：

1)建立和双方密钥相关的有效的椭圆曲线域参数。

2)选择有效的私钥 s。

3)取得对方的公钥 w'。

4)通过调用 ECSVDP-DH 原语，用自己的私钥 s 和对方的公钥 w' 计算出秘密值 z。

5)把秘密值 z 转化为字节串 Z。

6)双方从字节串 Z 产生密钥。

·ECSSA (Elliptic Curve Signature Scheme with Appendix)：代附加数据的椭圆曲线签名模式，其中包含实现应用程序中使用的 ECDSA 签名数据(c,d)产生操作和签名数据校验操作。该签名过程在我们的实现中定义如下：

1)选择有效的私钥 s 和有效的域参数。

2)把消息 M 编码为整数表示形式 f。

3)通过调用 ECSP-DSA 原语，从整数 f 和私钥 s 产生出签名数据(c, d)。

4)输出签名。

而 ECSSA 的校验过程定义如下：

1)取得对方的公钥 w' 和相关的域参数。

2)把消息 M 编码为整数表示形式 f。

3)通过调用 ECVP-DSA 原语，对签名进行校验。如果校验成功，输出“校验成功”，否则输出“校验失败”。

小结 本文介绍了椭圆曲线密码体制及其应用，并以 WTLS 为例探讨了该密码体制实现的具体措施。该方法已由我们在国内首先用软件设计实现，已在康佳 WAP 手机上调试成功，并在该系统上可靠稳定地运行六个月以上。

参 考 文 献

1 Institute of Electrical and Electronics Engineers. Standard Specification for Public Key Cryptography. IEEE P1363 working draft D13, March 2001

2 Institute of Electrical and Electronics Engineers. Standard Specifications for Public Key Cryptography: Additional Techniques. IEEE P1363A, working draft D5, Aug. 2000

3 Robshaw M J B, Yin Y L. Elliptic Curve Cryptosystems. RSA Laboratory. http://www. rsasecurity. com/rsalabs/ecc/elliptic_curve. html, June 27, 1997

4 WAP Forum. Wireless Transport Layer Security. 06-Apr-2001, URL: http://www. wapforum. org/

5 Dierks T, Allen C. The TLS Protocol. Jan. 1999. rfc2246