自适应搜索优化算法*)

周 晖[1,2] 徐 晨[1] 邵世煌[2] 李丹美[2]

(南通大学电子信息学院 南通 226019)[1] (东华大学信息科学与技术学院 上海 201620)[2]

摘 要 自由搜索算法是一种新的群集智能优化算法,已经成功地应用于函数优化问题。针对该算法所存在的对参数敏感等问题,提出自适应搜索算法。通过自适应实时调整搜索半径、搜索步、灵敏度等参数,提高算法对环境的适应性、鲁棒性和在"探索"和"开发"之间的平衡能力。对典型函数的试验结果证明,新算法不仅降低了对参数的依赖性,而且成功率高、收敛速度快,能有效避免陷入局部次优。

关键词 自由搜索(FS),自适应,群集智能,函数优化

Adaptive Free Search Algorithm

ZHOU Hui[1,2] XU Chen[1] SHAO Shi-huang[2] LI Dan-mei[2]

(School of Electronics and information, Nantong University, Nantong 226019, China)[1]

(College of Information Science and Technology, Donghua University, Shanghai 201620, China)[2]

Abstract Free search is a novel swarm intelligence algorithm. A new adaptive free search algorithm (AFS) was presented to solve the problem that the basic free search algorithm is sensitive to some parameters. The new algorithm (AFS), which is based on adaptively adjusting neighbour space and steps, sensitivity, can balance the global search and local search to improve AFS's convergence and roboutness. The experimental results show that the new algorithm not only solve the problem of the dependence on parameters but also has great advantage of convergence property over basic free search algorithm and particle swarm optimization, and can avoid the premature convergence problem effectively.

Keywords Free search (FS), Adaptability, Swarm intelligence, Function optimization

1 引言

群集智能(Swarm Intelligence)是计算智能领域的重要组成部分,作为一种新兴的演化计算技术已经成为越来越多研究者关注的焦点[1-3]。目前,群集智能的典型实现主要有两种:蚁群优化(Ant colony optimization, ACO)[4]和微粒群优化(Particle swarm optimization, PSO)[5]。

英国学者 K. Penev 和 G. Littlefair 基于"以不确定应对不确定,以无穷尽应对无穷尽"的思想,提出一种新的群集智能优化算法——自由搜索(Free Search, FS)[6]。该算法借鉴动物个体存在各异的嗅觉和机动性,提出了灵敏度和邻域搜索半径的概念,并利用蚂蚁释放信息素的机理,通过信息素和灵敏度的比较确定寻优目标,应用于函数优化,并显示出很好的性能[7]。目前对自由搜索算法的研究不多,挖掘其寻优潜能很有意义[8]。进一步研究发现,若参数设置不当,该算法收敛速度慢、收敛精度低,甚至易于陷入局部极值区。

针对上述问题,为了提高种群对环境的适应性,本文提出自适应搜索(Adaptive Free Search, AFS)算法,对基本 FS 算法做以下改进:实时调整搜索半径、搜索步和灵敏度等参数,精英保留,极值点判断与处理。新算法力图降低算法对设置参数的敏感性,提高其鲁棒性和搜索能力。

2 基本 FS 算法

FS 的算法模型中,个体在其搜索半径内随机产生 T 个坐标,找出其中最佳适应度坐标并且计算其信息素,通过信息素和灵敏度的比较,确定个体的新坐标。动物个体在其邻域内的行为描述如下:

$$\begin{cases} x_{tji}=x_{0ji}-\Delta x_{tji}+2\Delta x_{tji}\cdot random_{tji}(0,1) \\ \Delta x_{tji}=R_j\cdot(x_{i\max}-x_{i\min})\cdot random_{tji}(0,1) \end{cases} \tag{1}$$

这里,$random_{tji}(0,1)$是(0,1)内均匀分布的随机数;$x_{i\max}$和$x_{i\min}$是第 i 维变量的最大值和最小值;j 代表第 j 只个体($j=1,2,\cdots,m$);R_j 是第 j 只个体在搜索空间邻域的搜索半径($R_j\in[R_{\min},R_{\max}]$);$t$ 是搜索步(walk)中的当前小步($t=1,2,\cdots,T$);T 是搜索步数。

搜索过程中,对目标函数的符号做如下规定:$f_{tj}=f(x_{tji})$;$f_j=\max(f_{tj})$,这里 $f(x_{tji})$是一个个体完成搜索步后,信息素做标记位置的目标函数值。

信息素定义为

$$P_j=\frac{f_j}{\max(f_j)} \tag{2}$$

这里,$\max(f_j)$是搜索步内所有个体的当前最佳值。

灵敏度定义为

$$\begin{cases} S_j=S_{\min}+\Delta S_j \\ \Delta S_j=(S_{\max}-S_{\min})\cdot random_j(0,1) \end{cases} \tag{3}$$

这里 $S_{\min}$,$S_{\max}$是灵敏度的最小值和最大值,$random_j(0,1)$ 是均匀分布的随机数。

信息素的最小值和最大值分别为 $P_{\min}$,$P_{\max}$。规定:$S_{\min}=P_{\min}$,$S_{\max}=P_{\max}$。在一轮搜索结束后,确定个体 j 的新坐标,

*)国家 863 计划专项课题(2007AA01Z330),江苏省科技厅高技术研究项目(BG2007022),江苏省高校自然科学基金项目(07KJB510095)。

周 晖 博士研究生,副教授,主要研究方向为智能计算、计算机网络等;邵世煌 教授,博士生导师。

即新一轮搜索的起始点。

$$x'_{0ji}=\begin{cases}x_{0ji},(P_k<S_j)\\x_{ji},(P_k\geqslant S_j)\end{cases}\tag{4}$$

3 自适应搜索算法

3.1 搜索半径的自适应调节

邻域搜索半径 R_j 是反映个体 j 活动范围的参数，其大小决定了寻优的性质，对寻优性能的影响很大。而基本 FS 算法中 R_j 是固定的，若其取值较大，则搜索时间长、收敛精度低；若取值较小，则搜索范围小，容易出现早熟现象，因此算法对 R_j 非常敏感。

演化计算的关键问题之一，就是在“搜索”和“利用”之间建立平衡。本文提出实时调整搜索半径的策略，取初始值 $R_j(0)=1$，寻优过程中 $R_j(t)$ 递阶减小，变化规律如式(5)所示。

$$R_j(g)=\begin{cases}\rho_1 R_j(g-1),R_j(g)\geqslant R_{\min}\\R_{\min},\qquad\text{else}\end{cases}\tag{5}$$

式(5)中，g 是搜索代数；$R_{\min}$ 是最小搜索半径；ρ_1 是收缩系数；$1\geqslant\rho_1>0$，当 $\rho_1=1$ 时，即为基本 FS 算法。

3.2 搜索步 T 的实时控制

FS 算法中搜索步 T 是不变的，与搜索半径、目标函数值无关。这在初始阶段是可行的，但随着寻优搜索的发展，各个个体在不同区域得到的目标函数值不同，仍然保持 T 值不变会影响收敛速度。为了提高收敛效率，本文针对不同的个体采用不同的搜索步 T_j，T_j 随个体的目标函数不同而变化，规律如下：

$$T_j=INT\left[m\left(1+\frac{f_j(X)}{f_{\min}(X)}\right)\right]\tag{6}$$

3.3 极值区的判断处理

对于含有 n 个变量的函数最大值优化问题

$$J=\max(f(X))$$
$$X\in S=\{(x_1,x_2,\cdots,x_n)\in[x_{i\min},x_{i\max}]\}$$
$$i\in\{1,2,\cdots,n\}\tag{7}$$

式中，$f(X)$ 是适应值函数。设 $X^*=(x_1^*,x_2^*,\cdots,x_n^*)$ 是一个极大值点，它可以是全局的或局部的。从工程角度看，对于给定的误差 $\varepsilon>0$，如果对于所有 $i\in\{1,2,\cdots,n\}$，有 $|x_i-x*_i|<\varepsilon$，则称 X 已收敛到 X^*。当某一个体收敛到一个极值点时，应及时停止，同时记录该点，以防止因跳出极值区造成极值点丢失。

搜索过程中，当个体 j 在某点附近滞留 k 代，则可以认为该点是极值点，个体 j 陷入极值区。记录该点函数的适应值和坐标，并令个体 j 跳出该区域，重新初始化个体 j 的坐标点。

3.4 灵敏度调整

灵敏度是 FS 算法的重要参数。适当减小灵敏度，个体在邻域搜索的随机性增大，有助于提高搜索能力。对于跳出极值区的个体，将式(3)的灵敏度定义修改成

$$S_j=S_{\min}+\rho_2\Delta S_j;\ 0.99\geqslant\rho_2\geqslant0.95\tag{8}$$

3.5 精英保留

虽然随着群体的进化过程会产生越来越多的优良群体，但搜索过程的本质是随机的，它们有可能破坏当前群体中适应度最好的个体，影响算法的运行效率和收敛性。为此采用精英保留策略，具体操作过程是：

1)保留前群体中适应度最高的个体；

2)若当前最佳个体的适应度值大于迄今为止适应度最高的个体，则替换；

3)将式(2)信息素的定义修改成

$$P_j=\frac{f_j}{\max(f)}\tag{9}$$

式(9)中，$\max(f)$ 是迄今为止整个群体的最高适应度值。

3.6 算法设计和实现

自适应搜索算法的实现包括 3 部分：初始化、寻优搜索和终止判断，算法流程如图 1 所示。

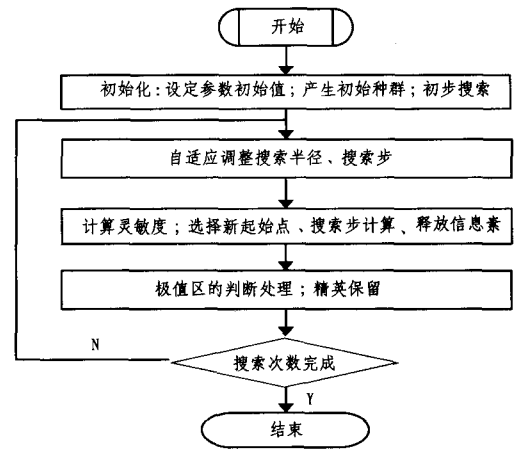

图 1 AFS 的算法流程

AFS 算法的实现描述如下。

Step1 初始化

1.1 设定：种群规模 m、搜索代数 G、搜索步长 $T(0)$、收缩系数 ρ_1、ρ_2 的初始值；

1.2 产生初始种群。按照随机方式产生初始种群；

1.3 初始化搜索。根据上述两步产生的初始值，生成初始信息素，释放初始信息素 $P_j\rightarrow X_k$，得到初始搜索结果 P_k，X_{kp}。

Step2 搜索过程

2.1 实时调整搜索半径、搜索步 T_j；

2.2 计算灵敏度。按照式(8)计算灵敏度 S_j；

2.3 确定起始点。选择新一轮搜索的起始点，$x'_{0j}=x_k(S_j,P_k)$；

2.4 搜索步计算。计算目标函数 $f_{tj}(x'_{0j}+\Delta x_t)$，这里 Δx_t 由式(5)计算；

2.5 释放信息素。按照式(7)计算信息素 P_j；按照式(9)释放信息素 $P_j\rightarrow x_k$，得到本次搜索结果；

2.6 极值区的判断与处理；

2.7 精英保留。

Step3 终止判断

判断终止条件：若不满足，则跳转至 Step2；若满足，则终止运行，输出结果。

4 实验研究

4.1 实验设计

为了验证本文提出的自适应搜索算法的性能，选取 4 个典型 Benchmark 函数，设计 3 种算法的测试实验：(1)粒子群(PSO)优化实验；(2)基本自由搜索(FS)优化实验；(3)自适应搜索(AFS)优化实验，并对它们的实验结果进行比较研究。典型函数的形式、维数(Dim)、搜索范围(Range)、理论极值(Optimal)和优化目标精度(Goal)见表 1。

选取的典型函数中，Sphere 函数是一个简单的单峰二次函数，全局极小值在 $x_i=0(i=1,2,\cdots,n)$；Rosenbrock 函数虽然是一个单模态函数，但它是一个经典复杂优化问题，它的全局最优点位于一个平滑、狭长的抛物线形山谷内，找到全局最小点的机会微乎其微，因此通常用来评价优化算法的执行效率；Griewank 函数的全局极小点 $x_i=0(i=1,2,\cdots,n)$，且有众多的局部极小点；Schaffer 函数是典型的非线性函数，具有广泛的搜索空间、大量的局部极小点和高大的障碍物，通常被认为是遗传算法很难处理的复杂多模态问题。

3 种算法的初始化和终止策略均设计为：

(1) 种群的初始化策略

采用随机方式产生初始种群，初始群体可以在搜索域的任意位置，如式(10)所示。

$$x_{oji}=x_{i\min}+(x_{i\max}-x_{i\min})\cdot random_{ji}(0,1) \tag{10}$$

(2) 终止策略

算法的终止策略采用固定目标函数收敛精度值，评估算法达到该精度目标所需的迭代次数、代数，即($f_{\max}\geqslant f_{opt}$) 或 ($g\geqslant G$)，其中 f_{opt} 为收敛精度值，G 为最大进化代数。

实验参数设置：3 种算法的种群数 m 和最大进化代数 G 相同：$m=30$，$G=1000$。其他参数如下：

(1)AFS 算法：$\rho_1=0.99$；$\rho_2=0.98$

(2)FS 算法：$T=m$；搜索半径；搜索半径

$$R_j=\begin{cases}1,(j=1,\cdots,\dfrac{m}{3})\\0.5,(j=\dfrac{m}{3}+1,\cdots,\dfrac{2m}{3})\\0.1,(j=\dfrac{2m}{3}+1,\cdots,m)\end{cases}$$

(3)PSO 算法

PSO 的参数选取依照文献[7]：加速常数 $c_1=c_2=1.7$；惯性权重 $w=0.6$。

表 1 用于实验的测试函数

Name and code	Formula	Dim	Range	Optimal	Goal
Sphere $f1$	$f_1(X)=\sum_{i-1}^{n}x_i^2$	10	$-100\leqslant x_i\leqslant 100$	0	10^{-5}
Rosenbrock $f2$	$f_2(\vec{x})=\sum_{i=1}^{n}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	10	$-30\leqslant x_i\leqslant 30$	0	10^{0}
Griewank $f3$	$f_3(\vec{x})=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	10	$-600\leqslant x_i\leqslant 600$	0	10^{-1}
Schaffer's $f6$	$f_6(\vec{x})=0.5-\frac{(\sin\sqrt{x_1^2+x_2^2})^2-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	2	$-100\leqslant x_i\leqslant 100$	1	10^{-5}

4.2 实验结果和分析

图 2－图 5 为分别用 AFS，FS，PSO 三种算法对上述 Benchmark 函数进行寻优的最佳适应度进化曲线。为了方便进化曲线的显示和观察，对 Sphere 函数、Rosenbrock 函数和 Griewank 函数的适应度值取对数，对 Schaffer 函数的进化代数取对数。

最佳适应度进化曲线是算法整个寻优搜索过程的客观反映。图 2－图 5 的实验结果显示，在种群数和精度要求相同的条件下，FS 算法的收敛速度快于 PSO 算法，而 AFS 算法的收敛速度明显快于其它两种算法。

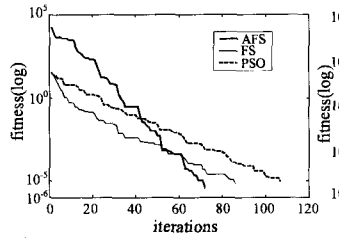

图 2 Sphere 函数的最佳适应度进化曲线

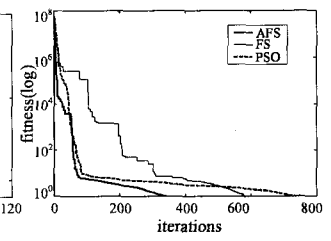

图 3 Rosenbrock 函数的最佳适应度进化曲线

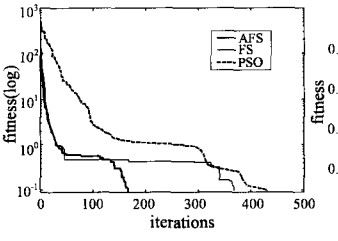

图 4 Griewank 函数的最佳适应度进化曲线

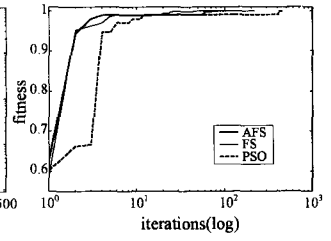

图 5 Schaffer 函数的最佳适应度进化曲线

对典型函数从不同的初始种群出发，采用上述 3 种算法分别进行 50 次独立实验。表 2 给出了在表 1 规定的函数维数、搜索范围、最小误差阈值等条件下运行各函数的平均进化代数(Mean iterations)、成功率(Success rate)和期望迭代次数(Expected iterations)。其中，

$$\text{成功率}=\frac{(\text{达到精度的运行次数})}{(\text{该项目的总运行次数})}$$

$$\text{期望迭代次数}=\frac{(\text{种群数})\times(\text{平均迭代数})}{(\text{成功率})}$$

期望迭代次数是对算法性能评价的主要依据。

表 2 三种实验结果的比较

Function	Mean iterations			Success rate			Expected iterations		
	AFS	FS	PSO	AFS	FS	PSO	AFS	FS	PSO
Sphere f_1	72	86	107	1	1	1	2160	2580	3210
Rosenbrock f_2	342	590	752	0.88	0.86	0.2	11659	20581	112800
Griewank f_3	131	253	287	0.94	0.76	0.30	4181	9987	28700
Schaffer's f_6	108	220	465	0.98	0.92	0.14	3306	7174	99643

从表 2 可见，虽然 FS 算法的收敛速度较之 PSO 提高得不多，但成功率提高很多，这是由 FS 算法本身的工作机理决定的。FS 算法中，个体具有各异的灵敏度和搜索半径，其本质是赋予个体更高的智能和运动特性，这也是 FS 算法和其他群集智能算法(如 PSO，ACO)最大的区别。而本文提出的自适应搜索算法在保持和有所提高 FS 算法成功率的基础上，收敛速度大大加快。这是因为新算法提高了种群对环境的自适应能力。实验结果表明，AFS 算法不仅消除了基本 FS 算法对参数的敏感性，而且在成功率和收敛速度两个方面优于 FS 算法，更是明显优于 PSO 算法。

结束语 本文在文献[5]的基础上提出了自适应搜索算法，其目的是提高算法对环境的适应性和鲁棒性，兼顾全局搜索和局部搜索。通过实时调整搜索半径、灵敏度，有效地解决了“探索”与“开发”之间的平衡；实时改变搜索步、精英保留策略，提高了算法的收敛速度；极值区的判断与处理，较好地避免了陷入局部极值区，并提高搜索效率。再者，新算法简化了参数设置，很多初始参数，如 $R_j(0)$，$T_j(0)$ 已经固定，毋需设

置，这对于该算法的推广应用很有益处。基于典型测试函数的实验结果验证了新算法的正确性和高效性。

另外，FS算法的研究刚刚开始，进一步研究其生物学背景、深入挖掘该算法的潜力、充分发挥其寻优潜能，同时和其他进化算法构成混合算法，以及新算法在工程中的应用等方面都是值得研究的课题。

参 考 文 献

[1] Denbya B. Swarm intelligence in optimisation problems. Nuclear Instruments and Methods in Physics Research, 2003, 502: 364-368

[2] Elbeltagia E, Hegazyb T, Griersonb D. Comparison among five evolutionary-based optimization algorithms[J]. Advanced Engineering Informatics, 2005, 19: 43-53

[3] Rafal K, Tomasz A, Kenneth D J. Evolutionary computation and structural design: A survey of the state-of-the-art. Computers and Structures, 2005, 83: 1943-1978

[4] Dorigo M, Blum C. Ant colony optimization theory: A survey [J]. Theoretical Computer Science, 2005, 344: 243-278

[5] Kennedy J, Eberhart R. Particle swarm optimization [A]// Proc. IEEE Int. Conf. on Neural Networks [C]. Perth, WA, Australia, 1995: 1942-1948

[6] Penev K. Adaptive computing in support of traffic management [J]. Adaptive Computing in Design and Manufacturing, 2004: 295-306

[7] 周晖，等. 一种新的群集智能算法——自由搜索[J]. 东华大学学报：自然科学版，2007，33(5)：579-583

[8] 周晖，等. 一种新的群集智能优化及其改进研究[J]. 系统工程与电子技术，2008，30(2)：337-340

[9] Trelea I C. The particle swarm optimization algorithm: convergence analysis and parameter selection [J]. Information Processing Letters, 2003, 85(6): 317-325

(上接第 139 页)

的结点得到项 g 和 h，S 串没有正的生成结点。

K：这个串发送密钥〈$+K$〉。串空间模型中限定密钥不会和任何加密的消息相同，所以 $\{MSARc_r\}_{K_{SA}}$ 不会在入侵者的 K 串生成。

E：串的形式是〈$-K, -h, +\{h\}_K$〉。在这里 $\{h\}_K = \{MSARc_r\}_{K_{SA}}$。根据自由加密假设，有 $h = \{MSARc_r\}$，$K = K_{SA}$。这意味着入侵者在第一个结点接收到密钥 K_{SA}。由于合法主体不会发送没有加密的秘密密钥，因此，$\{MSARc_r\}_{K_{SA}}$ 不会在 E 串生成。

D：串的形式是〈$-K^{-1}, -\{h\}_K, +h$〉。由于为正的项是由接收到的消息解密得到的，没有正结点作为生成结点，因此，$\{MSARc_r\}_{K_{SA}}$ 不会在 D 串生成。

由以上分析可知，$\{MSARc_r\}_{K_{SA}}$ 不会在入侵者结点生成，因此 $\{MSARc_r\}_{K_{SA}}$ 在丛 C 中的一个正常结点生成。

同理，可以证明 $\{MSARc_r\}_{K_{RA}}$ 也在丛 C 中的一个正常结点生成。

结合图 1，生成 $\{MSARc_r\}_{K_{SA}}$ 的正常结点只能是申请者结点，生成 $\{MSARc_r\}_{K_{RA}}$ 的正常结点只能是 RADIUS 服务器结点。因为项 $\{MSARc_r\}_{K_{SA}}$ 出现在认证者串的最后一个结点，根据丛 C 的属性，可知认证者串的丛高为 10。同样，$\{MSARc_r\}_{K_{RA}}$ 出现在 RADIUS 服务器串的最后一个结点，这使得 RADIUS 服务器串的丛高为 4。

4.2 认证者的一致属性

命题 2 设 Σ 是 802.11i 协议的串空间，C 是包含一个丛高为 10 的认证者串 $S_{auth} \in \mathrm{Auth}[M, S, A, R, c_i, q_j, f_k, K_{SA}, K_{RA}]$ 的丛，f_k, c_i 唯一生成于 Σ 中，$K_{SA} \notin K_p$，$K_{RA} \notin K_p$，那么 C 包含一个丛高至少为 5 的申请者串 $S_{sup} \in \mathrm{Sup}[M, S, A, R, c_i, q_j, f_k, K_{SA}]$ 和一个丛高为 4 的 RADIUS 服务器串 $S_{RADIUS} \in \mathrm{Radius}[M, S, A, R, f_k, c_i, K_{RA}]$。

证明：S_{auth} 的迹是〈$-\{MSA\}, +\{MSAq_1\}_{K_{SA}}, -\{MSAf_1\}_{K_{SA}}, +\{MSAf_1\}_{K_{RA}}, -\{MSARc_1\}_{K_{RA}}, +\{MSARc_1\}_{K_{SA}}, -\{MSARf_2\}_{K_{SA}}, +\{MSARf_2\}_{K_{RA}}, -\{MSARc_r\}_{K_{RA}}$〉。

采用命题 1 的分析方法，可以推断出 $\{MSARc_r\}_{K_{RA}}$ 在丛 C 中的一个正常结点生成，并且 $\{MSARc_r\}_{K_{RA}}$ 是在 RADIUS 服务器串的最后一个结点生成，从而 RADIUS 服务器串的丛高为 4。类似地，可以推断出项 $\{MSARf_2\}_{K_{SA}}$ 位于一个正常的申请者串上。从图 1 可以看出，$\{MSARf_2\}_{K_{SA}}$ 位于申请者串的第 5 个结点，因此申请者串的丛高是 5。这里不能保证申请者串的丛高为 6，因为存在入侵者丢弃合法主体消息的情况。

结束语 安全协议的形式化分析对于确保协议的正确性至关重要。本文利用串空间理论模型化了 IEEE802.11i 协议的认证过程。分析表明，802.11i 协议能够安全实现它的认证功能，并且分析过程简单直观。今后的工作包括在串空间模型中引入计算方法的观点[7,8]，使分析结果具有计算可靠性，以及形式化分析 802.11i 协议的其他安全属性。

参 考 文 献

[1] IEEE Std 802.11i™ Amendment 6: Medium Access Control (MAC) Security Enhancements[S], 2004

[2] Thayer F, Herzog J C, Guttman J D. Strand spaces: proving security protocols Correct [J]. Journal of Computer Security, 1999, 7(2): 191-230

[3] Yang Jie, Deng Huifang. Security electronic commerce protocol by the third kind entities[C]// 5th International Conference on Machine Learning and Cybernetics. IEEE Computer Society Press, 2006: 13-16

[4] 王继志，王英龙. 基于改进的串空间分析 Ad Hoc 路由协议安全性[J]. 软件学报，2006，17(11)：256-261

[5] Doghmi S F, Guttman J D, Thayer F. Sheletons, homomorphisms and shapes: characterizing protocol executions[J]. Notes Theor. Comput. Sci, 2007, 173: 85-102

[6] Lowe G. A hierarchy of Authentication Specifications[C]// 10th Computer Security Foundations Workshop Proceedings. IEEE Computer Society Press, 1997: 31-43

[7] Blanchet B. A computationally sound mechanized prove for security protocols[C]// IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 2006: 140-154

[8] Canetti R, Herzog J C. Universally composable symbolic analysis of mutual authentication and key exchange protocols[C]// 3th Theory of Cryptography Conference (TCC). Springer, 2006: 380-403