# 一种无线传感器网络的混沌 Hash 算法

黄锦旺 胡志辉 冯久超
(华南理工大学电子与信息学院 广州 510641)

**摘 要** 无线传感器网络是当前的一个研究热点,在军事、工业、地质监测和医疗健康等方面有重要的应用价值,这些应用经常包含敏感信息,因此无线传感器网络的信息安全很重要。由于无线传感器网络节点运算能力低、存储空间小且能量有限,在 PC 上使用的混沌 Hash 算法不能直接在无线传感器网络中使用,因此提出一种可以在无线传感器网络中使用的混沌 Hash 算法。理论分析和仿真结果表明,该方法与 PC 上使用的方法具有同等的安全性能。

**关键词** 无线传感器网络,混沌,Hash,安全
**中图法分类号** TN918 **文献标识码** A

## Chaos-based Hash Function for Wireless Sensor Networks

HUANG Jin-wang HU Zhi-hui FENG Jiu-chao
(School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, China)

**Abstract** Wireless sensor networks (WSNs) are one of hot research issues because of their applications in fields of military, industry, ecological and health care. These applications often also include to monitor sensitive information, and security is therefore important. WSNs are restricted from many constraints, including low computation capability, small memory, limited energy resources. Hash function ran on PC can not used in WSNs directly, so we proposed a chaos-based hash function that can be ran on WSNs. Theoretical analysis and simulation results show that it has the same security level as the hash that ran on PC.

**Keywords** WSNs, Chaos, Hash, Security

随着 WSNs (无线传感器网络)应用的迅速发展[1],WSNs 的安全问题得到广泛的关注[2,3]。由于 WSNs 节点的处理能力不足、存储空间小和能量有限等,传统的安全算法不能直接应用于 WSNs,迫切需要可以应用于 WSNs 的安全算法[4,5]。目前提出来的各种 Hash 函数都是基于 PC 的,如广泛应用的 MD5[6]和 SHA-1[7],还没有基于 WSNs 的 Hash 算法。本文在传统混沌 Hash 算法基础上,根据 WSNs 的实际资源受限条件,提出一种适用于 WSNs 的基于混沌的 Hash 算法。本文第 1 节提出整数化的混沌方程;第 2 节给出本文的混沌 Hash 算法;第 3 节对算法的资源占用和各项性能指标进行理论和仿真分析;最后是本文的总结。

## 1 混沌及其整数化

WSNs 的计算能力有限,一般都不能进行浮点运算,所以需要对混沌函数做变换,使之适用于整数的运算,同时又保证混沌的各种相关性能。

Logistic 映射[8,9]是经典的一维混沌系统,由下述方程描述:

$$x_{i+1}=\mu x_i(1-x_i) \tag{1}$$

式中,$x_i$ 的取值范围为 $0<x_i<1$,$\mu$ 是系统的控制变量;当 $3.57\leqslant\mu\leqslant4$ 时,系统为混沌状态。

本文采用文献[10]提出的整数化混沌方程,即:

$$\begin{aligned}y_{i+1}&=4\times2^{-n}\times y_i(2^n-y_i)=2^{-n+2}\times y_i(2^n-y_i)-1\\&=4\times y_i-2^{-n+2}\times y_i^2-1\end{aligned} \tag{2}$$

式(2)就是运行在 WSNs 节点上的混沌方程,变量的取值范围在节点处理器的字长范围之内,系数 $2^{-n+2}$ 的乘法可以通过左移 $-n+2$ 位操作获得,这将极大地减少运算量。

## 2 算法描述

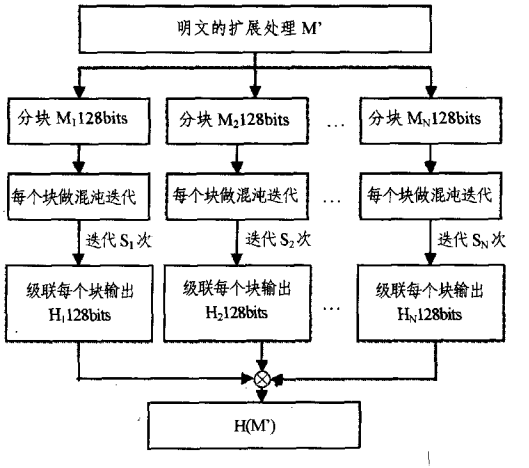


图 1 算法流程图

到稿日期:2012-09-06 返修日期:2012-12-10 本文受国家自然科学基金(60872123),国家-广东省自然科学基金联合基金(U0835001)资助。
黄锦旺(1984-),男,博士生,主要研究方向为电路与系统,E-mail:hgjw2008@163.com;冯久超(1964-),男,教授,博士生导师,主要研究方向为数字信号处理、数字通信、非线性动力学及混沌理论与应用,E-mail:fengjc@scut.edu.cn(通信作者)。

Hash 函数能将任意长度的明文信息压缩映射到固定长度的 Hash 值。设本文 Hash 函数产生的 Hash 值长度为 128bits，根据混沌 Hash 方程的一般结构[11]，算法包括以下几个步骤：明文的位数扩展、明文的处理、生成 Hash 值，其流程如图 1 所示。

### 2.1 明文扩展

明文消息是任意长度的 $n$bits，首先将明文扩展成一个 $N$ 行、14×8bits(14 个字节)列的矩阵，预留最后的 8 个字节用来存放明文消息的长度 $n$，填充消息长度可以防止一类选择消息攻击[13]。参考 MD5 算法的填充，在原始明文后面填充 $m$bits ($0 \leqslant m < 128$)，$m$bits 是一个 1 和后续的 $m-1$ 个 0 组成的 bit 串，形如“1000…0”，最后的 64 bits 是原始明文的长度 $n$。行数为 $N = floor[(n+64)/(14*8)]+1$，这里 $floor[*]$ 表示取下限的整数，填充的长度为 $m = 14*8*N-n-64$。将扩展明文的每一行的元素分别进行求和，并对求和结果用 2^8 取模，得到一列数据如下：

$$m_{i,15} = (\sum_{j=1}^{14} m_{i,j}) \bmod 2^8 \tag{3}$$

将扩展明文的每一行的元素分别进行异或操作，得到另外一列数据如下：

$$m_{i,16} = \bigotimes_{j=1}^{14} m_{i,j} \tag{4}$$

再将上面获得的 2 列分别列为矩阵的第 15 列和第 16 列。

下面对矩阵的列进行类似的操作，对每一列分别进行求和，将结果用 2^8 取模，得到一行数据：

$$m_{N+1,j} = (\sum_{i=1}^{N} m_{i,j}) \bmod 2^8, 1 \leqslant j \leqslant 16 \tag{5}$$

将矩阵的每一列进行异或操作，得到另外一行数据：

$$m_{N+2,j} = \bigotimes_{i=1}^{N} m_{i,j}, 1 \leqslant j \leqslant 16 \tag{6}$$

最后将获得的 2 行分别并到矩阵的最后作为矩阵的第 $N+1$行和第 $N+2$ 行，即：

$$M = \begin{bmatrix} m_{1,1} & \cdots & m_{1,14} & m_{1,15} & m_{1,16} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{N,1} & \cdots & m_{N,14} & m_{N,15} & m_{N,16} \\ m_{N+1,1} & \cdots & m_{N+1,14} & m_{N+1,15} & m_{N+1,16} \\ m_{N+2,1} & \cdots & m_{N+2,14} & m_{N+2,15} & m_{N+2,16} \end{bmatrix} \tag{7}$$

### 2.2 明文的分块处理

扩展之后的明文为 $M'$，是一个 $(N+2)\times 16$ 字节的矩阵，将每一行作为一组进行处理(处理方法见 2.3 节)，处理完后得到矩阵 $M'$，每一行有 16 个字节，也就是 128bits，记为 $H_i$ ($1 \leqslant i \leqslant N+2$)，将这 $N+2$ 个向量异或得到最终的 Hash 值，即：

$$H = \bigotimes_{i=1}^{N+2} H_i, 1 \leqslant i \leqslant N+2 \tag{8}$$

### 2.3 明文的分组处理

以 $M_i$ 行为例，说明每一行的处理过程。首先将该行里面的 16 个字节求和，结果对 256($2^8=256$)求模运算，运算结果记为 $S_i$，将 $S_i$ 作为该组数据的混沌迭代次数。

$$S_i = (\sum_{j=1}^{16} m_{i,j}) \bmod (2^8) \tag{9}$$

该组中的每一个块作为初始值代入式(7)迭代 $S_i$ 次，得到 $M_i'(m_{i1}', m_{i2}', \cdots, m_{i16}')$；将 $M_i'$ 里面的字节级顺序联成一个 128bits 的值 $H_i$，即

$$H_i = m_{i1}' m_{i2}' \cdots m_{i16}' \tag{10}$$

上述 Hash 算法的特点是：(1)整个运算过程简单，只有移位、异或、模求和等运算，这些都是处理器本身汇编指令具有的操作，可以在一个机器指令周期之内完成，降低了处理器的计算量并提高了效率，适合在资源受限的 WSNs 节点中使用。(2)加入混沌函数后，明文发生微小变化，这会使得迭代之后的数据产生重大的改变，从而影响整个 Hash 结果，提高 Hash 函数的混乱和扩散能力。(3)混沌迭代的次数是可变的，每一个分组的混沌迭代次数均由该组内的数据决定，使得 Hash 的逆过程变得更为困难，同时也可以增加对明文的置乱能力。

## 3 性能分析

### 3.1 算法资源需求分析

由式(7)可知，$4\times y_i = y_i << 2$，将 $y_i$ 左移 2 位，记作一次移位操作 $s$；$2^{-n+2}\times y_i^2 = y_i^2 >> 2^{n-2}$，将 $y_i^2$ 右移 $(n-2)$位，记作一次乘法操作 $m$ 和一次移位操作 $s$，最后总体进行 2 次加法操作 $a$(减法可以看作是补码的加法)。进行一次混沌迭代的计算量为：

$$P_c = s+m+s+2a = m+2s+2a \tag{11}$$

每一个分组内有 16 个元素，每一个元素进行了 $S_i$ 次的混沌迭代，一共进行了 $16*S_i$ 次混沌迭代运算，一个组的计算量为：

$$P_g = 16\times S_i \times P_c \tag{12}$$

设明文长度为 $n$，一共有 $n_g$ 个组：

$$n_g = floor((n+8)/14)+3 \tag{13}$$

算法执行的运算量：

$$P_{exe} = P_g \times n_g \tag{14}$$

扩展后的矩阵是一个 $n_g\times 16$ 的矩阵，列扩展的运算量：

$$P_1 = 14\times(n_g+1)\times a + 14\times(n_g+1)\times \text{xor} \tag{15}$$

其中 xor 表示一次异或操作。

行扩展的运算量：

$$P_2 = 16\times(n_g+1)\times a + 16\times(n_g+1)\times \text{xor} \tag{16}$$

初始化的计算量：

$$\begin{aligned} P_{init} &= P_1 + P_2 \\ &= 14\times(n_g+1)\times a + 14\times(n_g+1)\times \text{xor} + 16\times(n_g+1)\times a + 16\times(n_g+1)\times \text{xor} \\ &= 30\times(n_g+1)\times a + 30\times(n_g+1)\times \text{xor} \end{aligned} \tag{17}$$

该算法初始化和执行的计算量与明文长度呈线性关系。

算法的 SRAM 空间占用主要是明文扩展后的矩阵空间占用，算法的操作结果可以存回这个矩阵；算法操作的中间变量都是使用处理器的寄存器，不占用 SRAM，所以 SRAM 的占用量为：

$$S_{ram} = 16\times n_g \tag{18}$$

### 3.2 Hash 值分布

Hash 值的均匀分布是评价一个散列函数最重要的性能之一，它将直接影响散列函数的安全性。下面对本文的英文摘要进行 Hash 处理，经本文算法处理后的 Hash 值为：DF05001A9A148FA7B6B27ECFA04D6E12。

使用二维图形来形象地展示消息明文与 Hash 值之间的区别：图 2 左边中消息明文的 ASCII 码值比较集中地分布在一个较小的范围内(32，98～115)；图 2 右边十六进制 Hash 值则呈均匀随机的分散状态。可见，经分组和混沌迭代之后，Hash 值已经不包含明文消息的统计概率信息。

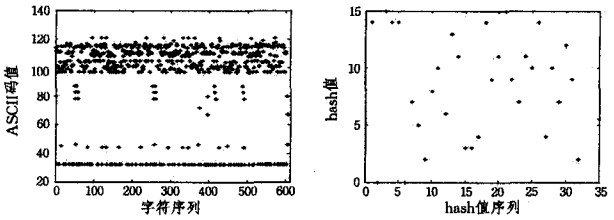


图 2　明文及 Hash 值的分布

### 3.3　Hash 值对消息明文的敏感性分析

测试本文的 Hash 算法对消息明文的敏感程度，分别在以下 6 种条件下对文本的 Hash 值进行仿真：

条件 1　初始文本 1 为本文的英文摘要；

条件 2　初始文本 2 将文本 1 的首字母 P 改为 p；

条件 3　初始文本 3 将文本 1 的首字母 P 改为 Q；

条件 4　初始文本 4 将文本 1 的首字母删除；

条件 5　初始文本 5 将文本 1 末尾的句号改成逗号；

条件 6　初始文本 6 在文本 1 开始处加一个空格。

仿真得到的 Hash 结果用 16 进制数表示如下：

条件 1　DF05001A9A148FA7B6B27ECFA04D6E12；

条件 2　5A28F7A4F71B20116F86F1F43BBA633A；

条件 3　0B13A0F24D846FE07475A6593FBDE920；

条件 4　D366AA1FD6BC5D6E38B997EB771BE0C2；

条件 5　490A4B8CF66472593D49EC8FD3CA04AD；

条件 6　0F55F8226D729636ADCA637DEAC01B4F。

用 0,1 序列的图形化表示，如图 3 所示。

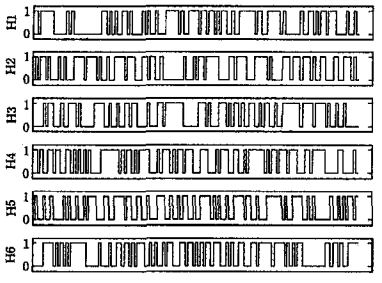


图 3　明文敏感度测试图

从图 3 可以看出，明文的一个微小变化，都会导致 Hash 值的巨大变化，说明本文的 Hash 算法具有对原始明文的敏感性，同时由于混沌和算法的不可逆性，本文的 Hash 具有良好的单向性。

### 3.4　混乱与扩散性质统计分析

混乱和扩散是加密算法的两个性能指标[12]，目的是每一位明文的变化能够影响到尽可能多的 Hash 值，让明文和密文有更复杂的关系，从而掩盖明文的概率特性。理想情况下的扩散效果是初值的细微变化能导致结果的每比特都以 50% 的概率变化。定义如下 4 个统计量作为算法性能的指标：

$$\overline{B}=\frac{1}{N}\sum_{i=1}^{N}B_i \tag{19}$$

$$\Delta B=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i-\overline{B})^2} \tag{20}$$

$$P=(\overline{B}/128)\times 100\% \tag{21}$$

$$\Delta P=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(B_i/128-P)^2}\times 100\% \tag{22}$$

式中，$N$ 为统计的次数，$B_i$ 为第 $i$ 次试验结果的变化比特数，$P$ 为平均变化概率，$\overline{B}$ 为平均变化比特数，$\Delta B$ 为 $B$ 的均方差，$\Delta P$ 为 $P$ 的均方差。测试方法如下：每次测试时，在消息明文空间中随机地选取一段明文求其 Hash 值，然后改变消息明文 1bit 的值得到另一个 Hash 结果，比较 2 个 Hash 结果，求出变化的比特数 $B_i$。经 256,512,1024,2048 次测试，得到基于该算法明文 1bit 变化下的 $B,P,\Delta B,\Delta P$，如表 1 所列。

表 1　混乱和扩散性能表

| 统计量 | N | | | | |
|---|---|---|---|---|---|
| | 256 | 512 | 1024 | 2048 | 均值 |
| B | 63.63 | 63.80 | 63.38 | 63.65 | 63.61 |
| ΔB | 5.615 | 5.65 | 5.65 | 5.61 | 5.63 |
| P | 49.71 | 49.85 | 49.86 | 49.72 | 49.79 |
| ΔP | 4.387 | 4.41 | 4.41 | 4.38 | 4.40 |

从表 1 可以看出，明文变化 1bit 时，算法的平均变化比特数和每比特平均变化概率都接近理想状况下的 64bit 和 50%；$\Delta B,\Delta P$ 标志着 Hash 混乱与扩散性质的稳定性，值越小就越稳定，实验所得的 Δ 值都较小，说明本文算法对明文的混乱与扩散能力强，而且稳定。

### 3.5　抗碰撞分析

抗碰撞性能是指找到具有相同 Hash 值的两个不同明文在计算上是不可行的。本文用以下方法测试算法的抗碰撞性能：在消息明文空间中随机地选取一段明文求出 Hash 值并以 ASCII 码形式存储，然后随机地选择并改变消息明文中 1 bit 的值得到另一 Hash 值，同样以 ASCII 码形式存储，比较两个 Hash 值，若两个值中相同位置上的字符相同，则称为被击中一次，统计被击中的次数。在 2048 次的测试中，有 84 次击中一次，有 12 次击中两次，剩下的 1952 次没有击中发生，本文的算法有较好的抗碰撞性能。

另外一种测试抗碰撞性能的方法是测试两个 Hash 值的绝对差异度，用下面的公式表示：

$$d=\sum_{i=1}^{N}|t(e_i)-t(e_i')| \tag{23}$$

式中，$e_i$ 和 $e_i'$ 分别是原始明文消息 Hash 值和修改后的明文消息 Hash 值第 $i$ 个位置的 ASCII 码，函数 $t(*)$ 将 ASCII 码转换成对应的十进制数值。在 2048 次这样的实验后，其结果如表 2 所列，两个 Hash 值的绝对差异度值平均值是 81.45，比较接近理论值 85.333。

表 2　Hash 值的差异度

| 最大值 | 最小值 | 平均值 | 平均值/字符 |
|---|---|---|---|
| 1907 | 620 | 1303.1 | 81.45 |

**结束语**　本文提出了一种适合在 WSNs 中使用的混沌 Hash 算法，它首先对明文进行一个扩展和填充，通过对行和列的扩展，把明文的变化进一步扩展出来，将扩展的明文进行分组和混沌迭代，利用混沌的初值敏感性和置乱特性，使得 Hash 算法有更好的混乱和扩散的性能。在算法设计中，考虑到了 WSNs 的节点计算能力和存储空间有限，针对节点处理器的特点对算法做了优化，尽量使用处理器硬件具有的指令功能，以降低处理时间，提高算法的效率。理论分析和仿真表明本文的算法资源占用小。

## 参 考 文 献

[1]　Estrin D, et al. Instrumenting the World with Wireless Sensor Networks[C]// Proc. Int'l. Conf. Acoustics, Speech and Signal Processing. Salt Lake City, UT, May 2001

（下转第 89 页）

**结束语** 上下文信息的有效利用可以使计算服务更具人性化和智能化。传统的使用上下文信息的应用一般都是针对个体的或者少数群体的上下文信息，本文面向大规模的上下文态势展开了研究，基于端和云相结合的模式，提出了面向大规模上下文信息获取的统一抽象模型及基于 MapReduce 和规则的云端上下文聚合算法，进而设计并实现了一个大规模上下文管理框架。在未来工作中，我们将对云端大规模上下文聚合的性能，以及移动终端收集和发送上下文信息时产生的能耗问题做进一步的研究。

## 参 考 文 献


[1] Schilit B, Adams N, Want R. Context-Aware Computing Applications[C]// Proceedings of Workshop on Mobile Computing Systems and Applications. 1994

[2] Choudhury T, et al. The Mobile Sensing Platform: An Embedded System for Activity Recognition[J]. IEEE Pervasive Comp, 2008, 7(2): 32-41

[3] Lane N D, Miluzzo E, Lu Hong, et al. A Survey of Mobile Phone Sensing[J]. Communications Magazine, IEEE, 2010, 48(9): 140-150

[4] MapReduce[OL]. http://en. wikipedia. org/wiki/MapReduce, 2011

[5] Dartmouth College. Mobile Sensing Group[OL]. http://sensorlab. cs. dartmouth. edu/

[6] Mun M, et al. Peir, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research[C]// Proc. 7th ACM MobiSys. 2009: 55-68

[7] Thiagarajan A, et al. VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones[C]// Proc. 7th ACM SenSys. Berkeley, CA, Nov. 2009

[8] UC Berkeley/Nokia/NAVTEQ. Mobile Millennium [OL]. http://traffic. berkeley. edu/

[9] witterStorm[OL]. http://www. infoq. com/news/2011/09/twitter-storm-real-time-hadoop, 2011

[10] Ding Bo, Wang Huai-min, Shi Dian-xi. Pervasive middleware technology[J]. 计算机科学与探索, 2007(3)

[11] Yau S, Karim F, Wang Y, et al. Reconfigurable context-sensitive middleware for pervasive computing[J]. Pervasive Computing, 2002, 1(3): 33-40

[12] Da T, Zhang Q. A middleware for building context- aware mobile services [C]// Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th, 2004, 5: 2656-2660

[13] Korpipaa P, Koskinen M, Peltola J, et al. Bayesian approach to sensor-based context awareness [J]. Personal and Ubiquitous Computing, 2003, 7(2): 113-124

[14] Schmidt A, Aidoo K A, Takaluoma A, et al. Advanced Interaction in Context [C]// Handheld and Ubiquitous Computing. Springer Berlin Heidelberg, 1999: 89-101

[15] Castro P, Munz R. Managing context data for smart spaces[J]. Personal Communications, 2000, 7(5): 44-46


(上接第 51 页)


[2] Chan H, Perrig A. Security and Privacy in Sensor Networks[J]. IEEE Comp. , 2003, 36(10): 103-105

[3] Shi E, Perrig A. Designing Secure Sensor Networks[J]. Wireless Commun, 2004, 11(6): 38-43

[4] Gura N, et al. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs [C]// CHES'04: Proc. Wksp. Cryptographic Hardware and Embedded Systems. Aug. 2004

[5] Gaubatz G, Kaps J-P, Sunar B. Public Key Cryptogra-phy in Sensor Networks-Revisited[C]// ESAS'04: 1st European Wksp. Security in Ad-Hoc and Sensor Networks. 2004

[6] Rivest R L. RFC 1321. The MD5 message digest algorithm request for comments [S]. Cambridge: MIT and RSA Data Security, 1992

[7] Nits F P. FIPS PUB 180, Secure hash standard [S]. Washington DC: U. S. Department of Commerce, 1993

[8] Wang S Y, Feng J C. Chaotic communication scheme by parameter division multiple access[J]. Acta Electronica Sinica, 2007, 35(7): 35-40

[9] 刘式达，梁福明，刘式适，等. 自然科学中的混沌和分形[M]. 北京：北京大学出版社，2003：1-12，26，31-32，125

[10] 陈帅. 无线微传感器网络混沌加密理论及其关键技术研究[D]. 重庆：重庆大学，2006：55-78

[11] Peng Fei, Qiu Shui-sheng. One-way Hash Functions Based on Iterated Chaotic Systems[C]// Communications, Circuits and Systems International Conference. 2007: 1070-1074

[12] Shannon C E. Communication theory of secrecy systems[J]. Bell System Technology Journal, 1949, 28: 656-715

[13] 郑世慧，张国艳，杨义先，等. 基于混沌的带密钥散列函数安全分析[J]. 通信学报，2011，32(5)：146-152


(上接第 62 页)


[10] Shih H-S, Shyur H-J, Lee E S. An extension of TOPSIS for group decision making[J]. Mathematical and Computer Modeling, 2007, 45(7/8): 801-813

[11] Jahanshahloo G R, Lotfi F H, Davoodi A R. Extension of TOPSIS for decision-making problems with interval data[J]. Interval efficiency, Mathematical and Computer Modeling, 2009, 49(5/6): 1137-1142

[12] Ai Li-feng, Tang Mao-lin, Fidge C. Partitioning composite Web services for decentralized execution using a genetic algorithm [J]. Future Generation Computer Systems, 2011, 27(2): 157-172

[13] Zhao Xin-chao, Song Bo-qian, Huang Pan-yu, et al. An improved discrete immune optimization algorithm based on PSO for QoS-driven webservice composition [J]. Applied Soft Computing, 2012, 12(8): 2208-2216

[14] Fei Tao, Zhao Dong-ming, Hu Ye-fa, et al. Correlation-aware resource service composition and optimal-selection in manufacturing grid[J]. European Journal of Operational Research, 2010, 201(1): 129-143

[15] Ma Yue, Zhang Cheng-wen. Quick convergence of genetic algorithm for QoS-driven web service selection[J]. Computer Networks, 2008, 52(5): 1093-1104