基于遗传策略的格基约化算法

刘向辉*①② 韩文报①② 权建校③
①(解放军信息工程大学 郑州 450002)
②(数学工程与先进计算国家重点实验室 郑州 450002)
③(江南计算技术研究所 无锡 214083)

摘 要：格基约化算法是密码分析的重要工具。该文借鉴遗传算法的基本策略，通过对初始格基的调整变换，提出了一种新的格基约化算法，新算法总能得到给定格中长度更短的向量和质量更高的一组基。利用该算法，针对最短向量问题(SVP)挑战的部分数据进行了测试，新算法的输出结果达到或超过了挑战的公开记录，约化效果良好。

关键词：密码学；格基约化；LLL 算法；遗传算法；最短向量问题挑战
中图分类号：TN918 **文献标识码**： A **文章编号**：1009-5896(2013)08-1940-06
DOI: 10.3724/SP.J.1146.2012.01560

A New Lattice Reduction Algorithm Based on Genetic Strategy

Liu Xiang-hui①② Han Wen-bao①② Quan Jian-xiao③
①(*PLA Information Engineering University*, *Zhengzhou* 450002, *China*)
②(*State Key Laboratory of Mathematical Engineering and Advanced Computing*, *Zhengzhou* 450002, *China*)
③(*Jiangnan Institute of Technology*, *Wuxi* 214083, *China*)

Abstract: Lattice reduction algorithms play an important role in the field of cryptanalysis. In this paper, based on the strategies of genetic algorithm, a new lattice reduction algorithm is proposed through the transformation of the initial lattice basis. The new algorithm always can obtain a shorter vector and a higher quality basis compared with the original algorithms. By the new algorithm, some lattice bases of the Shortest Vector Problem (SVP) challenge are experienced and the outputs of the new algorithm can always reach or break the records on the internet which illustrates that the new algorithm behaves well.

Key words: Cryptography; Lattice reduction; LLL algorithm; Genetic algorithm; Shortest Vector Problem (SVP) challenge

1 引言

自 20 世纪 80 年代以来，由于在密码分析中显现出的巨大威力，格基约化成为密码学领域一个新的研究热点。1982 年 Lenstra, Lenstra 和 Lovasz 提出了著名的 LLL 算法[1]，能在多项式时间内得到一个相对较短的向量，一经提出就成为最为出名的格基约化算法。随后，基于背包问题的公钥密码体制在多项式时间被 LLL 算法攻破。1996 年，Coppersmith[2]提出了利用 LLL 算法求解整系数同余方程的方法，此后该方法被广泛应用于对 RSA 体制的小指数攻击、部分密钥泄漏攻击等领域中。目前，关于格基约化在密码分析中的应用研究层出不穷[3,4]。

LLL 算法能够在多项式时间内找到一个长度不超过格中最短向量长度 $2^{(n-1)/2}$ 倍(称为近似因子)的向量。Schnorr[5]结合枚举算法将 LLL 算法进行了一般化，提出了可以得到更短向量的 BKZ 算法。BKZ 算法的复杂度和分块规模(参数 k)相关，目前为止还不确定它是否是多项式时间算法[6]。BKZ 算法的理论近似因子为 $k^{(n-1)/k}$，当 k=2 时，它和 LLL 算法是等价的。由于 BKZ 算法具有更好的约化效果，因此在实际中也具有更为广泛的应用。为了促进格基约化算法的有效实施，德国 Darmstadt 大学给出了一系列 SVP 挑战，挑战者可以上传给定格中长度更短的向量或者新格中满足一定长度的短向量。目前排名前三的挑战记录是由 Schneider 完成的，他采取的方法是 BKZ 算法和枚举算法相结合[7]。

LLL算法和BKZ算法都是从一组给定的基出发，对其进行不断变换的过程。实际上，初始基的选择对格基约化算法的效率和输出基的质量有很大

2012-11-30 收到，2013-03-29 改回
国家自然科学基金(61003291)资助课题
*通信作者：刘向辉 lxhkz2002@163.com

影响。Backes通过实验方法考察了输入格基的降序和升序排列对LLL算法运行效率的影响，如果格基采用合适的序，那么就可以有效降低LLL算法的运行时间[8]。而在实际的测试中也表明，输入的格基不同，输出向量的长度也不尽相同。遗传算法提供了一种求解系统优化问题的通用框架，是实际中经常使用的一种优化算法。遗传算法从问题的多个解开始搜索，而不是从单个解开始，给格基约化算法提供了一种新的思路：对多组基进行约化，找到其中最好的一组基，从而摆脱了传统格基约化算法对一组基求解的局限。

本文借鉴遗传算法的基本策略，从全新的角度提出了一种格基约化算法，并利用该算法对SVP挑战数据进行了测试。论文具体章节安排如下：第2节介绍基本知识，第3节基于遗传策略提出新的格基约化算法，第4节给出SVP挑战的实验结果及分析，最后是结束语。

2　准备知识

本节对格基约化及遗传算法的基本概念和理论进行介绍，具体细节可参考文献[1,9]。

定义 1　设 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_m\in\boldsymbol{R}^n$ 为一组线性无关的向量，其中 $\boldsymbol{R}^n$ 表示实数域上的 n 维向量空间。集合 $\boldsymbol{L}(\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_m)=\left\{\boldsymbol{z}=\sum_{i=1}^{m}\lambda_i\boldsymbol{b}_i|\lambda_i\in\boldsymbol{Z}\right\}$ 称为以 $(\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_m)$ 为基的格，其中 m 称为格的维数。若 $m=n$，则格 L 称为满秩的。如果 $\boldsymbol{R}=\boldsymbol{Z}$，也即格中元素取自整数环，则称其为整格。

一个格可以用不同的基来表示。在解决格上相关问题时，希望能够找到满足一定条件的基，以有利于解决相应的问题，选择这样一组基的过程就称为格基约化，并且称这组基为格的一组约化基。由于具体问题的需要不同，约化基的标准也有所不同，例如，LLL 基就是一类重要的约化基。对于 n 维向量 $\boldsymbol{b}$，以下用 $\|\boldsymbol{b}\|$ 表示其欧氏范数，也即向量的长度。

定义 2　设 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_m\in\boldsymbol{R}^n$ 是格 L 一组基，如果满足：

(1) $|\mu_{ij}|\le 1/2, 1\le j<i\le n$；

(2) $\delta\left\|\boldsymbol{b}_{k-1}^*\right\|^2\le\left\|\boldsymbol{b}_k^*\right\|^2+\mu_{k,k-1}^2\left\|\boldsymbol{b}_{k-1}^*\right\|^2, k=2,3,\cdots,n, 1/4<\delta\le 1$。

则称这组有序的基 $(\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n)$ 是以 δ 为参数的 LLL 约化基，其中 $\boldsymbol{b}_k^*$ 是对应的 Gram-Schmidt 正交基，μ_{ij} 为对应的正交化系数。

3　基于遗传策略的格基约化算法

传统格基约化算法从一组基出发寻找最短向量，而遗传算法根据多个候选解进行搜索，并根据适应度进行遗传操作，本节借鉴遗传算法的基本策略，设计新型的格基约化算法。需要注意的是，本文所提出的是一个通用框架，适用于 LLL 算法、BKZ 算法等格基约化算法，这里选取 LLL 算法作为代表算法进行描述，在实际使用时可根据需求选择具体的算法。

遗传算法是借鉴生物界的进化规律演化而来的随机化搜索方法，由美国的Holland教授于1975年提出。遗传算法首先需要产生一组候选解(初始种群)，然后依据专门的适应性标准(适应度函数)对种群个体进行评估，此后用倾向于较适应个体的准则，选择种群的子集进行某些操作(交叉、变异)，生成新的候选解并进入下一次遗传。根据这些遗传算法的基本特征，本文选取LLL运算作为遗传算法的基本操作，提出如下格基约化算法框架。

算法 1　基于遗传策略的 LLL 算法框架

输入：格 L 的一组基 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n\in\boldsymbol{Z}^n$、约化参数 δ $(1/4<\delta\le 1)$

输出：以 δ 为参数的LLL约化基

(1) LLL 约化：$\mathrm{LLL}(\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n,\delta)$；

(2)选取初始种群，对 $(\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n)$ 进行置乱，产生 m 组基；

(3)对每组基进行 LLL 约化，产生 m 个适应性评估结果；

(4)根据评估结果，对 m 组基进行遗传算子操作，产生 m 个子代个体；

(5)根据终止条件决定是否终止算法；

(6)输出一组基。

图 1 给出了基于遗传策略的 LLL 算法框架基本流程。下面分别从初始种群的选取、适应度函数、遗传算子和终止条件 4 个方面对算法框架的基本流程进行设计。

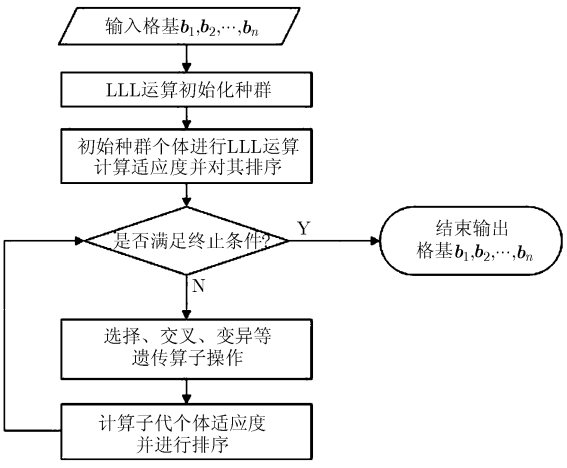

图 1 基于遗传策略的 LLL 算法框架

3.1 初始种群的选取

初始种群的选取是遗传算法的基础。对于格基约化算法而言，初始格基只有一个，也即初始解只有一个，我们需要对其进行变换来生成初始种群。设 $\boldsymbol{B}$ 为格 L 的一组基，如果 $\boldsymbol{U}$ 为 n 维的幺模矩阵，也即 $\det(\boldsymbol{U})=1$，那么 $\boldsymbol{B}^*=\boldsymbol{UB}$ 也为格 L 的一组基。于是，对初始格基进行变换也即对其左乘幺模矩阵的过程。

假设初始种群的规模为 m，那么需要随机产生 m 个 n 维幺模矩阵。令

$$\boldsymbol{A}_i=\begin{bmatrix}1 & 0 & \cdots & 0\\ a_{21} & 1 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ a_{n1} & a_{n2} & \cdots & 1\end{bmatrix},\ \boldsymbol{C}_i=\begin{bmatrix}1 & c_{12} & \cdots & c_{1n}\\ 0 & 1 & \cdots & c_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & 1\end{bmatrix} \tag{1}$$

$\boldsymbol{A}_i$，$\boldsymbol{C}_i$ 中的未知元素为随机生成的 0 或者 1，$1\le i\le m$。也即幺模矩阵的选取采用 $\boldsymbol{U}_i=\boldsymbol{A}_i\boldsymbol{C}_i$ 的形式，其中 $\boldsymbol{A}_i$ 为下三角幺模矩阵，$\boldsymbol{C}_i$ 为上三角幺模矩阵，根据 $\boldsymbol{B}_i=\boldsymbol{U}_i\boldsymbol{B}$ 就产生了 m 组基，这也就构成了初始种群。

注释 1 变换矩阵采用下三角幺模矩阵和上三角矩阵幺模乘积的形式，一方面省去了判断矩阵是否为幺模矩阵的过程，另一方面能够达到对原始格基充分置乱的效果。实际上，变换矩阵的选取有多种方法，最简单的做法是产生初等矩阵来进行变换。但是，初等变换往往只是改变初始基中的某些位置或者某些位置上的值，达不到混乱的效果。另一种方法是采用下三角矩阵或上三角矩阵，但对于这种方式而言，新生成的向量相关性太高，也不能达到完全的混乱效果。因此，为了达到充分混乱的效果，从而采用 $\boldsymbol{B}_i=\boldsymbol{U}_i\boldsymbol{B}$ 的形式。对于这两种变换策略以及本文所采用的变换策略，第 4 节在实验的基础上，给出了它们的对比和分析。

注释 2 $\boldsymbol{A}_i$，$\boldsymbol{C}_i$ 中的未知元素为随机生成的 0 或者 1，而没有随机生成大整数。这主要是考虑到在实际使用中，格中元素的规模都比较大，如果采用与格中元素规模相当的整数，那么矩阵相乘之后会引起数据膨胀，从而造成计算效率下降。

注释 3 初始种群的数量是遗传算法的重要参数，如果初始种群数量过多，算法会占用大量系统资源；如果初始种群数量过少，算法很可能忽略掉最优解。由于格基约化算法的复杂度相对较高，在实际操作时，一般选取种群规模为 50 左右。实验结果显示，即使种群规模选择为 20，也往往能够得到比原始算法质量更高的一组基。

3.2 适应度函数

适应度函数是遗传算法的重要部分，它的好坏决定了种群进化的方向以及是否能够进化出需要的个体。格基约化算法的目标是求出格中的最短向量，因此，约化基首向量的长度是适应度函数的基本度量。同时，对于约化基而言，如果基向量的长度之和相对较小，也即这组基的整体向量长度较短，那么这组基整体质量就较高，因此，我们选取基向量长度的平均值也作为适应度函数的一个基本度量。于是，适应度函数就是首向量长度和基向量长度的平均值这两个度量的加权和。

对于格 L 的一组基 $\boldsymbol{B}=(\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n)$，设其约化基首向量的长度为 $y_1(\boldsymbol{B})$，约化基向量长度的平均值为 $y_2(\boldsymbol{B})$，适应度函数为 $y(\boldsymbol{B})$。假设初始种群的集合为 $\mathcal{A}$，那么，适应度函数为

$$y(\boldsymbol{B})=\frac{\max\{\omega_1y_1(\boldsymbol{A})+\omega_2y_2(\boldsymbol{A}),\boldsymbol{A}\in\mathcal{A}\}}{\omega_1y_1(\boldsymbol{B})+\omega_2y_2(\boldsymbol{B})}$$

其中 ω_1,ω_2 为加权系数，需要根据实际情况进行确定。

对于初始种群中的 m 组基，对每一个进行适应度函数计算，于是可以产生 m 个适应性结果。将这些结果按照适应性结果排序，适应度值较大的向量组为“较好适应”结果，适应度值较小的向量组为“较差适应”结果。

3.3 遗传算子

对初始种群进行适应度计算后，遗传算子的任务就是对种群中的个体按照适应度施加一定的操作，从而实现优胜劣汰的进化过程，它是遗传算法的核心。从优化搜索的角度而言，遗传算子可以使问题的解逐代优化，并逼进最优解。遗传算子主要包含选择、交叉、变异 3 个基本操作，它总是通过这些操作对种群加以变化，从而形成下一代种群。对于一般的遗传算法，总是需要对个体进行适当的编码，比如利用浮点数、二进制串或者十进制串，从而能够进行标准的遗传算子操作。然而，本文只是采用遗传算法的基本思想，而且格基难以进行合适的编码，因此并没有对其进行编码，从而在进行遗传算子操作时，也与传统遗传算法不同。

选择操作 选择操作的目的是让优秀的基因(即短向量)遗传到下一代个体中，我们将当前代的所有个体按照适应度排序，根据排序结果选择靠前的个体参与交叉运算。选择种群数量的一半作为父代个体进行交叉，个体被选择后，可随机组成交配对，以供后面的交叉操作。

交叉操作 交叉是将选择出来的两个父代个体的部分结构加以替换重组而生成新个体的操作。通过交叉，遗传算法的搜索能力得以飞跃提高，它在遗传算法中起着核心作用。随机选择父代的两个个

体 $\boldsymbol{B}_1$ 和 $\boldsymbol{B}_2$，按照一定的概率进行交叉操作。对这两组基联合进行 LLL 运算，就得到了一组新的基，这样短向量就被保存了下来，所有的父代基因也都参与到了下一代的运算中去。

一般情况下，交叉算子将两个个体的基因进行替换组合，对于格基约化算法而言，如果采用这种方式，格基的维数可能会降低，就构不成给定格的一组基。因此，我们采取联合进行 LLL 运算的方式。直观上看，进行 LLL 运算的向量个数变为原来的 2 倍，计算量将增加。但是，$\boldsymbol{B}_1$ 和 $\boldsymbol{B}_2$ 构成的向量组是线性相关的，其维数仍然为 n，在进行 LLL 运算时首先会把其消除，其运算量可以忽略不计。同时，进行下一次的适应度计算仍然需要求取其 LLL 约化基，这里提前进行计算也避免了适应度计算时的 LLL 约化过程。因此，交叉算子采用父代个体联合进行 LLL 约化的方式。

变异操作　变异是指在遗传过程中，新产生个体中的元素会以一定的概率出错，它可以防止种群进化停滞不前，陷入局部最优解而无法跳出。变异操作通常会按照预先设定的概率，随机选定变异位置，用新的值来代替个体相应位置元素的值。变异一般有单点变异、双点变异、多点变异等方式。对于格基而言，我们采取双点变异操作。设格基为 $\boldsymbol{B}=(\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n)$，按照预先给定的概率随机在 $1,2,\cdots,n$ 中选取两个位置作为变异点，并对变异点的向量取反，这样就可以产生一组新的基，完成变异操作。

假设初始种群个数为 m，交叉概率为 p_c，变异概率为 p_m，根据前述选择、交叉和变异操作，我们对遗传算子进行描述如下。

算法 2　格基约化算法框架的遗传算子

输入：按照适应度排序的 m 组基 $\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_m$

输出：子代种群共m组基

(1)选择 $\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_{\lfloor m/2\rfloor}$ 作为父代个体，$\lfloor m/2\rfloor$ 表示小于 $m/2$ 的最大整数；

(2)从父代个体中随机选择两组基 $\boldsymbol{B}_i,\boldsymbol{B}_j$；

(3)生成 (0,1) 之间的数 p_1；

(4)如果 $p_1\le p_c$，进行交叉操作 $\mathrm{LLL}(\boldsymbol{B}_i,\boldsymbol{B}_j)$，产生新的基 $\boldsymbol{B}_{\mathrm{new}}$；

(5)生成 (0,1) 之间的数 p_2；

(6)如果 $p_2\le p_m$，对交叉产生的基 $\boldsymbol{B}_{\mathrm{new}}$ 进行双点变异操作；

(7)将产生的个体加入到子代种群中，如果种群个数小于 m，返回第(2)步；

(8)输出子代种群 m 组基。

图 2 给出了基于遗传策略格基约化算法框架遗传算子的基本流程。

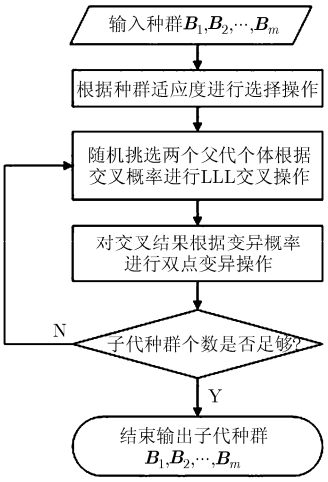

图 2 格基约化算法框架的遗传算子

3.4 终止条件

遗传算法一般在个体的适应度达到给定的阈值或者个体的适应度比较稳定的情况下终止，而预设固定的代数作为终止条件也是常用的一种方法。但是，格基约化算法经过遗传算子操作后往往会趋近于同一组基。也就是说，种群中的个体经过格基约化算法后往往会得到同一组基，这就造成种群规模减小，从而无法选出足够的父代个体。同时，测试发现，格基约化算法比较敏感，常常会在较小的代数就趋于稳定。因此，算法在种群规模小于初始规模的一半时终止，也即当种群中有一半以上的个体适应度相同时，算法终止。

4　实验结果及分析

基于遗传策略的格基约化算法能够得到质量更高的一组基，本节对其进行了有效实现，给出了部分实验结果并进行了分析。

实验采用 Intel Core2 Duo CPU E7500 2.93 GHz, 2 G 内存，Windows XP 操作系统，编程语言采用 C++，编程环境为 Visual Studio 2005。实验的基本数据类型以及部分函数使用了 NTL 包 5.5.2 版本[10]，实验中使用的格基约化函数是在 NTL 包的基础上修改得到的。实验数据采用 Goldstein-type 格[11,12]，它是密码学中常用的格，也是 SVP 挑战中的格，我们根据 SVP 挑战中格的产生方法来生成格基。随机产生维数为 $n=100$ 的格，格中元素的数据长度为 $10n$，约为 1000 bit。

根据 LLL 算法的实现效率，选取种群规模为 20，适应度函数中的加权系数 $\omega_1=0.5,\omega_2=0.5$，交叉概率为 0.9，变异概率为 0.1, LLL 算法的约化参数为 $\delta=0.99$。图 3 和图 4 给出了基于遗传策略的 LLL 算法框架最优个体适应度、平均适应度的进化过程以及最短向量随种群进化的长度变化。

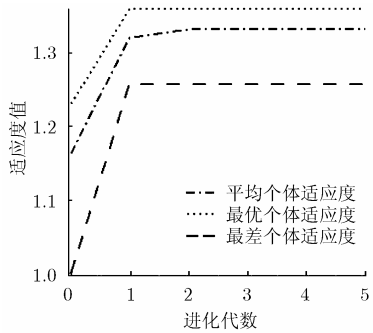

图 3 适应度的进化过程

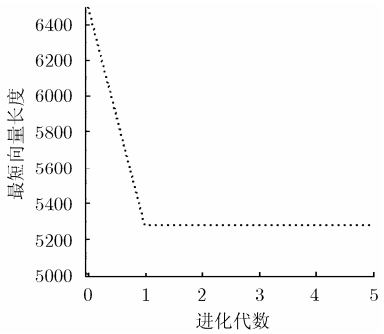

图 4 最短向量的长度变化

对于基于遗传策略的 LLL 算法框架而言，随着进化代数的增加，个体适应度值逐步增加，但是，适应度往往会在第 2 代或者第 3 代就趋于稳定，算法能够达到局部最优解。由图 4 可以看出该算法能够找到比原始算法结果更好的短向量，例如 LLL 算法求得的短向量长度为 6496，采用该算法得到的短向量长度为 5277。需要说明的是，实际所得向量长度不是一个整数，但按照 SVP 挑战的做法，我们对小数部分四舍五入，向量的长度都用整数表示。

为了检验算法的有效性，我们利用基于遗传策略的格基约化算法挑选部分 SVP 挑战数据进行了测试，表 1 和表 2 分别给出了从 50 维到 72 维关于 SVP 挑战的详细测试结果。表中数据说明新算法都能达到或超过公开的挑战结果。例如在表 1 中，也即维数相同种子相同的情况下，维数为 58 种子为 9999 时可以得到长度为 1993 的向量，而记录为 2009；维数为 66，种子为 3 时可以得到长度为 1970 的向量，而记录为 2099；维数为 72 种子为 0 时可以得到长度为 2115 的向量，而记录为 2179。在表 2 中，对于相同维数的格，新算法总能给出比挑战记录更短的向量，维数为 58 得到长度为 1747 的向量，而记录为 2009；维数为 68 得到长度为 1964 的向量，而记录为 2038；维数为 72 得到长度为 2048 的向量，而记录为 2179。

表 1 维数和种子均相同的 SVP 挑战测试结果

维数	种子	挑战记录	本文结果
50	0	1893	1893
52	0	1906	1906
54	0	1921	1921
56	0	1974	1974
58	9999	2009	1993
60	0	1943	1943
62	0	2092	2092
64	0	2103	2103
66	3	2099	1970
68	0	2141	2141
70	0	2143	2143
72	0	2179	2115

表 2 维数相同的 SVP 挑战测试结果

维数	挑战记录		本文结果	
	种子	长度	种子	长度
50	3203	1472	3203	1472
52	45	1722	796	1615
54	25	1877	988	1696
56	2	1776	539	1710
58	9999	2009	898	1747
60	45	1776	526	1752
62	137	2048	776	1817
64	5	1938	40	1886
66	3	2099	3	1970
68	3545421	2038	16	1964
70	17	1986	17	1986
72	0	2179	16	2048

基于遗传策略的格基约化算法能够得到长度更短的向量，但需要说明的是，和原始格基约化算法相比，新算法时间开销较大，下面我们来分析新算法的复杂度。新算法的时间开销主要有两个：一是初始种群选取时大量的矩阵相乘，二是遗传算子中每组基的 LLL 约化。显然，对于 $n\times n$ 矩阵来说，如果采用标准的矩阵乘法，其时间复杂度为 $O(n^3)$，而对于 LLL 算法而言，其时间复杂度为 $O(n^4\log A)$，其中 A 为格基中向量长度的最大值。于是，假设初始种群的数量为 m，进化代数为 r，那么新算法的整体时间开销为 $O(mn^3)+O(mrn^4\log A)$。由于只是在初始种群选取时进行矩阵相乘，因此，相对于遗传算子中每组基的 LLL 约化，矩阵相乘的时间消耗还是比较小的。但是总体而言，新算法的时间开销相对较大，这也是新算法的缺陷之一。

需要指出的是，初始种群的选取对算法的影响很大。如果算法在初始种群选取时就采用合理的策略，那么算法会很快就能够得到需要的短向量，种群的数量以及进化代数都会减少，从而算法也就会

有更佳的效率。也即是说，初始种群的选取是影响算法实现效率和约化结果的一个重要环节。如前所言，对初始基的变换可以采取初等矩阵、下(上)三角矩阵以及本文所采用的幺模乘积矩阵 3 种形式，下面我们在实验结果的基础上，给出这些策略的一些分析和对比。

由于初始种群的选取直接影响到了算法的进化代数，因此，很难从理论上给出不同的选取策略所得到算法的复杂度对比。我们以 60~70 维的格为实验对象，分别考察上述 3 种策略所消耗的时间以及所得到的短向量长度，表 3 给出了具体数据。

由表 3 可以看出，采用初等矩阵来选取初始种群时间消耗最少，而本文所采用的方法时间消耗较高。但是，本文所采用的方法往往能够得到长度更短的向量，在 6 组实验中，只有格的维数为 66 时 3 种不同方法得到的向量长度相等。实际上，这是可以预见到的。采用初等矩阵的变换方法，一是矩阵的乘积花费的时间较少，二是算法迭代的代数减少，更容易收敛，从而整体时间消耗较少。然而，相对于本文所采用的幺模乘积矩阵的方法，初等矩阵的变换方法的混乱效果较差，从而所得到的短向量相对较长。在实际的使用中，我们可以根据所需要的向量长度来选取合适的策略产生初始种群。当然，这里的结果是在实验基础上进行的一些分析，并且我们提出的只是最为普通的一种方式，因此，是否还有其它方式进行初始种群的选取或者说如何设计更为有效的选取策略使得算法整体效率更高就成为下一步重点研究的内容。

5　结束语

近年来，格基约化逐渐成为密码分析领域中一个强有力的工具，格基约化算法也成为研究的热点。本文提出基于遗传策略的格基约化算法，能够获得长度更短的向量和约化效果更好的一组基。和传统的格基约化算法不同，新算法从多个初始格基入手，为格基约化算法的研究提出了一个新的思路。需要指出的是，本文所提出的算法只是一个基本框架，算法中的模块都采用较为简单的方式实现，如何设计更为有效的基本模块也就成为我们下一步研究的重点。同时，针对格基约化算法本身，由于它是一个循环迭代的结构，如何利用遗传算法的思想对其进行优化实现也是下一步需要考虑的问题。

表 3 不同初始种群选取策略实验结果

维数	初等矩阵		下(上)三角矩阵		幺模乘积矩阵	
	时间消耗(s)	短向量长度	时间消耗(s)	短向量长度	时间消耗(s)	短向量长度
60	47	2054	66	1956	76	1943
62	57	2125	69	2125	84	2092
64	68	2123	85	2116	109	2103
66	85	2157	96	2157	122	2157
68	96	2268	118	2220	147	2141
70	113	2238	127	2238	164	2143

参 考 文 献

[1] Lenstra A K, Lenstra H W, and Lovasz L. Factoring polynomials with rational coefficients[J]. *Mathematiche Annalen*, 1982, 261(4): 515–534.

[2] Coppersmith D. Finding a small root of a univariate modular equation[J]. *LNCS*, 1996, 1070: 155–165.

[3] Santanu S. Some results on cryptanalysis of RSA and factorization[D]. [Ph.D. disstertions], Indian Statistical Institute, Kolkata, 2011.

[4] Kumar R S, Narasimam C, and Setty S P. Lattice based tools in cryptanalysis for public key cryptography[J]. *International Journal of Network Security & Its Applications*, 2012, 4(2): 155–162.

[5] Schnorr C P. Block reduced lattice bases and successive minima[J]. *Combinatorics, Probability and Computing*, 1994, 3(4): 507–522.

[6] Hanrot G, Pujol X, and Stehle D. Terminating BKZ [OL]. http://eprint.iacr.org/2011/198, 2011, 4.

[7] Darmstadt University. TU Darmstadt Lattice challenge[OL]. http://www.latticechallenge.org, 2011, 4.

[8] Backes W and Wetzel S. The effect of sorting on lattice basis reduction[OL]. http://www.cs.stevens.edu/~wbackes/paper/, 2007, 7.

[9] Nguyen P Q and Valle B. The LLL Algorithm: Survey and Applications[M]. 1st Edition, Berlin: Springer Publishing Company, 2009: 19–71.

[10] Shoup V. Number Theory Library (NTL) for c++[OL]. http:// www.shoup.net/ntl/, 2010, 5.

[11] Nguyen P Q and Stehle D. LLL on the average[J]. *LNCS*, 2006, 4076: 238–256.

[12] Goldstein D and Mayer A. On the equidistribution of Hecke points[J]. *Forum Mathematicum*, 2003, 15(2): 165–189.

刘向辉：　男，1984 年生，博士生，研究方向为公钥密码、网络密码.

韩文报：　男，1963 年生，教授，博士生导师，研究方向为密码学和信息安全.

权建校：　男，1983 年生，助理研究员，研究方向为网络密码.