一种适用于高维格基约化的综合算法

曹金政[1] 程庆丰[1),2)] 李兴华[3)]

[1)](战略支援部队信息工程大学 郑州 450001)

[2)](数学工程与先进计算国家重点实验室 郑州 450001)

[3)](西安电子科技大学网络与信息安全学院 西安 710071)

摘 要 格基约化算法是求解格上最短向量问题(SVP)的一类算法,在格理论中有重要地位,尤其在格理论构造的公钥密码中发挥重要作用.目前公认效率最高的主流算法是 Blockwise-Korkine-Zolotarev(BKZ)及其改进形式 BKZ 2.0,主要思想是分块约化,调用多项式次的局部格上 SVP 算法.但是 BKZ 类算法仍然存在约化程度不够充分、在高维度格中约化效率不高的问题,也存在多种改进的算法.本文在已有算法的基础上,对 BKZ 结构进行优化,并应用筛法的最新研究成果,设计了一种新的综合算法——Blockwise-Sieving-Reduction(BSR).在预处理阶段,将格矩阵划分后分别进行 BKZ 预处理,该过程可直接进行并行化.在格基约化阶段,该算法结合 BKZ 算法与筛法的优点,使用分块逐次增大的多轮 BKZ 算法进行预处理,并在 BKZ 结构中使用改进的筛法替代原有的枚举子过程,通过插入向量改进局部格的性质,提高了 BKZ 算法的效率,使之能在更大的分块下求解 SVP.针对更高维度的格矩阵,设计了递归调用的算法变种,称为 i-BSR 算法,该算法使用了渐进约化等实现技术,可以进行更大维度格的约化.从理论角度进行分析,论证了该算法可以进行格基约化并求格上短向量.实验结果表明,该算法在较大分块下,能够以可接受的时间代价完成 SVP 求解,且得到的向量优于已有算法的实验结果,新算法得到的首向量长度可以缩短至 BKZ 2.0 的 90%.

关键词 格基约化;最短向量问题;BKZ 算法;筛法

中图法分类号 TP309 **DOI 号** 10.11897/SP.J.1016.2021.00937

A Synthetic Algorithm for High Dimension Lattice Reduction

CAO Jin-Zheng[1] CHENG Qing-Feng[1),2)] LI Xing-Hua[3)]

[1)](*Strategic Support Force Information Engineering University, Zhengzhou* 450001)

[2)](*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou* 450001)

[3)](*School of Cyber Engineering, Xidian University, Xi'an* 710071)

Abstract The lattice reduction theory is key to the lattice theory and specifically lattice cryptology. It is recognized that lattice reduction algorithm is the most practical method to solve the shortest vector problem in the lattice and therefore plays an increasingly important role in cryptography. Today the most efficient and most welcome algorithm is Blockwise-Korkine-Zolotarev(BKZ) and its improved form BKZ 2.0, which will give a short lattice vector after calling SVP oracle on projected sublattices for polynomial times, yet these algorithms still cannot reduce the lattice to our expectations, and there have been several proposals for improved or modified algorithms using the basic idea and structure of BKZ. Based on the existing algorithms, we optimize the block wise reduction structure and apply the latest research results of the sieving method to design a new comprehensive algorithm in large lattices, Blockwise-Sieving-Reduction(BSR). We explain that our algorithm can serve as a lattice reduction algorithm and give a short vector in the

收稿日期:2020-03-21;在线发布日期:2020-11-25.本课题得到国家自然科学基金项目(61872449,U1708262,U1736203)资助.**曹金政**,硕士研究生,主要研究方向为公钥密码、格算法.E-mail:caojinzheng@126.com.**程庆丰**(通信作者),博士,副教授,主要研究方向为公钥密码、密码协议.E-mail:qingfengc2008@sian.com.**李兴华**,博士,教授,主要研究领域为无线网络安全、隐私保护.

lattice not worse than BKZ. It has been found that certain versions of sieving are more efficient in solving SVP of large dimension than enumeration that is embedded in BKZ-like algorithms. We show that our algorithm combines the block wise reduction structure of BKZ algorithm and the efficiency of sieve method. Furthermore, we try a new way to preprocess the lattice before the main reduction process. We divide the entire lattice matrix into several small lattice matrices while maintain the lattice vectors, then we apply multiple rounds of BKZ algorithm with different small block sizes to increase the speed of pre-processing. In this part, we can run parallel threads, each for the reduction of a sub-lattice. This is a plain application of parallel computing and requires no modifications on the algorithm. In the main reduction phase, our algorithm replaces the original enumeration sub-process in the BKZ structure with the improved version of sieve method to get a short vector in projected lattices. We expect that this modification will give us shorter sublattice vectors and therefore improve the properties of the local projected lattices, so as to improve the efficiency of the lattice reduction algorithm to make it capable of solving the SVP under a larger block. For the reduction of larger lattice matrices, we implement a recursive variant of BSR, named i-BSR. Our algorithm run the BSR process recursively, and uses more specific techniques such as progressive reduction. We design i-BSR as a optimized version of BSR algorithm to make it suitable for larger lattices of bigger dimension. In theory, we demonstrate that the algorithm can solve approximate SVP. We make experiments on lattices of different dimensions, from 35 to 888, and ranks from 9 to 111. The experiment results show that the algorithm can complete the SVP solution with an acceptable time cost with larger block than BKZ, and the obtained vector is better than the experimental results of the existing algorithm. By comparation, the length of the first vector obtained by the new algorithm can be shortened to 90% of BKZ 2.0.

Keywords lattice basis reduction; shortest vector problem; BKZ lattice reduction algorithm; sieving

1 引 言

密码学是格理论的重要应用领域[1],尤其格理论中的格基约化在公钥密码分析中发挥了关键作用.1983 年,Shamir 利用格基约化算法攻破了基于背包问题的公钥密码体制[2];1996 年,Coppersmith 提出了利用格基约化算法求解整系数多项式同余方程的方法[3],此后该方法被广泛应用于对 RSA 体制的小指数攻击、部分密钥泄露攻击等领域中[4].近年来,格上困难问题的抗量子计算特性使其在密码研究领域成为了新的关注热点[4-5].目前格上计算性难题的研究主要关注点是 SVP 算法的改进.格上其它问题如 LWE、CVP 等也可以规约为 SVP 问题[6].求解 SVP 的算法分为精确算法和近似算法.前者包括枚举、筛法等,其时间复杂度和存储复杂度随格维度增加过大,在实际的格难题求解中不适合单独使用.目前求解这些格上困难问题的主要方法是格基约化算法.

格基约化算法将格的一组基约化得到约化基,每种算法都有不同的约化条件,而各种约化基的共同特征是格基长度递增且接近正交.根据上述定义,评价一组约化基质量的衡量标准是其最短向量的长度.实际常用的格基约减算法有枚举算法、筛法、采样算法和 BKZ 类算法等.BKZ 的基础是 LLL 算法[7].1982 年,Lenstra 等人提出了 LLL 算法,该算法能够在多项式时间内得到一组格基,其中首向量的长度近似于格上的最短向量.LLL 是第一个具有实用价值的约化算法,能够在多项式时间内求解格中短向量,广泛应用于格密码系统的攻击和分析[8].自此以后,格基约化开始在公钥密码分析中发挥重要作用.BKZ 算法的最早版本由 Schnorr 等人在 1994 年提出[9],已经发展为目前实用性最强的格基约化算法.该算法最早被应用于解决子集和问题(Subset Sum Problem).2011 年,Chen 等人提出了

BKZ 2.0 算法[10]，对 BKZ 算法进行了优化，使得格理论及基于格困难问题的密码体制得到了进一步的发展[11]. 格基约化算法可以应用于很多方面，如在格理论方面可以用来求解小整数解问题和容错学习问题[12]. 尤其在公钥密码分析方面，已被用于攻击 RSA 和 DSA 的特殊情况. BKZ 类算法的时间复杂度仍然是指数级的，消耗时间随着格规模增大和分块大小增加而增加，BKZ 实际运行消耗时间过大，超过一般计算机的处理能力.

本文从以下方面对 BKZ 类算法进行优化，提出新的格基约化算法. 第一是对算法本身的改进，主要工作是 BKZ 与筛法结合. 使用简化的单轮 BKZ 类算法对基进行分块，而后在分块的局部格上使用筛法替代枚举法作为求解局部 SVP 的语句，使用筛法找到短向量，同时利用筛法能输出多个短向量的特性，使用这些向量进一步提高约化的效率，得到更好的输出结果. 在分块约化算法的基础上，另一个优化的方面是格基预处理. 为了提高 SVP、LWE 等格上困难问题解的质量，需要输入经过约化的近似正交基. 应用 BKZ 等算法进行初步约化可以达到格基预处理的效果，为进一步求解格难题做准备. 在格基预处理阶段，使用耗时较少的算法在尽量短的时间内达到更好的约化效果. 例如采用渐进式简化 BKZ 预处理，从而提高格基约化的效率.

本文的主要贡献如下：

(1) 提出了新的格基约化算法，BSR 算法：在预处理阶段进行递进式的单轮 BKZ 处理；在约化阶段使用与 BKZ 相同的分块约化结构，并使用筛法求解局部 SVP. 经过实验，验证了 BSR 算法相比现有 BKZ 类算法具有明显的优势. 以求解 SVP 为标准进行算法结果比较，BSR 算法在分块小于 100 的情况下可以进行格基约化并输出向量，具有较好的通用性. 且 BSR 得到的向量优于 BKZ 类算法的实验结果，新算法得到的首向量长度可以缩短至 BKZ 2.0 的 0.9 倍. 在较大的格上，BSR 可以达到 BKZ 的 0.59 倍.

(2) 特别针对高维度格基约化问题，在 BSR 算法的基础上改进约化策略和预处理方法，提出 i-BSR 算法，更适合大规模格矩阵的处理，能获得更优的输出结果. 实验结果表明，以输出向量长度为指标，在维数 100 以上的格中，i-BSR 算法输出的结果相比 BSR 可以提高 10%.

(3) 本文提出的 BSR 算法和 i-BSR 算法具有通用性，在各种规模的格上都可以进行格基约化. 其中 BSR 消耗时间较少，利于在较小的格上实行约化；i-BSR 在较大的格上约化效果更强，代价是需要更多轮的计算，时间比 BSR 有增加.

本文第 2 节首先介绍格理论的相关预备知识和格基约化算法的基本概念；第 3 节针对 BKZ 的具体不足设计 BSR 算法和 i-BSR 算法，并对算法原理进行分析；第 4 节对于所设计算法的效率进行分析比较；最后在第 5 节对全文进行总结.

2　预备知识

本部分介绍格的预备知识，包括格的定义、Gram-Schmidt 正交化、最短向量问题、格基约化算法等，具体更多细节可以参看文献[13].

定义 1.　格. R^m 上 n 个线性无关向量 $\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n$ 的整数线性组合得到的集合，称为 R^m 上的格 $\mathcal{L}$. 用数学符号表示如下：

$$\mathcal{L}(\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n)=\left\{\sum_{i=1}^{n}x_i\boldsymbol{v}_i:x_i\in Z\right\}.$$

其中，称整数 n 为格的秩，记为 $dim(\mathcal{L})=n$；称整数 m 为格的维度. 若 $n=m$. 则称格 $\mathcal{L}$ 是满秩的，称向量组 $\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n$ 为格 $\mathcal{L}$ 的一组基，可以记为 $\boldsymbol{B}=(\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n)\in R^{m\times n}$. $\boldsymbol{B}$ 称为格基矩阵或格矩阵，矩阵的各行即为格的基向量. 利用此记号. 可以将格记为 $\mathcal{L}(\boldsymbol{B})=\{\boldsymbol{xB}\mid\boldsymbol{x}\in Z^n\}$，其中 $\boldsymbol{xB}$ 表示通常的向量-矩阵的乘法.

定义 2.　最短向量. 如果格 $\mathcal{L}$ 中的向量 $\boldsymbol{v}$ 满足 $\|\boldsymbol{v}\|=\min\{\|\boldsymbol{b}\|:\boldsymbol{b}\in\mathcal{L}\setminus\{0\}\}$，也即向量 $\boldsymbol{v}$ 为格 $\mathcal{L}$ 中所有向量长度的最小值，那么称向量 $\boldsymbol{v}$ 为格 $\mathcal{L}$ 中的最短向量.

最短向量问题指的是寻找一个长度最短的非零格向量 $\boldsymbol{v}\in\mathcal{L}$. SVP 是 NP 困难问题，在维度为 n 的格 $\mathcal{L}$ 中最短向量的高斯期望为

$$gh(\mathcal{L})=\frac{(\Gamma(1+n/2)\det(\mathcal{L}))^{1/n}}{\sqrt{\pi}}.$$

其中 $\Gamma(1+n/2)$ 是 Gamma 函数，上式也称为高斯启发式(GH). 向量与最短向量长度的比值称为近似因子，是最短向量问题的重要指标.

SVP 存在诸多变种，如 γ-近似最短向量问题(SVP-λ)、逐次最小长度问题(Successive Minima Problem，SMP)、最短线性无关向量问题(Shortest Independent Vector Problem，SIVP)、判定版本最短向量问题(the Decision Version of the Shortest Vector Problem，GapSVP).

大多数格上困难问题如小整数解问题、最近向量问题、容错学习问题都可以规约为最短向量问题(SVP),即找到格上的最短向量.求解 SVP 等格上计算性难题的主要算法是格基约化算法.格基约化算法的目标是找到格的近似正交的基向量,其中首向量最短.格基约化是近似求解格困难问题的一种方法,同时也是求解精确格困难问题时对格基进行预处理的重要工具.

定义 3. 格基约化.给定格 $\mathcal{L}$ 的一组基 $\{\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n\}$,然后对它进行约化.格基约化的主要目的是将这组任意给定的基转化为一组正交性较好的优质基,并使得这个优质基中的各个向量尽量最短.一般步骤是首先得到能够通过算法找到的最短向量,然后找到比这个最短向量稍长一点的向量,依次类推,直到最后找到这组基中的最后一个向量为止.通过算法输出的第一个向量即最短向量的长度可以评价格基约化算法的性能,输出向量越短,则算法越好.对格 $\mathcal{L}(\boldsymbol{B})$ 的约化也可以表示为对各格基矩阵 $\boldsymbol{B}$ 的约化.

3 BSR 算法

本部分对分块格基约化算法进行研究,提出 BSR 算法.BSR 的主要创新工作如下:一是将筛法作为解局部 SVP 的语句应用到分块结构中,替代原有的枚举法;二是优化格基预处理的策略,提高约化效率;三是对 BSR 中筛法本身的改良.最后总结上述内容,提出了新的格基约化算法:Blockwise-Sieving-Reduction(以下简称 BSR 算法);并针对大维度格的约化,提出 BSR 的变形 i-BSR 算法.

3.1 BKZ 结构中应用筛法

BKZ 与其它算法的结合是格基规约算法研究的一个方向.本部分结合筛法的最近成果,在 BKZ 的分块过程中采用 SubSieve 筛法[14],改进局部格的几何性质.

将 BKZ 的分块结构与筛法结合有两方面的根据.第一,分块约化算法是一种通用性很强的算法结构,在实际运行中,大部分的运算量用于在分块中调用局部枚举,而筛法可以达到与枚举同样的作用,即输出局部最短向量.第二,最新的研究成果表明在较小的格上,存在比枚举效率更高的筛法算法,且达到的效果与枚举相同,根据这一点,可以将 BKZ 结构与筛法相结合,提出兼具通用性和较高效率的 SVP 求解算法.在算法的第一步首先使用简化的单轮 BKZ 类算法对基进行约化,而后在约化基上使用筛法找到短向量.结合 BKZ 与筛法可以得到更好的输出结果.

在分块约化算法中应用筛法需要注意两个技术问题.首先,原始的 BKZ 算法中使用的枚举法是一种求解 SVP 的精确算法(应用剪枝法可以加快枚举速度,代价是牺牲产生最短向量的概率),因此需要找到在理论上至少比枚举更加高效的筛法;其次,根据之前的讨论,可知局部的筛法可以产生多个较短的格向量,可以将这些向量插入格中,再进行约化,进一步优化局部格的性质.此处采用筛法的一种版本 SubSieve.SubSieve 算法引入了求解 BDD 问题的 Babai 算法,提高 SubSieve 中 Babai 的维度,则可以提高算法运行的速度,起到加快局部 SVP 求解的效果.需要注意 Babai 的维度不能无限增大,否则会降低求最短向量的成功率.实现中需要达到时间效率的平衡.

筛法的输出信息中不止一个最短向量.实际上,文献[15]说明了筛法的过程中可以输出格 $\mathcal{L}$ 的前 N 位的短向量,即所有长度小于 $\sqrt{3/4}gh(\mathcal{L})$ 的向量,其中 $N=(4/3)^{n/2+o(n)}$.在局部格中,通过筛法所得的额外向量,可以求解整体格上的 SVP.假设选择指数 d,在投影局部格 $\mathcal{L}_d$ 中执行 $n-d$ 维的筛法,可以得到向量列表:

$$\boldsymbol{L}=\{\boldsymbol{x}\in\mathcal{L}_d\{0\}\mid\|\boldsymbol{x}\|\leq\sqrt{4/3}gh(\mathcal{L}_d)\} \tag{1}$$

设格 $\mathcal{L}$ 的最短向量 $\boldsymbol{s}$(长度期望为 $gh(\mathcal{L})$)投影到 $\mathcal{L}_d$ 中的一个向量上.即 $\boldsymbol{s}$ 满足 $\pi_d(\boldsymbol{s})\in\mathcal{L}_d$ 或其等价条件 $\|\pi_d(\boldsymbol{s})\|\leq\sqrt{4/3}gh(\mathcal{L}_d)$.另外,因为关系式 $\|\pi_d(\boldsymbol{s})\|\leq\|\boldsymbol{s}\|=gh(\mathcal{L})$,所以可以满足下式:

$$gh(\mathcal{L})\leq\sqrt{4/3}gh(\mathcal{L}_d) \tag{2}$$

实际情况下投影向量长度一般更短,故条件(2)可以放松:如果 $\boldsymbol{s}$ 的方向是随机均匀的,与基无关,则 $\boldsymbol{s}$ 的投影长度满足 $\pi_d(\boldsymbol{s})\approx\sqrt{(n-d)/n}\|\boldsymbol{s}\|$.确切地说,存在常量概率满足条件 $\pi_d(\boldsymbol{s})\leq\sqrt{(n-d)/n}\|\boldsymbol{s}\|$.故可以对要求进行优化,需要

$$\sqrt{(n-d)/n}\cdot gh(\mathcal{L})\leq\sqrt{4/3}gh(\mathcal{L}_d) \tag{3}$$

根据筛法得到的 $\boldsymbol{s}_d$ 可以求得完整向量 $\boldsymbol{s}\in\mathcal{L}$ 使得 $\|\boldsymbol{s}\|\approx gh(\mathcal{L})$,且 $\boldsymbol{s}_d=\pi_d(\boldsymbol{s})\in\mathcal{L}$:将 $\boldsymbol{s}$ 用格基表示为 $\boldsymbol{s}=\boldsymbol{B}\boldsymbol{x}$,其中 $\boldsymbol{B}=[\boldsymbol{B}'\mid\boldsymbol{B}'']$,$\boldsymbol{x}=(\boldsymbol{x}',\boldsymbol{x}'')$,$\boldsymbol{x}'\in Z^d$ 且 $\boldsymbol{x}''\in Z^{n-d}$.根据 $\boldsymbol{s}_d=\pi_d(\boldsymbol{B}\boldsymbol{x})=\boldsymbol{B}_d\boldsymbol{x}''$,可从 $\boldsymbol{s}_d$ 中恢复 $\boldsymbol{x}''$.为了确定 $\boldsymbol{x}'\in Z^d$,使得 $\boldsymbol{B}'\boldsymbol{x}'+\boldsymbol{B}''\boldsymbol{x}''$ 较小,其中 $\boldsymbol{B}=[\boldsymbol{B}'\mid\boldsymbol{B}'']$ 表示矩阵的分块,可将其转化为一个简单的 BDD 实例,范围是 $\boldsymbol{B}'$ 张成的 d 维格.更确切地

说，使用 Babai 的最近平面算法的充分条件是对任意 $i<d$，$|\langle \boldsymbol{b}_i^*, \boldsymbol{s}\rangle| \leq 1/2\|\boldsymbol{b}_i^*\|^2$. 放松后有充分条件：

$$gh(\mathcal{L}) \leq 1/2 \min_{i<d}\|\boldsymbol{b}_i^*\|.$$

文献[14]论证了该条件可以成立. 其原因有两点：首先，即使对于充分约化的基，前 d 个最短向量也不会小于 $1/2gh(\mathcal{L})$. 其次，假设最短向量的方向是随机的，则可以经验性地得出：

$$|\langle \boldsymbol{b}_i^*, \boldsymbol{s}\rangle| \leq \omega(\ln n)/\sqrt{n} \cdot \|\boldsymbol{b}_i^*\| \cdot \|\boldsymbol{s}\|.$$

文献给出的 SubSeive 算法步骤如下：

算法 1.　SubSieve($\mathcal{L}$, d).

输入：格 $\mathcal{L}(\boldsymbol{B})$的一组基 $\boldsymbol{B}=[\boldsymbol{B}'|\boldsymbol{B}'']$

输出：格 $\mathcal{L}(\boldsymbol{B})$的一个短向量

1. $\boldsymbol{L} \leftarrow$Sieve($\mathcal{L}_d$)，Sieve 表示原始筛法
2. 对每个 $w_i \in \boldsymbol{L}$
3. 　计算 $\boldsymbol{x}_i''$ 使得 $\boldsymbol{B}_d \cdot \boldsymbol{x}_i'' = \boldsymbol{w}_i$
4. 　$\boldsymbol{t}_i = \boldsymbol{B}'' \cdot \boldsymbol{x}''$
5. 　$\boldsymbol{s}_i \leftarrow$Babai($\boldsymbol{B}'$, $\boldsymbol{t}_i$)$+\boldsymbol{t}_i$
6. RETURN 最短的向量

SubSieve 算法的分析表明，n 维格上，SubSieve 的速度以亚指数级的系数 $2^{\Theta(n/\ln n)}$ 高于原始筛法，详细的复杂度分析见 3.5 节. 对于随机的格，在条件(2)和(4)下，SubSieve($\mathcal{L}_d$)输出格的最短向量，且复杂度为 Sieve($\mathcal{L}$)的 $poly(n)$倍.

算法的效果很大程度上取决于 GS 约化向量的长度 $\|\boldsymbol{b}_i^*\|$，实际上，对于固定的 d，$gh(\mathcal{L}_d)$的值只取决于 $\prod_{i\geq d}\|\boldsymbol{b}_i^*\|$ 的大小. 算法在 $d=\theta(n/\ln n)$条件下可以成功运行，其中进行一轮预处理的开销可以忽略.

需要指出，筛法的输出信息中不止一个最短向量. 实际上，筛法可以输出格 $\mathcal{L}$ 的前 N 位短向量，即所有长度小于 $\sqrt{4/3}gh(\mathcal{L})$的向量. SubSieve 算法可以利用这些向量求解最短向量，从而加速了 SVP 的求解，具体在 3.4 节介绍. 另外，考虑到对 BSR 算法运行效果有较大影响的因素，对算法进行了进一步优化. 首先，利用之前实验的结果，将经过 BKZ 2.0 约化的矩阵作为输入，再执行 BSR 算法；为了得到更短的向量，可以增大分块大小. 规定 BSR 的输入为经过 BKZ 预处理的格基.

3.2　格基预处理

分块约化算法的运行效率与输入的矩阵性质有很大关系，经过 BKZ 处理的矩阵有更高的筛法效率. 这就要求在约化开始阶段对格基矩阵进行充分的预处理. 此处可以多次执行 BKZ 对其进行规约.

得到格矩阵后，首先对其进行一系列的单轮 BKZ 约化，每次令 BKZ 的分块大小 β 逐次增加. 由于使用的 BKZ 设置为单轮，所以消耗的时间可以尽量减小. 预处理所用算法可以使用 BKZ 的改进型算法 BKZ 2.0，相比使用原始 BKZ 在相同分块下用时约为 BKZ 的 80%. 虽然 BKZ 2.0 输出结果比 BKZ 稍差，但是在预处理阶段这种差别较小，可以接受.

3.3　局部格筛法的优化

BSR 算法在每个分块中调用 SubSieve，作用与枚举在 BKZ 中的作用相似，是为了找到局部格中最短向量. BKZ 类算法中，当维度增大，算法大部分的计算量都在分块执行局部求解 SVP 子过程，类似地，BSR 中筛法的效率决定了整体算法的表现. 为了加速局部筛法，得到分块中最短向量，可以利用筛法本身性质对算法做进一步优化. 已知筛法的过程中可以输出格 $\mathcal{L}$ 的前 N 位的短向量，在得到的最短向量以外，保留其余的较短向量，并将其插入进格矩阵中，以此改良格基排列的性质，这在实验中是有效的算法优化方法. 具体每个向量插入的位置由插入指数定义，设 $\boldsymbol{B}$ 是格的基，$\boldsymbol{v}$ 是格 $\mathcal{L}(\boldsymbol{B})$中的向量，实数 δ 小于等于 1，则 $\boldsymbol{v}$ 的插入指数定义为

$$h(\boldsymbol{v}) = \min\{j : \|\pi_j(\boldsymbol{v})\| < \delta\|\boldsymbol{b}_j^*\|^2, n+1\}.$$

其中 $\boldsymbol{b}_j^*$ 为 Gram-Schmidt 约减基. 计算插入指数方法为 j 从 1 到 n 逐个比较判断是否满足条件 $\|\pi_j(\boldsymbol{v})\| < \delta\|\boldsymbol{b}_j^*\|^2$，并取最小的满足条件的 j 值作为插入指数. 具体内容参见文献[14]. 将 $\boldsymbol{v}$ 插入到插入指数 $h(\boldsymbol{v}) \leq n$ 处，可以提高约减效率. 利用局部筛法所得的短向量，计算其插入位置(插入指数)将筛法输出向量插入局部格，为保证各向量的线性无关，需要再对局部格进行 LLL 约化.

3.4　BSR 算法

综合以上讨论，本小节提出了新的格基约化算法，Blockwise-Sieving-Reduction(BSR).

BSR 算法主体部分流程描述如下：

算法 2.　Blockwise-Sieving-Reduction(BSR).

输入：格 $\mathcal{L}(\boldsymbol{B})$的一组基，预处理分块 β_1，总体分块 β，$f_d(x)$

输出：约化基 $\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n$

1. LLL($\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_n, \delta$)，$z=0$，$j=0$
2. FOR $\beta'=1, 2, \cdots, \beta_1$
3. 　执行子过程 β'-BKZ 2($\boldsymbol{B}$). 即分块为 β' 的 BKZ 2.0 算法
4. 赋值 $\boldsymbol{b}_1^* = \boldsymbol{b}_1$
5. LOOP WHILE $z<n-1$
6. 　$j=j+1$, $k=\min(j+\beta-1, n)$
7. 　赋值 $\boldsymbol{b}_k^* = \boldsymbol{b}_k - [\mu_{k,j}]\boldsymbol{b}_j$

8. IF $j=n$
9. $j=1,k=\beta,d=f(\beta)$
10. 在 $\boldsymbol{b}_j,\cdots,\boldsymbol{b}_k$ 张成的局部格上运行 SubSieve 算法：$List=\text{SubSieve}(\mathcal{L}(j,k),d)$
11. FOR $\boldsymbol{v}\in List$
12. 将 $\boldsymbol{v}$ 插入 $\mathcal{L}(j,k)$，并执行 LLL($\mathcal{L}(j,k)$)
13. $h=\min(k+1,n)$
14. IF $\delta c_j>\bar{c}_j$
15. LLL($\boldsymbol{b}_1,\cdots,\boldsymbol{b}_j^{\text{new}},\cdots,\boldsymbol{b}_h,\delta=0.99$)，参数为 $j,k=0$
16. ELSE
17. LLL($\boldsymbol{b}_1,\cdots,\boldsymbol{b}_h,\delta=0.99$)，参数为 $h-1,z=z+1$
18. AutoAbort
19. END LOOP
20. RETURN 约化基 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n$

算法的参数中，β_1 是预处理的最大分块大小，β 是最终的分块大小. $f_d(x)$ 是计算 SubSieve 算法输出向量个数的函数，实际计算 BSR 实例时定义为 $f(x)=11+0.075x$.

3.5 算法效率分析

为了进一步讨论 BSR 算法的时间效率，给出关于 BKZ 约化基几何关系的基本假设. 该假设由 Schnorr 提出[16]，由 Atjai 等实验验证.

假设 1. 几何级数假设(Geometric Series Assumption，GSA).

假设 $\boldsymbol{B}$ 是 BKZ-b 约化基，其体积为 1，则：

$$\|\boldsymbol{b}_i^*\|=\alpha_b^{(n-1)i/2}.$$

SubSieve 的输入格满足以下条件：

$$gh(\mathcal{L})\leq\sqrt{4/3}gh(\mathcal{L}_d) \tag{4}$$

在 GSA 的前提下，可以假设格 $\mathcal{L}$ 体积为 1，$\boldsymbol{B}$ 是 BKZ-b 约化基. 根据几何级数假设可以计算局部格 $\mathcal{L}(\boldsymbol{B}_d)$ 的体积：

$$Vol(\mathcal{L}_d)=\prod_{i=d}^{n-1}\|\boldsymbol{b}_i^*\|=\prod_{i=d}^{n-1}\alpha_b^{(n-1)/2-i}=\alpha_b^{d(d-n)/2}.$$

对于 k 维格，其最短向量的高斯启发式为 $gh(\mathcal{L})\approx Vol(\mathcal{L})^{1/k}\sqrt{k/(2\pi e)}$.

条件(4)可写成：

$$\sqrt{n/2\pi e}\leq\sqrt{4/3}\cdot\sqrt{(n-d)/2\pi e}\cdot\alpha_b^{(n-1)/2-i}.$$

两边取对数，上式可写为

$$d\ln\alpha_b\leq\ln(4/3)+\ln((1-d)/n).$$

为了使 BKZ 预处理的时间相比筛法可忽略，假设 $b=n/2$，在上式中，注意到 $\ln\alpha_b=\Theta((\ln b)/b)=\Theta((\ln n)/n)$，因此可以得到结论，条件(1)对 $d=\Theta(n/\ln n)$ 可以满足.

Babai 要求的条件(5)

$$gh(\mathcal{L})\leq 1/2\min_{i<d}\|\boldsymbol{b}_i\| \tag{5}$$

筛法可以输出足够短的向量，故该要求容易满足. 对 $i<d=o(n)$ 有：

$$\|\boldsymbol{b}_i^*\|=gh(\boldsymbol{b})^{(n-o(n))/b}=gh(\boldsymbol{b})^{2-o(1)}=n^{1-o(1)},$$

而 $gh(n)=\Theta(n^{1/2})$. 于是有结论：对于格 $\mathcal{L}$ 经过分块大小为 $n/2$ 的 BKZ 预处理的基 $\boldsymbol{B}$，SubSieve($\mathcal{L}_d$) 在 $d=\theta(n/\ln n)$ 参数下，由 $n/2$ 维 Sieve($\mathcal{L}$) 的 $poly(n)$ 倍复杂度可以找到格 $\mathcal{L}$ 的最短向量. 特别地，SubSieve 算法的速度以亚指数级的系数 $2^{\Theta(n/\ln n)}$ 高于原始筛法.

根据文献[17]，对于维数 n 和分块 β，存在 $C>0$，对输入的格基 $\mathcal{L}(\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n)$ 做 BKZ 约化，BKZ 在

$$C\frac{n^3}{\beta^2}\left(\log n+\log\log\max_i\frac{\|\boldsymbol{b}_i\|}{(\det\mathcal{L})^{1/n}}\right)$$

轮循环后输出满足的基 $(\boldsymbol{c}_i)_{i\leq n}$ 满足条件

$$\|c_1\|\leq 2(\nu_\beta)^{\frac{n-1}{2(\beta-1)}+\frac{3}{2}}\cdot(\det\mathcal{L})^{\frac{1}{n}},\nu_\beta<\beta.$$

即 BKZ 为了输出足够短的向量，需要运行 $\Omega\left(\frac{n^3}{\beta^2}\left(\log n+\log\log\max_i\frac{\|\boldsymbol{b}_i\|}{(\det\mathcal{L})^{1/n}}\right)\right)$ 次 SVP 语句.

在原始 BKZ 算法中，局部 SVP 求解是由枚举 ENUM 实现的，故整体的时间复杂度估计值为

$$\Omega\left(\frac{n^3}{\beta^2}\left(\log n+\log\log\max_i\frac{\|\boldsymbol{b}_i\|}{(\det\mathcal{L})^{1/n}}\right)2^{\Theta(\beta\log\beta)}\right).$$

在 BSR 算法中，用 SubSieve 替代枚举 ENUM 的作用，相当于 BSR 调用 $poly(n)$ 次 SubSieve，忽略更新 GS 矩阵等中间操作的时间复杂度，则算法整体的复杂度为

$$\Omega\left(\frac{n^3(3/2)^{\beta/2+o(\beta)}}{\beta^2 2^{\Theta(\beta/\ln\beta)}}\left(\log n+\log\log\max_i\frac{\|\boldsymbol{b}_i\|}{(\det\mathcal{L})^{1/n}}\right)\right).$$

与 BKZ 相比，BSR 具有更低的时间复杂度.

3.6 i-BSR 算法

分块格基约化算法主要的问题是在大维度的格上约化不够充分，无法得到足够短的向量. 对于 BSR 算法，在小维度的表现也优于大维度的格. 针对大维度格约化这一问题，3.4 节提出的 BSR 算法可以进行进一步的改进. 首先针对输入格基的预处理，先将格矩阵分割为若干矩阵. 在每个部分矩阵中进行 BKZ 约化. 这样的预处理可以避免过大的 BKZ 预处理规模. 从而加速算法，过程如下.

算法 3. BSR 预处理.

输入：格 $\mathcal{L}(\boldsymbol{B})$ 的一组基，m，预处理分块大小 β

输出：格基 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n$

1. 将格矩阵 $\boldsymbol{B}$ 按行分割为若干矩阵 $\boldsymbol{B}_1,\boldsymbol{B}_2,\cdots,\boldsymbol{B}_m$
2. 对 $\boldsymbol{B}_i,i=1,2,\cdots,m$，执行 BKZ 约化 $\boldsymbol{B}_i'=\beta-\text{BKZ}(\boldsymbol{B}_i)$
3. 将矩阵 $\boldsymbol{B}_1',\boldsymbol{B}_2',\cdots,\boldsymbol{B}_m'$ 合并为 $\boldsymbol{B}'$
4. 随机化：将 $\boldsymbol{B}'$ 的行向量随机排列

另外，可以针对 BSR 中筛法的运行策略做改进. BSR 的筛法是在分块矩阵上进行的，使用的策略是每个分块中都运行一次 Subsieve 算法. 而各个分块的基向量之间有重合部分. 在一个分块上运行一次 SubSieve 之后，会影响临近分块的格基排列，使得附近多个分块的性质都达到比较好的水平. 因此没有必要在每一步都运行一次 SubSieve，可以跳过若干轮的筛法步骤. 这个改进是基于经验提出的，在测试中，多次实验结果表明，j 的值每增加 1 运行一次 SubSieve 与 j 每增加 3 运行一次 SubSieve 所输出的最短向量没有差别. 具体计算 BSR 实例过程中，可以令 j 增加 3 时运行一次筛法. 对于 SubSieve 算法的实现也可以进一步优化：分块上执行的 Subsieve 有筛法和 Babai 两个模块，故增加 Babai 的维度可以提高运行速度，需要掌握增加的程度不能过大，否则得到向量的长度有可能增大. 最后，引入 BSR 算法的迭代形式. 将经过 BSR 约化的基再次作为输入，经过反复约化，可以获得更好的输出结果.

综上，给出 BSR 的一种改进形式 i-BSR 算法.

算法 4.　Blockwise-Sieving-Reduction(i-BSR).

输入：格 $\mathcal{L}(\boldsymbol{B})$的一组基，预处理分块 β_1，总体的分块大小 β_{begin}，β_{end}，$f_d(x)$，$jp=3$(可设为其它数值)

输出：约化基 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n$

1. LLL($\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_n,\delta$)，$z=0$，$j=0$
2. BSR 预处理
3. 赋值 $\boldsymbol{b}_1^*=\boldsymbol{b}_1$
4. LOOP β from β_{begin} to β_{end}
5. 　$j=j+jp$，$k=\min(j+\beta-1,n)$
6. 　赋值 $\boldsymbol{b}_k^*=\boldsymbol{b}_k-\lfloor\mu_{k,j}\rceil\boldsymbol{b}_j$
7. 　IF $j=n$
8. 　　$j=1$，$k=\beta$，$d=f(\beta)$
9. 　　在 $\boldsymbol{b}_j,\cdots,\boldsymbol{b}_k$ 张成的局部格上运行 SubSieve 算法：$List=\text{SubSieve}(L(j,k),d)$
10. 　FOR $\boldsymbol{v}\in List$
11. 　　将 $\boldsymbol{v}$ 插入 $\mathcal{L}(j,k)$，并执行 LLL($\mathcal{L}(j,k)$)
12. 　$h=\min(k+1,n)$
13. 　IF $\delta c_j>\bar{c}_j$
14. 　　LLL($\boldsymbol{b}_1,\cdots,\boldsymbol{b}_j^{\text{new}},\cdots,\boldsymbol{b}_h,\delta=0.99$)参数为 $j,k=0$
15. 　ELSE
16. 　　LLL($\boldsymbol{b}_1,\cdots,\boldsymbol{b}_h,\delta=0.99$)，参数为 $h-1,z=z+1$
17. 　AutoAbort；
18. 　END LOOP
19. 　返回步骤 4

比较 BSR、i-BSR 与 BKZ、BKZ 2.0 等，本节提出的新算法 BSR、i-BSR 从多个方面对已有算法进行了改进，如预处理策略、局部 SVP 语句、分块大小等，相比 BKZ 等算法具有明显优势. 各算法对比见表 1.

表 1　各算法对比

	BKZ	BKZ 2.0	pBKZ	BSR	i-BSR
解 SVP	枚举	剪枝枚举	枚举	筛法	筛法
分块大小	固定	固定	动态	固定	动态
预处理	无	LLL	BKZ	BKZ	BKZ
结束策略	无	有	无	有	有

4　实验测试

本部分对 BSR 算法的效率做了实验验证，与筛法、枚举、BKZ、BKZ 2.0 等已有算法对比，对算法解 SVP 的实际效率进行比较. 实验数据选取共 12 个不同维度 SVP 问题矩阵，规模分别为 9×35、9×35、21×125、21×125、65×450、65×450、101×666、101×666、106×830、106×830、111×888、111×888.

4.1　实验环境和参数设置

实验运行环境的 CPU 为 3.00GHz，内存 4GB. 本实验所用算法用 C++ 和 Python 实现，过程中使用了 NTL 函数库和 fplll 函数库.

影响 BKZ、BKZ 2.0 和 BSR 的主要参数是分块的大小 β. 实验使用的 BKZ 分块由 40 增加至 80，BSR 的 SubSieve 参数根据分块大小确定. 关于分块大小与算法效率的关系将在本文后续讨论

4.2　对照算法

以求解 SVP 的结果作为评价依据，比较各个算法输出向量的长度. 实验选择常见的 SVP 问题求解方法与 BSR 算法进行对比：包括精确算法中的枚举法、筛法，已知近似算法中的 BKZ 算法、BKZ 2.0 和 progressive-BKZ 算法，定义向量 $\boldsymbol{b}$ 的欧几里德范数为 $\|\boldsymbol{b}\|=(b_1^2+b_2^2+\cdots+b_n^2)^{1/2}$，又称为向量 $\boldsymbol{b}$ 的长度. 求解 SVP 需要求出格中长度最短的向量.

4.2.1　枚举

枚举法是一种求解 SVP 的精确算法，通过穷举找到格上最短向量. 为了减小复杂度，可以采用裁剪法进行优化，缩小枚举范围，代价是找到的向量有可能不是最短.

在 BKZ 中应用了枚举法，用来求局部投影格中的最短向量. 枚举算法作为 BKZ 算法的一个子程序，给出了局部投影格中的最短向量.

4.2.2　筛法

理论上已知最好的 SVP 精确解法是筛法，而枚

举算法的理论复杂度是超指数 $2^{\vartheta(n\log n)}$. 但已有的实验结果表明,筛法的实际效果比裁剪枚举算法慢几个数量级[18]. 枚举和筛法可以求解 SVP,当格规模较小时,枚举效率更高,格较大时筛法效率更高. 但是无论枚举还是筛法,能求解 SVP 的格规模都是十分有限的. 其中枚举法被应用在 BKZ 中作为局部 SVP 的求解语句使用.

4.2.3 BKZ 算法

Schnorr 等人在 LLL 算法的基础上提出了 BKZ 算法,能够得到近似因子为 $\beta^{(n-1)/\beta}$ 的短向量,其中 β 为 BKZ 算法的分块规模. BKZ 算法得到的每一块的首向量是给定格中的短向量.

4.2.4 Progressive-BKZ 算法

Progressive-BKZ 算法(pBKZ)是 BKZ 算法的变种,于 2016 年由 Aono 等人提出[19]. 其主要思想是在进行约化时,使 BKZ 分块在过程中逐渐增大,避免在约化初期使用较大分块而消耗过多时间. Progressive-BKZ 也是一种比较高效的格基约化算法,可用于 LWE 等问题的求解[20].

4.3 实验数据与分析

使用 BSR 算法和 i-BSR 分别对 12 个格矩阵进行 SVP 求解,结果如表 2 所示.

表 2 算法结果汇总表

题号	BSR	i-BSR	题号	BSR	i-BSR
1	3.7	3.70	7	164.4	159.8
2	4.0	4.00	8	156.3	163.6
3	11.4	11.55	9	210.0	192.3
4	11.0	11.00	10	200.0	195.9
5	47.4	47.10	11	234.6	222.3
6	46.3	46.40	12	240.8	218.7

使用 BKZ 求解 SVP,令分块大小为 60,所得结果见表 3.

表 3 BKZ 算法结果汇总表

题号	BKZ	题号	BKZ
1	3.7	7	247.4
2	3.9	8	256.1
3	10.6	9	309.0
4	10.2	10	313.6
5	65.2	11	432.3
6	66.1	12	444.4

实验所用的格基矩阵维数很大,这种情况下 BKZ 算法的运行效率很低,因此选用 BKZ 的改进变种更为合适.

用 Progressive-BKZ 得到的向量长度如表 4 所示.

表 4 pBKZ 算法结果汇总表

题号	pBKZ	题号	pBKZ
1	3.7	7	225.92
2	3.9	8	259.50
3	10.6	9	345.20
4	11.0	10	331.70
5	67.3	11	388.20
6	68.8	12	384.50

用 BKZ 2.0 得到的向量长度见表 5.

表 5 BKZ 2.0 算法结果汇总表

题号	BKZ 2.0	题号	BKZ 2.0
1	3.8	7	215.0
2	4.0	8	223.5
3	10.6	9	225.4
4	10.6	10	226.8
5	60.7	11	260.0
6	60.6	12	265.2

由本节实验所得数据,可以验证 BSR 算法的实际效率,并与筛法、枚举、BKZ、Progressive-BKZ、BKZ 2.0 进行比较.

表 6 汇总了本节上述各方法得到的向量长度. 分析结果可以看出,BSR 算法在高维度格中的运算较之前的各类算法有明显优势,从第 5 题到第 12 题的结果尤为如此,得到的向量长度达到了已知算法的 0.9 倍.

表 6 各算法结果汇总

题号	筛法	BKZ	pBKZ	BKZ 2.0	BSR	i-BSR
1	**3.7**	**3.7**	**3.7**	3.8	**3.7**	**3.7**
2	**3.8**	**3.8**	**3.8**	4.0	4.0	4.0
3	**9.7**	10.5	10.5	10.6	11.4	11.4
4	**10.2**	10.2	11.0	10.6	11.0	11.0
5	61.4	65.1	67.3	60.7	47.4	**45.0**
6	—	66.0	68.7	60.6	46.3	**46.3**
7	—	247.3	225.9	215.0	164.4	**159.8**
8	—	256.0	259.5	223.5	**156.3**	163.6
9	—	308.9	345.2	225.4	210.0	**192.3**
10	—	313.6	331.7	226.8	200.0	**195.9**
11	—	432.3	388.2	260.0	234.6	**222.3**
12	—	444.3	384.5	265.2	240.8	**218.7**

前四个 BSR 实例中 BSR 算法未能求出最优的解. 同时注意到在这些 BSR 实例中 BKZ 的输出向量也比筛法的结果略短. 这是因为 BSR 和 BKZ 采用了同样的分块约化算法结构,更适合较大格矩阵上的分块约化,而前四个格矩阵的规模较小,分块求解反而不利于求出整体上的最优解. 而筛法在这种情况下效率较高,可以求出整个格上的最短向量. 但是在更高维度下,筛法已经无法在有限的时间内解出格向量,而此时BSR算法可以输出结果,且格规

模越大，对比 BKZ 的提升越明显.

与 BKZ 对比可以得出结论，BSR 与 BKZ 相似，是适用性较强的格基约化算法，输出近似最短的格向量. 其 BKZ、BKZ 2.0 进行对比结果如图 1 所示.

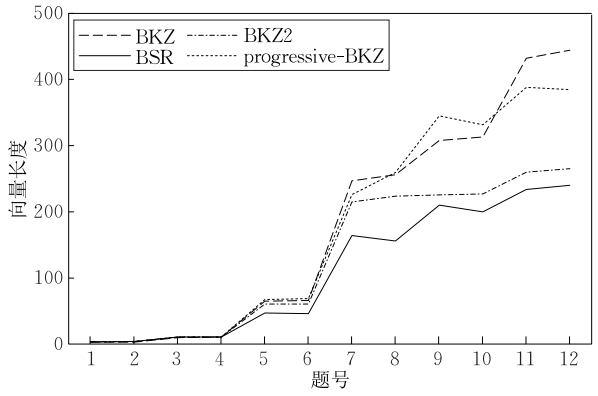


图 1　BKZ、BKZ 2、BSR 输出向量长度

分析结果可以看出，在较小的格上，各算法的结果区别并不明显，但是 BSR 算法在高维度格中的运算较之前的各类算法有明显优势，从第 5 个 BSR 实例到第 12 个 BSR 实例的结果尤为如此. 12 个实例的平均统计数据表明，BSR 的输出长度分别为 BKZ、Progressive-BKZ 和 BKZ 2.0 的 0.61、0.63、0.85 倍. 在维数高于 100 的格上，BSR 可以达到 BKZ 输出的 0.59 倍.

需要注意，BSR 运行时间显著高于原有算法. 分析其中原因如下：

(1) 实验中约化分块较大，为 100；而之前算法的分块小于 60，显然 BSR 算法需要更大的运算量.

(2) BSR 算法中包括了反复的预处理，而预处理使用的是 BKZ 2.0. 因此，新算法的运行时间实际上包含了多次运行 BKZ 2.0 的时间，所以消耗了更多的计算资源.

BSR 算法与 i-BSR 算法得到的向量长度对比见图 2. 可以看出，第 1 个 BSR 实例到第 4 个 BSR 实例，算法的输出没有变化；而在第 5 个 BSR 实例之后，除第八个 BSR 实例之外，调整参数后的 BSR 算法所得向量长度更短. 如果设置更大的分块，算法可以得到更短的向量，其代价是算法时间消耗会相应增大.

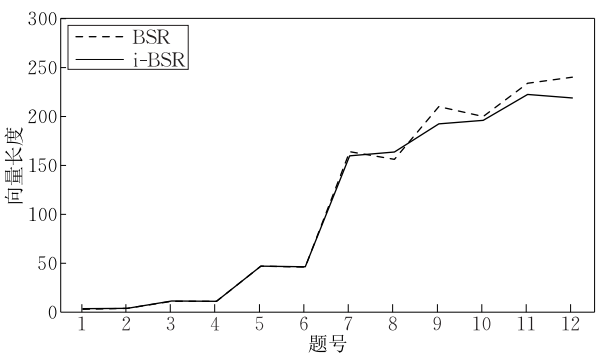


图 2　BSR 算法与 i-BSR 算法输出向量长度

通过对比，在较低维度的格上(100 以下)，BSR 与 i-BSR 没有区别. 在维度较大的情况下(100 以上)，i-BSR 比 BSR 产生了更短的输出向量. 在最好的情况下，i-BSR 解出向量的长度为 BSR 的 0.91 倍. 对于 100 维以上的格，i-BSR 输出的平均长度是 BSR 的 0.94 倍. 在较高维度的 BSR 和 i-BSR 算法结果见图 3.

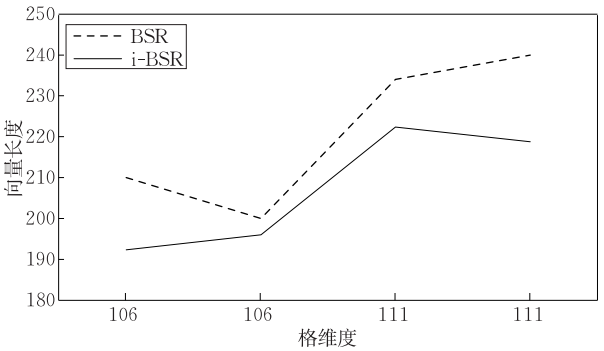


图 3　高维度下 BSR 与 i-BSR 输出向量长度

在较高格维度运行的 i-BSR 算法求得的向量优于 BSR 算法的结果. 其分块大小为 BKZ 实验的 2 倍以上，而其时间上的消耗为 3～4 倍，考虑到 BKZ 类算法的复杂度为指数级，这种时间效率是可以接受的，这也说明了 BSR 算法有较好的实用性. 算法可以从两个方面进一步提高：(1) 对 BSR 算法的拓展. 当前的 BSR 算法主要应用筛法作为约化的工具，未来可以尝试采用不同的方法，如使用随机采样算法或筛法的其它优化形式等. 此外，可以采取更灵活的策略，在算法运行过程中随时调节分块大小、SVP 语句等参数；(2) 算法的并行化. 在算法的具体实现中，如能实现并行化，可以进一步提升算法效率，减少时间消耗. 首先需要确保算法的线程安全，之后尝试实现多个处理器分别运行不同的实例和单个实例的并行处理.

5　总　结

使用格基约化算法求出格上短向量或得到格的优质基是格密码的重要理论基础. 本文在已有格基约化算法 BKZ 的基础上，结合对筛法和 BKZ 的理

论分析，提出了一种新的格基约化算法 BSR 算法. 主要思想是：(1) 在 BKZ 的框架中将局部格的求解 SVP 语句由枚举替换为 SubSieve，采用分块约化的思想，达到格基约化目的；(2) 同时结合预处理、插入向量、等优化技巧，对算法进行优化；(3) 针对大维数格的约化，改进了预处理，增加了跳选筛法轮数和迭代约化的特性，给出 BSR 的变种形式 i-BSR 算法. 相比 BKZ 等已有算法，新算法使用了新的优化技术，使格基约化效率进一步提高.

本文通过求解 SVP 问题分析了 BSR 算法的效率. 初步的理论分析和实验验证了该算法的实用性，可以得出格上的短向量. 同时其效率的优化基本合理，解出最短向量的时间仍在可接受范围内，且输出质量与已知算法相比有较大提升. 在与枚举、筛法、BKZ 等已有 SVP 算法对比中发现 BSR 综合表现更具通用性，尤其是大维度格上，在合适的参数可以达到速度与质量的较好平衡. 相比 BKZ，算法运行时间有一定程度的增加，需要继续改进.

参 考 文 献

[1] Ducas L, Plancon M, Wesolowski B. On the shortness of vectors to be found by the Ideal-SVP quantum algorithm//Proceedings of the Advances in Cryptology — Crypto 2019. Santa Barbara, USA, 2019: 322-351

[2] Shamir A. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. IEEE Transactions on Information Theory, 1984, 30(5): 699-704

[3] Coppersmith D. Finding a small root of a univariate modular equation//Proceedings of the EUROCRYPT 1996. Paris, France, 1996: 155-165

[4] Schnorr C. Factoring integers and computing discrete logarithms via diophantine approximation//Proceedings of the EUROCRYPT 1991. Brighton, UK, 1991: 281-293

[5] Bos J W, Costello C, Ducas L. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE//Proceedings of the 2016 ACM CCS. Vienna, Austria, 2016: 1006-1018

[6] Albrecht M R, Fitzpatrick R, Gopfert F. On the efficacy of solving LWE by reduction to unique-SVP//Proceedings of the Advances in Cryptology — Crypto 2013. Santa Barbara, USA, 2013: 293-310

[7] Lenstra A K, Lenstra H W, Lovasz L. Factoring polynomials with rational coefficients. Mathematische Annalen, 1982, 261(4): 515-534

[8] Ding J, Schmitt K, Zhang Z. A key exchange based on the short integer solution problem and the learning with errors problem//Proceedings of the C2SI-2019. Rabat, Morocco, 2019: 105-117

[9] Schnorr C, Euchner M. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Mathematical Programming, 1994, 66(2): 181-199

[10] Chen Y, Nguyen P Q. BKZ 2.0: Better lattice security estimates//Proceedings of the ASIACRYPT 2011. Seoul, South Korea, 2011: 1-20

[11] Nguyen P Q, Vallée B. The LLL Algorithm. Springer Berlin Heidelberg, 2010

[12] Ajtai M, Dwork C. A public-key cryptosystem with worst-case/average-case equivalence//Proceedings of the STOC 1997. El Paso, USA, 1997: 284-293

[13] Zhou Fu-Cai, Xu Jian. Lattice Theory and Cryptology. Beijing: Science Press, 2013: 60-77(in Chinese)
(周福才，徐剑. 格理论与密码学. 北京：科学出版社，2013：60-77)

[14] Ducas L. Shortest vector from lattice sieving: A few dimensions for free//Proceedings of the EUROCRYPT 2018. Tel Aviv, Israel, 2018: 125-145

[15] Albrecht M R, Ducas L, Herold G. The general sieve kernel and new records in lattice reduction//Proceedings of the EUROCRYPT 2019. Darmstadt, Germany, 2019: 717-746

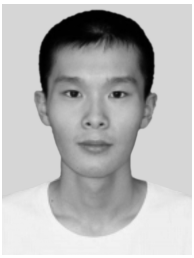
[16] Schnorr C. Lattice reduction by random sampling and birthday methods//Proceedings of the STACS 2003. Berlin, Germany, 2003: 145-156

[17] Cao Jin-Zheng, Cheng Qing-Feng. A new lattice reduction algorithm based on block random sampling method. Journal of Cryptologic Research, 2019, 6(1): 73-82(in Chinese)
(曹金政，程庆丰. 一种基于分块采样方法的格基约减算法. 密码学报，2019，6(1)：73-82)

[18] Micciancio D, Voulgaris P. Faster exponential time algorithms for the shortest vector problem//Proceedings of the ACM-SIAM 2010. Austin, USA, 2010: 1468-1480

[19] Aono Y, Wang Y, Hayashi T. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator//Proceedings of the Advances in Cryptology — Crypto 2016. Santa Barbara, USA, 2016: 789-819

[20] Wang Y, Aono Y, Takagi T. Hardness evaluation for search LWE problem using progressive BKZ simulator. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, 1984, 101(12): 2162-2170



CAO Jin-Zheng, M. S. candidate. His research interests include public-key cryptosystem and lattice algorithm.

CHENG Qing-Feng, Ph. D., associate professor. His research interests include public-key cryptosystem and cryptographic protocol.

LI Xing-Hua, Ph. D., professor. His research interests include wireless network security and privacy protection.

Background

The lattice theory is the recent trend in public key cryptology and has received increasing attention for its prospect of anti-quantum properties. The core of lattice cryptology is the hard problems in lattice, namely the shortest vector problem (SVP), the closest vector problem (CVP), and the learning with errors problem (LWE). It is proved that most hard problems in lattice can be reduced to SVP. In order to solve the shortest vector problem, that is, to find the shortest non-zero vector in a given lattice, it is essential to make use of the lattice reduction algorithms. Such algorithms aim to find a basis of the lattice which is approximately orthogonal. The lattice reduction theory is key to the lattice theory and specifically lattice cryptology. The lattice reduction algorithm is the mostly used method to solve the shortest vector problem on the lattice and plays an important role in cryptography. The current mainstream algorithm is Blockwise-Korkine-Zolotarev (BKZ) and its improved form BKZ 2.0, yet these algorithms still face the challenge of solving SVP in large lattices. There has not been a precise boundary of the time complexity of BKZ algorithm. But based on experiments and simulation, its time complexity is exponential in larger lattices.

Our study views the existing algorithms and optimize the block reduction structure of BKZ. Then we apply the latest research results of the sieving method to design a new comprehensive algorithm in large lattices, Blockwise-Sieving-Reduction (BSR). Our algorithm is designed to reduce the lattice basis and can also solve the shortest vector problem in the lattice. We intend to combine the structure of BKZ algorithm and the advantages of sieve method. It also uses multiple rounds of BKZ algorithm with different block sizes to increase the pre-processing. In the reduction phase, the algorithm replaces the original enumeration sub-process with the improved sieve method in the BKZ structure. We adopt techniques such as deep insertion to improve the properties of the local lattice, so as to improve the efficiency of the BKZ algorithm, so that it can solve the SVP under a larger block. In larger lattice matrices we implement a variant of BSR, named i-BSR. This algorithm uses more specific techniques and is suitable for larger lattices of bigger dimension. In theory, we demonstrate that the algorithm can solve SVP. The experimental results show that the algorithm can complete the SVP solution with an acceptable time cost under the larger block, and the obtained vector is better than the experimental results of the existing algorithm. The length of the first vector obtained by the new algorithm can be shortened to 90% of BKZ 2.0.