# 并行LLL算法研究综述

刘 洋[1],陈经纬[2],冯 勇[2],吴文渊[2]
1.重庆交通大学 信息科学与工程学院,重庆 400074
2.中国科学院重庆绿色智能技术研究院 自动推理与认知重庆市重点实验室,重庆 400714

**摘 要:**Lenstra-Lenstra-Lovasz(LLL)格基约化算法自1982年被提出以来,已被成功应用于计算机代数、编码理论、密码分析、算法数论、整数规划等众多领域。经过三十多年的发展,串行LLL算法的理论分析和实际效率都已得到显著改进,但仍不能满足密码分析等领域处理较大规模问题的需要。因此,并行LLL算法研究被寄予厚望。对并行LLL算法的研究现状进行了综述,总结了当前并行LLL算法设计与分析中存在的问题和难点,并对其未来发展趋势进行了展望。

**关键词:**格;格基约化;LLL算法;并行计算
**文献标志码:**A **中图分类号:**TP311.1 doi:10.3778/j.issn.1002-8331.1903-0355

刘洋,陈经纬,冯勇,等.并行LLL算法研究综述.计算机工程与应用,2019,55(16):36-41.
LIU Yang, CHEN Jingwei, FENG Yong, et al. Survey on parallel LLL algorithms. Computer Engineering and Applications, 2019, 55(16):36-41.

## Survey on Parallel LLL Algorithms

LIU Yang[1], CHEN Jingwei[2], FENG Yong[2], WU Wenyuan[2]
1.School of Information Science and Engineering, Chongqing Jiaotong University, Chongqing 400074, China
2.Chongqing Key Lab of Automated Reasoning & Cognition, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China

**Abstract:** Since 1982, the Lenstra-Lenstra-Lovasz(LLL) algorithm has been successfully applied in computer algebra, coding theory, cryptanalysis, algorithmic number theory, integer programming, etc. After over 30 years, both of theoretical and practical aspects of the sequential LLL algorithm have been significantly improved. However, it still does not satisfy the need of problems with large size, especially in cryptanalysis. Therefore, parallel LLL algorithms have its own importance. This paper surveys the state-of-the-art of parallel LLL algorithms, summarizes and analyzes the existing problems and difficulties for the current parallel LLL algorithms, and proposes the future directions for further study.

**Key words:** lattice; lattice basis reduction; LLL algorithm; parallel computing

## 1 引言

格是$\mathbb{R}^m$中一组线性无关向量的整系数线性组合形成的集合。这组线性无关的向量被称作格的一组基。关于格的一个核心计算问题是最短向量问题(Shortest Vector Problem,SVP):给定格的一组基,找到该格中的一个非零最短向量。然而,SVP问题是NP-难的[1-2]。基于SVP以及与之相关的格上计算问题的困难性,近年来涌现了许多基于格的密码方案[3],并被普遍认为可以抵御量子计算机的攻击。正因如此,格的理论和算法受到越来越多的关注。

在格算法方面,尽管SVP问题是NP-难的,但仍有大量求解SVP的指数时间算法。Hanrot等对求解SVP

**基金项目:**国家自然科学基金(No.11501540,No.11671377,No.61572024,No.11771421);中国科学院青年创新促进会项目(No.Y71A120D10)。

**作者简介:**刘洋(1984—),女,博士,讲师,CCF会员,研究领域为形式化验证、符号计算;陈经纬(1984—),通讯作者,男,博士,副研究员,CCF会员,研究领域为格基约化算法、基于格的密码学,E-mail:chenjingwei@cigit.ac.cn;冯勇(1965—),男,博士,研究员,研究领域为符号数值混合计算;吴文渊(1976—),男,博士,研究员,研究领域为符号数值混合计算。

**收稿日期:**2019-03-22 **修回日期:**2019-06-13 **文章编号:**1002-8331(2019)16-0036-06
CNKI**网络出版:**2019-06-27, http://kns.cnki.net/kcms/detail/11.2127.tp.20190626.1722.008.html

的算法进行了很好的总结归纳[4]。另一方面,由于SVP计算困难,在实际应用中,人们通常采用该问题的近似版本,比如计算格的一组Lenstra-Lenstra-Lovasz(LLL)约化基[5]。

粗略地讲,LLL约化基是格中一组比较短的基,基向量的长度不超过格中最短向量的 $2^{O(n)}$ 倍。尽管近似因子是指数的,但能在多项式时间内计算得到,并且能够满足大量实际应用的需要。比如,自LLL算法[5]被提出以来,已被成功应用于众多学科领域,包括计算机代数[5-6]、编码理论[7]、密码分析[8-9]、算法数论[10]、整数规划[11]等。另外,一些求解SVP的算法也会用LLL算法进行预处理,或重复地调用低维LLL算法作为其子算法。由于LLL约化基的应用相当广泛,相应的算法改进也大量涌现(详见文献[12]及其中的参考文献)。LLL算法的复杂度通常与格的秩 $n$ 以及输入那组基的向量的最大长度的比特位数 $\beta$ 有关。理论上,目前最优的LLL算法[13]复杂度不超过 $O(n^{4+\varepsilon}\beta^{1+\varepsilon})$,其中 $\varepsilon>0$。实际应用中,效率最高的是HPLLL[14]。然而,即使使用HPLLL,在恢复代数极小多项式的应用中,对一个维数仅为325的例子,耗时竟高达 $1.11\times10^4$ s[15]。其主要原因有以下两方面。

(1)经过三十多年的发展,尽管LLL算法的理论分析和实际计算效率都已有长足的进步,但尚存进一步改进的空间。

(2)上述这些针对LLL算法的改进和优化大多是针对串行版本的算法,对并行LLL算法的讨论相对不多。

随着LLL算法应用范围和应用规模的进一步扩大,尤其是在密码分析领域的一些应用中,采用大规模的并行计算是大势所趋。本文将对已有的并行LLL算法进行归纳和总结,探讨并行LLL算法设计和分析中存在的问题和难点,并展望并行LLL算法研究的发展趋势。

## 2　格、LLL约化基及串行LLL算法

下面主要介绍格的一些基本概念,并简要介绍串行的LLL算法及其研究现状。本文的复杂度均指位复杂度(bit-complexity);假设均使用快速算术,即两个 $n$ 比特整数的乘法复杂度为 $O(n^{1+\varepsilon})$,其中 $\varepsilon>0$;对数log均以2为底。

格是由 $\mathbb{R}^m$ 中一组线性无关向量 $(\boldsymbol{b}_i)_{i\leqslant n}$ 的所有整系数线性组合形成的集合,记为 $\mathcal{L}=\mathbb{Z}\boldsymbol{b}_1+\mathbb{Z}\boldsymbol{b}_2+\cdots+\mathbb{Z}\boldsymbol{b}_n$。这组线性无关的向量 $(\boldsymbol{b}_i)_{i\leqslant n}$ 被称为格 $\mathcal{L}$ 的一组基。这里的参数 $m$ 被称作格的维数,记为 $dim(\mathcal{L})=m$。参数 $n$ 被称作格的秩,记为 $rank(\mathcal{L})=n$。若 $dim(\mathcal{L})=rank(\mathcal{L})$,则称 $\mathcal{L}$ 为满秩格。不失一般性,本文仅考虑满秩格的情形,因为对非满秩格,可以通过仅考虑该格所在的线性子空间的方式转换为满秩格来处理。当 $rank(\mathcal{L})\geqslant2$ 时,格 $\mathcal{L}$ 有无穷组基,它们之间可以由幺模变换(行列式为 $\pm1$ 的整系数线性变换)互相转换。

给定格的任意一组基,通过对当前基进行一系列的幺模变换得到一组新的基,逐步改善基的质量(向量的长度更短或基向量间的正交程度更高)。这一过程通常被称作格基约化(Lattice Basis Reduction)。

在众多的格约化定义中,LLL约化基[5]是应用最为广泛的。在给出LLL约化基的定义之前,首先需要回顾Gram-Schmidt正交化过程。记一个向量的 $\ell_2$-范数为 $\|\cdot\|$。设 $B=(\boldsymbol{b}_i)_{i\leqslant n}$ 是格的一组基,则其Gram-Schmidt正交化是另一组互相正交的向量 $B^*=(\boldsymbol{b}_i^*)_{i\leqslant n}$,其中

$$\boldsymbol{b}_1^*=\boldsymbol{b}_1$$

$$\boldsymbol{b}_i^*=\boldsymbol{b}_i-\sum_{j=1}^{i-1}\mu_{i,j}\boldsymbol{b}_j^*,\ i=2,3,\cdots,n$$

$$\mu_{i,j}=\frac{\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{b}_j^*}{\left\|\boldsymbol{b}_j^*\right\|^2},\ 1\leqslant j<i\leqslant n$$

若令 $\mu_{i,i}=1$,且当 $i<j$ 时令 $\mu_{i,j}=0$,并记 $\boldsymbol{M}=(\mu_{i,j})$,则Gram-Schmidt正交化的矩阵表达为 $B=B^*\boldsymbol{M}^{\mathrm{T}}$。

**定义1**(LLL约化基) 格 $\mathcal{L}\subset\mathbb{Z}^n$ 的一组基 $B=(\boldsymbol{b}_i)_{i\leqslant n}$ 被称作是一组LLL约化基(LLL-reduced basis),若这组基同时满足下面两个条件:

(1)对 $1\leqslant j<i\leqslant n$ 有 $|\mu_{i,j}|\leqslant\frac{1}{2}$(规模约化的);

(2)对 $1<i\leqslant n$ 有 $\left\|\boldsymbol{b}_{i-1}^*\right\|^2\leqslant2\left\|\boldsymbol{b}_i^*\right\|^2$。

此定义的第(2)项条件与原始的LLL约化基定义有略微差别,但都具有如下性质。

**引理1**[5] 若 $B=(\boldsymbol{b}_i)_{i\leqslant n}$ 是格 $\mathcal{L}$ 的一组LLL约化基,则对任意的 $\boldsymbol{x}\in\mathcal{L}$ 有:

$$\|\boldsymbol{b}_1\|\leqslant2^{\frac{n-1}{2}}\|\boldsymbol{x}\|$$

在文献[5]中,也给出了一个计算LLL约化基的算法,通常被称为LLL算法。

LLL算法[5]

输入:格的一组基 $B=(\boldsymbol{b}_i)_{i\leqslant n}$。

1. 计算 $B^*=(\boldsymbol{b}_i^*)_{i\leqslant n}$ 和 $\boldsymbol{M}=(\mu_{i,j})$

2. 令 $i=2$

3. while $i<n$ do

　　for $j=i-1$ to 1 do

　　if $|\mu_{i,j}|>1/2$ then

　　$\boldsymbol{b}_i=\boldsymbol{b}_i-\lfloor\mu_{i,j}+0.5\rfloor\boldsymbol{b}_j$

　　if $\left\|\boldsymbol{b}_{i-1}^*\right\|^2>2\left\|\boldsymbol{b}_i^*\right\|^2$ then

　　$\boldsymbol{b}_i\leftrightarrow\boldsymbol{b}_j$;更新 $\boldsymbol{M}$;$i=i-1$

　　else $i=i+1$

输出:$B=(\boldsymbol{b}_i)_{i\leqslant n}$。

**定理1**[5] 若 $B=(\boldsymbol{b}_i)_{i\leqslant n}$ 是格 $\mathcal{L}\subset\mathbb{Z}^n$ 的一组基,则LLL算法在 $O(n^4\beta)$ 次对位长不超过 $O(n\beta)$ 的整数操作后输

出格 $\mathcal{L}$ 的一组LLL约化基，其中 $\beta=\max_i \mathrm{lb}\|\boldsymbol{b}_i\|$，即复杂度不超过 $O(n^{5+\varepsilon}\beta^{1+\varepsilon})$。

尽管原始LLL算法[5]已是多项式时间的算法，但其运行效率并不理想。随后的改进大致可以分为两类：一类以改进算法对 $n$ 的依赖程度为目标，主要采用的方式是用模算术替代原始算法中的有理算术，目前最好的结果归功于Storjohann[16]，该算法复杂度上界是 $O(n^{3+1/(5-\omega)+\varepsilon}\beta^{2+\varepsilon})$，其中 $\omega\leqslant 2.376$ 是线性代数指数。另一类以改进算法对 $\beta$ 的依赖程度为目标，关键技术是将误差可控的浮点算术引入算法中，目前最好的结果归功于Novocin等[17]，该算法的复杂度上界是 $O(n^{5+\varepsilon}\beta^{\varepsilon}+n^{4+\varepsilon}\beta^{1+\varepsilon})$。值得注意的是，Neumaier和Stehle于2017年提出了一个新型的LLL算法，融入了求解SVP的BKZ（Block Korkin-Zolotarev）算法思想，并结合模算术成功地将复杂度上界降至 $O(n^{4+\varepsilon}\beta^{1+\varepsilon})$。目前，这是理论上最好的结果。能否设计更快的LLL算法，进一步降低其复杂度是一个公开问题。

在实际应用中，有很多计算软件都实现了LLL算法，比如商用的Maple、Mathematica、Magma，开源的NTL[18]、FLINT[19]、FPLLL[20]、HPLLL[14]等。如前所述，目前运行效率最高的是HPLLL。针对LLL算法，尽管有的软件包支持多线程操作，但那是在同时计算不同格的LLL约化基意义下的并行计算，都不涉及并行的LLL算法。

近年来，我国学者在格的理论和算法方面也取得了不错的进展，包括王小云院士及其团队在格的转移定理方面有深入研究[21]，王源华等也研究过格中依次最短无关组与Minkowski 约化基之间的关系[22]，温金明等改进了关于格的Hermite常数的线性界[23]；在SVP的计算方面，王小云院士团队等在筛法方面有贡献[24-27]，潘彦斌等研究了格困难问题间的规约[28]；彭力强、毕经国等在LLL的应用方面取得了突破[29-32]。另外还有诸多关于格算法在通信中的应用和优化[33-34]。

## 3 并行LLL算法

事实上，对于计算格中短向量这一问题是否有快速的并行算法仍然是一个公开问题。具体的并行复杂性分类以及NC规约的定义可以参考文献[35]。von zur Gathen[36]证明了两个整数最大公因子的计算问题可以被 $\mathrm{NC}^1$ 规约到格的短向量计算问题。但计算两个整数最大公因子这一问题本身是否属于NC类也是一个公开问题[35]。

### 3.1 all-swap策略

在定理1中已经指出，串行的LLL算法所需的算法操作次数不超过 $O(n^4\beta)$，这可以粗略地解释为步骤3的循环次数为 $O(n^2\beta)$，而每次循环都需要 $O(n^2)$ 次算术操作。1992年，Roch和Villard提出了首个并行LLL算法[37-38]，该算法的核心思想是对不满足定义1条件（2）的那些相邻的向量尽可能并行地进行交换。具体来讲，这一策略分为两个阶段：在第一阶段中，当算法进入步骤3时，对所有奇数下标的基向量考察定义1条件（2）是否满足，若不满足，则同时交换；在第二阶段中，当接下来一次进入步骤3时，对所有的偶数下标基向量考察定义1条件（2）是否满足，若不满足，则同时交换。完整执行以上两个阶段一次被称为一次“all-swap”。在串行环境下，一次“all-swap”需要 $O(n^3)$ 次算术操作。但是，因为基的正交化过程和规模约减这两个主要步骤都适合并行，所以在有 $n^2$ 个处理器的网格上，一次“all-swap”可以在 $O(n)$ 次并行步骤（包括算术操作和通信开销）内完成。并且可以证明，在采用“all-swap”策略后，步骤3的执行次数将由 $O(n^2\beta)$ 降低到 $O(n\beta)$。但是，因为在采用“all-swap”策略后，算法中涉及的整数规模较难分析，所以在文献[37-38]中仅对并行LLL算法的复杂度进行启发式的分析。

该算法严格的复杂度分析是由Heckler和Thiele完成的[39]。同时，这篇文献还分析了基于“all-swap”策略的其他并行LLL算法[40-41]的复杂度。特别地，若采用有理算术（精确算术），在 $n\times n$ 的网格上，Roch和Villard的并行LLL算法[37-38]的复杂度不超过 $O(n^{3+\varepsilon}\beta^{2+\varepsilon})$。从而，在 $n\times n$ 的网格上，基于“all-swap”策略的并行LLL算法相对于定理1给出的串行LLL算法的复杂度达到了 $n^2$ 的（最优）加速比。于是，“all-swap”策略成为现有并行LLL算法的基本策略。

all-swap策略的变种如下：

Wetzel给出了一个分块的“all-swap”策略，设计了一个并行LLL算法[42]。在原始LLL算法中，可以看成局部的块的规模是2（两个向量参与交换），在这个分块算法中，将格的基分成了 $k$ 个互不相交的块，每个块的规模是 $\ell$。当 $k=n$ 时，就是串行的LLL算法。同时，实验也指出，最佳的分块规模取决于并行系统的体系结构。

在文献[43]中，也提及了一种被称为“全序all-swap”的策略，即不对向量下标进行奇偶区分，按照 $\|\boldsymbol{b}_i^*\|^2$ 的排序结果，在同一阶段交换所有不满足定义1条件（2）的向量。这一策略在类似的求解整数关系问题的算法PSLQ中也被用到过[44]，但这样的策略可能导致算法进入无限循环，借助文献[45]中的修复技术，可以证明算法的终止性。

### 3.2 基于近似算术的并行LLL算法

基于有理算术的算法虽是精确的，但由于众所周知的中间过程膨胀，会大大降低相关算法的实用性。为了追求效率的提高，通常会诉诸于近似算术的算法，包括基于定点数算术（Fixed-Point Arithmetic）和浮点数算术（Floating-Point Arithmetic）的算法。

事实上，Joux给出的并行LLL算法[41]就是基于定点算术的。该算法实际上是将基于定点数算术的串行LLL算法[46]和Roch-Villard的并行LLL算法各自优势结合起来而得到的。Joux对这个并行LLL算法也给出了一个启发式的复杂度分析：针对常用类型的格，比如背包格和多项式因式分解产生的格[47]，在 $n^2$ 个处理器上的复杂度不超过 $O(n^{3+\varepsilon}\beta^{2+\varepsilon})$，但这个结果建立在一个未被证明的猜想之上。

随后，Heckler和Thiele将浮点算术应用于Roch-Villard并行LLL算法，也提出了一个并行LLL算法[48]。在 $n^2$ 个处理器上，这个算法的并行算术操作次数不超过 $O(n^2\beta)$，但该算法的复杂度分析难度颇大，尚未完成。因此，在随后的诸多并行LLL算法设计中，尽管都不约而同地采用了浮点算术，但都没有理论上的复杂度分析。

**表1　并行LLL算法的复杂度**

| 算法 | $n\times n$ 的网格 |
|---|---|
| 基于有理算术[23-24] | $O(n^{3+\varepsilon}\beta^{2+\varepsilon})$ |
| 基于浮点算术[27] | $O(n^{3+\varepsilon}\beta^{2+\varepsilon})$，启发式 |

### 3.3　针对不同硬件平台的进一步改进

对于LLL格约化算法在MIMO（Multiple-Input Multiple-Output）通信系统中的应用，有一些针对超大规模集成电路（Very Large Scale Integration Circuit，VLSI）平台而设计的LLL算法[49-52]。在通信系统的应用中，所处理的对象都是复数域 $\mathbb{C}$ 上的格。对于这种情形，Jalden等指出对某些特殊情况，无法证明LLL算法的终止性[53]。因此，人们通常只关心实际的运算效率，而不考虑算法的复杂度。

针对多线程的并行LLL算法，Backer和Wetzel做了一系列的工作。他们对可移植操作系统接口（Portable Operating System Interface，POSIX）[54-55]和多核系统[56-57]都分别给出了多线程的并行LLL算法。另外，Luo和Qiao也采用了延迟规模约减的技术设计了一个多线程环境下的并行LLL算法[58]。

近年来，针对GPU支持单指令多数据（Single Instruction Multiple Data，SIMD）的特点而设计的并行LLL算法也陆续涌现。有的是针对LLL算法本身的改进，比如Jeremic和Qiao将Jacobi条件引入格约化算法，并在GPU上进行了实现[59]；有的则是通过重新设计适合GPU体系结构的数据结构，来优化并行LLL算法，达到加速的目的[60-62]。

然而，这些改进的加速效果并不明显。比如，Mariano等的实验显示[62]，在SIMD和GPU的帮助下，他们重新设计的基于向量化数据结构的并行LLL算法仅比NTL中实现LLL计算程序提速35%。由此可见，在这方面的研究还有待进一步完善。

## 4　结束语

综上所述，针对并行LLL算法的研究尽管已有大量论文发表，相关并行算法的理论分析和实际效率都有明显的改善，但是进一步改进的空间还很大。尤其是并行LLL算法的复杂度分析，尚未完成。正如Jalden等[53]所指出的那样，有的学者在证明了所设计的LLL算法在多项式次算术操作内终止后便声称相应算法具有多项式的时间复杂度，但这是不科学的。因为其中涉及的浮点数规模以及浮点运算的误差控制等因素都没有考虑。这一点，从串行LLL算法的发展历程可见一斑。

串行LLL算法自1982年[5]被提出以来，效率也是在引入浮点算术后[63]才得到显著提升。尽管基于浮点算术的LLL算法长期以来被广泛应用，但是一直没有证明其正确性，也没有分析其复杂度。直到2005年，Nguen和Stehle给出了第一个严格证明的浮点LLL算法[64]，随后串行的基于浮点算术的严格的LLL算法被陆续提出[13,17,47,65]，其复杂度也随之被改进。这些工作的关键在于如何在LLL约化基意义下进行数值分析[66]。这正是并行LLL算法目前面临的最大难点。

由于并行计算打乱了原来串行LLL算法的算术操作执行秩序，导致原有的数值稳定性分析不能被移植到并行LLL算法中，从而不能从理论上对算法进行前向误差分析，也不能得到浮点数的误差控制条件，进而不能保证浮点并行LLL算法的正确性和终止性。

鉴于此，在并行LLL算法的设计和分析方面，未来的研究可以从如下两方面入手：第一，设计支持将目前串行浮点LLL算法数值分析移植过来的新型并行LLL算法。第二，分析当前并行LLL算法的数值稳定性，并进行误差控制，从理论上保证算法的正确性和终止性，进而提出优化的改进方案。

## 参考文献：

[1] van Emde Boas P.Another NP-complete partition problem and the complexity of computing short vectors in a lattice[R].Mathematics Department，University of Amsterdam，1981.

[2] Ajtai M.The shortest vector problem in $l_2$ is NP-hard for randomized reductions（extended abstract）[C]//ACM Symposium on Theory of Computing，1998：10-19.

[3] Micciancio D，Regev O.Lattice-based cryptography[C]//Proceedings of Post-Quantum Cryptography.Berlin：Springer，2009：147-191.

[4] Hanrot G，Pujol X，Stehle D.Algorithms for the shortest and closest lattice vector problems[C]//Proceedings of Coding and Cryptology.Berlin：Springer，2011：159-190.

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

[6] van Hoeij M,Novocin A.Gradual sub-lattice reduction and a new complexity for factoring polynomials[J].Algorithmica,2012,63(3):616-633.

[7] Ling C,Howgrave-Graham N.Effective LLL reduction for lattice decoding[C]//IEEE International Symposium on Information Theory,2007:196-200.

[8] Coppersmith D.Finding a small root of a univariate modular equation[C]//International Conference on the Theory and Applications of Cryptographic Techniques.Heidelberg:Springer,1996:155-165.

[9] Coppersmith D.Small solutions to polynomial equations,and low exponent RSA vulnerabilities[J].Journal of Cryptology,1997,10(4):233-260.

[10] Kannan R,Lenstra A K,Lovasz L.Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers[J].Mathematics of Computation,1988,50(181):235-250.

[11] Lenstra Jr H W.Integer programming with a fixed number of variables[J].Mathematics of Operations Research,1983,8(4):538-548.

[12] Stehle D.Lattice reduction algorithms[C]//International Symposium on Symbolic and Algebraic Computation.New York:ACM,2017:11-12.

[13] Neumaier A,Stehle,D.Faster LLL-type reduction of lattice bases[C]//International Symposium on Symbolic and Algebraic Computation.New York:ACM,2016:373-380.

[14] Villard G.HPLLL:software library for linear algebra and Euclidean lattice problems[CP/OL].[2019-03-19].http://perso.ens-lyon.fr/gilles.villard/hplll/.

[15] Chen J.Algorithms and experiments for computing integer relations[C]//10th Conference on Computer Mathematics,Wuhan,2018.

[16] Storjohann A.Faster algorithms for integer lattice basis reduction[R].Zurich:ETH,Department of Computer Science,1996.

[17] Novocin A,Stehle D,Villard G.An LLL-reduction algorithm with quasi-linear time complexity:extended abstract[C]//ACM Symposium on Theory of Computing,2011:403-412.

[18] Shoup V.NTL:a library for doing number theory[CP/OL].[2019-03-20].https://shoup.net/ntl/.

[19] Hart W,Johansson F,Pancratz S.FLINT:fast library for number theory[CP/OL].[2019-03-20].http://flintlib.org/.

[20] The FPLLL Development Team.FPLLL:a lattice reduction library[CP/OL].[2019-03-20].https://github.com/fplll/fplll.

[21] Wei W,Tian C,Wang X.New transference theorems on lattices possessing $n^{\varepsilon}$-unique shortest vectors[J].Discrete Mathematics,2014,315/316:144-155.

[22] 王源华,尚士魁,高峰,等.格中依次最短无关组与Minkowski约化基等价的充分条件[J].中国科学:数学,2010,40(8):723-730.

[23] Wen J,Chang X W.On the KZ reduction[J].IEEE Transactions on Information Theory,2019,65(3):1921-1935.

[24] Wang X,Liu M,Tian C,et al.Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem[C]//ACM Asia Conference on Information,Computer and Communications Security,2011:1-9.

[25] Zhang F,Pan Y,Hu G.A three-level sieve algorithm for the shortest vector problem[C]//Conference on Selected Areas in Cryptography.Heidelberg:Springer,2013:29-47.

[26] 曹金政,程庆丰.一种基于分块采样方法的格基约减算法[J].密码学报,2019,6(1):73-82.

[27] Zheng Z X,Wang X Y,Xu G W,et al.Orthogonalized lattice enumeration for solving SVP[J].Science China:Information Sciences,2018,61:032115.

[28] Pan Y,Zhang F.Solving low-density multiple subset sum problems with SVP oracle[J].Journal of Systems Science and Complexity,2016,29(1):228-242.

[29] 彭力强,胡磊,黄章杰,等.模背包向量问题的实际复杂度与基于格密码体制的实际安全性[J].密码学报,2014,1(3):225-234.

[30] Bi J,Nguyen P Q.Rounding and chaining LLL:finding faster small roots of univariate polynomial congruences[C]//International Conference on Practice and Theory of Public Key Cryptography,2014:185-202.

[31] Bi J,Cheng Q,Maurice Rojas J.Sublinear root detection and new hardness results for sparse polynomials over finite fields[J].SIAM Journal on Computing,2016,45(4):1433-1447.

[32] Chen J W,StehléD,Villard G.Computing an LLL-reduced basis of the orthogonal lattice[C]//International Symposium on Symbolic and Algebraic Computation.New York:ACM,2018:127-133.

[33] Zhang W,Qiao S,Wei Y.HKZ and Minkowski reduction algorithms for lattice-reduction aided MIMO detection[J].IEEE Transactions on Signal Processing,2012,60(11):5963-5976.

[34] Zhang W,Qiao S,Wei Y.A diagonal lattice reduction algorithm for MIMO detection[J].IEEE Signal Processing Letters,2012,19(5):311-314.

[35] Cook S A.The classification of problems which have fast parallel algorithms[C]//Proceedings of Fundamentals of Computation Theory.Berlin:Springer,1983:78-93.

[36] von Zur Gathen J.Parallel algorithms for algebraic problems[J].SIAM Journal on Computing,1984,13(4):802-824.

[37] Villard G.Parallel lattice basis reduction[C]//International Symposium on Symbolic and Algebraic Computation.New York:ACM,1992:269-277.

[38] Roch J L,Villard G.Parallel GCD and lattice basis reduc-

tion[C]//2nd Joint International Conference on Vector and Parallel Processing.Berlin:Springer,1992:557-564.

[39] Heckler C,Thiele L.Complexity analysis of a parallel lattice basis reduction algorithm[J].SIAM Journal on Computing,1998,27(5):1295-1302.

[40] Heckler C,Thiele L.A parallel lattice basis reduction for mesh-connected processor arrays and parallel complexity[C]//Symposium on Parallel & Distributed Processing.Piscataway:IEEE,1993:400-407.

[41] Joux A.A fast parallel lattice reduction algorithm[C]//2nd Gauss Symposium,Munich,1993:1-15.

[42] Wetzel S.An efficient parallel block-reduction algorithm[C]//3rd International Symposium on Algorithmic Number Theory.Berlin:Springer,1998:323-337.

[43] Bartkewitz T.Improved lattice basis reduction algorithms and their efficient implementation on parallel systems[D].Bochum:Ruhr-University Bochum,2009.

[44] Bailey D H,Broadhurst D J.Parallel integer relation detection:techniques and applications[J].Mathematics of Computation,2000,70(236):1719-1736.

[45] Feng Y,Chen J,Wu W.Two variants of HJLS-PSLQ with applications[C]//International Workshop on Symbolic-Numeric Computation.New York:ACM,2014:88-96.

[46] Schnorr C,Euchner M.Lattice basis reduction:improved practical algorithms and solving subset sum problems[J].Mathematical Programming,1994,66(1):181-199.

[47] van Hoeij M,Novocin A.Gradual sub-lattice reduction and a new complexity for factoring polynomials[J].Algorithmica,2012,63(3):616-633.

[48] Heckler C,Thiele L.Parallel complexity of lattice basis reduction and a floating-point parallel algorithm[C]//International Conference on Parallel Architectures and Languages Europe.Berlin:Springer,1993:744-747.

[49] Burg A,Seethaler D,Matz G.VLSI implementation of a lattice-reduction algorithm for multi-antenna broadcast precoding[C]//IEEE International Symposium on Circuits and Systems,2007:673-676.

[50] Gestner B,Zhang W,Ma X,et al.VLSI implementation of a lattice reduction algorithm for low-complexity equalization[C]//4th IEEE International Conference on Circuits and Systems for Communications,2008:643-647.

[51] Bruderer L,Studer C,Wenk M,et al.VLSI implementation of a low-complexity LLL lattice reduction algorithm for MIMO detection[C]//2010 IEEE International Symposium on Circuits and Systems,2010:3745-3748.

[52] Ahmad U,Amin A,Li M,et al.Scalable block-based parallel lattice reduction algorithm for an SDR baseband processor[C]//2011 IEEE International Conference on Communications,2011:1-5.

[53] Jalden J,Seethaler D,Matz G.Worst-and average-case complexity of LLL lattice reduction in MIMO wireless systems[C]//2008 IEEE International Conference on Acoustics,Speech and Signal Processing,2008:2685-2688.

[54] Backes W,Wetzel S.A parallel LLL using POSIX threads[R].The Center for Discrete Mathematics and Theoretical Computer Science,2008.

[55] Backes W,Wetzel S.Parallel lattice basis reduction using a multi-threaded Schnorr-Euchner LLL algorithm[C]//European Conference on Parallel Processing.Berlin:Springer,2009:960-973.

[56] Backes W,Wetzel S.Improving the parallel Schnorr-Euchner LLL algorithm[C]//2011 International Conference on Algorithms and Architectures for Parallel Processing.Berlin:Springer,2011:27-39.

[57] Backes W,Wetzel S.Parallel lattice basis reduction-the road to many-core[C]//2011 IEEE International Conference on High Performance Computing and Communications,2011:417-424.

[58] Luo Y,Qiao S.A parallel LLL algorithm[C]//4th International Conference on Computer Science and Software Engineering.New York:ACM,2011:93-101.

[59] Jeremic F,Qiao S.A parallel Jacobi-type lattice basis reduction algorithm[J].International Journal of Numerical Analysis and Modeling:Series B,2014,5(1/2):1-12.

[60] Jozsa C M,Domene F,Pinero G,et al.Efficient GPU implementation of lattice-reduction-aided multiuser precoding[C]//10th International Symposium on Wireless Communication Systems.Piscataway:IEEE,2013:876-880.

[61] Józsa C M,Domene F,Vidal A M,et al.High performance lattice reduction on heterogeneous computing platform[J].Journal of Supercomputing,2014,70(2):772-785.

[62] Mariano A,Correia F,Bischof C.A vectorized,cache efficient LLL implementation[C]//12th International Meeting on High Performance Computing for Computational Science.Berlin:Springer,2017:162-173.

[63] Schnorr C P.A more efficient algorithm for lattice basis reduction[J].Journal of Algorithms,1988,9(1):47-62.

[64] Nguen P Q,Stehle D.Floating-point LLL revisited[C]//24th Annual International Conference on the Theory and Applications of Cryptographic Techniques.Berlin:Springer,2005:215-233.

[65] Morel I,StehléD,Villard G.H-LLL:using householder inside LLL[C]//2009 International Symposium on Symbolic and Algebraic Computation.New York:ACM,2009:271-278.

[66] Chang X W,Stehlé D,Villard G.Perturbation analysis of the QR factor R in the context of LLL lattice basis reduction[J].Mathematics of Computation,2012,81(279):1487-1511.