通过迭代压缩实现的快速格约化

Keegan Ryan, Nadia Heninger

2025 年 3 月 6 日

摘要

我们引入了一种新的格基约化算法，其近似质量类似于 LLL 算法，而实际运行表现远远快于当前的最好算法。我们通过在递归算法的结构中迭代应用精度管理技术实现了这些结果，并展示了这种方法的稳定性。我们分析了算法的渐近行为，并展示了启发式运行时间为 $O(n^{\omega}(C+n)^{1+\epsilon})$，其中 n 为格的维度，$\omega \in (2,3]$ 限制了规模约化、矩阵乘法和 QR 分解的成本，C 控制输入基 B 的条件数对数。由此可知，在实际使用中，若设精度为 $p = O(\log \|B\|_{\max})$，则运行时间为 $O(n^{\omega}(p+n)^{1+\epsilon})$。我们的算法完全可代码实现，我们已经发布了我们的实现。我们通过实验验证了我们的启发式假设的合理性，对密码学中使用的大量不同类型的格给出了标准检查程序，并展示了我们的算法显著优于现有的实现。

1 引言

格基约化是密码分析中的一项基本技术。著名的 LLL 算法 [38] 在时间 $O(n^{5+\epsilon}(p+\log n)^{2+\epsilon})$ 内为维度为 n 且元素规模为 $p = O(\log \|B\|_{\max})$ 的格实现了 $2^{O(n)}$ 的近似因子。这是多项式时间复杂度，但较大的指数使得即使对于中等规模的格，该算法在实际中也很快变得不实用。

目前实践中使用的黄金标准格基约化算法是在 fpLLL [55] 中实现的 L^2 算法 [47]，它通过精心管理精度改进了对 p 的依赖，运行时间为 $O(n^{4+\epsilon}(p+\log n)(p+n))$。当前的实现利用了硬件浮点支持，在几百维以内运行速度很快，但在此之后运行时间再次成为障碍。

另一条研究路线通过开发具有递归结构的约化算法来减少对 n 的运行时间依赖 [35,43]。这些算法具有令人印象深刻的运行时间表现，但在实际中存在一些缺点，即它们只输出一个短向量，并且缺乏代码实现。

Kirchner、Espitau 和 Fouque [32] 通过给出一种具有递归结构的算法将这两种方法结合起来，该算法在格基约化过程中降低了工作精度。他们在高维度和高精度的格基上报告了令人印象深刻的性能数据，并声称运行时间为 $\tilde{O}(n^{\omega}C)$，其中 $C > \log(\|B\|\|B^{-1}\|)$ 且 ω 等于矩阵乘法指数。这些结果基于一个关于与几何级数假设（GSA）相关的对数 Gram-Schmidt 范数线性回归的强启发式假设条件。他们利用这一假设来论证在算法执行过程中所需的精度呈指数级落差。

启发式精度问题　不幸的是，[32] 的启发式假设对于包括 NTRU [16]、Coppersmith 格（用于求解模整数的低次多项式 [14,28]）以及部分信息分解 [29] 在内的大量密码分析相关格类不成立。这些与 GSA 的偏差导致了由于精度不足而产生的不利运行时间或计算错误。在实际中，绝大多数情况下他们的算法无法约化上述的格。

我们的贡献 在这篇论文中，我们利用一种新颖的迭代策略来管理精度，而设计了一种新的 LLL 算法的递归变体。我们的算法在实际中非常快速，并且可以自然地并行化。我们针对一系列具有理论和实际意义的格族对其性能进行了测试，以展示这些格在格约化过程中由于其截然不同的结构而表现出的不同行为。我们发现我们的算法在每个测试用例中都显著优于 fpLLL，并且在几乎所有格族中都优于 [32] 的算法。我们的测试用例包括一个 8192 维的三吉字节格基，我们能够在 6.4 个核年的时间内完成约化。

现有的分析工具不适用于我们的算法，因此我们开发了新的理论分析工具来分析我们的方法的表现，证明了其稳定性和正确性。我们的渐近运行时间与 [32] 所声称的运行时间相匹配，同时所需的启发式假设显著更弱，并且我们的约化基的近似因子可以保证与 LLL 算法一一对应。

我们已经将我们的实现 [1] 提供给了社区，旨在使其成为 fpLLL 的实用即插即用替代品。

1.1 技术概览

迭代压缩 我们开发了若干新工具来支持我们算法的实现与分析。我们的算法采用了一种有别于以往工作的格约化新指标，我们称之为格基的“落差”（drop）。我们的新定义类似于传统 LLL 约化中的 Lovász 条件。我们称一个秩为 n 的基 B 是 α-格约化的，如果它是规模约化的，并且满足 $\mathrm{drop}(B) \leq \alpha n$。

此外，我们应用了一种格基“压缩”技术，将格基转换为具有相似几何特性但矩阵元规模更小且数值稳定性的好格基。这种压缩与 [51] 中的方法大致相似，尽管我们的分析是全新的。

我们的算法在算法 1 的伪代码中进行了总结。

虽然算法的一般描述简单且类似于之前的算法，但其细节和分析并非如此。压缩函数的设计对于在实践中实现数值稳定性至关重要，与 [51] 一样，它比简单地取最显著的位数更为复杂。与 [51] 不同，我们的算法迭代压缩过程，因此需要格外小心以确保累积的舍入误差保持在可管理的范围内。我们利用摄动理论得到了一阶渐近结果，以限制压缩基的元素大小和迭代压缩的精度。

在启发式假设的基础上，我们证明了以下算法的运行时间，此处以简化形式给出。

定理 1（简化版） 设 B 是一个维度为 n 的整数格基，且 $C > \log(\|B\|\|B^{-1}\|)$。如果规模约化、矩阵乘法和 QR 分解的运行时间对维度的依赖关系为 $O(n^\omega)$，其中 $\omega \in (2, 3]$，并且我们的启发式假设成立，那么我们算法的运行时间为

$$O(n^\omega (C + n)^{1+\epsilon})$$

对于在密码分析攻击中常见的上三角且规模约化的基 B，运行时间为

$$O(n^\omega (\log \|B\|_{\max} + n)^{1+\epsilon})$$

深入剖析轮廓 我们分析算法的行为，而不依赖于对主要密码学重要格类（包括 Coppersmith、NTRU、LWE 和隐藏数问题格）显然不成立的几何级数假设（GSA）。我们通过研究格约化过程中 Gram-Schmidt 向量的对数范数（即轮廓）的变化来实现这一点。与 [32] 不同，后者通过启发式方法假设轮廓紧密跟随已知斜率的线性趋势变化，我们的分析考虑了所有可能的轮廓形状及其变化。我们使用从轮廓计算得到的基落差来关联格的势的变化来调整所需的精度。

除了在分析算法行为方面的实用性外，我们以基落差重新定义约化质量，从而得到具有与 LLL 约化基相同质量的基，并且我们取得了与 [45, 定理 9] 类似的结果。在实践中，我们的算法返回与 LLL 等效质量的近似因子。

定理 2 设 B 是一个满足我们新定义的约化质量的 α-格约化的秩为 n 的基。令 $\vec{b}_i^*$ 表示第 i 个 Gram-Schmidt 向量，$\lambda_i(B)$ 表示由 B 生成的格的第 i 个剪切最小值。则 B 满足以下条件：

1. $\|\vec{b}_1\| \leq 2^{\alpha n}(\det B)^{1/n}$。
2. $\|\vec{b}_n^*\| \geq 2^{-\alpha n}(\det B)^{1/n}$。
3. 对于所有 $i \in \{1, \ldots, n\}$，$\|\vec{b}_i\| \leq 2^{\alpha n + O(n)}\lambda_i(B)$。
4. $\|\vec{b}_1\| \times \cdots \times \|\vec{b}_n\| \leq 2^{\alpha n^2 + O(n^2)} \det B$。

关注应用 最后，我们对各种格族进行了基准测试。我们全面分析了我们的算法在现在研究中常见的格上的行为，特别是来自众多密码分析应用的格构造。我们选择的测试用例表现出截然不同的轮廓变化，我们的实现算法在这些格上显著优于现有工具。

我们论文的主要目标是描述一个完全实用且可实现的算法，该算法优于现有的格约化实现，并且旨在实际应用中使用。为此，我们的理论结果描述了算法的行为，并借助比以往工作更弱的启发式假设，论证了我们的算法的数值稳定性。我们通过实验验证了我们的启发式方法的合理性。

2 背景知识

3 格的轮廓和它们的应用

4 改进的格约化算法