

The Effect of Sorting on Lattice Basis Reduction

Werner Backes and Susanne Wetzel

Stevens Institute of Technology,
Castle Point on Hudson, Hoboken, NJ 07030, USA
{wbackes,swetzel}@cs.stevens.edu

1 Introduction

Our goal is to find strategies to reduce the running time of lattice basis reduction algorithms. We therefore investigate the impact of applying simple transformations to the lattice basis before reducing it by means of the Schnorr-Euchner LLL algorithm [3].

2 Lattice Bases

In our experiments presented in this paper, we have focused on unimodular lattice bases of dimension n for a number reasons. First, they are easy to generate. Second, they all reduce to a permutation of the unit vectors $\pm e_i$ for $1 \leq i \leq n$. Transformations to the lattice basis like sorting in ascending or descending order not only impact the running time of the reduction algorithm but also the quality of a reduced lattice basis. By using unimodular lattice bases we can focus our attention on the reduction time rather than the quality of the reduction. Thus, we are able to conclude that the improvements in reduction time are a direct result of our strategy instead of a decrease in the quality of the reduced basis [4].

In addition, previous experiments [1, 2] have shown that unimodular lattice bases are hard to reduce in terms of the time needed for the reduction, if compared to other types of bases with similar dimension and length of the basis entries. Recent experiments also revealed problems with the reduction of unimodular lattice bases for NTL [7] and fpLLL [5]. In some cases, the NTL variant of the LLL either gets stuck in an endless loop or simply returns an incorrect result. While the *proved* variant of fpLLL was able to reduce all tested bases, the *fast* and *heuristic* variant failed for rather small unimodular lattice bases already.

In contrast to earlier experiments [1, 2], we have used three different methods to generate the bases and analyze the effect of a simple transformation, like sorting in ascending or descending order, on the running time of the LLL reduction algorithm. We have reduced the bases with two experimental variants of the Schnorr-Euchner LLL to see how the construction type and the sorting method affect the tested variants. We have used a modified version of LiDIA [6] for our tests.

The first construction method M_{1x} uses lower and upper triangular matrices with diagonal elements set to 1 and all other entries chosen at random. Using lower triangular matrices U_j , upper triangular matrices U_j with $1 \leq j \leq 2$ and permutation matrices P_i for $1 \leq i \leq 4$, we have used the following 6 variants to create a unimodular lattice basis B :

$$\begin{aligned}
M_{11} : B &= (U_1 \cdot V_1) \\
M_{12} : B &= (V_1 \cdot U_1) \\
M_{13} : B &= (U_1 \cdot V_1) \cdot (V_2 \cdot U_2) \\
M_{14} : B &= (U_1 P_1 \cdot V_1 P_2) \\
M_{15} : B &= (V_1 P_1 \cdot U_1 P_2) \\
M_{16} : B &= (U_1 P_1 \cdot V_1 P_2) \cdot (V_2 P_3 \cdot U_2 P_4)
\end{aligned}$$

The second construction method M_{23} starts with an identity matrix and applies swap and translation operations to the basis until the entries of the basis have reached the desired bit length. To increase the amount of variants, we have allowed the method to limit the vector operations to subsets of the lattice basis vectors. This method has been tested with 9 different parameter sets.

The third construction method M_{3x} is the combination of a lattice basis C generated by method M_{16} and D generated by M_{23} . We used 5 different parameter sets for the bases C and D .

$$\begin{aligned}
M_{31} : B &= C \cdot D \\
M_{32} : B &= D \cdot C
\end{aligned}$$

3 Experiments

The main goal of our experiments is to evaluate the impact of simple transformations on the running time of the LLL reduction algorithms and develop reduction heuristics or strategies based on our findings. In this paper we concentrate on the impact of sorting the lattice basis in ascending or descending order. To analyze the impact on the reduction time we have performed a large amount of experiments with different types of unimodular lattice bases. We have derived 25 different types of unimodular lattice bases using the three construction methods with different variants and parameter sets. For each type we have generated 1000 bases of dimension 50. In our experiments, we have fixed the maximal bit length to 500 in order to allow the use of the `double` data type for the approximation within our variants of the Schnorr-Euchner LLL. Furthermore, using a fixed dimension and a fixed maximum bit length allows us to test a larger number of lattice basis types as well as an increased of test instances per type.

Figure 1 demonstrates the impact of sorting on the running time of the LLL algorithm for unimodular lattice bases of types M_{16} and M_{31} with parameter set P_2 . All running times have been sorted in ascending order. For our tests, we have used two experimental variants V_1 and V_2 of the Schnorr-Euchner LLL algorithm. Our experiments show that the effectiveness of the LLL variants and the sorting approach are strongly dependent on the type of the lattice basis.

For M_{16} (and for $M_{1,x}$ in general), LLL V_2 is faster than LLL V_1 . Sorting in ascending order decreased the running time of LLL V_2 significantly. On the other hand, sorting in ascending order led to a major increase in the reduction time. Our experiments for M_{16} also show that sorting the lattice basis in ascending or descending order impacts the two LLL variants differently. For the hybrid basis type M_{31} (here with parameter set P_2), the sorting of the lattice basis in ascending or descending order has no impact on the reduction time. Yet, for this type of lattice bases, LLL V_1 performs better than LLL V_2 .

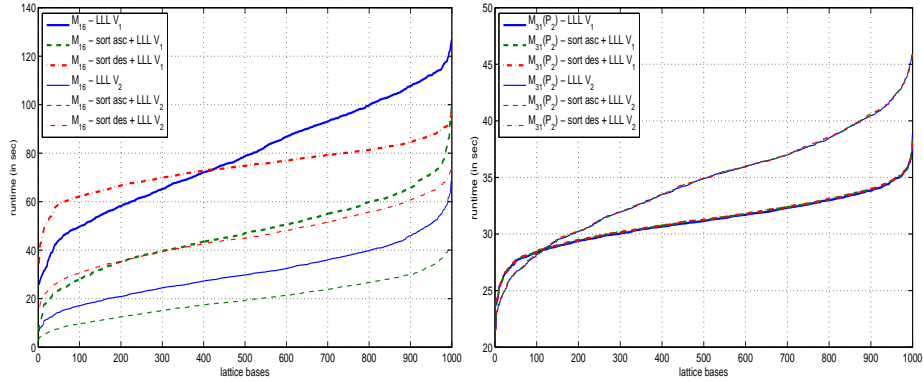


Figure 1. Effect of sorting for M_{16} , M_{31} in combination with LLL algorithm V_1 and V_2

One cannot derive a general rule as to when to use LLL V_1 instead of LLL V_2 or when to use sorting in ascending instead of descending order. For a majority of cases, we have seen a decrease in the running time after sorting the basis. On the other hand, we have found types of unimodular lattice bases where the sorting approach increases the running time significantly. Based on these observations,

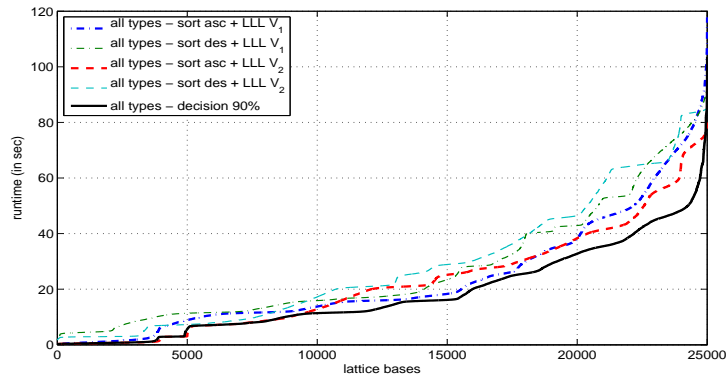


Figure 2. LLL reduction times using different strategies

a more fine-grained strategy is needed which takes the type of the lattice basis into account. In particular, one could reduce the running time of the reduction

algorithm if one was able to find a method that can efficiently decide which combination of variant of the reduction algorithm and sorting should be used. Figure 2 shows the running times for all tested types of unimodular lattice bases and every combination of sorting and LLL variants. All running times have been sorted in ascending order. Obviously, a decision algorithm that can efficiently find the right combination of sorting and LLL variant with a probability of at least 90% will reduce the overall running time considerably.

4 Future Work

We are currently developing a decision algorithm which finds the optimal combination of LLL variant and sorting in ascending or descending order. Early experiments indicate that for our setting with unimodular lattice basis it is feasible to find the optimal combination with a probability of at least 90%.

Furthermore, we are expanding our experiments to different lattice dimensions and maximum bit lengths of the lattice basis entries which will in turn enable us to also adapt our strategies and decision algorithms to these new parameters.

References

1. W. Backes and S. Wetzel. New Results on Lattice Basis Reduction in Practice. In *Algorithmic Number Theory (ANTS-00)*, volume 1838 of *LNCS*, pages 135–152. Springer, 2000.
2. W. Backes and S. Wetzel. Heuristics on Lattice Basis Reduction in Practice. *ACM Journal on Experimental Algorithms*, 7, 2002.
3. C. Schnorr and M. Euchner. Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. In *Proceedings of Fundamentals of Computation Theory '91*, volume 529 of *LNCS*, pages 68–85. Springer, 1991.
4. S. Wetzel. *Lattice Basis Reduction Algorithms and their Applications*. PhD thesis, Universität des Saarlandes, 1998.
5. fpLLL - Homepage (Damien Stehlé), May 2007. <http://www.loria.fr/~stehle/>.
6. LiDIA - Homepage, May 2007. <http://www.informatik.tu-darmstadt.de/TI/LiDIA/>.
7. NTL - Homepage, May 2007. <http://www.shoup.net/ntl/>.