SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2014, Vol. 57 092111:1–092111:10
doi: 10.1007/s11432-013-4967-6

A new parallel lattice reduction algorithm for BKZ reduced bases

LIU XiangHui[1,2]*, FANG Xing[2], WANG Zheng[1,2] & XIE XiangHui[2]

[1]*Department of Applied Mathematics, Zhengzhou Information Science and Technology Institute, Zhengzhou* 450002*, China;*
[2]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi* 214083*, China*

Received April 29, 2013; accepted June 5, 2013

Abstract In order to implement the original BKZ algorithm in parallel, we describe it in terms of parallelism and give its parallel implementation scheme. Then we analyze the efficiency of algorithm's parallel implementation and show that the speedup factor of BKZ algorithm in parallel is extremely low. Therefore we present a new parallel lattice reduction algorithm suitable for multiprocessor computer architecture. The new algorithm can obtain a BKZ reduced basis and the parallel speedup is effective. Also with the practical results, although the computational complexity increases compared with the original BKZ algorithm, we still indicate that the new algorithm performs well in parallel and the time cost in parallel is less. At the same time, we show that the length of the shortest vector is smaller.

Keywords lattice reduction, BKZ algorithm, Goldstein-type lattices, parallel algorithm, multiprocessor computer architecture

Citation Liu X H, Fang X, Wang Z, et al. A new parallel lattice reduction algorithm for BKZ reduced bases. Sci China Inf Sci, 2014, 57: 092111(10), doi: 10.1007/s11432-013-4967-6

1 Introduction

Lattice reduction is a fundamental problem in the computational aspects of the geometry of numbers, a mathematical theory coming from Lagrange's quadratic form reduction theory. It was generalized systematically by Minkowski in the 1890s [1]. In 1982, Lenstra et al. [2] presented the famous lattice reduction algorithm—LLL algorithm, which could obtain a relatively short vector in polynomial time. Since then, lattice reduction has been showing the tremendous power to cryptanalysis and becoming a new research hotspot in the field of cryptology. In 1983, the public key cryptosystem based on knapsack problem was cracked successfully by LLL algorithm in polynomial time. In 1996, Coppersmith proposed a new method for computing small roots of whole coefficient congruence equations with LLL algorithm [3]. And since then this method has been widely applied in the attacks of RSA, such as small exponent attack, partial key exposure attack. At present, the application of lattice reduction in cryptology has received increasing attention and there come out more and more research achievements [4–6].

For an n-dimensional lattice, LLL algorithm can find a non-zero short vector in polynomial time and the length of the vector is at most $2^{(n-1)/2}$ (called approximation factor) times longer than the shortest

*Corresponding author (email: lxhkz2002@163.com)

vector. Based on LLL algorithm, Schnorr and Euchner propose BKZ algorithm by adjusting a parameter to balance the efficiency and approximation factor of short vector [7]. In BKZ algorithm, the basis of the lattice is divided into some blocks and every block could generate a shortest vector with ENUM algorithm. BKZ algorithm can often find a shorter vector compared with LLL algorithm. However, the generation of shortest vector by ENUM algorithm is an exhaustive search process. When better quality of the short vector is required, a larger block is needed correspondingly and the time cost will increase exponentially. The computational complexity of BKZ algorithm is due to the block size, and so far, it is not certain wether BKZ is a polynomial time algorithm. BKZ algorithm has better reduction effect, so it has become the standard method in many applications requiring a strong reduced basis. In order to promote the implementation of lattice reduction algorithms, Darmstadt University in German supplies a sequence of lattice bases with increasing dimension as a worldwide challenge including lattice challenge and SVP challenge. For lattice challenge, the top three records so far are completed with BKZ algorithm. Because of much shorter vector required in SVP challenge, the top three records were completed through the combination of BKZ algorithm and ENUM algorithm[1)].

The complexity of BKZ algorithm is much higher when the block size is large. Therefore the serial version can hardly play an important role in dealing with high-dimensional lattices or asking for shorter vectors. Parallelization is an important way to improve the efficiency of algorithms [8], but to our knowledge, there is no parallelization scheme of BKZ algorithm in open literature as yet. However, there are some research results on the parallelization of LLL algorithm and ENUM algorithm—the sub-algorithms of BKZ algorithm. Backes and Wetzel introduces a parallel variant of LLL algorithm suitable for multithread environment. Experiments with the parallel LLL show a speedup factor of about 1.75 for the 2-thread and close to factor 3 for the 4-thread version. Then they improve the algorithm and get an increase of the speedup by a factor of 4.5 for the 8-thread version [9]. Dagdelen et al. present a parallel version of ENUM algorithm, which could gain a speedup of up to factor 14 in multicore CPU systems with 16 cores [10]. However, these algorithms are all proposed in multithread environment requiring to share memory, which limits the parallel expansibility.

In this paper, we study the parallelization scheme of BKZ algorithm aiming at the current computer architecture and we do not discuss the parallelization of LLL algorithm and ENUM algorithm. The rest of this paper is organized as follows. In Section 2, we introduce the fundamental knowledge of BKZ algorithm briefly. In Section 3, we present a new parallel algorithm suitable for multiprocessor and independent memory computer architecture, and we also give the correctness and speedup analysis of the new algorithm. Section 4 is the experimental results on parallel implementation. Finally Section 5 is conclusions.

2 Preliminaries

In this section, we give a brief introduction to the theory of lattice reduction and BKZ algorithm. For further details we refer to [11].

Definition 1. Let $b_1, b_2, \ldots, b_m \in R^n$ be linearly independent vectors where R^n is n-dimensional vector space on real number field. Then $L(b_1, b_2, \ldots, b_m) = \{z = \sum_{i=1}^{m} \lambda_i b_i | \lambda_i \in Z\}$ is a lattice and $(b_1, b_2, \ldots, b_m)$ is a basis of the lattice. We call m the dimension of the lattice. If $m = n$, the lattice is full-rank. If all the vectors of the lattice are in the integral ring Z, namely $R = Z$, the lattice is called integral lattice.

Obviously, a lattice has many different bases. When solving problems over lattices, we always want to find some interesting bases with certain special properties. Computing such bases is called lattice reduction and the special bases are called reduced bases. For different situations, the required reduced basis is always different. For example, we always want to construct a basis such that the vectors are as short as possible and as orthogonal as possible to each other.

Let $L = L(b_1, b_2, \ldots, b_n) \subset R^n$ be a lattice with basis $(b_1, b_2, \ldots, b_n)$ and let $\pi_i : R^n \to \text{span}(b_1, \ldots,$

1) Darmstadt University. TU Darmstadt lattice challenge. http://www.latticechallenge.org.

$b_{i-1})^{\perp}$ denote the orthogonal projection. Suppose $L_i = \pi_i(L)$, then L_i is a new lattice of dimension $n-i+1$ which is orthogonal with the first $i-1$ vectors. Next we always denote the Euclidean norm (the length of the vector) by $||b||$ and let $\lambda_1(L)$ denote the length of the shortest vector in lattice L.

Definition 2. For a lattice $L \subset R^n$ with basis $(b_1, b_2, \ldots, b_n)$, $(b_1^*, b_2^*, \ldots, b_n^*)$ is the corresponding Gram-Schmidt orthogonalization basis and μ_{ij} are the orthogonalization coefficients. The basis is called BKZ reduced if the following conditions are satisfied:

1) $|\mu_{ij}| \leqslant \frac{1}{2}, 1 \leqslant j < i \leqslant n$;

2) $\delta||b_i^*|| \leqslant \lambda_1(L_i(b_1, \ldots, b_{\min\{i+\beta-1,n\}}))$, $i = 1, \ldots, n-1$, $\beta \in Z^+, 2 \leqslant \beta < n$ and $\frac{1}{2} < \delta \leqslant 1$.

BKZ reduced basis has a better approximation factor $\beta^{(n-1)/\beta}$ in theory and it can be obtained by BKZ algorithm. BKZ algorithm uses LLL and ENUM as its subroutines and the description of BKZ algorithm could be referred to [12]. The complexity of BKZ algorithm is determined by the block size β and it is between the ENUM and LLL algorithm. No reasonable complexity upper bound is known and it is observed experimentally that its practical runtime seems to grow exponentially with the lattice dimension and block size β [13]. Actually, when the block size β is more than 50, ENUM subroutine takes about more than 99% time cost in BKZ algorithm [10]. So far, to our knowledge, whether BKZ algorithm can be terminated in polynomial time is still an open problem. However, BKZ algorithm behaves well in practice.

3 Design and analysis of parallel BKZ algorithm

In this section, firstly we describe the original BKZ algorithm from the viewpoint of parallelism and give the parallel implement scheme. Then we propose a new parallel algorithm suitable for multiprocessor computer architecture and analyze the efficiency of the new algorithm.

3.1 Parallel implementation of the original BKZ algorithm

In the original algorithm, there are two cycle variables z and j. Variable j cyclically shifts from $j = 1$ to $n-1$ which is called a tour. Variable z counts the number of positions j satisfying condition 2) in Definition 2. If it dose not hold, we let $z = 0$. The algorithm will not end until $z = n-1$. The terminating condition is that all the positions j satisfy condition 2) in the $n-1$ continuous comparison. Therefore, BKZ algorithm proceeds by iterating tours consisting of $n-1$ calls to a $\beta-$dimensional SVP solver (ENUM subroutine). Algorithm stops when no change occurs during a tour. As has been stated, the major time cost of BKZ algorithm is ENUM subroutine, so we can assign the subroutines into $n-1$ computing units. The termination condition is that there is no insert in every unit. Thus, we present the parallel version of original BKZ algorithm in the following.

In the serial algorithm, when no change occurs, we will run $\mathrm{LLL}(b_1, \ldots, b_h, \delta)$ again. However, in the parallel algorithm, we will run $\mathrm{LLL}(b_1, \ldots, b_n, \delta)$ after all the data is transmitted. Obviously, if the basis $(b_1, \ldots, b_n)$ satisfies the LLL reduced conditions, $(b_1, \cdots, b_h)$ also satisfies the conditions. Therefore, Algorithm 1 is the parallel scheme of the serial BKZ algorithm.

Next we analyze the efficiency of the parallel BKZ algorithm. The major time cost of BKZ algorithm is ENUM subroutine, so we can suppose that the time costs of LLL algorithm and communication are negligible. For an n-dimensional lattice L, suppose there are r ENUMs in the whole algorithm where m does not satisfy the conditions and A is the time cost of each ENUM subroutine. Thus, it is clear that the time cost of serial algorithm is rA and the parallel algorithm is mA such that the parallel speedup factor is $k = r/m$. However, the ideal speedup factor is always unreachable. The real speedup factor is always related to the block size β. The larger the block size β, the longer the time cost on ENUMs. And then the speedup factor will be closer to the ideal value. At the same time, the quality of the reduced basis is much better and the efficiency of the parallel algorithm is much higher.

The speedup of Algorithm 1 depends on the numbers of total ENUMs and the positions j not satisfying the condition (called special ENUMs). However, for different lattices, the numbers are always different.

Algorithm 1 Parallel BKZ algorithm

Input: Lattice basis $b_1, b_2, \ldots, b_n \in Z^n$, reduced parameter $\delta \in (1/2, 1]$ and block size $\beta \in [2, n)$.

Output: a BKZ reduced basis with parameter (δ, β).

There are p computing units and $p = n$ is the dimension of the lattice in default. $p_1, p_2, \ldots, p_{n-1}$ are computing nodes, and p_0 is control node, determining whether algorithm terminates.

1. p_0 : LLL reduction, namely p_0 : LLL$(b_1, b_2, \ldots, b_n, \delta)$.
2. p_0 : send$(b_1, b_2, \ldots, b_n; p_i (1 \leqslant i \leqslant n-1))$.
3. each node computes as follows:
4. $p_i (1 \leqslant i \leqslant n-1)$:
5. recv$(b_1, b_2, \ldots, b_n; p_0)$;
6. $k = \min(i + \beta - 1, n)$;
7. $b_i^{\text{new}} = \text{ENUM}(i, k)$;
8. **if**$(\delta ||b_i^*|| > ||b_i^{\text{new}^*}||)$ **then** flag_$i = 1$;
9. **else** flag_$i = 0$;
10. **fi**
11. send(flag_$i; p_0$);
12. **end**
13. p_0 :
14. recv(flag_$i; p_i$);
15. **if**(sum(flag_i) == 0) **then** exit;
16. **else**
17. find the next flag_$i = 1$; $h = \min(i + \beta, n)$;
18. recv($\text{b}_i^{\text{new}}; p_i$);
19. LLL$(b_1, \ldots, b_{i-1}, b_i^{\text{new}}, b_i, \ldots, b_h, \delta)$; LLL$(b_1, b_2, \ldots, b_n, \delta)$;
20. send$(b_1, b_2, \ldots, b_n; p_i (1 \leqslant i \leqslant n-1))$; goto step 3;
21. **fi**
22. **end**
23. **return** $b_1, b_2, \ldots, b_n \in Z^n$.

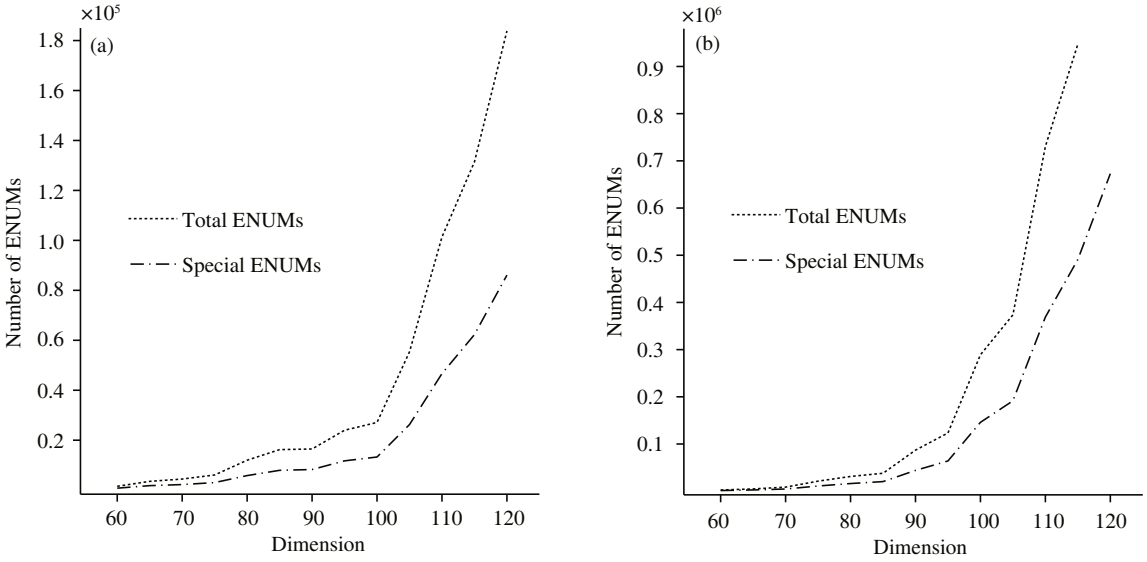

Figure 1 Total and special ENUMs on Goldstein-type lattices with (a) BKZ 24 and (b) BKZ 30.

Even if bases for the same lattice are different, the numbers are also distinct. And now, there is no theory available to predict the numbers as yet. We test the common lattices–Goldstein-type lattices (random lattices in the sense of Goldstein and Mayer [14]) and get some numerical results in Figure 1.

The numerical results show that there are always excessive special positions. For example, the average number of special positions is about half of the total in Goldstein-type lattices. In other words, the

parallel speedup factor is about double for the multiprocessor implementation. Therefore, if we take the parallel implementation of the original BKZ algorithm directly in Algorithm 1, the efficiency is extremely low and we should design new parallel algorithm.

3.2 New BKZ algorithm suitable for parallel implementation

In Algorithm 1, if one position j does not satisfy the conditions, all the values should be computed again, thus severely limiting the parallel efficiency of the algorithm. In fact, we could run LLL algorithm in the whole after all the ENUMs terminate in one tour. We propose a new parallel lattice reduction algorithm in the following.

Algorithm 2 New parallel BKZ algorithm

Input: Lattice basis $b_1, b_2, \ldots, b_n \in Z^n$, reduced parameter $\delta \in (1/2, 1]$ and block size $\beta \in [2, n)$.

Output: a BKZ reduced basis with parameter (δ, β).

There are p computing units and $p = n$ is the dimension of the lattice in default. $p_1, p_2, \ldots, p_{n-1}$ are computing nodes, and p_0 is control node, determining whether algorithm terminates.

1. p_0 : LLL reduction, namely p_0 : LLL$(b_1, b_2, \ldots, b_n, \delta)$.
2. p_0 : send$(b_1, b_2, \ldots, b_n; p_i (1 \leqslant i \leqslant n-1))$.
3. each node computes as follows:
4. $p_i (1 \leqslant i \leqslant n-1)$:
5. recv$(b_1, b_2, \ldots, b_n; p_0)$;
6. $k = \min(i + \beta - 1, n)$;
7. $b_i^{\text{new}} = \text{ENUM}(i, k)$;
8. **if**$(\delta ||b_i^*|| > ||b_i^{\text{new}^*}||)$ **then** flag_$i = 1$;
9. **else** flag_$i = 0$;
10. **fi**
11. send(flag_$i; p_0$);
12. **end**
13. p_0 :
14. recv(flag_$i; p_i$);
15. **if**(sum(flag_i) == 0) **then** exit;
16. **else**
17. find all the flag_$i = 1$;
18. recv$(b_i^{\text{new}}; p_i)$;
19. insert all the b_i^{new} to the basis and obtain $b_1, \ldots, b_{i-1}, b_i^{\text{new}}, b_i, \ldots, b_{j-1}, b_j^{\text{new}}, b_j, \ldots, b_n$;
20. LLL$(b_1, \ldots, b_{i-1}, b_i^{\text{new}}, b_i, \ldots, b_{j-1}, b_j^{\text{new}}, b_j, \ldots, b_n; \delta)$;
21. send$(b_1, b_2, \ldots, b_n; p_i (1 \leqslant i \leqslant n-1))$; goto step3;
22. **fi**
23. **end**
24. **return** $b_1, b_2, \ldots, b_n \in Z^n$.

In Algorithm 2, we no longer deal with the special positions in order but insert the new vectors into the basis after all the ENUMs terminate, which is the essential difference from Algorithm 1. Obviously, Algorithm 2 improves the parallel efficiency. However, compared with Algorithm 1, the correctness and computational complexity cannot be guaranteed. Next we analyze performance of the new parallel algorithm.

Theorem 1. The output basis of Algorithm 2 is BKZ reduced.

Proof. Obviously, a basis is BKZ reduced if and only if it satisfies:

$$\delta ||b_i|| \leqslant ||\text{ENUM}(b_i, b_{i+1}, \ldots, b_{\min\{i+\beta-1, n\}})||,$$

where $i = 1, \ldots, n-1$. From the description of Algorithm 2, the output basis satisfies the above conditions, so it is a BKZ reduced basis.

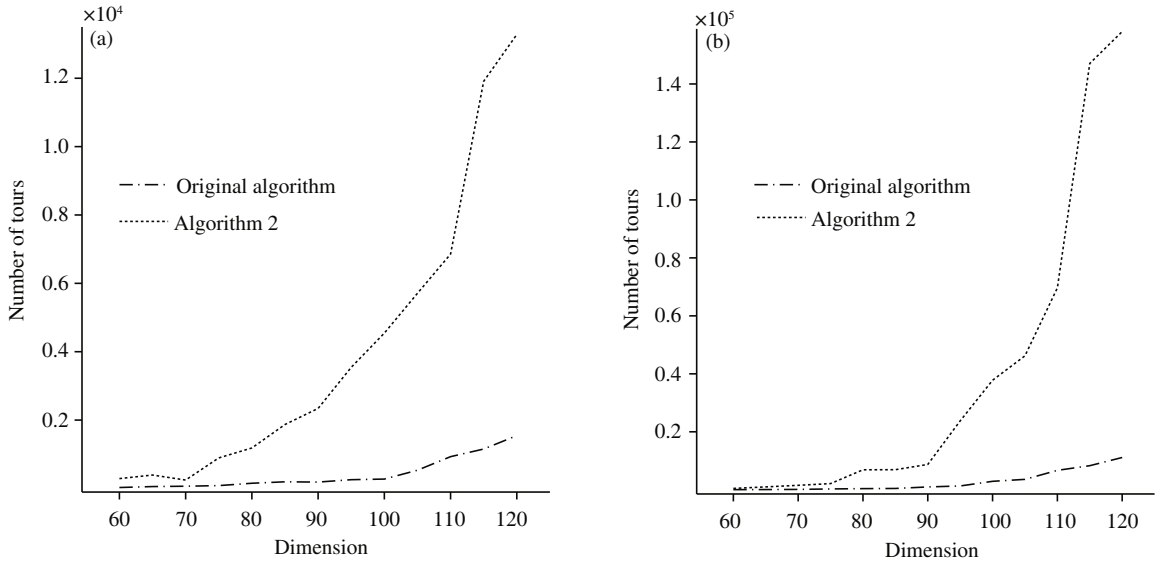

Figure 2 Tour number comparison with (a) BKZ 24 and (b) BKZ 30.

Remark 1. We can not prove that Algorithm 2 runs in polynomial time, but the experimental results show that Algorithm 2 always behaves well.

Remark 2. The reduced basis is not unique for the same lattice, so the output of Algorithm 2 may be different from the original algorithm and the length of the shortest vector also may be distinct.

Remark 3. The number of tours determines the complexity of BKZ algorithm. Algorithm 2 may have much more tours than the original algorithm. The difference of the two algorithms lies in when to run the LLL subroutine. However, there is no certain relation on the output between the two algorithms and we also cannot compare the tours of two algorithms in theory. Therefore, we can prove the correctness of Algorithm 2, but we cannot illustrate the algorithm's efficiency. That is just to say, Algorithm 2 is more suitable for parallel implementation from the viewpoint of parallelism.

We also test Goldstein-type lattices and get some numerical results on the tours and shortest vectors. Figure 2 shows the tour number comparison between original algorithm and Algorithm 2.

Experiments show that Algorithm 2 could often be terminated. Compared with original algorithm, tours in Algorithm 2 always increase. For example, multiples of tour number in Algorithm 2 with BKZ 24 are 16 at most and 3 at least. For BKZ 30, the multiples are 18 at most and 7 at least. Generally, the number is about 10 on average. Although the total computing amount is increasing, Algorithm 2 could be implemented efficiently in parallel. So the increased cost could be acceptable compared with the parallel speedup. In the analysis below, we will show that the original algorithm cannot behave well at high dimensions and large block size, so Algorithm 2 will be much stronger in practice. However, if the lattices and the block sizes are different, the results are often different. In some cases, Algorithm 2 has much more tours than the original algorithm. The former has no time advantage even if in parallel version. Tours are often related to the lattice type and the block size, so it is an important problem how to choose the proper block size for the certain lattices. The numerical results also show that Algorithm 2 often gets a shorter vector. Figure 3 depicts the shortest vector length comparison of the two algorithms with BKZ 24 and BKZ 30.

Similarly, next we analyze the efficiency of Algorithm 2 and suppose that the time costs of LLL algorithm and communication can be ignored. For an n-dimensional lattice L, supposing there are m tours, and r ENUMs in the whole algorithm. Then $r = mn$. Also let A be the time cost of each ENUM. Clearly, the time cost of Algorithm 2 in serial version is rA and the parallel version is mA, so the ideal parallel speedup factor is $k = r/m = n$. Algorithm 2 in parallel is still much effective than the original algorithm. Let the multiple of tour number be K. Then the speedup factor over original algorithm is n/K. For the Goldstein-type lattices, K is about 10 from the experiments and the speedup factor is

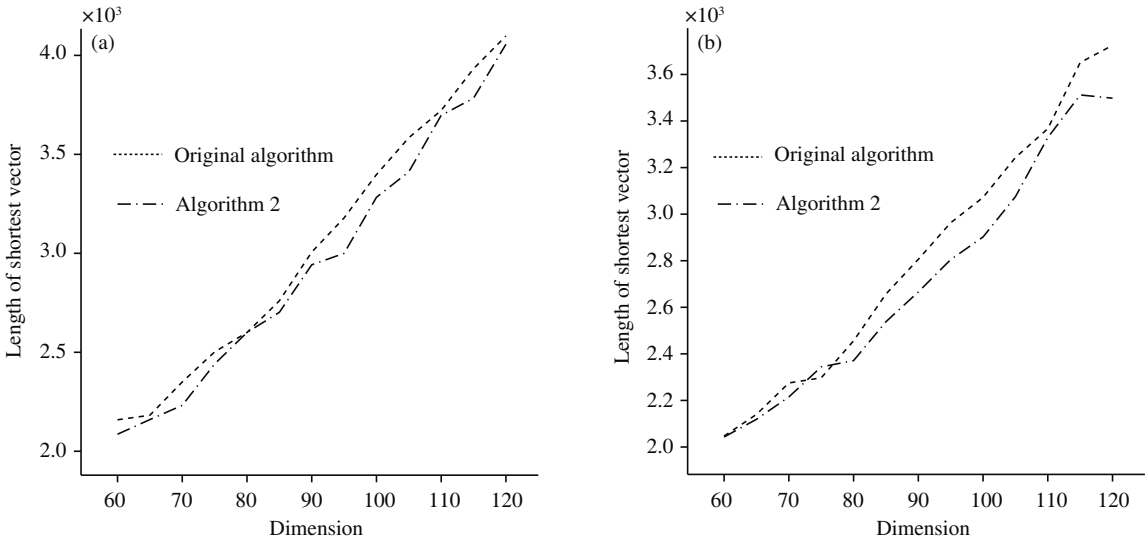

Figure 3 Length comparison with (a) BKZ 24 and (b) BKZ 30.

about $n/10$. In practice, due to the time costs of LLL algorithm and communication, the speedup factor is often not up to the ideal value. But in fact, the speedup factor is related to the block size of the algorithm usually. While the block size is over 40, the time costs of LLL subroutine and communication are much smaller than ENUM subroutine.

4 Experimental results on parallel implementation

Algorithm 2 can get a BKZ reduced basis and also has a good speedup in theory. In this section, we implement Algorithm 2 in Sunway parallel processing system and offer some experimental results.

Sunway system is a general cluster computer with 16 Intel Xeon(R) E5620 2.4GHZ CPUs and every CPU has 4 cores. The operation system is Linux with MPI compiler. We use c++ as the programming language. The experiments are based on Shoup's NTL in version 5.5.2[2)] and the data types and some functions are same as those of NTL. The experimental object is also to examine Goldstein-type lattice which is common in cryptology. Goldstein-type lattices are also the lattices in SVP challenge.

We generate some n-dimensional lattices randomly and set the data length at about $10n$ where n is from 60 to 120. We chose $\delta = 0.99$ and fix the block size β at 24 and 30. For the different dimensions, we ran experiments on a large number of samples, so that an average behavior can be reasonably conjectured. we generate 10 lattices as one group, and for each selected lattice, we generate at least 10 randomly chosen bases. We test the lattice bases with original algorithm and Algorithm 2 both in serial and parallel versions respectively, and then compute the average speedup factor. Figure 4 is the result on time costs of the two algorithms with block size 24 and 30. Figure 5 is the result on speedup factors of the algorithms.

As is evident in the figures, Algorithm 2 in parallel consumes the least time, followed by the original algorithm and Algorithm 2 in serial is most time-consuming. In fact, the computational complexity is same for Algorithm 2 in serial and in parallel which is larger than the original algorithm. However, Algorithm 2 is suitable for parallel implementation, so the time cost of Algorithm 2 in parallel is less than the original algorithm.

In Figure 5 the speedup factors are stable and they will be enlarged slowly with the increase of dimension. But the tendency is not strict absolutely. For example, the parallel speedup factors are between 3 and 12 for the block size 24. when the dimension is 70, the factor is much larger. For the block size 30, the factors are between 14 and 40, and dimension 90 is the exception. The speedup factors over

2) Shoup V. Number theory library(NTL) for c++. http://www.shoup.net/ntl/.

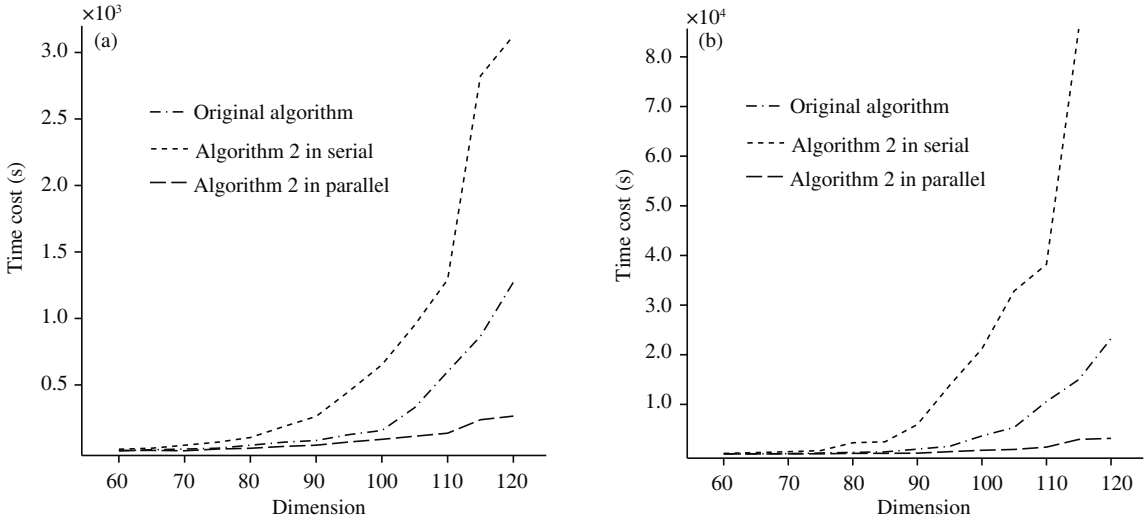

Figure 4 Time cost with (a) BKZ 24 and (b) BKZ 30.

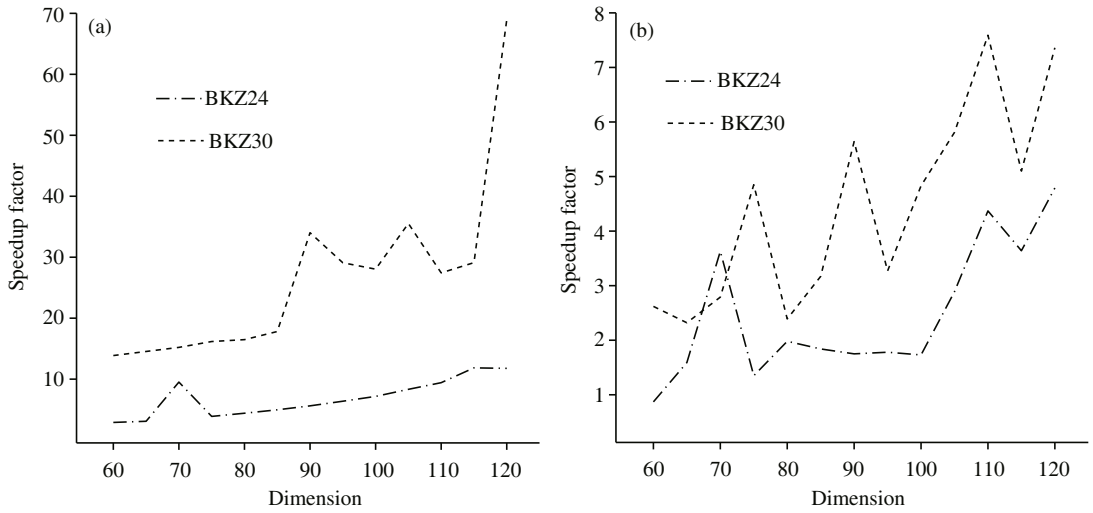

Figure 5 (a) Parallel speedup factor of Algorithm 2; (b) speedup factor over original algorithm.

original algorithm also have the similar phenomenon. The factors are between 1 and 5 for block size 24. When the block size is 30, they are between 3 and 8.

At the same time, it is shown that the speedup effect is much more obvious if the block size is large enough. Algorithm 2 in parallel performs well for the block size 30, but much weakly for the block size 24. In order to verify this result, we generate some 100-dimensional lattices randomly and test them with block size β from 20 to 40. Figure 6(a) gives the time cost and Figure 6(b) gives the speedup factors for every block size. In the figure, we just test Algorithm 2 in serial up to block size 30 and original algorithm up to block size 35 because the the time costs for large block size are beyond reach.

Algorithm 2 is more suitable for parallel realization than the original algorithm. Moreover, with the increasing block size, time cost of ENUMs will be much longer and the parallel speedup will be more effective. Although the computation complexity increases, for the high-dimensional lattices, we always require the corresponding block size must be large, and we can predict that Algorithm 2 in parallel will perform well. However, it also should be noted that the parallel efficiency is not good for the small block sizes. For example, when the dimension is 60, the time cost of Algorithm 2 is the same as original algorithm for a block size 24. In other words, Algorithm 2 is suitable for the situation of high-dimensional lattices and large block sizes.

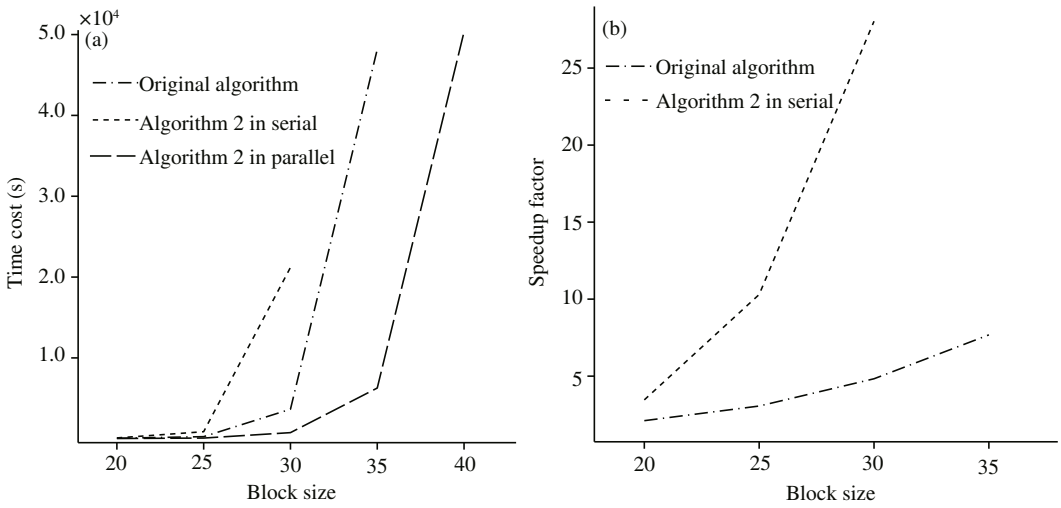

Figure 6 (a) Time cost and (b) speedup factor of the algorithms.

5 Conclusion

In recent years, lattice reduction has been becoming a powerful tool in the area of cryptanalysis and the lattice reduction algorithms have also received increasing attentions. Parallelization is an important way to improve the efficiency of algorithms, and we study the parallel scheme of BKZ algorithm on the multiprocessor computer architecture in this paper. We analyze the efficiency of original algorithm's parallel implementation and make some tests. The results show that original algorithm is not suitable for parallel implementation. Then we propose a new parallel algorithm. The theoretical analysis shows that new algorithm can get a BKZ reduced basis and the new algorithm has a good speedup.

Although the computational complexity increases compared with the original algorithm, the experimental results show that the time cost of the new algorithm in parallel is much less and the vector of the reduced basis is often shorter. However, the efficiency of lattice reduction algorithms is closely related to the concrete lattice bases. The algorithm proposed in this paper is also the same. If the styles of the input lattice or the bases are different, the effect of the parallel implementation of the new algorithm is distinct, which is also the research focus of this area in our next work.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No. 61003291), and Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing (Grant No. 2013A03).

References

1 Grotschel M, Lovasz L, Schrijver A. Geometric Algorithm and Combinatorial Optimization. Berlin: Springer-Verlag, 1993

2 Lenstra A K, Lenstra H W, Lovasz L. Factoring polynomials with rational coefficients. Mathematiche Annalen, 1982, 261: 515–534

3 Coppersmith D. Finding a small root of a univariate modular equation. In: Proceedings of International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, 1996. 155–165

4 Han L D, Wang X Y, Xu G W. On an attack on RSA with small CRT-exponents. Sci China Inf Sci, 2010, 53: 1511–1518

5 Santanu S. Some results on cryptanalysis of RSA and factorization. Dissertation for Ph.D. Degree. Kolkata: Indian Statistical Institute, 2011

6 Kumar R S, Narasimam C, Setty S P. Lattice based tools in cryptanalysis for public key cryptography. Int J Netw Secur Appl, 2012, 4: 155–162

7 Schnorr C P. Block reduced lattice bases and successive minima. Comb Probab Comput, 1994, 3: 507–522
8 Joux A. A tutorial on high performance computing applied to cryptanalysis. In: Proceedings of 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, 2012. 1–7
9 Backes W, Wetzel S. Improving the parallel Schnorr-Euchner LLL algorithm. In: Proceedings of 11th International Conference, ICA3PP, Melbourne, 2011. 27–39
10 Dagdelen O, Schneider M. Parallel enumeration of shortest lattice vectors. In: Proceedings of 16th International Euro-Par Conference, Ischia, 2010. 211–222
11 Nguyen P Q, Valle B. The LLL Algorithm: Survey and Applications. 1st ed. Berlin: Springer, 2009. 19–71
12 Schnorr C P, Euchner M. Lattice basis reduction: improved practical algorithms and sovling subset sum problems. Math Program, 1994, 66: 181–199
13 Hanrot G, Pujol X, Stehle D. Analyzing blockwise lattice algorithms using dynamical systems. In: Proceedings of 31st Annual Cryptology Conference, Santa Barbara, 2011. 447–464
14 Nguyen P Q, Stehle D. LLL on the average. In: Proceedings of 7th International Symposium, ANTS-VII, Berlin, 2006. 238–256