Parallel and Distributed Boolean Gröbner Bases Computation in SageMath

Akira Nagai
nagai.akira@lab.ntt.co.jp
NTT Secure Platform Laboratories
Japan

Yosuke Sato
ysato@rs.kagu.tus.ac.jp
Tokyo University of Science
Japan

Abstract

We introduce a parallel and distributed computation method of Boolean Gröbner bases, which are implemented in the computer algebra system SageMath using the PolyBoRi library. We present an easy way to parallelize Boolean Gröbner bases computation using a decorator for parallel computation supported in SageMath, we also present a way of distributed computation using a multiprocessing module in Python. Our software achieves satisfactory speed-up comparing to our sequential implementation in SageMath.

1 Introduction

A residue class ring $\mathbf{B}[X_1, \ldots, X_n]/\langle X_1^2 + X_1, \ldots, X_n^2 + X_n\rangle$ over a Boolean ring $\mathbf{B}$ is called a *Boolean polynomial ring*. A Gröbner basis in a Boolean polynomial ring is called a *Boolean Gröbner basis*, which is first introduced in [10] together with its computation algorithm. The Galois field $\mathbb{F}_2$ is the simplest Boolean ring. Since it is actually a field, a Boolean Gröbner basis in the Boolean polynomial ring $\mathbb{F}_2[X_1, \ldots, X_n]/\langle X_1^2 + X_1, \ldots, X_n^2 + X_n\rangle$ can be computed with no novel theoretical advances. We can easily compute such a Boolean Gröbner basis in most computer algebra systems with a facility to compute Gröbner bases in a polynomial ring over $\mathbb{F}_2$. When the Boolean ring $\mathbf{B}$ is a power set algebra, a Boolean Gröbner basis is of great importance for solving certain types of combinatorial problems.

By the technique introduced in [7], we can now compute Boolean Gröbner bases of an arbitrary finite power set algebra by the computation of Boolean Gröbner bases of a polynomial ring over $\mathbb{F}_2$. This method is implemented in the computer algebra system Risa/Asir [4]. It is further optimized in [7] for developing a much faster program in the computer algebra system SageMath using the PolyBoRi library [8]. The experimental data presented in [8] suggest that SageMath is faster and more optimal software to compute Boolean Gröbner bases than other computer algebra systems such as Risa/Asir, Singlar, Mathematica or Maple. Even by sequential computation, the program introduced in [8] achieves tremendous speed-up comparing to their previous implementation in Risa/Asir of [4]. It enables us to have a first-ever real time solver of Sudoku puzzles by the computation of Gröbner bases. For solving one Sudoku puzzle, we need to compute at most 10 Boolean Gröbner bases. Our sequential program can handle

them in a few seconds, we do not need any parallel computation of Boolean Gröbner bases for solving one Sudoku puzzle.

In order to compute a s-rank of a Sudoku puzzle which is a purely mathematical index introduced in [5] to present its level of difficulty, we need to compute much more Boolean Gröbner bases. For some Sudoku puzzles, a sequential program needs computation time beyond several minutes. We need parallel and distributed computation of Boolean Gröbner bases. In this paper we use the word "parallel computation" for parallel computation by one computer with a multi-core processor, "distributed computation" for distributed computation by several computers connected on the Internet. So, "parallel and distributed computation" means parallel computation using several connected computers.

In this paper we introduce our parallel and distributed computation method of Boolean Gröbner bases of a Boolean polynomial ring over a power set algebra, which is implemented in the computer algebra system SageMath using the PolyBoRi library [1]. We parallelize a Boolean Gröbner bases program introduced in [8]. Note that a parallel computation program of Boolean Gröbner bases of a power set algebra is already implemented in [12, 11] using a parallel logic programming language KLIC. Unfortunately, their program does not achieve enough speed-up, our program achieves satisfactory speed-up.

The paper is organized as follows. In Section 2 and 3, we give a quick review of a Boolean ring of a power set algebra and the computation method of Boolean Gröbner bases of a power set algebra introduced in [7]. In Section 4 and 5, we describe parallel computation and distributed computation of Boolean Gröbner bases. Section 6 contains some data we have obtained through our computation experiments using our program.
The reader is referred to [14] for a comprehensive description of Boolean polynomial rings and Boolean Gröbner bases, also to [5] for more detailed description of the application of Boolean Gröbner bases to Sudoku puzzles.

We also put our prototype program as an open software at the following URL:

http://www.mi.kagu.tus.ac.jp/~nagai/BoolGB_Sage/

2 Boolean ring of a power set algebra

In this section, we review Boolean ring of a power set algebra.

Definition 1 A commutative ring $\mathbf{B}$ with an identity 1 is called *a Boolean ring* if every element a of $\mathbf{B}$ is idempotent, i.e., $a^2 = a$.

$(\mathbf{B},\vee,\wedge,\neg)$ becomes a Boolean algebra with the Boolean operations $\vee,\wedge,\neg$ defined by $a \vee b = a + b + a \cdot b, a \wedge b = a \cdot b, \neg a = 1 + a$. Conversely, for a Boolean algebra $(\mathbf{B},\vee,\wedge,\neg)$, if we define $+$ and $\cdot$ by $a + b = (\neg a \wedge b) \vee (a \wedge \neg b)$ and $a \cdot b = a \wedge b$, $(\mathbf{B},+,\cdot)$ becomes a Boolean ring. Note that $+$ is nothing but an exclusive OR operator. Note also that $-a = a$.

Definition 2 Let S be an arbitrary set and $\mathcal{P}(S)$ be its power set, i.e., the family of all subsets of S. Then, $(\mathcal{P}(S),\vee,\wedge,\neg)$ becomes a Boolean algebra with the operations $\vee,\wedge,\neg$ as union, intersection and the complement of S respectively. It is called a *power set algebra* of S.

Definition 3 Let $\mathbf{B}$ be a Boolean ring. A residue class ring $\mathbf{B}[X_1, \ldots, X_n]/\langle X_1^2 + X_1, \ldots, X_n^2 + X_n\rangle$ modulo an ideal $\langle X_1^2 + X_1, \ldots, X_n^2 + X_n\rangle$ becomes a Boolean ring. It is called *a Boolean polynomial ring* and denoted by $\mathbf{B}(X_1, \ldots, X_n)$, its element is called *a Boolean polynomial*.

Note that a Boolean polynomial of $\mathbf{B}(X_1, \ldots, X_n)$ is uniquely represented by a polynomial of $\mathbf{B}[X_1, \ldots, X_n]$ that has at most degree 1 for each variable X_i.
In what follows, we identify a Boolean polynomial with such a representation.
Multiple variables such as $X_1, \ldots, X_n$ or $Y_1, \ldots, Y_m$ are abbreviated to $\bar{X}$ or $\bar{Y}$ respectively. Lower small roman letters such as a, b, c are usually used for elements of a Boolean ring $\mathbf{B}$. The symbol $\bar{a}$ denotes an m-tuple of elements of $\mathbf{B}$ for some m.

3 Computation of Boolean Gröbner bases of a finite power set algebra

In this section we review the computation theory of Boolean Gröbner bases.

Let S be a finite set and k be its cardinality. Then the Boolean ring $\mathbf{B}$ of the power set algebra $\mathcal{P}(S)$ is isomorphic to the direct product $\mathbb{F}_2^k$. More precisely, let $S = \{a_1, a_2, \ldots, a_k\}$ then the isomorphism θ from $\mathcal{P}(S)$ to $\mathbb{F}_2^k$ is defined by $\theta(A) = (e_1, e_2, \ldots, e_k)$ for each $A \subseteq S$, where $e_i = 1$ if $a_i \in A$ and $e_i = 0$ if $a_i \notin A$ for each $i = 1, \ldots, k$.

For an element $v \in \mathbb{F}_2^k$, $\pi_i(v)$ denotes the i-th component of v. This projection is naturally extended to a Boolean polynomial of $\mathbb{F}_2^k(\bar{X})$. The following theorem reduces the computation of a Boolean Gröbner basis of a Boolean polynomial ring $\mathbb{F}_2^k(\bar{X})$ to the computation of Boolean Gröbner bases of $\mathbb{F}_2(\bar{X})$.

Theorem 4 In a Boolean polynomial ring $\mathbb{F}_2^k(\bar{X})$, let G be a finite set of Boolean closed polynomials. Then, G is a (reduced) Boolean Gröbner basis of an ideal I in $\mathbb{F}_2^k(\bar{X})$ if and only if $\pi_i(G) = \{\pi_i(g) | g \in G\} \setminus \{0\}$ is a (reduced) Gröbner basis of the ideal $\pi_i(I) = \{\pi_i(f) | f \in I\}$ in $\mathbb{F}_2(\bar{X})$ for each $i = 1, \ldots, k$.

For each $i = 1, \ldots, k$, define a map ϕ_i from $\mathbb{F}_2$ to $\mathbb{F}_2^k$ by $\phi_i(0) = (0, \ldots, 0)$ and $\phi_i(1) = (e_1, \ldots, e_k)$ where $e_i = 1$ and $e_j = 0$ for any j such that $j \neq i$. It is also naturally extended to a map from $\mathbb{F}_2(\bar{X})$ to $\mathbb{F}_2^k(\bar{X})$.

Algorithm: Boolean GB
input: F a finite subset of $\mathbb{F}_2^k(\bar{X})$ and a term order $>$ on $T(\bar{X})$
output: G a reduced Boolean Gröbner basis of $\langle F\rangle$ w.r.t. $>$
For each $i = 1, \ldots, k$ compute the reduced Boolean Gröbner basis G_i of the ideal $\langle \pi_i(F)\rangle$ in $\mathbb{F}_2(\bar{X})$. Set $G = \cup_{i=1}^k \phi_i(G_i)$.

Example 5 Let $\mathbf{B} = \mathbb{F}_2 \times \mathbb{F}_2$. In a polynomial ring $\mathbf{B}(X)$, $(1,0)X, (0,1)X$ and $(1,1)X$ are both reduced Gröbner bases of the same ideal.

In order to have a unique Gröbner basis, we need the following manipulation.

Algorithm: Stratification
input: G a reduced Boolean Gröbner basis in $\mathbb{F}_2^k(\bar{X})$

output: G' a stratified Boolean Gröbner basis
Let $\{t_1, \ldots, t_s\}$ be the set of all leading terms of some polynomial in G. For each $i = 1 \ldots, s$, let $g_i = \sum_{LT(g)=t_i, g \in G} g$. Set $G' = \{g_1, \ldots, g_s\}$.

In the above example, $(1,1)X$ is the stratified Gröbner basis, but the other is not. We conclude this section with a simple example of our method.

Example 6 We show a calculation process of the Boolean Gröbner basis of the following F.

$$F = \left\{ \begin{array}{l} (\{e_1, e_2\} + 1) * X * Y + \{e_1\} * X + Y + \{e_2\} \\ X * Y + \{e_1\} * Y + X + \{e_1, e_2\} \end{array} \right.$$

We compute $\pi_i(F)$ as follows.

$$\pi_1(F) = \left\{ \begin{array}{l} X + Y \\ X * Y + Y + X + 1 \end{array} \right. \quad \pi_2(F) = \left\{ \begin{array}{l} Y + 1 \\ X * Y + X + 1 \end{array} \right. \quad \pi_3(F) = \left\{ \begin{array}{l} X * Y + Y \\ X * Y + X \end{array} \right.$$

We compute the reduced Boolean Gröbner basis G_i of the ideal $\langle \pi_i(F) \rangle$ in $\mathbb{F}_2(X, Y)$.

$$G_1 = \left\{ \begin{array}{l} X + 1 \\ Y + 1 \end{array} \right. \quad G_2 = \{1\} \quad G_3 = \{X + Y\}$$

We compute $\phi_i(G_i)$ as follows.

$$\phi_1(G_1) = \left\{ \begin{array}{l} \{e_1\} * X + \{e_1\} \\ \{e_1\} * Y + \{e_1\} \end{array} \right. \quad \phi_2(G_2) = \{\{e_2\}\} \quad \phi_3(G_3) = \{(\{e_1, e_2\} + 1) * (X + Y)\}$$

Finally, we do Stratification in order to get a stratified Boolean Gröbner basis.

$$G' = \left\{ \begin{array}{l} (\{e_2\} + 1) * X + (\{e_1, e_2\} + 1) * Y + \{e_1\} \\ \{e_1\} * Y + \{e_1\} \\ \{e_2\} \end{array} \right.$$

4 Parallel Boolean Gröbner bases

In this section we describe a parallelization method for Boolean Gröbner basis computation and its implementation in SageMath. In the previous section we see that each computation of $\langle \pi_i(F) \rangle$ for $i = 1, \ldots, k$ is done independently. Hence, the computation of the reduced Gröbner basis G_i of the ideal $\langle \pi_i(F) \rangle$ in $\mathbb{F}_2(\bar{X})$ is also done independently. Therefore we can easily have an algorithm for parallel Boolean Gröbner basis computation. It is also easy to implement a parallel Boolean Gröbner basis computation by using a decorator which gives a function of a parallel interface. This decorator represented by "@parallel" is supported in SageMath. Our program to compute a Boolean Gröbner basis of a finite power set algebra $\mathcal{P}(\{s_1, \ldots, s_k\})$ has the following rather simple shape.

```
@parallel()
def parallel_gb(In):
  I = ideal(In[1])
```

```
  BG=I.groebner_basis(heuristic=False)
  return [In[0],BG]

def parallel_bgb(Polys,Vars,Eles):
  B = BooleanPolynomialRing(len(Vars)+len(Eles), Vars+Eles, order='lex')
  BPolys=(B.ideal(Polys)).gens()
  BVars=(B.ideal(Vars)).gens()
  BEles=(B.ideal(Eles)).gens()
  Polys_set=divide(BPolys,BVars,BEles)
  Input=[[i,Polys_set[i]] for i in range(len(Polys_set))]
  Output=list(parallel_gb(Input))
  Bgb_set=[]
  Bgb_set=[Bgb_set+Output[j][1][1] for i in range(len(Output))
    for j in range(len(Output))if Output[j][1][0]==i]
  Ele_BPolys=mulatom(Bgb_set,BEles)
  Bgb=stratify(Ele_BPolys,BVars)
  return Bgb
```

`BooleanPolynomialRing` is a PolyBoRi command which defines a polynomial ring $\mathbb{F}_2(\bar{X}, s_1, \ldots, s_k)$. For the input F of `Polys`, `divide` computes $\pi_i(F)$ for each $i = 1, \ldots, k$. `parallel_gb` computes a reduced Gröbner basis G_i of the ideal $\langle \pi_i(F) \rangle$ in $\mathbb{F}_2(\bar{X})$ for each $i = 1, \ldots, k$ in parallel. After that, an output of `parallel_gb` stored in an array is sorted. `mulatom` is a program to compute $\phi_i(G_i)$. Finally `stratify` compute the stratified Boolean Gröbner basis G'.

We give a snapshot of timing data of the following example of parallel Boolean Gröbner basis computations. In the example, we have 40 variables $x_1, \cdots, x_{40}$. The symbols $e_1, e_2, \cdots, e_{10}$ denote strings, so $\{e_1\}$, $\{e_2\}$, $\cdots$ are elements of our $\mathbf{B}$.

Example 7 $F = \{x_8x_{40}+x_{11}x_{15}+x_{13}x_{30}+x_{23}x_{27}, x_{19}+x_{26}x_{38}, \{e_1\}x_{14}x_{40}+x_{15}x_2+x_1, \{e_7\}x_{26}+\{e_{10}\}x_{37}+\{e_7\}, x_8x_{23}+x_{11}x_{39}+x_{16}x_{18}, x_{25}x_{27}+\{e_4\}x_{35}, \{e_5\}x_{12}+x_{33}x_4+x_{17}x_6+x_{33}, \{e_4\}x_{10}+x_{22}x_{24}+x_{12}x_3, \{e_1\}x_{27}+\{e_6\}x_{34}, \{e_6\}x_{14}x_2+x_{20}x_{28}+x_{27}x_{38}+x_{31}x_{38}+e_9, \{e_1\}x_{18}x_{37}+x_{12}x_{16}+x_1x_{20}+x_{22}, x_4x_{13}+x_5, x_{10}x_{14}x_{32}, \{e_6\}x_{16}x_{22}, \{e_1\}x_{22}x_{37}+\{e_8\}x_{25}, \{e_3\}x_{10}+x_{12}x_{30}, \{e_4\}x_{15}+\{e_9\}x_{22}+x_{25}, \{e_5\}x_{35}+x_{12}+x_7, \{e_2\}x_2x_{29}+\{e_7\}x_{21}x_{33}+\{e_8\}x_{36}x_9, x_1+x_8+x_{15}+x_{16}+x_{22}, \{e_3\}x_7+\{e_3\}, \{e_8\}x_{28}+\{e_8\}\}$

The computation is done by the computer with OS: Ubuntu 14.04 LTS 64bit, Software: SageMath 7.1, CPU: Intel(R) Core(TM) i7-3970X, Clock: 3.50GHz, Number of Cores: 6, Memory: 64GB. Total computation time is 16.7 seconds using parallel Boolean Gröbner basis computation. Whereas, sequential computation time is 70 seconds.

```
sage: load("bgb.sage")
sage: %time B=parallel_bgb(Polys_ex6,Vars_ex6,Eles_ex6)
For the element 3 GB Computation time  2.95557999611
For the element 4 GB Computation time  4.66137695312
For the element 8 GB Computation time  4.94551992416
For the element 1 GB Computation time  7.18508601189
For the element 9 GB Computation time  9.36309504509
```

```
For the element 10 GB Computation time  10.1585040092
For the element 2 GB Computation time  11.1377859116
For the element 7 GB Computation time  11.1130139828
For the element 11 GB Computation time  11.1987161636
For the element 5 GB Computation time  13.8662071228
For the element 6 GB Computation time  14.5633881092
CPU times: user 4.82 s, sys: 153 ms, total: 4.97 s
Wall time: 16.7 s
```

This parallel computation is essentially same as the parallel computation introduced in [12, 11]. Though we have satisfactory speed-up for this example, there are two problems for this type of parallel computation. One is that we can expect only $n + 1$-times speed-up, where n is the number of elements. The another one is that, unless each computation has almost same computation time, we can not expect enough speed-up. For example, if two computations need 5 minutes and other need only a few seconds then the speed-up can be only double. Therefore the above example is a good example of the speed-up, but in general it is difficult to obtain an equivalent speed-up to the above example.

As is described in the introduction, we need to compute at most 10 Boolean Gröbner bases for solving one Sudoku puzzle. Since each Boolean Gröbner bases computation is done within 1 seconds, we do not need parallel computation for the solver of a Sudoku puzzle. For the computation of the s-rank of a Sudoku puzzle, we need to compute much more Boolean Gröbner bases. Since they are divided into many independent computations, parallel and distributed computation of Boolean Gröbner bases is effective for its computation.

5 Distributed Boolean Gröbner bases

In this section we describe how we implemented distributed Boolean Gröbner basis computation in SageMath. We have implemented distributed computation using "multiprocessing" module in Python which provide the remote manager. Client code is as follows.

```
from multiprocessing.managers import BaseManager
class QueueManager(BaseManager):pass
QueueManager.register('bgb_remote')
QueueManager.register('parallel_bgb_remote')

def distributed_bgb(Polys,Vars,Eles,IP,Port,Parallel=False):
  B = BooleanPolynomialRing(len(Vars)+len(Eles), Vars+Eles, order='lex')
  m = QueueManager(address=(IP,Port),authkey='abracadabra')
  m.connect()
  if Parallel==True:
    Bgb = (m.parallel_bgb_remote(Polys,Vars,Eles))._getvalue()
  else:
    Bgb = (m.bgb_remote(Polys,Vars,Eles))._getvalue()
  return Bgb
```

Clients setup IP address and Port number of a distributed computing server. Basically, clients set any password to "authkey" for the authentication when "BaseManager" object is used. We fix "authkey" here for simplicity. Next, server code is as follows.

```
from multiprocessing.managers import BaseManager
class QueueManager(BaseManager):pass

def _bgb(polys,vars,eles):
  print("bgb call ")
  w=walltime()
  Bgb=bgb(polys,vars,eles)
  wtime=walltime(w)
  print '\033[94m'+"BGB Computation time "+'\033[0m',wtime
  return Bgb

def _parallel_bgb(polys,vars,eles):
  print("parallel bgb call ")
  return parallel_bgb(polys,vars,eles)

if __name__ == '__main__':
  load("bgb.sage")
  QueueManager.register('bgb_remote', callable=_bgb)
  QueueManager.register('parallel_bgb_remote', callable=_parallel_bgb)
  m = QueueManager(address=("127.0.0.1",50000),authkey='abracadabra')
  s = m.get_server()
  s.serve_forever()
```

When client calls `bgb_remote` function, server executes `_bgb` function. Although there are other implementation methods to distribute, this implementation method is very simple and easy to deal with.

The following computation example shows how to use our program. It compute the stratified Boolean Gröbner basis $\{x+\{s_1\}, y+\{s_2\}\}$ of the ideal $\langle(1+\{s_1,s_2\})(XY+X+Y), \{s_1\}X+\{s_1\}, \{s_2\}Y+\{s_2\}, XY\rangle$ in a Boolean polynomial ring $\mathcal{P}(\{s_1,s_2\})(x,y)$ w.r.t. a lex order such that $x > y$.

```
sage: load("bgb.sage")
sage: var("x,y,s1,s2")
(x, y, s1, s2)
sage: distributed_bgb([(1+s1+s2)*(x*y+x+y),s1*x+s1,s2*y+s2,x*y],[x,y],[s1,s2],
....: "127.0.0.1",50000,Parallel=True)
[x + s1, y + s2]
```

Parallel computation is available as an option. In this way, we can easily use distributed Boolean Gröbner basis computation.

6 Computation Data

In this section we give some computation data of the s-rank of Sudoku puzzles obtained by our parallel and distributed Boolean Gröbner basis program in SageMath in order to show its effect. The computing environment used for our experiments is summarized below.

	Computer 1 and 2	Computer 3
OS	Ubuntu 14.04 LTS 64bit	Ubuntu 14.04 LTS 64bit
SageMath	7.1	7.1
CPU Clock	Intel(R) Core(TM) i7-3970X 3.50GHz	Intel(R) Core(TM) i7-4960X 3.60GHz
Num of Cores	6	6
Memory	64GB	64GB

Table 1: The computing environment

For the computation of s-ranks, we empirically compute more than 100 Boolean Gröbner bases. These Boolean Gröbner basis computations are independent, so we can expect reasonable speed-up by using parallel and distributed Boolean Gröbner basis computation.

The following Table 2 contains computation time (in seconds) obtained by three types of computation. The first one, "Sequential" is a serial computing, which means all Boolean Gröbner basis computations are executed sequentially. The second one, "Parallel" means Boolean Gröbner basis computations executed simultaneously by a single computer with multiple processors/cores. We run a Boolean Gröbner bases program introduced in [8] as regards "Sequential" and "Parallel". The third one, "Parallel and Distributed" means "Parallel" computations by three computers. The first row in Table 2 is the average time (in seconds) of 10 puzzles in the Sudoku book UltraHard [2] for obtaining s-rank of a puzzle. Example A and Example B contains the data for two puzzles among them.

	Sequential	Parallel	Parallel and Distributed
Average	74.13	15.55	6.49
Example A	184.9	37.32	14.39
Example B	64.12	14.04	5.66

Table 2: Computation time of Srank (Sec)

2						9		
		5	1				6	
	6			3				4
	1			9				2
		4	5					7
					8		4	
7						4		
	5				6		2	
		1	2	7				3

Example A

2						9	8	
	3				1			6
		9		5				4
			6				3	
		3				1		
	9				4			
9				7		8		
4			5				9	
	6	2						5

Example B

The computation time of Example A and B shows that our parallel and distributed computation is efficient for s-rank. When a given Sudoku puzzle is not basic solvable (see [5]), for computing its s-rank, we need computations of many Boolean Gröbner bases. For such computations, our computing method is practical and useful.

7 Conclusions and Remarks

In this paper we have introduced parallel computation of Boolean Gröbner bases of an arbitrary finite power set algebra. It is implemented in the computer algebra system SageMath using the PolyBoRi library. A parallel Boolean Gröbner basis computation program is implemented with "@parallel" decorator. We do not use any sophisticated technique of parallel programming. Nevertheless, our method is sufficiently effective as we saw in section 4.

As is described in the introduction, our parallel Boolean Gröbner basis computation does not have an enough effect on just solving one Sudoku puzzle. For solving a Sudoku puzzle which has a size 16 $\times$ 16, however, we think parallel and distributed Boolean Gröbner basis computations are useful. For solving a normal Sudoku puzzle of the size 9 $\times$ 9, we need about 800 Boolean polynomials with 81 variables, whereas for solving a Sudoku puzzle of the size 16 $\times$ 16, we need about 3700 Boolean polynomials with 196 variables.

We have also introduced the distributed computation of Boolean Gröbner bases. Data of our experiments presented in section 6 show that our programs work properly and effectively. Our parallel and distributed program in SageMath(PolyBoRi) can obtain s-rank within 10 seconds.

We can also parallelize the computation of a comprehensive Boolean Gröbner basis by using the same parallel computation presented in section 4 (see [3]). A comprehensive Boolean Gröbner basis is an ideal tool for obtaining an elimination of an ideal of a Boolean polynomial (see [13]). Computation of such an elimination is very important for many types of combinatorial problems. We expect that our parallel and distributed computation method of Boolean Gröbner bases also contributes to speed-up solving those problems.

References

[1] Brickenstein, M., Dreyer, A., 2009. A framework for Gröbner -basis computations with Boolean polynomials. J. Symbolic Comput.44 (9), 1326-1345. PolyBoRi Polynomials over Boolean Rings.
http://polybori.sourceforge.net/.

[2] Gohnai,K., and Cross Word editorial desk.(2008). Number Placement Puzzles(UltraHard), (In Japanese) Kosaido Publishing Co., 2008.

[3] Inoue, S.(2009). On the Computation of Comprehensive Boolean Gröbner Bases. Proceedings of the 11th International Workshop on Computer Algebra in Scientific Computing(CASC 2009), LNCS 5743, pp 130-141, Springer-Verlag Berlin Heidelberg.

[4] Inoue, S., 2009. BGSet Boolean Groebner bases for Sets.
http://www.mi.kagu.tus.ac.jp/~inoue/BGSet/.

[5] Inoue, S and Sato, Y. A Mathematical Hierarchy of Sudoku Puzzles and its Computation by Boolean Gröbner Bases. Proceedings of 12th International Conference, AISC2014, pp 88-98 LNAI 8884, 2014.

[6] Inoue, S and Nagai, A. On the Implementation of Boolean Gröbner Bases. Proceedings of the 9th Asian Symposium on Computer Mathematics(ASCM2009), LNCS, pp.87-92, Springer 2014

[7] Nagai, A and Inoue, S.(2014). An Implementation Method of Boolean Gröbner Bases and Comprehensive Boolean Gröbner Bases on General Computer Algebra Systems. Proceedings of ICMS2014, pp 531-536, Springer LNCS 8592, 2014.

[8] Nagai, A and Yosuke, S.(2015). An efficient implementation of Boolean Gröbner Bases of a power set algebra. Proceedings of ATCM2015, pp 326-335,

[9] Noro, M. et al. (2009). A Computer Algebra System Risa/Asir. http://www.math.kobe-u.ac.jp/Asir/asir.html.

[10] Sakai,K. and Sato, Y. and Menju, S. (1991). Boolean Gröbner bases(revised). ICOT Technical Report 613.

[11] Sato,Y.(1998). Set Constraint Solvers(KLIC Version). http://www.jipdec.or.jp/archives/icot/ARCHIVE/Museum/FUNDING/funding-98-E.html

[12] Sato, Y and Akira, S.(1999). Parallel Computation of Boolean Gröbner Bases. Electronic Proceedings of ATCM1999. http://epatcm.any2any.us/10thAnniversaryCD/EP/1999/contributed_papers.html

[13] Sato, Y., Nagai, A. and Inoue, S.(2008). On the computation of elimination ideals of boolean polynomial rings. In: LNCN, vol. 5081. Springer, pp. 334-348.

[14] Sato, Y. et al.(2011). Boolean Gröbner bases. J. Symbolic Comput.46 (2011), 622-632.