On Solving Sparse Algebraic Equations over Finite Fields II

Igor Semaev

Department of Informatics, University of Bergen, Norway

ACCT, 20.06.2008

Outline

- Motivation
- Sparse equation systems over finite fields
- Known approaches
- Gluing and Agreeing Procedures
- Solve with Gluing Algorithm
- Solve with Agreeing-Gluing Algorithms
- Asymptotical estimates
- Conclusions

Motivation

- One way function $x \longrightarrow f(x)$
- Easy to compute and hard to invert
- Examples
  1. $x \longrightarrow a^x \mod p$
  2. M - plain-text,
     K - key,
     $E_K(M)$ cipher-text in the AES:
     $$K \longrightarrow E_K(M)$$
- Still one-way

Motivation

- **To Compute:** Represent f in small number of small gates
- E.g.
$$f(x_1, x_2, x_3, x_4) = F(g_1(x_1, x_2), g_2(x_2, x_3), g_3(x_3, x_4))$$
- **To Invert:** Given y solve $f(x) = y$ in x
- Introduce new variables to simplify equations
- E.g.
$$f(x_1, x_2, x_3, x_4) = y \Leftrightarrow \begin{array}{c} g_1(x_1, x_2) = y_1 \\ g_2(x_2, x_3) = y_2 \\ g_3(x_3, x_4) = y_3 \\ F(y_1, y_2, y_3) = y \end{array}$$

Formal Definitions

- X variable set of size n over F_q
- f_i polynomials in $X_i \subseteq X$
- Find all solutions in F_q to equations:

$$f_1(X_1) = 0, \ldots, f_m(X_m) = 0$$

- We study $|X_i| \leq l$ for a small parameter $l = 3, 4, \ldots$
- No other restrictions
- Brute force search complexity q^n trials
- **GOAL: Fastest Way to Solve**

Typical equations $\mod 2$:

$$x_1x_6 + x_3 \equiv 0$$

$$x_2x_4 + x_5 + 1 \equiv 0$$

$$x_1x_2x_5 + x_1 + x_2 \equiv 0$$

$$x_3x_4 + x_5 + 1 \equiv 0$$

$$x_3x_6 + x_3 + x_5 \equiv 0$$

$$x_4x_5 + x_1 + x_4 \equiv 0$$

Gröbner basis Algorithms

- Destroy sparseness
- Require huge memory even for relatively small problems
- Generally, only efficient(complexity$<\ q^n$) for quadratic and very over-defined systems (m>n)

Write equations as l-SAT formulas($q = 2$)

- One equation

$$f(x_1, \ldots, x_l) = 0 \quad \Leftrightarrow \quad F_f = \wedge_{f(a_1,\ldots,a_l)=1}(x_1^{(a_1)} \vee \ldots \vee x_l^{(a_l)}) = 1,$$

  where $x^{(1)} = \bar{x}$ and $x^{(0)} = x$

- The system is equivalent to $\wedge_i F_{f_i} = 1$. An l-SAT problem
- Worst case bounds, survey in [Iwama,04]:

$l =$	3	4	5	6
the worst case	1.324^n	1.474^n	1.569^n	1.637^n

- $\Rightarrow$ Worst case bounds for Sparse equations

Another Representation of Equations

- First in [Zakrevskij-Vasilkova,00], independently in [Raddum,04]
- $f_i(X_i) = 0 \Leftrightarrow$ solutions V_i in variables $X_i \Leftrightarrow S_i = (X_i, V_i)$
- E.g.

$$x_1 x_2 + x_3 \equiv 0 \mod 2 \Leftrightarrow \begin{array}{ccc} x_1 & x_2 & x_3 \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$$

- Solve equations $S_1, \ldots, S_m$ with:
- Gluing
- Pairwise Agreeing

Gluing Procedure

$$\begin{array}{ccc} x_1 & x_2 & x_3 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{array} \quad \circ \quad \begin{array}{ccc} x_1 & x_2 & x_4 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array} = \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array}$$

- Common variables $\{x_1, x_2\}$
- Glue vectors with the same sub-vector in $\{x_1, x_2\}$
- The number of resulting vectors may grow
- Appears in [Semaev, WCC'07].

Gluing Algorithm

- **input**: Equations:

$$S_1 = (X_1, V_1), \ldots, S_m = (X_m, V_m).$$

- Compute $S_1 \circ S_2 \circ \ldots \circ S_m = (X(m), U_m)$
- **output**: Solutions U_m
- Intermediate $S_1 \circ S_2 \circ \ldots \circ S_k = (X(k), U_k)$ require large memory

Gluing Algorithm Example

Given 3 equations

$$
\begin{array}{c|cc}
 & x_1 & x_2 \\
\hline
a_1 & 0 & 0 \\
a_2 & 1 & 0 \\
a_3 & 1 & 1
\end{array}
,\qquad
\begin{array}{c|cc}
 & x_2 & x_3 \\
\hline
b_1 & 0 & 0 \\
b_2 & 1 & 0 \\
b_3 & 1 & 1
\end{array}
,\qquad
\begin{array}{c|cc}
 & x_1 & x_3 \\
\hline
c_1 & 0 & 0 \\
c_2 & 0 & 1 \\
 & &
\end{array}
,
$$

Compute two gluings:

$$
\begin{array}{cc}
x_1 & x_2 \\
\hline
0 & 0 \\
1 & 0 \\
1 & 1
\end{array}
\circ
\begin{array}{cc}
x_2 & x_3 \\
\hline
0 & 0 \\
1 & 0 \\
1 & 1
\end{array}
=
\begin{array}{ccc}
x_1 & x_2 & x_3 \\
\hline
0 & 0 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
1 & 1 & 1
\end{array}
,\qquad
\begin{array}{ccc}
x_1 & x_2 & x_3 \\
\hline
0 & 0 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
1 & 1 & 1
\end{array}
\circ
\begin{array}{cc}
x_1 & x_3 \\
\hline
0 & 0 \\
0 & 1
\end{array}
=
\begin{array}{ccc}
x_1 & x_2 & x_3 \\
\hline
0 & 0 & 0
\end{array}
$$

One solution

Gluing1 Algorithm

- The same expected running time
- Requires polynomial memory
- Algorithm walks through a Search tree
- Easy to understand with Example

Gluing1 Algorithm Example

- Equations: $V_1 = \{a_1, a_2, a_3\}$, $V_2 = \{b_1, b_2, b_3\}$, and $V_3 = \{c_1, c_2\}$

	x_1	x_2
a_1	0	0
a_2	1	0
a_3	1	1

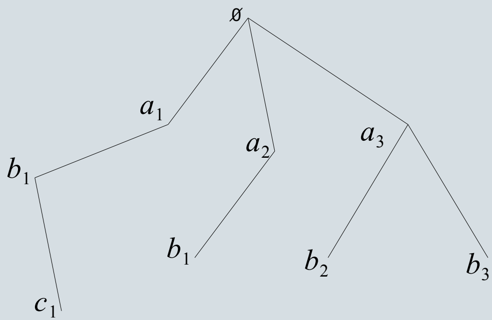
,

	x_2	x_3
b_1	0	0
b_2	1	0
b_3	1	1

,

	x_1	x_3
c_1	0	0
c_2	0	1

- The search tree:



- The solution $a_1 \circ b_1 \circ c_1 = (x_1, x_2, x_3) = (0, 0, 0)$

Agreeing Procedure

x_1	x_2	x_3
0	0	1
0	0	0
0	1	0
1	1	1

x_1	x_2	x_4
0	0	0
1	0	1
1	1	0
1	1	1

- Common variables $\{x_1, x_2\}$
- Projections on $\{x_1, x_2\}$:
- $00, 01, 11$ and $00, 10, 11$
- Remove vectors with projection not in the projections of another list
- Appears in [Zakrevskij-Vasilkova,00] and [Raddum,04]

Agreeing-Gluing1 Algorithm

- Follow the Search tree as in Gluing1 and compute

$$a \circ b \ldots \circ c$$

  a solution to $S_1 \circ S_2 \circ \ldots \circ S_k$

- **If** $a \circ b \ldots \circ c$ **contradicts to at least one of**

$$S_{k+1}, \ldots, S_m,$$

  Then remove every branch passing through $a, b, \ldots, c$**.**

- Lots of branches are cut
- Complexity abruptly falls

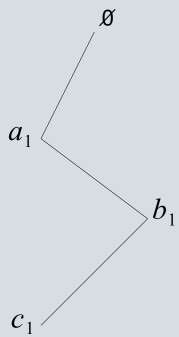
- A more general algorithm in [Raddum-Semaev,06].

Agreeing-Gluing1 Algorithm Example

- Equations: $V_1 = \{a_1, a_2, a_3\}$, $V_2 = \{b_1, b_2, b_3\}$, and $V_3 = \{c_1, c_2\}$

$$\begin{array}{c|cc} & x_1 & x_2 \\ \hline a_1 & 0 & 0 \\ a_2 & 1 & 0 \\ a_3 & 1 & 1 \end{array}, \quad \begin{array}{c|cc} & x_2 & x_3 \\ \hline b_1 & 0 & 0 \\ b_2 & 1 & 0 \\ b_3 & 1 & 1 \end{array}, \quad \begin{array}{c|cc} & x_1 & x_3 \\ \hline c_1 & 0 & 0 \\ c_2 & 0 & 1 \\ & & \end{array}$$

- The search tree:



- The solution $a_1 \circ b_1 \circ c_1 = (x_1, x_2, x_3) = (0, 0, 0)$

Probabilistic Model

- Agreeing-Gluing algorithms are **deterministic**
- **Equiprobable** instances distribution:
- For natural numbers m, n and $l_1, \ldots, l_m \leq l$
  1. Independent equations $f_i(X_i)$
  2. X_i uniformly random l_i-subsets of X
  3. f_i uniformly random polynomials of degree $\leq q-1$ in each variable
- Running time is a random variable. **Find expectation**

Gluing Algorithm Asymptotic

- With Gluing
$$S_1 \circ S_2 \circ \ldots \circ S_k = (X(k), U_k)$$
.
- Gluing Algorithm Complexity is
$$O(\sum_k |U_k|) = O(m \max_k |U_k|)$$
- $X_1, \ldots, X_k$ are fixed, then
$$E_{f_1,\ldots,f_k}|U_k| = q^{|X(k)|-k}$$
.
- Expected complexity is roughly
$$\max_k \quad E_{X_1,\ldots,X_k}(q^{|X(k)|-k})$$
- Estimated in [Semaev, WCC'07] with Random Allocations Theory.

Agreeing-Gluing Algorithm Asymptotic

- $S_1 \circ S_2 \circ \ldots \circ S_k = (X(k), U_k)$
- U'_k solutions in U_k agreed to each of $S_{k+1}, \ldots, S_m$
- Algorithm's Complexity

$$O(\sum_k |U'_k|) = O(m \max_k |U'_k|)$$

- $X_1, \ldots, X_k$ are fixed, then

$$E_{f_1,\ldots,f_k}|U'_k| = E_{f_1,\ldots,f_k}|U_k| \prod_{i=k+1}^{m} (1 - (1 - \frac{1}{q})^{q^{|X_i \setminus X(k)|}})$$

- Expected complexity is roughly

$$\max_k \quad E_{X_1,\ldots,X_k}(q^{|X(k)|-k} \prod_{i=k+1}^{m} (1 - (1 - \frac{1}{q})^{q^{|X_i \setminus X(k)|}})$$

- Estimated in the Proceedings of ACCT'08

Algorithms Running Time(q=2)

n Boolean equations in n variables, each equation depends on at most l variables

$l =$	3	4	5	6
the worst case	1.324^n	1.474^n	1.569^n	1.637^n
Gluing1, expectation,[WCC07]	1.262^n	1.355^n	1.425^n	1.479^n
Agreeing-Gluing1, expectation[ACCT08]	1.113^n	1.205^n	1.276^n	1.334^n

- Worst and average cases of the problem are excitingly different
- Why?
- Any **Clause** in l variables has $2^l - 1$ satisfying assignments
- Average number of solutions to a random **Equation** in l variables is 2^{l-1}
- Average l-SAT problem is apparently harder

Conclusions

- Proven **here** expected complexity bounds are significantly **lower** than known worst case bounds

- **At least theoretically** new methods seem **better** than Gröbner Basis Algorithms and SAT solvers