Phase Transition in a System of Random Sparse Boolean Equations

Thorsten Ernst Schilling and Pavol Zajac

University of Bergen, KAIVT FEI STU Bratislava
thorsten.schilling@ii.uib.no, pavol.zajac@stuba.sk

Abstract. Many problems, including algebraic cryptanalysis, can be transformed to a problem of solving a (large) system of sparse Boolean equations. In this article we study 2 algorithms that can be used to remove some redundancy from such a system: Agreeing, and Syllogism method. Combined with appropriate guessing strategies, these methods can be used to solve the whole system of equations. We show that a phase transition occurs in the initial reduction of the randomly generated system of equations. When the number of (partial) solutions in each equation of the system is binomially distributed with probability of partial solution p, the number of partial solutions remaining after the initial reduction is very low for p's below some threshold p_t, on the other hand for $p > p_t$ the reduction only occurs with a quickly diminishing probability.

Key words: Algebraic cryptanalysis, Agreeing, Boolean equations, SAT problem.

1 Introduction

Given an equation system (1) over a finite field $\mathbb{F}_q$ it is a well known NP-complete problem to determine a common solution to all equations. Finding a solution to such an equation system can be interesting in algebraic cryptanalysis, e.g. when the solution to the equation system is a constraint to a used, unknown key.

Experiments with different solving algorithms suggest that during the solving the number of possible solutions is not decreasing *continously*. That means that during the solving process the overall number of solutions does not decrease constantly, but that at some point the number of possible solutions decreases rapidly.

In this paper we try to determine this point of *phase transition* in order to get a better measure for the hardness of a given problem.

The paper is organized as follows. In Section 2 we explain the basic representation of equations and the idea how the number of potential solutions to the equation system can be reduced. Section 3 explains the Agreeing algorithm and the reduction by Agreeing. Section 4 explains the reduction technique by syllogisms. In Section 5 we make a direct comparison of these both techniques. Section 6 shows our experimental results on a series of random sample instances and Section 7 concludes the paper.

2 Representation of the system of sparse Boolean equations and its reduction

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of variables (unknowns), and let $X_i \subset X$ for $i = 1, ..m$, such that $|X_i| = l$. We consider X_i to be chosen uniformly at random from all possible l-subsets of X. Let $\mathcal{F}$ be a system of Boolean equations

$$f_1(X_1) = 0, ...f_m(X_m) = 0, \tag{1}$$

such that f_i depends only on variables from the set X_i. Let V_i be a set of vectors that are projections of solutions of $f_i(X) = 0$ into variables of X_i. We call (X_i, V_i) a symbol, and we say that the symbol represents the equation $f_i(X_i) = 0$. We call vectors of V_i partial solutions of the system.

To compute all solutions of the whole system we can apply the so called Gluing procedure [2]. The procedure is as follows: We merge two symbols (X_i, V_i), (X_j, V_j) together and enumerate all possible solutions V_{ij} of a new symbol $(X_i \cup X_j, V_{ij})$. Then we replace the original two symbols with a new one. Until some point the total number of solutions grows (very quickly). Gluing new symbols together removes some of the partial solutions, until only the valid solutions of the system remain. More advanced algorithms based on Gluing use different Gluing strategies, and strategies for removal excess solutions before/without Gluing, and some combinations with guessing variable values or solutions of individual equations. The fastest algorithm based on Gluing up to date is the Improved Agreeing-Gluing Algorithm introduced in [5].

In this article we want to focus on methods that do not use Gluing or any guessing. Consider the situation where V_i contains just one solution. We know immediately the values of l variables. Thus these values can be substituted into all other equations, and conflicting partial solutions get removed. Solutions from the set V_j can be removed if it shares some variables with V_i. If the remaining number of possible solutions in V_j is small, we can find new "fixed" values of variables, and spread this information, until (almost) all variables have fixed values. This technique is also called the Spreading of constants [9]. A more advanced version, the local reduction technique [9], uses fixed /resp. forbidden/ solutions for groups of variables. The similar method, although differently formulated is the Agreeing method. Agreeing uses a more efficient representation, and can be extended to more efficient variants [3, 4]. A different reduction method based on Syllogism rule (transitiveness of the implication relation) was also presented in [9], and was later adapted to the symbol representation [7].

We investigate the behavior of the reduction methods in a random sparse Boolean equation system as a function of one additional parameter: The probability of a partial solution p. We do not explicitly write down the closed form for f_i, instead we generate each symbol in a stochastic manner. We want to investigate the systems that have at least one solution, so we generate first a random solution $\mathbf{x}$. Then for each symbol we generate the set V_i in such a way that the probability of $v \in V_i$ is 1, if v is a projection of $\mathbf{x}$ to X_i, and p otherwise. The number of solutions in each symbol is then binomially distributed $|V_i| \sim Bi(2^l, p)$.

We call the variable x_j fixed, if the projection of all $v \in V_i$ to x_j in some equation (X_i, V_i), with $x_j \in X_i$ contains only one value, either 0 or 1. The system is solved by an algorithm A, if all variables are fixed after the application of the algorithm A. To investigate various algorithms we run the following experiment:

1. Given the set of parameters (m, n, l, p)[1], generate a set of N random equation systems (as defined above).
2. For each system, apply the reduction algorithm A.
3. Compute the fixation ratio $r = f/n$, where f is the number of fixed variables (after the application of A).
4. Compute the average fixation ratio $\hat{r} = 1/N \sum r$ for the whole set of experiments.

If the average fixation ratio stays near 0, then we didn't learn any significant information about the solution of the system by the application of the algorithm A. To solve the system, we must either use a different algorithm, or reduce the system by guessing some solutions. The basic guessing is exponential in nature, thus the system (in our settings) need an exponential time to solve. On the other hand, if the average fixation ratio is near 1, we have a high chance to solve the whole (randomly generated) system just by applying A. In this case, if the runtime of algorithm A is bounded in polynomial time, we can say that the average instance of the problem (m, n, l, p) is solvable in polynomial time.

3 Reduction by Agreeing

In order to find a solution to a set of symbols the Agreeing algorithm attempts to delete vectors from symbols S_i which cannot be part of a common solution. In the following, the projection of a vector v_k on variables X is denoted by $v_k[X]$ and $V[X]$ denotes the set of projections of all vectors $v_k \in V$ on variables X.

Given two symbols $S_i = (X_i, V_i)$ and $S_j = (X_j, V_j)$ with $i \neq j$ we say that S_i and S_j are in a non-agreeing state if there exists at least one vector $a_p \in V_i$ such that $a_p[X_i \cap X_j] \notin V_j[X_i \cap X_j]$. If there exists a solution to the system, each symbol will contain one vector that matches the global solution. The vector a_p cannot be combined with any of the possible assignments in symbol S_j, hence it cannot be part of a solution to the whole system and can be deleted. The deletion of all vectors $a_p \in V_i$ and $b_q \in V_j$ which are incompatible with all vectors in V_j and V_i, respectively, is called agreeing. If by agreeing the set of vectors of a symbol gets empty, there exists no solution to the equation system. The agreeing of all pairs of symbols in a set of symbols $\mathcal{S} = \{S_0, \ldots, S_{m-1}\}$ until no further deletion of vectors can be done is called the Agreeing algorithm.

After running Algorithm 1 on $\mathcal{S}$ we call $\mathcal{S}$ *pair-wise agreed*. On the average all $S_i \in \mathcal{S}$ have exactly one one vector left. The solution to the system is then the gluing of the remaining vectors and the system can be regarded as *solved*. If on the other hand one or more symbols get *empty*, i.e. $V_i = \emptyset$, the system has no common solution.

[1] We use $m = n$, as this is the most important situation.

Algorithm 1 Agreeing Algorithm

```
1: procedure AGREE(S)
2:     while (X_i, V_i), (X_j, V_j) ∈ S which do not agree do
3:         Y ← X_i ∩ X_j
4:         Delete all a_p ∈ V_i for which a_p[Y] ∉ V_j[Y]
5:         Delete all a_q ∈ V_j for which a_q[Y] ∉ V_i[Y]
6:     end while
7: end procedure
```

Example 1 (Agreeing). The following pair of symbols is in a non-agreeing state:

S_0	0	1	2
a_0	0	0	0
a_1	0	0	1
a_2	0	1	0
a_3	1	1	1

S_1	0	1	3
b_0	0	0	0
b_1	1	0	1

.

The vectors a_2, a_3 differ from each b_j in their projection on common variables x_0, x_1 and can be deleted. Likewise, b_1 cannot be combined with any of the a_i and can also be deleted. After agreeing the symbols become:

S_0	0	1	2
a_0	0	0	0
a_1	0	0	1

S_1	0	1	3
b_0	0	0	0

.

Guessing and Agreeing In a usual setting, e.g. given as an input equation systems from ciphers, Agreeing does not yield a solution immediately. The algorithm has to be modified in a way that one has to introduce *guesses*.

4 Reduction by Syllogisms

Let (X, V) be an equation, $x_i, x_j \in X$. Let us have two constants $a, b \in \mathbf{F}_2$ such that for each $\mathbf{v} = (x_{i_1}, \ldots, x_i, x_j, \ldots, x_{i_l}) \in V : (x_i + a)(x_j + b) = 0$. We say that equation (X, V) is constrained by $(x_1 + a)(x_2 + b) = 0$, or that $(x_1 + a)(x_2 + b) = 0$ is a 2-constraint for the equation (X, V). A solution $\mathbf{x}$ of the whole system $\mathcal{F}$ projected to variables X_i must also be a partial solution in V_i. Thus $\mathbf{x}$ is constrained by every 2-constraint we place on each of the equations in the system. Thus we can apply 2-constraints found in (X_i, V_i) to remove those partial solutions of (X_j, V_j), that violate some of the 2-constraints. This is the basis of the syllogism reduction technique, that is similar to Agreeing. The main difference is the addition of creating new 2-constraints by the syllogism rule (see below).

We can see each 2-constraint $(x_i + a)(x_j + b) = 0$ as one clause of type $x_i^{(a)} \vee x_j^{(b)}$, , where $x^{(0)} = \overline{x}$ (negation of x), and $x^{(1)} = x$. All such clauses must be satisfied by the solution of the system. However, if some vector $\mathbf{y}$ satisfies all

such clauses, it does not automatically mean it is a solution of the system[2]. To check whether the set of 2-constraints written in a form of clauses is satisfiable is the well known 2-SAT problem. We must note, that we are not solving the 2-SAT problem, if we already know that the solution exists. However, if the system contains a large set of 2-constraints, we expect that if we remove the correct solution the system becomes unsatisfiable. Then we expect to be able to remove almost all invalid solutions from the system using just the 2-constraints.

We can also rewrite the 2-constraint in the form of two (equivalent) implications: $x_i^{(a+1)} \Rightarrow x_j^{(b)}$, and $x_i^{(b+1)} \Rightarrow x_j^{(a)}$. Implication is a transitive relation, i.e. if $x \Rightarrow y$ and $y \Rightarrow z$, it follows that $x \Rightarrow z$. This derivation is also called the syllogism rule. Thus, if we have two 2-constraints $(x_i + a)(x_j + b) = 0$, $(x_j + b + 1)(x_k + c) = 0$, we can derive a new 2-constraint $(x_i + a)(x_k + c) = 0$. The new 2-constraints then can be used to remove additional partial solutions from the system. It is also possible to derive special 2-constraints in the form $(x_k + a)(x_k + a) = 0$, which simply means that $x_k = a$, and thus x_k is fixed.

A set of 2-constraints is transitively closed, if we cannot derive any more 2-constraints using the transitiveness property of the underlying implications. A transitively closed set of 2-constraints thus contain the maximum of information we can get from the system (using just 2-constraints). We represent a set of 2-constraints in a form of the implication graph. Vertices of the graph are labelled by $\{x_1, x_2, \ldots, x_n, \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$. Edge (x, y) exists if there is an implication $x \Rightarrow y$ (so a single 2-constraint is always represented by 2 edges). To find the transitively closed set of 2-constraints, we compute the transitive closure of the implication graph (by some of the known algorithms).

The Syllogism reduction method thus works as follows:

1. Examine the set of equations, and find all 2-constraints.
2. For each 2-constraint, add corresponding implications to the implication graph.
3. Compute the transitive closure of the implication graph.
4. Apply all 2-constraints back to the set of equations, i.e. remove all solutions from each V_i that violate any of the 2-constraints stored in the implication graph.
5. If some solutions were removed, repeat the algorithm, otherwise output the reduced system.

The transitive closure of an implication graph can be computed e.g. by Warshall's algorithm [6] in $O(n^3)$. After each repetition of the transitive closure algorithm, we must remove add at least one partial solution, otherwise the method stops. Thus the worst case complexity is upper bounded in $O(Mn^3)$, where $M \approx mp2^l$ is the initial number of solution. Actually the number of repetitions of the algorithm is very small in practice, especially if the system cannot be reduced (usually just one repetition). However, we need an additional $O(n^2)$

[2] It is only true, if we the set of 2-constraints is tight, i.e. for each equation (X, V) we can find such a set of 2-constraints, that no other assignment of variables is permissible except those in V.

memory storage for the implication graph. A more detailed analysis is provided in [8].

The original method presented in [9] uses immediate resolution of transitive closure after adding each new 2-constraint (also the implication graph is represented differently), but the algorithm gives the same results (although the running times differ, but these depend also on the implementation, and the platform used, respectively). Experimental results in [9] also show that a phase transition effect exists, but no theoretical explanation or expected parameters are provided.

4.1 The heuristic model for the expected behavior

The phase transition in the syllogism method can be connected to the corresponding representation of the problem in CNF clauses $x_i^{(a)} \vee x_j^{(b)}$. Each of these clauses must be satisfied simultaneously, so we get a 2-SAT problem instance in n variables with k clauses, where k is the total number of clauses (2-constraints) in the system. It was shown in [1] that if we have a random 2-SAT problem with k clauses in n variables, having $k/n = \alpha$ fixed as $n \to \infty$, then for $\alpha > 1$ almost every formula is unsatisfiable, and for $\alpha < 1$ almost all formulas can be satisfied. To use this result for the syllogism method, we must first estimate the number of constraints in the system.

Lemma 1. *Let $\mathcal{S} = (X, V)$ be a randomly chosen symbol with $l = |X| \geq 2$ active variables, and $s = |V|$ distinct solutions. Let $p_{s,l}$ denote a probability, that a randomly chosen constraint $(x_i + a)(x_j + b) = 0$, $x_i, x_j \in X$, $a, b \in \{0, 1\}$ holds for an equation defined by symbol $\mathcal{S}$. Then*

$$p_{s,l} = \prod_{i=0}^{s-1} \frac{3 \cdot 2^{l-2} - i}{2^l - i} \tag{2}$$

Proof. There are 2^l possible solutions. For $s = 1$, there are 2^{l-2} solutions for which the constraint $(x_i + a)(x_j + b) = 0$ does not hold, namely those where $x_i = a + 1$ and $x_j = b + 1$. For all other $3 \cdot 2^{v-2}$ solutions the constraint holds, so the probability $p_{1,l} = \frac{3 \cdot 2^{l-2}}{2^l} = 3/4$. If we have already i constrained solutions, we can choose the next constrained solution from only $3 \cdot 2^{l-2} - i$ vectors out of $2^l - i$, thus $p_{i+1,l} = p_{i,l} \frac{3 \cdot 2^{l-2} - i}{2^l - i}$. By expanding this recursion we get equation (2).

Using $p_{s,l}$ from equation (2), we can compute the probability of a constrained solution in a symbol from system generated with the binomial distribution:

$$P_{l,p} = \sum_{s=0}^{2^l} \binom{2^l}{s} p^s (1-p)^{2^l - s} p_{s,l}. \tag{3}$$

The expected number of constraints in an equation is $\alpha(l, p) = 4\binom{l}{2} P_{l,p}$. The total number of expected constraints is $k = \alpha m$. We do not take into account

the constraints found by the syllogism rule. The phase transition point should be near the value p_t for which $k/n = 1$. For our experiments $m = n$, thus we are looking for p_t for which $\alpha(l, p_t) = 1$. If $p > p_t$ we get $\alpha(p, l) < 1$, thus the corresponding 2-SAT problem is very likely satisfiable, and the syllogism method cannot eliminate much solutions. If $p < p_t$, $\alpha(p, l) > 1$, and the corresponding 2-SAT problem is very likely unsatisfiable. Then almost all excess solutions get removed by 2-constraints during the application of the syllogism method. The expected phase transition probabilities are summarized in Table 1.

l	p_t	$p_t \cdot 2^l$
5	0.3694	11.8
6	0.2258	14.5
7	0.1293	16.6
8	0.0711	18.2
9	0.0381	19.5
10	0.0201	20.6

Table 1. Probabilities p_t at which the phase transition in syllogism method is expected to occur.

5 Qualitative comparison of the methods

There exists a set of equations with all partial solutions in Agreeing state, that can be reduced by the Syllogism method. One of the examples is presented in Table 1. In the example, we get constraints between variables 1, 2 ($x_2 \Rightarrow x_1$), variables 2, 3 ($x_3 \Rightarrow x_2$), but originally no constraint between variables 1, 3. A new constraint ($x_3 \Rightarrow x_1$) can be derived using the transitive closure. This new constraint removes one partial solution ($x_1 = 0, x_3 = 1, x_6 = 1$), and furthermore allows us to find a fixed solution $x_6 = 0$. We remark that the same effect is obtained, if we glue two of the equations together, and agree them with the third equation. It is thus possible, that the syllogism method can reduce the system that the agreeing method is unable to.

S_0	1 2 4
a_0	0 0 0
a_1	0 0 1 ,
a_2	1 0 1
a_3	1 1 1

S_1	2 3 5
b_0	0 0 1
b_1	1 0 0 ,
b_2	1 0 1
b_3	1 1 1

S_2	1 3 6
c_0	0 0 0
c_1	0 1 1
c_2	1 0 0
c_3	1 1 0

Fig. 1. Example of the agreeing equation system (or a part of one) reducible by the method of Syllogisms.

If two equations have only one or two common variables, and if they are not agreeing, it is possible to find a 2-constraint in at least one of them, that can be used to reduce the solutions in the second one. After the reduction we get the same result as if agreeing was run. However, if we have more than two common variables, it is possible that no 2-constraints can be found that restrict the solutions, one such example is provided in the Figure 2. As l — the number of variables per equation — grows, this situation becomes more probable, and the agreeing method will be able to reduce more solutions as the syllogism method.

S_0	1	2	3
a_0	0	0	0
a_1	0	0	1
a_2	0	1	0
a_3	1	0	0
c_4	1	1	1

’

S_1	1	2	3
b_0	0	0	0
b_1	1	1	0
b_2	0	1	1
b_3	1	0	1
b_4	0	0	0

Fig. 2. Example of the disagreeing equation system without any 2-constraints.

The Syllogism method is preferable, if only weak connections (usually only one common variable) are between equations. In these cases, we can derive more information using the Syllogism rule than just by Agreeing (which only checks projection to this single common variable). In a system of random equations, this situation is more probable, when the system is very sparse. The Agreeing method provides more information when there are 3 or more common variables, and a low probability of 2-constraints. The practical experiments show (see Section 6) that the two methods have almost the same behaviour when $l = 7$. The method of syllogisms is preferable for $l < 7$, and vice-versa.

6 Experimental Results

In this section we present the results of the experiments used to locate the point of phase transition for equation systems with $m = n = 80$ variables and varying sparsity. We used each of the methods on the same set of $N = 1000$ random equation system, and $p = 0, 0.005, \ldots, 0.35$ and sparsities $l = 6, 7, 8$. Figure 3 shows the phase transition for different methods, and sparsities, respectively.

Table 2 summarizes the upper and lower bound for the transition in systems with $m = n = 100$. Precision for p is 0.02. The lower bound is the highest p, for which all 1000 equations were solved, and the upper bound is the lowest p, for which no equation was solved, respectively.

7 Conclusions

The experimental results confirm the phase transition effect. The transition is not sharp for smaller systems and sparsities. There is a region of probabilities p,

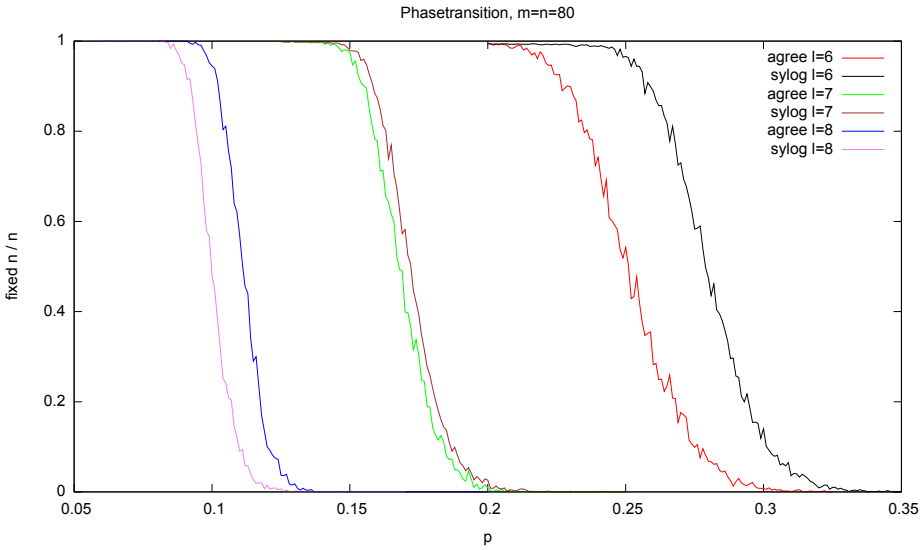

Fig. 3. Plot of the average fixation ratio showing the phase transition effect.

	Agreeing		Syllogisms	
l	low	up	low	up
5	0.26	0.42	0.34	0.46
6	0.18	0.32	0.22	0.34
7	0.12	0.20	0.14	0.22
8	0.08	0.14	0.06	0.14
9	0.04	0.10	0.04	0.08

Table 2. Experimental bounds on phase transition for $m = n = 100$.

where it is possible to generate both solvable and unsolvable systems. However, as the number of variables and equation grows, the phase transition becomes sharper, and it is less probable to reduce the system above the phase transition point.

A typical situation for the random equation system is $p = 1/2$, which is above the phase transition point in every case examined. However, the consequence of the phase transition effect for smaller p's is that we can reduce the required number of guesses required before we can solve the whole equation system even if it is originally above the phase transition point.

Let us suppose we have a system of m (random) equations with $n = m$ variables, l-sparse. Each of 2^l $\{0, 1\}$-vectors can be a solution of an equation in the system with probability p (usually $1/2$), i.e. the expected number of solutions in each equation is $p2^l$. The expected total number of partial solutions (listed in symbols) is then $mp2^l$. Let us guess the value of one variable, without the loss of generality x_1. We expect x_1 to be an active variable on average in l

equations. Thus we expect that we remove on average a half of $lp2^l$ partial solutions. The expected new number of solutions is thus $mp2^l - p2^{l-1}$, which is the same number, as if expect from a system generated with a lower solution probability $p' = p(1 - \frac{l}{2m})$.

After x (independent) guesses we expect the same number of partial solutions as in a system generated with $p_x = p(1 - \frac{l}{2m})^x$. To reach the zone below the phase transition point, we need to find $p_x \leq p_t$. The expected number of required guesses to reach this point is then

$$x = \frac{\log p_t - \log p}{\log(1 - \frac{l}{2m})}$$

It means, that we have to check only 2^x instead of the full 2^n possible vectors to eliminate incorrect/find the correct solution. If we can write $x = cn$ for some constant c, we get the complexity estimate $O(2^{cn})$ to determine the whole solution of the system by the guessing algorithm (in combination with A, e.g. Agreeing or Syllogism method). Estimates based on lower bounds from experimental results (see Table 2) are summarized in Table 3. We must stress, that this is only an estimate based on experiments. It is necessary to provide proper mathematical models to find the exact asymptotic behaviour of the methods. However, a full mathematical model for the reduction that takes into account all parameters m, n, p, l for both the Agreeing and Syllogism methods is still an open question.

Table 3. Estimated complexities $O(C^n)$ of the guessing algorithm for different l's. p_A is the experimental lower bound for phase transition of Agreeing, and p_S is the experimental lower bound for phase transition of Syllogism method. Columns Worst and IAG are provided for comparison with [5].

l	p_A	C	p_S	C	Worst	IAG
5	0.26	1.199	0.34	1.113	1.569	1.182
6	0.18	1.266	0.22	1.209	1.637	1.239
7	0.12	1.327	0.14	1.287		
8	0.08	1.373	0.06	1.444		

Another consequence is for the guessing order. If we want to guess a new value, we should choose the variable in such a way, so that we affect the highest number of partial solutions by the guess (resp. by guessing 0 as well as guessing 1). In this way, after removing the partial solutions that have an incorrect value for the guessed variable, we get nearer to the phase transition point. This should be the best generic guessing strategy possible. If we want to evaluate more advanced guessing strategy, e.g. applications of learning [4], it can be considered effective, if it gives a solution to the system in lower number of guesses (on average) than the guessing strategy using the phase transition.

The phase transition point is also useful for evaluating the different reduction algorithms. If two polynomial time reduction algorithms A_1, A_2 both have a

phase transition effect at solution probabilities $p_1 < p_2$, then a theoretically a more effective one is A_2. However in practice the advantage of A_2 can only be realized in large systems, which cannot be solved in practice with the present computational resources.

References

1. Goerdt, A.: *A threshold for unsatisfiability* , J Compute System Sci **53** (1996), 469–486.
2. Raddum, H., Semaev, I.: *New Technique for Solving Sparse Equation Systems*, Cryptology ePrint Archive: Report 2006/475. Available at `http://eprint.iacr.org/2006/475`.
3. Raddum, H., Semaev, I.: *Solving Multiple Right Hand Sides linear equations,* Designs, Codes and Cryptography **49** (2008), 147–160.
4. Schilling, T.E., Raddum, H.: *Solving Equation Systems by Agreeing and Learning,* preprint (2010).
5. Semaev, I.: *Improved Agreeing-Gluing Algorithm*, Cryptology ePrint Archive: Report 2010/140. Available at `http://eprint.iacr.org/2010/140`.
6. Warshall, S.: *A theorem on Boolean matrices,* Journal of the ACM **9**, No. 1 (1962), 11–12.
7. Zajac, P.: *Solving SPN-based system of equations with syllogisms*, in: 1st Plenary Conference of the NIL-I-004, Bergen, August, 24–27, 2009. (A. Kholosha — K. Nemoga — M. Sýs eds.), STU v Bratislave, 2009, pp. 21–30.
8. Zajac, P.: *Implementation of the method of syllogisms,* preprint (2010).
9. Zakrevskij, A., Vasilkova, I.: *Reducing Large Systems of Boolean Equations*, in: 4th International Workshop on Boolean Problems, Freiberg University, September, 21–22, 2000.