Publ. Math. Debrecen
Supplementum **100** (2022), 683–700
DOI: 10.5486/PMD.2022.Suppl.8

On solving sparse MRHS equations with bit-flipping

By Pavol Zajac

Abstract. We present a new class of probabilistic algorithms that can be used to solve (sparse) MRHS equation systems. The algorithms are based on the idea of bit-flipping: start from a random vector as a potential solution, and try to improve individual bits according to some selected heuristic strategy. We have evaluated a group of algorithms experimentally on a model of sparse MRHS systems based on AND-XOR circuits with low gate count. We compare bit-flipping algorithms with a more complex hill-climbing algorithm and show that bit-flipping algorithms can achieve better success probability for the same number of MRHS evaluations.

1. Introduction

Modern lightweight cipher designs, such as LowMC [2], and MiMC [1], try to minimize the number of (specific) logic gates required to realize the encryption algorithm. Lowered number of gates leads to a more efficient implementation [6], and a lower power consumption. Furthermore, there are important cryptographic applications of primitives with low number of (non-linear) gates, such as homomorphic encryption [8], side channel protection [7], MPC [5], and others.

On the other hand, a low number of gates, especially non-linear AND gates, can have a negative impact on cipher security. In [13] we have investigated a generic reduction of an algebraic attack to an instance of a decoding problem. We have provided a lower bound on the number of AND gates required in the cipher design (relative to the security level) to resist attacks utilizing generic decoding algorithms.

Mathematics Subject Classification: 94A60, 08-08, 14G50.
Key words and phrases: MRHS, bit-flipping, algebraic cryptanalysis.
This research was sponsored by Slovak Republic under grant VEGA 2/0072/20.

Suppose that a lightweight design has a low number of both AND gates and XOR gates. Algebraic cryptanalysis of such a design can be transformed into a multiple right-hand sides (MRHS) equation system [9] with a specific form of right-hand side sets (RHS) and a sparse joint matrix.

Unlike our prior research in deterministic solving methods [11], [10], in our present contribution, we investigate heuristic methods for solving sparse MRHS systems. The main idea of the solving algorithm is an adaptive bit flipping. We start from a random potential solution, and then we try to change some bits based on the information obtained from the RHS sets. We investigate different bit-flipping strategies and their combinations.

We extend the investigation also to bit flipping strategies based on global information similar to Hill climbing. Our optimization function is the number of unsolved right-hand sides. Starting with a random potential solution, we evaluate each potential bit-flip and continue by a greedy strategy: we flip that bit which provides the best improvement in the number of solved right-hand sides. If no bit flip leads to a better situation, we restart from another random potential solution. Note that this approach is different from classical heuristic optimization approaches to solving ciphers, as it does not work on keyspace bits but instead works with all internal bits and their interconnections.

2. Preliminaries

Notes on notation: In this article, we always use row vectors. We denote vectors with small bold letters, such as $\mathbf{v}$. Vector $\mathbf{e}^i$ contains exactly one value 1 at position i, and 0's otherwise. Matrices are denoted by capital bold letters such as $\mathbf{M}$. Sets are denoted by simple capital letters such as S, with special sets using specific notation such as $\mathbb{F}$ (a generic finite field).

Definition 1. Let $\mathbb{F}$ be a finite field. Let $\mathbf{M}$ be an $n \times l$ matrix over $\mathbb{F}$, and let $S \subset \mathbb{F}^m$. *MRHS equation* is an inclusion in the form

$$\mathbf{x} \cdot \mathbf{M} \in S. \tag{1}$$

Vector $\mathbf{x} \in \mathbb{F}^n$ is a solution of the MRHS equation (1) if the corresponding inclusion is satisfied.

A MRHS equation system (simplified to a *MRHS system*) is a set of m MRHS equations given by matrices $\mathbf{M}_1, \ldots, \mathbf{M}_m$ with the same number of rows n and corresponding *right hand side* (RHS) sets $S_1, \ldots, S_m$. Vector $\mathbf{x} \in \mathbb{F}^n$ is

a solution of the MRHS system if it is a solution of each MRHS equation in the system. When the internal structure of the MRHS system is not important, we will denote it simply by the symbol $\mathcal{M}$.

Solving MRHS systems is believed to be difficult (in general). We have already shown that it is an NP-complete problem to decide, whether an MRHS system has a solution [12]. There are multiple existing exponential time algorithms that can solve an MRHS system [9, 10, 11]. These algorithms mainly focus on an efficient search through the solution space similar to the DPLL algorithm for solving the SAT problem.

In this paper, we focus on sparse MRHS systems over a finite field $\mathbb{F} = GF(2)$.

Definition 2. Let us have a MRHS system consisting of m MRHS equations given by left-hand side matrices $\mathbf{M}_1, \ldots, \mathbf{M}_m$, and corresponding right hand side sets $S_1, \ldots, S_m$. We can write the MRHS system in a *joint form*

$$\mathbf{x} \cdot (\mathbf{M}_1 | \cdots | \mathbf{M}_m) \in S_1 \times \cdots \times S_m.$$

We call left-hand side matrix $(\mathbf{M}_1 | \cdots | \mathbf{M}_m)$ a joint matrix of the MRHS system.

Joint form of the MRHS system can be understood as a single MRHS equation defined by a (joint) matrix $\mathbf{M} = (\mathbf{M}_1 | \cdots | \mathbf{M}_m)$ on the left-hand side and a (large) right-hand side set $S = S_1 \times \cdots \times S_m$. We represent S as a Cartesian product, because the size of the whole set S grows exponentially with increasing m.

We say that the MRHS system is sparse if each column of its joint matrix contains $1 + o(1)$ non-zero elements. In our experiments, we require that all columns for each MRHS equation are linearly independent (otherwise its dimension will degenerate). This means that each column of the joint matrix of the MRHS system contains at least one non-zero element. The density of the joint matrix is defined as $d = N_1 - N_{col}$, where N_1 is the number of non-zero elements of the joint matrix $\mathbf{M}$, and N_{col} is the number of columns of $\mathbf{M}$.

We extend the MRHS system notation by adding a constant vector $\mathbf{c} = (\mathbf{c}_1 | \cdots | \mathbf{c}_m)$ to the left-hand side as follows:

$$\mathbf{x} \cdot (\mathbf{M}_1 | \cdots | \mathbf{M}_m) + (\mathbf{c}_1 | \cdots | \mathbf{c}_m) \in S_1 \times \cdot \times S_m. \tag{2}$$

Constants from the left-hand side can be moved to the right-hand side by subtracting its parts $\mathbf{c}_i$ from each vector in S_i. However, the addition of a constant to the left-hand side will allow us to represent an MRHS system with uniform right-hand side sets $S_i = S$ suitable for modeling a computational circuit with AND gates (sets S_i), XOR gates (joint matrix $\mathbf{M}$) and constants (constant c).

It is easy to check, whether vector $\mathbf{x}$ is a solution of the (extended) MRHS system from (2). We can compute vectors $\mathbf{u}_i = \mathbf{x} \cdot \mathbf{M}_i + \mathbf{c}_i$. Vector $\mathbf{x}$ is a solution of the MRHS system if and only if each $\mathbf{u}_i \in S_i$. If $\mathbf{x}$ is not a solution of the MRHS system, there exists at least one i such that $u_i \notin S_i$. It also means that even though $\mathbf{x}$ is not a solution of the MRHS system, there is a possibility that for some i, $u_i \in S_i$.

We say that unknown x_i *influences* RHS set S_j (or that S_j *is influenced by* x_i), if and only if there exists a non-zero element in the i-th row of $\mathbf{M}_j$. In sparse MRHS systems, each unknown influences only a limited number of RHSs, and vice-versa. E.g., if the density of the MRHS system is $d = 0$, and each $\mathbf{M}_i$ has dimension $n \times l$, each RHS set is influenced by (exactly) l unknowns, and each unknown influences l RHS sets on average. With density d, each RHS set is influenced by $l + d/n$ unknowns on average, and each variable influences $l + d/m$ RHS sets on average.

To simplify discussion, we say that RHS (set) S_i is *satisfied* (for some $\mathbf{x}$), if $\mathbf{u}_i \in S_i$, and we call RHS (set) S_i *unsatisfied*, if $\mathbf{u}_i \notin S_i$. We define function $\mathrm{UNSAT}(\mathbf{x}, \mathcal{M})$ to represent the number of unsatisfied RHS sets in the MRHS system $\mathcal{M}$ for vector $\mathbf{x}$, i.e.

$$\mathrm{UNSAT}(\mathbf{x}, \mathcal{M}) = \{i; \quad \mathbf{x} \cdot \mathbf{M}_i \notin S_i\}.$$

Note that if $\mathbf{x}$ is a solution of the MRHS system $\mathcal{M}$, $\mathrm{UNSAT}(\mathbf{x}, \mathcal{M}) = 0$. We investigate a generic class of algorithms that start from some random $\mathbf{x}$ and try to change its individual bits one by one in such a way, that we minimize the value of UNSAT function. We call them "bit-flipping" algorithms.

3. Modelling algebraic cryptanalysis with sparse MRHS systems

Our aim is to investigate a general behavior of the "bit-flipping" class of algorithms for solving sparse MRHS equations. We do not use completely random MRHS systems, but instead, we focus on a specific model related to algebraic cryptanalysis.

Instead of focusing on some specific cipher, we use a generic AND-XOR circuit (each gate has 2 inputs and one output), with some number of circuit inputs and outputs. We suppose that the attacker is given the outputs of the system and her task is to find the input of the system that produces such output. In our model, the fixed inputs (and other potential constants in the circuit) are treated as random constants $\mathbf{c}_i$.

We can solve the circuit problem by creating a corresponding MRHS system in such a way, that any solution of the circuit problem can be computed in polynomial time (with linear algebra) from a solution of the MRHS system.

The MRHS system is based on non-linear AND gates. Circuit inputs or their linear combinations, as well as selected internal values (typically the outputs of the AND gates) in the circuit, represent the unknown vector $\mathbf{x}$ of the MRHS system.

A circuit with m AND gates leads to a MRHS system with m MRHS equations of type $\mathbf{x} \cdot \mathbf{M}_i + \mathbf{c}_i \in S_{\text{AND}}$, where $S_{\text{AND}} = \{(0,0,0),(1,0,0),(0,1,0),(1,1,1)\}$. First two coordinates of vectors in S_{AND} correspond to AND gate inputs, and the last one corresponds to an AND gate output. The inputs and outputs of AND gates are expressed as linear combinations of unknowns (columns of $\mathbf{M}_i$).

Let some column of $\mathbf{M}_i$ have w non-zero elements. Such a linear combination can be realized by $w-1$ XOR gates. Thus, a sparse MRHS system with density d corresponds to a circuit with at most d XOR gates (note that the circuit can be realized with fewer XOR gates due to possible gate reuse).

In our experiments, we create random sparse MRHS systems with MRHS equations of the form $\mathbf{x} \cdot \mathbf{M}_i + \mathbf{c}_i \in S_{\text{AND}}$. Each $\mathbf{M}_i$ has 3 columns. Firstly, we ensure that each column of each $\mathbf{M}_i$ has one non-zero element (in random positions), and all 3 columns are linearly independent. Then, we randomly distribute d ones into random positions within the whole joint matrix. Finally, we generate random constants $\mathbf{c}_i$ in such a way, that the MRHS system has a solution (this is a typical use case for a bit-flipping algorithm application). Solution is ensured by generating a random vector $\mathbf{x}$ and selecting $\mathbf{c}_i$'s from sets $S_{\text{AND}} + \mathbf{x} \cdot \mathbf{M}_i$. Our parameter choice is $n = m$ also ensures that there is a single expected solution of the system. Note that such systems are considered the hardest to solve.

4. Bit-flipping algorithm for solving (sparse) MRHS systems

Let $\mathbf{s}$ be a solution of a sparse MRHS system $\mathcal{M}$ over $\mathbb{F}_2$ with m MRHS equations, n unknowns, and density d. Let $\mathbf{x}$ be a random vector from $\mathbb{F}_2^n$ with Hamming distance to $\mathbf{s}$ denoted by $w = w_H(\mathbf{x} + \mathbf{s})$. We show that in (sparse) MRHS system, value w is related to value $u = \text{UNSAT}(\mathbf{x}, \mathcal{M})$.

Equation $\mathbf{x} \cdot \mathbf{M}_i + \mathbf{c}_i \in S_{\text{AND}}$ is influenced (on average) by $3 + d/m$ unknowns. This equation is satisfied with probability 1, if each of the unknowns influencing it are correct, i.e., each $x_j = s_j$. Otherwise, it is satisfied with probability $1/2$

(4 vectors in S_{AND} out of 8 possible). Thus, each MRHS equation is satisfied with probability

$$Pr(\text{SAT}) = \frac{1}{2} + \frac{1}{2} \cdot \left(1 - \frac{w}{n}\right)^{3+\frac{d}{m}}.$$

If $w = 0$, this probability is 1, and $u = 0$. If w increases, $Pr(\text{SAT})$ decreases, and expected $u = m \cdot (1 - Pr(\text{SAT}))$ increases as well. The rate of growth of u as function w is largest for w small, and is near to 0 for w nearing n. If

$$\frac{w}{n} \geq 1 - \left(\frac{2}{3 + \frac{d}{m}}\right)^{\frac{1}{2+\frac{d}{m}}},$$

the expected change of w by 1 increases u by less than 1 on average (as $u'(w) \leq 1$). On the other hand, if we have successfully located some "near solution" with low enough w, we can quickly get a better approximation to the real solution s by changing those bits that decrease the number of unsatisfied equations in the MRHS system.

Algorithm 1 Generic MRHS bit-flipping

Require: MRHS system: $\mathbf{x} \cdot (\mathbf{M}_1 | \cdots \mathbf{M}_m) + (\mathbf{c}_1 | \cdots | \mathbf{c}_m) \in S_1 \times \cdots \times S_m$
Ensure: Solution $\mathbf{x}$ of the MRHS system, or $\perp$

```
x ←_R F^n                                   ▷ Initialize with random x.
while stopping condition not reached do
    for all i ∈ {1, 2, ..., m} do
        Compute u_i = x · M_i + c_i
        if u_i ∉ S_i then                   ▷ Unsatisfied RHS.
            Send bit-flipping information to each x_j influencing M_i.
        end if
    end for
    if all MRHS equations were satisfied then
        return x
    else if restart condition reached then
        x ←_R F^n
    else
        Select bit(s) of x to flip, based on bit-flipping information.
    end if
end while
return ⊥                                    ▷ Solution not found.
```

This analysis leads us to a general idea of a bit-flipping algorithm presented as Algorithm 1. We start from a random point $\mathbf{x}$ and compute values $\mathbf{u}_i = \mathbf{x} \cdot \mathbf{M}_i + \mathbf{c_i}$. If some $\mathbf{u}_i \notin S_i$, the i-th MRHS equation is unsatisfied. We should mark each variable that can influence the value of such an unsatisfied equation as a potential bit to flip (change value of the guessed variable). We evaluate information from each MRHS equation and either find that the system is solved or decide to change some specific bit or restart the process altogether with some other starting $\mathbf{x}$.

We have instantiated this generic scheme with multiple concrete algorithms based on different information propagated from unsatisfied MRHS equations and bit flipping selection strategies. One of the most successful strategies was an algorithm we called "Weighted flip", presented as Algorithm 2.

Algorithm 2 Bit-flipping algorithm "Weighted Flip"

Require: MRHS system: $\mathbf{x} \cdot (\mathbf{M}_1 | \cdots \mathbf{M}_m) + (\mathbf{c}_1 | \cdots | \mathbf{c}_m) \in S_1 \times \cdots \times S_m$
Ensure: Solution $\mathbf{x}$ of the MRHS system, or $\perp$
  $NumIterations \leftarrow 0$
  $\mathbf{x} \leftarrow_R \mathbb{F}^n$ ▷ Initialize with random $\mathbf{x}$.
  while $NumIterations < MaxIterations$ **do**
    $NumIterations \leftarrow NumIterations + 1$
    $Solved \leftarrow True$
    for all $j \in \{1, 2, \ldots, n\}$ **do**
      $w_j \leftarrow 0$ ▷ Reset weights for flipping bits.
    end for
    for all $i \in \{1, 2, \ldots, m\}$ **do**
      Compute $\mathbf{u}_i = \mathbf{x} \cdot \mathbf{M}_i + \mathbf{c_i}$
      if $\mathbf{u}_i \notin S_i$ **then** ▷ Unsatisfied RHS.
        $Solved \leftarrow False$
        for all j such that $\mathbf{M}_{i,j} = 1$ and $(\mathbf{x} + \mathbf{e}^j) \cdot \mathbf{M}_i + \mathbf{c_i} \in S_i$ **do**
          $w_j \leftarrow w_j + 1$ ▷ Add weight to bits influencing given RHS.
        end for
      end if
    end for
    if $Solved$ **then return** $\mathbf{x}$ ▷ Each RHS satisfied.
    else
      Choose j randomly with probability $\frac{w_j}{\sum_{i=1}^{n} w_i}$. ▷ Weighted selection.
      $\mathbf{x}_j \leftarrow \mathbf{x}_j + 1$ ▷ Flip the selected bit.
    end if
  end while
  return $\perp$ ▷ No solution found within the given number of iterations.

The "Weighted flip" algorithm is based on a strategy of only flipping those bits that can change the state of the MRHS equation from unsatisfied to satisfied. This is expressed by condition $(\mathbf{x}+\mathbf{e}^j)\cdot\mathbf{M}_i+\mathbf{c_i} \in S_i$, but can be quickly computed according to table 1.

$\mathbf{u_i}$	x_{j_1}	x_{j_2}	x_{j_3}
001	0	0	1
011	1	0	1
101	0	1	1
110	1	1	1

Table 1. "Weighted flip" algorithm: Possible bits to flip based on the value of $\mathbf{u}_i$.

If only one equation is unsatisfied, each bit will accumulate

- either weight 0: changing its value will not correct the unsatisfied equation;
- or weight 1: changing its value will correct the unsatisfied equation.

We can change any of the bits with weight 1, and the equation will become satisfied. However, this does not guarantee that we have obtained a solution. In such a case, some other equation must become unsatisfied, and we repeat the process with new weights based on the new unsatisfied equation.

If multiple equations are unsatisfied, some bits can get a higher weight than 1. In this case, changing their value improves more than one MRHS equation. In the "Weighted flip" algorithm, we randomly select any bit with non-zero weight. The probability of a particular bit selection is proportional to the total accumulated weight. Variables that can improve 2 MRHS equations are selected twice as likely as those that can only improve 1 MRHS equation, and so on.

In the experimental section, we compare the "Weighted flip" selection with two different strategies. The first one chooses only those bits that can improve the maximum number of MRHS equations at once ("Flip maximum"). Firstly, we find the maximum weight w_m. Then we choose randomly from only those variables that attain weight w_m. Our experiments show that this strategy suffers from the problem of cyclical wrong assignment: we flip the wrong variable back and forth (or some subset of variables in a cycle), never reaching the solution of the system. To mitigate the problem of the cycles, we have defined the strategy "Flip maximum restarts". In this strategy, we restart the algorithm if a cycle of length two is detected: the only suggested candidate is the bit changed in the previous iteration.

5. Hill climbing algorithm for solving (sparse) MRHS systems

The general bit-flipping algorithm (Algorithm 1) is based on propagating information from unsatisfied MRHS equations to individual variables influencing those MRHS equations. As it is formulated, it does not take into account the possible negative influence bit flipping might have on satisfied MRHS equations.

We can take a different approach: Investigate the influence of each bit flip on the whole system, and change bits accordingly. Each candidate for solution $\mathbf{x}$ can be changed in n positions to value $\mathbf{x} + \mathbf{e}^i$ (*successor* of $\mathbf{x}$). We want to select a sequence of bit flips, that will minimize the number of unsatisfied MRHS equations. If we reach some vector $\mathbf{s}$ for which $UNSAT(\mathbf{s}, \mathcal{M}) = 0$, we have solved the system. We are working with a case of optimization algorithm, that tries to find the minimum of the function $UNSAT(\mathbf{x}, \mathcal{M})$ over the space $\mathbb{F}_2^n$.

Algorithm 3 Hill climbing algorithm for MRHS problem

Require: MRHS system $\mathcal{M}$: $\mathbf{x} \cdot (\mathbf{M}_1 | \cdots \mathbf{M}_m) + (\mathbf{c}_1 | \cdots | \mathbf{c}_m) \in S_1 \times \cdots \times S_m$
Ensure: Solution $\mathbf{x}$ of the MRHS system, or $\perp$

```
x ←_R F^n                                   ▷ Initialize with random x.
NumIterations ← 0
while NumIterations < MaxIterations do
    Compute c_0 ← Unsat(x, M)
    if c_0 = 0 then
        return x                            ▷ Every MRHS satisfied.
    end if
                ▷ Try to minimize number of unsatisfied MRHS equations.
    for all i ∈ {1, 2, ..., n} do
        Compute c_i ← Unsat(x + e^i, M)
    end for
    Choose j such that c_j ≤ c_i for each i ∈ {1, 2, ..., n}
    if c_j ≥ c_0 then
        x ←_R F^n                           ▷ No improvement possible, restart.
    else
        x_j ← x_j + 1                       ▷ Flip bit.
    end if
    NumIterations ← NumIterations + 1
end while
return ⊥          ▷ No solution found within the given number of iterations.
```

One of the simplest solutions is to use the greedy approach: Select (one of) the successor which minimizes the value of UNSAT, restart with random $\mathbf{x}$ if we are stuck in local minimum (no successor provides a lower value of UNSAT). This is an instance of the Hill climbing heuristic and is summarized as Algorithm 3.

Note that the Hill climbing (HC) algorithm is in principle similar to the "Flip maximum" (FM) algorithm with these main differences:

- HC takes into account also information from satisfied MRHS equations, as it computes the expected value of UNSAT function after each possible bit flip;
- HC does not get stuck in a cycle of bit-flips because if no improvement is possible, we restart the algorithm;
- HC requires n-times more MRHS evaluations per each bit flip. FM (and other instances of generic bit-flipping) evaluate MRHS equation only in one point $\mathbf{x}$, whereas HC requires to evaluate also each $\mathbf{x} + \mathbf{e}^i$.

We compare the Hill climbing algorithm with selected single-evaluation bit flipping algorithms in the experimental section.

Note that there are multiple different heuristic algorithms successfully used in cryptanalysis [3], [4]. A more advanced algorithm can improve the success chance based on the number of bit-flips even further but might increase the cost of every individual bit-flip assessment. On the other hand, our experiments show that faster bit-flip assessment might be preferable, and a simple bit-flipping-based attack can find a solution faster (in real-time) than with a more complex method.

6. Experimental results

Our exploration of the bit flipping algorithms focuses on a simple model of sparse MRHS equations related to algebraic cryptanalysis summarized in Section 3. In the experiments, we generate 100 random sparse MRHS systems according to our model and execute 100 randomized instances of the evaluated algorithm on each system. We measure how many bit flips were required to solve the system (bounded at 1000 bit-flips, after this value, we stopped the algorithm). Each experiment provided us with 10000 results, which we used to plot the success rate of the algorithm per number of bit flips for a given configuration. In the case of the Hill climbing algorithm, we also use a re-scaling to the number of MRHS evaluations, with the number of MRHS evaluations computed as n-times the number of the bit flips (technically, in our algorithm it is $(n + 1)$-times, but it is possible to reuse one of the values for the next iteration).

For a fair comparison, we also include the results from a "Random" algorithm: It tries to find a solution by random guesses (try random $\mathbf{x}$, stop if it is a solution). "Random" algorithm provides a better reference for experimental results (instead of a simple theoretical estimate), as it takes into account specifics of the randomly generated systems we tried to solve (e.g., the fact that some of them have multiple solutions).

Results of the "Weighted flip" (WF) algorithm are plotted in Figure 1. It has a relatively low success rate for the small number of iterations but can solve almost all sparse equations with a large enough number of iterations. It is suitable for systems with low density (for dense systems, see WF-10-120, it is worse than random search). It also scales well with the system size. Sparse ($d = 0$) MRHS system with $n = 30$ variables could be on average solved with a comparable effort to a random search of system with only 10 variables.

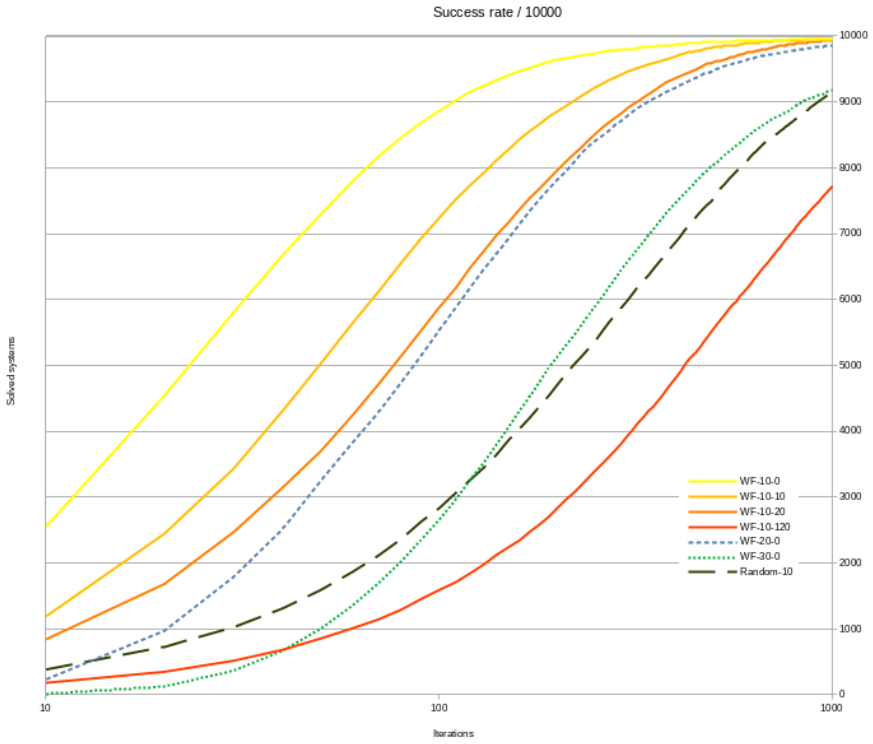

Figure 1. "Weighted Flip" algorithm, success rate per number of bitflips. System size for WF-X-Y was $m = n = X$, with density Y.

The "Flip maximum" (FM) algorithm performance is depicted in Figure 2. While it has a better success rate for very sparse systems ($d = 0$) and a low number of iterations, it quickly reaches its peak success rate. After this, the algorithm typically gets stuck in a loop of bit-flips.

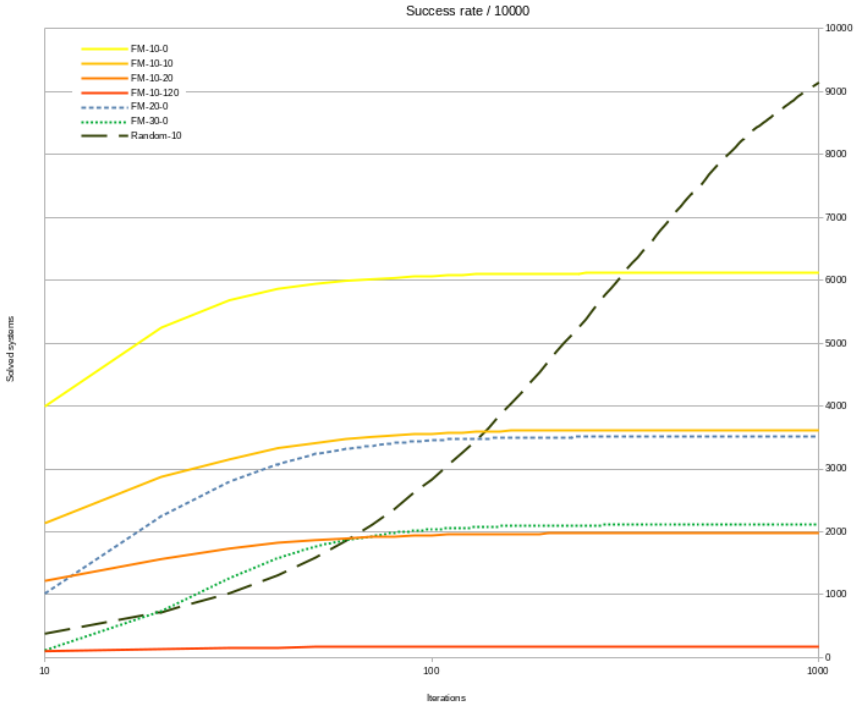

Figure 2. "Flip Maximum" algorithm, success rate per number of bitflips. System size for FM-X-Y was $m = n = X$, with density Y.

FM algorithm can be significantly improved with restart strategy, see the results of the "Flip maximum restarts" (FMR) algorithm plotted in Figure 3. It has the best success probability for a low number of iterations (among the explored bit-flipping algorithms). However, its overall success rate for a large number of iterations is lower than WF, and it scales worse with increasing system size and density. We suspect the FMR algorithm could be improved by more advanced cycle finding algorithms (for the restart condition) but at a higher computational cost.

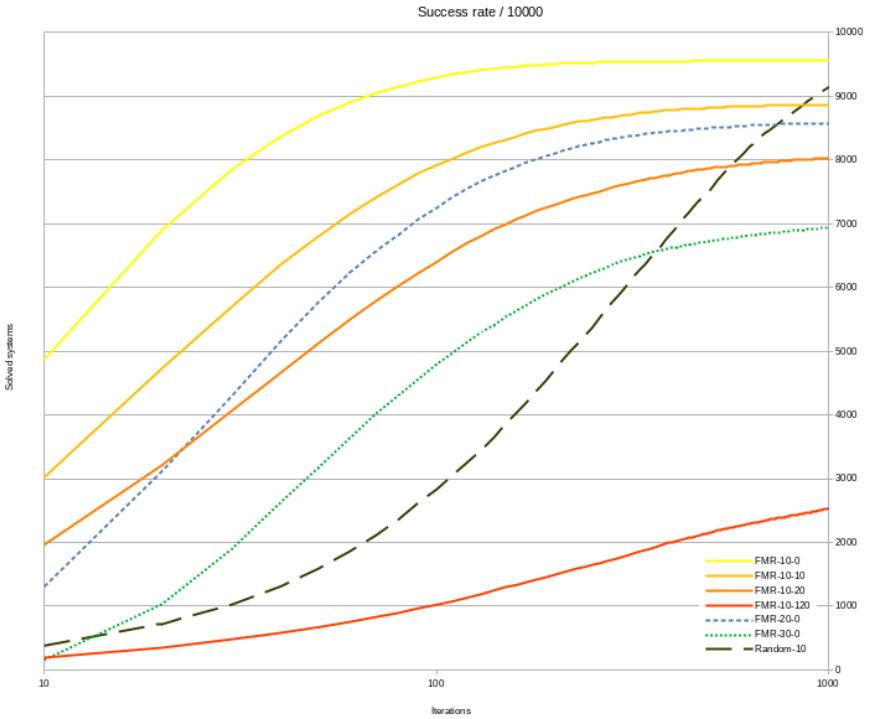

Figure 3. "Flip Maximum Restarts" algorithm, success rate per number of bitflips. System size for FMR-X-Y was $m = n = X$, with density Y.

The Hill climbing (HC) algorithm (see Figure 4) has a significantly higher success rate for both low and high iteration numbers. It is also less affected with increased system size and density. On the other hand, Figure 4 does only take into account the number of bit flips and not the complexity of the algorithm.

Figure 5 scales the results of the HC algorithm to account for factor n in the complexity of evaluating individual bitflips. However, the comparison with the "Random" algorithm here is misleading, as the true "Random" algorithm needs to realize whole vector-matrix multiplication in each iteration, which is comparable to n evaluations required in the HC algorithm. On the other hand, we can replace the "Random" algorithm with the "Random bit-flip" algorithm, where a single bit changes randomly in each iteration. Alternatively, we can find solutions more efficiently by exhaustive search using Grey codes, which also requires only a single vector addition and MRHS evaluation in each step. Thus, we can see that the HC algorithm is still an effective improvement over the exhaustive search, but only for sparse MRHS systems (with low density).

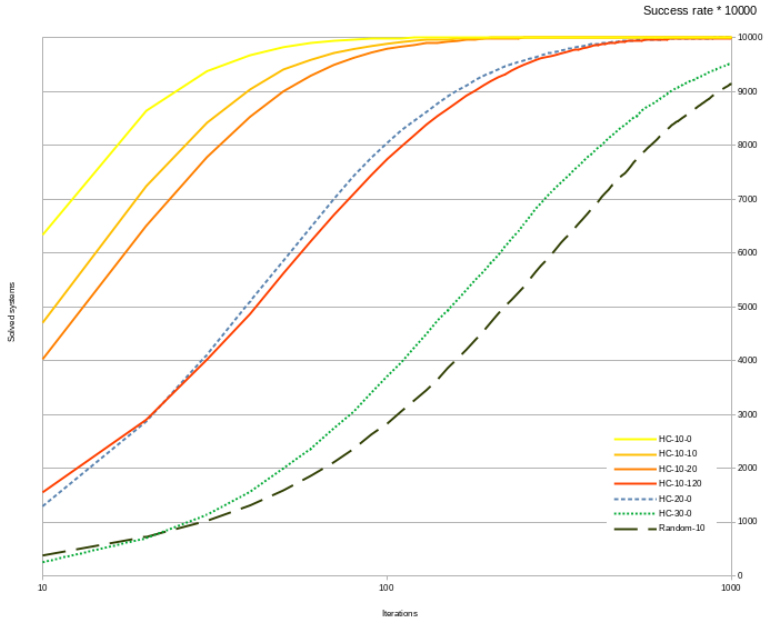

Figure 4. "Hill Climbing" algorithm, success rate per number of bitflips. System size for WF-X-Y was $m = n = X$, with density Y.

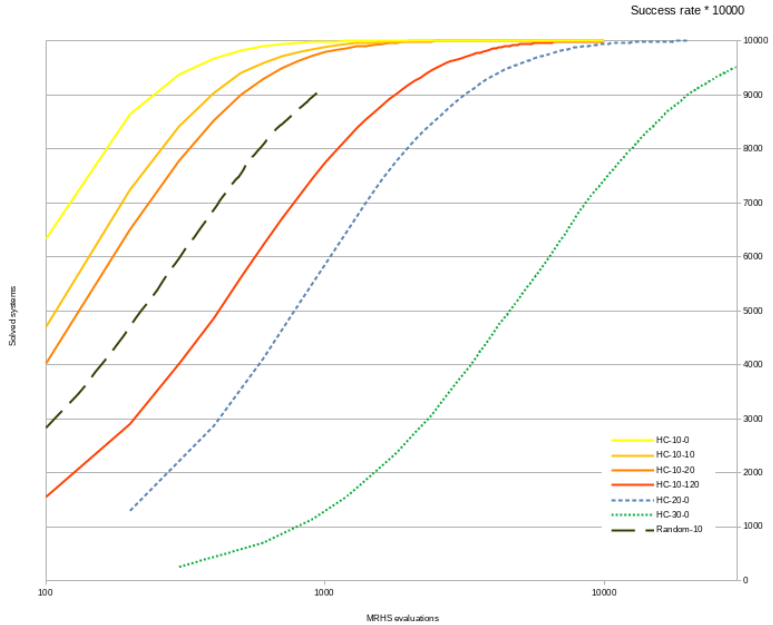

Figure 5. "Hill Climbing" algorithm, success rate per number of MRHS evaluations. System size for WF-X-Y was $m = n = X$, with density Y.

The final Figures 6 and 7 provides a comparison of the three most successful algorithms: "Weighted flip" (WF), "Flip maximum with restarts" (FMR) and "Hill climbing" (HC). On Figure 6, we also add "Flip maximum" (FM) and "Random" algorithms for a full comparison. Figure 7 shows a comparison of success rates based on the number of MRHS evaluations. We can see that bit-flipping algorithms outperform the hill-climbing algorithm due to a very efficient evaluation of the bit-flipping decision. We suppose that the most successful algorithm would be a combination of FMR and WF: start with FMR, and when the number of iterations reaches a certain threshold without finding a solution (or maximum weight is below some predetermined limit), switch to the WF algorithm.

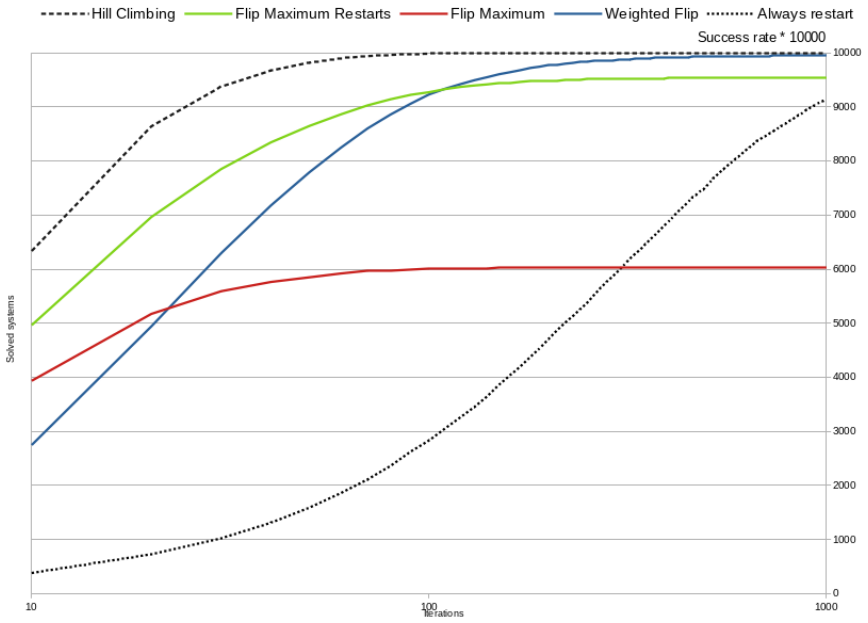

Figure 6. Comparison of different bit-flipping strategies, and hill climbing algorithm. Success rate per number of bitflips. System size $m = n = 10$, density 0.

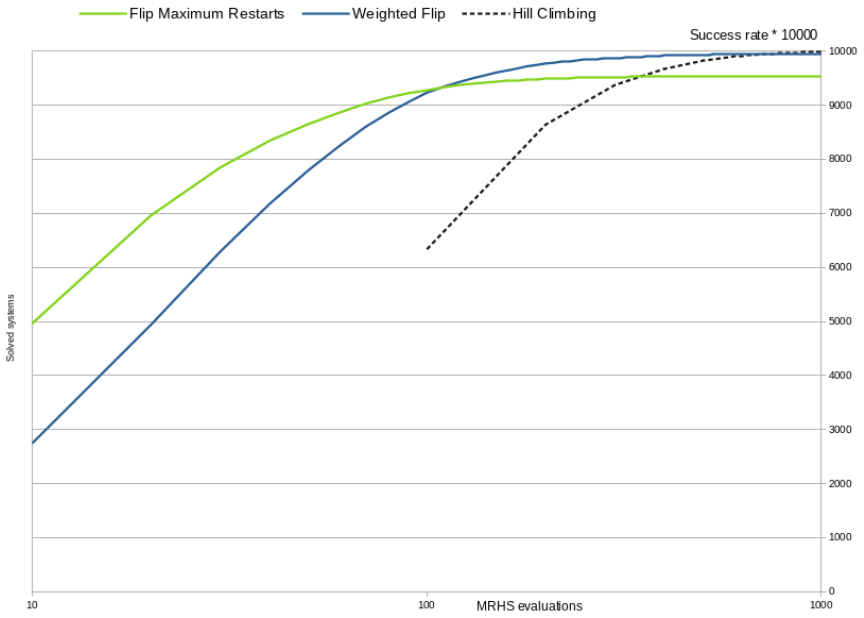

Figure 7. Comparison of different bit-flipping strategies, and hill climbing algorithm. Success rate per number of MRHS evaluations. System size $m = n = 10$, density 0.

7. Conclusions

In this article, we have presented a new generic algorithm (Algorithm 1) to solve (sparse) MRHS equations. We have experimentally explored the performance of some of its instantiations (WF, FM, FMR) against random search and a more advanced hill-climbing algorithm. The experimental results indicate that for sparse enough systems bit-flipping algorithms can significantly outperform random search. E.g., the success rate of the WF algorithm applied to a system with 30 variables is comparable to a random search for a solution in a system with only 10 variables.

On the other hand, our model is based on an artificial random AND-XOR circuit with a low number of both AND and XOR gates. It is not clear how the results would change if we apply the algorithms to structured cipher designs, such as iterated block ciphers with high diffusion. However, unlike other heuristic methods, the bit-flipping approach is based on a local inconsistency (based on

internal bits), instead of trying to assess specific key bits, which influence a large part of the encryption algorithm. As such, the performance of our algorithms can be improved by hints provided by side-channel analysis but does not scale well if the number of internal variables grows quickly with the number of rounds.

The presented general algorithmic scheme can be instantiated by multiple bit-flipping and restart strategies. Our article was focused on formulating the general idea and experimental evaluation of different bit flipping strategies. Finding the optimal strategy for a specific class of MRHS systems and computing the exact complexity of such an algorithm is left as an open problem.

References

[1] M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy and T. Tiessen, MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity, In: International Conference on the Theory and Application of Cryptology and Information Security, 2016, 191–219.

[2] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen and M. Zohner, Ciphers for MPC and FHE, In: Advances in Cryptology–EUROCRYPT, 2015, 430–454.

[3] E. Antal, Nature-inspired heuristic methods in classical cipher cryptanalysis, In: Norwegian–Slovakian Workshop in Crypto, 2016, 21.

[4] E. Antal, P. Javorka and Hliboký, Cryptanalysis of the columnar transposition using meta-heuristics, *Tatra Mountains Mathematical Publications* **73** (2019), 39–60.

[5] M. Chase, D. Derler, S. Goldfeder, J. Katz, V. Kolesnikov, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, X. Wang and G. Zaverucha, The Picnic signature algorithm, *Microsoft* (2020), https://github.com/microsoft/Picnic/raw/master/spec/spec-v3.0.pdf.

[6] N. Courtois, D. Hulme, and T. Mourouzis, Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis, *IACR Cryptol. ePrint Arch.* (2011), https://eprint.iacr.org/2011/475.pdf.

[7] D. Goudarzi, and M. Rivain, On the multiplicative complexity of boolean functions and bitsliced higher-order masking, In: International Conference on Cryptographic Hardware and Embedded Systems, 2016, 457–478.

[8] P. Méaux, C. Carlet, A. Journault and F. Standaert, Improved filter permutators: Combining symmetric encryption design, boolean functions, low complexity cryptography, and homomorphic encryption, for private delegation of computations, In: Proceedings of INDOCRYPT, 2019.

[9] H. Raddum and I. Semaev, Solving Multiple Right Hand Sides linear equations, *Design, Codes and Cryptography* **49** (2008), 147–160.

[10] H. Raddum and P. Zajac, MRHS solver based on linear algebra and exhaustive search, *Journal of Mathematical Cryptology* **12** (2018), 143–157.

[11] P. Zajac, A new method to solve MRHS equation systems and its connection to group factorization, *Journal of Mathematical Cryptology* **7** (2013), 367–381.

[12] P. Zajac, MRHS equation systems that can be solved in polynomial time, *Tatra Mountains Mathematical Publications* **67** (2016), 205–219.

[13] P. Zajac, Connecting the complexity of MQ-and code-based cryptosystems, *Tatra Mountains Mathematical Publications* **70** (2017), 163–177.

PAVOL ZAJAC
DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
ILKOVIČOVA 3, 81219 BRATISLAVA
SLOVAKIA

E-mail: pavol.zajac@stuba.sk