Published in final edited form as:

Inf Process Lett. 2018 ; 137: . doi:10.1016/j.ipl.2018.04.010.

Improved upper bounds for the expected circuit complexity of dense systems of linear equations over GF(2)

Andrea Visconti[a], **Chiara Valentina Schiavo**[a], and **René Peralta**[b]

[a]Università degli Studi di Milano, Department of Computer Science, via Comelico 39/41, 20135, Milano, Italy

[b]National Institute of Standards and Technology, 100 Bureau Dr, Gaithersburg, MD, United States

Abstract

Minimizing the Boolean circuit implementation of a given cryptographic function is an important issue. A number of papers [1], [2], [3], [4] only consider cancellation-free straight-line programs for producing small circuits over GF(2). Cancellation is allowed by the Boyar-Peralta (*BP*) heuristic [5, 6]. This yields a valuable tool for practical applications such as building fast software and low-power circuits for cryptographic applications, e.g. AES [5, 7], HMAC-SHA-1 [8], PRESENT [9], GOST [9], and so on. However, the *BP* heuristic does not take into account the matrix density. In a dense linear system the rows can be computed by adding or removing a few elements from a “common path” that is “close” to almost all rows. The new heuristic described in this paper will merge the idea of “cancellation” and “common path”. An extensive testing activity has been performed. Experimental results of the new and the *BP* heuristic were compared. They show that the Boyar-Peralta results are not optimal on dense systems.

Keywords

Gate complexity; linear systems; dense matrices; circuit depth; XOR gates

1. Introduction

Circuits are important in many areas of computer science, including computer architecture and engineering, cryptography and computer security, and privacy-preserving multiparty computations. Minimizing the total number of gates in the Boolean circuit implementation of a given function *f* can lead to high-speed software as well as low-power hardware for *f*. Particularly important are hardware optimizations of cryptographic circuits. The speed and power consumption are often a limiting constraint in security chips — e.g., RFID, smart cards, TPMs.

Circuits for linear functions can be represented as *linear straight-line programs* (SLPs). These are sequential programs (see [10] and [11], for example) in which the instructions are of the form $X_i = X_j + X_k$ where

Email addresses: andrea.visconti@unimi.it (Andrea Visconti), chiara.schiavo@gmail.com (Chiara Valentina Schiavo), rene.peralta@nist.gov (Ren´e Peralta).

- X_t has not appeared before in the program;
- X_j *and* X_k are either inputs or have appeared before in the program;
- "+" denotes Boolean exclusive-or.

The *shortest SLP problem* is to find the shortest linear program which computes a set of linear functions over a field. Solving the shortest SLP problem over $GF(2)$ corresponds to finding a gate-optimal Boolean circuit that computes the linear functions. This problem is known to be MAX SNP-hard [6]. This means that, unless P=NP, there is no efficient algorithm that can compute solutions that are arbitrarily close to optimal. In [5], it is shown that known polynomial-time heuristics do quite poorly on random $n \times n$ systems of equations, and an exponential-time heuristic is described which does significantly better and is fast enough to be used in many practical situations. The Boyar-Peralta (BP) heuristic [5] has been successfully applied to a number of circuit optimization problems of interest to cryptology. These include a compact implementation of Present S-Box [9], HMAC-SHA-1 optimizations [8], finite-field arithmetic and binary multiplication [10] and [12].

A *random* $m \times n$ *linear system* is constructed as follows: given a *density* $0 < \rho < 1$, construct an $m \times n$ binary matrix by placing a 1 in position i, j of the matrix with probability ρ. Each row of the resulting matrix is interpreted as the sum of variables (columns) containing a 1. We will call these rows *targets*. There are no known tight combinatorial bounds for the gate complexity of random linear systems. An obvious upper bound is $O(mn)$, the only non-trivial combinatorial upper bound we are aware of is $O\left(\frac{mn}{\log m}\right)$ This can be derived from a lemma by Lupanov [13] about matrix decompositions.

A number of papers [1], [2], [3], and [4] only consider cancellation-free straight-line programs for producing small cryptographic circuits over GF(2). In 2009, Boyar and Peralta show that these circuits can be improved in a model that is not restricted to producing cancellation-free circuits [5], [10], [14]. However, the BP heuristic does not take into account that rows of a dense linear system have a long path of elements in common and, allowing cancellations, these rows can be easily computed by adding or removing a few elements from a "common path". In this paper, we present a new heuristic for constructing circuits that evaluate dense linear systems. In particular, our heuristic has been developed taking into account the possibility to (a) work in a model that is not restricted to producing cancellation-free circuits, and (b) add/remove a few elements from a "common path". We conducted extensive testing on random systems for evaluating the performance of our heuristic. Experimental results show that the new heuristic outperforms (on average) the BP heuristic, when applied to random dense linear systems.

2. The Boyar-Peralta heuristic

The BP heuristic [5] is for optimizing arbitrary circuits. The first step is to minimize the number of AND gates in the circuit. This typically results in a circuit with large linear connected components. The second step optimizes the linear components. We briefly describe their technique for this second step.

Let $f(\mathbf{x}) = M\mathbf{x}$, where $\mathbf{x} = [x_1, \ldots, x_n]$ is a vector of input variables and M is an $m \times n$ matrix with coefficients over $GF(2)$. We denote with y_i the i^{th} row of the matrix

M. Let S be the set of "known" linear functions. The members of S are called *base* elements. S initially contains the variables $x_1,\ldots,x_n$. Given a linear predicate g, $\delta(S, g)$ is defined as the minimum number of additions of elements from the set S necessary to compute g. The vector $D[i] = \delta(S, y_i)$ is called the *distance* from S to M. At the beginning of the computation $D[i]$ is one less than the Hamming weight of the i^{th} row of M. The following loop is performed until $D[] = \mathbf{0}$:

- create a new base element by adding two base elements in S;
- update S and the vector $D[]$.

The choice of new base element is performed by picking a base which minimizes the sum of elements of the updated $D[]$ vector. Ties are solved by maximizing the Euclidean norm of the new distance vector.

3. New heuristic

Let $\mathbf{y} = f(\mathbf{x}) = M\mathbf{x}$ where M is an $m \times n$ matrix with coefficients in $GF(2)$. We would like to find a small circuit which computes $\mathbf{y}$ given an input vector $\mathbf{x} = [x_1,\ldots,x_n]$. We consider the problem space consisting of random matrices in which elements $A[i, j]$ are Bernoulli trials. We call these matrices *dense* when $prob(A[i, j] = 1) \geq 0.6$.

Given a circuit C, a *signal* computed by C is either an input to the circuit or the output of any gate in the circuit.

When M is dense, its Boolean complement $\overline{M}$ is sparse. The naive approach with a dense matrix is to compute the complement of the matrix M and then to apply the BP heuristic. Hence, it is appealing to try the following steps:

i. **(i)** use the *BP* heuristic to find a small circuit that implements $\overline{M}_{\mathbf{X}}$;

ii. **(ii)** use signals computed in (i) to compute a "common path" $yy = \sum_{i=1}^{n} x_i$;

iii. **(iii)** at a cost of m additional gates, add the signal yy to each of the outputs of the circuit computed in step (i).

We have experimentally verified that this heuristic, as well as several variations, yield circuits with more gates than does the *BP* heuristic. The naive approach fails because the base elements chosen in (i) do not guarantee the reachability of the "common path" in few steps. Therefore, the new heuristic first computes the "common path" by picking the base elements that may not necessarily minimize the sum of elements of the distance vector. Then, all targets are computed by allowing cancellations from it. Below we describe the method that did improve over *BP*.

Let $\mathbf{y} = \{y_1,...,y_m\}$ be the set of rows of M (we call these *targets*). We will keep track of two distance vectors D (distance from S to M) and D^* (distance from S to the Boolean complement of M). The heuristic is as follows:

1. 1. (Initialization) Set S to the set of variables $x_1,...,x_n$. For $i = 1,...,m$, set $D[i] = HammingWeight(y_i) - 1$.
2. 2. (Create complement instance) Let $\mathbf{y}^* = \{y_1^*, \ldots, y_m^*\}$ be the set of complement targets $(\text{i.e.} \quad y_i^* = \bar{y}_i)$ Add target $y_{m+1}^* = [1, \ldots, 1]$ to the set of complement targets (note y_{m+1}^* encodes the function $yy = \sum_{i=1}^{n} x_i$). Let $M^* = [y_1^*, \ldots, y_m^*, y_{m+1}^*]^T$ Let D^* be the distance vector for M^*, initialized to $D^*[i] = HammingWeight\,(y_i^*)$ for $i = 1,...,m + 1$.
3. 3. (Compute the common path) Until target y_{m+1}^* is found — i.e., until $D^*[m+1] - 1$; — pick a new base element x_i, $i = |S| + 1$, by adding two existing base elements such that:
    a. (a) x_i decreases the distance to y_{m+1}^* by one — i.e., to $D^*[m + 1] - 1$;
    b. (b) x_i minimizes the sum of distances D^* under the restriction (a).

Output the SLP instruction that computes x_i. Update distance vectors D and D^*. Add x_i to S.

1. 4. (Allow cancellations) Apply the *BP* heuristic to matrix M, but skipping the initialization steps for S and D.

Note that S and D are well-defined at step 4, as they have been continuously updated every time an SLP instruction is output. We resolve ties at step 3 by maximizing the Euclidean norm of the vector D^*. The "common path" of our heuristic is the target $y_{m+1}^* = [1, \ldots, 1]$

3.1. A toy example

To understand the details of this new heuristic, we present a toy example. Let $y_1, \ldots, y_m$ ($m = 6$) be the set of rows of:

$$\begin{cases} y_1 = x_1 + x_2 + x_3 \\ y_2 = x_2 + x_4 + x_5 \\ y_3 = x_1 + x_3 + x_4 + x_5 \\ y_4 = x_2 + x_3 + x_4 \\ y_5 = x_1 + x_2 + x_4 \\ y_6 = x_2 + x_3 + x_4 + x_5 \end{cases}$$

Step 1: Initialization.—The initial basis vector is $S = \{x_1,x_2,x_3,x_4,x_5\}$, the target is $\mathbf{y} = \{y_1,y_2,y_3,y_4,y_5,y_6\}$, the distance vector is $D = [2,2,3,2,2,3]$.

Step 2: Create a complement instance.—We generate the "common path" $yy = x_1 + x_2 + x_3 + x_4 + x_5$, the new target is $\mathbf{y}^* = \{\bar{y}_1, \bar{y}_2, \bar{y}_3, \bar{y}_4, \bar{y}_5, \bar{y}_6, yy\}$ the new distance vector is $D^* = [1, 1, 0, 1, 1, 0, 4]$, and the new matrix is

$$M^* = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Step 3: Compute the common path.—We compute y^*_{m+1}, also called yy or "common path".

- $x_7 = x_1 + x_3$.
- $D = [1, 2, 2, 2, 2, 3]$ $D^* = [1, 0, 0, 1, 1, 0, 3]$
- $x_8 = x_4 + x_5$.
- $D = [1, 1, 1, 2, 2, 2]$ $D^* = [0, 0, 0, 1, 1, 0, 2]$
- $x_9 = x_2 + x_7$. Found target $y_1 = x_9$.
- $D = [0, 1, 1, 2, 2, 2]$ $D^* = [0, 0, 0, 1, 1, 0, 1]$
- $x_{10} = x_8 + x_9$. Found target $yy = x_{10}$.
- $D = [0, 1, 1, 2, 2, 1]$ $D^* = [0, 0, 0, 1, 1, 0, 0]$

Step 4: Allow cancellations.—We apply the *BP* heuristic to matrix *M*. The distance vector is $D = [0, 1, 1, 2, 2, 1]$, the basis vector is $S = \{x_1,...,x_{10}\}$ and the new distance vector D^* is no more updated.

- $x_{11} = x_1 + x_{10}$. Found target $y_6 = x_{11}$.
- $D = [0, 1, 1, 1, 2, 0]$ $S = \{x_1,...,x_{11}\}$
- $x_{12} = x_2 + x_8$. Found target $y_2 = x_{12}$.
- $D = [0, 0, 1, 1, 2, 0]$ $S = \{x_1,...,x_{12}\}$
- $x_{13} = x_2 + x_{10}$. Found target $y_3 = x_{13}$.
- $D = [0, 0, 0, 1, 2, 0]$ $S = \{x_1,...,x_{13}\}$
- $x_{14} = x_5 + x_{11}$. Found target $y_4 = x_{14}$.
- $D = [0, 0, 0, 0, 1, 0,]$ $S = \{x_1,...,x_{14}\}$
- $x_{15} = x_7 + x_{14}$. Found target $y_5 = x_{15}$.
- $D = [0, 0, 0, 0, 0, 0]$ $S = \{x_1,...,x_{15}\}$

4. Experimental results

We conducted extensive testing to gauge the performance of our new heuristic, against that of *BP*. Experiments were performed on square and non-square matrices (more details can be found in [15]). Due to space limitations, this paper only discusses our results on square matrices. The useful conclusions drawn are also valid for rectangular matrices. Although we are able to solve systems larger than 30 × 30, we limited our experiments to size 30 due to the exponential time complexity of both heuristics.

1.1. Gate Count

We generated several $n \times n$ matrices [16], $n = 15, 16, \ldots, 30$, for biases ρ. For each size n and each bias ρ, we randomly pick 100 matrices from our benchmark set, hence we tested 9600 matrices.

Circuits for $M\mathbf{x}$ were constructed for each matrix M using *BP* and our heuristic. We identified four matrix size thresholds, one for each of the bias values $\rho = 0.6, 0.7, 0.8, 0.9$, beyond which the new heuristic performs on average better than the old one. Our experiments also suggest that there exists a lower bound ρ_L for the bias beyond which it is convenient to use the new heuristic on large enough matrices. Of course, at a cost of roughly doubling the running time, one can run both heuristics and pick the best circuit.

Bias $\rho = 0.4$ and 0.5. Experimental results show that the average number of XOR gates computed by the new heuristic is, on average, worse than those computed by the old one. As expected, the new heuristic does not perform well on sparse matrices. However, over 3200 matrices handled, the new heuristic gets better results in 577 cases — i.e. 211 when $\rho = 0.4$ and 366 when $\rho = 0.5$. This means that, as expected, the *BP* heuristic sometimes fails to find the best solution.

Bias $\rho = 0.6$. When the bias grows, the new heuristic behaves better than the old one. Experimental results suggest that the difference between the average number of XOR gates computed by the two heuristics gradually increases with the increasing size of the matrix. In particular the new heuristic will perform better than *BP* when applied to large-enough matrices of density 0.6. The threshold over which the new heuristic performs as well as or better than *BP* lies between 20 × 20 and 22 × 22.

Bias $\rho = 0.7$. The new heuristic performs, on average, better than the old one. For 16 × 16 matrices the new heuristic gets the best, or the same, solution in 73% of cases. This value grows up to 98% for 30 × 30 matrices.

Bias $\rho = 0.8$. In this case the new heuristic performs better compared to BP when $\rho = 0.7$. In fact, it beats *BP* on matrices as small as 15 × 15. For 16 × 16 matrices we get the best, or the same, solution in 84% of cases, while for matrices larger than size 24 × 24, this percentage is greater or equal to 98%.

Bias $\rho = 0.9$. In this case, the behavior of the new heuristic is similar to that of the $\rho = 0.7$ and 0.8 cases. However, the threshold is higher — i.e., around size 20 × 20 — and the

observed probability of the new heuristic beating *BP* on matrices larger than size 20 × 20 is between 0.70 and 0.96.

Figure 1 visualizes the output data collected, showing the difference between the average number of XOR gates required by the new heuristic and *BP*. Negative values indicate that the new heuristic performs better than *BP*, while positive values indicate it performs worse. This data can help us identify when the new heuristic is expected to provide the best results for specific values of ρ and n. Therefore it is possible to identify a lower bound ρ_L that indicates a threshold beyond which it is convenient to use the new heuristic.

When $\rho = 0.4$ or 0.5, and $n \leq 30$, *BP* will perform on average a bit better than the new one (see Figure 1). This is no longer true for $\rho = 0.6$ and $n = 20$. Therefore, the lower bound ρ_L lies between 0.5 and 0.6 as long as $n \leq 30$. We have not determined the density value at which the new heuristic is asymptotically better than *BP*. It is some number smaller than 0.6, and we conjecture it is greater than 0.5. It is conceivable that $0.5 + \epsilon$, for any $\epsilon > 0$, is dense enough for sufficiently large matrices.

4.2. Circuit Depth

The reduction of gate complexity of a circuit is not the only important measure on combinational logic implementation. The depth of a circuit, — i.e., the length of the longest path in it — is another one. Indeed, when depth of the combinational logic increases, an important performance metric worsens: the delay. In general, it is not difficult decreasing circuit depth at the cost of increasing circuit width, or vice versa. As shown in Section 4.1, the new heuristic reduces gate count when applied to dense linear systems. In this section, we experimentally show that new heuristic not only reduces the gate count but also decreases (on average) the circuit depth. An extensive testing activity has been conducted to evaluate the depth of the circuits. The set of data used is the same described in Section 4.1 — i.e., $n \times n$ matrices, $n = 15, 16, \ldots, 30$. For these experiments, we focus on the most interesting (dense) biases $\rho = 0.6, \ldots, 0.9$. We tested 6400 dense matrices previously generated, 100 matrices from each value of ρ and n. Experimental results were collected and analyzed.

Bias $\rho = \mathbf{0.6}$. Experimental results show that *BP* generates, on average, circuits with a shorter critical path (for $n \leq 30$).

Bias $\rho = \mathbf{0.7}, \mathbf{0.8}, \mathbf{0.9}$. This is no longer true for $\rho \geq 0.7$ and $n = 15$. In these cases, the new heuristic provides (on average) circuits with a shorter critical path. Experimental results show that our heuristic outperforms *BP* for high density matrices of size bigger than 15 × 15.

5. Concluding remarks

There are at least two ways to gauge how interesting these results are. This paper shows that the new heuristic outperforms (on average) *BP* heuristic, when applied to random dense linear systems. Experimental results suggest that the solutions provided usually have a shorter critical path and a reduced number of XOR gates. In [17, 18], Fuhs et al. were able to prove that, in specific cases, the circuits generated by *BP* are optimal. Our work shows that,

if we play with dense linear system, better results can be obtained encouraging circuits to benefit from the cancellation.

At a practical level, we note that linear systems of the sizes considered in this work show up in practice — e.g. AES [5, 6], HMAC-SHA-1 [8], Present [9], etc. In particular, the new heuristic has been used in 2013 to show that the bottom linear part of the circuit presented in [6] was sub-optimal by at least one gate. Then, exploring all ties, Cagdas Calik pointed out that the BP heuristic yields a better circuit [10].

References

[1]. Paar C, Optimized arithmetic for Reed-Solomon encoders, in: Proceedings of 1997 IEEE International Symposium on Information Theory, 1997, p. 250.

[2]. Satoh A, Morioka S, Takano K, Munetoh S, A Compact Rijndael Hardware Architecture with S-Box Optimization, in: Advances in Cryptology, Vol. 2248 of Lecture Notes in Computer Science, Springer, 2001, pp. 239–254.

[3]. Paar C, Some remarks on efficient inversion in finite fields, in: Proceedings of 1995 IEEE International Symposium on Information Theory, 1995, p. 58.

[4]. Canright D, A Very Compact S-Box for AES, in: Cryptographic Hardware and Embedded Systems, Vol. 3659 of Lecture Notes in Computer Science, Springer, 2005, pp. 441–455.

[5]. Boyar J, Peralta R, A new combinational logic minimization technique with applications to cryptology, in: Experimental Algorithms, Vol. 6049 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 178–189.

[6]. Boyar J, Matthews P, Peralta R, Logic minimization techniques with applications to cryptology, Journal of Cryptology 26 (2013) 280–312.

[7]. J. Boyar, M. G. Find, R. Peralta, Low-depth, low-size circuits for cryptographic applications, Cryptography and CommunicationsIn press.

[8]. A. Visconti, F. Gorla, Exploiting an HMAC-SHA-1 optimization to speed up PBKDF2, Cryptology ePrint Archive, Report 2018/097 (2018). URL https://eprint.iacr.org/2018/097.pdf

[9]. N. Courtois, D. Hulme, T. Mourouzis, Solving circuit optimisation problems in cryptography and cryptanalysis, Cryptology ePrint Archive, Report 2011/475 (2011). URL http://eprint.iacr.org/2011/475.pdf

[10]. CMT, Circuit Minimization Team. URL http://www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html

[11]. A. De Piccoli, A. Visconti, O. G. Rizzo, Polynomial multiplication over binary finite fields: new upper bounds, Cryptology ePrint Archive, Report 2018/091 (2018). URL https://eprint.iacr.org/2018/091

[12]. Bernstein DJ, High-speed cryptography in characteristic 2 (2009). URL http://binary.cr.yp.to/index.html

[13]. Lupanov O, On rectifier and switching-and-rectifier schemes, Dokl. Akad. 30 Nauk SSSR 111, 1171–1174.

[14]. Boyar J, Find MG, Cancellation-free circuits in unbounded and bounded depth, Theor. Comput. Sci. 590 (2015) 17–26.

[15]. A. Visconti, C. Schiavo, R. Peralta, Improved upper bounds for the expected circuit complexity of dense systems of linear equations over GF(2), Cryptology ePrint Archive, Report 2017/194 (2017). URL https://eprint.iacr.org/2017/194.pdf

[16]. A. Visconti, A benchmark set. URL http://homes.di.unimi.it/visconti/RESEARCH/BenchmarkSet.zip

[17]. Fuhs C, Schneider-Kamp P, Synthesizing Shortest Linear Straight-Line Programs over GF(2) Using SAT, in: Theory and Applications of Satisfiability Testing SAT 2010, Vol. 6175 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 71–84.

[18]. Fuhs C, Schneider-Kamp P, Optimizing the AES S-Box using SAT, in: 8th International Workshop on the Implementation of Logics IWIL 2010, Vol. 2 of EPiC Series in Computing, EasyChair, 2012, pp. 64–70.

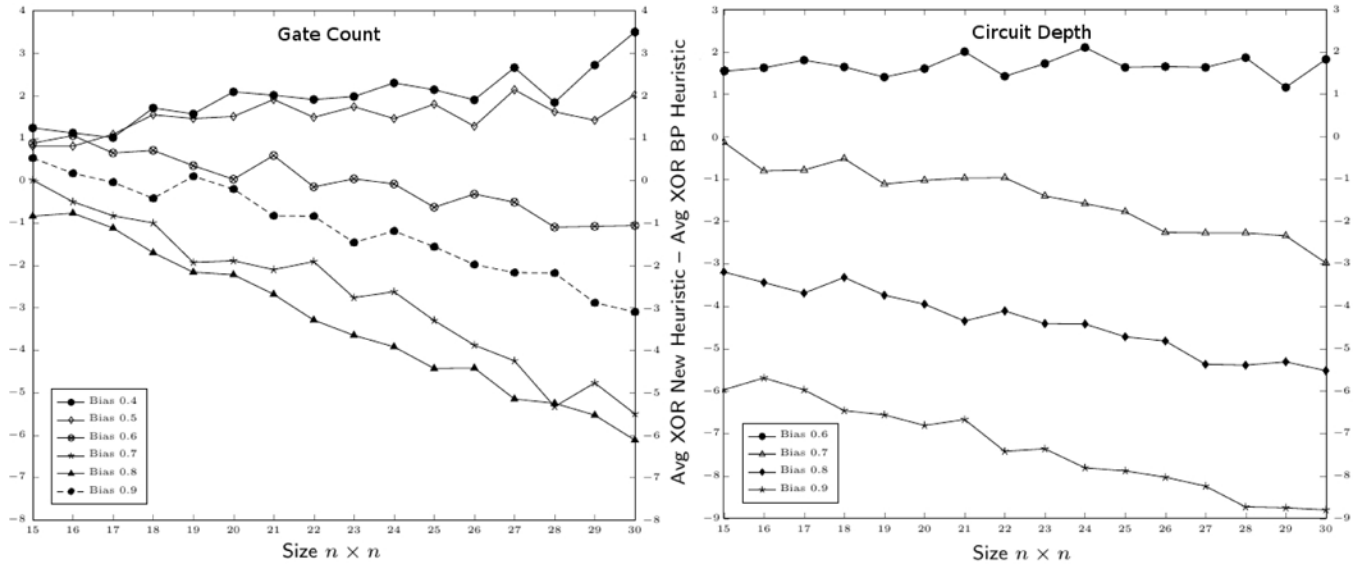

Figure 1:
Avg num XORs new heuristic compared to *BP* : Gate Count – Circuit Depth