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

Andrea Visconti[1], Chiara V. Schiavo[1], and René Peralta[2]

[1] Department of Computer Science, Università degli Studi di Milano
andrea.visconti@unimi.it, chiara.schiavo@unimi.it,
[2] Information Technology Laboratory, NIST
rene.peralta@nist.gov

Abstract. Minimizing the Boolean circuit implementation of a given cryptographic function is an important issue. A number of papers [12], [13], [11], [5] only consider cancellation-free straight-line programs for producing short circuits over GF(2) while [4] does not. Boyar-Peralta (BP) heuristic [4] yields a valuable tool for practical applications such as building fast software and low-power circuits for cryptographic applications, e.g. AES [4], PRESENT [7], and GOST [7]. However, 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 new and BP heuristic were compared. They show that the Boyar-Peralta bounds are not tight on dense systems.

Keywords: Gate complexity; linear systems; dense matrices; circuit depth; gate count; 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 in which the instructions are of the form $X_i = X_j + X_k$ where

– X_i 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-complete [2, 3]. This means that, unless P=NP, there is no efficient algorithm that can compute even approximately optimal solutions. In [4]

– 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 [4] has been successfully applied to a number of circuit optimization problems of interest to cryptology. These include a compact implementation of Present S-Box [7], finite-field arithmetic and binary multiplication [6] and [1].

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 *targets* these rows. There are no known tight combinatorial bounds for the gate complexity of random linear systems[3]. An obvious upper bound is $O(mn)$, the only non-trivial combinatorial upper bound we are aware of is $O(\frac{mn}{\log m})$. This can be derived from a lemma by Lupanov [10] about matrix decompositions.

A number of papers [12], [13], [11], and [5] only consider cancellation-free straight-line programs for producing short 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 [4], [6]. However, 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

[3] This is a well-defined problem: what is the expected number of gates sufficient and necessary to solve a random $m \times n$ system of equations? The answer is unknown.

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) BP heuristic, when applied to random dense linear systems.

2 The Boyar-Peralta heuristic

The BP heuristic [4] is for optimizing arbitrary circuits [4]. 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 a $m \times n$ 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 a $m \times n$ 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.

[4] The full technique was issued a US patent on 11-20-2012, Patent number 8,316,338.

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 applies the BP heuristic. Hence, it is appealing to try the following steps:

(i) use the BP heuristic to find a small circuit that implements $\overline{M}\mathbf{x}$;
(ii) use signals computed in (i) to compute a "common path" $yy = \sum_{i=1}^{n} x_i$;
(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 firstly 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 $\mathbf{y} = \{y_1, \ldots, y_m\}$ 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. (Initialization) Set S to the set of variables $x_1, \ldots, x_n$. For $i = 1, \ldots, m$, set $D[i] = HammingWeight(y_i) - 1$.
2. (Create complement instance) Let $\mathbf{y}^* = \{y_1^*, \ldots, y_m^*\}$ be the set of complement targets (i.e. $y_i^* = \overline{y_i}$). Add target $y_{m+1}^* = [1, \ldots, 1]$ to the set of complement targets (note y_{m+1}^* encodes the function $yy = \Sigma_{i=1}^{n} x_i$). Let $M^* = [y_1^*, \ldots, y_m^*, y_{m+1}^*]^T$. Let D^* be the distance vector for M^*, initialized to $D^*[i] = HammingWeight(y_i^*) - 1$ for $i = 1, \ldots, m+1$.
3. (Compute the common path) Until target y_{m+1}^* is found — i.e., until $D^*[m+1] = 0$ — pick a new base element x_i, $i = |S| + 1$, by adding two existing base elements such that:
   (a) x_i decreases the distance to y_{m+1}^* by one — i.e., to $D^*[m+1] - 1$;
   (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.
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 $y_1, \ldots, y_m$ 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 step.
The initial set of basis is $S = \{x_1, x_2, x_3, x_4, x_5\}$, the set of targets 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: We create a complement instance.
We generate the "common path" $yy = x_1 + x_2 + x_3 + x_4 + x_5$, the new set of targets is $\mathbf{y}^* = \{\overline{y}_1, \overline{y}_2, \overline{y}_3, \overline{y}_4, \overline{y}_5, \overline{y}_6, yy\}$, 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: We compute the "common path" — i.e., yy.

- $x_7 = x_1 + x_3$.
- $x_8 = x_4 + x_5$.
- $x_9 = x_2 + x_7$. Found target $y_1 = x_9$.
- $x_{10} = x_8 + x_9$. Found target $yy = x_{10}$.

Step 4: We apply the BP heuristic to matrix M.

- $x_{11} = x_1 + x_{10}$. Found target $y_6 = x_{11}$.
- $x_{12} = x_2 + x_8$. Found target $y_2 = x_{12}$.
- $x_{13} = x_2 + x_{10}$. Found target $y_3 = x_{13}$.
- $x_{14} = x_5 + x_{11}$. Found target $y_4 = x_{14}$.
- $x_{15} = x_7 + x_{14}$. Found target $y_5 = x_{15}$.

4 Experimental results

Experiments provided in Tables 1-8 will have the following structure:

- **Size n × n**: size of the matrix;
- **New is the best**: number of times the new heuristic performs better than BP;
- **BP is the best**: number of times BP performs better than the new heuristic;
- **Same Solution**: number of times the new heuristic performs as BP;
- **Avg New**: average number of XOR gates required by the new heuristic;
- **Avg BP**: average number of XOR gates required by BP;
- **Avg Diff**: difference between values collected in fields *"Min AVG number of XORs New Heuristic"* and *"Min AVG number of XORs BP heuristic"*;
- **Stand Dev New**: standard deviation of the data collected (New heuristic)
- **Stand Dev BP**: standard deviation of the data collected (BP heuristic)
- **Best Case**: minimum distance between the number of XOR gates computed by the new heuristic and those computed by the old one on the same set of random matrices M_i:

  $$\text{BestCase} = \min_{i=1\ldots100}(\#\text{ XORs New heur.}(M_i) - \#\text{ XORs } BP \text{ heur.}(M_i))$$

- **Worst Case**: maximum distance between the number of XOR gates computed by the new heuristic and those computed by the old one on the same set of random matrices M_i:

  $$\text{WorstCase} = \max_{i=1\ldots100}(\#\text{XORs New heur.}(M_i) - \#\text{XORs } BP \text{ heur.}(M_i))$$

Experiments were performed on square and non-square matrices. Due to space limitations, this paper only discusses our results on square matrices. The useful conclusions drawn are also valid for rectangular matrices.

4.1 Gate Count

We conducted extensive testing to gauge the performance of our new heuristic, against that of BP. 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. Therefore, we generated $n \times n$ matrices, $n = 15, 16, \ldots, 30$, for biases $\rho = 0.4, 0.5, \ldots, 0.9$. We tested 100 matrices from each distribution for a total of 9600 matrices. Circuits for $M\mathbf{x}$ were constructed for each matrix M using BP and our heuristic. The results are shown in Tables 1-4.

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 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 matrix. However, over 3200 matrices handled, the new heuristic gets better results in 577 cases — i.e. respectively, 211 when $\rho = 0.4$ and 366 when $\rho = 0.5$. This means that 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. In particular, Table 1 shows that the difference between the average number of XOR gates computed by the two heuristics gradually decrease with the increasing size of the matrix. The data collected show that 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 $\rho = 0.7$. Table 2 shows the threshold over which the new heuristic performs better than the old one. For 16×16 matrices the new heuristic gets the best, or the same, solution in 73% of cases (see Table 2). This value grows up to 98% for 30×30 matrices.

Bias $\rho = 0.8$. In this case the new heuristic performs better than $\rho = 0.7$. In fact, it beats BP on matrices as small as 15×15 (see Table 3). Note that, 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%.

Table 1: Gate Count, Bias = 0.6

Size $n \times n$	New is the best	BP is the best	Same solution	Avg New	Avg BP	Avg Diff	Stand Dev New	Stand Dev BP	Best Case	Worst Case
15×15	21	62	17	47.340	46.460	+0.880	1.570	1.688	-4	+5
16×16	23	58	19	53.100	52.040	+1.060	2.347	2.209	-4	+7
17×17	29	53	18	59.220	58.570	+0.650	1.998	1.961	-4	+6
18×18	29	54	17	64.810	64.100	+0.710	2.053	2.086	-6	+8
19×19	38	48	14	71.340	70.990	+0.350	2.219	2.243	-5	+7
20×20	45	40	15	77.680	77.650	+0.030	2.437	2.151	-5	+5
21×21	31	51	18	85.380	84.790	+0.590	2.682	2.188	-6	+7
22×22	40	42	18	91.870	92.020	-0.150	2.327	2.668	-7	+5
23×23	42	45	13	99.950	99.910	+0.040	2.805	2.298	-8	+7
24×24	46	43	11	107.610	107.690	-0.080	2.756	2.477	-8	+6
25×25	52	40	8	115.570	116.200	-0.630	2.981	2.771	-9	+9
26×26	45	40	15	124.020	124.340	-0.320	2.789	2.804	-7	+9
27×27	49	38	13	132.630	133.140	-0.510	3.261	2.946	-9	+10
28×28	56	33	11	141.390	142.490	-1.100	3.082	2.941	-8	+7
29×29	58	33	9	150.020	151.100	-1.080	3.403	3.158	-8	+11
30×30	52	27	21	159.520	160.580	-1.060	2.927	2.808	-7	+5

Table 2: Gate Count, Bias = 0.7

Size $n \times n$	New is the best	BP is the best	Same solution	Avg New	Avg BP	Avg Diff	Stand Dev New	Stand Dev BP	Best Case	Worst Case
15×15	35	42	23	44.540	44.530	+0.010	2.406	2.179	-5	+4
16×16	54	27	19	49.430	49.930	-0.500	2.224	2.219	-4	+3
17×17	59	18	23	54.120	54.950	-0.830	2.334	2.273	-5	+3
18×18	58	21	21	59.870	60.870	-1.000	2.873	2.792	-6	+4
19×19	76	10	14	65.900	67.830	-1.930	2.696	2.635	-7	+3
20×20	73	17	10	71.780	73.670	-1.890	3.236	3.027	-8	+3
21×21	83	9	8	78.080	80.180	-2.100	3.065	2.906	-7	+4
22×22	75	15	10	84.700	86.610	-1.910	3.068	3.304	-7	+4
23×23	82	10	8	91.720	94.480	-2.760	3.244	2.872	-9	+2
24×24	80	12	8	98.320	100.940	-2.620	3.608	3.104	-9	+3
25×25	80	9	11	105.850	109.150	-3.300	4.048	3.528	-9	+4
26×26	87	7	6	113.280	117.160	-3.880	3.329	3.605	-11	+3
27×27	90	6	4	121.280	125.530	-4.250	3.884	3.486	-11	+2
28×28	94	5	1	128.450	133.780	-5.330	3.380	3.448	-12	+4
29×29	91	4	5	137.180	141.950	-4.770	3.451	3.182	-10	+1
30×30	95	2	3	145.500	151.000	-5.500	4.211	4.123	-13	+3

Table 3: Gate Count, Bias = 0.8

Size $n \times n$	New is the best	BP is the best	Same solution	Avg New	Avg BP	Avg Diff	Stand Dev New	Stand Dev BP	Best Case	Worst Case
15×15	55	19	26	38.530	39.370	-0.840	2.368	2.335	-5	+3
16×16	58	16	26	42.740	43.510	-0.770	2.536	2.876	-4	+2
17×17	63	16	21	46.700	47.820	-1.120	2.876	3.096	-5	+3
18×18	76	9	15	51.250	52.950	-1.700	2.670	2.563	-6	+2
19×19	83	6	11	55.660	57.820	-2.160	2.804	3.176	-6	+2
20×20	82	11	7	60.520	62.740	-2.220	3.211	3.568	-7	+3
21×21	88	1	11	65.580	68.260	-2.680	3.365	3.480	-9	+1
22×22	89	4	7	71.220	74.510	-3.290	3.422	3.571	-9	+3
23×23	89	7	4	76.180	79.830	-3.650	3.748	4.079	-9	+2
24×24	95	2	3	81.980	85.900	-3.920	4.266	4.590	-9	+2
25×25	97	1	2	87.420	91.850	-4.430	3.990	4.203	-11	+1
26×26	98	0	2	93.970	98.390	-4.420	4.006	4.282	-10	0
27×27	96	1	3	99.710	104.860	-5.150	4.224	4.860	-13	+2
28×28	99	1	0	106.680	111.930	-5.250	4.777	4.682	-11	+1
29×29	97	2	1	112.680	118.210	-5.530	4.793	5.307	-12	+3
30×30	100	0	0	120.580	126.700	-6.120	4.548	4.095	-13	0

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.

Table 4: Gate Count, Bias = 0.9

Size $n \times n$	New is the best	BP is the best	Same solution	Avg New	Avg BP	Avg Diff	Stand Dev New	Stand Dev BP	Best Case	Worst Case
15×15	21	55	24	32.270	31.740	+0.530	1.766	2.062	-3	+3
16×16	29	43	28	34.770	34.600	+0.170	1.654	1.965	-3	+3
17×17	39	40	21	37.860	37.900	-0.040	2.140	2.385	-4	+4
18×18	46	25	29	40.590	41.010	-0.420	2.200	2.678	-4	+3
19×19	34	41	25	43.580	43.480	+0.100	2.237	2.559	-4	+3
20×20	43	30	27	46.970	47.170	-0.200	2.427	2.735	-4	+4
21×21	57	22	21	50.810	51.640	-0.830	2.848	3.294	-7	+3
22×22	62	25	13	54.140	54.980	-0.840	3.353	3.990	-5	+3
23×23	71	17	12	56.650	58.110	-1.460	3.061	3.580	-6	+2
24×24	72	15	13	59.920	61.110	-1.190	3.062	3.518	-6	+5
25×25	73	14	13	63.710	65.270	-1.560	4.041	4.781	-6	+4
26×26	76	12	12	67.110	69.090	-1.980	4.140	4.954	-7	+4
27×27	81	10	9	71.580	73.750	-2.170	3.421	4.346	-8	+4
28×28	76	13	11	75.120	77.300	-2.180	4.255	4.967	-7	+2
29×29	88	4	8	79.810	82.690	-2.880	4.039	4.573	-8	+2
30×30	85	4	11	84.003	87.100	-3.097	4.121	4.988	-14	+2

Figure 1 visualizes the output data collected in columns *"Avg Diff"*. 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.

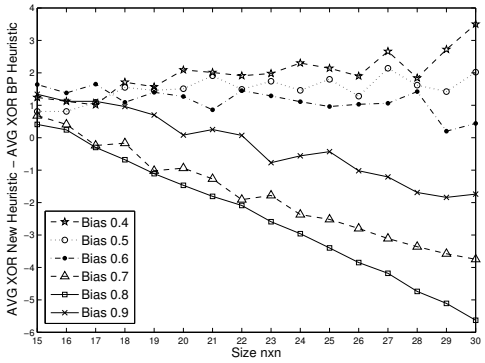

Fig. 1: Gate Count: Avg number of XORs of New heuristic compared to BP

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. A comparison between the two heuristics can be found in Tables 5-8.

Table 5: Critical Path, Bias = 0.6

Size $n \times n$	**New is the best**	**BP is the best**	**Same solution**	**Avg New**	**Avg BP**	**Avg Diff**	**Stand Dev New**	**Stand Dev BP**	**Best Case**	**Worst Case**
15×15	16	68	16	11.280	9.740	+1.540	1.575	1.683	-5	+5
16×16	18	70	12	11.930	10.310	+1.620	1.458	1.547	-4	+6
17×17	17	74	9	12.430	10.630	+1.800	1.351	1.474	-4	+5
18×18	18	73	9	12.750	11.110	+1.640	1.431	1.794	-4	+6
19×19	23	59	18	12.920	11.520	+1.400	1.488	1.706	-4	+6
20×20	16	69	15	13.500	11.900	+1.600	1.360	1.936	-4	+7
21×21	9	75	16	13.900	11.900	+2.000	1.700	1.646	-2	+11
22×22	18	66	16	14.090	12.670	+1.420	1.281	1.784	-5	+6
23×23	14	73	13	14.390	12.670	+1.720	1.392	1.631	-3	+7
24×24	11	77	12	15.020	12.920	+2.100	1.536	1.831	-6	+8
25×25	22	65	13	14.960	13.330	+1.630	1.777	1.778	-4	+8
26×26	19	71	10	15.380	13.730	+1.650	1.617	1.902	-4	+7
27×27	18	71	11	15.740	14.110	+1.630	1.659	1.843	-6	+7
28×28	17	68	15	15.880	14.020	+1.860	1.693	1.860	-3	+7
29×29	25	61	14	15.890	14.730	+1.160	1.455	1.777	-5	+6
30×30	18	68	14	16.240	14.420	+1.820	1.656	1.511	-3	+6

Bias $\rho = 0.6$. Table 5 shows that BP generates, on average, circuits with a short critical path (for $n \leq 30$).

Table 6: Critical Path, Bias = 0.7

Size $n \times n$	**New is the best**	**BP is the best**	**Same solution**	**Avg New**	**Avg BP**	**Avg Diff**	**Stand Dev New**	**Stand Dev BP**	**Best Case**	**Worst Case**
15×15	42	40	18	11.010	11.140	-0.130	1.603	1.778	-6	+9
16×16	54	23	23	11.020	11.830	-0.810	1.077	1.691	-6	+5
17×17	53	29	18	11.440	12.230	-0.790	1.402	1.624	-5	+4
18×18	56	30	14	12.280	12.800	-0.520	1.457	1.744	-6	+5
19×19	63	19	18	12.110	13.230	-1.120	1.232	1.618	-6	+4
20×20	58	19	23	12.510	13.540	-1.030	1.360	1.729	-6	+5
21×21	57	28	15	12.910	13.890	-0.980	1.320	1.720	-6	+4
22×22	60	30	10	13.370	14.340	-0.970	1.369	1.867	-6	+6
23×23	64	22	14	13.520	14.920	-1.400	1.323	2.062	-14	+4
24×24	71	20	9	13.590	15.170	-1.580	1.429	1.772	-8	+4
25×25	73	16	11	13.790	15.560	-1.770	1.267	1.956	-7	+3
26×26	75	13	12	13.890	16.160	-2.260	1.399	2.028	-9	+3
27×27	77	9	14	14.270	16.540	-2.270	1.326	1.763	-8	+3
28×28	72	17	11	14.370	16.640	-2.270	1.383	2.100	-12	+2
29×29	80	13	7	14.700	17.040	-2.340	1.404	1.995	-8	+3
30×30	87	6	7	14.620	17.600	-2.980	1.147	2.263	-11	+1

Table 7: Critical Path, Bias = 0.8

Size $n \times n$	New is the best	BP is the best	Same solution	Avg New	Avg BP	Avg Diff	Stand Dev New	Stand Dev BP	Best Case	Worst Case
15×15	89	5	6	9.350	12.540	-3.190	1.099	1.819	-8	+1
16×16	92	3	5	9.750	13.190	-3.440	1.090	1.901	-11	+2
17×17	93	4	3	9.900	13.590	-3.690	1.170	1.950	-8	+2
18×18	94	1	5	10.430	13.750	-3.320	1.210	1.728	-8	+1
19×19	94	2	4	10.800	14.540	-3.740	1.233	1.962	-11	+3
20×20	93	3	4	11.100	15.050	-3.950	1.277	1.956	-10	+2
21×21	96	1	3	11.110	15.460	-4.350	1.067	1.878	-9	+1
22×22	96	1	3	11.480	15.590	-4.110	1.261	1.877	-10	+1
23×23	94	0	6	11.520	15.930	-4.410	1.053	2.080	-11	0
24×24	99	1	0	11.690	16.110	-4.420	1.164	1.928	-10	+1
25×25	98	2	0	12.090	16.810	-4.720	1.150	1.809	-10	+1
26×26	98	1	1	12.180	17.000	-4.820	1.126	1.860	-10	+3
27×27	97	0	3	12.290	17.660	-5.370	1.116	2.150	-12	0
28×28	98	0	2	12.400	17.790	-5.390	1.095	2.334	-11	0
29×29	99	0	1	12.880	18.190	-5.310	1.125	2.199	-13	0
30×30	99	0	1	12.920	18.440	-5.520	1.146	2.351	-12	0

Table 8: Critical Path, Bias = 0.9

Size $n \times n$	New is the best	BP is the best	Same solution	Avg New	Avg BP	Avg Diff	Stand Dev New	Stand Dev BP	Best Case	Worst Case
15×15	100	0	0	7.730	13.700	-5.970	1.009	1.868	-13	0
16×16	100	0	0	8.020	13.710	-5.690	0.836	1.920	-12	0
17×17	100	0	0	8.280	14.250	-5.970	0.884	1.705	-10	0
18×18	100	0	0	8.610	15.070	-6.460	1.048	1.909	-11	0
19×19	100	0	0	8.610	15.170	-6.560	0.773	1.929	-13	0
20×20	99	0	1	8.830	15.640	-6.810	1.001	2.504	-14	0
21×21	99	0	1	9.270	15.940	-6.670	0.958	2.125	-12	0
22×22	100	0	0	9.320	16.740	-7.420	0.835	2.175	-14	0
23×23	100	0	0	9.620	16.980	-7.360	0.988	2.131	-13	0
24×24	100	0	0	9.870	17.680	-7.810	0.891	2.690	-14	0
25×25	100	0	0	10.070	17.950	-7.880	1.022	2.372	-15	0
26×26	100	0	0	10.170	18.200	-8.030	0.960	2.542	-14	0
27×27	100	0	0	10.160	18.400	-8.240	0.880	2.241	-14	0
28×28	100	0	0	10.460	19.190	-8.730	0.767	2.583	-15	0
29×29	100	0	0	10.580	19.330	-8.750	0.908	2.600	-15	0
30×30	99	0	1	10.780	19.580	-8.800	0.901	2.926	-14	0

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 short critical path. Tables 6, 7, and 8 show that our heuristic outperforms BP for high density matrices of size bigger than 15×15.

5 Concluding remarks

There are at least three 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 short critical path and a reduced number of XOR gates. In [8, 9], Fuhs et. al. were able to prove that, in specific cases, the circuits generated by BP are optimal. These results from the SAT-solvers community strongly hinted at the BP method producing optimal, or near optimal, solutions in the range in which it is applied. Our work has disproved this.

At a practical level, we note that linear systems of the sizes considered in this work ($n \leq 30$) show up in practice — e.g. AES, Present, etc. The new heuristic has been used to show that the bottom linear part of the circuit presented in [3] was sub-optimal by at least one gate (see Appendix A). Then, exploring all ties, Cagdas Calik pointed out that the BP heuristic yields a better circuit [6].

The third optic on these results comes from asking questions such as "what is the actual linear complexity of random $n \times n$ systems of equations"?; "how symmetric is the curve around density $\rho = 0.5$"?; "is it something like $comp(n, \rho) = \Theta(comp(n, 1 - \rho)) + O(n)$"? or is the deviation from symmetry sub-linear (e.g. $O(\sqrt{n})$), or even polylogarithmic in n? Figure 2 This work sheds information on what the answers to these questions might be. But we are far from being able to prove any conjecture that this figure might trigger. Figure 2 depicts this finding in a scale-free way. The upper curve is the result of running the BP heuristic. The lower curve is the result of running our heuristic. The figure plots results obtained by running the programs on 20×20 matrices. The distance between lower and upper curve increases with the size of the matrix. We would be surprised if the curve that describes the linear complexity of $n \times n$ systems of equations for different density values is significantly different than what this picture shows (both in the practical range and in the asymptotic case).

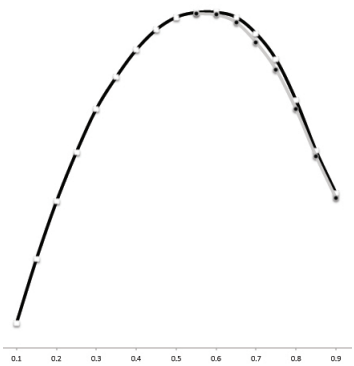

Fig. 2: The likely complexity of random square linear systems of different densities

References

1. Bernstein, D.J.: High-speed cryptography in characteristic 2 (2009), `http://binary.cr.yp.to/index.html`
2. Boyar, J., Matthews, P., Peralta, R.: On the shortest linear straight-line program for computing linear forms. In: Ochmaski, E., Tyszkiewicz, J. (eds.) Mathematical Foundations of Computer Science 2008, Lecture Notes in Computer Science, vol. 5162, pp. 168–179. Springer Berlin Heidelberg (2008)
3. Boyar, J., Matthews, P., Peralta, R.: Logic minimization techniques with applications to cryptology. Journal of Cryptology pp. 1–33 (2012), `http://dx.doi.org/10.1007/s00145-012-9124-7`
4. Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In: Festa, P. (ed.) Experimental Algorithms, Lecture Notes in Computer Science, vol. 6049, pp. 178–189. Springer Berlin Heidelberg (2010), `http://dx.doi.org/10.1007/978-3-642-13193-6_16`
5. Canright, D.: A Very Compact S-Box for AES. In: Cryptographic Hardware and Embedded Systems (CHES). Lecture Notes in Computer Science, vol. 3659, pp. 441–455. Springer (2005)
6. CMT: Circuit Minimization Team, `http://www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html`
7. Courtois, N., Hulme, D., Mourouzis, T.: Solving circuit optimisation problems in cryptography and cryptanalysis, http://eprint.iacr.org/2011/475.pdf
8. Fuhs, C., Schneider-Kamp, P.: Synthesizing Shortest Linear Straight-Line Programs over GF(2) Using SAT. In: Strichman, O., Szeider, S. (eds.) Theory and Applications of Satisfiability Testing SAT 2010, Lecture Notes in Computer Science, vol. 6175, pp. 71–84. Springer Berlin Heidelberg (2010)
9. Fuhs, C., Schneider-Kamp, P.: Optimizing the aes s-box using sat. In: Sutcliffe, G., Schulz, S., Ternovska, E. (eds.) IWIL 2010. The 8th International Workshop on the Implementation of Logics. EPiC Series, vol. 2, pp. 64–70. EasyChair (2012)
10. Lupanov, O.: On rectifier and switching-and-rectifier schemes. Dokl. Akad. 30 Nauk SSSR 111, 1171-1174. (1965)
11. Paar, C.: Some remarks on efficient inversion in finite fields. In: IEEE International Symposium on Information Theory. p. 58 (1995)

12. Paar, C.: Optimized arithmetic for reed-solomon encoders. In: IEEE International Symposium on Information Theory (1997)
13. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 2248, pp. 239–254. Springer (2001)

Appendix A: An S-box circuit for AES

In [4] authors show a decomposition of the AES S-box in three parts: a top linear part (i.e., a 22×8 linear system), a middle non-linear part, and a bottom linear part (i.e., 8×18 linear system).

The optimality of the straight-line programs (SLPs) for the two linear transformations proposed in [4] has been studied in [8]. While it has been demonstrated that the SLP for the 22×8 linear transformation is optimal (23 XOR gates), to the best of our knowledge, the SAT-solver community was unable to do the same for the bottom linear part.

By applying our heuristic to the 8×18 bottom linear transformation ($0.4 < \text{bias} < 0.5$) we generate many equivalent SLPs with 29 XOR gates and different depth. As an example, we show the first one we found (see Table 9). This yields a circuit for the AES S-box which contains a total of 114 gates and depth 30.

To find a better circuits, we have to explore all ties, list all SLPs found (bottom linear part) and compute the depth of the full AES S-box circuits (top linear and middle non-linear part included). Such exhaustive search is out of the scope of this paper.

Table 9: Optimized bottom linear transformation of AES S-box. Inputs: $x_0, \ldots, x_{17}$. Outputs: $y_0, \ldots, y_7$

X18 = x15 + x16	X19 = x10 + X18	X20 = x4 + X19	X21 = x3 + X20
X22 = x0 + x2	X23 = x7 + X21	X24 = X22 + X23	X25 = x1 + x9
X26 = x8 + x12	X27 = x0 + X25	y3 = X21 + X27	X28 = x3 + x5
X29 = x6 + X23	y0 = x9 + X29	X30 = X22 + X28	y4 = y3 + X30
X31 = X24 + X26	X32 = x13 + X30	X33 = x14 + X31	y5 = x11 + X33
X34 = x10 + X31	y6 = X32 + X34	X35 = x12 + X18	y7 = X32 + X35
X36 = X19 + y3	y1 = X29 + X36	X37 = x15 + x17	X38 = X29 + X33
y2 = X37 + X38			