SAT solver performance on Circuit Satisfiability Problem

Mitchell Billard, Adebayo Emmanuel Adeoya

July.9, 2018

1 Introduction

The Circuit Satisfiability problem (also known as CIRCUIT-SAT, Circuit-SAT, CSAT, etc.) is the decision problem of determining whether a given Boolean circuit has an assignment of its inputs that makes the output True/False. Circuit-SAT has been proven to be NP-Complete. [Wik14] we can defined as follows:

Instance:
A Boolean circuit with n inputs, m gates, d layers, and each gate having up to $fanin$ inputs.

Acceptance Condition:
Accept if there is a combination of n's such that the circuit outputs 1 (True).

In this project, we generate, reduce, and determine the satisfiability of varying Boolean circuits. The code for this project can be found at https://github.com/mbillard11/Circuit-SAT.

2 Reducing Circuit Satisfiability Problem to SAT

We reduce the Circuit Satisfiability Problem to SAT as follows:

Variables:
Circuit = $\{G | \exists$ some combination of $g_1, g_2, ...g_n, ..., g_m \in \{0,1\}$ $\}$
The circuit has n inputs ($g_1, g_2, ...g_n$) and m gates ($g_{n+1}, ...g_m$) where g_m is the output gate. In our SAT solver, we introduced variable ($z_1, z_2, ...z_n$, $z_{n+1}, ...z_m$) which encoded values that circuit gates accepts.

Constraints:

1. If g_i is a NOT gate with input g_j, then add to U :
   $(\neg z_i \iff z_j) \equiv (\neg z_i \to z_j) \wedge (z_j \to \neg z_i)$
   $\equiv(\neg z_i \vee \neg z_j) \wedge (z_i \vee z_j)$

2. If g_i is an AND gate with input g_j and g_k, then add to U :
   $(z_i \iff z_j \wedge z_k)$
   $\equiv(z_i \to z_j \wedge z_k) \wedge (z_j \wedge z_k \to z_i)$
   $\equiv (\neg z_i \vee (z_j \wedge z_k)) \wedge (\neg(z_j \wedge z_k) \vee z_i)$
   $\equiv (\neg z_i \vee z_j) \wedge (\neg z_i \vee z_k) \wedge (\neg z_j \vee \neg z_k \vee z_i)$

3. If g_i is an OR gate with input g_j and g_k, then add to U :
   $(z_i \iff z_j \vee z_k)$
   $\equiv(z_i \to z_j \vee z_k) \wedge (z_j \vee z_k \to z_i)$
   $\equiv (\neg z_i \vee (z_j \vee z_k)) \wedge (\neg(z_j \vee z_k) \vee z_i)$
   $\equiv (\neg z_i \vee z_j \vee z_k) \wedge [(\neg z_j \wedge \neg z_k) \vee z_i]$
   $\equiv (\neg z_i \vee z_j \vee z_k) \wedge (\neg z_j \vee z_i) \wedge (\neg z_k \vee z_i)$

4. To get the final CNF form of a circuit, 'AND' all of the encoded gate values together with z_m.
   $U = z_1 \wedge z_2 \wedge \cdots \wedge z_n \wedge \cdots \wedge z_m$

The SAT Solver outputs satisfying assignments for all z_i. We only consider z_1 to z_n because we only need assignments for the input gates. We assigned z_1 ... z_n to g_1 ... g_n respectively.

3 Generating random instances of the Circuit Satisfiability problem

In generating random instances for the circuit satisfiability, our generator created random "layered" Boolean circuit. This implies that the gates in layer i was able to connect to gates from the previous layer $i-1$. In our experiments, we applied four parameters to the generator which are:

1. n = number of input gates

2. m = number of gates in the circuit

3. $fanin$ = maximum number of inputs a gate can accept (at most it will equal the number of gates in the previous layer. The minimum fanin for a NOT gate is 1, and for both AND and OR gates is 2)

4. d = depth of the circuit. (Where depth is a layered circuit, d also equals the number of layers in the circuit.)

Our generator accepts integer values for each of these parameters. We restricted the range of our choices so that we don't make excessively large circuits that will not be able to be solved in a reasonable amount of time. This range can be experimented with in the future.

To start, each layer will have the same amount of gates, calculated by dividing m by d (give or take 1 due to integer division). We compared circuits of different orientations by setting measure the time taken by changing the value of m at the value of d fixed to constant. This was also done vice-versa for d against m (i.e. 2x5 vs. 5x2).

Each gate in the layer was chosen randomly from one of the 3 options (AND, OR, NOT), and was given a random amount of inputs (min $fanin \leqslant$ amount of inputs $\leqslant$ generator $fanin$). We also performed some experiments by giving each gate the same amount of inputs, and compared that with given gate at random amount of inputs. The inputs assigned to each gate was randomly chosen from the outputs of the previous layer of gates.

We created a class for the generator that stores the information of each gate when created. The gates have the following parameters:

1. *layer* = The layer at which the gate exists
2. i = The number of that gate in its layer
3. *type* = Either AND, OR, or NOT
4. $fanin$ = The number of inputs that exists in the gate.
5. *inputs* = What are the inputs of gate (i.e. outputs from other gates)
6. *output* = The output of the gate

The generator outputs a list of these gate objects which can be found in the github link for our project.

4 Experimental results

We used the Syrup Solver running on Ubuntu 16.04 LTS. The system has 8 Gb of ram, and has a quad-core Intel Core i5-4440 CPU @ 3.10GHz. The Syrup Solver is a SAT Solver for doing symbolic circuit analysis.It reads a circuit description written in DIMACS CNF notation, generates the nodal equation and returns the solutions. The Reducer.py program writes the DIMACS reduction of the circuit directly into the input.cnf file for the solver.

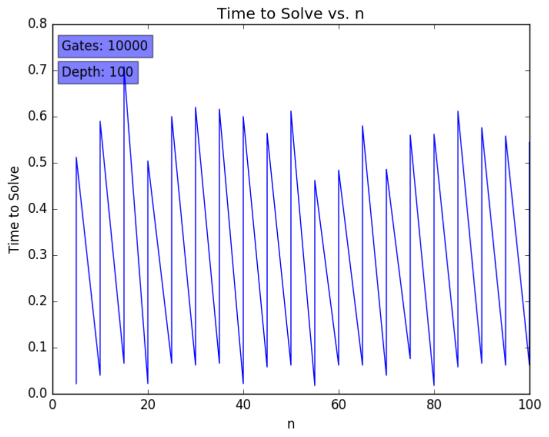


Figure 1: Gates: 10000, Depth: 100
From this figure we observe that there is no relationship between time to solve and n. As n increases the time to solve remains within a constant range.

In our experimentation, we ran 2 batches of tests (Appendix A). The first batch,the value of n used ranges from 5 to 100, incrementing n by 5. In the second batch, the value of n ranges from 5 to 1000, incrementing by 50. We ran both batches for 16 times each, varying the values of m and d. m set to 10, 100, 1000, and 10000. d was set to 3, 10, 100, and 1000. For each increment of n for a given m and d, 10 circuits were generated and solved. For example, for n = (5, 100, 5), m = 1000, and d = 10, 10 circuits were created and solved. The average time taken to solve the 10 instances for the particular value of n is plotted on our graphs. Table 1 shows the values of our experiments. Some spaces are left blank because d had to be less than or equal to m.

For all of our experiments, the fanin of each gate was constant. It was always 1 for NOT gates, and always 2 for AND and OR gates. Allowing for more fanin would require considerable changes to the Generator and the Reducer. We plan to make these changes and report back with the results.

The solver ran surprisingly fast for most instances. A lot of the time it solved the circuit so fast that it would record it taking 0 seconds. We found that in situations where m = d, the solver would take 0 seconds to run. If the graphs for the first batch (n = 5, 100, 5) weren't 0, then they were "Zig-Zaggy", showing that as n increases, the solver does not fine the circuit harder to solve on average. For the second batch (n = 5, 1000, 50) there were some interesting results, but nothing that showed the solver finding significant differences in time to find the solution. These graphs just connect the scattered data points. Implementing a line of best fit would help to visualize the results better (See additional experiments).

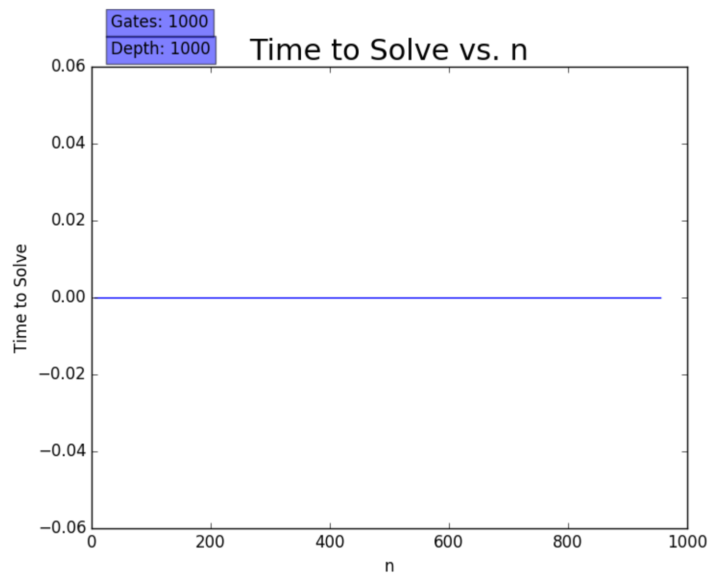


Figure 2: Gates: 1000, Depth: 1000

Here we observed that at gate equals 1000 and depth equals 1000 (i.e setting the both values to be the same), the generator speed was faster compared to that of figure one. The average time taken for all instances of n was the same.

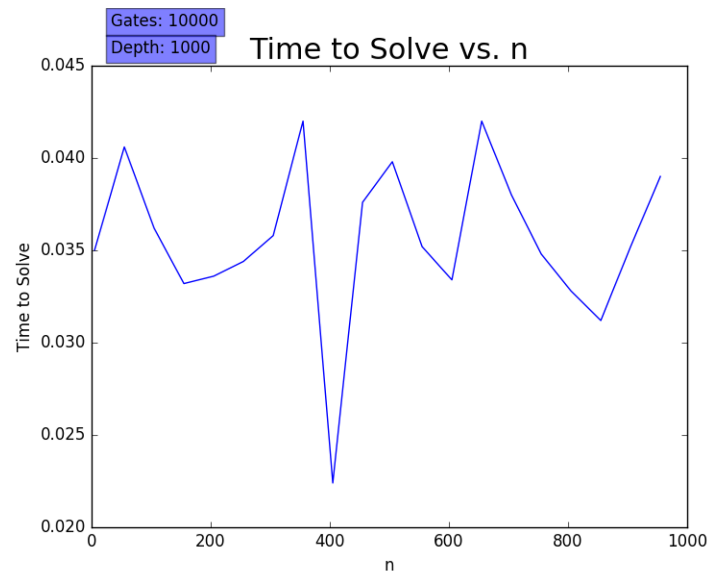


Figure 3: Gates: 10000, Depth: 1000

Similarly to the previous 2 tables, this figure shows no correlation between time to solve and n. A line of best fit for this figure would be close to a horizontal line.

Our solver seems to not depend on n. Again, I believe this is due to the fact that with the limited fanin, the solver is able to very quickly identify if it is satisfiable or not. When the solver doesn't solve it instantly (i.e. when d=m), the number of satisfiable circuits seems to range between $10\% - 80\%$. Below are some example figures of how long the solver took to solve a problem with respect to n.

5 Additional experiments

Since the initial report, the Generator and Reducer has been upgraded to allow for gates to take more inputs. Also a RunExp.py script has been created to run multiple experiments one after another. We now have 4 variables we can change. Each variable has multiple possible settings. When running the experiments the values were set as follows:

1. n = [[1,21,1],[5,101,5],[5,1001,50]]
2. m = [3, 10, 100, 1000, 10000]
3. $fanin$ = [2,4,10,50]
4. d = [2, 5, 10, 100, 1000]

As you can see, the total number of combinations of these variables is 3*5*4*5 = 300 experiments. Each experiment is ran over the range of n which is always 20. And for every instance of n, 10 random circuits are generated and solved. Thus in total we have generated and solved a total of 300*20*10 = 60,000 circuits. The output from RunExp.py can be seen in Appendix B. The satisfiability percentage as well as the maximum time to determine satisfiability for each experiment is given. Based on these results, we plotted graphs for the most interesting cases. These graphs now display a line of best fit for the data points.

We notice that the time to solve a circuit become observable once $m \leq 1000$. Also, there is a clear linear relation between time to solve and $fanin$. Graphs of these relations can be seen in Appendix C. Below are graphs of 2 examples showing the difference in time to solve vs n. Both examples have m = 10,000 and the same range for n, but the second example has 20 times as many layers (depth) and about one twentieth the $fanin$. Notice how the second example solves instances about 5 times faster than the first.

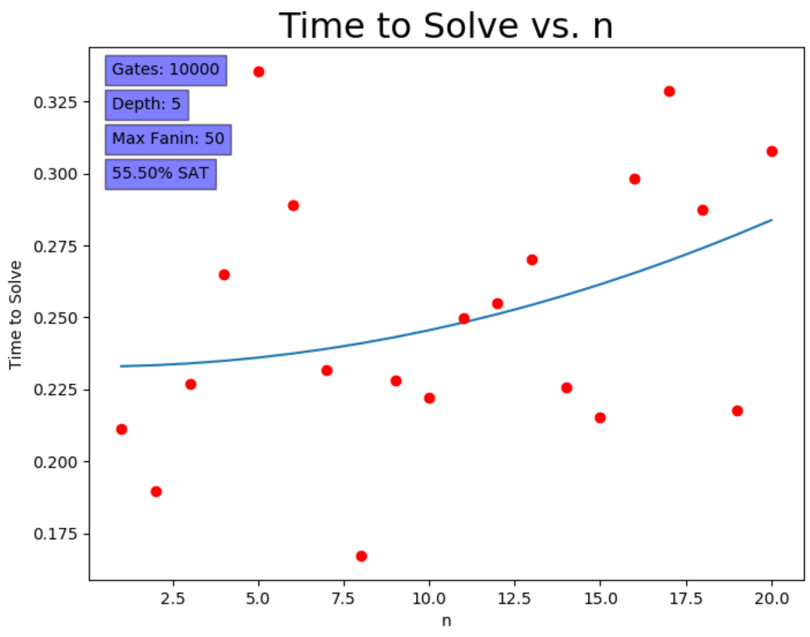


Figure 4: Example 1

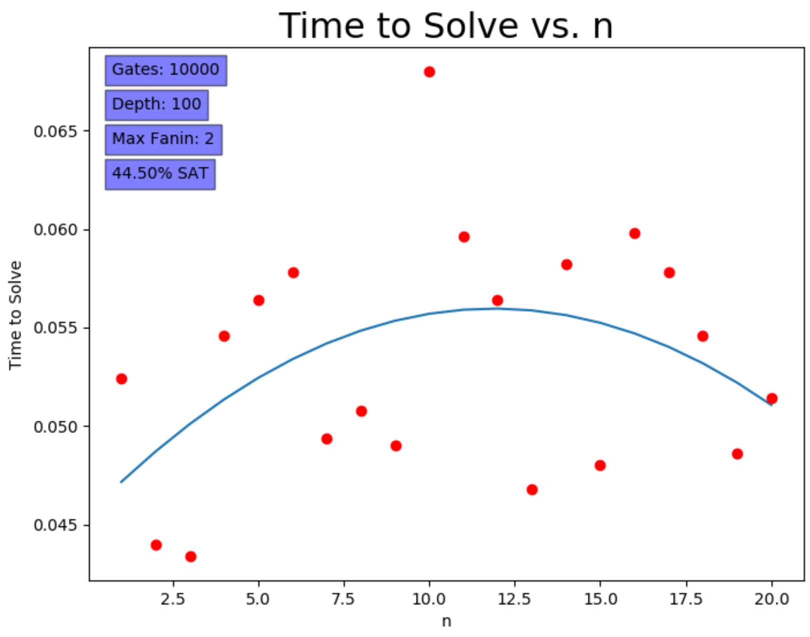


Figure 5: Example 2

6 Conclusion

In our original experiment we experimented with changing the values for n (5 to 1000), m (10 to 10000), and d (3 to 1000). For all smaller instances of the problem, the solver always solved it very fast. When d = m, it was always find a solution immediately as well. Even when the solver took some time to find a solution, there doesn't seem to be a relation between the time it took to solve the problem and n.

In the second experiment, we used the parameters discussed in section 5, displayed in Appendix B. We can now clearly see a relation between time to solve, m, and $fanin$ (Appendix C). The main takeaway from these experiments is that the number of gates in a circuit and the fanin for each gate are the variables that have the largest effect on the time to determine if a circuit is satisfiable.

References

[Wik14] Wikipedia contributors. Circuit satisfiability problem — Wikipedia, the free encyclopedia. 2014. [Online; accessed 8-July-2018].

A Results from Experiment 1

n (start, stop, increment)	m	d (<= m)	Max Time	Figure No.
5, 100, 5	10	3	0	
”	100	3	0	
”	1000	3	0	
”	10000	3	0.45	
”	10	10	0	
”	100	10	0.004	
”	1000	10	0.012	
”	10000	10	0.5	
”	10			
”	100	100	0	
”	1000	100	0.025	
”	10000	100	0.8	1
”	10			
”	100			
”	1000	1000	0	
	10000	1000	0.5	
5, 1000, 50	10	3	0	
”	100	3	0	
”	1000	3	0.025	
”	10000	3	0.6	
”	10	10	0	
”	100	10	0	
”	1000	10	0.25	
”	10000	10	0.058	
”	10			
”	100	100	0	
”	1000	100	0.0025	
”	10000	100	0.075	
”	10			
”	100			
”	1000	1000	0	2
”	10000	1000	0.05	3

B Experiment 2 Results

Max Inputs	Gates	Depth	Fanin	SAT %	Max Time
20	3	2	2	97	0
20	3	2	4	97	0
20	3	2	10	97	0
20	3	2	50	97	0
20	10	2	2	98	0
20	10	2	4	92	0
20	10	2	10	84	0
20	10	2	50	83	0
20	10	5	2	83	0
20	10	5	4	76	0
20	10	5	10	79	0
20	10	5	50	77	0
20	10	10	2	100	0
20	10	10	4	100	0
20	10	10	10	100	0
20	10	10	50	100	0
20	100	2	2	97	0
20	100	2	4	95	0
20	100	2	10	81	0
20	100	2	50	72	0
20	100	5	2	92	0
20	100	5	4	76	0
20	100	5	10	54	0
20	100	5	50	52	0
20	100	10	2	75	0
20	100	10	4	64	0
20	100	10	10	49	0
20	100	10	50	46	0
20	100	100	2	100	0
20	100	100	4	100	0
20	100	100	10	100	0
20	100	100	50	100	0
20	1000	2	2	98	0
20	1000	2	4	94	0
20	1000	2	10	77	0.0024
20	1000	2	50	76	0.0068
20	1000	5	2	92	0
20	1000	5	4	81	0.0012
20	1000	5	10	56	0.012
20	1000	5	50	43	0.019
20	1000	10	2	85	0
20	1000	10	4	57	0.0032

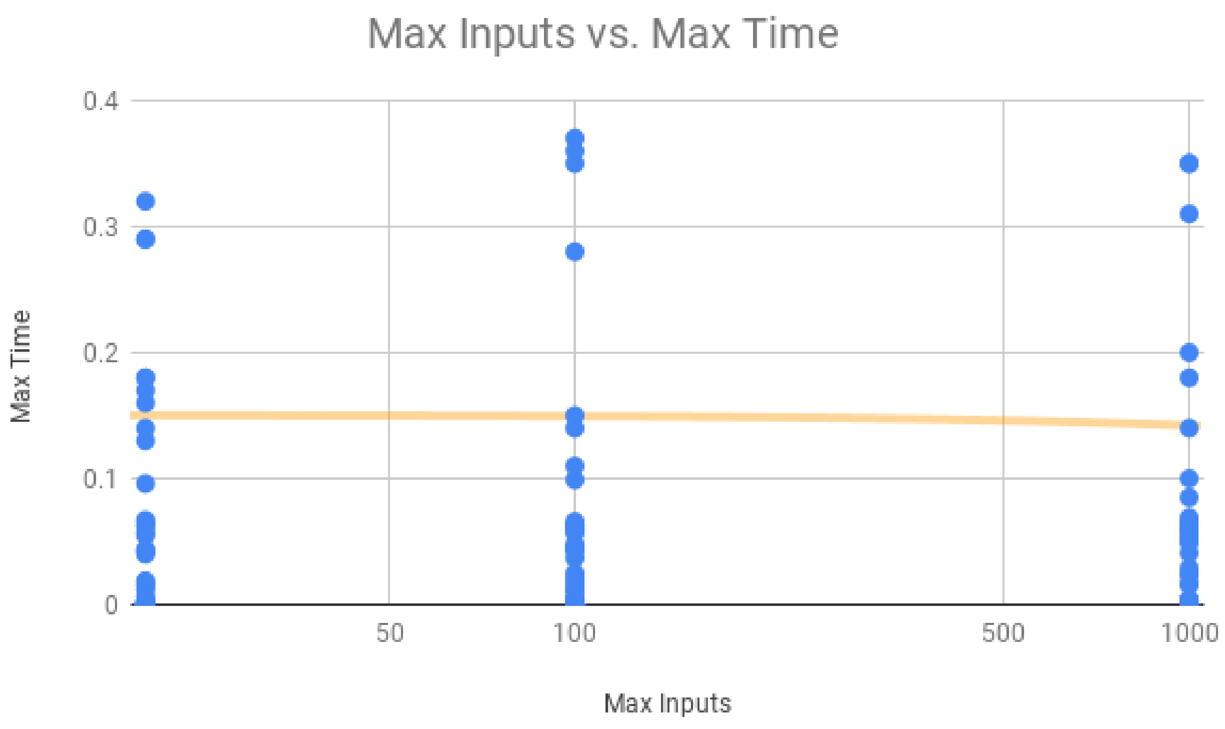
20	1000	10	10	51	0.015
20	1000	10	50	44	0.017
20	1000	100	2	48	0.0008
20	1000	100	4	51	0.002
20	1000	100	10	48	0.0032
20	1000	100	50	51	0.0028
20	1000	1000	2	100	0
20	1000	1000	4	100	0
20	1000	1000	10	100	0
20	1000	1000	50	100	0
20	10000	2	2	97	0.14
20	10000	2	4	93	0.16
20	10000	2	10	82	0.18
20	10000	2	50	71	0.17
20	10000	5	2	91	0.04
20	10000	5	4	77	0.055
20	10000	5	10	59	0.13
20	10000	5	50	54	0.29
20	10000	10	2	85	0.042
20	10000	10	4	67	0.062
20	10000	10	10	54	0.18
20	10000	10	50	50	0.32
20	10000	100	2	51	0.057
20	10000	100	4	55	0.066
20	10000	100	10	44	0.096
20	10000	100	50	48	0.29
20	10000	1000	2	50	0.043
20	10000	1000	4	46	0.044
20	10000	1000	10	46	0.063
20	10000	1000	50	51	0.067
100	3	2	2	100	0
100	3	2	4	100	0
100	3	2	10	99	0
100	3	2	50	99	0
100	10	2	2	99	0
100	10	2	4	97	0
100	10	2	10	92	0
100	10	2	50	87	0
100	10	5	2	79	0
100	10	5	4	78	0
100	10	5	10	83	0
100	10	5	50	77	0
100	10	10	2	100	0
100	10	10	4	100	0

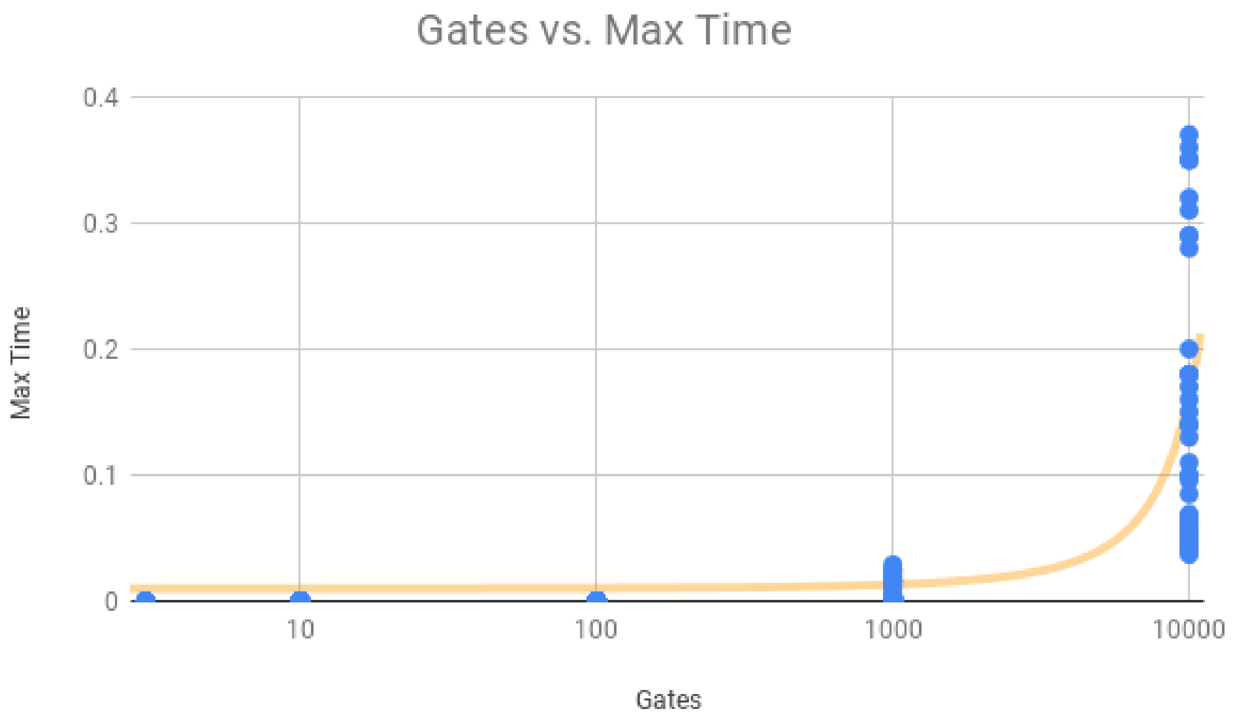
100	10	10	10	100	0
100	10	10	50	100	0
100	100	2	2	100	0
100	100	2	4	98	0
100	100	2	10	91	0
100	100	2	50	76	0
100	100	5	2	97	0
100	100	5	4	79	0
100	100	5	10	56	0
100	100	5	50	53	0
100	100	10	2	82	0
100	100	10	4	58	0
100	100	10	10	47	0
100	100	10	50	44	0
100	100	100	2	100	0
100	100	100	4	100	0
100	100	100	10	100	0
100	100	100	50	100	0
100	1000	2	2	99	0
100	1000	2	4	100	0
100	1000	2	10	89	0.0036
100	1000	2	50	75	0.01
100	1000	5	2	97	0
100	1000	5	4	85	0.0032
100	1000	5	10	62	0.013
100	1000	5	50	50	0.021
100	1000	10	2	92	0
100	1000	10	4	66	0.0036
100	1000	10	10	51	0.017
100	1000	10	50	45	0.025
100	1000	100	2	49	0.0012
100	1000	100	4	51	0.002
100	1000	100	10	46	0.0028
100	1000	100	50	53	0.0036
100	1000	1000	2	100	0
100	1000	1000	4	100	0
100	1000	1000	10	100	0
100	1000	1000	50	100	0
100	10000	2	2	100	0.045
100	10000	2	4	97	0.062
100	10000	2	10	86	0.11
100	10000	2	50	70	0.36
100	10000	5	2	99	0.043
100	10000	5	4	87	0.059

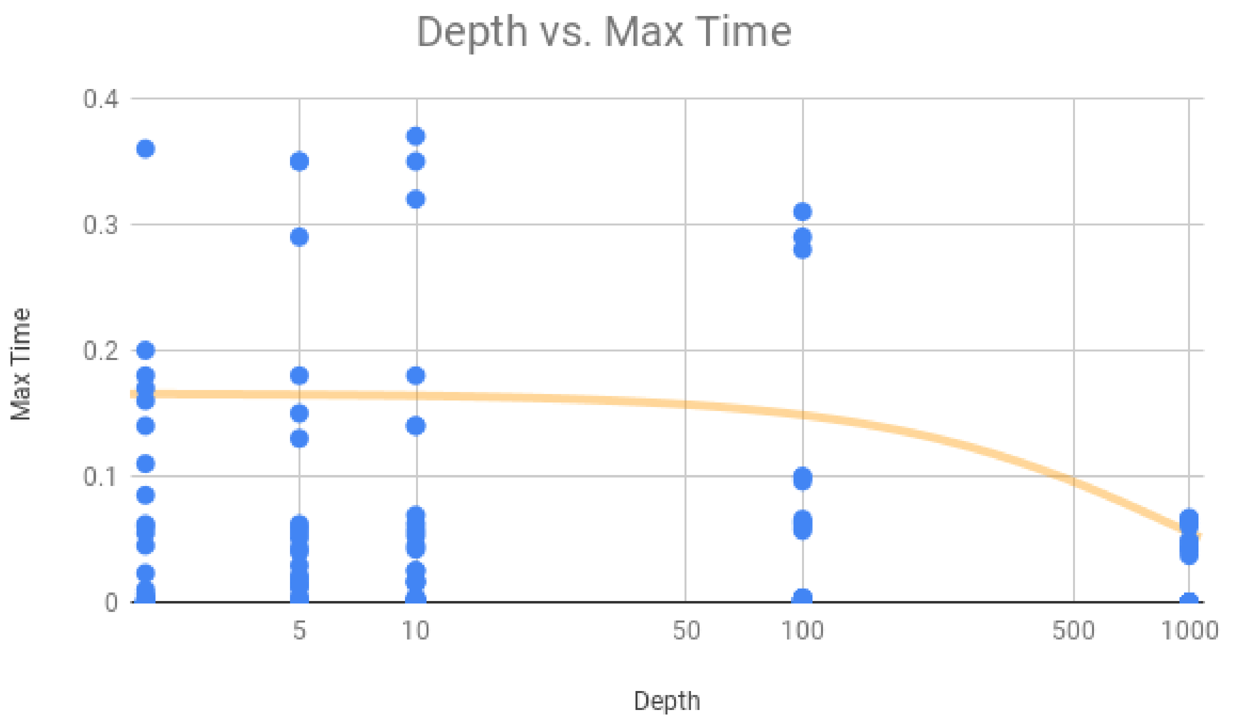
100	10000	5	10	58	0.15
100	10000	5	50	50	0.35
100	10000	10	2	96	0.045
100	10000	10	4	68	0.057
100	10000	10	10	49	0.14
100	10000	10	50	45	0.37
100	10000	100	2	55	0.063
100	10000	100	4	45	0.061
100	10000	100	10	49	0.099
100	10000	100	50	51	0.28
100	10000	1000	2	49	0.037
100	10000	1000	4	48	0.048
100	10000	1000	10	49	0.066
100	10000	1000	50	48	0.063
1000	3	2	2	100	0
1000	3	2	4	99	0
1000	3	2	10	100	0
1000	3	2	50	100	0
1000	10	2	2	99	0
1000	10	2	4	99	0
1000	10	2	10	98	0
1000	10	2	50	97	0
1000	10	5	2	83	0
1000	10	5	4	79	0
1000	10	5	10	82	0
1000	10	5	50	79	0
1000	10	10	2	100	0
1000	10	10	4	100	0
1000	10	10	10	100	0
1000	10	10	50	100	0
1000	100	2	2	100	0
1000	100	2	4	100	0
1000	100	2	10	93	0
1000	100	2	50	75	0
1000	100	5	2	96	0
1000	100	5	4	89	0
1000	100	5	10	58	0
1000	100	5	50	52	0
1000	100	10	2	79	0
1000	100	10	4	59	0
1000	100	10	10	56	0
1000	100	10	50	49	0
1000	100	100	2	100	0
1000	100	100	4	100	0

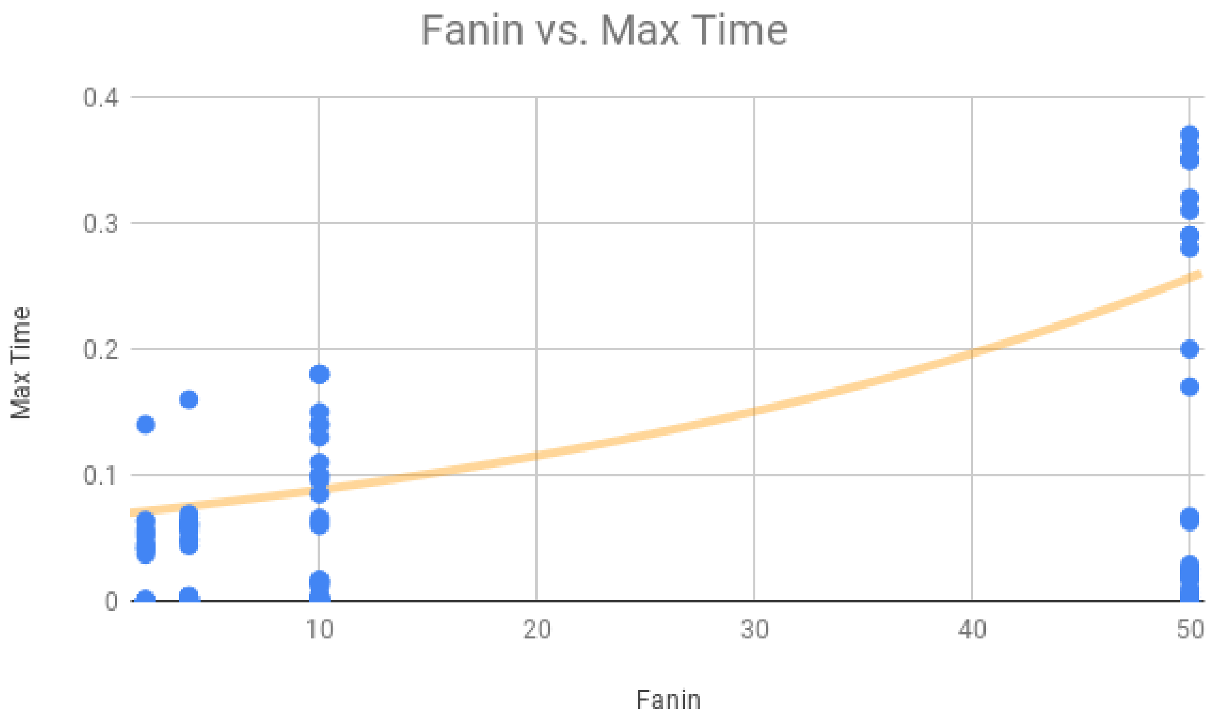
1000	100	100	10	100	0
1000	100	100	50	100	0
1000	1000	2	2	100	0.0004
1000	1000	2	4	100	0.0028
1000	1000	2	10	98	0.004
1000	1000	2	50	76	0.023
1000	1000	5	2	99	0
1000	1000	5	4	94	0.004
1000	1000	5	10	73	0.016
1000	1000	5	50	53	0.029
1000	1000	10	2	94	0.0004
1000	1000	10	4	66	0.0032
1000	1000	10	10	56	0.016
1000	1000	10	50	48	0.025
1000	1000	100	2	48	0.0008
1000	1000	100	4	48	0.0028
1000	1000	100	10	44	0.0028
1000	1000	100	50	52	0.0032
1000	1000	1000	2	100	0
1000	1000	1000	4	100	0
1000	1000	1000	10	100	0
1000	1000	1000	50	100	0
1000	10000	2	2	100	0.055
1000	10000	2	4	99	0.06
1000	10000	2	10	97	0.085
1000	10000	2	50	70	0.2
1000	10000	5	2	99	0.051
1000	10000	5	4	98	0.062
1000	10000	5	10	71	0.18
1000	10000	5	50	57	0.35
1000	10000	10	2	98	0.053
1000	10000	10	4	86	0.069
1000	10000	10	10	51	0.14
1000	10000	10	50	53	0.35
1000	10000	100	2	52	0.064
1000	10000	100	4	46	0.059
1000	10000	100	10	49	0.1
1000	10000	100	50	55	0.31
1000	10000	1000	2	43	0.041
1000	10000	1000	4	49	0.049
1000	10000	1000	10	48	0.06
1000	10000	1000	50	51	0.066

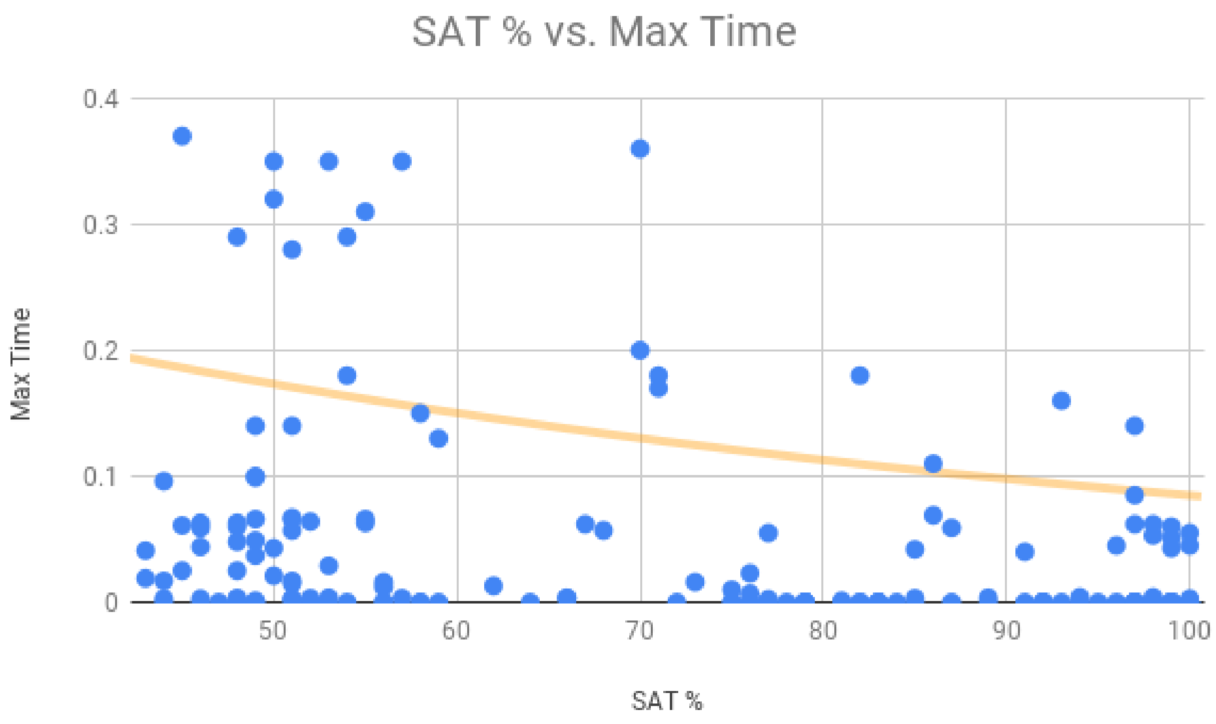
C Variables vs. Max Time from Experiment 2




Gates vs. Max Time
Max Time
0.4
0.3
0.2
0.1
0
10
100
1000
10000
Gates


Depth vs. Max Time
Max Time
0.4
0.3
0.2
0.1
0
5
10
50
100
500
1000
Depth


Fanin vs. Max Time
Max Time
0.4
0.3
0.2
0.1
0
10
20
30
40
50
Fanin


SAT % vs. Max Time
Max Time
0.4
0.3
0.2
0.1
0
50
60
70
80
90
100
SAT %