MaxSAT Evaluation 2020 - Benchmark: Identifying Maximum Probability Minimal Cut Sets in Fault Trees

Martín Barrère and Chris Hankin
Institute for Security Science and Technology, Imperial College London, UK
{m.barrere, c.hankin}@imperial.ac.uk

***Abstract*—This paper presents a MaxSAT benchmark focused on the identification of Maximum Probability Minimal Cut Sets (MPMCSs) in fault trees. We address the MPMCS problem by transforming the input fault tree into a weighted logical formula that is then used to build and solve a Weighted Partial MaxSAT problem. The benchmark includes 80 cases with fault trees of different size and composition as well as the optimal cost and solution for each case.**

***Index Terms*—MaxSAT, Benchmark, Fault trees, Fault Tree Analysis, Reliability, Cyber-Physical Security, Dependability.**

I. Problem overview

Fault Tree Analysis (FTA) is an analytical tool aimed at modelling and evaluating how complex systems may fail. FTA is widely used as a risk assessment tool in safety and reliability engineering for a broad range of industries including aerospace, power plants, nuclear plants, and others high-hazard fields [1]. Essentially, a fault tree is a directed acyclic graph (DAG) which involves a set of basic events (e.g. component failures) that are combined using logic operators (e.g. AND and OR gates) to model how these events may lead to an undesired state of the system normally represented at the root of the tree (top level event).

Our work is focused on a novel measure for FTA in the form of a hybrid analysis technique that involves quantitative and qualitative aspects of fault trees. From a qualitative perspective, we focus on Minimal Cut Sets (MCS). An MCS is a minimal combination of events that together cause the top level event. As such, MCSs are fundamental for structural analysis. The problem is that, in large scenarios, computing all MCSs might be very expensive and there might be hundreds of MCSs, which makes it hard to handle and prioritise which MCSs should be attended first. In that context, the goal of this work is to identify the MCS with maximum probability. We call this problem the MPMCS. This is an NP-complete problem and we use a MaxSAT-based approach to address it.

II. Simple example

The fault tree shown in Fig. 1 illustrates the different combinations of events that may lead to the failure of an hypothetical Fire Protection System (FPS) based on [2]. The FPS can fail if either the fire detection system or the fire suppression mechanism fails. In turn, the detection system can fail if both sensors fail simultaneously (events x_1 and x_2), while the suppression mechanism may fail if there is no water ($x3$), the sprinkler nozzles are blocked (x_4), or the triggering system does not work. The latter can fail if neither of its operation modes (automatic (x_5) or remotely operated) works properly. The remote control can fail if the communications channel fails (x_6) or the channel is not available due to a cyber attack, e.g. DDoS attack (x_7). Each basic event has an associated value that indicates its probability of occurrence $p(x_i)$.

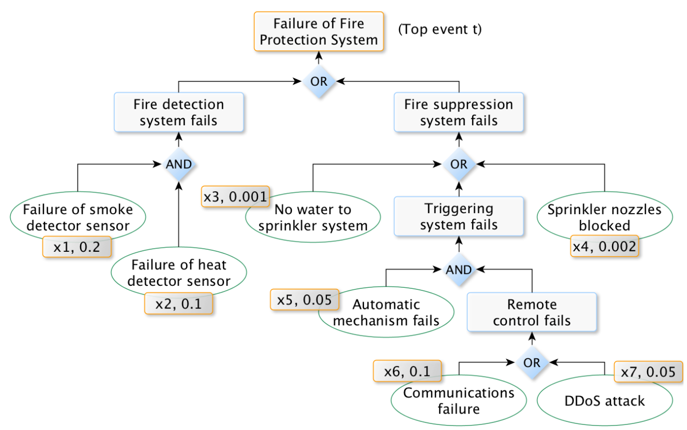

Fig. 1. Fault tree of a cyber-physical fire protection system (simplified)

A fault tree F can be represented as a Boolean equation $f(t)$ that expresses the different ways in which the top event t can be satisfied [3]. In our example, $f(t)$ is as follows:

$$f(t) = (x_1 \wedge x_2) \vee (x_3 \vee x_4 \vee (x_5 \wedge (x_6 \vee x_7)))$$

The objective is to find the minimal set of logical variables that makes the equation $f(t)$ *true* and whose joint probability is maximal among all minimal sets. In our example, the MPMCS is $\{x_1, x_2\}$ with a joint probability of 0.02.

III. MaxSAT formulation strategy

Given a fault tree and its logical formulation $f(t)$, we carry out a series of steps to compute the MPMCS as follows.

1. Logical transformation. Since we are interested in minimising the number of satisfied clauses, which is opposed to what MaxSAT does (maximisation), we flip all logic gates but keep all events in their positive form. In our example, we obtain: $g(t) = (x_1 \vee x_2) \wedge (x_3 \wedge x_4 \wedge (x_5 \vee (x_6 \wedge x_7)))$.

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant No 739551 (KIOS CoE). To appear in Proceedings of the MaxSAT Evaluation 2020 (MSE'20), https://maxsat-evaluations.github.io/2020/.

Then, the objective is to satisfy $\neg g(t)$ where the falsified variables will indicate the minimum set of events that must simultaneously occur to trigger the top level event. A more detailed explanation of this transformation can be found in [4]. We then use the Tseitin transformation to produce in polynomial time an equisatisfiable CNF formula [5].

2. MaxSAT weights. Due to the fact that MaxSAT is additive in nature and the MPMCS problem involves the multiplication of decision variables, we transform the probabilities into a negative log-space so the multiplication becomes a sum. In addition, many SAT solvers only support integer weights so we perform a second transformation by right shifting (multiplying by 10) every value until the smallest value is covered with an acceptable level of precision. For example, 0.001 and 0.00007 would become 100 and 7 (right shift 5 times). Additional variables introduced by the Tseitin transformation have weight 0. We then specify the problem as a Partial Weighted MaxSAT instance by assigning the transformed probability values as a penalty score for each decision variable.

3. Parallel SAT-solving architecture. Since different SAT solvers normally use different resolution techniques, some of them are very good at some instances and not that good at others. To address this issue, we run multiple SAT-solvers in parallel and pick the solution of the solver that finishes first. We have experimentally observed that the combination of different solvers provides good results in terms of performance and scalability. Once the solution has been found, we translate back the transformed values into their stochastic domain and output the MCS with maximum probability.

IV. Fault tree generation

The benchmark presented in this paper relies on our open source tool MPMCS4FTA [6]. We have used MPMCS4FTA to generate and analyse synthetic pseudo-random fault trees of different size and composition. We use AND/OR graphs as the underlying structure to represent fault trees. The benchmark presented in [7] also considers AND/OR graphs as a means to represent operational dependencies between components in industrial control systems [8]. However, the instances presented in this paper differ in that: 1) they are restricted to directed acyclic graphs (DAGs), 2) only the basic events represented at the leaves of the fault tree involve a probability of failure, and 3) leaves can have more than one parent in order to relax the definition of strict logical trees.

We control the size and composition of a random fault tree of size n according to a configuration $R = (R_{AT}, R_{AND}, R_{OR})$. $R_{AT} \in [0, 1]$ indicates the proportion of atomic nodes (basic events) with respect to size n (e.g. 0.2 means 20%) whereas R_{AND} and R_{OR} indicate the proportion of AND and OR nodes respectively. To create a fault tree of size n, we first create two lists: $L = \{l_1, \dots, l_m\}$ and $A = \{a_1, \dots, a_s\}$. L is a random sequence of AND and OR nodes with the specified proportions for each operator where $m = n * (R_{AND} + R_{OR})$. A is a list of atomic nodes where $s = n * R_{AT}$, thus $n = m + s$. In addition, each atomic node has a random probability of failure $p(a_i) \in [0, 1]$.

To ensure connectivity, we first create the root node t and connect l_1 to t ($l_1 \rightarrow t$). Then, for each logic node l_i in the sequence L, we randomly choose k nodes l_j ahead (thus $j > i$) and create k edges ($l_j \rightarrow l_i$) in the tree. When the remaining nodes in L are not enough to cover k nodes, we use random atomic nodes from A. At this point, we also make sure that l_i points to at least one previous node in the sequence L. If that is not the case, we choose a random node l_h (with $h < i$) and create an edge ($l_i \rightarrow l_h$). Once the sequence L has been processed, we traverse the list A and connect each atomic node a_i as follows. First, we draw a random value k' between 1 and k. Then, we add random edges ($a_i \rightarrow l_j$) from a_i to logic nodes l_j until we cover k' connections.

V. Benchmark description

Out dataset includes 80 cases in total, and can be obtained at [6]. It contains fault trees with four different sizes: 2500, 5000, 7500, and 10000 nodes (20 cases each). For each tree size, we consider two different graph configurations, $R_1 = (0.8, 0.1, 0.1)$ and $R_2 = (0.6, 0.2, 0.2)$, which determine the composition of the fault trees (10 cases each). Table I shows the identifiers of the cases within each one of these categories.

#Nodes/Configurations	$R_1 = (0.8, 0.1, 0.1)$	$R_2 = (0.6, 0.2, 0.2)$
2500	1 to 10	11 to 20
5000	21 to 30	31 to 40
7500	41 to 50	51 to 60
10000	61 to 70	71 to 80

TABLE I
BENCHMARK CASES AND CONFIGURATIONS

Each case is specified in an individual **.wcnf** (DIMACS-like, weighted CNF) file named with the case id and the number of nodes involved. The weight for hard clauses (*top* value) has been set to 2.0×10^9. The weight of each soft constraint is an integer value that corresponds to the transformation (right shifting) of the probability value in $-log$ space. Tables II and III detail each case as well as the results obtained with our tool. The field **id** identifies each case. **gNodes** and **gEdges** indicate the total number of nodes and edges in the fault tree. **gAT**, **gAND**, and **gOR**, indicate the approximate composition of the graph in terms of atomic (basic events), AND, and OR nodes. **tsVars** and **tsClauses** show the number of variables and clauses involved in the MaxSAT formulation after applying the Tseitin transformation. **time** shows the resolution time reported by MPMCS4FTA in milliseconds. Currently, the MaxSAT solvers used in MPMCS4FTA are SAT4J [9] and a Python-based linear programming approach using Gurobi [10]. **size** indicates the number of nodes identified in the MPMCS solution. **intLogCost** indicates the cost of the solution in $-log$ space as an integer value (right shifted). **logCost** indicates the cost of the solution in $-log$ space. **MPMCS probability** indicates the joint probability of the MPMCS. These experiments have been performed on a MacBook Pro (16-inch, 2019), 2.4 GHz 8-core Intel Core i9, 32 GB 2666 MHz DDR4.

id	gNodes	gEdges	gAT	gAND	gOR	tsVars	tsClauses	time	size	intLogCost	logCost	MPMCS probability
1	2500	7151	2002	250	250	1978	6258	618	1	246	2.46E-4	0.999754
2	2500	7192	2002	250	250	4268	15026	850	447	464771733	464.771733	1.42239870668983E-202
3	2500	7196	2002	250	250	1207	3763	290	1	27591	0.027591	0.972787
4	2500	7140	2002	250	250	4211	14673	833	1	238	2.38E-4	0.999763
5	2500	7107	2002	250	250	3907	13325	821	1	7879	0.007879	0.992153
6	2500	7202	2002	250	250	3410	11350	749	70	81474531	81.474531	4.147681160335815E-36
7	2500	7126	2002	250	250	3304	10922	711	1	315	3.15E-4	0.999685
8	2500	7181	2002	250	250	3752	12713	826	1	2576	0.002576	0.997428
9	2500	7157	2002	250	250	3011	9847	625	1	4301	0.004301	0.995709
10	2500	7156	2002	250	250	642	1982	211	19	12423488	12.423488	4.0231156723921624E-6
11	2500	6831	1502	500	500	3873	14170	912	1	28842	0.028842	0.971571
12	2500	6782	1502	500	500	2377	7941	550	1	32680	0.03268	0.96785
13	2500	6814	1502	500	500	3216	11235	700	13	10769787	10.769787	2.1025796252653052E-5
14	2500	6700	1502	500	500	3268	11376	728	197	207945092	207.945092	4.9088521396478804E-91
15	2500	6897	1502	500	500	3063	10555	817	1	3262	0.003262	0.996744
16	2500	6849	1502	500	500	2044	6765	470	1	191116	0.191116	0.826037
17	2500	6787	1502	500	500	3158	10955	723	1	284520	0.28452	0.752376
18	2500	6872	1502	500	500	3433	12147	773	139	130484455	130.484455	2.1453798325228181E-57
19	2500	6821	1502	500	500	2506	8439	534	17	9662887	9.662887	6.36019885647539E-5
20	2500	6831	1502	500	500	3848	14095	821	1	3507	0.003507	0.996501
21	5000	14324	4002	500	500	4149	13224	932	229	217397271	217.397271	3.8565352927569054E-95
22	5000	14313	4002	500	500	8532	29961	925	614	641968767	641.968767	1.5912873405576694E-279
23	5000	14329	4002	500	500	6971	23338	842	240	251915559	251.915559	3.9351584673463555E-110
24	5000	14361	4002	500	500	8020	27645	843	1	793	7.93E-4	0.999209
25	5000	14370	4002	500	500	8965	32190	843	1	1858	0.001858	0.998144
26	5000	14317	4002	500	500	5443	17581	827	1	3615	0.003615	0.996391
27	5000	14407	4002	500	500	8113	28023	842	277	253971185	253.971185	5.035082961027143E-111
28	5000	14365	4002	500	500	8952	32153	837	1041	994658460	994.65846	0.0
29	5000	14321	4002	500	500	8859	31477	833	379	378308687	378.308687	5.051735441001231E-165
30	5000	14316	4002	500	500	7948	27315	830	1	970	9.7E-4	0.999032
31	5000	13607	3002	1000	1000	6384	22218	938	1	2530	0.00253	0.997474
32	5000	13730	3002	1000	1000	7330	26390	863	65	63984958	63.984958	1.62844121698006E-28
33	5000	13687	3002	1000	1000	3181	10354	683	1	25289	0.025289	0.975029
34	5000	13600	3002	1000	1000	6293	21870	834	407	424495269	424.495269	4.413071223454673E-185
35	5000	13712	3002	1000	1000	7361	26650	895	179	171277203	171.277203	4.1251154050451916E-75
36	5000	13709	3002	1000	1000	6231	21647	831	22	19249301	19.249301	4.366753474609794E-9
37	5000	13612	3002	1000	1000	6202	21523	931	257	273826234	273.826234	1.2035873310274229E-119
38	5000	13664	3002	1000	1000	4482	14952	824	1	4317	0.004317	0.995693
39	5000	13631	3002	1000	1000	7395	26641	827	83	89562456	89.562456	1.2695246380697898E-39
40	5000	13641	3002	1000	1000	7825	28775	831	1	5974	0.005974	0.994045

TABLE II
BENCHMARK DESCRIPTION - CASES 1 TO 40

id	gNodes	gEdges	gAT	gAND	gOR	tsVars	tsClauses	time	size	intLogCost	logCost	MPMCS probability
41	7500	21502	6002	750	750	8871	28951	965	1	160	1.6E-4	0.999841
42	7500	21515	6002	750	750	7191	23069	852	1	393	3.93E-4	0.999607
43	7500	21497	6002	750	750	5716	18114	843	1	1095	0.001095	0.998906
44	7500	21536	6002	750	750	6476	20645	849	600	607247314	607.247314	1.8912103369207186E-264
45	7500	21472	6002	750	750	10277	34266	859	251	235979386	235.979386	3.279829621872166E-103
46	7500	21607	6002	750	750	10235	34064	849	31	27638401	27.638401	9.927826703704467E-13
47	7500	21609	6002	750	750	11377	38597	920	689	644477962	644.477962	1.2810988897753624E-280
48	7500	21397	6002	750	750	4488	14083	815	1	18442	0.018442	0.981728
49	7500	21410	6002	750	750	12792	44789	1031	668	672741572	672.741572	6.81228495760467E-293
50	7500	21566	6002	750	750	13253	47290	851	1	9154	0.009154	0.990888
51	7500	20454	4502	1500	1500	11031	39763	972	1	2151	0.002151	0.997852
52	7500	20450	4502	1500	1500	8927	30739	855	1	738	7.38E-4	0.999263
53	7500	20616	4502	1500	1500	11843	43792	894	1	37	3.7E-5	0.999964
54	7500	20530	4502	1500	1500	9961	35071	1053	502	480184105	480.184105	2.8797108920892045E-209
55	7500	20563	4502	1500	1500	9462	32930	1368	769	739302414	739.302414	8.45E-322
56	7500	20493	4502	1500	1500	9084	31398	833	1	7545	0.007545	0.992484
57	7500	20491	4502	1500	1500	4922	16088	817	1	104472	0.104472	0.9008
58	7500	20594	4502	1500	1500	5943	19507	987	267	256660486	256.660486	3.4340775952647096E-112
59	7500	20406	4502	1500	1500	9340	32356	898	158	148111431	148.111431	4.74472781242486E-65
60	7500	20445	4502	1500	1500	8882	30572	827	1	14066	0.014066	0.986033
61	10000	28613	8002	1000	1000	16234	56222	1087	1	1904	0.001904	0.998099
62	10000	28675	8002	1000	1000	14261	47804	914	197	185985480	185.98548	1.6901841317920728E-81
63	10000	28558	8002	1000	1000	13755	45717	893	1	43	4.3E-5	0.999957
64	10000	28738	8002	1000	1000	13370	44343	882	1	127	1.27E-4	0.999874
65	10000	28752	8002	1000	1000	15537	53105	917	643	606121928	606.121928	5.826520007473361E-264
66	10000	28803	8002	1000	1000	9981	32065	852	1	796	7.96E-4	0.999205
67	10000	28632	8002	1000	1000	13418	44550	861	448	439405919	439.405919	1.4772121624185204E-191
68	10000	28830	8002	1000	1000	17774	63650	874	1	3047	0.003047	0.996959
69	10000	28717	8002	1000	1000	14505	48831	861	1	1691	0.001691	0.998311
70	10000	28604	8002	1000	1000	16032	55089	855	1	436	4.36E-4	0.999564
71	10000	27114	6002	2000	2000	15244	55476	2286	652	652324945	652.324945	5.016628484164324E-284
72	10000	27515	6002	2000	2000	10588	36029	867	1	15974	0.015974	0.984154
73	10000	27411	6002	2000	2000	9596	32332	862	422	440653751	440.653751	4.240514855635819E-192
74	10000	27271	6002	2000	2000	15985	59167	873	1	2033	0.002033	0.997969
75	10000	27228	6002	2000	2000	13506	47651	2223	621	639112478	639.112478	2.7423451190246526E-278
76	10000	27345	6002	2000	2000	12066	41598	1253	326	307525901	307.525901	2.779537506735469E-134
77	10000	27310	6002	2000	2000	10310	34812	835	1	10970	0.01097	0.989091
78	10000	27306	6002	2000	2000	12092	41711	1004	228	218680041	218.680041	1.0684631282749114E-95
79	10000	27315	6002	2000	2000	14069	50130	848	1	1447	0.001447	0.998555
80	10000	27375	6002	2000	2000	14851	53699	859	1	180	1.8E-4	0.999821

TABLE III
BENCHMARK DESCRIPTION - CASES 41 TO 80

References

[1] E. Ruijters and M. Stoelinga, “Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools,” *Computer Science Review*, vol. 15-16, pp. 29 – 62, 2015.

[2] S. Kabir, “An overview of Fault Tree Analysis and its application in model based dependability analysis,” *Expert Systems with Applications*, vol. 77, pp. 114 – 135, 2017.

[3] W. Vesely, M. Stamatelatos, J. Dugan, J. Fragola, J. Minarick III, and J. Railsback, “Fault Tree Handbook with Aerospace Applications,” *Office of Safety and Mission Assurance, NASA Headquarters, US*, 2002.

[4] M. Barrère and C. Hankin, “Fault Tree Analysis: Identifying Maximum Probability Minimal Cut Sets with MaxSAT,” https://arxiv.org/abs/2005.03003, May 2020.

[5] G. S. Tseitin, “On the Complexity of Derivation in Propositional Calculus,” in *Studies in Constructive Mathematics and Mathematical Logic, Part II*, A. Slisenko, Ed., 1970, pp. 234–259.

[6] M. Barrère, “MPMCS4FTA - Maximum Probability Minimal Cut Sets for Fault Tree Analysis,” https://github.com/mbarrere/mpmcs4fta, March 2020.

[7] M. Barrère, C. Hankin, N. Nicolaou, D. Eliades, and T. Parisini, “MaxSAT Evaluation 2019 - Benchmark: Identifying Security-Critical Cyber-Physical Components in Weighted AND/OR Graphs. In MaxSAT Evaluation 2019 (MSE’19),” https://arxiv.org/abs/1911.00516, 2019.

[8] M. Barrère, C. Hankin, N. Nicolaou, D. Eliades, and T. Parisini, “Measuring cyber-physical security in industrial control systems via minimum-effort attack strategies,” *Journal of Information Security and Applications*, vol. 52, pp. 1–17, June 2020. [Online]. Available: https://doi.org/10.1016/j.jisa.2020.102471

[9] “SAT4J,” http://www.sat4j.org/, Cited June 2020.

[10] Gurobi, “Gurobi Optimizer,” https://www.gurobi.com/, 2020, Cited June 2020.