MaxSAT Evaluation 2019 - Benchmark: Identifying Security-Critical Cyber-Physical Components in Weighted AND/OR Graphs

Martín Barrère*, Chris Hankin*, Nicolas Nicolaou†, Demetrios G. Eliades†, Thomas Parisini‡
*Institute for Security Science and Technology, Imperial College London, UK
{m.barrere, c.hankin}@imperial.ac.uk
†KIOS Research and Innovation Centre of Excellence, University of Cyprus
{nicolasn, eldemet}@ucy.ac.cy
‡Department of Electrical and Electronic Engineering, Imperial College London, UK
{t.parisini}@imperial.ac.uk

***Abstract*—This paper presents a MaxSAT benchmark focused on identifying critical nodes in AND/OR graphs. We use AND/OR graphs to model Industrial Control Systems (ICS) as they are able to semantically grasp intricate logical interdependencies among ICS components. However, identifying critical nodes in AND/OR graphs is an NP-complete problem. We address this problem by efficiently transforming the input AND/OR graph-based model into a weighted logical formula that is then used to build and solve a Weighted Partial MAX-SAT problem. The benchmark includes 80 cases with AND/OR graphs of different size and composition as well as the optimal cost and solution for each case.**

I. Problem overview

Over the last years, Industrial Control Systems (ICS) such as water treatment plants and energy facilities have become increasingly exposed to a wide range of cyber-physical threats, having massive destructive consequences. Our work is focused on security metrics and techniques that can be used to measure and improve the security posture of ICS environments [1]. We use AND/OR graphs to model these systems as they allow more realistic representations of the complex interdependencies among cyber-physical components that are normally involved in real-world settings [2], [3]. In that context, we have designed a security metric, detailed in [1], whose objective is to identify the set of critical AND/OR nodes (ICS network components) that must be compromised in order to disrupt the operation of the system, with minimal cost for the attacker.

From a graph-theoretical perspective, our security metric looks for a minimal weighted vertex cut in AND/OR graphs. This is an NP-complete problem as shown in [3], [4], [5]. While well-known algorithms such as Max-flow Min-cut [6] and variants of it could be used to estimate such metric over OR graphs in polynomial time, their use for general AND/OR graphs is not evident nor trivial as they may fail to capture the underlying logical semantics of the graph. In that context, we take advantage of state-of-the-art MaxSAT techniques to address our problem.

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant No 739551 (KIOS CoE).

II. Simple example

Let us consider a simple ICS network whose operational dependencies are represented by the AND/OR graph shown in Figure 1. The graph reads as follows: the actuator $c1$ depends on the output of software agent d. Agent d in turn has two alternatives to work properly; it can use either the readings of sensor a and the output from agent b together, or the output from agent b and the readings of sensor c together. In addition, each cyber-physical component has an associated attack cost that represents the effort required by an attacker to compromise that component. Now, considering these costs, the question we are trying to answer is: which nodes should be compromised in order to disrupt the operation of actuator $c1$, with minimal effort (cost) for the attacker? In other words, what is the least-effort attack strategy to disable actuator $c1$?

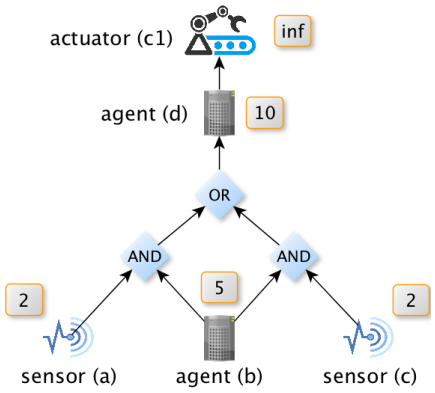

Fig. 1. AND/OR graph with sensors, software agents and actuators

Our example involves many attack alternatives, however, only one is minimal. The optimal strategy is to compromise nodes a and c with a total cost of 4. The compromise of these sensors will disable both AND nodes and consecutively the OR node, which in turn will affect node d and finally node $c1$.

III. MAX-SAT FORMULATION STRATEGY

Given a target node t, the input graph G can be used as a map to decode the dependencies that node t relies on. Therefore, G can be traversed backwards in order to produce a propositional formula that represents the different ways in which node t can be fulfilled. We call this transformation $f_G(t)$. In our example, $f_G(c1)$ is as follows:

$$f_G(c1) = c1 \wedge (d \wedge ((a \wedge b) \vee (b \wedge c)))$$

The goal of the attacker, however, is precisely the opposite, i.e., to disrupt node $c1$ somewhere along the graph. Therefore, we are actually interested in satisfying $\neg f_G(c1)$, which describes the means to disable $c1$. After applying a few logical rules, the conjunctive normal form (CNF) of $\neg f_G(c1)$ is:

$$\neg f_G(c1) = (\neg c1 \vee \neg d \vee \neg a \vee \neg b) \wedge (\neg c1 \vee \neg d \vee \neg b \vee \neg c)$$

In practice, we do not use the naive CNF conversion approach since it might lead to exponential computation times over large graphs. Instead, we use the Tseitin transformation [7], which can be done in polynomial time and essentially produces a new formula in CNF that is not strictly equivalent to the original formula (because there are new variables) but is equisatisfiable. This means that, given an assignment of truth values, the new formula is satisfied if and only if the original formula is also satisfied. Under that perspective, a logical assignment such that $\neg f_G(t) = true$ will indicate which nodes must be compromised (i.e. logically falsified) in order to disrupt the operation of the system.

Considering the CNF formula produced by the Tseitin transformation and a cost function $\varphi(n)$ that indicates the attack cost of a node n, we model our problem as a Weighted Partial MAX-SAT problem [8]. Hard clauses are essentially the clauses within the CNF formula:

$$\neg c1 \vee \neg d \vee \neg a \vee \neg b \qquad \neg c1 \vee \neg d \vee \neg b \vee \neg c$$

whereas soft clauses correspond to each atomic node in the graph with their corresponding penalties (costs) as follows:

a	b	c	d	$c1$
$\varphi(a) = 2$	$\varphi(b) = 5$	$\varphi(c) = 2$	$\varphi(d) = 10$	$\varphi(c1) = inf$

Therefore, a MAX-SAT solver will try to minimise the number of falsified variables as well as their weights, which in our problem equals to minimise the compromise cost for the attacker. Note: the additional variables introduced by the Tseitin transformation have cost/weight 0 in the formulation.

IV. AND/OR GRAPH GENERATION

The benchmark presented in this paper relies on META4ICS, a Java-based security metric analyser for ICS [1], [9]. We have used META4ICS to produce and analyse synthetic pseudo-random AND/OR graphs of different size and composition. To create an AND/OR graph of size n, we first create the target node. Afterwards, we create a predecessor which has one of the three types (atomic, AND, OR) according to a probability given by a compositional configuration predefined for the experiment. For example, a configuration of $(60, 20, 20)$ means 60% of atomic nodes, 20% of AND nodes and 20% of OR nodes. We repeat this process creating children on the respective nodes until we approximate the desired size of the graph n. Node costs, represented by $\varphi(n)$, are integer values randomly selected between 1 and 100.

The benchmark also includes the solutions obtained by META4ICS for each case, including resolution time, total cost and critical nodes. Currently, META4ICS uses SAT4J [10] and a Python-based linear programming approach as MaxSAT solvers. The tool runs all available solvers in parallel and picks the first one that comes up with a valid solution.

V. BENCHMARK DESCRIPTION

Out dataset includes 80 cases in total, and can be obtained at [9]. There are four different sizes of AND/OR graphs involving 5000, 10000, 15000, and 20000 nodes (20 cases each). For each graph size, we consider two different graph configurations, 80/10/10 and 60/20/20, which determine the composition of the graphs (10 cases each). Table I shows the identifiers of the cases within each one of these categories.

Nodes/Configurations	**80/10/10 config**	**60/20/20 config**
5000	1 to 10	11 to 20
10000	21 to 30	31 to 40
15000	41 to 50	51 to 60
20000	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 1.0×10^6. Tables II and III detail each case as well as the results obtained with our tool. The field **id** identifies each case; **gNodes** indicates the total number of nodes in the original AND/OR graph; **gAT**, **gAND** and **gOR** indicate the approximate composition of the graph in terms of atomic (cyber-physical components), AND and OR nodes; **tsVars** and **tsClauses** show the number of variables and clauses involved in the MaxSAT formulation after applying the Tseitin transformation; **cost** and **time** show the total solution cost reported by META4ICS and the time needed for its resolution in milliseconds; **solution** shows the set of critical nodes expressed as a list of nodes with their respective costs (weights). These experiments have performed on a MacBook Pro (15-inch, 2018), 2.9 GHz Intel Core i9, 32 GB 2400 MHz DDR4.

As a final remark, it can be observed that some cases with the same size and composition parameters have very different resolution times. This is an interesting phenomenon and it is due to the internal logical composition of the AND/OR graph and how well the underlying solver performs with each case. Within our experiments, we have observed that none of the two solvers used in META4ICS is faster than the other in all of the cases. We believe this is an interesting problem that should be further investigated in the context of MaxSAT solvers.

id	gNodes	gAT	gAND	gOR	tsVars	tsClauses	cost	time	solution
1	5000	3977	512	512	8978	23981	2	1163	[(14:2)]
2	5000	3967	529	505	8968	23971	16	1264	[(2:16)]
3	5000	3984	515	502	8985	23988	2	920	[(98:2)]
4	5000	4008	461	532	9009	24012	8	919	[(8:1),(961:7)]
5	5000	3990	510	501	8991	23994	1	930	[(4057:1)]
6	5000	4002	486	513	9003	24006	5	946	[(788:3),(8431:2)]
7	5000	4026	477	498	9027	24030	20	920	[(8:1),(759:12),(6706:3),(6860:1),(7304:3)]
8	5000	3996	518	487	8997	24000	6	894	[(4:6)]
9	5000	4046	488	467	9047	24050	5	904	[(3177:5)]
10	5000	4026	493	482	9027	24030	5	887	[(4029:4),(5503:1)]
11	5000	3029	979	993	8030	23033	31	880	[(1258:30),(2189:1)]
12	5000	2998	1012	991	7999	23002	68	935	[(920:2),(1128:3),(1324:9),(1926:3),(2690:13),(2773:35),(3441:3)]
13	5000	3021	1008	972	8022	23025	2	879	[(300:2)]
14	5000	2986	1020	995	7987	22990	46	956	[(1027:12),(3624:3),(3778:17),(6453:2),(6496:4),(6544:8)]
15	5000	3014	1006	981	8015	23018	152	1889	[(5:1),(2930:5),(3245:7),(3434:34),(3925:17),(4158:14), (4457:34), (6307:19),(7048:5),(7177:2),(7186:1), (7191:7),(7337:1),(7367:1),(7570:4)]
16	5000	3030	990	981	8031	23034	12	859	[(3:12)]
17	5000	3015	1034	952	8016	23019	3	867	[(4:2),(7561:1)]
18	5000	3008	956	1037	8009	23012	22	877	[(2:19),(6253:3)]
19	5000	3031	970	1000	8032	23035	8	865	[(5444:8)]
20	5000	3033	972	996	8034	23037	13	870	[(2:13)]
21	10000	7983	1017	1001	17984	47987	1	1126	[(16139:1)]
22	10000	8026	996	979	18027	48030	13	1313	[(2:13)]
23	10000	8024	991	986	18025	48028	12	1310	[(24:1),(1580:1),(2201:2),(9663:3),(14360:5)]
24	10000	8058	974	969	18059	48062	14	1184	[(5:9),(12130:2),(12207:2),(12422:1)]
25	10000	8026	989	986	18027	48030	7	1459	[(9009:3),(12030:4)]
26	10000	7984	1003	1014	17985	47988	12	1309	[(7:9),(17023:1),(17074:2)]
27	10000	8051	939	1011	18052	48055	50	6248	[(1997:4),(2744:9),(3398:17),(5610:6),(6304:1),(11315:1),(11771:4), (14136:1),(16399:4),(16831:3)]
28	10000	7998	995	1008	17999	48002	3	1141	[(2:3)]
29	10000	8095	972	934	18096	48099	10	1224	[(8330:4),(11696:5),(13881:1)]
30	10000	8022	987	992	18023	48026	2	1178	[(2:2)]
31	10000	6013	1991	1997	16014	46017	49	1275	[(4:32),(4857:17)]
32	10000	5981	2039	1981	15982	45985	7	1098	[(5929:5),(6199:2)]
33	10000	6023	1957	2021	16024	46027	12	1127	[(3:12)]
34	10000	6015	2021	1965	16016	46019	60	1247	[(5641:17),(5858:3),(5969:16),(5997:6),(6025:6),(6033:3), (6133:2),(9790:2),(11731:5)]
35	10000	5953	1975	2073	15954	45957	24	1071	[(2:24)]
36	10000	6030	2034	1937	16031	46034	36	1279	[(12930:22),(15804:2),(15947:12)]
37	10000	6008	1996	1997	16009	46012	3	1092	[(13780:3)]
38	10000	6054	1963	1984	16055	46058	1	1111	[(3447:1)]
39	10000	6015	1977	2009	16016	46019	27	1194	[(2391:27)]
40	10000	6042	1967	1992	16043	46046	73	2607	[(2820:1),(7720:3),(8418:14),(8586:2),(10093:46),(12532:7)]

TABLE II
BENCHMARK DESCRIPTION - CASES 1 TO 40

id	gNodes	gAT	gAND	gOR	tsVars	tsClauses	cost	time	solution
41	15000	12090	1443	1468	27091	72094	16	3024	[(97:8),(6443:8)]
42	15000	11973	1517	1511	26974	71977	3	2139	[(14702:3)]
43	15000	12004	1529	1468	27005	72008	2	1511	[(1021:1),(22132:1)]
44	15000	11969	1517	1515	26970	71973	18	10965	[(2:18)]
45	15000	12123	1457	1421	27124	72127	5	1576	[(2350:1),(2665:1),(3626:3)]
46	15000	11969	1480	1552	26970	71973	5	1564	[(13:4),(25198:1)]
47	15000	11949	1490	1562	26950	71953	9	1605	[(58:1),(3272:6),(6793:2)]
48	15000	11982	1542	1477	26983	71986	18	2863	[(5:18)]
49	15000	12015	1486	1500	27016	72019	3	1511	[(2649:1),(6731:2)]
50	15000	11980	1496	1525	26981	71984	1	1627	[(1853:1)]
51	15000	9043	2893	3065	24044	69047	11	1423	[(22630:2),(22728:9)]
52	15000	9036	2976	2989	24037	69040	123	9571	[(10932:44),(13135:5),(15118:5),(15681:3),(15695:66)]
53	15000	9046	2938	3017	24047	69050	14	1182	[(3:6),(14013:8)]
54	15000	9015	3028	2958	24016	69019	47	1502	[(3:42),(11757:2),(13221:3)]
55	15000	8987	3035	2979	23988	68991	8	1208	[(17563:3),(17567:5)]
56	15000	9015	3002	2984	24016	69019	1	1483	[(15321:1)]
57	15000	9099	2966	2936	24100	69103	80	5040	[(3352:8),(3770:4),(8682:5),(10152:8),(10159:3),(10239:1), (10468:3), (18507:6),(18555:8),(18573:2),(18628:8),(18753:14), (19274:5),(22843:2),(23171:3)]
58	15000	9011	2943	3047	24012	69015	4	1152	[(18688:2),(19011:2)]
59	15000	9035	2998	2968	24036	69039	53	10802	[(2:53)]
60	15000	9013	3066	2922	24014	69017	14	1264	[(11895:1),(12533:3),(12590:3),(16787:2),(23583:5)]
61	20000	16004	1975	2022	36005	96008	1	1637	[(16475:1)]
62	20000	15977	1958	2066	35978	95981	3	1686	[(34706:3)]
63	20000	16007	2071	1923	36008	96011	1	2040	[(32144:1)]
64	20000	16067	1978	1956	36068	96071	1	1572	[(35189:1)]
65	20000	15999	1990	2012	36000	96003	2	1873	[(12825:2)]
66	20000	15947	2037	2017	35948	95951	4	2074	[(3:4)]
67	20000	16019	1992	1990	36020	96023	7	2153	[(6:3),(20228:4)]
68	20000	15976	2036	1989	35977	95980	5	1492	[(7:2),(17657:2),(24220:1)]
69	20000	16019	1997	1985	36020	96023	9	2545	[(3:9)]
70	20000	16052	1909	2040	36053	96056	2	2031	[(29145:2)]
71	20000	12037	4033	3931	32038	92041	2	1847	[(21278:2)]
72	20000	11977	3997	4027	31978	91981	373	12887	[(1172:17),(11919:5),(12728:2),(1295:1),(13840:2),(14081:4), (14453:22), (15461:5),(16291:13),(17369:3),(18459:20),(19920:1), (19925:4),(20651:2), (20789:37),(20929:9),(21080:41),(22175:7), (22642:1),(22804:27),(22806:12),(22809:3),(22818:26),(22829:6), (22852:31),(23065:7),(28543:65)]
73	20000	12037	3999	3965	32038	92041	3	1513	[(22359:2),(30489:1)]
74	20000	12004	3980	4017	32005	92008	1	1434	[(2:1)]
75	20000	12059	3930	4012	32060	92063	28	3071	[(2:18),(26845:10)]
76	20000	12094	3858	4049	32095	92098	2	1772	[(6431:2)]
77	20000	12000	4014	3987	32001	92004	3	1474	[(27418:3)]
78	20000	12110	3940	3951	32111	92114	4	1665	[(4:4)]
79	20000	12036	4012	3953	32037	92040	71	1600	[(2:9),(14499:3),(17840:14),(19998:32),(28910:2), (29045:9),(29937:2)]
80	20000	12035	4055	3911	32036	92039	8	1688	[(13697:8)]

TABLE III
BENCHMARK DESCRIPTION - CASES 41 TO 80

REFERENCES

[1] M. Barrère, C. Hankin, N. Nicolaou, D. Eliades, and T. Parisini, "Identifying Security-Critical Cyber-Physical Components in Industrial Control Systems," https://arxiv.org/abs/1905.04796, May 2019.

[2] Y. Desmedt and Y. Wang, "Maximum Flows and Critical Vertices in AND/OR Graphs," in *Computing and Combinatorics*, O. H. Ibarra and L. Zhang, Eds. Springer Berlin Heidelberg, 2002, pp. 238–248.

[3] ——, "Analyzing Vulnerabilities Of Critical Infrastructures Using Flows And Critical Vertices In And/Or Graphs," *Int. J. Found. Comput. Sci.*, vol. 15, no. 1, pp. 107–125, 2004.

[4] G. Jakimoski and M. Burmester, "Using Faulty Flows in AND/OR Graphs to Model Survivability and Reliability in Distr. Systems," 2004.

[5] U. dos Santos Souza, F. Protti, and M. D. da Silva, "Revisiting the complexity of and/or graph solution," *Journal of Computer and System Sciences*, vol. 79, no. 7, pp. 1156 – 1163, 2013.

[6] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, ser. RAND Corporation research study. University Press, 1962.

[7] 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.

[8] J. Davies and F. Bacchus, "Solving MAXSAT by Solving a Sequence of Simpler SAT Instances," in *Principles and Practice of Constraint Programming – CP 2011*, J. Lee, Ed. Springer, 2011, pp. 225–239.

[9] M. Barrère, "META4ICS - Metric Analyser for Industrial Control Systems," https://github.com/mbarrere/meta4ics, May 2019.

[10] "SAT4J," http://www.sat4j.org/, Cited June 2019.