Generating Minimal Unsatisfiable SAT Instances from Strong Digraphs

Gábor Kusper
Eszterházy Károly University
Email: kusper.gabor@uni-eszterhazy.hu

Tamás Balla
Eszterházy Károly University
Email: balla.tamas@uni-eszterhazy.hu

Csaba Biró
Eszterházy Károly University
Email: biro.csabar@uni-eszterhazy.hu

Tibor Tajti
Eszterházy Károly University
Email: tajti.tibor@uni-eszterhazy.hu

Zijian Győző Yang
Eszterházy Károly University
Email: yang.zijian.gyozo@uni-eszterhazy.hu

Imre Baják
Eszterházy Károly University
Email: bajak.imre@uni-eszterhazy.hu

***Abstract*—We present a model generator which generates SAT problems from digraphs. There are a few restrictions on the input digraphs. There must be no self-loops, and its vertices must be Boolean variables or labeled by distinct Boolean variables. We call such digraphs communication graphs. The model is pretty straightforward: if the communication graph contains the edges (a, b) and (a, c), and there is no other edge from a, then this is encoded by the clause: $\{\neg a, b, c\}$. The intuition is that a can send a message to b or c. We have to represent all cycles as well. If (a, b, c, a) is a cycle with the set of exit points $\{d, e\}$ in the input communication graph, then it is encoded by the clause: $\{\neg a, \neg b, \neg c, d, e\}$. The intuition is the following: if there is a message in the cycle, then it has to leave the cycle and we have to be sent to d or e. We call this model as the weak model of communication graphs. We show that the weak model is a Black-and-White SAT problem if and only if the input is a strongly connected communication graph. We prove also that all clauses in such models are independent. From this we obtain that a weak model generated from a strong digraph is a minimal unsatisfiable SAT instance if we add to it the black and the white clauses, which are the only solutions of a Black-and-White SAT problems. Minimal unsatisfiable SAT instances are one of the hardest unsatisfiable clause sets, so they are interesting from the viewpoint of testing SAT solvers. There are some techniques which generates special minimal unsatisfiable SAT instances from digraphs, see the work of H. Abbasizanjani, and O. Kullmann, but there was no general solution before our work. Although, our solution is general one, but the generation of weak models is difficult because we have to find all cycles, including non-simple cycles. Therefore, we discuss how to create models of digraphs without cycle detection. Finally, we present some test results using state-of-the-art SAT solvers. It seems that these minimal unsatisfiable SAT instances are very difficult for them even with 20 variables.**

I. Introduction

Propositional satisfiability is the problem of determining, for a formula of the propositional logic, if there is an assignment of truth values to its variables for which that formula evaluates to true. By SAT we mean the problem of propositional satisfiability for formulas in conjunctive normal form (CNF). SAT is one of the most-researched NP-complete problems [10] in several fields of computer science, including theoretical computer science, artificial intelligence, hardware design, and formal verification [6].

A SAT problem may contain redundant clauses. For example blocked clauses are redundant because if we delete a blocked clause from a clause set then its satisfiabilty is not changed [17], [18]. If we delete all redundant clause from an unsatisfiable clause set then we get its minimal unsatisfiable core, which is a minimal unsatisfiable SAT problem, i.e., any proper subset of it is satisfiable.

On the one hand, such problems are interesting because they reflect the structure which makes the problem unsatisfiable. On the other hand, such problems are usually difficult, so they are useful to test SAT solvers.

The notion of minimal unsatisfiable SAT problem, and the notion of minimal unsatisfiability core of a SAT problem is discussed in detailed for example by Davydov et al. in [11].

In our previous papers, we have presented some links between the fields of SAT problems and digraphs. Our motivation was twofold. First of all, we studied Wireless Sensor Notworks (WSNs), and we wanted to construct a model where we can use our knowledge from the field of logic. Furthermore, our intuition is the following. It worth to represent SAT problems as digraphs, because if we create a CNF formula out of formula, then the original structures of the original formula are blurred, but in a digraph it is easy to see structures.

We have studied WSNs and we have created the so called communication graph, which is a restricted digraph: it may not contain self loops, and its vertices must be Boolean variables or labeled by distinct Boolean variables. We have introduced several logic based models (strong-, weak-, and Balatonboglár models, see below) to represent a communication graph. Our biggest success was that we were able to show that in these models the strongly connected communication graphs are represented by Black-and-White SAT problems. A SAT problem is a Black-and-White SAT problem if and only if it has only two solutions: the black assignment, where each variable is false, and the white assignment, where each variable is true [8]. So, our motivation to study the Black-and-White SAT problem is the fact that this notion is a strong link between two prominent fields, SAT problem and digraphs, because, strong digraphs are represented by Black-and-White SAT problems.

In a communication graph we can use the intuitive notion of sending a message. If we have an edge from node A to node B in a communication graph, then we might say, that

node A can send a message to node B. This intuition helps to understand our models.

Our first result was that we proved that the strong model of a strong digraph is a Black-and-White SAT problem, and vice versa, if the strong model of a digraph is a Black-and-White SAT problem, then it is a strong digraph, see [8].

In logic the most natural representation of an edge of a digraph, say $a \to b$, is to use implication, i.e., $a \implies b$. This means that the edge $a \to b$ can be represented by the binary clause: $(\neg a \vee b)$. If a digraph contains two edges: $a \to b$, and $a \to c$, then those can be represented by the formula: $(a \implies b) \wedge (a \implies c)$, which is equivalent to two 2-clauses $(\neg a \vee b) \wedge (\neg a \vee c)$. We used this representation in paper [8]. The original name of this representation was 'the 2-SAT representation of directed graphs', but we renamed it to be the strong model in [15].

This representation helps us to translate a digraph into a 2-SAT problem. We can also translate a Black-and-White 2-SAT problem into a digraph. We can also translate any other 2-SAT problem into a directed graph, if it does not subsume neither the black nor the white clause, i.e., it does not contain a clause which contains only positive or negative literals, like $(a \vee b)$ or $(\neg a \vee \neg b)$

Actually, there is no point in translating a 2-SAT problem into a digraph, because the 2-SAT problem is solvable in linear time [3].

Preferably, we should translate 3-SAT problems into digraphs, because then we could use graph tools to study them, like Depth-first search, which is a linear algorithm in the size of the graph [25]. Or we could translate it back to 2-SAT by using our strong model.

Our second paper aims to link the field of digraphs and the field of 3-SAT problems [15]. In the second paper the idea is the following: If a digraph contains two edges: $a \to b$, and $a \to c$, then those can be represented by the formula: $(a \implies b) \vee (a \implies c)$, which is equivalent to $a \implies (b \vee c)$, which is equivalent to a 3-clause $(\neg a \vee b \vee c)$.

This idea is not enough to be able to generate a Black-and-White SAT formula from a strong digraph, although, our intuition was that in any SAT based model the representation of a strong digraph must be a Black-and-White SAT formula. So we need to represent cycles of the graph, too. If $a_1 \to a_2 \to \cdots \to a_n \to a_1$ is a cycle with exit points $b_1, b_2, \ldots, b_m$, then this cycle can be represented by the clause: $(\neg a_1 \vee \neg a_2 \vee \cdots \vee \neg a_n \vee b_1 \vee b_2 \vee \cdots \vee b_m)$.

We showed that the weak model has the same interesting property as the strong model, i.e., the model of a strong digraph is a Black-and-White SAT problem. This result is submitted to ToC [15]. We are waiting for the reviews. Since this paper is not accessible, we recall the most relevant parts of it.

We proved a very strong theorem, that any model between the strong and weak model has the property that the generated SAT problem is a Black-and-White SAT problem if and only if the represented graph is a strong digraph [15].

We have found two models between the strong and the weak one. These are the Balatonboglár model [15], and the simplified Balatonboglár model [20]. In this work we do not recall the Balatonboglár model. We mention it in order to discuss what is our motivation.

Then we have found that the Balatonboglár model is redundant [19]. Our motivation is to find a less redundant model. It turned out that what we try to find, has been already constructed. As we will see, in the weak model there is no redundancy. This is our main contribution in this work.

Then we show some corollaries. One of these is the following: a weak model generated from a strong digraph is a minimal unsatisfiable SAT instance if we add to it the black and the white clauses, which are the only solutions of a Black-and-White SAT problem.

Finally, we present some test results using state-of-the-art SAT solvers. It seems that these minimal unsatisfiable SAT instances are very difficult for them even with 20 variables.

II. Related works

These days one of the most promising branch of mathematics is the idea that we try to unify different theories, like in case of Langlands program [21], which relates algebraic number theory to automorphic forms and representation theory. Another famous example is the modularity theorem [26] (formerly called the Taniyama–Shimura–Weil conjecture), which states that elliptic curves over the field of rational numbers are related to modular forms. Without the modularity theorem Andrew Wiles could not have proved Fermat's Last Theorem [27].

We study how to unify the fields of digraphs and propositional logic formulae. The most prominent related works are:

- An implication graph [3] is a skew-symmetric directed graph, where vertices are literals (Boolean variables, and their negation), edges represent implication. Note that the binary clause $x \vee y$ is represented by two implications in the implication graph: $\neg x \implies y$, and $\neg y \implies x$, and so the implication graph is skew-symmetric, i.e., it is isomorphic to its own transpose graph.
- AIG, And-Inverter Graph [14] is a directed acyclic graph where vertices are logical conjunctions with two input edges, a marked edge means logical negation, the Boolean variables are the input, the formula itself is the output.
- BDD, Reduced Ordered Binary Decision Diagram [9], which is a rooted, directed, acyclic graph, which consists of vertices, which are Boolean variables, and terminal vertices, called 0-terminals, which terminate paths, where the formula evaluates to false; and 1-terminals, which terminate paths, where the formula evaluates to true. Each non-terminal vertex has two child vertices called a low child, the corresponding edge is called a 0-edge; and a high child, the corresponding edge is called a 1-edge; which are possible values of the parent vertex. One has to merge any isomorphic subgraphs and eliminate any vertex whose two children are isomorphic.
- ZDD (also called ZBDD in the literature), Zero-Suppressed Binary Decision Diagram [22], is a kind

of binary decision diagram, where instead of the rule "eliminate any vertex whose two children are isomorphic" we use the rule "eliminate those vertices whose 1-edge points directly to 0-terminal". If a SAT problem has only a few solutions then ZDD is a better representation than BDD.

- The graph representation of logical games is a well-known connection between graphs and logical formulae. Some examples are [13], [23].

As we can see a great effort has been made in the direction from formulae to graphs. Here, we study the other way, the direction from graphs to formulae.

A very interesting work is the article: Minimal Unsatisfiability and Minimal Strongly Connected Digraphs by H. Abbasizanjani, and O. Kullmann [1]. They are interested in generating minimal unsatisfiable SAT instances with $n + 2 = m$, where n is the number of variables, and m is the number of clauses. They use several techniques. One of these is a technique to turn a minimal strongly connected digraph into a minimal unsatisfiable SAT instance. They use a similar notion as our strong model, and they add the black and white clauses to the instance at the end of the conversation like we do. They can do so, because our strong model and our weak model results in the same model if the input is a minimal strongly connected digraph, because such a graph is a cycle, which contains all the nodes: so each node has only one outgoing edge, and there is no other cycle except the one which contains all the nodes.

It seems that they do not know about our results. Some of their concepts are the same as our ones. What we call black and white clauses, they call monotone clauses. They use a similar concept to our strong model, but only for cycles. They use the name $\mathcal{F}n$. It seems that they do not use any similar notion to our weak model.

III. Definitions

A *literal* is a Boolean variable, called positive literal, or the negation of a Boolean variable, called negative literal. Examples for literals are: $a, \neg a, b, \neg b, \ldots$.

A *clause* is a set of literals. A *clause set* is a set of clauses. A *SAT problem* is a clause set. An *assignment* is a set of literals. In a clause or in an assignment, a variable may occur either as a positive literal or as a negative literal, but not as both, or it may not occur at all.

Clauses are interpreted as disjunctions of their literals. Assignments are interpreted as conjunctions of their literals. Clause sets are interpreted as conjunctions of their clauses.

If a clause or an assignment contains exactly k literals, then we say it is a *k-clause* or a *k-assignment*, respectively. A 1-clause is called a *unit*, a 2-clause is called a *binary clause*. A *k-SAT problem* is a clause set where its clauses have at most k literals. A clause from a clause set is a *full-length clause* iff it contains all variables from the clause set.

We define some auxiliary functions. If C is a clause, then let $V(C)$ be the set of variables which occur in C, let $N(C)$ be the set of negative literals from C, and let $P(C)$ be the set of positive literals from C. We have that $C = N(C) \cup P(C)$, $V(N(C)) \cap V(P(C)) = \emptyset$, and $P(C) = V(P(C))$.

We use two intuitive notions: NNP clause, and NPP clause. A clause is an NNP clause iff it contains exactly one positive literal. A clause is an NPP clause iff it contains exactly one negative literal.

If a is a literal in clause set S, and $\neg a$ is not a literal in S, then we say that a is a pure literal in S.

The negation of a set H is denoted by $\neg H$ which means that all elements in H are negated. Note that $\neg\neg H = H$.

Let V be the set of variables of a clause set. We say that WW is the *white clause* or the *white assignment* on variables V iff $WW = V$. We say that BB is the *black clause* or the *black assignment* on variables V iff $BB = \neg V$. For example if $V = \{a, b, c\}$, then $WW = \{a, b, c\}$, and $BB = \{\neg a, \neg b, \neg c\}$.

We say that clause C *subsumes* clause D iff C is a subset of D.

We say that clause set S *subsumes* clause C iff there is a clause in S which subsumes C. Formally: $S\ subsumes\ C \iff \exists D(D \in S \wedge D \subseteq C)$.

We say that clause C is *entailed* by clause set S iff all full-length clauses which are subsumed by C are also subsumed by S. The logical interpretation of this notion is the following: C is entailed by S iff C is a logical consequence of S. Subsumed clauses are also entailed.

We say that clause C is *independent* in clause set S iff C is not entailed by $S \setminus \{C\}$. A full-length clause is independent in a clause set iff it is not subsumed.

We say that assignment M is a *solution* for clause set S iff for all $C \in S$ we have $M \cap C \neq \{\ \}$.

We say that the clause set S is a *Black-and-White SAT problem* iff it has only two solutions, the white assignment (WW) and the black one (BB).

We say that clause sets A and B are equivalent iff they have the same set of solutions. We say that clause set A entails clause set B iff the set of solutions of A is a subset of the set of solutions of B, i.e., A may have no other solutions than B. This notion is denoted by $A \geq B$. Note that if A subsumes all clauses of B, then $A \geq B$.

We say that A is stronger than B iff $A \geq B$ and A and B are not equivalent. This notion is denoted by $A > B$.

The construction $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ is a digraph, where $\mathcal{V}$ is the set of vertices, and $\mathcal{E}$ is the set of edges. An edge is an ordered pair of vertices. The edge (a, b) is depicted by $a \to b$, and we can say that a has a child b. If (a, b) is an element of $\mathcal{E}$, then we may say that (a, b) is an edge of $\mathcal{D}$.

We say that $\mathcal{D}$ is a communication graph iff for all a in $\mathcal{V}$ have that (a, a) is not in $\mathcal{E}$, and if x is an element of $\mathcal{V}$ then $\neg x$ must not be an element of $\mathcal{V}$. We need this constraint because we generate a logical formula out of $\mathcal{D}$. If we speak about a communication graph then we may use the word node as a synonym of vertex.

A path from a_1 to a_j in digraph $\mathcal{D}$ is a sequence of vertices $a_1, a_2, \ldots, a_j$ such that for each $i \in \{1, \ldots, j-1\}$ we have that

(a_i, a_{i+1}) is an edge of $\mathcal{D}$. A path from a_1 to a_j in digraph $\mathcal{D}$ is a *cycle* iff (a_j, a_1) is an edge of $\mathcal{D}$. The cycle $a_1, a_2, \ldots, a_j, a_1$ is represented by the following tuple: $(a_1, a_2, \ldots, a_j)$. This tuple can be used as a set of its elements. Note that in this representation of a cycle the first and the last element must not be the same vertex.

If we have a cycle $(a_1, a_2, \ldots, a_n)$ then b is an exit point of it iff for some $j \in \{1, 2, \ldots, m\}$ we have that (a_j, b) is an edge and $b \notin \{a_1, a_2, \ldots, a_n\}$.

A digraph is complete iff every pair of distinct vertices is connected by a pair of unique edges (one in each direction). A digraph is strongly connected or is a strong digraph iff there is a path from each vertex to each other vertex. Note that a complete digraph is also a strong one. Note that a strong digraph contains a cycle which contains all vertices.

IV. The Weak Model of Communication Graphs

In this section we define the weak model of communication graphs.

Let us assume that $\mathcal{D} = (\{a, b, c\}, \{(a, b), (a, c)\})$, as in the previous section. The weak model of $\mathcal{D}$ is the formula: $(a \implies b) \vee (a \implies c)$, which means that if a node can send a message to more than one nodes, then it sends only to one of them, and if it gets back the message, then it sends to the next one. Note that $(a \implies b) \vee (a \implies c)$ is equivalent to $(\neg a \vee b \vee c)$.

The only problem with this representation is that the message can be trapped if a graph contains cycles. Let us assume that we have a cycle with two nodes, n_1 and n_2. Then n_1 may send the message to n_2, and n_2 to n_1, and so on, which means that other nodes could never get the message. This is not good, since our goal is to send a message to each node, if it is possible.

Let us add a new edge to the graph $(\{a, b, c\}, \{(a, b), (a, c)\})$. The new edge should go from b to a, so the new graph is this: $(\{a, b, c\}, \{(a, b), (a, c), (b, a)\})$. Now we have a cycle with the nodes a and b. Now we have to add a clause to our model which ensures that if b sends a message to a, and a can send a message to b or c, then a should not send back the message to b, but it has to send it to c, which can be formalized like this: $((b \implies a) \wedge (a \implies (b \vee c))) \implies (a \implies c)$. Note that this is equivalent to the clause: $(\neg a \vee \neg b \vee c)$.

In a more general way, node a which has outgoing edges to nodes $b_1, b_2, \ldots, b_k$ is represented by the clause: $(\neg a \vee b_1 \vee b_2 \vee \cdots \vee b_k)$; and the cycle $a_1, a_2 \ldots a_n, a_1$ with exit points $b_1, b_2, \ldots, b_m$ is represented by the clause: $(\neg a_1 \vee \neg a_2 \vee \cdots \vee \neg a_n \vee b_1 \vee b_2 \vee \cdots \vee b_m)$. We call this model as the weak model of communication graphs.

We define it also formally: Let $\mathcal{D}$ be a communication graph. Let $\mathcal{V}$ be the set of vertices of $\mathcal{D}$ and $\mathcal{E}$ the set of edges of $\mathcal{D}$. Since $\mathcal{D}$ is a communication graph, we know that elements of $\mathcal{V}$ can be used as positive literals. Then we define the following notions:

$OutE(a) := \{b \mid (a, b) \in \mathcal{E}\}$.

$NodeRep(a) := \{\neg a\} \cup OutE(a)$.

$NodeRep := \{NodeRep(a) \mid a \in \mathcal{V} \wedge OutE(a) \neq \emptyset\}$.

$Cycles := \{(a_1, a_2, \ldots, a_k) \mid k = 1 \vee \forall i(i = 1 \ldots k \implies a_{(i \bmod k)+1} \in OutE(a_i))\}$, and for any two elements of $Cycles$ they cannot be equal as a set.

$ExitPoints(\ (a_1, a_2, \ldots a_k)\ ) := \{b \mid \exists i(i = 1 \ldots k \wedge b \in OutE(a_i)) \wedge \neg\exists j(j = 1 \ldots k \wedge b = a_j)\}$.

$CycleRep := \{\neg C \cup ExitPoints(C) \mid C \in Cycles \wedge ExitPoints(C) \neq \emptyset\}$.

$\mathcal{WM} := NodeRep \cup CycleRep$.

$\mathcal{WM}$ is the weak model of $\mathcal{D}$.

Note that $NodeRep$ is a subset of $CycleRep$, because we take each node itself as a cycle, see the constraint $k = 1$ in the definition of $Cycles$ which is connected by a disjunction to the rest of the constraint. So alternatively we can define $\mathcal{WM}$ as follows: $\mathcal{WM} := CycleRep$. We do not use this observation in this work, although, some proofs would be shorter.

Please note that each clause in $\mathcal{WM}$ contains at least one positive and one negative literal, because a node is only represented if it has an outgoing edge, and a cycle is only represented if it has an exit point, therefore, $\mathcal{WM}$ is satisfiable for any communication graph $\mathcal{D}$.

Let us see some examples. It is not easy to check that this SAT problem is indeed a Black-and-White SAT problem. Therefore, first we show the set of all full-length clauses on the variables $\{a, b, c, d\}$ with an index number (each entry has the form: index number: full-length clause):

$$\begin{aligned}\{0 : \{\neg a, \neg b, \neg c, \neg d\},\ 1 : \{\neg a, \neg b, \neg c, d\},\\ 2 : \{\neg a, \neg b, c, \neg d\},\ 3 : \{\neg a, \neg b, c, d\},\\ 4 : \{\neg a, b, \neg c, \neg d\},\ 5 : \{\neg a, b, \neg c, d\},\\ 6 : \{\neg a, b, c, \neg d\},\ 7 : \{\neg a, b, c, d\},\\ 8 : \{a, \neg b, \neg c, \neg d\},\ 9 : \{a, \neg b, \neg c, d\},\\ 10 : \{a, \neg b, c, \neg d\},\ 11 : \{a, \neg b, c, d\},\\ 12 : \{a, b, \neg c, \neg d\},\ 13 : \{a, b, \neg c, d\},\\ 14 : \{a, b, c, \neg d\},\ 15 : \{a, b, c, d\}\},\end{aligned} \tag{1}$$

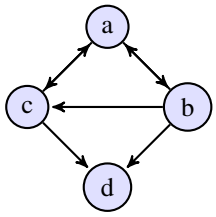

Fig. 1. A communication graph with 4 vertices, 3 cycles.

Figure 1 shows a communication graph with 3 cycles:

$$(a, b), (a, b, c), (a, c).$$

So its weak model is (after each clause we list the indices of subsumed full-length clauses from (1)):

$$\begin{aligned}\mathcal{WM} = \{\{\neg a, b, c\} : 6, 7,\ \{\neg b, a, c, d\} : 11,\\ \{\neg c, a, d\} : 9, 14,\ \{\neg a, \neg b, c, d\} : 3,\\ \{\neg a, \neg b, \neg c, d\} : 1,\ \{\neg a, \neg c, b, d\} : 5\}.\end{aligned} \tag{2}$$

Note that since d has no child node, it does not occur as a negative literal in the model.

Note that since the communication graph in Figure 1 is not strongly connected, its weak model (2) is not a Black-and-White SAT problem. For example $\{\neg a, \neg b, \neg c, d\}$ is a solution of the SAT problem in (2),

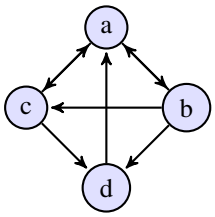

Fig. 2. A strongly connected communication graph with 4 vertices, 6 cycles.

As the second example we show the weak model of the communication graph in Figure 2. In that graph we have 6 cycles:

$$(a, b), (a, b, c), (a, c), (a, b, d), (a, c, d), (a, b, c, d).$$

The last cycle contains all vertices, so it has no exit point, therefore no clause is generated for it. Note that in that graph we have an edge from d to a. The corresponding clauses are the last 3 clauses in this example, the last 2 clauses corresponds to the new cycles. The rest of the clauses are the same as in the previous model. So the weak model is (after each clause we list the indices of subsumed full-length clauses from (1)):

$$\begin{aligned}\mathcal{WM} = \{&\{\neg a, b, c\} : 6, 7,\ \{\neg b, a, c, d\} : 11,\\ &\{\neg c, a, d\} : 9, 14,\ \{\neg a, \neg b, c, d\} : 3,\\ &\{\neg a, \neg b, \neg c, d\} : 1,\ \{\neg a, \neg c, b, d\} : 5,\\ &\{\neg d, a\} : 8, 10, 12, 14,\ \{\neg a, \neg b, \neg d, c\} : 2,\\ &\{\neg a, \neg c, \neg d, b\} : 4\}.\end{aligned} \tag{3}$$

Note that since the communication graph in Figure 2 is strongly connected, its weak model is a Black-and-White SAT problem, i.e., the SAT problem in (3) has only these two solutions: $\{a, b, c, d\}$, and $\{\neg a, \neg b, \neg c, \neg d\}$.

Each cycle was a simple cycle in the above 2 examples. A cycle is simple iff only the first and last vertices are repeated, so we might think that it is enough to consider only simple cycles. The next example shows a case when we need to consider also a non-simple cycle.

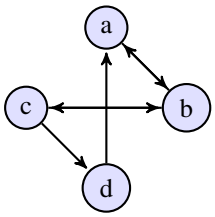

Fig. 3. A strongly connected communication graph with 4 vertices, 3 simple cycles, 1 non-simple cycle.

In Figure 3 we can see a strongly connected communication graph which contains 3 simple cycles: $(a, b), (a, b, c, d), (b, c)$, and 1 non-simple cycle: (a, b, c, b). As the third example we give the weak model of this communication graph (after each clause we list the indices of subsumed full-length clauses from (1)):

$$\begin{aligned}\mathcal{WM} = \{&\{\neg a, b\} : 4, 5, 6, 7,\ \{\neg b, a, c\} : 10, 11,\\ &\{\neg c, b, d\} : 5, 13,\ \{\neg d, a\} : 8, 10, 12, 14,\\ &\{\neg a, \neg b, c\} : 2, 3,\ \{\neg b, \neg c, a, d\} : 9,\\ &\{\neg a, \neg b, \neg c, d\} : 1\}.\end{aligned} \tag{4}$$

Note that the last clause is generated from the non-simple cycle, and the corresponding full-length clause is not subsumed by any other clause.

V. Theoretical results

We prove that if we use the weak model then the model of a communication graph is a Black-and-White SAT problem iff the graph is strongly connected. To be able to prove this we need some auxiliary lemmas. We recall some theoretical results from [15]. We do not give the proof of these lemmas and theorems, we give only the outline of their proof.

Lemma 1. *Let F be a Black-and-White SAT problem. Then $F \cup \{BB, WW\}$ is unsatisfiable.*

Proof: From the definition of the Black-and-White SAT problem we know that F has only two solutions: BB and WW. Since $\neg BB = WW$, and $\neg WW = BB$ we have that $F \cup \{BB, WW\}$ is unsatisfiable. ■

The next lemma states that the weak model has at least two solutions, the white assignment and the black one.

Lemma 2. *Let $\mathcal{D}$ be a communication graph. Let $\mathcal{WM}$ be the weak model of $\mathcal{D}$. Then $\mathcal{WM}$ has at least two solutions, namely the white assignment (WW) and the black assignment (BB).*

Proof: Let $\mathcal{D}$ be a communication graph. Let $\mathcal{WM}$ be the weak model of $\mathcal{D}$. Since there is a positive and also a negative literal in each clause of $\mathcal{WM}$, the white assignment (WW) and also the black assignment (BB) are solutions for $\mathcal{WM}$. ■

The next lemma states that if the weak model has only two solutions, then each cycle in the communication graph has at least one exit point, except the cycle which contains all vertices of the graph.

Lemma 3. *Let $\mathcal{D}$ be a communication graph. Let $\mathcal{WM}$ be the weak model of $\mathcal{D}$. Assume that $\mathcal{WM}$ has only two solutions. Then for each cycle $C \in Cycles$ we have that $ExitPoints(C)$ is not empty or C contains all vertices of $\mathcal{D}$.*

The proof of this lemma is submitted to ToC as [15]. The outline of the proof is the following. Assume that our goal is not true, i.e., there is a cycle C, which has no exit point and does not contain all vertices of $\mathcal{D}$. Let us construct the following assignment: $A = C \cup \neg(\mathcal{V} \setminus C)$, where $\mathcal{V}$ is the set of vertices of $\mathcal{D}$. Since C has no exit point there is no corresponding clause generated by $CycleRep$ which would falsify A. Since $A \neq WW$ and $A \neq BB$ this is a contradiction, because our assumptions and Lemma 2 makes sure that $\mathcal{WM}$ has only two solutions: WW and BB.

The next lemma states that a Black-and-White SAT problem may not contain pure literals.

Lemma 4. *If $\mathcal{S}$ is a Black-and-White SAT problem, then it contains no pure literal.*

The proof of this lemma is submitted to ToC as [15]. The outline of the proof is the following. Any pure literal can be found in all the solutions. But S has only two solutions, WW and BB, but they have no common literal, so S may not contain any pure literals.

The next lemma states that if we have a complete graph, then its weak model is a Black-and-White SAT problem. This result is somehow natural, since the weak model of a complete graph is the biggest possible that we can generate and the weak model always has at least two solutions.

Lemma 5. *If $\mathcal{D}$ is a complete communication graph, and $\mathcal{WM}$ is the weak model of $\mathcal{D}$, then $\mathcal{WM}$ is a Black-and-White SAT problem.*

The proof of this lemma is submitted to ToC as [15]. The outline of the proof is the following. Because $\mathcal{D}$ contains all possible cycles, and for each cycle we have that their exit points are the rest of the graph, and because of the construction of $CycleRep$ we have that $\mathcal{WM}$ contains all full-length clauses with at least one positive and one negative literal. So $\mathcal{WM}$ has only two solutions: WW and BB.

The next theorem states that the weak model of a strongly connected graph is a Black-and-White SAT problem.

Theorem 1. *Let $\mathcal{D}$ be a communication graph. Let $\mathcal{WM}$ be the weak model of $\mathcal{D}$. Then $\mathcal{WM}$ is a Black-and-White SAT problem iff $\mathcal{D}$ is strongly connected.*

Since this proof is submitted to ToC as [15], we give here only the outline of the proof.

From left to right we prove the theorem in a constructive way: Let us have a sequence of vertices which can be built by Lemma 4 and by the construction of $NodeRep$. Eventually, we will find a cycle at the end of this sequence. Lemma 3 ensures that there is an exit point from this cycle, which either results in a bigger cycle, or in a longer sequence. Using these two steps eventually we construct a strongly connected component. From Lemma 3 it follows that it is also maximal (otherwise there would be an exit point from it which would allow to do the above steps), i.e., the graph is strongly connected.

From right to left we use induction: We start the proof from a complete graph. We know that complete graphs are also strongly connected. As the induction step, we drop an edge from this graph in a way that it remains strongly connected. We assume that before the drop the weak model was a Black-and-White SAT problem, which is true by Lemma 5. We show that it remains a Black-and-White SAT problem also after the drop. To show this, we show that each clause in the model after the drop is a subset of some clause from the model before the drop. So the new model may not have more solutions then the old one. But the new model has to have at least two solutions by Lemma 2. So the new model is a Black-and-White SAT problem. Since our graph can be constructed by dropping edges from a complete graph, which has the same set of vertices, its weak model is also a Black-and-White SAT problem.

After recalling some results form [15], let us present some new ones.

The next lemma is interesting, but an idea in its proof is even more interesting.

Lemma 6. *Let $\mathcal{D}$ be a communication graph. Let $\mathcal{WM}$ be the weak model of $\mathcal{D}$. Then for all C in $\mathcal{WM}$ we have that C is not subsumed by $\mathcal{WM} \setminus \{C\}$.*

Proof: Assume $\mathcal{D}$ is a communication graph. Assume $\mathcal{WM}$ is the weak model of $\mathcal{D}$. We show that for all C in $\mathcal{WM}$ we have that C is not subsumed in $\mathcal{WM} \setminus \{C\}$. To show this let C' be an arbitrary but fixed clause from $\mathcal{WM}$, we show that C' is not subsumed in $\mathcal{WM} \setminus \{C'\}$, which means that for all B in $\mathcal{WM}$, such that $B \neq C'$ we have that $B \not\subseteq C'$. Let B' be an arbitrary but fixed clause from $\mathcal{WM} \setminus \{C'\}$, we show that $B' \not\subseteq C'$. We know that $C' = N(C') \cup P(C')$, and $B' = N(B') \cup P(B')$. Because of the construction of $\mathcal{WM}$ neither of these sets are empty. There are two cases: (a) $N(B') \not\subseteq N(C')$ or $P(B') \not\subseteq P(C')$, (b) $N(B') \subseteq N(C')$ and $P(B') \subseteq P(C')$. In case (a) we trivially have that $B' \not\subseteq C'$. In case (b) we have that $V(N(B'))$ represents a cycle, and $V(N(C'))$ also represents a cycle. From the construction of $\mathcal{WM}$ we know that $N(B') \neq N(C')$, because each cycle is represented only by one clause and $B' \neq C'$. Therefore, we have that $N(B') \subset N(C')$. From this, and from $P(B') \subseteq P(C')$ we know that there is a variable v such that v is not present in B' neither as a positive nor as a negative literal, but v is present as a negative literal in C'. This variable, v, is also a node in $\mathcal{D}$. Now we have the following situation: we have a small circle, $V(N(B'))$, and a bigger one, $V(N(C'))$, such that $N(B') \subset N(C')$. In the graph there is no exit point to v from $V(N(B'))$, so there must be a path from $V(N(B'))$ to v such that this bigger cycle is created. Without loss of generality, let us assume that this path is the following: $b, d, v_1 \ldots, v_q$, such that $q \geq 1$, $v_q = v$, b is in $V(N(B'))$, d is not in $V(N(B'))$. Note that d must be in $P(B')$ and in $V(N(C'))$. Therefore, it is not true that $B' \subseteq C'$, because there is a variable, d, which occurs as a positive literal in B' but as a negative literal in C', i.e., $B' \not\subseteq C'$. Hence, C is not subsumed in $\mathcal{WM} \setminus \{C\}$. ■

Now let us prove the generalization of this lemma. We do not use the previous lemma to prove the next theorem, but we use its main idea: clause C' is not subsumed by clause B' because there must be a path $b, d, \ldots, v$ from $V(N(B'))$ to $V(N(C'))$, such that d occurs as a positive literal in B', but as a negative literal in C', hence, C' is not subsumed by B'.

Theorem 2. *Let $\mathcal{D}$ be a communication graph. Let $\mathcal{WM}$ be the weak model of $\mathcal{D}$. Then for all C in $\mathcal{WM}$ we have that C is independent in $\mathcal{WM} \setminus \{C\}$.*

Proof: Assume $\mathcal{D}$ is a communication graph. Assume $\mathcal{WM}$ is the weak model of $\mathcal{D}$. We show that for all C in $\mathcal{WM}$ we have that C is independent in $\mathcal{WM} \setminus \{C\}$. To show that let C' be an arbitrary but fixed clause from $\mathcal{WM}$, we show that C' is independent in $\mathcal{WM} \setminus \{C'\}$. To show this it is enough to show that there exists a full-length clause C'' such that C' subsumes C'', i.e., $C' \subseteq C''$, and there is no other clause in $\mathcal{WM}$ which subsumes C'', i.e., C'' is independent in $\mathcal{WM} \setminus \{C'\}$, i.e., for all B in $\mathcal{WM}$, such that $B \neq C'$ we have that $B \not\subseteq C''$. To show this let B' be an arbitrary

but fixed clause from $\mathcal{WM}$, such that $B' \neq C'$, we show that $B' \not\subseteq C''$. To show this let us construct C'' as follows: C'' contains all literals from C' and the rest of variables as positive literals, i.e., $C'' := C' \cup V \setminus V(C')$, where V is the set of variables in $\mathcal{D}$. Because of this construction we have that $N(C'') = N(C')$. In the followings we use the idea from the proof of Lemma 6. We know that $C'' = N(C'') \cup P(C'')$, and $B' = N(B') \cup P(B')$. Because of the construction of $\mathcal{WM}$ neither of these sets are empty. There are two cases: either (a) $N(B') \not\subseteq N(C'')$ or $P(B') \not\subseteq P(C'')$; or (b) $N(B') \subseteq N(C'')$ and $P(B') \subseteq P(C'')$. In case (a) we trivially have that $B' \not\subseteq C''$. In case (b) we have that $V(N(B'))$ represents a cycle, and $V(N(C''))$ represents also a cycle. From the construction of $\mathcal{WM}$ we know that $N(B') \neq N(C'')$, because each cycle is represented only by one clause and $B' \neq C'$. From this and from $N(C'') = N(C')$, we have that $N(B') \subset N(C'')$. From this, and from $P(B') \subseteq P(C'')$ we know that there is a variable v such that v is not present in B' neither as a positive nor as a negative literal, but v is present as a negative literal in C''. This variable, v, is also a node in $\mathcal{D}$. Now we have the following situation: we have a small circle, $V(N(B'))$, and a bigger one, $V(N(C''))$, such that $N(B') \subset N(C'')$. In the graph there is no exit point to v from $V(N(B'))$, so there must be a path from $V(N(B'))$ to v such that this bigger cycle is created. Without loss of generality, let us assume that this path is the following: $b, d, v_1 \ldots, v_q$, such that $q \geq 1$, $v_q = v$, b is in $V(N(B'))$, d is not in $V(N(B'))$. Note that d must be in $P(B')$ and in $V(N(C''))$. Therefore, it is not true that $B' \subseteq C''$, because there is a variable, d, which occurs as a positive literal in B' but as a negative literal in C'', i.e., $B' \not\subseteq C''$. Hence, C is independent in $\mathcal{WM} \setminus \{C\}$. ■

This is a very powerful theorem. It states that the weak model is the weakest possible one. This theorem has several corollaries.

Corollary 1. *Let $\mathcal{D}$ be a strongly connected communication graph. Let $\mathcal{WM}$ be the weak model of $\mathcal{D}$. Then $\mathcal{WM} \cup \{BB, WW\}$ is a minimal unsatisfiable SAT instance.*

Proof: From the construction of the weak model we know that neither BB nor WW is entailed by $\mathcal{WM}$. From this, from Lemma 1 and from Theorem 2 follows this corollary. ■

This is the corollary which is mentioned in the title of this paper. This one gives us an algorithm, how to generate a minimal unsatisfiable SAT instance from a strong digraph: First we generate the weak model of the input strong digraph. This will be a Black-and-White SAT problem in which all clauses are independent. As a second step, we add the black and the white clauses to it. Then, we are ready, we have a minimal unsatisfiable SAT instance, i.e., its unsatisfiable core is itself, i.e., intuitively speaking, there is no redundancy in it.

Corollary 2. *Let $\mathcal{D}$ be a strongly connected communication graph. Let $\mathcal{WM}$ be the weak model of $\mathcal{D}$. Then there is no blocked clause in $\mathcal{WM} \cup \{BB, WW\}$.*

Proof: This is an immediate consequence of Corollary 1 and the property of blocked clauses that they can be removed from a clause set without changing its satisfiability. ■

This is a less interesting corollary. Maybe we can use it later if we try to generalize our results to the Balatonboglár model of communication graphs.

VI. Test results

We have generated two sets of SAT instances with our weak model. In each case the input digraph was a strong one. In each case we have added the black and the white clause to make the instance unsatisfiable.

The two sets can be downloaded from here:

- http://aries.ektf.hu/~birocs/cnf/BARABASI_UNSAT.ZIP
- http://aries.ektf.hu/~birocs/cnf/TELJES_UNSAT.ZIP

Our file naming convention is the following: NumberOfNodes_NumberOfEdges_Density_UsedModel.cnf

In each set there are 17 items, there is one with $4, 5, \ldots, 20$ nodes. So the number of variables is very small. But the number of clauses is very high. For example in the file 19_68_0.20_WM.cnf we have 19 variables and 133233 clauses.

The first set is generated from scale-free digraphs. The graphs was generated by this tool: https://networkx.github.io/documentation/networkx-1.9/reference/generated/networkx.generators.random_graphs.barabasi_albert_graph.html. This tool returns random graph using Barabási-Albert preferential attachment model [2]. The m parameter was 2.

The second set is generated from complete graphs, so they contain all the possible full-length clauses. Note, that there are 2^n possible full-length clauses, if n is the number of variables. This set was very good to test our generator script.

We have tested these instances by 5 state-of-the-art SAT solvers: MiniSat2.2.1, MapleLCMDistChronoBt-DL-v2.2, MapleLCMDistChronoBt-DL-v3, Glucose-4.2.1, CSFLOC19. We did not use any switches of them, we ran them with their default settings.

MapleLCMDistChr* [24] was the winner of SAT Race 2019, see http://sat-race-2019.ciirc.cvut.cz/. Glucose [4] is one of the best SAT solvers in the last few years. MiniSat [12] is used in many test cases, so it is a good reference point. CSFLOC [16] is our own SAT solver. In its newest version, see http://fmv.ektf.hu/files/CSFLOC19.java, we can measure how many clauses there are in the input which the solver does not touch. This output parameter is called 'numberOfUnUsedClauses'. This number has to be zero in case of a minimal unsatisfiable clause set, and indeed, this number was zero for all the test cases in these two sets.

To perform the tests we used a virtual machine with Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz, 4 GB RAM, Linux, Ubuntu Xenial.

Figure 4 shows the runtime results on the first problem set. $V + E$ is the sum of the number of vertices and the number of edges in the input strong digraph.

We can see that CSFLOC19 was the best when the number of variables was bigger than 13. CSFLOC is one of the best solver if the ratio n/m is very small, where n is the number of variables, and m is the number of clauses.

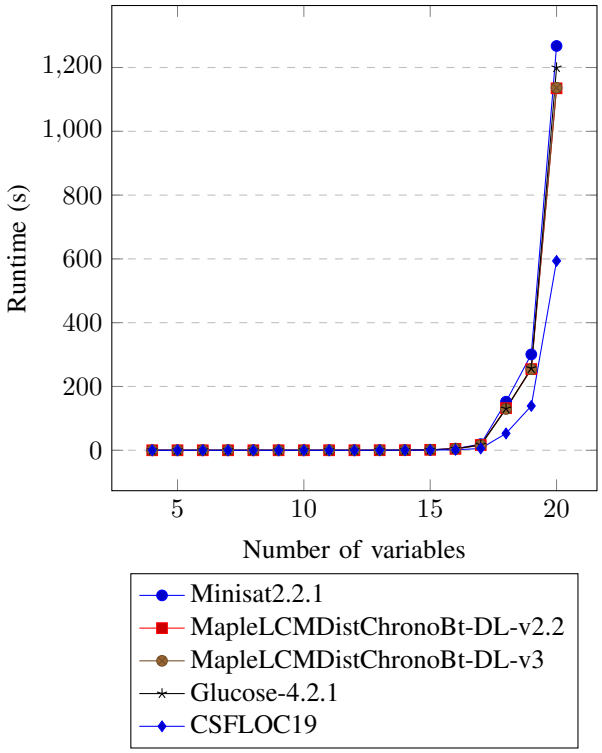

Fig. 4. Runtimes on the first problem set, all instances are UNSAT

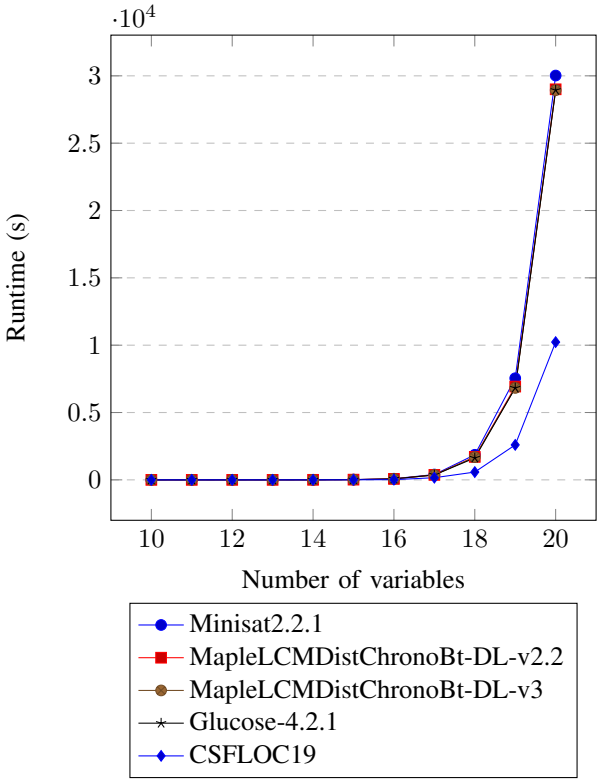

Fig. 5. Runtimes on the second problem set, all instances are UNSAT

VII. Future work

This work is purely theoretical, there is no usage described here. This does not mean that this is not possible. In this section we list some possible usages and future ideas.

The BaW 1.0 SAT solver solves the SAT problems generated by our strong model in linear time [7]. The proof of Theorem 1 suggests that we might find a similar algorithm for SAT problems generated by our weak model.

We are very close to generalize BaW 1.0, called BaW 2.0, for the Balatonboglár model. The main idea is to propagate two units (not one, as we used to do in BaW 1.0) in the first run to earn new units. This means that we have 4 branches at the highest level. We have found a way which guarantees no decisions in the first 3 branches. We have to understand the last branch better to be able to finish BaW 2.0.

It is an interesting question how to represent a 3-SAT problem as a directed graph. Most probably it is not feasible, because then we could translate a 3-SAT problem into a directed graph, and then that directed graph into a 2-SAT problem. Although this direction seems to be unrealistic, it is still interesting and very challenging for us.

In this direction, one of the problems is what to do with those clauses in a 3-SAT problem, which subsumes the white clause, or the black one. To solve this problem we need a technique which creates a Black-and-White SAT problem, if the input SAT problem is unsatisfiable. We give some possible answers to this question in one of our newest paper, see [5].

We believe that our new theorem, see Theorem 2, and its corollaries will help us to understand how to use resolution to create shorter clauses if we know the input communication graph.

Acknowledgment

The authors would like to thank to the project "Complex improvement of research capacities and services at Eszterházy Károly University", project ID: EFOP-3.6.1-16-2016-00001, to support this research.

References

[1] H. Abbasizanjani, O. Kullmann, *Minimal Unsatisfiability and Minimal Strongly Connected Digraphs*, Theory and Applications of Satisfiability Testing - SAT 2018, Lecture Notes in Computer Science, vol 10929, pp. 329–345, 2018.

[2] Réka Albert, Albert-László Barabási, *Statistical mechanics of complex networks*, Reviews of Modern Physics 74 (1), pp. 47-–97, 2002.

[3] B. Aspvall, M. F. Plass, R. E. Tarjan, *A Linear-Time Algorithm For Testing The Truth Of Certain Quantified Boolean Formulas*, Information Processing Letters, pp. 121–123, 1979.

[4] G. Audemard, L. Simon, *Glucose in the SAT Race 2019*, Proceedings of SAT Race 2019, pp. 19–20, 2019.

[5] T. Balla, Cs. Biró, G. Kusper, *The BWConverter Toolchain: An Incomplete Way to Convert SAT Problems into Directed Graphs*, Proceedings of ICAI 2020, http://ceur-ws.org/Vol-2650/paper3.pdf, pp. 24–29, 2020.

[6] A. Biere, M. Heule, H. van Maaren, T. Walsh, *Handbook of Satisfiability*, IOS Press, Amsterdam, 2009.

[7] Cs. Biró, G. Kusper, *BaW 1.0 - A Problem Specific SAT Solver for Effective Strong Connectivity Testing in Sparse Directed Graphs*, Proceedings of CINTI 2018, pp. 160–165, 2018.

[8] Cs. Biró, G. Kusper, *Equivalence of Strongly Connected Graphs and Black-and-White 2-SAT Problems*, Miskolc Mathematical Notes, Vol. 19, No. 2, pp. 755-768, 2018.

[9] R. E. BRYANT, *Graph-Based Algorithms for Boolean Function Manipulation*, IEEE Trans. Comput., pp. 677–691, 1986.

[10] S. A. COOK, *The Complexity of Theorem-Proving Procedures*, Proc. of STOC'71, pp. 151–158, 1971.

[11] G. DAVYDOV, I. DAVYDOVA, AND H.K. BÜNING, *An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF*, Annals of Mathematics and Artificial Intelligence 23, pp. 229-–245, https://doi.org/10.1023/A:1018924526592, 1998.

[12] N. EÉN, N. SÖRENSSON, *An extensible sat-solver*, Lecture Notes in Computer Science, vol. 2919, pp. 502-–518, 2003.

[13] R. A. HEARN, *Games, Puzzles, and Computation*, PhD thesis, MIT, June 2006.

[14] L. HELLERMAN, *A Catalog of Three-Variable Or-Invert and And-Invert Logical Circuits*, IEEE Transactions on Electronic Computers, pp. 198–223, 1963.

[15] G. KUSPER, CS. BIRÓ, *Convert a Strongly Connected Directed Graph to a Black-and-White 3-SAT Problem by the Balatonboglár Model*, submitted to Theory of Computing, 17 pages, submitted on 24.08.2019, status: under review.

[16] G. KUSPER, CS. BIRÓ, GY. B. ISZÁLY, *SAT solving by CSFLOC, the next generation of full-length clause counting algorithms*, Proceedings of IEEE International Conference on Future IoT Technologies 2018, DOI: 10.1109/FIOT.2018.8325589, 2018.

[17] O. KULLMANN, *New methods for 3-SAT decision and worst-case analysis*, Theoretical Computer Science, 223(1-2): pp. 1–72, 1999.

[18] O. KULLMANN, *On a generalization of extended resolution*, Discrete Applied Mathematics, 96-97(1-3) pp. 149–176, 1999.

[19] G. KUSPER, CS. BIRÓ, T. BALLA, *Investigation of the Efficiency of Conversion of Directed Graphs to 3-SAT Problems*, Proceedings of SACI-2020, DOI: 10.1109/SACI49304.2020.9118786, pp. 227–234, 2020.

[20] G. KUSPER, CS. BIRÓ, T. BALLA, *Representing Directed Graphs as 3-SAT Problems using the Simplified Balatonboglár Model*, ICAI-2020, poster, https://icai.uni-eszterhazy.hu/2020/abstracts/ICAI_2020_abstract_128.pdf, 2020.

[21] R. P. LANGLANDS, *Problems in the theory of automorphic forms to Salomon Bochner in gratitude*, Lectures in Modern Analysis and Applications III, pp. 18–61, 1970.

[22] S. MINATO, *Zero-suppressed BDDs for Set Manipulation in Combinatorial Problems*, Proceedings of the 30th International Design Automation Conference, pp.272–277, 1993.

[23] B. NAGY, *Truth-teller-liar puzzles and their graphs*, Central European Journal of Operations Research - CEJOR 11, pp. 57–72, 2003.

[24] V. RYVCHIN, A. NADEL, *Maple LCM Dist ChronoBT: Featuring chronological backtracking,*, Proceedings of SAT Competition, 2018.

[25] R. E. TARJAN, *Depth-First Search and Linear Graph Algorithms*, SIAM Journal on Computing, Vol. 1, No. 2, pp. 146–160, 1972.

[26] A. WEIL, *Uber die Bestimmung Dirichletscher Reihen durch Funktionalgleichungen*, Mathematische Annalen, 149–156, 1967.

[27] A. J. WILES, *Modular elliptic curves and Fermat's Last Theorem*, ANNALS OF MATH, pp. 443–551, 1995.