*mathematics*

MDPI

*Article*

# Resolvable Networks—A Graphical Tool for Representing and Solving SAT

**Gábor Kusper [1,*], Csaba Biró [1,2] and Benedek Nagy [3,*]**

[1] Faculty of Informatics, Eszterházy Károly Catholic University, 3300 Eger, Hungary; biro.csaba@uni-eszterhazy.hu
[2] Faculty of Informatics, Eötvös Lóránd University, 1117 Budapest, Hungary
[3] Department of Mathematics, Faculty of Arts and Sciences, Eastern Mediterranean University, North Cyprus, Mersin-10, Famagusta 99450, Turkey
* Correspondence: kusper.gabor@uni-eszterhazy.hu (G.K.); benedek.nagy@emu.edu.tr (B.N.)

**Abstract:** In this paper, we introduce the notion of resolvable networks. A resolvable network is a digraph of subnetworks, where subnetworks may overlap, and the inner structure of subnetworks are not interesting from the viewpoint of the network. There are two special subnetworks, Source and Sink, with the following properties: there is no incoming edge to Source, and there is no outgoing edge from Sink. Any resolvable network can be represented by a satisfiability problem in Boolean logic (shortly, SAT problem), and any SAT problem can be represented by a resolvable network. Because of that, the resolution operation is valid also for resolvable networks. We can use resolution to find out or refine the inner structure of subnetworks. We give also a pessimistic and an optimistic interpretation of subnetworks. In the pessimistic case, we assume that inside a subnetwork, all communication possibilities are represented as part of the resolvable network. In the optimistic case, we assume that each subnetwork is strongly connected. We show that any SAT problem can be visualized using the pessimistic interpretation. We show that transitivity is very limited in the pessimistic interpretation, and in this case, transitivity corresponds to resolution of clauses. In the optimistic interpretation of subnetworks, we have transitivity without any further condition, but not all SAT problems can be represented in this case; however, any such network can be represented as a SAT problem. The newly introduced graphical concept allows to use terminology and tools from directed graphs in the field of SAT and also to give graphical representations of various concepts of satisfiability problems. A resolvable network is also a suitable data structure to study, for example, wireless sensor networks. The visualization power of resolvable networks is demonstrated on some pigeon hole SAT problems. Another important application field could be modeling the communication network of an information bank. Here, a subnetwork represents a dataset of a user which is secured by a proxy. Any communication should be done through the proxy, and this constraint can be checked using our model.

**Keywords:** resolvable networks; digraphs; SAT problem; graphical representations and tools; resolution



**Citation:** Kusper, G.; Biró, C.; Nagy, B. Resolvable Networks—A Graphical Tool for Representing and Solving SAT. *Mathematics* **2021**, *9*, 2597. https://doi.org/10.3390/math9202597

Academic Editor: Miklós Krész

Received: 28 August 2021
Accepted: 12 October 2021
Published: 15 October 2021



## 1. Introduction

Complex networks can be found everywhere, and they belong to an interdisciplinary field of science [1–3]. Their modeling and analysis is of great importance. From a mathematical point of view, the networks appear in the theory of graphs. Topology can represent and characterize the properties of the entire network structure. A topology represents a real network and usually it is converted to either a directed or an undirected graph. The objects or events of the model are assigned to the vertices of the graph, and the edges are frequently used to describe various relations between the objects or events, e.g., dependency relations. The topological models are usually calculated based on some probabilities for large networks [4–7].

The other way to cope with the complexity of large networks is that we create a model that allows us to neglect some of the fine details of the network, but to highlight its large-scale structure, the topology. We neglect the details of the network by using the notion of subnetwork.

In this paper, we introduce the novel concept of resolvable networks, which are structures that have the following three properties: a resolvable network (1) consists of subnetworks, which are a set of nodes, and since subnetworks are sets, their inner structure is unconstrained; (2) non-overlapping subnetworks may communicate by passing messages over directed edges; and (3) subnetworks may overlap.

The inner structure of subnetworks is uninteresting from the viewpoint of resolvable networks, so we can highlight the topology. On the other hand, subnetworks may overlap, and we can use this feature to reveal the inner structure of subnetworks. However, we pay a price for that, as the graphical representation (in 2D) will be less and less intuitive if we add an increasing number of details. This is why we allow and use two different interpretations for subnetworks.

Before we detail these interpretations, let us recall the concept of Boolean satisfiability problems. From now on, we will use the short term SAT problem as it is widely used in the literature for these problems. By SAT problem, we mean the problem of propositional satisfiability for formulas in CNF (conjunctive normal form), i.e., formulas that are represented by clause sets. SAT is one of the most known NP-complete problems [8], and it has connections to many subfields of computer science, including theoretical computer science, artificial intelligence, theory of algorithms, hardware design, and formal verification [9].

Now, we are turning back to the possible interpretations of our subnetworks. The less constrained one is called the pessimistic interpretation of subnetworks. In this interpretation, there are no further constraints, i.e., the only constraints are those which are needed to define a resolvable network. In this interpretation, any resolvable network can be represented as a clause set, i.e., as a SAT problem. Moreover, it works also the other way around: any SAT problem can be represented by a resolvable network. This is the main property of the pessimistic interpretation of subnetworks, which allows people working on SAT to have a new graphical interpretation, a tool that helps to understand better the problem and helps to find its solution.

The main property of the pessimistic interpretation of subnetworks makes sure that we can use resolution to refine the structure of a resolvable network by adding more and more edges to it. We can also use this feature to depict any SAT problem as a resolvable network, although, the picture might be very complex as we are going to demonstrate it by some examples.

On the other hand, we discuss also the optimistic interpretation of subnetworks. In this case we have an extra constraint, inside a subnetwork each node can send a message to any other node directly or indirectly, so the subnetwork is strongly connected. In this interpretation any resolvable network can be represented as a clause set, i.e., as a SAT problem, but not the other way around; not all SAT problems can be represented by resolvable networks, if we use the optimistic interpretation. Resolution is still a valid operation in this interpretation. Furthermore, the system has the property of transitivity, which means that if we know that the subnetwork $A$ can send messages to subnetwork $B$ and $B$ to $C$, then $A$ can also send messages to $C$.

This work has several motivations. In general, since many people are visual, i.e., they can more easily obtain information from a diagram than from a text, graphical representations are important to provide better understanding of the concepts and thus also for teaching purposes.

This research is conducted as a part of the "Graph to SAT and SAT to Graph" project, (see https://www.researchgate.net/project/Graph-to-SAT-and-SAT-to-Graph, accessed on 13 October 2021), which aims to find a suitable graph representation for the SAT problem. We believe that if we could visualize the SAT problem, then we would gain more insight on it that could help also to find more efficient solutions for them, at least in some subcases.

This paper is inspired by one of the recent papers of the above project, which introduces the so-called weak model of communication graphs [10].

A communication graph is suitable to model, for example, a wireless sensor network. The sensors are the nodes of the communication graph. If a sensor can send a message to another one, then this fact is represented by a directed edge. Finally, each node should be labeled by a unique Boolean variable.

The weak model has some nice properties; for example, if the communication graph is strongly connected, then its weak model is a black and white SAT problem [10]. On the other hand, the weak model is computationally very expensive, because we have to find all circuits in the input communication graph to be able to generate its weak model. We have to find all circuits, not only the elementary ones, and some graphs might have exponentially many circuits compared to the number of their nodes, just like in the case of the complete graph $K_4$, see [11]. Therefore, our aim is to find a new digraph-based representation, which represents a digraph as a SAT problem.

In the next section, Section 2, we overview similar works. In Section 3, we give some basic definitions, and then we introduce the novel notion of the resolvable network in several subsections. Section 4, among other results, shows the close relation between resolution and resolvable networks. In Section 5, we present the pessimistic and the optimistic interpretation of subnetworks. Section 6 presents further applications. Finally, in Section 7, we present the conclusions and some possible future work.

## 2. Related Works

There are quite a few works which aim to represent the SAT problem in some graphical ways, usually as a digraph:

- An implication graph [12] is a digraph which represents implication between literals as edges, and the vertices are Boolean variables and their negation. Each implication graph is skew-symmetric, because $\neg A \implies B$ if and only if $\neg B \implies A$. Any 2-SAT problem can be represented by an implication graph, but the general SAT problem cannot be represented by this method. In these graphs, each edge represents logical consequence.
- AIG, And-Inverter Graph [13], is a directed acyclic graph (DAG) where the root is a formula, the leaves are Boolean variables, other vertices are AND gates with two input edges, and a marked edge means INVERTER, i.e., logical negation. Any SAT formula can be represented as an AIG, but is not suitable to represent big formulas. Each edge represents input read which is negated (in case of marked edge) or not.
- BDD, Reduced Ordered Binary Decision Diagram [14], is a DAG, where vertices are Boolean variables, except the leaves, which are either 0 or 1. Each non-leaf vertex has two edges: 0-edge, and 1-edge, which represents the interpretation of the vertex. Each path is an interpretation of the represented formula. A path which ends in a 1-leaf satisfies the represented formula. One has to merge any isomorphic subgraphs and eliminate any vertex whose two children are isomorphic. Thus, there will be only one 0 and one 1-leaf. In this way, we can reduce the graph, which makes it suitable to represent bigger formulas. Any SAT problem can be represented in this way. Each edge represents an interpretation of its source.
- ZDD or ZBDD, Zero-Suppressed Binary Decision Diagram [15], is a kind of BDD, but it uses the rule "eliminate those vertices whose 1-edge points directly to the 0", instead of: "eliminate any vertex whose two children are isomorphic". If we represent a SAT problem which has only a few solutions, then ZDD is a more suitable choice than BDD. Here, similarly to the previously recalled BDD, each edge represents an interpretation of its source.
- The graph representation of logical puzzles/games is a well-known connection between graphs and logical formulae. Some examples are [16,17]. The meaning of an edge depends on its interpretation. In a usual interpretation, an edge represents a move or a statement of a player.

- Minimal unsatisfiable SAT instances can be generated from minimal strongly connected digraphs [18] by representing edges as implications, and adding so-called monotone clauses to the resulting clause set. We note that this result is a special case of applying our results on weak models of communication graphs to minimal strongly connected digraphs.

Although there are lots of representations which either represent a SAT problem graphically, or some graph is represented as a SAT problem, there is no such representation yet which has the following property: any SAT problem can be represented as some graphical object, and any such graphical object can be represented as a SAT problem. This is one of the new and nice properties of the resolvable networks which we present in this work.

One could argue that, for example, a BDD can be represented by a logical formula, and any formula can be represented by a SAT problem, but this is not an immediate correspondence. Moreover, we will show another advantage of our new model over BDD.

## 3. Definitions and Basic Lemmas

In this section, first, we fix the notions and notations and recall some known facts that are needed to understand the paper. Then, in various subsections, we also give the definitions of our new concepts and we prove some facts about them.

Boolean variables are called *literals*; more precisely, they are positive literals. Their negations are called negative literals. Examples of literals are: $d, \neg d, e, \neg e, \ldots$.

A set of literals is called a *clause*. A set of them is called a *clause set*. Formally, a *SAT problem* is given by a clause set. An *assignment* is also represented by a set of literals. Let a set of literals be given as a clause or as an assignment. Any Boolean variable may occur in this set 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 an assignment or a clause contains exactly $n$ literals, then it is an *n-assignment* or an *n-clause*, respectively. In special cases, when $n = 1$ and $n = 2$ in an $n$-clause, a *unit* and a *binary clause* are obtained, respectively. An *n-SAT problem* is a clause set where its clauses have at most $n$ literals. A clause from a clause set is called a *full-length clause* if and only if it contains all variables of the clause set.

Some further notions will be used. Let $K$ be a clause. Then, let $V(K)$ denote the set of variables which occur in $K$; further, let $N(K)$ denote the set of negative literals of $K$; and finally, let $P(K)$ denote the set of positive literals of $K$. Obviously, $K = N(K) \cup P(K)$, i.e., the clause as a set of literals, is the union of its negative and positive literals. Further, $V(N(K)) \cap V(P(K)) = \emptyset$, that is, each literal of a clause, is either a negative or a positive literal, but cannot be both. Furthermore, $P(K) = V(P(K))$ as the positive literals appear in the clause exactly as variables.

The negation of a set $L$ of literals is denoted by $\neg L$ and it refers to elementwise negation, i.e., in $\neg L$ each element of $L$ is negated. Observe that $\neg\neg L = L$.

Let $H$ be the set of variables of a clause set. $WHITE$ is said to be the *white clause* or the *white assignment* on variables $H$ if and only if $WHITE = H$. Similarly, $BLACK$ is the *black clause* or the *black assignment* on variables $H$ if and only if $BLACK = \neg H$. For instance, let $H = \{a, b, c\}$, then $WHITE = \{a, b, c\}$, and $BLACK = \{\neg a, \neg b, \neg c\}$. Notice that $WHITE = \neg BLACK$, and $BLACK = \neg WHITE$.

The clause $K$ *subsumes* clause $C$ if and only if $K$ is a subset of $C$.

Further, the clause set $H$ *subsumes* clause $K$ if and only if there is a clause in $H$ which subsumes $K$. Formally: $H$ *subsumes* $K \iff \exists C(C \in H \wedge C \subseteq K)$.

The *clause difference* of two clauses $C$ and $D$, denoted by $diff(C, D)$, is defined as $diff(C, D) = C \cap \neg D$. If $diff(C, D) \neq \emptyset$, then we say that $C$ *differs from* $D$. If $|diff(C, D)| = k$, then we say that $C$ and $D$ *differs in k variables*. Note that $diff(C, C) = \emptyset$, i.e., $C$ does not differ from $C$, and $diff(C, \neg C) = C$.

Further, $diff(C,D) = \neg(diff(D,C))$, and $|diff(C,D)| = |diff(D,C)|$. Let the *symmetric difference* of two clauses $C$ and $D$ be defined as $C \oplus D = diff(C,D) \cup diff(D,C)$.

Examples: $diff(\{a,b,c\},\{\neg c,\neg d,\neg e\}) = \{c\}$, $diff(\{\neg c,\neg d,\neg e\},\{a,b,c\}) = \{\neg c\}$, and $\{a,b,c\} \oplus \{\neg c,\neg d,\neg e\}) = \{c,\neg c\}$.

*Resolution can be performed* on two clauses if and only if they differ in exactly one variable. If resolution can be performed, then the *resolvent*, denoted by $Res(K,C)$, is defined as $Res(K,C) = (K \cup C) \setminus (K \oplus C)$.

The assignment $S$ is said to be a *solution* of clause set $H$ if and only if for all $K \in H$, $S \cap K \neq \emptyset$.

Further, the clause set $H$ is a *BaW SAT* problem if and only if it has exactly two solutions, the white and black assignments, i.e., $WHITE$ and $BLACK$.

Now, we briefly recall some concepts of graph theory. The pair $\mathcal{D} = (\mathcal{V},\mathcal{E})$ is a *digraph*, with the set of vertices $\mathcal{V}$ and the set of edges $\mathcal{E}$. An edge is an ordered pair of vertices. If $(f,e)$ is an element of $\mathcal{E}$, then we may say that $(f,e)$ is an edge of $\mathcal{D}$.

We may also call $\mathcal{D}$ a communication graph if the following conditions hold: for all $e$ in $\mathcal{V}$, $(e,e)$ is not in $\mathcal{E}$; moreover, the elements of $\mathcal{V}$ are identified by Boolean variables, i.e., by positive literals. At communication graphs, the word node is also used as a synonym of vertex.

Now, we briefly recall a crucial property of the weak model of communication graphs from [10]. If we have a circuit $(a_1,a_2,\ldots,a_m,a_1)$ in the input communication graph with exit points $\{b_1,b_2,\ldots,b_n\}$, then this circuit is represented by the clause $\{\neg a_1,\neg a_2,\ldots,\neg a_m,b_1,$ $b_2,\ldots,b_n\}$.

In this paper, we are generalizing this idea as follows. If we have two subnetworks (see the next subsection for formal description) $A$ and $B$, such that $A$ consists of the nodes $\{a_1,a_2,\ldots,a_m\}$, and $B$ consists of the nodes $\{b_1,b_2,\ldots,b_n\}$, and $A$ can send messages to $B$, i.e., $(A,B)$ is an edge of the studied resolvable network, then we represent this by the same clause: $\{\neg a_1,\neg a_2,\ldots,\neg a_m,b_1,b_2,\ldots,b_n\}$. Since one edge of a resolvable network can be represented as a clause, the whole resolvable network can be represented by a SAT problem. This is the main idea of this work, and now we are ready to see the formal definition of the model.

### *3.1. Definition of Resolvable Networks*

In this subsection, we formally define the notion of a resolvable network and also provide an example. Some basic results as lemmas are coming in some subsequent subsections.

Let $\mathcal{V}$ be the set of vertices as it is usual in graph theory. *Subnetworks* are sets of vertices. Since subnetworks are sets, their inner structure is arbitrary, i.e., may not be precisely defined. A *network* consist of subnetworks connected by directed edges.

We say that a structure is a *resolvable network* if

(1) it consists of subnetworks;
(2) non-overlapping subnetworks may be connected by directed edges in it;
(3) its subnetworks may overlap.

More formally:

**Definition 1.** *Let $\mathcal{V}$ be the set of vertices. $A \neq \emptyset$ is a subnetwork if $A \subseteq \mathcal{V}$. The digraph $\mathcal{N} = (\mathcal{SN} \cup \{Source, Sink\}, \mathcal{R})$ is a resolvable network, where $\mathcal{SN}$ is the set of its subnetworks, and $\mathcal{R}$ is the set of its edges. $\mathcal{V}(\mathcal{N}) = \bigcup(\mathcal{SN})$ is the set of vertices of $\mathcal{N}$. An edge is an ordered pair of non-overlapping subnetworks, i.e., if $(A,B)$ is an element of $\mathcal{R}$, then $A \cap B = \emptyset$.*

*If $(A,B)$ is an element of $\mathcal{R}$, then we may say that $(A,B)$ is an edge of $\mathcal{N}$, or $(A,B)$ is a reach of $\mathcal{N}$. The reach $(A,B)$ is depicted by $A \to B$.*

*Source has the property that there is no incoming edge to it. Sink has the property that there is no outgoing edge from it. Both of them are representing the empty set, but in different roles.*

If there is no edge of $\mathcal{N}$ that contains the subnetwork $A$ or *Source* or *Sink*, then we may neglect them in graphical representations. In our graphical representations, we may use only those elements of $\mathcal{SN} \cup \{Source, Sink\}$ that appear in at least one edge in $\mathcal{R}$.

Example: Let us have three subnetworks: $A = \{a, b\}, B = \{c\}, C = \{d, e\}$ with the following two edges: $(A, B), (B, C)$. This resolvable network is depicted in Figure 1.

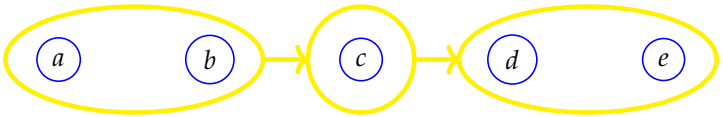


**Figure 1.** A resolvable network: $(\{A, B, C, Source, Sink\}, \{(A, B), (B, C)\})$, where $A = \{a, b\}$, $B = \{c\}$, $C = \{d, e\}$.

We may use the word reach as a synonym of edge to emphasize that the edge $(A, B)$ means a bit more than that subnetwork $A$ may send messages to subnetwork $B$, because there are vertices in these subnetworks which realize the message passing—see the next subsection.

### *3.2. Interpretation of Reaches*

The semantics of resolvable networks are motivated by the concept of message sending, which is a very basic operation of a network. In these semantics, vertices are called nodes, and nodes can be used as Boolean variables, similarly as in [10] and as we have already mentioned.

The reach $(\{a\}, \{b\})$ is interpreted as follows: The node $a$ can send messages to node $b$, or in other words, we can send messages from $a$ to $b$ in the network. This reach can be represented by the logical formula $(a \implies b)$. It is also worth noting here that, to make our description clear, we use $\implies$ to represent the Boolean logical implication. Furthermore, the reach $(\{a, b\}, \{c, d\})$ means that we can send messages from the subnetwork $\{a, b\}$ to $\{c, d\}$, which abbreviates the fact that messages can be sent

- from $a$ to $c$; or
- from $a$ to $d$; or
- from $b$ to $c$; or
- from $b$ to $d$.

This means that at least one of the above possibilities works, but the possibility is not fixed. Thus, this reach, which is also represented by $\{a, b\} \to \{c, d\}$, can also be represented by the logical formula: $(a \implies c) \vee (a \implies d) \vee (b \implies c) \vee (b \implies d)$, which is equivalent to this logical formula: $(\neg a \vee \neg b \vee c \vee d)$, which is equivalent to this clause: $\{\neg a, \neg b, c, d\}$.

This means that a reach can be represented by a clause, and vice versa, a clause can be represented by a reach; see later.

**Definition 2** (*Interpretation of a reach*)**.** *The disjunctive interpretation, or for short, the interpretation of the reach $(A, B)$, is defined as $(A, B)_\vee = \bigvee_{a \in A \wedge b \in B} (a \implies b)$, and it reads that $(A, B)_\vee$ is the (disjunctive) interpretation of the reach $(A, B)$.*

The next lemma is essential since it enables us to represent a reach as a clause, and the other way around, to represent a clause as a reach.

**Lemma 1.** *Let $From = \{a_1, a_2, \ldots, a_m\}$ and $To = \{b_1, b_2, \ldots, b_n\}$. Assume that From and To are distinct sets of nodes. If $F = (From, To)_\vee$ and $G = \neg From \cup To$, then $F = G$, i.e., formula F and the formula which is represented by clause G are logically equivalent to each other.*

**Proof.** We know in general that $(a \implies b) = (\neg a \vee b)$. We know also that *From* and *To* are distinct sets of nodes. From the definition of interpretation of a reach, we know that

$F = \bigvee_{i=1\ldots m,\ j=1\ldots n} (a_1 \implies b_j)$. From these, using the basic properties of logical disjunction, we obtain that $F = (\neg a_1 \vee \neg a_2 \vee \cdots \vee \neg a_m \vee b_1 \vee b_2 \vee \cdots \vee b_n)$, which is represented by the clause $\{\neg a_1, \neg a_2, \ldots, \neg a_m, b_1, b_2, \ldots, b_n\}$. Hence, $F = G$; the formula $F$ and the formula which is represented by clause $G$ are logically equivalent to each other. □

This lemma enables us to represent a reach as a clause, and a clause as a reach, because if $(A, B)$ is a reach, then we know, by the definition of reach, that $A$, $B$ are distinct sets, and in this case we have $(A, B)_\vee = \neg A \cup B$.

For example, the reach $(\{a, b\}, \{c, d\})$ can be represented by the clause $\{\neg a, \neg b, c, d\}$, and the other way around: the clause $\{\neg a, \neg b, c, d\}$ can be represented by the reach $(\{a, b\}, \{c, d\})$.

Note that the clause $\{a, b, c\}$ is represented by the reach $(\emptyset, \{a, b, c\}) = (Source, \{a, b, c\})$, and the other way around: a reach with $From = Source$ represents a clause with only positive literals of $To$.

Furthermore, the clause $\{\neg a, \neg b, \neg c\}$ is represented by the reach $(\{a, b, c\}, \emptyset) = (\{a, b, c\}, Sink)$, and the other way around: a reach with $To = Sink$ represents a clause with only negative literals of $From$.

In addition, as a special case, the empty clause is represented by the reach $(\emptyset, \emptyset) = (Source, Sink)$, and vice versa.

### *3.3. The Double Meaning of 'Reach': The Noun and the Verb*

The word 'reach' has two distinct types: it can be a noun or a verb. In this paper, we may use this word in both of its roles. However, these are not independent. In this subsection, we give some details on that.

If we have two reaches $(A, B)$ and $(B, C)$, then the following question arises: Can we send messages from the subnetwork $A$ to subnetwork $C$, or in other words, can we reach from $A$ to $C$? This question shows how we use the verb meaning of the word reach to describe the expressiveness of resolvable networks. Thus, we can use the word reach as a noun, such as in "$(A, C)$ is a reach", and also as a verb, such as "from $A$ we can reach $C$".

Now, to come back to our research, let us see how can we deal with the above question: If we have two reaches $(A, B)$ and $(B, C)$, then can we reach from $A$ to $C$? The answer to this question, in general, is that it depends on $B$ or on its inner structure. If $B$ has only one node, then the answer is definitely yes. Otherwise, it can be either no or yes. If inside $B$ the nodes cannot send messages to each other, then the answer is probably no. If inside $B$ we find a strongly connected communication graph, see [10], then the answer is yes. See also the pessimistic and the optimistic semantics of subnetworks. We will come back to this point in Section 5.

In general, we have no knowledge of what is inside a subnetwork, but if we have two subnetworks, $A$, $B$, then we know that one of the following pairs of statements is true: either $A$, $B$ are distinct or not; if they are distinct, then either $(A, B)$ is a reach or not, or $(B, A)$ is a reach or not. If they are not distinct, then neither $(A, B)$ is a reach, nor $(B, A)$ is a reach, but they overlap, so they have some common nodes. These structures might help us to find out the inner structure of a subnetwork using the clause representation of reaches and performing resolution over them.

Some of the reaches could be in a relation that helps us to discover some new relations among them. The following relation of the reaches plays a crucial role.

**Definition 3** (*Basic transitivity in a resolvable network*)**.** *We say that two reaches $(A, B)$ and $(B, C)$ of a resolvable network $N = (SN, R)$ are transitive if $|B| = 1$, and in this case, we say that from $A$ we can reach $C$ in the resolvable network $N$.*

### *3.4. Interpretation of Resolvable Networks*

In this subsection, we show that any resolvable network can be interpreted as a clause set. A resolvable network consist of reaches, and each reach can be represented by a clause.

The question is what the interpretation is if we have two reaches $(A, B)$, and $(A, C)$. How should we interpret the relation of those? There are two obvious ways to connect them: either by 'and' or by 'or'. Thus, the question is as follows. Does having both of the reaches $(A, B)$ and $(A, C)$ mean that $A$ can reach $B$ and $C$? Or, does this mean that $A$ can reach $B$ or $C$?

In the first case, it is very easy to create the SAT representation of a resolvable network, because between two reaches we have conjunction just like in the case of the SAT problem, between two clauses we have conjunction, and a reach can be represented as a clause, as we learned from Lemma 1.

In the second case, we have disjunction between two reaches, but still, using the ideas of the weak model of communication graphs, see [10], we could create a meaningful logical representation. However, in this paper, we choose and work with the first possibility, and thus, we are ready to see what logical formulae are representing the resolvable networks.

**Definition 4** (*Interpretation of a resolvable network*)**.** *The conjunctive interpretation, or for short, interpretation, of the resolvable network $N = (SN, R)$ is defined as $N_\wedge = (SN, R)_\wedge = \bigwedge_{(A,B)\in R}(A, B)_\vee$, and it reads that $(SN, R)_\wedge$ is the (conjunctive) interpretation of the resolvable network $(SN, R)$.*

The following lemma states that any resolvable network can be represented by a SAT problem.

**Lemma 2.** *Let $N = (SN, R)$ be a resolvable network. Let $F = N_\wedge$. Let $S = \{\neg A \cup B \mid (A, B) \in R\}$. Then, $F = S$, i.e., formula $F$ and the formula which is represented by clause set $S$ are logically equivalent to each other.*

**Proof.** By definition of interpretation of a resolvable network, we know that $F = (SN, R)_\wedge = \bigwedge_{(A,B)\in R}(A, B)_\vee$. By definition of a reach, we know that this can be written as $F = (SN, R)_\wedge = \bigwedge_{(A,B)\in R}(\bigvee_{a\in A \wedge b\in B}(a \implies b))$. From this, by Lemma 1, we obtain that $F = (SN, R)_\wedge = \bigwedge_{(A,B)\in R}(\neg A \cup B)$. From this, using the interpretation of a clause set, we obtain that $F$ can be represented by the clause set $S = \{\neg A \cup B \mid (A, B) \in R\}$. Hence, $F = S$, i.e., formula $F$ and the formula which is represented by clause set $S$ are logically equivalent to each other. □

Example:

Let us take the resolvable network in Figure 1, where we have three subnetworks: $A = \{a, b\}, B = \{c\}, C = \{d, e\}$, and two reaches: $(A, B), (B, C)$. This resolvable network is represented by the following clause set: $\{\{\neg a, \neg b, c\}, \{\neg c, d, e\}\}$.

The following lemma states that any SAT problem can be represented by a resolvable network. Before the lemma, we define an auxiliary function.

**Definition 5.** *We define the function ClauseToReach which creates a reach out of a clause, where the 'from' part is created from the negative literals, and the 'to' part is created from the positive literals. More formally, $ClauseToReach(C) = (A, B)$, where $A = V(N(C))$, if $N(C) \neq \emptyset$, otherwise $A = Source$, and $B = P(C)$, if $P(C) \neq \emptyset$, otherwise $B = Sink$.*

Examples:
$ClauseToReach(\{\neg a, b, \neg c, d\}) = (\{a, c\}, \{b, d\})$.
$ClauseToReach(\{\neg a, \neg b\}) = (\{a, b\}, Sink)$.
$ClauseToReach(\{a, b, c\}) = (Source, \{a, b, c\})$.
$ClauseToReach(\emptyset) = (Source, Sink)$.

Now, we are ready to state our lemma.

**Lemma 3.** *Let $S$ be a clause set. Let $R = \{ClauseToReach(C) \mid C \in S\}$. Let $SN = \bigcup_{(A,B)\in R}\{A, B\}$. Let $N = (SN, R)$. Then, $S = N_\wedge$.*

**Proof.** The key idea of the proof is that if $C \in S$ is a clause, then $N(C)$ is the set of its negative literals, $V(N(C))$ is the set of variables of its negative literals, and $P(C)$ is the set of its positive literals. Note that $P(C) = V(P(C)$. Furthermore, $V(N(C))$ and $P(C)$ are distinct sets. If we use Boolean variables as vertices, then $(V(N(C)), P(C))$ is a reach, so we can use the $ClauseToReach(C)$ function from Definition 5. From this, by Lemma 1 we have that $(V(N(C)), P(C))_\vee = C$. We know that $R = \{ClauseToReach(C) \mid C \in S\}$, and $SN = \bigcup_{(A,B)\in R}\{A, B\}$., since for any $C \in S$ we know that $ClauseToReach(C)$ is a reach, and we know that $N = (SN, R)$ is a resolvable network. From this, being Lemma 2, we obtain that $S = N_\wedge$. □

We have proven that, in fact, SAT problems and resolvable networks are equivalent, i.e., any SAT problem can be represented by a resolvable network and vice versa. These results allow us to transfer some of the well-known and widely used concepts to our resolvable networks including, e.g., *solution*. We can also say that a resolvable network is *satisfiable*. (It is if the represented SAT problem is satisfiable.) We provide also the formal definition of this latter notion:

**Definition 6** (*Satisfiable resolvable network*)**.** *We say that resolvable network $N = (SN, R)$ is satisfiable if and only if $S = \{\neg A \cup B \mid (A, B) \in R\}$ is satisfiable.*

Now, as we have concluded that based on the results presented in the lemmas of this section, we are able to represent SAT problems by resolvable networks, we are presenting some examples.

Our examples correspond to the well-known pigeon hole problem [19]. The pigeon hole problem with $m$ pigeons and $n$ holes is the problem to place $m$ pigeons in $n$ holes without placing two pigeons in the same hole. Depending on the values of $m$ and $n$, the problem may have or may not have solutions, i.e., the formula representing the problem may or may not be satisfiable.

We use the DIMACS format to provide the examples:

```
c a pigeon hole problem with 2 pigeons and 1 hole
p cnf 2 3 % a problem with 2 variables and 3 clauses
1 0 % pigeon1 sits in hole1
2 0 % pigeon2 sits in hole1
-1 -2 0 % pigeon1 and 2 cannot sit in hole1 at the same time
```

There are two Boolean variables used in this formula; we refer to them as 1 and 2. The 1 is true if and only if the pigeon1 sits in the hole (we have only one hole in this example), and 2 is true if and only if pigeon2 sits in the hole. There are three clauses (the three bottom lines of the DIMACS description) describing our aims:

- pigeon1 must sit somewhere, i.e., it should sit in the hole.
- pigeon2 must also sit somewhere, i.e., it should sit in the hole.
- there is only one hole; it is a place only of at most one pigeon, i.e., it could not happen that both pigeons are sitting there.

The problem can be written also as a SAT problem represented by the formula $1 \wedge 2 \wedge (\neg 1 \vee \neg 2)$; moreover, Figure 2 shows the corresponding resolvable network: $(\{Source, \{1\}, \{2\}, \{1,2\}, Sink\}, \{(Source, \{1\}), (Source, \{2\}), (\{1,2\}, Sink)\})$.

To answer the question of whether the represented SAT problem is satisfiable or not, we have to check whether any message from *Source* goes to *Sink*, or if the message can be lost. If the message can be lost, then the SAT problem is satisfiable, otherwise not. In Figure 2, there are two outgoing edges from *Source*, the blue ones. Each node broadcasts its message and sends it over using all of its outgoing edges. This means that the message from *Source* arrives at 1 and 2. A subnetwork waits until all of its nodes get the message, then it broadcasts the message. In this case, we have the subnetwork $\{1, 2\}$, since both nodes get the message from *Source* it sends to its children, which is in this case the *Sink*. Since

the message arrived from *Source* to *Sink*, the represented SAT problem, this resolvable network is unsatisfiable.

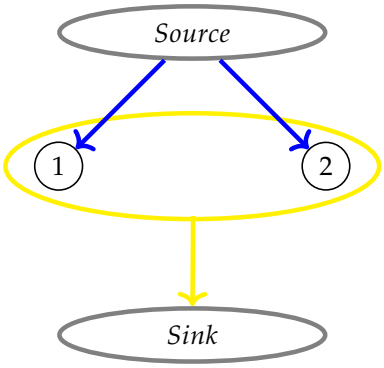


**Figure 2.** The resolvable network of the pigeon hole problem with 2 pigeons and 1 hole.

In our next example, we have $m = 2$ pigeons and $n = 2$ holes. The DIMACS representation of the problem is shown below:

```
c a pigeon hole problem with 2 pigeons and 2 holes
p cnf 4 4 % a problem with 4 variables and 4 clauses
1 2 0 % pigeon1 sits in hole1 or in hole2
3 4 0 % pigeon2 sits in hole1 or in hole2
-1 -3 0 % pigeon1 and 2 cannot sit in hole1 at the same time
-2 -4 0 % pigeon1 and 2 cannot sit in hole2 at the same time
```

The problem is described by four variables, i.e.,

- 1 is representing if pigeon1 sits in hole1;
- 2 is representing if pigeon1 sits in hole2;
- 3 is representing if pigeon2 sits in hole1; and finally,
- 4 is representing if pigeon2 sits in hole2.

The four clauses refer to the four bottom lines of the DIMACS representation, representing that both pigeons need to sit somewhere, and in any hole there could not be more than one pigeon. The corresponding SAT problem can be written by the following formula: $(1 \vee 2) \wedge (3 \vee 4) \wedge (\neg 1 \vee \neg 3) \wedge (\neg 2 \vee \neg 4)$. Figure 3 shows the resolvable network of the problem with four reaches corresponding to the above-mentioned clauses.

To answer the question of whether Figure 3 represents a satisfiable SAT instance or not, we have to check whether a message from *Source* might be lost or not. *Source* has two children, $\{1,2\}$ and $\{3,4\}$. This means that one of these pairs of nodes gets the message: $(1,3)$, $(1,4)$, $(2,3)$, or $(2,4)$. In the first and in the last cases, the message is delivered to *Sink*, since we have two subnetworks $\{1,3\}$, and $\{2,4\}$ which can send message to *Sink*, but in the second and third cases, the message cannot be delivered. Therefore, this problem represents a satisfiable SAT instance. Indeed, the solutions for the model of the formula of the problem are the assignments $\{1,4\}$ and $\{2,3\}$. They mean that either pigeon1 sits in hole1 and pigeon2 in hole2, or pigeon1 sits in hole2 while pigeon2 sits in hole1; the problem is clearly solvable with these conditions.

Finally, let us take an example which corresponds to the pigeon hole problem with three pigeons and two holes. The DIMACS format of the example is as follows:

```
c a pigeon hole problem with 3 pigeons and 2 holes
p cnf 6 9  % a problem with 6 variables and 9 clauses
1 2 0      % pigeon1 sits in hole1 or in hole2
3 4 0      % pigeon2 sits in hole1 or in hole2
5 6 0      % pigeon3 sits in hole1 or in hole2
-1 -3 0    % pigeon1 and 2 cannot sit in hole1 at the same time
-1 -5 0    % pigeon1 and 3 cannot sit in hole1 at the same time
```

```
-3 -5 0    % pigeon2 and 3 cannot sit in hole1 at the same time
-2 -4 0    % pigeon1 and 2 cannot sit in hole2 at the same time
-2 -6 0    % pigeon1 and 3 cannot sit in hole2 at the same time
-4 -6 0    % pigeon2 and 3 cannot sit in hole2 at the same time
```

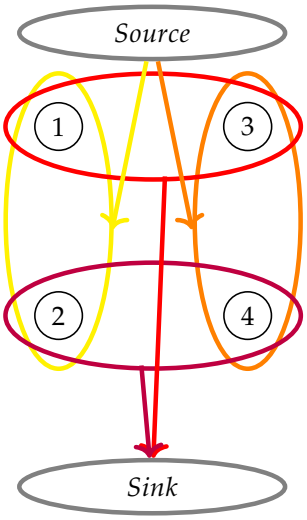


**Figure 3.** The resolvable network of the pigeon hole problem with 2 pigeons and 2 holes.

This SAT problem is depicted in Figure 4. We use colors because there are many edges. For example, the light green edge connects the light green subnetwork to *Sink*.

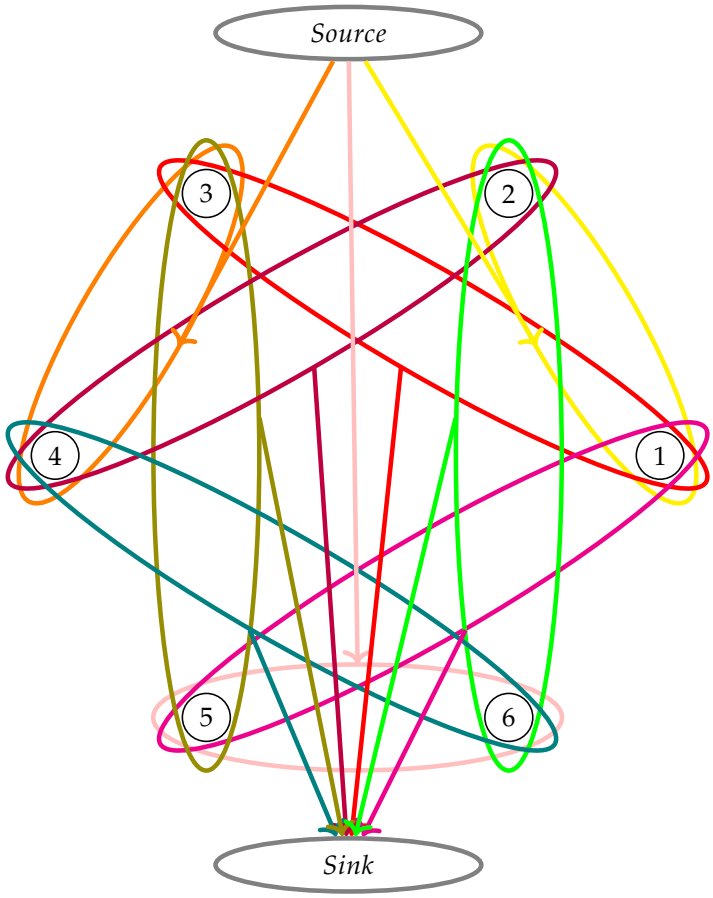


**Figure 4.** The resolvable network of the pigeon hole problem with 3 pigeons and 2 holes.

We do not reveal any secrets in recalling that the pigeon hole problem is solvable if and only if the number of pigeons is not greater than the number of holes. In the usual logical representation, the problem with $n+1$ pigeons and $n$ holes is considered with some value of $n$, as we have also shown examples of that type with $n = 1$ and $n = 2$.

### *3.5. Equivalence of Resolvable Networks*

A resolvable network is, in fact, interpreted as a formula of the SAT problem.

**Definition 7** (*Equivalence of resolvable networks*)**.** *We say that two resolvable networks are equivalent if and only if their interpretations are equivalent. More formally,* $N \equiv N' \iff N_\wedge \equiv N'_\wedge$*, where* $N$ *and* $N'$ *are resolvable networks.*

We expand the definition of the relation of reach to subnetworks as follows.

**Definition 8** (*Subnetwork can reach subnetwork*)**.** *We say that subnetwork* $A$ *can reach subnetwork* $B$ *in resolvable network* $N$ *if and only if the interpretation of the reach* $(A, B)$ *is a logical consequence of the interpretation of* $N$*, i.e.,* $N_\wedge \implies (A, B)_\vee$*.*

**Definition 9** (*Refined resolvable network*)**.** *Let* $N = (SN, R)$ *be a resolvable subnetwork. If subnetwork* $A$ *can reach subnetwork* $B$ *in* $N$*, then* $N' = (SN \cup \{A, B\}, R \cup \{(A, B)\})$ *is a refined resolvable network of* $N$*. We also overload the* $'+'$ *operator to abbreviate the above construction: If* $A$ *can reach* $B$ *in* $N$*, then* $N + (A, B) = (SN \cup \{A, B\}, R \cup \{(A, B)\})$*, where* $N = (SN, R)$*. In this case, we say that* $N + (A, B)$ *is a refined resolvable network of* $N$*.*

**Lemma 4.** *If subnetwork* $A$ *can reach subnetwork* $B$ *in resolvable network* $N$*, then* $N + (A, B)$ *and* $N$ *are equivalent.*

**Proof.** From the assumption that $A$ can reach $B$ in $N$, we know by Definition 9 that $N + (A, B)$ is a refined resolvable network of $N$. From the same assumption, we know by Definition 8 that interpretation of the reach $(A, B)$ is a logical consequence of the interpretation of $N$. From this, by the well-known fact that $(X \implies Y) \implies (X \equiv X \wedge Y)$, we know that $N_\wedge \equiv [N + (A, B)]_\wedge$, i.e., by applying Definition 7, they are equivalent resolvable networks. □

### *3.6. On the Scalability of the Method*

On the one hand, as usual, visual techniques, e.g., Venn diagrams have strict limitations, i.e., they work well and are really useful for better understanding and teaching if the represented example/problem has a relatively small size. On the other hand, visual techniques could help to find new techniques and methods which may also work on large(r) size problems. As we know, industrial-sized SAT problems might have millions of clauses and some hundreds of thousands variables. We cannot visualize such a big SAT instance in a meaningful way, except if we perform some simplifications. Thus, in this subsection, we investigate the question of how to create a bird's eye view perspective of big SAT instances.

One of the options is to split the set of nodes into two subsets, take them as subnetworks, abstract away the inner structure of these two subnetworks, and show the relations between them.

Since it is not easy to follow a raw text, we give this bird's perspective algorithm in a more structured way:

1. Create two almost equal size distinct subnetworks, say $A$ and $B$, such that each node belongs either to $A$ or $B$.
2. Show the two subnetworks without any inner structures without detailing their elements by abstracting away each clause $C$ which has the property $V(C) \subseteq A$ or $V(C) \subseteq B$.

3. If there is a clause $C$ such that $V(N(C)) \subseteq A$ and $P(C) \subseteq B$, then draw an edge from $A$ to $B$.
4. If there is a clause $C$ such that $V(N(C)) \subseteq B$ and $P(C) \subseteq A$, then draw an edge from $B$ to $A$.
5. If there is a clause $C$ such that $V(N(C)) \not\subseteq A \wedge V(N(C)) \not\subseteq B$ or $P(C) \not\subseteq A \wedge P(C) \not\subseteq B$, then create an additional subnetwork $D$ which intersects both $A$ and $B$.

The last step could be more detailed. We could depict in different ways the cases if $C = P(C)$, or $C = N(C)$, or $|V(N(C)) \cap A| = 1$, and so on. It is the subject of further investigations which are the interesting cases.

One of the interesting cases is the following: If $S$ is a satisfiable SAT instance and $M$ is one of its solutions, then we can choose $A$ and $B$ in this way: $A = V(N(M))$, and $B = P(M)$. In this way, we will see that there is either an edge from $A$ to $B$, or a subnetwork which intersects both $A$ and $B$, or both, or neither of them, but there will be no edge from $B$ to $A$. There will be no edge from $B$ to $A$ because the corresponding clause would not be satisfied by the solution $M$; see also Proposition 1.

**Proposition 1.** *Let $S$ be a satisfiable SAT problem. Let $M$ be a solution of $S$. Then, $S$ does not subsume $\neg M$.*

**Proof.** This is a special case of the Lemma 4.3.1. case (b) in [20]. □

We show this kind of bird's perspective created from the well-known SAT instance, called: $uf20-01.cnf$. This problem can be downloaded from the SATLIB - Benchmark Problems; see https://www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/RND3SAT/uf20-91.tar.gz, accessed on 13 October 2021. We only show the first few lines of this problem since it has 91 clauses, but our method can be explained already based on these data.

```
c This Formula is generated by mcnf
c
c    horn? no
c    forced? no
c    mixed sat? no
c    clause length = 3
c
p cnf 20  91
4 -18 19 0
3 18 -5 0
-5 -8 -15 0
-20 7 -16 0
10 -13 -7 0
-12 -9 17 0
```

This problem is satisfiable, and one of its solutions is $M = \{1, -2, -3, 4, -5, 6, -7, -8, -9, 10, -11, -12, 13, 14, 15, -16, 17, -18, -19, 20\}$; thus, we should choose $A$, $B$ as follows: $A = \{2, 3, 5, 7, 8, 9, 11, 12, 16, 18, 19\}$ and $B = \{1, 4, 6, 10, 13, 14, 15, 17, 20\}$.

Then, we get the bird's perspective shown in Figure 5. We can see that there is an edge from $A$ to $B$ (see step 3 of the bird's perspective algorithm) because $uf20-01.cnf$ contains, for example, the clause $-12\ \ -9\ 17\ 0$ (all its literals appear in the solution and it has both positive and negative literals). There is also a subnetwork which intersects $A$ and $B$ (see step 5 of the bird's perspective algorithm) because of, for example, the clause $4\ \ -18\ 19\ 0$: here, we have a subnetwork $\{4, 19\}$ which has an intersection with both $A$ and $B$.

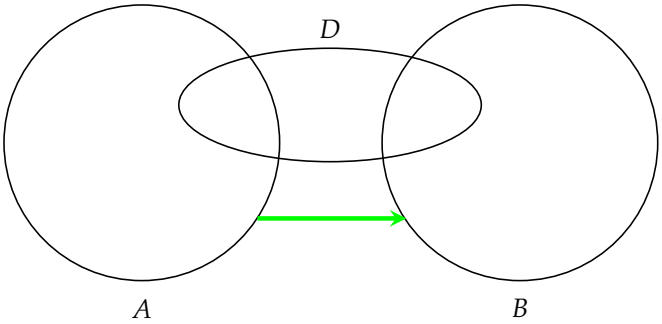


**Figure 5.** Bird's perspective representation of $uf20-01.cnf$ using $A = V(N(M))$ and $B = P(M)$, where $M$ is a solution of $uf20-01.cnf$.

There are also other options to obtain a simplified view on a SAT problem. For example, we can show only those clauses which have exactly $n_N$ negative literals and exactly $n_P$ positive literals. For example, if we have a 3-SAT problem and first we show the layer where $n_N = 0$ and $n_P = 3$, then the layer $n_N = 1$ and $n_P = 2$, then the layer where $n_N = 2$ and $n_P = 1$, and finally the layer with $n_N = 3$ and $n_P = 0$, then we could see some structure of the problem. This possibility is not investigated in this work.

In this subsection, we showed that not only small-sized problems can be handled by our model, but we may also have the chance to perform a kind of hierarchical modeling for large-sized problems; however, some of the technical details are left for a future study. Evidently, by making abstractions, some information is lost; however, for that price we can deal with large-sized problems.

## 4. Main Results

In this section, we present the main theoretical results of the paper in two subsections. We start with the connection of satisfiability and our model, and then, we show how resolution can be applied in our model.

### *4.1. Satisfiability of Resolvable Networks*

Based on the equivalence of the resolvable networks and the SAT problems, we can establish the following theorem.

**Theorem 1.** *A resolvable network is satisfiable if and only if there is an assignment of* 0*'s and* 1*'s to the nodes of the network such that each reach* $(A, B)$ *of the network has the following property:*

- *there is a node* $a \in A$ *such that* 0 *is assigned to* $a$*; or*
- *there is a node* $b \in B$ *such that* 1 *is assigned to* $b$*.*

**Proof.** A resolvable network is satisfiable if and only if the represented SAT problem is as well. The proof goes by directions.

Considering the first direction, let us assume that we have a satisfiable resolvable network. Then, there is a satisfiable SAT problem which is represented by this network (based on Lemma 2). Let $S$ be the solution of this SAT problem. Let us use this assignment to assign 1's to the literals which are in $S$ and 0's to the remaining ones. Since $S$ is a solution, every clause of the SAT problem is satisfied, i.e., in each clause there is a literal which also appears in $S$. Now, by applying Lemma 1, the clauses of the SAT problem can be "translated" to the reaches of our resolvable network. The above-mentioned property that each clause has is exactly the property stated in the theorem for the reaches, which proves the first direction.

Now, let us consider the other direction. Let us have a resolvable network with the condition stated in the theorem, and we will show that it is satisfiable.

By Lemma 2, we can obtain a SAT problem from our network (applying Lemma 1, the reaches of the given network can be "translated" to the clauses of the SAT problem). Since each reach has the property of the theorem, there is an assignment to the nodes with

the following property. Let $(A, B)$ be a reach of our network; then, by the condition of the theorem, there is a node $a \in A$ with assigned 0 or there is a node $b \in B$ with assigned 1. The clause we have obtained from this reach is then obviously satisfiable, since it contains a negative literal with assigned 0 or a positive literal with assigned 1. Since, in this way, each clause is satisfiable, the SAT problem itself is also satisfiable. Moreover, the assignment we have used for the network is providing the solution by taking the subset of nodes for which value 1 has been assigned. □

The theorem has the following immediate corollary for some special cases.

**Corollary 1.** *Let $(A, B)$ be a reach of a satisfiable resolvable network.*

- *If $A = Source$, then there is a node $b \in B$ such that 1 is assigned to $b$.*
- *If $B = Sink$, then there is a node $a \in A$ such that 0 is assigned to $a$.*

**Proof.** By remembering that both *Source* and *Sink* play the role of the empty set, the statements are clear from Theorem 1. □

*4.2. Resolution on Resolvable Networks*

The name resolvable network is explained in this subsection. Actually, resolution as a logical deduction step has a close connection to our reaches. On the one hand, usual resolutions can be represented in our networks as follows.

**Lemma 5.** *Let $N = (SN, R)$ such that $A, B, C, D \in SN$, $(A, B), (C, D) \in R$. Let $E_1 = (A, B)_\vee = \neg A \cup B$ and $E_2 = (C, D)_\vee = \neg C \cup D$. If we can perform resolution on $E_1, E_2$, such that $F = Res(E_1, E_2)$, then we can obtain the reach $ClauseToReach(F)$ from $N$.*

**Proof.** By Definition 9, we have to show that $Res(E_1, E_2)$ is the logical consequence of the interpretation of $N$. Since $E_1$ is the interpretation of a reach from $N$, and since $E_2$ is the interpretation of a reach from $N$, they are two clauses of $N_\wedge$. We assume that we can perform resolution on $E_1, E_2$, and the resolvent is $F = Res(E_1, E_2)$. In this case, using the basic properties of resolution, $F$ is a logical consequence of $N_\wedge$. Then, by using Definition 5 we can represent clause $F$ by the reach $ClauseToReach(F)$. Hence, we can obtain the reach $ClauseToReach(F)$ from $N$. □

To continue the description of the close relation of resolution techniques and our networks, some properties of them will be shown.

The next lemma states that basic transitivity in a resolvable network is the same as resolution.

**Lemma 6.** *Let $N = (SN, R)$ such that $A, B, C \in SN$, and $(A, B), (B, C) \in R$, $A, B, C$ are pairwise disjoint sets, and $|B| = 1$, i.e., $(A, B)$ and $(B, C)$ is transitive, so the reach $(A, C)$ can be obtained from $N$. Let $D_1 = (A, B)_\vee$ and $D_2 = (B, C)_\vee$. In this case, we know that $|diff(D_1, D_2)| = 1$, so we can perform resolution on $D_1, D_2$, and $Res(D_1, D_2) = (A, C)_\vee$.*

**Proof.** We know, by Lemma 1, that $D_1 = (A, B)_\vee = \neg A \cup B$ and $D_2 = (B, C)_\vee = \neg B \cup C$. From the assumption that $B$ is a singleton set and that $A, B, C$ are pairwise disjoint sets, we obtain that $|diff(D_1, D_2)| = 1$. By definition of resolution, we know that $Res(D_1, D_2) = \neg A \cup C$. From this, by Lemma 1, we obtain that $Res(D_1, D_2) = (A, C)_\vee$. Hence, by Lemma 5, the reach $(A, C)$ can be obtained from $N$. □

Example: let us consider the resolvable network shown in Figure 1. Here, applying Lemma 6, a new reach $(\{a, b\}, \{d, e\})$ is obtained, which is shown by a red arrow in Figure 6.

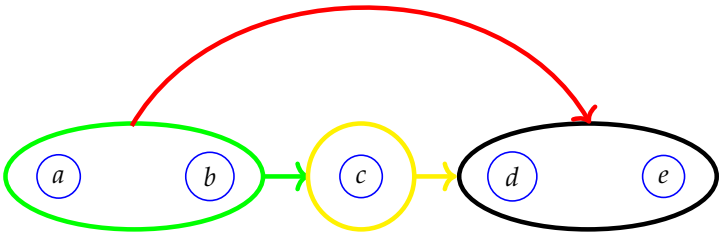


**Figure 6.** $A \implies B, B \implies C$ implies $A \implies C$, where $A = \{1,2\}, B = \{3\}, C = \{4,5\}$.

Now, we are ready to show how resolution can be defined as a step to evolve our resolution networks.

The next theorem shows the most general way of resolution over resolvable networks. This theorem also has a name, the *Auckland–Budapest Rule*. The name comes from the words "AUC" and "BUD" which we have if we read union using the "U" letter in the rule: If $|A \cap D| = 0$, and $|B \cap C| = 1$, then from the reaches $(A, B)$ and $(C, D)$ we obtain the reach $(A \cup C', B' \cup D)$, where $B' = B \setminus (B \cap C)$, and $C' = C \setminus (B \cap C)$. More formally:

**Theorem 2** (Auckland–Budapest Rule)**.** *Let $N = (SN, R)$ such that $A, B, C, D \in SN$, and $(A, B), (C, D) \in R$. If $|A \cap D| = 0$, and $|B \cap C| = 1$, then $(A \cup C', B' \cup D)$ can be reached from $N$, where $B' = B \setminus C$, and $C' = C \setminus B$.*

**Proof.** Let $E_1 = (A, B)_\vee = \neg A \cup B$, and $E_2 = (C, D)_\vee = \neg C \cup D$. We know by Definition 1 that $A \cap B = \emptyset$, and $C \cap D = \emptyset$, and from $|A \cap D| = 0$ we know also that $A \cap D = \emptyset$. Since there is no negation in a set of nodes, we also know that $|diff(A, C)| = 0$, and $|diff(B, D)| = 0$. From these and from $|B \cap C| = 1$, we obtain that resolution can be performed on $E_1$ and $E_2$, and $F = Res(E_1, E_2) = \neg A \cup B' \cup \neg C' \cup D$, where $B' = B \setminus (B \cap C)$, and $C' = C \setminus (B \cap C)$. Note that $B' = B \setminus C$, and $C' = C \setminus B$. From Lemma 1, we know that $F$ can be represented by the reach $ClauseToReach(F) = (A \cup C', B' \cup D)$. Since $(A \cup C', B' \cup D)_\vee = F = Res(E_1, E_2)$ by Lemma 5, we obtain that $(A \cup C', B' \cup D)$ can be reached from $N$. □

We provide several examples using the intuitive abbreviations $AUC'$ and $B'UD$:
Example 1: $A = \{1,2\}, B = \{3\}, C = \{3,4\}, D = \{5\}$, $AUC' = \{1,2,4\}, B'UD = \{5\}$.
Example 2: $A = \{1,2\}, B = \{3\}, C = \{3,4,5\}, D = Sink$, $AUC' = \{1,2,4,5\}, B'UD = Sink$.
Example 3: $A = \{1,2\}, B = \{3\}, C = \{2,3\}, D = \{4,5\}$, $AUC' = \{1,2\}, B'UD = \{4,5\}$.
Example 4: $A = \{1,2\}, B = \{3\}, C = \{2,3,4\}, D = \{5\}$, $AUC' = \{1,2,4\}, B'UD = \{5\}$.
Example 5: $A = \{1,2\}, B = \{3\}, C = \{2,3,4,5\}, D = Sink$, $AUC' = \{1,2,4,5\}, B'UD = Sink$.
Example 6: $A = Source, B = \{1,2,3\}, C = \{3\}, D = \{4,5\}$, $AUC' = Source, B'UD = \{1,2,4,5\}$.
Example 7: $A = Source, B = \{1,2,3\}, C = \{3,4,5\}, D = Sink$, $AUC' = \{4,5\}, B'UD = \{1,2\}$.
Example 8: $A = \{1,2\}, B = \{3\}, C = \{3,4\}, D = \{1,5\}$, the precondition $|A \cap D| = 0$ is not satisfied.

After those examples, we provide a more enhanced usage which extends the network using the following corollary.

**Corollary 2.** *Let $N = (SN, R)$ such that $A, B, C, D \in SN$, and $(A, B), (C, D) \in R$. If $|A \cap D| = 0$, and $|B \cap C| = 1$, then $N$ and $N + (A \cup C', B' \cup D)$ are equivalent, where $B' = B \setminus C$, and $C' = C \setminus B$.*

**Proof.** This corollary is an immediate consequence of Theorem 2 and Lemma 4. □

Using this corollary, we can use the new data structure to investigate the SAT problem. Here, we present a concrete case study.

As an application of Theorem 2, Figure 7 shows the newly established reaches of the resolvable network of the pigeon hole example displayed in Figure 4. We explain only one of those new reaches: the reaches $(Source, \{5,6\})$ and $(\{4,6\}, Sink)$ imply the reach $(\{4\},\{5\})$. Actually, each new reach is represented by an arrow between two singleton sets, i.e., sets containing exactly one node. As one may observe, if there is a reach between two nodes, then there is another one in the opposite direction; moreover, the set of all nodes forms a strongly connected communication graph.

In this way, by using the strong model of communication graphs from [10,21], we obtain that the underlying SAT problem of Figure 7 is a BaW SAT problem. We know that a BaW SAT problem has only two solutions, where each variable is true, and where each variable is false [21]. However, Figure 7 depicts only the reaches that we obtained using resolution, so we have to take into account also the reaches in Figure 4. On that resolvable subnetwork, we have several edges from *Source* to some other subnetwork (but not to *Sink*), so the assignment where all variables are false is not a model of the underlying SAT problem. Furthermore, since we have some edges to *Sink* from some subnetworks (but not from *Source*), we know that the assignment where all variables are true is not a model of the underlying SAT problem. Finally, all these together imply that this problem is clearly unsatisfiable, i.e., three pigeons do not fit into two holes. Considering the modification of the problem by erasing the condition represented in the original figure by the reach $(\{4,6\}, Sink)$ with the color teal, the new network is obtained without the reaches $(\{4\},\{5\})$ and $(\{6\},\{3\})$ shown by the dashed edges in Figure 7 that are missing.

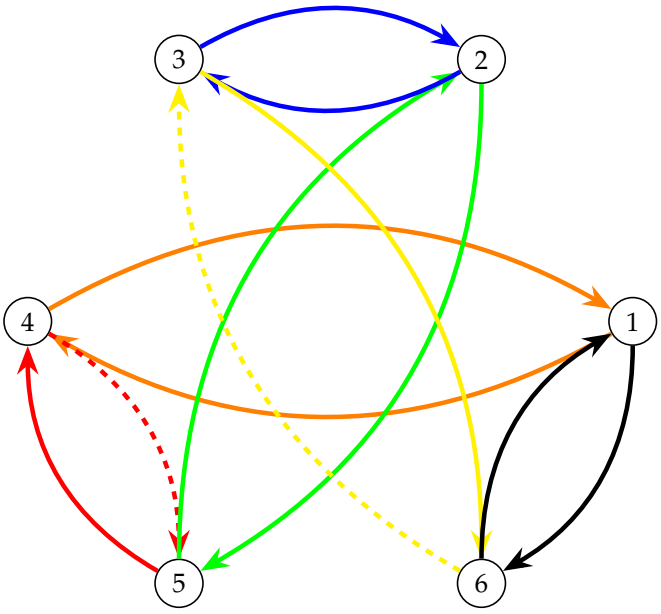


**Figure 7.** The new reaches obtained by resolution from the resolvable network of the pigeon hole problem shown in Figure 4.

## 5. On the Possible Interpretations of Subnetworks

As we may not be sure what the inner structure of the nodes of a subnetwork is, further study of the resolvable networks can be done by having various assumptions on these structures.

### *5.1. Pessimistic Interpretation of Subnetworks*

In the pessimistic interpretation, we assume that the subnetworks have no inner connections at all.

In this interpretation, we have very limited transitivity, which means that if we have two reaches: $(A, B)$ and $(B, C)$, then this does not necessary imply $(A, C)$. Now, using the intuitive $A \to B$ notation instead of $(A, B)$, we can give the above statement as follows. In this interpretation, we have very limited transitivity, which means that $A \to B \to C$ does not necessary implies $A \to C$. In this interpretation, we have proven transitivity if $|B| = 1$; see Lemma 6.

However, by using resolution, we may infer some information about the inner structures of the subnetworks, as we did in Figure 7. This means that the pessimistic interpretation is the basic one. If otherwise is not stated, we use the pessimistic one.

Now, as the other end of the scale, we present another possible interpretation with some assumptions on the inner structures of the subnetworks.

### *5.2. Optimistic Interpretation of Subnetworks*

In the optimistic interpretation, we assume that the subnetworks are strongly connected, i.e., all nodes can reach any other ones inside the subnetwork.

In this interpretation, by our assumption, we have transitivity, which means that $A \to B \to C$ always implies $A \to C$. We also present this result in a more formal way.

**Proposition 2.** *Let $N = (SN, R)$ such that $A, B, C \in SN$, and $(A, B), (B, C) \in R$, $A, B, C$ are pairwise disjoint sets, and $B$ is a strongly connected communication graph. Then, we can reach $C$ from $A$ in $N$.*

**Proof.** In this proof, we use the message-sending interpretation of our networks and reaches. Since $(A, B)$ is a reach, and $(B, C)$ is a reach, $B$ is neither the *Source* nor the *Sink*. Furthermore, there is a node $b_i$ in $B$ which receives messages from $A$, and there is a node $b_j$ in $B$ which sends messages to $C$. Since $B$ is a strongly connected communication graph, there is a path from $b_i$ to $b_j$, so any message sent by $A$ reaches $C$, so we can reach $C$ from $A$. Note that $b_i$ can be any node which is reachable from $A$, and $b_j$ can be any node from which $C$ is reachable, so they might not be fixed in all messages being passing around, but this does not change the proof. Hence, we can reach $C$ from $A$ in $N$. □

Now, after taking a look at the two extremal cases caused by the possible inner structures (or lack thereof) of the subnetworks, we may consider intermediate interpretations.

### *5.3. Intermediate Interpretation of Subnetworks*

We can imagine any other kind of interpretation of subnetworks, where any subnetwork might have its own set of constraints, such as that there must be at least one node be alive, and alive nodes should be able to communicate; or this subnetwork is optimized to save power; or this subnetwork uses star topology; or any other kind of restriction. We do not study these intermediate interpretations, but we encourage other research groups to consider these cases.

## 6. Application: Modeling Proxy with a Resolvable Network

In this section, we show an application of our system that provides a connection to another field of networking.

There are many kind of proxy. One of them is called secure proxy, which hides an important object or some secret. One can only access the secret through the proxy; there is no other way. The proxy should decide whether the access request can be granted or not.

Now, we model that subnetwork $A$, which might be a dataset of a user of an information bank, can be accessed only through $P$ (which is a proxy) from the rest (which is modeled by subnetwork $B$) of the system.

There are three subnetworks:

- $A$ consisting of the nodes $A_1, A_2$, which should be secured;
- $B$ consisting of the nodes $B_1, B_2, B_3$, which is the rest of the network;
- $P$ is a singleton set, which is the proxy that secures $A$.

An example for this scenario can be seen in Figure 8.

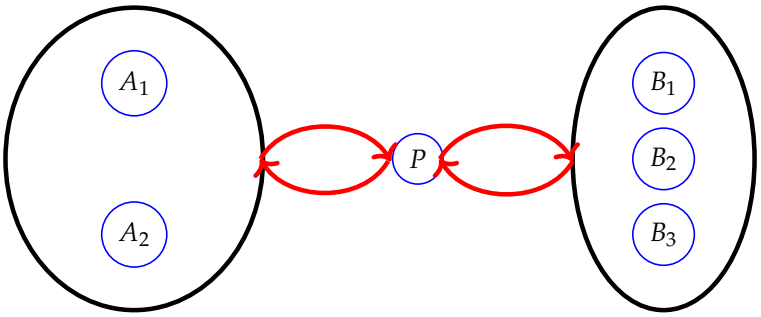


**Figure 8.** Resolvable network modeling communication through proxy $P$.

Based on the figure, we can use the following clause set to model this network:

$$\{\{A_1, A_2, \neg P\}, \{\neg A_1, \neg A_2, P\}, \{P, \neg B_1, \neg B_2, \neg B_3\}, \{B_1, B_2, B_3, \neg P\}\}$$

Let the value $P$ show whether the communication is granted through the proxy, i.e., if its value is true, the proxy grants and connects network $A$ with network $B$. Now, let us analyze the formula, i.e., to see what are its models and when this formula evaluates as true.

In the case when $P$ has a true truth-value, the formula is true if and only if at least one of $A_1$ and $A_2$ is true and at least one of $B_1$, $B_2$, and $B_3$ is also true. This means that there is a piece of data (or a person) in $B$ which has the right to communicate, let us say to ask a question, and there is also a node in the secured dataset, i.e., in subnetwork $A$, which has the right to answer the query. The proxy allows the communication exactly in these cases, showing the ability of our resolvable network to model this phenomenon of security networks.

## 7. Conclusions and Future Work

We have introduced a novel data structure, the resolvable network. In a sense, it is a generalization of digraphs, where a vertex represents a subnetwork. The inner structure of such a subnetwork is not constrained by the definition of resolvable network, but some usages might introduce constraint on them, just like in case of our optimistic interpretation of subnetworks. Subnetworks might overlap (there could be two different subnetworks such that they share some nodes), unlike digraphs (where distinct vertices could not have any common parts since they could be mapped to singleton sets). However, subnetworks are allowed to connect by edges only if they do not overlap, somehow inheriting that "non-overlapping" property of digraphs. Since digraphs are so widely used, we hope that this generalization will also be useful in some other areas of mathematics and modeling.

Our main result is the following. If we use the pessimistic interpretation of subnetworks, then any resolvable network can be represented by a SAT problem, and any SAT problem can be represented by a resolvable network. Thus, we can say that we have found an immediate connection between the SAT problems and their proposed graphical representations. This tight relationship between the SAT problems and resolvable networks allows us to enrich both sides by notions from the other side. For example, we can define the notion of satisfiable resolvable network (see Definition 6), or we can use resolution to obtain a refined resolvable network (see Definition 9 and Theorem 2). This bi-directional property is novel when comparing our new model to the existing ones mentioned in Section 2.

Note that in the new representation, i.e., in resolvable networks, an edge means a logical consequence relation. As we saw, an edge from subnetwork $A = \{a_1, a_2, \ldots, a_m\}$ to subnetwork $B = \{b_1, b_2, \ldots, b_n\}$ is represented by the clause $\{\neg a_1, \neg a_2, \ldots, \neg a_m, b_1, b_2, \ldots, b_n\}$, which is interpreted as $(\neg a_1 \vee \neg a_2 \vee \cdots \vee \neg a_m \vee b_1 \vee b_2 \vee \cdots \vee b_n\}$. This is equivalent to $((a_1 \wedge a_2 \wedge \cdots \wedge a_m) \implies (b_1 \vee b_2 \vee \cdots \vee b_n)\}$, i.e., an edge in a resolvable network can be

interpreted as logical consequence. We believe that logical consequence is a natural way to understand the role of the edges of graphs representing logical formulae.

In Section 2, we reviewed some other widely known graph representations of the SAT problem. As we have seen, an edge does not represent a logical consequence in the case of AIG, BDD, and ZDD; thus, although they are good for technical analysis, they do not support human understanding. On the other hand, one can represent each SAT problem by using these approaches, similarly to our new model. In the case of implication graphs, an edge means an implication relation, which is easy to follow, similarly to our proposed solution, but implication graphs can represent only 2-SAT problems, and not all SAT problems can be represented. Our new representation, the resolvable network, offers both benefits: an edge represents logical consequence, and, as we have already recalled, any SAT problem can be represented by a resolvable network. Based on the above-described reasons, we believe that our model contains more information about the nature of the SAT problems than other representations, and thus, further analysis of the model could bring real help and also more knowledge about the SAT problems and their solvability.

We studied some concrete examples of how to solve some SAT instances. Those examples suggest that we can solve the SAT problem without unit-propagation or resolution, which are the most basic techniques to solve SAT. To brew a generic SAT solver algorithm based on resolvable networks is our next goal.

Another direction of research, we should mention here, is based on the fact that in this work, we have defined and used the notion of *disjunctive* interpretation of reaches and the notion of *conjunctive* interpretation of resolvable networks. We can define alternative meaningful interpretations as follows. The *conjunctive* interpretation of the reach $(A, B)$ is defined as follows: $(A, B)_\wedge = \bigwedge_{a \in A \wedge b \in B} (a \implies b)$. The *disjunctive* interpretation of the resolvable network $(SN, R)$ is defined as $(SN, R)_\vee = \bigvee_{(A,B) \in R} (A, B)_\vee$. This later one is very interesting because in this case, we have disjunction between two reaches, so we could use the ideas of the weak model of communication graphs to create another SAT problem representation of resolvable networks, which might inherit the nice properties of the weak model of communication graphs.

In this work, each reach can be represented by a clause because we allow communication only of non-overlapping subnetworks; see *"(2) non-overlapping subnetworks may communicate by message passing over directed edge"* in the definition of resolvable networks. If we also allow communication of overlapping subnetworks, then the reach $(A, B)$ is represented by a tautology if $A \cap B \neq \emptyset$. This does not change the property that each SAT problem can be represented as a resolvable network. However, those reaches of a resolvable network which describe communication of overlapping subnetworks cannot be represented as a clause.

Finally, we hope to report usages from the field of biology. Subnetworks might represent tissues or cells and reaches might represent communication of them.

**Author Contributions:** Conceptualization, G.K. and C.B.; validation, G.K., C.B. and B.N.; formal analysis, G.K. and B.N.; writing—original draft preparation, G.K. and C.B.; writing—review and editing, G.K., C.B. and B.N.; visualization, C.B.; funding acquisition, G.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Hungarian Government, grant number: GINOP-2.1.2-8-1-4-16-2017-00176. Cs. Biró would like to thank the support of the Ministry of Innovation and Technology and the National Research, Development and Innovation Office within the Quantum Information National Laboratory of Hungary. The APC was funded by InnovITech Kft.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Dehmer, M.; Emmert-Streib, F. *Analysis of Complex Networks: From Biology to Linguistics*; Wiley-VCH: Weinheim, Germany, 2009.
2. Reijneveld, J.C.; Ponten, S.C.; Berendse, H.W.; Stam, C.J. The application of graph theoretical analysis to complex networks in the brain. *Clin. Neurophysiol.* **2007**, *118*, 2317–2331. [CrossRef] [PubMed]

3. Sayama, H. *Introduction to the Modeling and Analysis of Complex Systems*; Open SUNY Textbooks: New York, NY, USA, 2015.
4. Albert, R.; Barabási, A.L. Statistical mechanics of complex networks. *Rev. Mod. Phys.* **2002**, *74*, 77. [CrossRef]
5. Barabási, A.L.; Albert, R.; Jeong, H. Scale-free characteristics of random networks: The topology of the world-wide web. *Phys. A Stat. Mech. Its Appl.* **2000**, *281*, 69–77. [CrossRef]
6. Boccaletti, S.; Latora, V.; Moreno, Y.; Chavez, M.; Hwang, D.U. Complex networks: Structure and dynamics. *Phys. Rep.* **2006**, *424*, 175–308. [CrossRef]
7. Lesne, A. Complex Networks: From Graph Theory to Biology. *Lett. Math. Phys.* **2006**, *78*, 235–262. [CrossRef]
8. Cook, S.A. *The Complexity of Theorem-Proving Procedures*. In Proceedings of the STOC '71, Third Annual ACM Symposium on Theory of Computing, Shaker Heights, OH, USA, 3–5 May 1971; pp. 151–158.
9. Biere, A.; Heule, M.; van Maaren, H.; Walsh, T. *Handbook of Satisfiability*; IOS Press: Amsterdam, The Netherlands, 2009.
10. Kusper, G.; Biró, C. Convert a Strongly Connected Directed Graph to a Black-and-White 3-SAT Problem by the Balatonboglár Model. *Algorithms* **2020**, *13*, 321. [CrossRef]
11. Mateti, P.; Deo, N. On Algorithms for Enumerating All Circuits of a Graph. *SIAM J. Comput.* **1976**, *5*, 90–99. [CrossRef]
12. Aspvall, B.; Plass, M.F.; Tarjan, R.E. A Linear-Time Algorithm for Testing the Truth of Certain Quantified Boolean Formulas. *Inf. Process. Lett.* **1979**, *8*, 121–123. [CrossRef]
13. Hellerman, L. A Catalog of Three-Variable Or-Invert and And-Invert Logical Circuits. *IEEE Trans. Electron. Comput.* **1963**, *EC-12*, 198–223. [CrossRef]
14. Bryant, R.E. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Comput.* **1986**, *C-35*, 677–691. [CrossRef]
15. Minato, S. Zero-suppressed BDDs for Set Manipulation in Combinatorial Problems. In Proceedings of the 30th International Design Automation Conference, Dallas, TX, USA, 14–18 June 1993; pp. 272–277.
16. Hearn, R.A. Games, Puzzles, and Computation. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2006.
17. Nagy, B. Truth-teller-liar puzzles and their graphs. *Cent. Eur. J. Oper. Res.—CEJOR* **2003**, *11*, 57–72.
18. Abbasizanjani, H.; Kullmann, O. Minimal Unsatisfiability and Minimal Strongly Connected Digraphs. In *Theory and Applications of Satisfiability Testing—SAT 2018*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 10929, pp. 329–345.
19. Aloulb, F.A.; Ramani, A.; Markov, I.L.; Sakallah, K.A. ShatterPB: Symmetry-breaking for pseudo-boolean formulas. In Proceedings of the ASP-DAC 2004: Asia and South Pacific Design Automation Conference 2004 (IEEE Cat. No.04EX753), Yokohama, Japan, 27–30 January 2004; pp. 884–887. [CrossRef]
20. Kusper, G. Solving and Simplifying the Propositional Satisfiability Problem by Sub-Model Propagation. Ph.D. Thesis, RISC Institute, Johannes Kepler University Linz, Linz, Austria, 2005; pp. 1–146.
21. Biró, C.; Kusper, G. *Equivalence of Strongly Connected Graphs and Black-and-White 2-SAT Problems*. *Miskolc Math. Notes* **2018**, *19*, 755–768. [CrossRef]