Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I,
IMECS 2013, March 13 - 15, 2013, Hong Kong

SAT-Solving Of Certain Graph-Based CNF's

Stefan Porschen Tatjana Schmidt

***Abstract*—We propose an explicit linear-time algorithm for the satisfiability problem on the fraction of the CNF class whose members correspond to 1-outerplanar graphs. Our algorithm explicitly exploits the outerplanar structure.**

***Index Terms*—propositional satisfiability, 1-outerplanar graph, nested formula**

I. Introduction

The propositional satisfiability problem (SAT) of conjunctive normal form (CNF) formulas is an essential combinatorial problem, namely one of the first problems that have been proven to be NP-complete [6]. More precisely, it is the natural NP-complete problem and thus lies at the heart of computational complexity theory. Moreover SAT plays a fundamental role in the theory of designing exact algorithms, and it has a wide range of applications because many problems can be encoded as a SAT problem via reduction [8] due to the rich expressiveness of the CNF language. The applicational area is pushed by the fact that meanwhile several powerful solvers for SAT have been developed (cf. e.g. [12], [16] and references therein).

There are known several subclasses of CNF, restricted to which SAT behaves polynomial-time solvable. This article, from an algorithmic point of view, studies SAT restricted to a CNF fraction whose members are closely related to a specific class of simple graphs. Concretely, we design a linear-time SAT-algorithm for formulas whose *formula graph* is 1-outerplanar. Recall that an embedding of a graph G is 1-outerplanar if G is planar and all vertices lie on its outer face. Moreover, we define the vertex set of the formula graph G_C of formula C as the union of the variable set and the clause set of C. The edges of G_C are determined by the clause-vertex incidence, i.e., if a variable x occurs in a clause c, then the cooresponding vertices are joined by an edge in G_C. The outerplanar formulas form a subclass of the planar formula class where the formula graph is planar. Observe that SAT for 3-CNF instances which in addition are planar still remains NP-complete according to Lichtenstein [13].

In this paper we illustrate how SAT can be solved in linear time for the outerplanar formula class. The algorithm presented for this purpose works by exploiting the graph-structure of outerplanar formulas which is translated into a superstructure graph. Clearly outerplanar formulas are contained in the class of nested formulas studied by Knuth who also gives a linear-time SAT-algorithm [10]. Furthermore, since the treewidth of an outerplanar formula is 2, one can apply the nice tree decomposition due to Bodlaender [2], [3], [4], [5]. A corresponding linear-time SAT-algorithm for outerplanar formulas can be found in [15]. However, in this paper, we are specifically interested in the structure of the variable-clause graphs of outerplanar formulas, which we exploit for our algorithm explicit.

Mathematics Group, Department 4, HTW Berlin, D-10313 Berlin, Germany, e-mail: porschen@htw-berlin.de.
Waidmarkt 18, D-50676 Köln, Germany

II. Notation And Preliminaries

Let CNF denote the set of formulas (free of duplicate clauses) in conjunctive normal form over propositional variables $x \in \{0,1\}$. A *positive (negative)* literal is a (negated) variable. A clause c is a disjunction of different literals, and is represented as a set $c = \{l_1, \ldots, l_{|c|}\}$. Each formula $C \in$ CNF is considered as a set of its clauses $C = \{c_1, \ldots, c_{|C|}\}$ having in mind that it is a conjunction of these clauses. As usual, k-CNF denotes the subclass of CNF, where each member clause has length at most k, for fixed integer $k \geq 2$. The *negation (or complement)* of a literal l is $\bar{l}$. For formula C, clause c, by $V(C), V(c)$ we denote the variables contained (neglecting negations), correspondingly. Given $x \in V(c)$, by $l(x)$ we denote the literal over x that is contained in c. Furthermore, CNF_+ denotes the set of *monotone* formulas, i.e., every literal is positive.

The satisfiability problem (SAT) asks, whether input $C \in$ CNF has a *model*, which is a truth value assignment $t : V(C) \to \{0,1\}$ assigning at least one literal in each clause of C to 1. Let UNSAT denote the set of all unsatisfiable members of CNF, and let SAT denote the set of all satisfiable members of CNF. Clearly, we have $\emptyset \in$ SAT, i.e., the empty clause set is satisfiable.

A graph is planar if it can be embedded in the plane without edge crossings. An embedding of a graph G is *1-outerplanar* (*outerplanar* for short), if it is planar, and all vertices lie on the exterior face. A graph is called *outerplanar* if it has an outerplanar embedding.

III. Structural Properties Of Outerplanar Formulas

In the sequel, C is a formula with formula graph G_C. For simplicity, variables and the corresponding vertices are identified the same holds for clauses. As usual we call an edge inside an outerplanar circle a *chord*. An outerplanar circle with q variables and q clauses, obviously has at most $q-2$ chords. According to [7], a formula C is called *matched* if we have $|C'| \leq |V(C')|$, for every subformula $C' \subseteq C$. In [7] it is shown that matched formulas are always satisfiable by applying the theorem of Koenig and Hall [9], [11] to the bipartite incidence graph of the formula, where variables, respectively clauses, form the partition of the vertex set. As one can easily see, an outerplanar formula whose formula graph consists of paths (each two of which are allowed to share a vertex), disjoint circles and single vertices is a matched formula. A model for a matched formula C of n variables can be determined in time $O(n\|C\|)$ where $\|C\|$ denotes the size, i.e., length of C. In this paper we even provide an algorithm for determining a model for every outerplanar formula in linear time. Recall that a backbone

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

variable x of a formula C has the same value in every model of C. Let K be an outerplanar circle with chords. Omitting the chords we obtain the annulus of K which we denote by K^R. Let K_s be a circle then we denote the corresponding formula by C_s , the annulus of K_s by K_s^R and its corresponding 2-CNF formula by C_s^R. By the satisfiability of a circle K_s we mean the satisfiability of C_s. For simplicity, we shall often identify K_s with C_s as well as K_s^R with C_s^R. The next lemma states that a backbone variable of a circle also is a backbone variable of the corresponding annulus.

Lemma 1: Let C_s be a circle of a formula graph G_C that can have chords, then any backbone variable of C_s is also a backbone variable of C_s^R.

PROOF. For every clause c^R of C_s^R, there is exactly one clause c of C_s such that $V(c^R) \subseteq V(c)$, because C_s^R results from C_s by shortening clauses. Thus a model of C_s^R obviously is a model of C_s, too. Assume x is a backbone variable of C_s but not of C_s^R, then there are models of C_s^R in which x is set to 1 and models assigning x to 0. Since every model of C_s^R also satisfies C_s we obtain a contradiction. □

A characterization of a backbone variable in an annulus is given next.

Theorem 1: . Let C_s be a circle of G_C consisting of the variables $x_0, \ldots, x_{k-1}$ and the clauses $c_0, \ldots, c_{k-1}$. Then x_0 is a backbone variable of the 2-CNF formula C_s^R if and only if an implicational chain of the form

$$x_0 \rightarrow l(x_1) \rightarrow l(x_2) \rightarrow \cdots \rightarrow l(x_{k-1}) \rightarrow \overline{x_0}$$

or

$$\overline{x_0} \rightarrow l(x_1) \rightarrow l(x_2) \rightarrow \cdots \rightarrow l(x_{k-1}) \rightarrow x_0$$

holds for C_s^R. Moreover C_s^R has at most one backbone variable.

PROOF. Let $x_0, c_0, x_1, c_1, \ldots, x_{k-1}, c_{k-1}$ be the vertices of C_s^R in clockwise orientation. For the clauses, we have $V(c_i) = \{x_i, x_{i+1}\} \mod k$, $0 \leq i \leq k-1$. Clearly $\{x_i, x_{i+1}\}$ corresponds to an implication $\overline{x_i} \rightarrow x_{i+1}$ or $\overline{x_{i+1}} \rightarrow x_i$. If we can build a single implicational chain, e.g. $x_0 \rightarrow \cdots \rightarrow \overline{x_0}$, of all k clause-implications, then x_0 is a backbone variable which must be set to 0. For the reverse direction, assume that x_0 is a backbone variable of C_s^R but none of the implicational chains $\overline{x_0} \rightarrow \cdots \rightarrow x_0$ or $x_0 \rightarrow \cdots \rightarrow \overline{x_0}$ holds for C_s^R. Hence there is a variable x_i yielding chains that cannot be combined, e.g. $x_0 \rightarrow \cdots \rightarrow x_i$ and $\overline{x_i} \rightarrow \cdots \rightarrow \overline{x_0}$ to a single implicational chain. Then one easily observes that there are models for both values of x_1.

Finally, let x_1 be a backbone variable of C_s^R for value 0 and implicational chain $x_1 \rightarrow \cdots \rightarrow \overline{x_1}$. Suppose x is another backbone variable of C_s^R, then we have the implicational chains $x_1 \rightarrow \cdots \rightarrow x$ and $x \rightarrow \cdots \rightarrow \overline{x_1}$ which cannot be combined. Hence x cannot be a backbone variable contradicting the assumption. □

In the following we write $x_i \in_+ c_j$ (respectively $x_i \in_- c_j$) if the vertices x_i and c_j form a chord and x_i occurs positive (respectively negative) in c_j.

Theorem 2: Let x_1 be a backbone variable of C_s^R, for which the implicational chain $\overline{x_1} \rightarrow x_2 \rightarrow \cdots \rightarrow x_n \rightarrow x_1$ is assumed to hold. If none of the following *chord criteria* holds, then x_1 also is a backbone variable of the corresponding C_s, otherwise it is no backbone variable of C_s.

(1) C_s has chord $x_i \in_+ c_j$, for $1 \leq i < j \leq n$.

(2) C_s has chord $x_i \in_- c_j$, for $1 \leq j < i \leq n$.

(3) C_s has chord $x_1 \in_- c_j$, for $1 < j \leq n$.

PROOF. By assumption, we have $x_1 = 1$ in any model of C_s^R each of which yields a model of C_s too. In the absence of chords we have $C_s^R = C_s$, hence x_1 also is a backbone variable of C_s. Otherwise, if C_s has a chord of type (1) above, we have a model of C_s where $x_1 = 0$ and $x_k = 1$, for $1 < k \leq j$, $x_k = 0$, for $j < k \leq n$. Similarly one obtains models where $x_1 = 0$ in case of the presence of a chord of type (2) or (3) in C_s. □

Let C be an outerplanar formula and let C_1 and C_2 correspond to circles of its formula graph which are assumed to share the variable x only. Let x be a backbone 1 variable of C_1^R and a backbone 0 variable of C_2^R. Then C is unsatisfiable if x remains a backbone variable of C_1 as well as of C_2. We call such a situation a *backbone conflict*. We call a connected component of an outerplanar graph a *circle-component* if the component only consists of circles which either are pairwise disjoint, or share a single vertex only, or are joined by a path. For each circle-component $\mathcal{K}$ we can create a superstructure graph $S(\mathcal{K})$ as follows: The vertices of $S(\mathcal{K})$ correspond to the circles of $\mathcal{K}$. If there is a circle which is joined to another circle by a path in $\mathcal{K}$ then both circles are joined by an edge in $S(\mathcal{K})$. If two circles share a single variable (respectively clause) vertex x (or c) then an edge in $S(\mathcal{K})$ is introduced labelled with x (or c). Note that because C is outerplanar the superstructure graph is a tree always.

IV. THE MAIN ALGORITHM

Now we present an algorithm solving SAT for outerplanar formulas, and shall show below that it runs in linear time.

Main Algorithm:

INPUT: An outerplanar formula C with formula graph G_C.

OUTPUT: A model if C is satisfiable and UNSATISFIABLE, else.

1) If C is a 2-CNF formula, solve C by an appropriate linear-time algorithm [1].
2) Determine the outerplanar formula graph G_C of C.
3) As long as there is a variable joined to only one clause, set the variable such that the clause is satisfied.
4) As long as there is a clause adjacent to only one variable, set the variable such that the clause is satisfied.
5) As long as G_C has disjoint circles (with possible chords), for each circle with vertices $x_1, c_1, x_2, c_2, \ldots x_n, c_n, x_1$ in clockwise orientation, assign x_i satisfying c_i, $1 \leq i \leq n$.
6) If G_C consists of circles $K_1, \ldots, K_m$ only (there are no paths or other outerplanar objects in G_C) and each two circles are allowed to share at most one vertex, then C is satisfiable, and a model is determined as follows: For each circle, do: Let $x_1 - c_1 - x_2 - c_2 - \cdots - x_n - c_n - x_1$ be its vertices in clockwise direction. Set x_k such that c_k, $1 \leq k \leq n$, is satisfied.
7) If the remaining graph G_C consists of several circle-components, solve each circle-component $\mathcal{K}$ by Procedure Circle-Component($\mathcal{K}$).
8) If Procedure Circle-Component($\mathcal{K}$) returns a model for every circle-component of G_C, then the algorithm returns the combination of these models for C. Otherwise it returns UNSATISFIABLE.

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Next, we state in detail

Procedure Circle-Component($\mathcal{K}$):
INPUT: A circle-component $\mathcal{K}$ of G_C.
OUTPUT: A model, if (the formula corresponding to) $\mathcal{K}$ is satisfiable, UNSATISFIABLE, else.

1) Determine the superstructure tree $S(\mathcal{K})$ of $\mathcal{K}$.
2) For each circle in $\mathcal{K}$, do:
   Write each 2-clause on its annulus with variables x_i, x_{i+1} as the corresponding implication $l(x_i) \rightarrow l(x_{i+1})$ or $l(x_{i+1}) \rightarrow l(x_i)$ and check whether a single implicational chain of all these implications can be build. In the positive case, the backbone variable of the annulus according to Theorem 1 is found.
3) For each circle whose annulus admits a backbone variable, check according to Theorem 2 whether it remains a backbone variable of the whole circle.
4) Assign all backbone variables of $\mathcal{K}$ appropriately.
5) If there are two circles in $\mathcal{K}$ with a backbone conflict then return UNSATISFIABLE.
6) Start with considering the leaves of $S(\mathcal{K})$. As soon as all leaves are solved by Procedure Circle(), solve the parent of every leaf by Procedure Circle(). In general, a vertex is not solved until all its children in $S(\mathcal{K})$ are solved by Procedure Circle(). So beginning with the leaves in $S(\mathcal{K})$ and as long as the root of the tree is not reached, apply the following steps:
   - Let K_s be the current vertex in $S(\mathcal{K})$. If K_s is joined to its parent $F(K_s)$ in $S(\mathcal{K})$ by an edge labelled with x, we proceed as follows:
     If x is already assigned (e.g as a possible backbone variable of $F(K_s)$), then by Procedure Circle (K_s) check whether K_s can be satisfied by this value of x.
     (a) If Circle (K_s) returns that K_s is unsatisfiable, then the corresponding formula of $\mathcal{K}$ is unsatisfiable and the procedure returns UNSATISFIABLE.
     (b) If Circle (K_s) returns a model for K_s, under consideration of the assignment of x, then set all variables of K_s according to that model. After having solved all the sibling-vertices of K_s solve $F(K_s)$, the parent vertex of K_s.

     If x is not assigned yet, by Procedure Circle($K_s(x_1 = t)$) check whether K_s can be satisfied for $x_1 = t$, $t \in \{0, 1\}$:
     (a) If K_s can be satisfied for $x = 1$ as well as for $x = 0$, do not fix x directly but first consider all the sibling-vertices of K_s before considering the parent $F(K_s)$ of K_s.
     (b) If w.l.o.g. K_s can only be satisfied for $x = 1$, assign $x = 1$, solve all the sibling-vertices of K_s and afterwards solve the parent $F(K_s)$ of K_s. Note that x already is fixed to 1 when $F(K_s)$ is considered.
     (c) If K_s cannot be satisfied for neither $x = 1$ nor $x = 0$, the formula corresponding to $\mathcal{K}$ is unsatisfiable and the procedure returns UNSATISFIABLE.
   - Let K_s be the current vertex of $S(\mathcal{K})$. If K_s is joined to its parent vertex $F(K_s)$ in $S(\mathcal{K})$ by an edge labelled with the clause c, then proceed as follows: By calling Procedure Circle(K_s) check whether K_s is satisfiable: If yes, particularly c is satisfied, hence we label c as satisfied in $F(K_s)$. If not, we consider $F(K_s)$ after having applied Procedure Circle() on all its sibling-vertices before.
   - Let K_s be the current vertex of $S(\mathcal{K})$. If K_s is joined to its parent vertex $F(K_s)$ in $S(\mathcal{K})$ by an edge corresponding to a path (of length at least 2) in $\mathcal{K}$, then proceed as follows:
     – If K_s and $F(K_s)$ are joined by the path $P = x_1 - c_1 - x_2 - c_2 - \cdots - x_n - c_n$ sharing x_1 with K_s and c_n with $F(K_s)$, then proceed as follows:
     By Procedure Circle($K_s(x_1 = t)$) check whether K_s can be satisfied for $x_1 = t$, $t \in \{0, 1\}$. If K_s can be satisfied for $x_1 = 1$ as well as for $x_1 = 0$, set x_1 such that c_1 is satisfied. Further set x_2 such that c_2 is satisfied. In general, set x_i according to satisfy c_i, $1 \leq i \leq n$. It follows that c_n now can be labelled as satisfied in $F(K_s)$.
     If w.l.o.g. K_s is satisfiable only for $x_1 = 1$, check whether $x_1 = 1$ also satisfies c_1. If so, assign x_i according to satisfy c_i of path P, $2 \leq i \leq n$. Therefore, c_n can be labelled as satisfied in $F(K_s)$. Otherwise, i.e., c_1 cannot be satisfied by $x_1 = 1$, then assign x_{i+1} according to satisfy c_i, $1 \leq i \leq n-1$. If x_n occurs with the same polarity in c_n as in c_{n-1}, then label c_n as satisfied in $F(K_s)$.
     Finally, if K_s cannot be satisfied for neither $x_1 = 0$ nor $x_1 = 1$, then the procedure returns UNSATISFIABLE.
     – If K_s and $F(K_s)$ are joined by a path $P = x_1 - c_1 - x_2 - c_2 - \cdots - x_{n-1} - c_{n-1} - x_n$ such that P shares x_1 with K_s and x_n with $F(K_s)$, then proceed as follows:
     By calling Procedure Circle($K_s(x_1 = t)$) check whether K_s can be satisfied for $x_1 = t$, $t \in \{0, 1\}$. If K_s can be satisfied for $x_1 = 1$ as well as for $x_1 = 0$, set x_1 such that it satisfies c_1. Furthermore, set x_i such that it satisfies c_i, $1 \leq i \leq n-1$. Note that x_n remains unassigned.
     If w.l.o.g. K_s is only satisfiable for $x_1 = 1$, set $x_1 = 1$ and check whether $x_1 = 1$ also satisfies c_1. If so, set x_i such that it satisfies c_i, $2 \leq i \leq n-1$. But if $x_1 = 1$ does not satisfy c_1, set x_{i+1} such that it satisfies c_i, $1 \leq i \leq n-1$. Now x_n has a fixed value in $F(K_s)$.
     If K_s cannot be satisfied for neither $x_1 = 1$ nor $x_1 = 0$, the formula corresponding to $\mathcal{K}$ is unsatisfiable and the procedure returns UNSATISFIABLE.
     – If K_s and $F(K_s)$ are joined by the path $P = c_1 - x_1 - \cdots - c_{n-1} - x_{n-1} - c_n$ which shares clause c_1 with K_s and the clause c_n with $F(K_s)$, we proceed as follows:
     By Procedure Circle(K_s) check whether a model for K_s exists. If K_s is satisfiable, set

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

all the variables of K_s according to the model. Further set x_1 such c_2 is satisfied. In general, set x_i such that c_{i+1}, is satisfied $1 \leq i \leq n-1$. As a consequence label c_n as satisfied in $F(K_s)$.
If Procedure Circle(K_s) returns UNSATISFIABLE, set x_1 such that it satisfies c_1. Further check whether this assignment satisfies c_2. If so, further set x_i such that it satisfies c_{i+1}, for $2 \leq i \leq n-1$. Otherwise set x_2 appropriately. Next, check whether this assignment of x_2 also satisfies c_3. If so, then set x_i such that c_{i+1}, is satisfied for $3 \leq i \leq n-1$. Otherwise, set x_3 satisfying c_3 and so forth. If x_{n-1} occurs with the same polarity in c_n as in c_{n-1} then c_n is also satisfied by the assignment of x_{n-1} and label c_n as satisfied in $F(K_s)$. Otherwise $F(K_s)$ remains unmodified by the path.

– If K_s and $F(K_s)$ are joined by the path $P = c_1 - x_1 - c_2 - x_2 - \cdots - c_n - x_n$ sharing c_1 with K_s and x_n with $F(K_s)$ then proceed as follows:
By Procedure Circle(K_s) check whether K_s is satisfiable. If so, determine a model for K_s and set all variables of K_s according to this model. Further set x_1 according to satisfy c_2. Afterwards set x_2 such that it satisfies c_3 and generally set x_i such that it satisfies c_{i+1}, $1 \leq i \leq n-1$. Note that this does not influence x_n which P shares with $F(K_s)$ and so x_n remains unassigned.
If Circle(K_s) returns UNSATISFIABLE, set x_1 such that c_1 is satisfied. Next check whether c_2 also is satisfied by this assignment. If so, further set x_i such that it satisfies c_{i+1}, for $2 \leq i \leq n$. Otherwise, set x_2 such that c_2 is satisfied, and check whether this assignment also satisfies c_3. If so, further set x_i such that it satisfies c_{i+1}, $3 \leq i \leq n-1$. Else, set x_3 such that c_3 is satisfied and so forth until c_n is satisfied.

- As soon as the root vertex $R_{\mathcal{K}}$ of $S(\mathcal{K})$ is reached, by Procedure Circle($R_{\mathcal{K}}$) check whether $R_{\mathcal{K}}$ is satisfiable.
  - If $R_{\mathcal{K}}$ is unsatisfiable, the formula corresponding to $\mathcal{K}$ is not satisfiable, either, and the procedure returns UNSATISFIABLE.
  - If $R_{\mathcal{K}}$ is satisfiable, set all its variables accordingly. If all other vertices of $S(\mathcal{K})$ are satisfiable and a model for each is found, the procedure returns the model for $\mathcal{K}$.
  - If $R_{\mathcal{K}}$ is satisfiable and there is a vertex K_t in $S(\mathcal{K})$ for which a model is not found yet, proceed as follows for all these vertices: (Note that we only have to solve those vertices K_t for which we have not fixed a model yet in the first passage through the tree from the leaves up to the root of the tree) Beginning with the root vertex, traverse $S(\mathcal{K})$ until all vertices are solved or a vertex in $S(\mathcal{K})$ is found that cannot be satisfied. In the latter case the procedure returns UNSATISFIABLE. Here, a vertex K_t is treated not before a model for its parent $F(K_t)$ is found according to the following cases:
    If K_t is joined to $F(K_t)$ by an edge labelled with a variable x, then x now is fixed in K_t because of $F(K_t)$, w.l.o.g. let $x = 0$. As K_t can be satisfied for $x = 0$ as well as for $x = 1$, we set all variables of K_t according to the model which fixes $x = 0$.
    If K_t is joined to $F(K_t)$ by an edge labelled with c, call Procedure Circle($K_t \setminus \{c\}$), checking for a model of $K_t \setminus \{c\}$. Consider the next vertex in $S(\mathcal{K})$ which has not been treated so far.

Finally, it remains to formulate
Procedure Circle(K_s):
INPUT: A circle K_s with possible chords and possibly fixed variables.
OUTPUT: A model if the formula corresponding to K_s is satisfiable and UNSATISFIABLE, else.

1) All clauses which are labelled as satisfied are invisible for the procedure and not considered here.
2) As long as there is a variable x_i in K_s which is contained in only one clause c_j, set x_i such that it satisfies c_j.
3) As long as there is a clause c_j in K_s containing only one variable x_i, set x_i such that it satisfies c_j.
4) If the 2-CNF formula corresponding to the annulus K_s^R of K_s is satisfiable, then K_s is also satisfiable, and a model of the annulus is returned.
5) First, fix an order for the vertices of K_s^R and initialize an appropriate container $M(S)$ for each chord $S = x_i \in_{\pm} c_j$ for which x_i is not assigned yet. This container stores the following information: Let $x_m \to \cdots \to l(x_i)$ resp. $l(x_i) \to \cdots \to x_p$ be the two possible implicational chains for x_i, then the container holds whether there is a variable fixed to 1 in the first, respectively fixed to 0 in the second chain.
6) If the 2-CNF formula corresponding to K_s^R is unsatisfiable, for K_s^R one of the two following cases holds true:
   (a) The 2-CNF formula contains a smallest false implicational chain e.g. $x_k \to \cdots \to x_{k+l}$ where $x_k = 1, x_{k+l} = 0$ is already fixed and $x_{k+l} \neq x_k$. In order to test whether K^i can be satisfied when taking the chords into consideration, the following cases have to be distinguished (for more than one chains like this, the procedure can be adapted easily):
      (i) There is no chord of the form $x_i \in_{\pm} c_j$, with $k \leq j \leq k+l$. In this case K_s is unsatisfiable and the procedure returns UNSATISFIABLE.
      (ii) There is a chord of the form $x_i \in_{+} c_j$, with $k \leq i < j \leq k+l$. In this case the following assignment enables a model for K_s: $x_k = \ldots = x_j = 1$ and $x_{j+1} = \ldots = x_{k+l} = 0$.
      (iii) There is a chord of the form $x_i \in_{-} c_j$, with $k \leq j < i \leq k+l$. In this case the following assignment enables a model for K_s : $x_k = \ldots = x_j = 1$ and $x_{j+1} = \ldots = x_{k+l} = 0$

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

(iv) There is a chord of the form $x_i \in_{\pm} c_j$, with $i < k$ or $i > k+l$ and $k \leq j \leq k+l$ and x_i is already set to 1 (resp. to 0). Then the following assignment enables a model for K_s: $x_k = \ldots = x_j = 1$ and $x_{j+1} = \ldots = x_{k+l} = 0$.

(v) There is a chord of the form $S = x_i \in_{\pm} c_j$, with $i < k$ or $i > k+l$ and $k \leq j \leq k+l$ and x_i is not assigned yet. Then we proceed as follows:

- If there is only one such chord, we set x_i such that c_j is satisfied, then $x_k = \ldots = x_j = 1, x_{j+1} = \ldots = x_{k+l} = 0$ satisfies the clauses $c_k, \ldots, c_{k+l}$. Next we check with help of the container $M(S)$ whether the current assignment of x_i yields a false implicational chain $1 \rightarrow 0$ due to the following cases:
  The implicational chain $x_m \rightarrow \cdots \rightarrow l(x_i)$ already contains a variable set to 1, $l(x_i) = x_i$ and $x_i = 0$.
  The implicational chain $x_m \rightarrow \cdots \rightarrow l(x_i)$ already contains a variable set to 1, $l(x_i) = \overline{x_i}$ and $x_i = 1$.
  The implicational chain $l(x_i) \rightarrow \cdots \rightarrow x_p$ already contains a variable set to 0, $l(x_i) = x_i$ and $x_i = 1$.
  The implicational chain $l(x_i) \rightarrow \cdots \rightarrow x_p$ already contains a variable set to 0, $l(x_i) = \overline{x_i}$ and $x_i = 0$.
  If one of the four cases above occurs, the remaining 2-CNF formula cannot be satisfied apart from $x_k \rightarrow \cdots \rightarrow x_{k+l}$ because with the current assignment of x_i we obtain an additional false implicational chain. If we can satisfy this chain applying 6(a), then K_s is satisfiable, else unsatisfiable.
- If there are several chords $S = x_i \in_{\pm} c_j$ for which holds $i < k$ resp. $i > k+l$, $k \leq j \leq k+l$ and x_i is not fixed yet, we perform the following for each such chord S as long as we have not found a model for K_s:
  Set $x_i = 1$ (resp. $x_i = 0$) and $x_k = \ldots = x_j = 1$, $x_{j+1} = \ldots = x_{k+l} = 0$. Then by means of the container $M(S)$ check whether we obtain a false implicational chain $1 \rightarrow 0$ by the assignment of x_i, similarly, as above:
  The implicational chain $x_m \rightarrow \cdots \rightarrow l(x_i)$ already contains a variable set to 1, $l(x_i) = x_i$ and $x_i = 0$.
  The implicational chain $x_m \rightarrow \cdots \rightarrow l(x_i)$ already contains a variable set to 1, $l(x_i) = x_i$ and $x_i = 1$.
  The implicational chain $l(x_i) \rightarrow \cdots \rightarrow x_p$ already contains a variable set to 0, $l(x_i) = x_i$ and $x_i = 1$.
  The implicational chain $l(x_i) \rightarrow \cdots \rightarrow x_p$ already contains a variable set to 0, $l(x_i) = x_i$ and $x_i = 0$.
  If one of these cases occurs, the current assignment of x_i yields a false implicational chain. If this can be satisfied by applying 6(a), then K_s is satisfiable. If not, undo (release) the assignment of $x_k, \ldots, x_{k+l}$ and test whether there is another chord $x_i \in_{\pm} c_j$, with $i < k$ resp. $i > k+l$ and $k \leq j \leq k+l$ where x_i is not fixed yet, yielding a model for K_s.
- If we have considered all the chords $x_i \in_{\pm} c_j$ with $i < k$ resp. $i > k+l$ and $k \leq j \leq k+l$, where x_i is not fixed yet, and have not found a model for K_s, then K_s is unsatisfiable and the procedure returns UNSATISFIABLE.

(b) If K_s^R corresponds to $\overline{x_1} \rightarrow \cdots \rightarrow x_1$ (resp. $x_1 \rightarrow \cdots \rightarrow \overline{x_1}$), $x_1 = 0$ (resp. $x_1 = 1$) is already fixed and there are no other smaller false implicational chains on K_s^R, then solve K_s as follows (w.l.o.g. let $\overline{x_1} \rightarrow \cdots \rightarrow x_1$ and $x_1 = 0$ be fixed): If K_s has no chords, it is unsatisfiable and the procedure returns UNSATISFIABLE. Else, check whether there is a chord of the form $x_i \in_{\pm} c_j$ in K_s:

(i) Let $x_i \in_{+} c_j$ be a chord of K_s, with $i < j$, where $x_i = 1$ is already fixed. Then c_j is satisfied by x_i and as there are no false implicational chains valid for $K_s^R \setminus \{c_j\}$, the remaining 2-CNF formula $K_s^R \setminus \{c_j\}$ is satisfiable. Solve this 2-CNF formula by a linear-time algorithm.

(ii) Let $x_i \in_{-} c_j$ be a chord, with $i > j$ and $x_i = 0$ is already fixed. Then c_j is satisfied according to the assignment of x_i and as there are no false implicational chains for $K_s^R \setminus \{c_j\}$ the remaining 2-CNF formula $K_s^R \setminus \{c_j\}$ is satisfiable. Solve the corresponding 2-CNF formula by a linear-time algorithm.

(iii) Let $x_1 \in_{-} c_j$ be the chord, with $j > 1$. Then c_j is satisfied by the assignment of x_1 and as there are no false implicational chains for $K_s^R \setminus \{c_j\}$ the remaining 2-CNF formula $K_s^R \setminus \{c_j\}$ is satisfiable. Solve the corresponding 2-CNF formula by a linear-time algorithm.

(iv) If there is only one chord $x_i \in_{\pm} c_j$ with $i < j$ $(i > j)$ where x_i is not fixed, set x_i such that c_j is satisfied. If there are several chords of this form, perform the following for each such chord:
Assign x_i according to satisfy c_j and check whether this assignment satisfies the remaining 2-CNF formula $K_s^R \setminus \{c_j\}$. If so, a model for K_s is found. Otherwise, there is a false implicational chain, e.g. $x_i \rightarrow \cdots \rightarrow x_k$, with $x_i = 1$, $x_k = 0$ or $x_k \rightarrow \cdots \rightarrow \overline{x_i}$, with $x_i = 1$, $x_k = 1$. Check whether it can be solved by applying 6(a). If so, a model is found. Else, consider the next chord which satisfies (iv). If all such chords are considered and still no model is found, then

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

K_s is unsatisfiable and the procedure returns UNSATISFIABLE.

The correctness of our algorithm follows from the presentation above.

V. Analysis Of The Running Time

The running time of our algorithm in fact is linear, which is proved according to the following argumentation. If G_C consists of disjoint circles, circles which share a clause vertex only, or paths which may have a vertex in common (we assume that C has no unit clauses), then C is a matched formula and thus satisfiable. Steps 1) to 6) can obviously be performed in linear time.

Next consider Procedure Circle(K_s): Let K_s be a circle with q clauses and q variables. Then K_s has at most $q-2$ chords. For solving the 2-CNF formula corresponding to the annulus, we need $O(q)$ time. If this 2-CNF is satisfiable, a model for it also is a model for the formula of K_s. If the 2-CNF formula is unsatisfiable as a result of a false implicational chain, e.g., $x_k \rightarrow \cdots \rightarrow x_{k+l}$ on the annulus K_s^R with $x_k = 1, x_{k+l} = 0$ already fixed and $x_{k+l} \neq x_k$, then check whether there is a chord for which one of the cases (ii)-(iv) of 6(a) in Circle(K_s) is valid. If not, check for all chords $x_i \in_{\pm} c_j$ with the property of case (v) in 6(a) whether by setting $x_i = 1$ (resp. $x_i = 0$) a model for K_s can be provided. As soon as such a chord providing a model for K_s is found, the procedure outputs this model. If K_s is unsatisfiable, perform this process recursively until we have considered all the chords with the property of case (v) in 6(a) and have ascertained that none of them provides a model for K_s. Then the procedure stops with output UNSATISFIABLE. Hence every chord is considered at most once. Using containers constant time is consumed to check for each chord whether by the assignment of the chord variable a false implicational chain on the annulus is obtained. Since K_s has at most $q-2$ chords we need $O(q)$ running time to solve a circle with q variables and q clauses. Case 6(b) of Procedure Circle() also yields a running time of $O(q)$ because in worst-case we have to check all chords.

Regarding Procedure Circle-Component(), observe that for each circle-component $\mathcal{K}$ the superstructure tree $S(\mathcal{K})$ is traversed at most twice: first bottom-up from the leaves to the root. In case the root circle is satisfiable and but there are still vertices in $S(\mathcal{K})$ for which no model is determined, we traverse top-down from the root to the leaves until either models are found for all circles of $\mathcal{K}$ or an unsatisfiable vertex in $S(\mathcal{K})$ is determined. It follows that Procedure Circle() is called at most twice for every vertex of $S(\mathcal{K})$. As a circle of q variables and q clauses can be solved in time $O(q)$ time, the time complexity of our algorithm for an outerplanar formula C of n variables is $O(n)$.

VI. Concluding Remarks

We proposed a linear-time algorithm for outerplanar formulas that explicitly exploits the outerplanar structure of the underlying formula graph. Also the counting problem #SAT can be solved for this class in polynomial-time, which shows an algorithm that is based on the separator theorem by Lipton and Tarjan [14] and that is described in [15]. In [15] also a treatment using the nice tree decomposition [2], [3], [4], [5] is provided, for both the search and counting versions.

References

[1] Aspvall, B., Plass, M.R., Tarjan, R.E, A linear-time algorithm for testing the truth of certain quantified Boolean formulas, Inform. Process. Lett. 8 (1979) 121-123.
[2] Bodlaender, H.L., Kloks, T., Efficient and constructive algorithms for the pathwidth and treewidth of graphs, Journal of Algorithms 21 (1996) 358-402.
[3] Bodlaender, H.L., Treewidth: Algorithmic techniques and results, Proceedings 22nd MFCS, Springer-Verlag LNCS 1295 (1997) 19-36.
[4] Bodlaender, H.L., A partial k-arboretum of graphs with bounded treewidth, Theoretical Computer Science 209 (1998) 46-52.
[5] Bodlaender, H.L., Fomin, F.V., Tree decompositions with small cost, Proceedings 8th SWAT, Springer-Verlag LNCS 2368 (2002) 378-387.
[6] Cook, S.A., The Complexity of Theorem Proving Procedures. In: Proceedings of the 3rd ACM Symposium on Theory of Computing, pp. 151-158, 1971.
[7] Franco, J., VanGelder, A., A perspective on certain polynomial-time solvable classes of satisfiability, Discrete Appl. Math. 125 (2003) 177-214.
[8] Gu, J., Purdom, P.W., Franco, J., Wah, B.W., Algorithms for the Satisfiability (SAT) Problem: A Survey, in: D. Du, J. Gu, P. M. Pardalos (Eds.), Satisfiability Problem: Theory and Applications, DIMACS Workshop, March 11-13, 1996, DIMACS Series, vol. 35, pp. 19-151, American Mathematical Society, Providence, Rhode Island, 1997.
[9] Hall, P., On representatives of subsets, J. London Math. Soc. 10 (1935) 26-30.
[10] Knuth, D.E., Nested satisfiability, Acta Informatica 28 (1990) 1-6.
[11] König, D., Graphen und Matrizen, Math. Fiz. Lapok 38 (1931) 116-119.
[12] Le Berre, D., Simon, L., The Essentials of the SAT 2003 Competition, in: Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT'03), Springer-Verlag LNCS 2919 (2004) 172-187.
[13] Lichtenstein, D., Planar formulae and their uses, SIAM Journal on Computing 11 (1982) 329-343.
[14] Lipton, R.J., Tarjan R.E., A separator theorem for planar graphs, SIAM J. Appl. Math. 36 (1979) 177-189.
[15] Schmidt, T., Computational complexity of SAT, XSAT and NAE-SAT for linear and mixed Horn CNF formulas, dissertation, Univ. Köln, 2010.
[16] Speckenmeyer, E., Min Li, C., Manquinho V., Tacchella, A., (Eds.), Special Issue on the 2007 Competitions, Journal on Satisfiability, Boolean Modeling and Computation, 4 (2008).

ISBN: 978-988-19251-8-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)