Propositional Encodings of Acyclicity and Reachability by Using Vertex Elimination

Masood Feyzbakhsh Rankooh and Jussi Rintanen

Department of Computer Science, Aalto University, Helsinki, Finland
jussi.rintanen@aalto.fi, masood.feyzbakhshrankooh@aalto.fi

Abstract

We introduce novel methods for encoding acyclicity and s-t-reachability constraints for propositional formulas with underlying directed graphs. They are based on *vertex elimination graphs*, which makes them suitable for cases where the underlying graph has a low *directed elimination width*. In contrast to solvers with ad hoc constraint propagators for acyclicity and reachability constraints such as GraphSAT, our methods encode these constraints as standard propositional clauses, making them directly applicable with any SAT solver. An empirical study demonstrates that our methods together with an efficient SAT solver do often outperform both earlier encodings of these constraints as well as GraphSAT, particularly when underlying graphs have low directed elimination width.

Introduction

Many AI approaches incorporate graphs to maintain conceptual relations among their elements. Graphs introduce structure to AI methods. Once such a structure has been assumed, investigating the existence and exploitation of structural properties is only natural. Reachability and acyclicity are two of the most important structural properties of graphs.

Graph constraints are important in knowledge representation languages. For example, acyclicity constraints are part of reductions of Answer Set Programming to SAT (Lin and Zhao 2004; Gebser, Janhunen, and Rintanen 2014a), and implicit in fixpoint semantics of inductive definitions (Denecker and Ternovska 2008). In AI planning, acyclicity is needed for SAT encodings for classical planning that use partial orders (Rintanen, Heljanko, and Niemelä 2006), and for non-deterministic and partially observable planning (Chatterjee, Chmelik, and Davies 2016; Pandey and Rintanen 2018). Moreover, constraint-based methods for structure learning of Bayesian networks need the acyclicity of the networks with graph constraints (Cussens 2008).

The above-mentioned approaches have motivated the development of better encodings of acyclicity and other graph constraints in the propositional logic, as well as the study of specialized propagators for these constraints.

In this work we address the satisfiability of propositional formulas with underlying directed graphs, under reachability and acyclicity constraints. The motivation for our work is the difficult trade-off between size and propagation strength in existing clausal encodings of these constraints (Gebser, Janhunen, and Rintanen 2020) on one hand, and the effort in implementing specialized graph constraint propagators (Gebser, Janhunen, and Rintanen 2014b), and adapting and embedding them in new SAT solvers as ones become available, on the other.

Our goal is to develop encoding methods for graph constraints such as acyclicity and reachability, that are competitive with specialized ad hoc graph constraint propagators, and which suffer less from the large size of those traditional clausal encodings that have good propagation properties. We particularly address sparse graphs.

Our idea is to use vertex elimination graphs (Rose and Tarjan 1975) as a structure that preserves reachability and acyclicity properties of the underlying graph, and also allows succinct encoding of graph constraints into propositional formulas, particularly when a related graph measure named *directed elimination width* (Hunter and Kreutzer 2007) is low in comparison to number of vertices. The current state-of-the-art method for satisfying acyclicity and reachability constraints in the SAT context is GraphSAT (Gebser, Janhunen, and Rintanen 2014b). While GraphSAT relies on a specialized algorithm for satisfying graph constraints, our methods explicitly encode the constraints into propositional formulas, and therefore allow an easy reuse of the method with any other state-of-the-art SAT solver without additional implementation effort.

We provide the theoretical arguments for correctness of our methods, and also, deliver theoretical evidence for efficiency of the methods by undertaking a parameterized complexity analysis. Moreover, our empirical results show that by employing an efficient SAT solver, our new methods can outperform GraphSAT and other encoding methods, particularly when underlying graphs have low directed elimination width.

Preliminaries

In this section we provide formal definitions for propositional formulas with underlying directed graphs, encoding of graph constraints, and vertex elimination graphs.

Propositional Formulas with Underlying Directed Graphs

Assume that ϕ is a propositional formula over the set of propositional variables X. Let $G = (V, E)$ be a digraph and f be a partial function from X to E. Then we can consider ϕ as a propositional formula with underlying digraph G with respect to f. In this work, we avoid explicit definition of f by denoting the proposition in ϕ that is mapped by f to $(u, v) \in E$ by $e_{u,v}$. If there exists a model $\mathcal{M}$ for ϕ, we construct $G_{\mathcal{M}} = (V, E_{\mathcal{M}})$, the underlying graph of $\mathcal{M}$, where $E_{\mathcal{M}} = \{(v_i, v_j) | \mathcal{M}(e_{i,j}) = true\}$.

The main purpose for considering a propositional formula with an underlying directed graph is to add semantics to the formula by enforcing certain constraints on the underlying graph of the models found for the formula. One way to do this is producing the conjunction of original formula and additional formulas.

Definition 1 (Encoding of Graph Constraints)**.** *Let ϕ be a propositional formula with underlying graph G. The encoding of graph constraint $\mathcal{C}$ for ϕ is a propositional formula $\phi_{\mathcal{C}}$ with completeness and soundness properties stated below:*

- *(Completeness) if ϕ is satisfied by model $\mathcal{M}$ such that constraint $\mathcal{C}$ holds for $G_{\mathcal{M}}$, then $\phi \wedge \phi_{\mathcal{C}}$ is satisfiable.*
- *(Soundness) if $\phi \wedge \phi_{\mathcal{C}}$ is satisfied by model $\mathcal{M}$, then constraint $\mathcal{C}$ holds for $G_{\mathcal{M}}$.*

Encoding of *acyclicity*, *s-t-reachability*, *s-t-unreachability*, and *s-t-eventual-reachability* constraints are of particular interest in this work. We define these constraints below, where we survey background research.

Vertex Elimination Graphs

The concept of vertex elimination graph has originally been introduced in (Rose and Tarjan 1975). Let $G = (V, E)$ be a directed graph, $G^+ = (V, E^+)$ be the transitive closure of G, and $O = v_1, ..., v_{|V|}$ be any ordering of members of V. We construct a sequence of graphs $G_0 = G, ..., G_{|V|}$ by eliminating vertices of G according to ordering O. For each $i > 0$, G_i is obtained from G_{i-1}, by removing v_i, and adding edges from all its in-neighbors to all its out-neighbors. Formally, $G_i = (V_i, E_i)$ is constructed from $G_{i-1} = (V_{i-1}, E_{i-1})$ so that $V_i = V_{i-1} \backslash \{v_i\}$, and $E_i = E_{i-1} \backslash (\{(v_j, v_i) | (v_j, v_i) \in E_{i-1}\} \cup \{(v_i, v_k) | (v_i, v_k) \in E_{i-1}\}) \bigcup D_i$, where $D_i = \{(v_j, v_k) | (v_j, v_i) \in E_{i-1}, (v_i, v_k) \in E_{i-1}, j \neq k\}$. The vertex elimination graph of G according to elimination ordering O is $G^* = (V, E^*)$, where:

$$E^* = \bigcup_{i=0}^{|V|} E_i \tag{1}$$

The directed elimination width (Hunter and Kreutzer 2007) of ordering O for graph G is defined by the maximum over number of out-neighbors of v_i in G_i for $i = 1, ...|V|$. The directed elimination width of G is the minimum width over all directed elimination orderings for G.

We define Δ as the set of all triangles produced by elimination ordering O for graph G. Members of Δ are all ordered triples (v_i, v_j, v_k) such that (v_i, v_k) is a member of D_j.

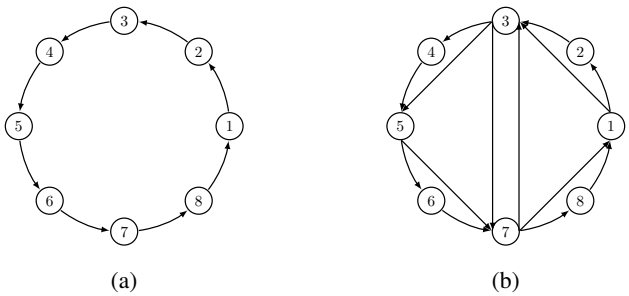


Figure 1: (a) A simple directed cycle (b) The vertex elimination graph

Clearly, for each i there is an edge $(v_j, v_k) \in E_i$ only if there is a path in E_{i-1} with length at most 2 from v_j to v_k. Therefore, if there is an edge (v_j, v_k) in E^*, there must exist a path in G from v_i to v_j. We can conclude that G^* is a subgraph of G^+. However, the difference between $|E^+|$ and $|E^*|$ depends both on the sparsity of G, and the elimination ordering. It has been shown that the problem of finding the optimal ordering, i.e., the ordering that results in the smallest number of edges in the vertex elimination graph, is NP-complete (Rose and Tarjan 1975). Nevertheless, there are effective heuristics for finding empirically usable orderings. An example is *minimum fill-in* heuristic, which chooses v_i so that elimination of v_i adds the minimum number of edges to G_{i-1}. Another example is *minimum degree* heuristic, which chooses v_i with the minimum degree from G_{i-1}.

Example 1 (Vertex elimination graphs)**.** *Consider G to be the graph depicted in Figure 1(a). There are several elimination orderings that can result in a vertex elimination graph depicted in Figure 1(b), among which one possible ordering is 2,4,6,8,1,5,3,7. The elimination width of this order is 1.*

Background

We now explain the methods that have already been introduced for checking acyclicity and reachability when propositional formulas are considered.

Acyclicity

A digraph is acyclic iff there is no path in it from a vertex to itself. Various methods have been introduced to encode acyclicity for symbolic structures with underlying graphs. Here, we review these methods.

Explicit Encodings Explicit encodings are those that prevent cycles by adding specific formulas to the original formula. An important advantage of this approach is that off-the-shelf SAT solvers can be used to check acyclicity for the output formula. Examples of explicit encodings are transitive closure (Brooks et al. 2007; Cussens 2008; Brewka, Eiter, and Truszczynski 2011), topological sorting with indices (Gebser, Janhunen, and Rintanen 2020), tree reduction (Corander et al. 2013; Tamura et al. 2009), and matrix multiplication encoding (Janota, Grigore, and Manquinho 2017).

Of the methods that use explicit acyclicity encoding, transitive closure is known for better performance (Janota, Grigore, and Manquinho 2017). Some of the other encodings (which are not discussed further) have far smaller size by also far poorer propagation and solving efficiency. Therefore, we only use the transitive closure method in this work for evaluating our method against explicit acyclicity encodings.

Given a formula ϕ with underlying graph $G = (V, E)$, encoding of transitive closure as described in (Gebser, Janhunen, and Rintanen 2014b) uses $\mathcal{O}(|V|^2)$ variables and produces $\mathcal{O}(|V||E|)$ clauses.

GraphSAT Another approach is to take acyclicity into account when checking the satisfiability of the given formula. This approach, which does not require adding extra clauses to the formula, has been used in GraphSAT (Gebser, Janhunen, and Rintanen 2014b). GraphSAT is the current state-of-the-art method for checking acyclicity for formulas with underlying graphs.

Given a mapping of arcs to variables, GraphSAT employs a specialized algorithm for detecting a cycle in the graph induced by those arcs that map to true variables, and to infer that a variable must be false to prevent a cycle emerging in the graph. This algorithm is run together with the unit propagation algorithm inside a standard CDCL implementation. GraphSAT has been shown to outperform explicit SAT encodings for acyclicity as well as non-Boolean representations in terms of linear arithmetic constraints in the SAT Modulo Theories framework (Gebser, Janhunen, and Rintanen 2014b).

Encoding Acyclicity by Leveraging Graph Structure Encoding acyclicity as SAT by leveraging low treewidth, the undirected counterpart of ordered elimination width, has previously been done in a number of works. Examples are the SAT encoding for finding treewidth (Samer and Veith 2009) and hypretreewidth (Schidler and Szeider 2020), and leveraging low treewidth for finding structure in Bayesian Networks (Berg, Järvisalo, and Malone 2014; Ramaswamy and Szeider 2021).

In all above-mentioned works, the main focus is to encode the treewidth into SAT. In this work, however, we encode neither directed elimination width nor the ordering that results in the directed elimination width. Instead, the production of vertex elimination graph is done in our work as a preprocessing phase. We use vertex elimination graphs to produce encodings that enable a more efficient propagation of acyclicity constraints.

Reachability

Let $G = (V, E)$ be a digraph, and $s, t \in V$. We say s-t-reachability holds for G iff there is a path from s to t in G. Checking s-t-reachability as SAT has been studied before. Here we survey three main approaches: explicit encoding, reachability via acyclicity, and implicit reachability checking using GraphSAT.

Explicit Encoding Checking unreachability can be done by adding additional formulas to ϕ (Chatterjee, Chmelik, and Davies 2016; Pandey and Rintanen 2018). Let $s = v_s$ and $t = v_t$ be members of V. Encoding of s-t-unreachability for ϕ, denoted by $\phi_{s\text{-}t\text{-}unreach}$, can be produced by conjunction of formulas (2) and (3).

$$r_{s,s} \wedge \bigwedge_{(v_i,v_j)\in E} (e_{i,j} \wedge r_{s,i} \rightarrow r_{s,j}) \tag{2}$$

$$\neg r_{s,t} \tag{3}$$

Checking reachability, on the other hand, is not as easy as checking unreachability. Encoding of s-t-reachability for ϕ, denoted by $\phi^{exp}_{s\text{-}t\text{-}reach}$, can be produced by conjunction of formulas (4) to (6).

$$r^0_{t,t} \wedge \bigwedge_{v_i\in V\setminus\{t\}} \neg r^0_{i,t} \tag{4}$$

$$\bigwedge_{n=1,\ldots|V|-1} \left(r^n_{i,t} \rightarrow r^{n-1}_{i,t} \wedge \bigvee_{(v_i,v_j)\in E} (e_{i,j} \wedge r^{n-1}_{j,t}) \right) \tag{5}$$

$$r^{|V|-1}_{s,t} \tag{6}$$

This encoding is derived from (Pandey and Rintanen 2018). Setting the variable $r^n_{i,t}$ to $true$ means that there is a path with length at most n from v_i to t. The encoding is based on the observation that if t is reachable from s, it is reachable by a path with length at most $|V| - 1$. Formula (4) ensures that there is a path from v_i to t with length zero iff $v_i = t$. Formula (5) guarantees that if there exists a path with length at most n from to v_i to t, then there must exist a path with length at most $n - 1$ from an out-neighbor of v_i to t. Finally, Formula (6) ensures that there is a path with length at most $|V| - 1$ from s to t.

Given a formula ϕ with underlying graph $G = (V, E)$, this encoding uses $\mathcal{O}(|V|^2)$ variables and produces $\mathcal{O}(|V||E|)$ clauses.

Reachability by Acyclicity This encoding of s-t-reachability has been introduced in (Pandey and Rintanen 2018). In this approach, an acyclic subgraph G' of G is encoded to trace the reachability of t from vertices through their out-neighbors. Propositional formulas are added to ensure that 1) t is reachable from itself; 2) if t is reachable from $v \neq t$, then t is reachable from an out-neighbor v' of v in G, and (v, v') is an edge in G'; 3) t is reachable from s; and 4) G' is acyclic. The acyclicity of G' is necessary because otherwise vertices may obtain reachability from one another in a cycle, without actually having a path to t.

This encoding uses $\mathcal{O}(|E|)$ variables and produces $\mathcal{O}(|V| + |E|)$ clauses plus the variables and clauses needed for encoding acyclicity of G'. Note that if GraphSAT is used for checking acyclicity, then there is no need for adding extra variables and clauses for checking the acyclicity of G', as this property will be taken into account by GraphSAT while solving the formula.

Reachability in GraphSAT As it was mentioned for the case of acyclicity, GraphSAT receives a description of the underlying graph in the input. GraphSAT also admits reachability and non-reachability constraints in its input. While searching for a model, GraphSAT persistently checks that the enabled edges conform to these constraints.

Eventual Reachability Let $G = (V, E)$ be a digraph, and $s, t \in V$. We say s-t-eventual-reachability constraint holds for G iff for every $v \in V$, if v is reachable from s, then t is reachable from v.

Both explicit encoding of reachability and encoding of reachability by acyclicity can be modified to produce encodings for s-t-eventual-reachability. For example, the explicit encoding of s-t-eventual-reachability, denoted by $\phi^{exp}_{s\text{-}t\text{-}event}$, can be produced by conjunction of formulas (2), (4), (5), and (7).

$$\bigwedge_{v_i \in V} r_{s,i} \rightarrow r^{|V|-1}_{i,t} \tag{7}$$

Encoding of s-t-eventual-reachability by acyclicity can be produced in similar way (Pandey and Rintanen 2018). It can easily be shown that size properties of the mentioned encodings for s-t-eventual-reachability are the same as their s-t-reachability counterparts.

Encodings with Vertex Elimination Graphs

Assume that ϕ is a propositional formula over the set X of variables, with underlying graph $G = (V, E)$. Let O be an elimination ordering for G, $G^* = (V, E^*)$ be the vertex elimination graph of G according to O, δ be the directed elimination width of O for G, and Δ be the set of all triangles produced for G by vertex elimination according to O. Also, if model $\mathcal{M}$ satisfies ϕ, let $G_{\mathcal{M}}$ be the underlying graph of $\mathcal{M}$, and $G^*_{\mathcal{M}} = (V, E^*_{\mathcal{M}})$ be the vertex elimination graph of $G_{\mathcal{M}}$ according to O.

Encoding of Acyclicity

The encoding of acyclicity for ϕ using vertex elimination according to O, denoted by ϕ^{ve}_{acycl}, is produced by conjunction of formulas (8) to (10):

$$\bigwedge_{(v_i,v_j)\in E} e_{i,j} \rightarrow e'_{i,j} \tag{8}$$

$$\bigwedge_{(v_i,v_j)\in E^*,(v_j,v_i)\in E^*,i<j} e'_{i,j} \rightarrow \neg e'_{j,i} \tag{9}$$

$$\bigwedge_{(v_i,v_j,v_k)\in \Delta} (e'_{i,j} \wedge e'_{j,k}) \rightarrow e'_{i,k} \tag{10}$$

Theorem 1 (Completeness of ϕ^{ve}_{acycl})**.** *If ϕ is satisfied by any model $\mathcal{M}$ such that $G_{\mathcal{M}}$ is acyclic, then $\phi \wedge \phi^{ve}_{acycl}$ is satisfiable.*

Proof. Consider O' to be a topological ordering of members of V according to $G_{\mathcal{M}}$. We construct valuation function $\mathcal{M}'$ for $\phi \wedge \phi^{ve}_{acycl}$ such that for each $x \in X$, $\mathcal{M}'(x) = \mathcal{M}(x)$, and for each $e'_{i,j}$, $\mathcal{M}'(e'_{i,j}) = true$ iff v_i precedes v_j according to O'. By definition, ϕ is trivially satisfied by $\mathcal{M}'$. Formula (8) is satisfied by $\mathcal{M}'$ because if $\mathcal{M}'(e_{i,j}) = true$, v_i precedes v_j according to O', thus $\mathcal{M}'(e'_{i,j}) = true$. Formula (9) is satisfied seeing that if $\mathcal{M}'(e'_{i,j}) = true$, then v_i precedes v_j, and therefore, v_j cannot precede v_i according to O'. Formula (10) is satisfied because if $\mathcal{M}'(e'_{i,j}) = \mathcal{M}'(e'_{j,k}) = true$, then v_i precedes v_j and v_j precedes v_k. Therefore, v_i precedes v_k according to O', and $\mathcal{M}'(e'_{i,k}) = true$. □

Lemma 1. *If $\phi \wedge \phi^{ve}_{acycl}$ is satisfied by model $\mathcal{M}$, then for every $(v_i, v_j) \in E^*_{\mathcal{M}}$, we have $\mathcal{M}(e'_{i,j}) = true$.*

Proof. By formula (8) if $\mathcal{M}(e_{i,j}) = true$ then $\mathcal{M}(e'_{i,j}) = true$. Let $\Delta_{\mathcal{M}}$ be the set of all triangles produced for $G_{\mathcal{M}}$ by vertex elimination according to O. Since $E^*_{\mathcal{M}}$ is a subset of E^*, we conclude that $\Delta_{\mathcal{M}}$ is a subset of Δ. The proof is complete by seeing that $\mathcal{M}$ satisfies formula (10). □

Lemma 2. *Let $G^* = (V, E^*)$ be a vertex elimination graph of an arbitrary graph $G = (V, E)$ according to an arbitrary elimination ordering O. If G has a cycle then for some v and v', we have $(v, v') \in E^*$ and $(v', v) \in E^*$.*

Proof. We give the proof by induction on the number of vertices in the cycle. Base case: for a cycle of two vertices, the conclusion clearly holds. Induction hypothesis: assume that for $k > 2$ and the conclusion holds for any cycle with $k - 1$ vertices. For a cycle with k vertices, the cycle has the form $v_0, ..., v_{k-1}, v_0$. Let v_i be the first vertex in the set $\{v_0, ..., v_{k-1}\}$ that is eliminated according to ordering O. The edges $(v_{[i-1]_k}, v_i)$ and $(v_i, v_{[i+1]_k})$ must be present prior to the elimination of v_i. Therefore, $(v_{[i-1]_k}, v_{[i+1]_k})$ is a member of E^*, constructing a cycle of length $k - 1$. The proof is then complete by the induction hypothesis. □

Theorem 2 (Soundness of ϕ^{ve}_{acycl})**.** *If $\phi \wedge \phi^{ve}_{acycl}$ is satisfied by model $\mathcal{M}$, then $G_{\mathcal{M}}$ is acyclic.*

Proof. Assume that $G_{\mathcal{M}}$ has a cycle. Since $G_{\mathcal{M}}$ is a subgraph of $G^*_{\mathcal{M}}$, we conclude that $G^*_{\mathcal{M}}$ has a cycle, too. According to Lemma 2, for some i and j, we have: $(v_i, v_j) \in E^*_{\mathcal{M}}$ and $(v_j, v_i) \in E^*_{\mathcal{M}}$. Then, according to Lemma 1, we have $\mathcal{M}(e'_{i,j}) = true$ and $\mathcal{M}(e'_{j,i}) = true$, which by considering formula (9) contradicts the assumption that $\phi \wedge \phi^{ve}_{acycl}$ is satisfied by $\mathcal{M}$. □

Theorem 1 and Theorem 2 show that ϕ^{ve}_{acycl} is an encoding of acyclicity for ϕ.

Complexity Analysis For analyzing the size of ϕ^{ve}_{acycl}, note that the number of variables in ϕ^{ve}_{acycl} is proportional to number of edges in G^*, which is $\mathcal{O}(\delta|V|) \subseteq \mathcal{O}(|V|^2)$. This means that in the worst case our vertex elimination based method uses the same asymptomatic number of variables as the transitive closure method. However, directed elimination width can be significantly smaller than $|V|$. By using heuristic methods mentioned before, one can come up with an ordering with directed elimination width close to that of G.

The number of clauses in ϕ^{ve}_{acycl} is proportional to $|\Delta| + |E^*|$, i. e., the total number of triangles produced by eliminating all vertices plus the number of edges in the vertex elimination graph. When eliminating v, the number of triangles produced is at most $in\text{-}degree(v) \times out\text{-}degree(v) \leq in\text{-}degree(v) \times \delta$. By summing over all vertices, we reach to $\delta|E^*|$, which is $\mathcal{O}(\delta^2|V|) \subseteq \mathcal{O}(|V|^3)$. In graphs with low ordered elimination withs, δ^2 can be significantly smaller

than $|E|$, causing production of smaller number of clauses in comparison with the transitive closure method.

Encoding of s-t-Reachability

For encoding s-t-reachability by using vertex elimination according to elimination ordering O, we add a restriction on O. We demand that $s = v_s$ and $t = v_t$ are ordered after all other vertices by O. Assuming this, the encoding of s-t-reachability by using vertex elimination according to elimination ordering O, denoted by $\phi^{ve}_{s\text{-}t\text{-}reach}$, is produced by conjunction of formulas (11) to (13), where $f(e_{i,j})$ is $e_{i,j}$ if $(v_i, v_j) \in E$ and $false$ otherwise.

$$\bigwedge_{(v_i,v_j)\in E^*} (e'_{i,j} \rightarrow f(e_{i,j}) \vee \bigvee_{(v_i,v_k,v_j)\in\Delta} t_{i,k,j}) \tag{11}$$

$$\bigwedge_{(v_i,v_k,v_j)\in\Delta} t_{i,k,j} \rightarrow e'_{i,k} \wedge e'_{k,j} \tag{12}$$

$$e'_{s,t} \tag{13}$$

Theorem 3 (Completeness of $\phi^{ve}_{s\text{-}t\text{-}reach}$)**.** *If ϕ is satisfied by any model $\mathcal{M}$ such that $G_{\mathcal{M}}$ has s-t-reachability, then $\phi \wedge \phi^{ve}_{s\text{-}t\text{-}reach}$ is satisfiable.*

Proof. Let $\Delta_{\mathcal{M}}$ be a subset of Δ constructed similar to Δ but only by considering edges that are in $G_{\mathcal{M}}$, i.e., the underlying graph of $\mathcal{M}$. We construct valuation function $\mathcal{M}'$ for $\phi^{ve}_{s\text{-}t\text{-}reach}$ such that for each $x \in X$, $\mathcal{M}'(x) = \mathcal{M}(x)$, for each $e'_{i,j}$, $\mathcal{M}'(e'_{i,j}) = true$ iff $(v_i, v_j) \in E^*_{\mathcal{M}}$, and $\mathcal{M}'(t_{i,k,j}) = true$ iff $(v_i, v_k, v_j) \in \Delta_{\mathcal{M}}$.

Formula (11) is satisfied by $\mathcal{M}'$ because for i and j such that $(v_i, v_j) \in E^*$, if $\mathcal{M}'(e'_{i,j}) = true$ and either $(v_i, v_j) \notin E$ or $\mathcal{M}'(e_{i,j}) = false$, then for some k, (v_i, v_j) has been added to $G^*_{\mathcal{M}}$ when eliminating some v_k, and therefore, $\mathcal{M}'(t_{i,k,j}) = true$. Formula (12) is trivially satisfied by $\mathcal{M}'$. Also, Formula (11) is satisfied by seeing that since we have assumed that v_s and v_t are eliminated after all other vertices, if there is a path in $G^*_{\mathcal{M}}$ from v_s to v_t, then (v_s, v_t) must be a member of $E^*_{\mathcal{M}}$. □

Lemma 3. *If the conjunction of ϕ, formula (11), and formula (12) is satisfied by model $\mathcal{M}$, and $\mathcal{M}(e'_{i,j}) = true$, then there is a path in $G_{\mathcal{M}}$ from v_i to v_j.*

Proof. Without loss of generality assume that vertices are indexed according to elimination ordering O. We give the proof by strong induction on $m = min(i, j)$. Base case: for $m = 1$, since there are no (v_1, v_k, v_j) or (v_i, v_k, v_1) in Δ, from (11) we deduce that there is an edge in $G_{\mathcal{M}}$ from e_i to e_j. Induction hypothesis: assume that for all n such that $1 \leq n \leq m$ and all $i, j \leq |V|$, if $n = min(i, j)$ and $\mathcal{M}(e'_{i,j}) = true$, then there is a path in $G_{\mathcal{M}}$ from e_i to e_j. We prove that for any i and j such that $min(i, j) = m + 1$ and $\mathcal{M}(e'_{i,j}) = true$, there is a path in $G_{\mathcal{M}}$ from e_i to e_j. Consider formula (11). If $e_{i,j} \in E$ and $\mathcal{M}(e_{i,j}) = true$, then conclusion obviously holds. If $e_{i,j} \notin E$ or $\mathcal{M}(e_{i,j}) = false$, then there must exist k such that $(v_i, v_k, v_j) \in \Delta$ and $\mathcal{M}(t_{i,k,j}) = true$. However, in this case since (v_i, v_j) has been added when eliminating v_k, k must be smaller than both i and j. By formula (12), we must have: $\mathcal{M}(e_{i,k}) = true$, and $\mathcal{M}(e_{k,j}) = true$. Therefore, by induction hypothesis there must be paths from e_i to e_k, and from e_k to e_j in $G_{\mathcal{M}}$. Thus, the conclusion holds. □

Theorem 4 (Soundness of $\phi^{ve}_{s\text{-}t\text{-}reach}$)**.** *If $\phi \wedge \phi^{ve}_{s\text{-}t\text{-}reach}$ is satisfied by model $\mathcal{M}$, then $G_{\mathcal{M}}$ has s-t-reachability.*

Proof. Since $\mathcal{M}$ satisfies (13), by Lemma 3, there must be a path from v_s to v_t in $G_{\mathcal{M}}$. □

Complexity Analysis The number of variables used in $\phi^{ve}_{s\text{-}t\text{-}reach}$ is proportional to $|\Delta| + |E^*|$, which we showed to be $\mathcal{O}(\delta^2|V|) \subseteq \mathcal{O}(|V|^3)$. The number of clauses is also $\mathcal{O}(\delta^2|V|)$, making this encoding suitable for underlying graphs with low ordered elimination widths.

Encoding of s-t-Eventual-Reachability

Without loss of generality assume that vertices are indexed according to elimination ordering O. We also require t to be ordered after all other vertices by O. Assuming these, the encoding of s-t-eventual-reachability by using vertex elimination according to elimination ordering O, denoted by $\phi^{ve}_{s\text{-}t\text{-}event}$, is produced by conjunction of formulas (2), (11), (12), and (14).

$$\bigwedge_{v_i\in V\setminus\{t\}} (r_{s,i} \rightarrow \bigvee_{(v_i,v_j)\in E^*, i<j} e'_{i,j}) \tag{14}$$

Theorem 5 (Completeness of $\phi^{ve}_{s\text{-}t\text{-}event}$)**.** *If ϕ is satisfied by any model $\mathcal{M}$ such that $G_{\mathcal{M}}$ has s-t-eventual-reachability, then $\phi \wedge \phi^{ve}_{s\text{-}t\text{-}event}$ is satisfiable.*

Proof. Let $\Delta_{\mathcal{M}}$ be constructed as it was in the proof of Theorem 3. We construct valuation function $\mathcal{M}'$ for $\phi \wedge \phi^{ve}_{s\text{-}t\text{-}event}$ such that for each $x \in X$, $\mathcal{M}'(x) = \mathcal{M}(x)$, $\mathcal{M}'(e'_{i,j}) = true$ iff $(v_i, v_j) \in E^*_{\mathcal{M}}$, $\mathcal{M}'(t_{i,k,j}) = true$ iff $(v_i, v_k, v_j) \in \Delta_{\mathcal{M}}$, and $\mathcal{M}'(r_{s,i}) = true$ iff v_i is reachable from v_s in $G_{\mathcal{M}}$.

Formulas (2) and (12) are trivially satisfied by $\mathcal{M}'$. Formula (11) is satisfied by $\mathcal{M}'$ by the same argument made in the proof of Theorem 3. If $v_i \in V\setminus\{t\}$ is reachable from v_s in $G_{\mathcal{M}}$, since $G_{\mathcal{M}}$ has s-t-eventual-reachability property, there must exist a path from v_i to t. Not all vertices in such a path can have indices less than i. That is because we have assumed that O puts t after every other vertex. Assume that we traverse the mentioned path until we visit the first vertex v_j such that $i < j$. Since according to O all vertices before visiting v_j are eliminated before eliminating v_i and v_j, we conclude that $(v_i, v_j) \in E^*_{\mathcal{M}}$ and thus, $(v_i, v_j) \in E^*$. Then we have: $\mathcal{M}'(e'_{i,j}) = true$. We can conclude that (14) is also satisfied by $\mathcal{M}'$. □

Theorem 6 (Soundness of $\phi^{ve}_{s\text{-}t\text{-}event}$)**.** *If $\phi \wedge \phi^{ve}_{s\text{-}t\text{-}event}$ is satisfied by model $\mathcal{M}$, then $G_{\mathcal{M}}$ has s-t-eventual-reachability.*

Proof. From formula (2), we can conclude that if v_i is reachable from s, then we have: $\mathcal{M}(r_{s,i}) = true$. By formula (14), for some j such that $i < j$, we have $\mathcal{M}(e'_{i,j}) = true$. By Lemma 3, there must be a path from v_i to v_j in $G_{\mathcal{M}}$. Therefore, v_j is reachable from v_s. We can repeat the same

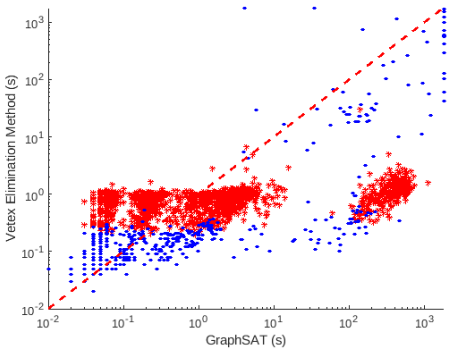


Figure 2: Time (in seconds) needed to solve instances with acyclicity constraints by the vertex elimination method versus GraphSAT

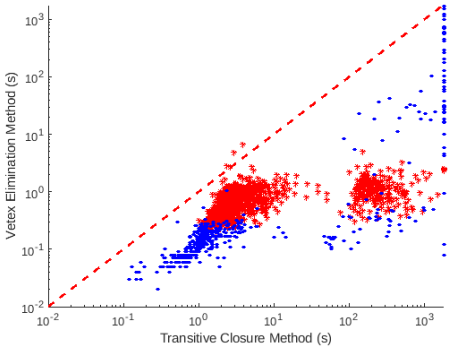


Figure 3: Time (in seconds) needed to solve instances with acyclicity constraints by the vertex elimination method versus transitive closure

argument and find paths from s to vertices with increasing indices. Because O puts t after every other vertex, such paths must at some point reach t. □

It is easy to see that size properties of the encoding of s-t-eventual-reachability by using vertex elimination is asymptotically the same as those of the encoding of s-t-reachability by using vertex eliminations.

Empirical Results and Discussion

For analyzing our methods empirically, we have used two different problem sets. We have used randomly produced 3-SAT formulas with various number of clauses and underlying graphs with fixed number of vertices and different elimination widths. Moreover, we have used planning problem set of (Pandey and Rintanen 2018), which are solvable by both acyclicity checking and eventual reachability checking of their underlying graphs.

We implemented our vertex elimination encodings, as well as the transitive closure method described in (Gebser, Janhunen, and Rintanen 2014b). As the heuristic for elimination orderings of the vertex elimination methods, we have used *mindegree*, i.e., eliminating a vertex with minimal total

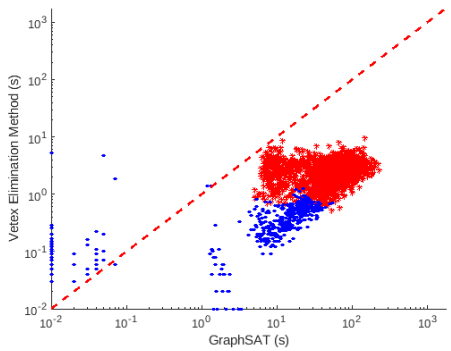


Figure 4: Time (in seconds) needed to solve instances with reachability constraints by the vertex elimination method vesus GraphSAT

number of incoming and outgoing edges in the graph produced after the elimination of previously eliminated vertices. We have solved all instances with GraphSAT, which becomes the Glucose SAT solver (Audemard and Simon 2009) in the absence of special graph constraints in the input formula. For the planning benchmarks we have also used *Kissat* (Biere et al. 2020), which has won the first place in the main track of the SAT Competition 2020. All experiments were run on a cluster of Linux machines, using a timeout of 1800 seconds per instance, and a memory limit of 64 GB.

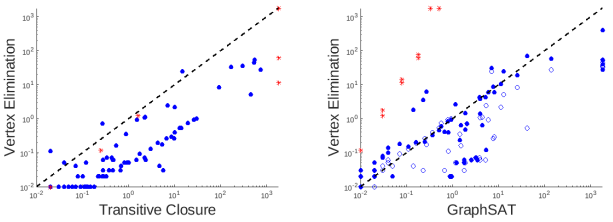
Results on Random Formulas with Underlying Graphs

To test our methods on random formulas, we have used the fixed size benchmark graphs of (Planken, de Weerdt, and van der Krogt 2011) as the underlying graphs. This graph set helps us observe the impact of elimination width measure on the efficiency of our methods. All graphs in this benchmark set have 200 vertices. For $\delta = 5, 10, ..., 195, 199$, there are 10 different randomly generated instances in this set with elimination width δ. For each graph we produced randomly generated 2000,4000,6000,8000, and 10000 3-SAT clauses with propositions referring to edges of the graph, to produce instances of varying size with underlying graphs. The total number of instances used for comparison is therefore 2000.

For comparing the acyclicity checking methods, we require the underlying graph of the found model (if any), to be acyclic. For testing the reachability methods on each instance, we require that a randomly chosen vertex is reachable from all other vertices in the underlying graph of the (possibly) found model. All instances have been solved using GraphSAT, which becomes Glucose SAT solver for vertex elimination and transitive closure encodings.

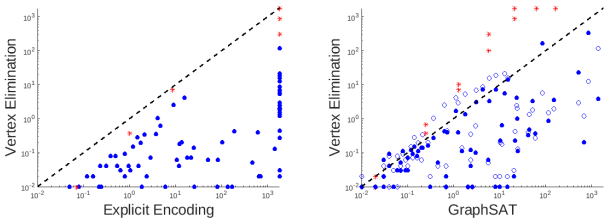
Figures 2 to 4 show the results of our vertex elimination based methods versus that of GraphSAT and the transitive closure method on all instances. Results for problems with widths less than 50 (25 percent of the number of vertices) are demonstrated by dots, while results of other instances are distinguished by "*" symbols.

As it can be seen in Figure 2, our vertex elimination



(a) (b)

Figure 5: Time (in seconds) needed to solve planning instances with acyclicity constraints using vertex elimination versus (a) transitive closure, and (b) GraphSAT



(a) (b)

Figure 6: Time (in seconds) needed to solve planning instances with reachability constraints using vertex elimination versus (a) the explicit encoding, and (b) GraphSAT

method for guaranteeing acyclicity performs better than GraphSAT in almost all problems that are solved by at least one of the methods in more than one second. For the easier instances, GraphSAT can outperform our method, and margin of outperformance of GraphSAT is wider for problems with higher elimination widths. This is mainly due to the fact that formulas produced by our method have greater sizes compared to the original formulas given to GraphSAT. Also, the time needed for computing vertex elimination graphs causes a small time lag for our method. In total, our vertex elimination method solves 1928 instances, 12 instances more than the 1916 instances solved by GraphSAT.

Figure 3 shows that our vertex elimination method almost always outperforms the transitive closure method, and the margin of outperformance is greater for problems with low elimination widths. In total, the transitive closure method solves 1881 instances, 47 instances less than the 1928 instances solved by our vertex elimination method.

For formulas with reachability constraints, as it can be seen in Figure 4, our vertex elimination method performs better than GraphSAT for all instances, except for a few instances that are quite easy to solve. The margin of outperformance is visibly wider for instances with lower elimination widths. Both methods are able to solve all 2000 instances in 1800 seconds.

We do not here show the comparison between our method and the explicit reachability encoding, because the latter method fails to solve even the smallest instances in 1800 seconds.

Results on Planning Problems

To show that our methods have direct impact on current research in AI, we have used the benchmark problem sets of (Pandey and Rintanen 2018) that includes a total of 108 satisfiable and unsatisfiable instances with underlying graphs. These instances can be solved by both reachability and eventual acyclicity checking, and are accompanied with tools that transform reachability constrained instances to equivalent acyclicity constrained instances.

Figures 5 and 6 present the results of our vertex elimination based encodings versus the competing methods on planning benchmark problems. Instances with elimination widths greater than 25 percent of number of vertices are distinguish with "*" symbols in the figures. For comparison with GraphSAT, we have used both Glucose (distinguished with filled dots), and Kissat (distinguished with "o" symbols) solvers. For comparison with transitive closure (Gebser, Janhunen, and Rintanen 2014b) and the explicit reachability encoding ($\phi^{exp}_{s\text{-}t\text{-}event}$ above), we have only used Kissat solver, as it produces slightly better results for all comparing methods.

Our vertex elimination based methods are considered to be explicit in the sense that they incorporate graph constraints into the encoding. Therefore, it would be interesting to see how these methods compare with other explicit methods.

Figures 5a and 6a show that our vertex elimination methods outperform the explicit encodings both on acyclicity constrained and reachability constrained instances, even for instances with high elimination widths. The outperformance margin is wider for reachability constrained instances. These results are significant because explicit encodings allow using off-the-shelf state-of-the-art SAT solvers without any necessity for modifying the solver.

Figures 5b and 6b show that our vertex elimination methods generally outperform GraphSAT both on acyclicity constrained and reachability constrained instances, for instances with low elimination widths. As in the case of the explicit encodings, the outperformance margin is wider for reachability constrained instances. GraphSAT, on the other hand, demonstrates clear dominance for problems with high elimination widths.

Conclusion

We have addressed the problem of checking the satisfiability of propositional formulas with underlying graphs, in the presence of acyclicity and reachability constraints. Novel methods that leverage low elimination width of the underlying graph in order to produce compact encodings for the constraints were introduced. We proved soundness and completeness for each method, and also provided theoretical evidence for efficiency of the methods by parameterized complexity analysis based on the elimination width of the elimination orderings. Moreover, we empirically showed that our new methods can outperform GraphSAT and other encoding methods, especially when underlying graphs have low elimination widths.

References

Audemard, G.; and Simon, L. 2009. Predicting learnt clauses quality in modern SAT solvers. In *Twenty-First International Joint Conference on Artificial Intelligence*, 399–404. AAAI Press.

Berg, J.; Järvisalo, M.; and Malone, B. M. 2014. Learning Optimal Bounded Treewidth Bayesian Networks via Maximum Satisfiability. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*, volume 33 of *JMLR Workshop and Conference Proceedings*, 86–95. JMLR.org.

Biere, A.; Fazekas, K.; Fleury, M.; and Heisinger, M. 2020. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling Entering the SAT Competition 2020. In *Proceedings of SAT COMPETITION 2020*, 50–54.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Communications of the ACM*, 54(12): 92–103.

Brooks, D. R.; Erdem, E.; Erdogan, S. T.; Minett, J. W.; and Ringe, D. 2007. Inferring Phylogenetic Trees Using Answer Set Programming. *Journal of Automated Reasoning*, 39(4): 471–511.

Chatterjee, K.; Chmelik, M.; and Davies, J. 2016. A Symbolic SAT-Based Algorithm for Almost-Sure Reachability with Small Strategies in POMDPs. In Schuurmans, D.; and Wellman, M. P., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, 3225–3232. AAAI Press.

Corander, J.; Janhunen, T.; Rintanen, J.; Nyman, H. J.; and Pensar, J. 2013. Learning Chordal Markov Networks by Constraint Satisfaction. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, 1349–1357.

Cussens, J. 2008. Bayesian network learning by compiling to weighted MAX-SAT. In McAllester, D. A.; and Myllymäki, P., eds., *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008*, 105–112. AUAI Press.

Denecker, M.; and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM Transactions on Computational Logic (TOCL)*, 9(2): 1–52.

Gebser, M.; Janhunen, T.; and Rintanen, J. 2014a. Answer Set Programming by SAT Modulo Acyclicity. In *ECAI 2014. Proceedings of the 21st European Conference on Artificial Intelligence*, 351–356. IOS Press.

Gebser, M.; Janhunen, T.; and Rintanen, J. 2014b. SAT Modulo Graphs: Acyclicity. In *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings*, volume 8761 of *Lecture Notes in Computer Science*, 137–151. Springer-Verlag.

Gebser, M.; Janhunen, T.; and Rintanen, J. 2020. Declarative encodings of acyclicity properties. *Journal of Logic and Computation*, 923–952.

Hunter, P.; and Kreutzer, S. 2007. Digraph measures: Kelly decompositions, games, and orderings. In Bansal, N.; Pruhs, K.; and Stein, C., eds., *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, 637–644. SIAM.

Janota, M.; Grigore, R.; and Manquinho, V. M. 2017. On the Quest for an Acyclic Graph. In Maratea, M.; and Serina, I., eds., *Proceedings of the 24th RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion 2017 co-located with the 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), Bari, Italy, November 14-15, 2017*, volume 2011 of *CEUR Workshop Proceedings*, 33–44. CEUR-WS.org.

Lin, F.; and Zhao, Y. 2004. ASSAT: computing answer sets of a logic program by SAT solvers. *Artificial Intelligence*, 157(1): 115–137.

Pandey, B.; and Rintanen, J. 2018. Planning for Partial Observability by SAT and Graph Constraints. In de Weerdt, M.; Koenig, S.; Roger, G.; and Spaan, M. T. J., eds., *Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling, ICAPS 2018, Delft, The Netherlands, June 24-29, 2018*, 190–198. AAAI Press.

Planken, L.; de Weerdt, M.; and van der Krogt, R. 2011. Computing All-Pairs Shortest Paths by Leveraging Low Treewidth. In Bacchus, F.; Domshlak, C.; Edelkamp, S.; and Helmert, M., eds., *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*. AAAI.

Ramaswamy, V. P.; and Szeider, S. 2021. Turbocharging Treewidth-Bounded Bayesian Network Structure Learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 3895–3903. AAAI Press.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(12-13): 1031–1080.

Rose, D. J.; and Tarjan, R. E. 1975. Algorithmic Aspects of Vertex Elimination. In Rounds, W. C.; Martin, N.; Carlyle, J. W.; and Harrison, M. A., eds., *Proceedings of the 7th Annual ACM Symposium on Theory of Computing, May 5-7, 1975, Albuquerque, New Mexico, USA*, 245–254. ACM.

Samer, M.; and Veith, H. 2009. Encoding Treewidth into SAT. In Kullmann, O., ed., *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, 45–50. Springer.

Schidler, A.; and Szeider, S. 2020. Computing Optimal Hypertree Decompositions. In Blelloch, G. E.; and Finocchi, I., eds., *Proceedings of the Symposium on Algorithm Engineering and Experiments, ALENEX 2020, Salt Lake City, UT, USA, January 5-6, 2020*, 1–11. SIAM.

Tamura, N.; Taga, A.; Kitagawa, S.; and Banbara, M. 2009. Compiling finite linear CSP into SAT. *Constraints*, 14(2): 254–272.