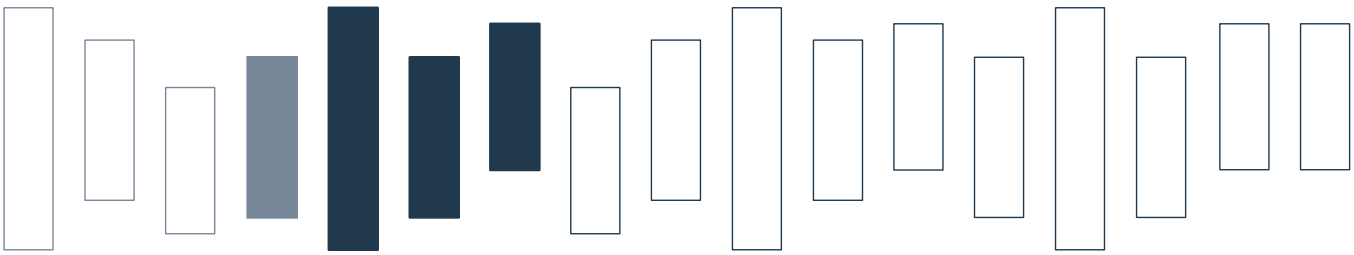
Technical Report AC-TR-18-010

Revised on September 2019

A SAT Approach for Finding Sup-Transition-Minors

Benedikt Klocker, Herbert Fleischner, and Günther R. Raidl



This is the authors' copy of a paper that will be published in the proceedings of the 13th LION Learning and Intelligent OptimizatioN Conference.

www.ac.tuwien.ac.at/tr

A SAT Approach for Finding Sup-Transition-Minors*

Benedikt Klocker, Herbert Fleischner, Günther R. Raidl

Institute of Logic and Computation, TU Wien,
Favoritenstraße 9-11/192-01, 1040 Vienna, Austria
{klocker,fleischner,raidl}@ac.tuwien.ac.at

Abstract. The cycle double cover conjecture is a famous longstanding unsolved conjecture in graph theory. It is related and can be reduced to the compatible circuit decomposition problem. Recently Fleischner et al. (2018) provided a sufficient condition for a compatible circuit decomposition, which is called SUD-K_5-minor freeness. In a previous work we developed an abstract mathematical model for finding SUD-K_5-minors and based on the model a MIP-formulation. In this work we propose a respective SAT-model and compare it with the MIP model in computational tests. Non-trivial symmetry breaking constraints are proposed, which improve the solving times of both models considerably. Compared to the MIP model the SAT approach performs significantly better. We use the faster algorithm to further test graphs of graph theoretic interest and were able to get new insights. Among other results we found snarks with 30 and 32 vertices that do not contain a perfect pseudo-matching, that is a spanning subgraph consisting of K_2 and $K_{1,3}$ components, whose contraction leads to a SUD-K_5-minor free graph.

Keywords: Transition Minor, Cycle Double Cover, Compatible Circuit Decomposition, SAT

1 Introduction

The famous cycle double cover (CDC) conjecture states that every bridgeless graph has a cycle double cover, which is a collection of cycles such that every edge of the graph is part of exactly two cycles. It was originally posed by Szekeres [13] and Seymour [11] over 40 years ago and is still unsolved. As Jaeger shows in [5], the CDC conjecture can be reduced to the consideration of a special class of graphs called snarks by considering a minimum counter example. There are multiple similar definitions of snarks, we will use the one from Jaeger [5]: A *snark* is a simple cyclically 4-edge-connected cubic graph with chromatic index four. A *cyclically 4-edge-connected* graph is a graph that has no 4-edge cut after whose removal at least two components contain a cycle. Although snarks are simple, we consider in general undirected multigraphs without loops in this work.

A problem related to the CDC conjecture is the compatible circuit decomposition (CCD) problem. It is formulated on a *transitioned graph* $(G, \mathcal{T})$, which is a graph G

* This work is supported by the Austrian Science Fund (FWF) under grant P27615 and the Vienna Graduate School on Computational Optimization, grant W1260.

Fig. 1: Example contraction of parts of a PPM. The edges of the PPM getting contracted are drawn dashed. The transitions in the resulting graph are represented by a vee ($\vee$) between the two edges of the transition.

together with a set of transitions $\mathcal{T}$. A *transition* consists of a vertex and two incident edges. We write $\mathcal{T}(v)$ for the set of all transitions at vertex v. A transition system has to satisfy that the transitions in $\mathcal{T}(v)$ are edge-disjoint. A *compatible circuit decomposition* of a transitioned graph is a collection of circuits such that each edge of the graph is part of exactly one circuit and each circuit does not contain any pair of edges of a transition. The CCD problem asks if a given transitioned graph contains a compatible circuit decomposition. To see the connection between the CDC conjecture and the CCD problem we consider a cubic graph C, for example a snark. A *perfect pseudo-matching (PPM)* of C is a subgraph spanning C whose connected components are either two vertices connected by an edge, i.e. the K_2, or one vertex together with its three incident edges and its three neighbors, i.e. the $K_{1,3}$ which we also call *claw*. Given a PPM of C we can define now a transitioned graph $(G, \mathcal{T})$ by contracting all edges of the PPM. We define a transition in $\mathcal{T}$ for each pair of adjacent edges in G that remain after the contraction, see Figure 1 for an illustration.

Note that the contracted graph may contain loops, but we can ignore them since they are not relevant in the context of circuit decompositions. If we contract a PPM of a snark there are no loops since a snark is simple and has no triangles. As described in [8] if the constructed transitioned graph $(G, \mathcal{T})$ contains a CCD one can construct a CDC in the original graph C. Already in 1980 Fleischner [3] proved that every transitioned graph $(G, \mathcal{T})$ where G is 2-connected and planar contains a CCD. This result was then improved in 2000 by Fan and Zhang [2] who showed that if G is 2-connected and K_5-minor free it must contain a CCD. Those two sufficient conditions for the existence of a CCD are only based on the structure of G and do not consider the transition system $\mathcal{T}$. Recently Fleischner et al. [4] generalized the minor term to transitioned graphs and proved that if $(G, \mathcal{T})$ is 2-connected and SUD-K_5-minor free it must contain a CCD. For the definition of a SUD-K_5-minor we refer to [4] or [8].

Because of the complex nature of the definition of a SUD-K_5-minor it is highly non-trivial to check if a graph contains a SUD-K_5-minor. In a previous work [8] we generalized the problem of SUD-K_5-minor containment by allowing to replace the K_5 by any 4-regular graph H. Formally, given a transitioned graph $(G, \mathcal{T})$ and a 4-regular completely transitioned graph $(H, \mathcal{T})$ the decision problem *Existence of Sup-Transition-Minors (ESTM)* asks if $(G, \mathcal{T})$ has a sup-$(H, \mathcal{S})$-transition minor. A sup- A graph is *completely transitioned* if it has a transition system such that for each vertex every incident edge is part of a transition at this vertex. A transitioned graph $(G, \mathcal{T})$

has a sup-$(H, \mathcal{S})$ if it has a H-minor where every vertex w of H corresponds to a subgraph C_w of G that has a cut vertex v_w. The cut vertex must split C_w in at least two components such that there is a transition at v_w whose edges are part of one of those components C_w^1 and all other edges incident to v_w are part of other components. Furthermore, the transition at v_w must correspond to a transition of H at w such that the two edges of the transition in H are connected to the component C_w^1. For a formal definition of a sup-$(H, \mathcal{S})$-transition minor we refer to [8].

The mathematical model developed in [8] for deciding the ESTM allowed to derive a MIP model, which could be solved for small graphs such that we could derive interesting graph theoretic results from it. In this work we present a more powerful SAT formulation for the mathematical model developed in [8], which allows addressing significantly larger graphs. To further improve the solving times of the MIP as well as the SAT model we propose a non-trivial symmetry breaking based on graph automorphisms of the two input graphs $(G, \mathcal{T})$ and $(H, \mathcal{S})$. The idea of breaking symmetries using automorphism groups has been studied in a general context, see e.g. [1], and in problem-specific contexts, see e.g. [7]. We extend the definition of automorphisms to transitioned graphs and propose problem specific symmetry breaking constraints based on a vertex mapping between the two input graphs $(G, \mathcal{T})$ and $(H, \mathcal{S})$.

Using the new SAT model, which outperforms the MIP model significantly, together with the symmetry breaking constraints we were able to check for all snarks with up to 32 vertices if they contain a PPM whose contraction is SUD-K_5-minor free. Within those tests we were able to find snarks that do not contain such a PPMs. This result answers the previously open question that the notion of SUD-K_5-minor freeness in the context of contractions of PPMs in snarks is not enough to prove the CDC conjecture.

In the following section we will present a SAT model for finding a sup-$(H, \mathcal{S})$-transition minor. Then we will discuss symmetry breaking constraints, which can be used in the MIP and in the SAT model, in Section 3. Section 4 gives computational results for the new SAT model in comparison to the MIP model. Also we show the impact of the symmetry breaking. Finally, we will conclude and propose some future work in Section 5.

1.1 Terminology and Notation

As already mentioned, when referring to a graph we mean here an undirected multigraph without loops if not other specified. We denote a graph by $G = (V, E, r)$ with a vertex set V, an edge set E and a function r that maps an edge $e \in E$ to the set of its two end vertices $\{v_1, v_2\}$. If $r(e) = \{v_1, v_2\}$ we also write $e = v_1v_2$. Note that $e = v_1v_2$ and $e' = v_1v_2$ does not imply $e = e'$ since we can have parallel edges. For a vertex $v \in V$ we write $E(v) = \{e \in E \mid v \in r(e)\}$ for the set of all incident vertices and $N(v) = \{v' \in V \mid \exists e \in E : e = vv'\}$ for the set of all neighbors.

For a partial function $\alpha : A \nrightarrow B$ and a subset $X \subseteq A$ we write $\alpha[X] = \{b \in B : \exists a \in X : b = \alpha(a)\}$ for the image of X under α. Furthermore, if $Y \subseteq B$ we write $\alpha^{-1}[Y] = \{a \in A : \alpha(a) \in Y\}$ for the preimage of Y under α. We also abbreviate the notation in case of only one element by $\alpha[a] = \alpha[\{a\}]$ and $\alpha^{-1}[b] = \alpha^{-1}[\{b\}]$ for $a \in A$ and $b \in B$.

2 The SAT model

In this section we present a SAT model for checking if a given transitioned graph $(G, \mathcal{T})$ contains a sup-$(H, \mathcal{S})$-transition minor for a given completely transitioned 4-regular graph $(H, \mathcal{S})$. For the formal definition of a sup-$(H, \mathcal{S})$-transition minor see [8].

In the following we first repeat the mathematical model developed in [8] on which the SAT model will be based. The model will use simple trees C with vertices in G for which we will use the following notation: $E_w^i := \{e \in E(G) \mid r(e) \in E(C)\}$. The model is defined as finding

1. a partial surjective function $\varphi\colon V(G) \twoheadrightarrow V(H)$,
2. a partial injective and surjective function $\kappa\colon E(G) \twoheadrightarrow E(H)$,
3. a partial injective function $\theta\colon E(G) \twoheadrightarrow V(H)$,
4. $\forall w \in V(H)$ a pair (T_w, S_w) of transitions with $T_w \in \mathcal{T}$ and $S_w \in \mathcal{S}(w)$, and
5. $\forall w \in V(H)$ two simple trees C_w^1 and C_w^2 with $V(C_w^i) \subseteq V(G)$ for $i = 1, 2$,

such that

$$E(C_w^i) \subseteq r_G[E(G)] \qquad \forall w \in V(H), \forall i \in \{1,2\} \tag{1}$$

$$\kappa(e) = f \Rightarrow \varphi[r_G(e)] = r_H(f) \qquad \forall e \in E(G), \forall f \in E(H) \tag{2}$$

$$V(C_w^1) \cup V(C_w^2) = \varphi^{-1}[w] \qquad \forall w \in V(H) \tag{3}$$

$$\{\pi_1(T_w)\} = V(C_w^1) \cap V(C_w^2) \qquad \forall w \in V(H) \tag{4}$$

$$\pi_2(T_w) \subseteq \kappa^{-1}[\pi_2(S_w)] \cup \theta^{-1}[w] \cup E_w^1 \qquad \forall w \in V(H) \tag{5}$$

$$\left(\kappa^{-1}[\pi_2(S_w)] \cap E(\pi_1(T_w))\right) \cup \theta^{-1}[w] \subseteq \pi_2(T_w) \qquad \forall w \in V(H) \tag{6}$$

$$e \in \mathrm{dom}(\kappa) \wedge \kappa(e) \in \pi_2(S_w) \Rightarrow r_G(e) \cap V(C_w^1) \neq \emptyset \qquad \forall w \in V(H), \forall e \in E(G) \tag{7}$$

$$e \in \mathrm{dom}(\kappa) \wedge \kappa(e) \in E(w) \setminus \pi_2(S_w) \Rightarrow r_G(e) \cap V(C_w^2) \neq \emptyset \qquad \forall w \in V(H), \forall e \in E(G) \tag{8}$$

$$v \in V(C_w^1) \setminus \{\pi_1(T_w)\} \wedge \deg_{C_w^1}(v) = 1 \wedge v \notin \bigcup r_G[\theta^{-1}[w]] \Rightarrow E(v) \cap \kappa^{-1}[\pi_2(S_w)] \neq \emptyset \qquad \forall w \in V(H), \forall v \in V(G) \tag{9}$$

$$E_{C_w^1}(\pi_1(T_w)) \subseteq r_G[\pi_2(T_w)] \qquad \forall w \in V(H) \tag{10}$$

$$\theta(e) = w \Rightarrow r_G(e) \subseteq V(C_w^1) \qquad \forall e \in E(G), \forall w \in V(H) \tag{11}$$

$$\theta(e) = w \Rightarrow r_G(e) \notin E(C_w^1) \qquad \forall e \in E(G), \forall w \in V(H) \tag{12}$$

holds.

In [8] we proved that the feasibility of this model is equivalent to the existence of a sup-$(H, \mathcal{S})$-transition minor in $(G, \mathcal{T})$. Most constraints of this model can more or less directly be translated into SAT clauses. One critical aspect is how to model the tree C_w^i for $w \in V(H)$ and $i \in \{1, 2\}$. Constraints (3) and (4) ensure that the subgraph C_w formed by C_w^1 and C_w^2 together is a tree and all trees C_w are disjoint for $w \in V(H)$. Combining all trees C_w for $w \in V(H)$ we obtain a forest and for each of the trees C_w we define a unique root $\pi_1(T_w)$ by (4). When modeling the forest in a directed fashion, we then only have to take care to avoid any cycles. There are different techniques in literature to model acyclicity in SAT models. Some of those techniques are summarized in [6]. We will use the approach based on a transitive closure for ensuring acyclicity in our model. Our SAT model uses the following variables:

- x_v^w for $v \in V(G)$, $w \in V(H)$ represents $\varphi(v) = w$,
- y_e^f for $e \in E(G)$, $f \in E(H)$ represents $\kappa(e) = f$,
- z_e^w for $e \in E(G)$, $w \in V(H)$ represents $\theta(e) = w$,
- a_T^w for $w \in V(H)$, $T \in \mathcal{T}$ represents $T = T_w$,
- b_S^w for $w \in V(H)$, $S \in \mathcal{S}(w)$ represents $S = S_w$,
- $o_v^{i,w}$ for $v \in V(G)$, $w \in V(H)$, $i \in \{1, 2\}$ represents $v \in V(C_w^i)$,
- $p_a^{i,w}$ for $a \in A(G)$, $w \in V(H)$, $i \in \{1, 2\}$ represents $a \in E(C_w^i)$,
- t_{v_1,v_2} for $v_1, v_2 \in V(G)$ is the transitive closure relation of all $p_a^{i,w}$ variables.

The trees C_w^i are modeled as a directed rooted out-trees and the variables $p_a^{i,w}$ decide which directed arcs are part of the tree. Set $A(G)$ is the set of all directed arcs of edges in G when eliminating parallel edges. So for every pair of adjacent vertices in G there are two arcs in opposite direction in $A(G)$. We write $A^{\text{in}}(v)$ for the ingoing arcs at v and $A^{\text{out}}(v)$ for the outgoing arcs at v. In the following we list all constraints of our SAT model. For simplicity, we will present the constraints in form of propositional logic formulas. To transform them into clauses we use De Morgan's law and the distributive property. One alternative would be to use Tseitin transformations [14], although for the constraints we will present the number of resulting clauses using the naive transformation is still small and therefore this is not needed. In the following we will use for a given $v \in V(G)$, $w \in V(H)$, and $i \in \{1, 2\}$

$$\text{oneIn}(v, i, w) := \Big(\bigvee_{a \in A^{\text{in}}(v)} p_a^{i,w} \Big) \wedge \bigwedge_{\substack{a_1, a_2 \in A^{\text{in}}(v) \\ a1 \neq a_2}} \left(\neg p_{a_1}^{i,w} \vee \neg p_{a_2}^{i,w} \right).$$

The basic structures as defined in the mathematical model are expressed by

$$\neg(x_v^{w_1} \wedge x_v^{w_2}) \qquad \forall v \in V(G), \forall w_1, w_2 \in V(H), w_1 \neq w_2 \tag{13}$$

$$\bigvee_{v \in V(G)} x_v^w \qquad \forall w \in V(H) \tag{14}$$

$$\neg(y_e^{f_1} \wedge y_e^{f_2}) \qquad \forall e \in E(G), \forall f_1, f_2 \in E(H), f_1 \neq f_2 \tag{15}$$

$$\neg(y_{e_1}^f \wedge y_{e_2}^f) \qquad \forall e_1, e_2 \in E(G), e_1 \neq e_2, \forall f \in E(H) \tag{16}$$

$$\bigvee_{e \in E(G)} y_e^f \qquad \forall f \in E(H) \tag{17}$$

$$\neg(z_e^{w_1} \wedge z_e^{w_2}) \qquad \forall e \in E(G), \forall w_1, w_2 \in V(H), w_1 \neq w_2 \tag{18}$$

$$\neg(z_{e_1}^w \wedge z_{e_2}^w) \qquad \forall e_1, e_2 \in E(G), e_1 \neq e_2, \forall w \in V(H) \tag{19}$$

$$\neg(a_{T_1}^w \wedge a_{T_w}^w) \qquad \forall w \in V(H), \forall T_1, T_2 \in \mathcal{T}, T_1 \neq T_2 \tag{20}$$

$$\bigvee_{T \in \mathcal{T}} a_T^w \qquad \forall w \in V(H) \tag{21}$$

$$\neg(a_T^{w_1} \wedge a_T^{w_2}) \qquad \forall w_1, w_2 \in V(H), w_1 \neq w_2 \forall T \in \mathcal{T} \tag{22}$$

$$\neg(b_{S_1}^w \wedge b_{S_2}^w) \qquad \forall w \in V(H), \forall S_1, S_2 \in \mathcal{S}(w), S_1 \neq S_2 \tag{23}$$

$$\bigvee_{S \in \mathcal{S}(w)} b_S^w \qquad \forall w \in V(H) \tag{24}$$

$$o_v^{i,w} \rightarrow \bigvee_{T \in \mathcal{T}(v)} a_T^w \vee \text{oneIn}(v, i, w) \qquad \forall v \in V(G), w \in V(H), i \in \{1, 2\} \tag{25}$$

$$\bigvee_{T\in\mathcal{T}(v)} a_T^w \to o_v^{i,w} \wedge \bigwedge_{a\in A^{\text{in}}(v)} \neg p_a^{i,w} \qquad \forall v \in V(G), w \in V(H), i \in \{1,2\} \quad (26)$$

$$\neg o_v^{i,w} \to \bigwedge_{a\in A^{\text{in}}(v)\cup A^{\text{out}}(v)} \neg p_a^{i,w} \qquad \forall v \in V(G), w \in V(H), i \in \{1,2\} \quad (27)$$

$$\neg(t_{v_1,v_2} \wedge t_{v_2,v_1}) \qquad \forall a = (v_1, v_2) \in A(G) \quad (28)$$

$$t_{v_1,v_2} \wedge t_{v_2,v_3} \to t_{v_1,v_3} \qquad \forall a = (v_1, v_2) \in A(G), v_3 \in V(G) \quad (29)$$

$$\bigvee_{w\in V(H),i\in\{1,2\}} p_a^{i,w} \to t_{v_1,v_2} \qquad \forall a = (v_1, v_2) \in A(G). \quad (30)$$

Constraints (13), (15), (18), (20), and (23) ensure that φ, κ, θ, $w \mapsto T_w$, and $w \mapsto S_w$ are partial functions with the special restriction that $S_w \in \mathcal{S}(w)$. Furthermore, constraints (14) and (17) enforce that φ and κ are surjective. On the other hand, constraints (16), (19), and (22) ensure that κ, θ, and $w \mapsto T_w$ are injective. Note that the mathematical model does not state directly that $w \mapsto T_w$ should be injective, but it does indirectly by constraints (3) and (4). Additionally, constraints (21) and (24) guarantee that there exists a T_w and a S_w for each $w \in V(H)$.

Constraints (25)-(27) characterize three types of vertices in G. The root vertices of the trees C_w^i, which are defined by the vertices of the transitions T_w by (4), do not have any ingoing arcs in C_w^i. Other vertices in C_w^i that are not roots have exactly one ingoing arc in C_w^i and vertices that are not in C_w^i have no ingoing or outgoing arc in C_w^i. Last but not least, constraints (28)-(30) ensure that the trees C_w^i have no cycles by using the transitive closure variables t_{v_1,v_2} similarly as it is described in [6]. Instead of having just one variable, which represents if a directed edge is part of the forest, we use in our case the disjunction $\bigvee_{w\in V(H),i\in\{1,2,\}} p_a^{i,w}$ for an arc a. With this we ensured all structural properties formulated in the mathematical model. What is left is to model constraints (1)-(12) which is achieved by

$$y_e^f \to (x_{v_1}^{w_1} \wedge x_{v_2}^{w_2}) \vee (x_{v_2}^{w_1} \wedge x_{v_1}^{w_2}) \qquad \begin{array}{r}\forall e = v_1, v_2 \in E(G),\\ \forall f = w_1 w_2 \in E(H)\end{array} \quad (31)$$

$$o_v^{1,w} \vee o_v^{2,w} \leftrightarrow x_v^w \qquad \forall v \in V(G), \forall w \in V(H) \quad (32)$$

$$\bigvee_{T\in\mathcal{T}(v)} a_T^w \leftrightarrow o_v^{1,w} \wedge o_v^{2,w} \qquad \forall v \in V(G), \forall w \in V(H) \quad (33)$$

$$\bigvee_{\substack{T\in\mathcal{T}\\ e\in\pi_2(T)}} a_T^w \to \bigvee_{S\in\mathcal{S}(w)} \left(b_S^w \wedge \bigvee_{f\in\pi_2(S)} y_e^f\right) \vee z_e^w \vee p_{(v_1,v_2)}^{1,w} \vee p_{(v_1,v_2)}^{2,w} \qquad \begin{array}{r}\forall e = v_1 v_2 \in E(G),\\ \forall w \in V(H)\end{array} \quad (34)$$

$$a_T^w \wedge b_S^w \to \neg \bigvee_{f\in\pi_2(S)} y_e^f \qquad \begin{array}{r}\forall w \in V(H), \forall S \in \mathcal{S}(w),\\ \forall T \in \mathcal{T}, \forall e \in E(\pi_1(T)) \setminus \pi_2(T)\end{array} \quad (35)$$

$$a_T^w \to \neg z_e^w \qquad \begin{array}{r}\forall w \in V(H),\\ \forall T \in \mathcal{T}, \forall e \in E(\pi_1(T)) \setminus \pi_2(T)\end{array} \quad (36)$$

$$b_S^w \wedge \bigvee_{f\in\pi_2(S)} y_e^f \to o_{v_1}^{1,w} \vee o_{v_2}^{1,w} \qquad \begin{array}{r}\forall w \in V(H), \forall S \in \mathcal{S}(w),\\ \forall e = v_1 v_2 \in E(G)\end{array} \quad (37)$$

$$\left(b_S^w \wedge \bigvee_{f\in E(w)\setminus\pi_2(S)} y_e^f\right) \to o_{v_1}^{2,w} \vee o_{v_2}^{2,w} \qquad \begin{array}{r}\forall w \in V(H), \forall S \in \mathcal{S}(w),\\ \forall e = v_1 v_2 \in E(G)\end{array} \quad (38)$$

$$b_S^w \wedge o_v^{1,w} \wedge \bigwedge_{v' \in N(v)} \neg p_{(v,v')}^{1,w} \wedge \bigwedge_{e \in E(v)} \neg z_e^w \rightarrow \left(\bigvee_{e \in E(v), f \in \pi_2(S)} y_e^f \vee o_v^{2,w} \right) \qquad \forall w \in V(H), \forall S \in \mathcal{S}(w), \forall v \in V(G) \tag{39}$$

$$a_T^w \rightarrow \neg p_{(\pi_1(T),v)}^{1,w} \wedge \neg p_{(v,\pi_1(T))}^{1,w} \qquad \forall w \in V(H), \forall T \in \mathcal{T}, \forall v \in N(\pi_1(T)) \setminus \bigcup r_G[\pi_2(T)] \tag{40}$$

$$z_e^w \rightarrow (o_{v_1}^{1,w} \wedge o_{v_2}^{1,w}) \qquad \forall w \in V(H), \forall e = v_1v_2 \in E(G) \tag{41}$$

$$z_e^w \rightarrow (\neg p_{(v_1,v_2)}^{1,w} \wedge \neg p_{(v_2,v_1)}^{1,w}) \qquad \forall w \in V(H), \forall e = v_1v_2 \in E(G). \tag{42}$$

Constraints (1) are already satisfied implicitly and constraints (2)-(5) are realized by constraints (31)-(34) respectively. Furthermore, constraints (6) are guaranteed by (35) and (36). All the other constraints (7)-(12) are modeled via (37)-(42) respectively.

By using our SAT model we can develop an algorithm that checks for a given snark if it contains a PPM whose contraction leads to a planar, a K_5-minor free, a SUD-K_5-minor free, or a CCD-containing graph. The algorithm enumerates all PPMs iteratively by ordering the vertices of the snark and always trying to add all possible edges or claws to the pseudo-matching that contain the smallest not yet visited vertex of the snark. Then it checks for each generated PPM if its contraction leads to a planar graph. If it does not find such a matching it checks for K_5-minor free contractions, if this also is not the case it checks for SUD-K_5-minor free contractions and otherwise it checks for CCD-containing contractions. Using this algorithm one can specify for each snark the type of the strongest matching found for this snark.

3 Symmetry Breaking

The input graphs G and H, especially H, often have symmetries. In this case our model leads to symmetric solutions. To avoid such symmetries we analyze the structure of the symmetries in G and H and incorporate symmetry breaking constraints into our model such that those symmetries are eliminated.

To formalize the concept of symmetries in transitioned graphs we extend the definition of automorphisms on graphs, i.e. a graph isomorphism from a graph to itself, to transitioned graphs. Before we can do that we need to define the terms for multigraphs since they are traditionally only defined for simple graphs.

Definition 1. *Let G and H be two multigraphs. A function $f : V(G) \rightarrow V(H)$ is a* graph homomorphism *from G to H if and only if there exists a function $g : E(G) \rightarrow E(H)$ such that*

$$f[r(e)] = r(g(e)) \quad \forall e \in E(G).$$

If a homomorphism $f : V(G) \rightarrow V(H)$ is bijective and the inverse function is also a homomorphism it is called a isomorphism*. A isomorphism $f : V(G) \rightarrow V(G)$ from a graph to itself is called an automorphism of G.*

Note that we did not define the edge mappings g as part of the homomorphism but just ensured their existence. If we would consider them as part of the homomorphism

this would result in another definition that allows more different homomorphisms just based on different edge mappings. We focus here on symmetries between the vertices to formulate symmetry breaking constraints within the context of the vertex mapping φ of the model. If we would want to also eliminate edge symmetries this would lead to more complex symmetry breaking constraints and would only help in cases where there are a lot of parallel edges. For simple graphs this definition coincides with the traditional definition of a homomorphism and therefore this is really an extension to multigraphs.

Now we can extend this definition further to transitioned graphs.

Definition 2. *Let $(G, \mathcal{T})$ and $(H, \mathcal{S})$ be two transitioned graphs. A function $f : V(G) \to V(H)$ is a* homomorphism *from $(G, \mathcal{T})$ to $(H, \mathcal{S})$ if and only if f is a graph homomorphism from G to H with a function $g : E(G) \to E(H)$ as described in Definition 1 such that*

$$(f(v), \{g(e_1), g(e_2)\}) \in \mathcal{S} \quad \forall T = (v, \{e_1, e_2\}) \in \mathcal{T}$$

If a homomorphism $f : V(G) \to V(H)$ is bijective and the inverse function is also a homomorphism it is called an isomorphism*. An isomorphism $f : V(G) \to V(G)$ from a transitioned graph to itself is called an automorphism of $(G, \mathcal{T})$.*

It is easy to verify that the set of all automorphisms of a transitioned graph form a group in the same way as they do for simple graphs. We denote this group by $\mathrm{Aut}(G, \mathcal{T})$. Given input graphs $(G, \mathcal{T})$ and $(H, \mathcal{S})$ we can use automorphisms to transform feasible solutions into other feasible solutions. More formally for any feasible solution and any pair of automorphisms $f \in \mathrm{Aut}(G, \mathcal{T})$ and $g \in \mathrm{Aut}(H, \mathcal{S})$ of which at least one of both is not the identity we can construct another feasible solution by replacing all vertices in G according to f and all vertices in H according to g. Since f and g preserve all edges and all transitions this is sufficient to get a new feasible solution.

Next we propose an approach how to eliminate some of those symmetries. Let S be a feasible solution with vertex mapping $\varphi : V(G) \twoheadrightarrow V(H)$. We assume that $V(G)$ and $V(H)$ are totally ordered sets. We can define for any pair of automorphisms $f \in \mathrm{Aut}(G, \mathcal{T})$ and $g \in \mathrm{Aut}(H, \mathcal{S})$ a sequence $\alpha^{f,g} := (x_w^{f,g})_{w \in V(H)}$ by

$$\alpha_w^{f,g} := \min f[\varphi^{-1}[g(w)]]$$

which is well-defined since φ is surjective. The sequence $\alpha^{f,g}$ contains the smallest vertex of each preimage of φ after applying the automorphisms f and g to the solution. The idea is to enforce that $\alpha := \alpha^{\mathrm{id}_{V(G)}, \mathrm{id}_{V(H)}}$ is lexicographically minimal compared to all $\alpha^{f,g}$ for all pairs of automorphisms $f \in \mathrm{Aut}(G, \mathcal{T})$ and $g \in \mathrm{Aut}(H, \mathcal{S})$, i.e.

$$\alpha \leq_{\mathrm{lex}} \alpha^{f,g} \quad \forall f \in \mathrm{Aut}(G, \mathcal{T}), \forall g \in \mathrm{Aut}(H, \mathcal{S}). \tag{43}$$

Note that there may be multiple different feasible solutions with the same sequence α and therefore this only eliminates some symmetries. Such different solutions with the same α may differ in the mapped edges or transitions, or differ in vertices in G that are not mapped by φ or are not the smallest vertices of the preimages of φ. But if H is simple this restriction eliminates all symmetries occurring only in H, i.e. if we only apply an automorphism in $\mathrm{Aut}(H, \mathcal{S}) \setminus \{\mathrm{id}_{V(H)}\}$ to a feasible solution satisfying (43) the resulting solution will not satisfy (43). This can be seen by the fact that φ

is surjective and since H is simple every automorphism really changes at least some vertices, which results in a changed α sequence. To formalize (43) in such a way that it can be modeled in a MIP or a SAT formulation we have to expand the definition of a lexicographical ordering. Condition (43) is equivalent to

$$\begin{aligned}
&\forall w \in V(H), \forall f \in \mathrm{Aut}(G,\mathcal{T}), \forall g \in \mathrm{Aut}(H,\mathcal{S}) : \\
&\alpha_w \leq \alpha_w^{f,g} \vee \exists w' < w : \alpha_{w'} < \alpha_{w'}^{f,g} \\
\Leftrightarrow &(\forall v < \alpha_w : f(v) \notin \varphi^{-1}[g(w)]) \vee (\exists w' < w : \forall v \leq \alpha_w : f(v) \notin \varphi^{-1}[g(w')]).
\end{aligned}$$

This constraint is still quite complicated and results in a lot of constraints in SAT or MIP models. To avoid bloating the models we consider only the variant for the smallest vertex $w_0 := \min(V(H))$ of H. Then the condition gets much easier and can further be simplified using orbits.

Definition 3. *Let f be an automorphism on a transitioned graph $(G,\mathcal{T})$. The set*

$$\mathrm{orb}(v) := \{v' \in V \mid \exists f \in \mathrm{Aut}(G,\mathcal{T}) : f(v) = v'\}$$

is called the orbit *of $v \in V$. Orbits are the equivalence classes of the equivalence relation corresponding to $\mathrm{Aut}(G,\mathcal{T})$ in which two vertices are equivalent if there exists an automorphism mapping one vertex to the other.*

Using the definition of orbits we can simplify our condition for the special case w_0

$$\begin{aligned}
&f(v) \notin \varphi^{-1}[g(w_0)]\ \forall v < \alpha_{w_0}, \forall f \in \mathrm{Aut}(G,\mathcal{T}), \forall g \in \mathrm{Aut}(H,\mathcal{S}) \\
\Leftrightarrow &v' \notin \varphi^{-1}[w']\ \forall v < \alpha_{w_0}, \forall v' \in \mathrm{orb}(v), \forall w' \in \mathrm{orb}(w_0) \\
\Leftrightarrow &\varphi(v') = w' \rightarrow \alpha_{w_0} \leq v\ \forall v \in V(G), \forall v' \in \mathrm{orb}(v), \forall w' \in \mathrm{orb}(w_0) \\
\Leftrightarrow &\varphi(v') = w' \rightarrow \exists v'' \leq v : \varphi(v'') = w_0\ \forall v' \in V(G), \forall v \in \mathrm{orb}(v'), \forall w' \in \mathrm{orb}(w_0) \\
\Leftrightarrow &\varphi(v) = w \rightarrow \exists v' \leq \min \mathrm{orb}(v) : \varphi(v') = w_0\ \forall v \in V(G), \forall w \in \mathrm{orb}(w_0). \qquad (44)
\end{aligned}$$

Another specialization of (43) is if we only consider automorphisms on H, i.e. fix $f = id_{V(G)}$. In this case we simply have $\alpha_w^g := \alpha_w^{\mathrm{id}_{V(G)},g} = \min \varphi^{-1}[g(w)] = \alpha_{g(w)}$, i.e. the α values are simple permutations of each other based on g. Therefore the symmetry breaking condition holds if and only if

$$(\alpha_w)_{w \in V(H)} \leq_{\mathrm{lex}} (\alpha_{g(w)})_{w \in V(H)} \quad \forall g \in \mathrm{Aut}(H,\mathcal{S}). \qquad (45)$$

Note that $(\alpha_w)_{w \in V(H)} \leq_{\mathrm{lex}} (\alpha_{g(w)})_{w \in V(H)}$ if and only if for the first vertex w for which $\alpha_w \neq \alpha_{g(w)}$, $\alpha_w < \alpha_{g(w)}$ holds. Since all values in α_w are different we know that $\alpha_w = \alpha_{g(w)}$ if and only if $w = g(w)$. Therefore, if w is the first value where they are different this implies that g fixes all $w' < w$, i.e. $g(w') = w'$ for all $w' < w$.

Definition 4. *Let $S \subseteq V(G)$, then the* stabilizer *of $\mathrm{Aut}(H,\mathcal{S})$ with respect to S is defined by $\mathrm{Aut}_S(H,\mathcal{S}) := \{g \in \mathrm{Aut}(H,\mathcal{S}) \mid \forall s \in S : g(s) = s\}$. For each set $S \subseteq V(G)$ the $\mathrm{Aut}_S(H,\mathcal{S})$ is a subgroup of $\mathrm{Aut}(H,\mathcal{S})$. Furthermore, we can again define stabilizer orbits according to the automorphisms in the stabilizer. The* orbit of v in the stabilizer $\mathrm{Aut}_S(H,\mathcal{S})$ *is defined by*

$$\mathrm{orb}_S(v) := \{v' \in V \mid \exists f \in \mathrm{Aut}_S(G,\mathcal{T}) : f(v) = v'\}.$$

With this definition we can reformulate (45) in the following way:

$$\begin{aligned} &\alpha_w < \alpha_{g(w)} \quad \forall w \in V(H), \forall g \in \mathrm{Aut}_{\{w' \in V(H): w' < w\}}(H, \mathcal{S}) : g(w) \neq w \\ \Leftrightarrow &\alpha_w < \alpha_{w''} \quad \forall w \in V(H), \forall w'' \in \mathrm{orb}_{\{w' \in V(H): w' < w\}}(w) \setminus \{w\}\,. \end{aligned}$$

The condition $\alpha_w < \alpha_{w''}$ can be expressed such that the statement is equivalent to

$$\varphi(v) = w'' \rightarrow \exists v' < v : \varphi(v') = w \qquad \begin{array}{r} \forall v \in V(G), \forall w \in V(H), \\ \forall w'' \in \mathrm{orb}_{\{w' \in V(H): w' < w\}}(w) \setminus \{w\}\,. \end{array} \tag{46}$$

To model constraints (44) and (46) we use the inequalities

$$x_v^w \leq \sum_{v' \leq \min \mathrm{orb}(v)} x_{v'}^{w_0} \qquad \forall v \in V(G), \forall w \in \mathrm{orb}(w_0) \tag{47}$$

$$x_v^{w''} \leq \sum_{v' < v} x_{v'}^{w} \qquad \begin{array}{r} \forall v \in V(G), \forall w \in V(H), \\ \forall w'' \in \mathrm{orb}_{\{w' \in V(H): w' < w\}}(w) \setminus \{w\} \end{array} \tag{48}$$

for the MIP model and the constraints

$$x_v^w \rightarrow \bigvee_{v' \leq \min \mathrm{orb}(v)} x_{v'}^{w_0} \qquad \forall v \in V(G), \forall w \in \mathrm{orb}(w_0) \tag{49}$$

$$x_v^{w''} \rightarrow \bigvee_{v' < v} x_{v'}^{w} \qquad \begin{array}{r} \forall v \in V(G), \forall w \in V(H), \\ \forall w'' \in \mathrm{orb}_{\{w' \in V(H): w' < w\}}(w) \setminus \{w\} \end{array} \tag{50}$$

for the SAT model.

3.1 Finding all Automorphisms and Stabilizers

To add constraints (47)-(48) or (49)-(50) to our model we need to compute the automorphism group $\mathrm{Aut}(G, \mathcal{T})$, its orbits, the automorphism group $\mathrm{Aut}(H, \mathcal{S})$, its orbits, and the orbits $\mathrm{orb}_{\{w' \in V(H): w' < w\}}(w)$ of the stabilizers for each $w \in V(H)$.

The problem of computing a set of generators of the automorphism group of a simple graph is well studied. It is closely related to the famous graph isomorphism problem. Since no polynomial time algorithm is known for the graph isomorphism problem, which can be reduced to computing generators of the automorphism group of the graph, all proposed algorithms in literature require exponential time in general. Nevertheless, if we restrict the problem to graphs with bounded degree, like it is the case for the input graph H, which is always 4-regular, there are polynomial time algorithms, see [9].

For the input graph G we do not have an upper bound on the degree. Nevertheless, there are algorithms for finding the automorphism group of general graphs, which work well in practice also for graphs with up to multiple thousand vertices, depending on the graph structure: see for example McKay and Piperno [10].

The algorithm of McKay and Piperno and also other algorithms in the literature working similarly get as an input a simple undirected graph $G = (V, E)$ with a vertex coloring $c : V \rightarrow \{1, \ldots m\}$ and return among other things a base B of $\mathrm{Aut}^c(G)$ and a strong generating set for $\mathrm{Aut}^c(G)$ relative to B.

Fig. 2: Construction example of the auxiliary graph for a part of a transitioned graph $(G, \mathcal{T})$. The newly added artificial vertices have color 2, which is drawn white and the original vertices have color 1, which is drawn black.

Definition 5. *Let G be a graph and c a vertex coloring of G. The subgroup*

$$\mathrm{Aut}^c(G) := \{f \in \mathrm{Aut}(G) \mid c(f(v)) = c(v) \quad \forall v \in V(G)\}$$

is called the color-preserving *automorphism group of G with respect to c.*

A base *B for $\mathrm{Aut}^c(G)$ is a sequence $B = (v_1, \dots v_k)$ of vertices of G such that the stabilizer $\mathrm{Aut}^c_{\{v_1,\dots,v_k\}}(G)$ is trivial, i.e. only contains the identity. A* strong generating set*(SGS) for $\mathrm{Aut}^c(G)$ relative to B is a set S of generating functions such that S generates $\mathrm{Aut}^c(G)$ and $S \cap \mathrm{Aut}^c_{\{v_1,\dots,v_k\}}(G)$ generates $\mathrm{Aut}^c_{\{v_1,\dots,v_k\}}(G)$ for all $1 \leq j < k$.*

Given a generator set of a group it is easy to compute the orbits of this group. Therefore, if we have a basis $B = (v_1, \dots, v_k)$ and a strong generating set relative to B we are able to compute the orbits $\mathrm{orb}_{v_1,\dots,v_j}$ of the stabilizers. Since we did not fix yet an ordering of the vertices in our input graphs G and H we can always use the partial ordering given by a basis B and extend it by putting all other vertices not occurring in the partial ordering in an arbitrary order after the vertices in B.

Since we need to compute automorphism groups of transitioned multigraphs, we need to transform our graphs in such a way that we can apply McKay's algorithm to it. Let $(G, \mathcal{T})$ be a transitioned graph. We construct an auxiliary graph $\mathrm{Aux}(G, \mathcal{T})$ by inserting in each edge $e = v_1v_2$ of G two vertices $w_e^{v_1}$ and $w_e^{v_2}$. This gives us immediately a simple graph. Furthermore, for each transition $t = (v, e_1, e_2) \in \mathcal{T}$ we add an edge between the vertices $w_{e_1}^v$ and $w_{e_2}^v$. We also define a coloring c on the auxiliary graph by coloring all original vertices with the color 1 and all artificially added vertices with the color 2. See Figure 2 for an example on how to construct the auxiliary graph for a part of a given transitioned graph $(G, \mathcal{T})$.

Theorem 1.

$$\mathrm{Aut}(G, \mathcal{T}) = \left\{f|_{V(G)} \mid f \in \mathrm{Aut}^c(\mathrm{Aux}(G, \mathcal{T}))\right\}$$

Proof. By adding the two artificial vertices with a second color between each edge we can associate with each automorphism in the auxiliary graph a vertex mapping and an edge mapping in the original graph. The edge mapping is defined by mapping an edge e_1 to an edge e_2 if the two artificial vertices on e_1 get mapped to the two artificial vertices on e_2 in the auxiliary graph. The edge mapping satisfies (1). Furthermore, since there are edges between two added vertices $w_{e_1}^v$ and $w_{e_2}^v$ if and only if there is a transaction $(v, \{e_1, e_2\})$ in the original graph we also get that the edge mapping satisfies (2). On the other hand, given a vertex mapping and an edge mapping satisfying (1) and (2), we can use those to formulate an edge-preserving vertex mapping on the auxiliary graph.

Corollary 1. *If G is a set of generators of $\mathrm{Aut}^c(\mathrm{Aux}(G,\mathcal{T}))$ then the set $G' := \{f|_{V(G)} \mid f \in G\}$ is a generator of $\mathrm{Aut}(G,\mathcal{T})$.*

Theorem 1 shows us that we can use the auxiliary graph $\mathrm{Aux}(G,\mathcal{T})$ to get the automorphism group of a transitioned graph $(G,\mathcal{T})$ by using an algorithm to compute $\mathrm{Aut}^c(\mathrm{Aux}(G,\mathcal{T}))$ and Corollary 1 shows us that we can use generators of $\mathrm{Aut}^c(\mathrm{Aux}(G,\mathcal{T}))$ to get the generators of $\mathrm{Aux}(G,\mathcal{T})$.

What remains is how to compute a basis and a strong generating set of $\mathrm{Aut}(G,\mathcal{T})$. Since the basis for $\mathrm{Aut}^c(\mathrm{Aux}(G,\mathcal{T}))$ may contain artificial vertices that are not in $V(G)$ we cannot use it for G. But since we already have a generating set for $\mathrm{Aut}(G,\mathcal{T})$ we can use any algorithm that computes, given a generating set of the group, a basis together with a strong generating set, such as the Schreier-Sims algorithm [12].

4 Computational Results

To test our SAT model and compare it with the MIP model proposed in [8] we implemented both in C++ using Glucose 4.1 to solve the SAT model and Gurobi 8.1 to solve the MIP model. We also tested the impact of the symmetry breaking constraints for both models. To get the automorphism groups as described in Section 3.1 we used nauty 2.6 [10] and to get a strong generating set we used the implementation of the Schreier-Sims algorithm contained in the nauty program. All tests were performed on a single core of an Intel Xeon E5-2640 v4 processor with 2.40GHz and 8GB RAM.

We consider the instance sets S1, S2, and G1 from [8] together with a new instance set G2 of larger random graphs to also test the limits of the SAT model. Set S1 consists of four random perfect matching contractions of all snarks with up to 26 vertices plus 1000 snarks with 28 vertices using the UD-K_5 as transitioned graph $(H,\mathcal{S})$. The set S2 consists of all PPM contractions of all snarks with up to 22 vertices. Furthermore, set G1 consists for each combination of $n \in \{9,\ldots,15\}$ and $m \in \{5,\ldots,7\}$ of ten instances, where each of those consists of a random 4-regular completely-transitioned graph G with n vertices and a random 4-regular completely-transitioned graph H with m vertices. The additional new instance set G2 is constructed the same way as G1 but with $n \in \{16,\ldots,30\}$ and $m \in \{6,\ldots,10\}$.

We compare the running times of four algorithms for the given instances, the original MIP model, the MIP model with the symmetry breaking constraints (47)-(48), which will be called $\mathrm{MIP}_{\mathrm{sym}}$, the SAT model, and the SAT model with the symmetry breaking constraints (49)-(50), which will be called $\mathrm{SAT}_{\mathrm{sym}}$.

Table 1 lists the computational results for instance set S1 for all four algorithms. The instances are grouped by the number of vertices $|V(C)|$ of the snark C used for the generation, one column per group. Column $|I|$ contains the numbers of instances, $|I_{\mathrm{feas}}|$ the numbers of feasible instances, and $|I_{\mathrm{inf}}|$ the numbers of infeasible instances. The time columns $t_{\mathrm{feas}}[s]$ and $t_{\mathrm{inf}}[s]$ list median running times of all feasible instances respectively the infeasible instances in seconds rounded to integer. Furthermore, for the MIP models columns I_{tl} contain the numbers of instances that could not be solved within the CPU-time limit of 3600 seconds. The best running times of the groups of feasible instances and infeasible instances are marked bold.

Table 1: Computation results for instance set S1.

				MIP			MIP_{sym}			SAT		SAT_{sym}	
$\lvert V(C)\rvert$	$\lvert I\rvert$	$\lvert I_{\text{feas}}\rvert$	$\lvert I_{\text{inf}}\rvert$	$t_{\text{feas}}[s]$	$t_{\text{inf}}[s]$	$\lvert I_{\text{tl}}\rvert$	$t_{\text{feas}}[s]$	$t_{\text{inf}}[s]$	$\lvert I_{\text{tl}}\rvert$	$t_{\text{feas}}[s]$	$t_{\text{inf}}[s]$	$t_{\text{feas}}[s]$	$t_{\text{inf}}[s]$
10	4	4	0	$< \mathbf{1}$	-	0	$< \mathbf{1}$	-	0	$< \mathbf{1}$	-	$< \mathbf{1}$	-
18	8	8	0	3	-	0	4	-	0	$< \mathbf{1}$	-	$< \mathbf{1}$	-
20	24	24	0	2	-	0	2	-	0	$< \mathbf{1}$	-	$< \mathbf{1}$	-
22	124	121	3	4	2035	0	5	727	0	$< \mathbf{1}$	11	$< \mathbf{1}$	**1**
24	620	604	16	8	3600	15	6	2955	2	$< \mathbf{1}$	26	$< \mathbf{1}$	**2**
26	5188	5124	64	12	3600	64	9	3600	58	$< \mathbf{1}$	78	$< \mathbf{1}$	**4**
28	4000	3970	30	19	3600	30	14	3600	30	$< \mathbf{1}$	166	$< \mathbf{1}$	**7**

Table 2: Computation results for instance set S2.

				MIP			MIP_{sym}			SAT		SAT_{sym}	
$\lvert V(C)\rvert$	$\lvert I\rvert$	$\lvert I_{\text{feas}}\rvert$	$\lvert I_{\text{inf}}\rvert$	$t_{\text{feas}}[s]$	$t_{\text{inf}}[s]$	$\lvert I_{\text{tl}}\rvert$	$t_{\text{feas}}[s]$	$t_{\text{inf}}[s]$	$\lvert I_{\text{tl}}\rvert$	$t_{\text{feas}}[s]$	$t_{\text{inf}}[s]$	$t_{\text{feas}}[s]$	$t_{\text{inf}}[s]$
18	98	15	83	2	194	0	1	9	0	**0.04**	0.12	**0.04**	**0.04**
20	1116	416	700	3	468	6	2	24	0	**0.05**	0.28	**0.05**	**0.06**
22	10694	4873	5821	4	1173	892	3	74	0	**0.06**	0.78	**0.06**	**0.08**

As we can see the SAT model outperforms the MIP model considerably and the symmetry breaking constraints improve the running times for the infeasible instances, especially for the SAT model but also for the MIP model.

To further compare the four models we applied a Wilcoxon signed-rank test for each pair of them using a p-value of 5%. The algorithm MIP_{sym} is significantly faster than MIP for the instance groups with $|V(C)| \geq 24$, for the infeasible but also for the feasible instances. The two SAT models are significantly faster than both MIP models for all instance groups except for $|V(C)| = 18$ and the infeasible instances of $|V(C)| = 22$ since those are too few to get a significant result. In fact the SAT models are faster on almost all instances except a few feasible instances. For the SAT model the variant without the symmetry breaking constraints is significantly faster on all feasible instance groups with $|V(C)| \geq 22$ although the difference in the values is only within hundredth of seconds. On the other hand for the infeasible instance groups with $|V(C)| \geq 24$ the model with the symmetry breaking constraints is significantly faster.

Table 2 shows the computational results for instance set S2. The columns are the same as in Table 1. The results are similar as for instance set S1, but this time MIP_{sym} can solve all instances within the time limit. Applying the Wilcoxon signed-rank test we get that MIP_{sym} is significantly faster than MIP except for the infeasible instance group with $|V(C)| = 18$. Both SAT models are significantly faster than the MIP models for all instance groups. This time SAT is not significantly faster than SAT_{sym} on the feasible instance groups, SAT_{sym} is even significantly faster than SAT for the feasible instance group with $|V(C)| = 22$. For the infeasible instances SAT_{sym} is significantly faster.

Table 3: Computation results for instance set G1.

					MIP		MIP_{sym}		SAT	SAT_{sym}
$\lvert V(G)\rvert$	$\lvert V(H)\rvert$	$\lvert I\rvert$	$\lvert I_{\text{feas}}\rvert$	$\lvert I_{\text{inf}}\rvert$	$t[s]$	$\lvert I_{\text{tl}}\rvert$	$t[s]$	$\lvert I_{\text{tl}}\rvert$	$t[s]$	$t[s]$
09	5	30	15	15	106	0	91	0	$< \mathbf{1}$	$< \mathbf{1}$
09	6	30	4	26	440	1	409	0	1	$< \mathbf{1}$
09	7	30	0	30	2059	11	2735	14	$< \mathbf{1}$	$< \mathbf{1}$
10	5	30	19	11	90	1	87	1	$< \mathbf{1}$	$< \mathbf{1}$
10	6	30	4	26	1939	12	1862	10	**1**	**1**
10	7	30	0	30	3600	16	3600	16	2	**1**
11	5	30	25	5	42	1	19	0	$< \mathbf{1}$	$< \mathbf{1}$
11	6	30	9	21	2777	14	3600	16	3	**2**
11	7	30	1	29	3600	22	3600	20	**3**	**3**
12	5	30	28	2	50	1	17	0	$< \mathbf{1}$	$< \mathbf{1}$
12	6	30	21	9	2204	13	2124	11	3	**2**
12	7	30	1	29	3600	30	3600	30	8	**7**
13	5	30	28	2	23	2	26	2	$< \mathbf{1}$	$< \mathbf{1}$
13	6	30	20	10	3600	17	2055	13	**4**	**4**
13	7	30	7	23	3600	30	3600	27	14	**13**
14	5	30	30	0	24	0	30	0	$< \mathbf{1}$	$< \mathbf{1}$
14	6	30	28	2	562	7	823	8	**2**	**2**
14	7	30	8	22	3600	29	3600	27	**27**	28
15	5	30	30	0	30	0	24	0	$< \mathbf{1}$	$< \mathbf{1}$
15	6	30	29	1	670	2	1475	11	3	**2**
15	7	30	18	12	3600	26	3600	27	**27**	30

Table 3 shows the computation results for the instance set G1. We group the instances by the number of vertices of the input graphs G and H. We do not distinguish between feasible and infeasible instance groups in this table, since the running time characteristics are similar for both types of instances. Columns $t[s]$ show the median running time for all instances of the instance group. Again both SAT models could solve all instances within one hour and outperforms the MIP models. This time the differences between the models with symmetry breaking constraints and without are smaller, since the probability that a random graph has symmetries is small. Now the SAT models are on all instances faster than the MIP models. Between the MIP models there are only few instance groups where there is a significant difference in the running times in favor of both models. The situation between the two SAT models is similar although there are slightly more instance groups where SAT_{sym} is significantly faster.

All instances in all three instance sets S1, S2, and G1 could be solved within the time limit of one hour by both SAT models. To also analyze the limits of our SAT models we also tested instance set G2. Figure 3 shows the median running times of the SAT_{sym} model for different sizes of $|G|$ and $|H|$. As we can see the running time heavily depends on the size of H and not so strongly on the size of G. For $|H| = 10$ and $|G| \geq 20$ we run into the time limit of one hour in most of the instances. Similarly, as for instance set $G1$ also in $G2$ the running times for SAT_{sym} and SAT are similar.

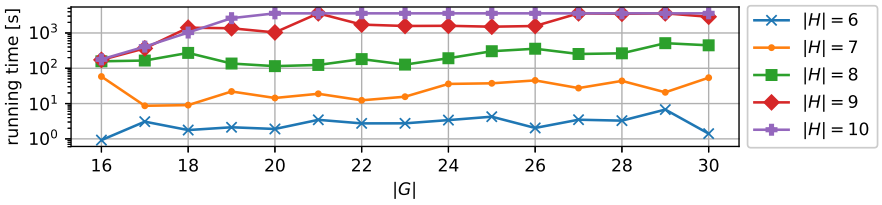


Fig. 3: Median running times of SAT_{sym} for instance set G2.

Using SAT_{sym} we also implemented the framework described at the end of Section 2. We use Boosts implementation of the Boyer-Myrvold planarity test to check for planar graphs. Furthermore, we use a simple SAT model for checking if a graph contains a K_5-minor and another SAT model for checking if it has a CCD. Since the bottleneck of this framework are the solving times for checking SUD-K_5-minor freeness, the running time improvements by the SAT model were crucial to check for all snarks with up to 32 vertices if they contain a planar contraction, a K_5-minor free contraction, a SUD-K_5-minor free contraction, or a CCD-containing contraction of a PPM. From the 1 918 812 tested snarks we found 25 248 snarks that do not contain a planar contraction of a PPM, 19 130 snarks that do not contain a K_5-minor free contraction of a PPM, and 1 095 snarks that do not contain a SUD-K_5-minor free contraction of a PPM.

Up until now it was not known if there exist snarks that do not have a PPM whose contraction leads to planar/K_5-minor free/SUD-K_5-minor free graphs. With our implementation we could find many examples of snarks that have those properties. Nevertheless, all tested snarks always had a PPM whose contraction leads to a graph which has a CCD. Therefore, it remains an open question if there exists a snark that does not have a PPM whose contraction leads to a CCD-containing graph.

5 Conclusion and Future Work

In this work we proposed a SAT model for checking if a given transitioned graph $(G, \mathcal{T})$ has a Sup-$(H, \mathcal{S})$-transition minor. The model is based on the mathematical model developed in a previous work [8]. To improve the performance of the SAT model, but also of the MIP model we developed symmetry breaking constraints that are based on the automorphism groups of both input graphs restricted by the additional structure given through the transition systems. In our computational study we could verify that the SAT model outperforms the MIP model significantly and the symmetry breaking constraints could improve the running times especially for proving infeasibility. Using the SAT model in a framework we were able to find many snarks that do not have PPM whose contraction leads to SUD-K_5-minor free graphs.

In future work it may be interesting to consider a CP model for our problem to be able to use non-binary variables in the model for representing the mappings between the two input graphs. Furthermore, the framework for finding snarks that do not contain a SUD-K_5-minor free contraction of a PPM may be improved by adding symmetry breaking during the enumeration of the PPMs.

References

1. F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah. Solving Difficult SAT Instances in the Presence of Symmetry. In *Proceedings of the 39th Annual Design Automation Conference*, DAC '02, pages 731–736, New York, NY, USA, 2002. ACM.
2. G. Fan and C.-Q. Zhang. Circuit Decompositions of Eulerian Graphs. *Journal of Combinatorial Theory, Series B*, 78(1):1–23, 2000.
3. H. Fleischner. Eulersche Linien und Kreisüberdeckungen, die vorgegebene Durchgänge in den Kanten vermeiden. *Journal of Combinatorial Theory, Series B*, 29(2):145–167, 1980.
4. H. Fleischner, B. Bagheri Gh., C.-Q. Zhang, and Z. Zhang. Cycle covers (III) - Compatible circuit decomposition and K5-transition minor. Technical report, Algorithms and Complexity Group, TU Wien, 2018.
5. F. Jaeger. A survey of the cycle double cover conjecture. In B.R. Alspach and C. D. Godsil, editors, *Annals of Discrete Mathematics (27): Cycles in Graphs*, volume 115 of *North-Holland Mathematics Studies*, pages 1–12. North-Holland, 1985.
6. M. Janota, R. Grigore, and V. Manquinho. On the Quest for an Acyclic Graph. *arXiv:1708.01745 [cs]*, 2017. arXiv: 1708.01745.
7. T. Januschowski and M. E. Pfetsch. Branch-Cut-and-Propagate for the Maximum k-Colorable Subgraph Problem with Symmetry. In T. Achterberg and J. C. Beck, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Lecture Notes in Computer Science, pages 99–116. Springer Berlin Heidelberg, 2011.
8. B. Klocker, H. Fleischner, and G. Raidl. A Model for Finding Transition-Minors. Technical report, Algorithms and Complexity Group, TU Wien, 2018.
9. E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.
10. B. D. McKay and A. Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60(0):94 – 112, 2014.
11. P. D. Seymour. Sums of circuits. *Graph theory and related topics*, 1:341–355, 1979.
12. C. C. Sims. Computational methods in the study of permutation groups. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 169–183. Pergamon, 1970.
13. G. Szekeres. Polyhedral decompositions of cubic graphs. *Bulletin of the Australian Mathematical Society*, 8(03):367, 1973.
14. G. S. Tseitin. On the Complexity of Derivation in Propositional Calculus. In J. H. Siekmann and G. Wrightson, editors, *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, Symbolic Computation, pages 466–483. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983.