Searching for Smallest Universal Graphs and Tournaments with SAT

Tianwei Zhang
Algorithms and Complexity Group, TU Wien, Austria

Stefan Szeider
Algorithms and Complexity Group, TU Wien, Austria

Abstract

A graph is induced k-universal if it contains all graphs of order k as an induced subgraph. For over half a century, the question of determining smallest k-universal graphs has been studied. A related question asks for a smallest k-universal tournament containing all tournaments of order k.

This paper proposes and compares SAT-based methods for answering these questions exactly for small values of k. Our methods scale to values for which a generate-and-test approach isn't feasible; for instance, we show that an induced 7-universal graph has more than 16 vertices, whereas the number of all connected graphs on 16 vertices, modulo isomorphism, is a number with 23 decimal digits Our methods include static and dynamic symmetry breaking and lazy encodings, employing external subgraph isomorphism testing.

2012 ACM Subject Classification Mathematics of computing → Extremal graph theory; Software and its engineering → Constraint and logic languages; Hardware → Theorem proving and SAT solving; Mathematics of computing → Graph enumeration

Keywords and phrases Constrained-based combinatorics, synthesis problems, symmetry breaking, SAT solving, subgraph isomorphism, tournament, directed graphs

Digital Object Identifier 10.4230/LIPIcs.CP.2023.39

Supplementary Material *Software*: `https://doi.org/10.5281/zenodo.8147732`

Funding The project leading to this publication has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101034440, and was supported by the Vienna Science and Technology Fund (WWTF) within the project ICT19-065.

Acknowledgements This work was carried out in part while the second author visited the Simons Institute for the Theory of Computing, University of Berkeley, within the program *Extended Reunion: Satisfiability*. The authors thank Ciaran McCreesh for advising them to use and modify the Glasgow subgraph solver.

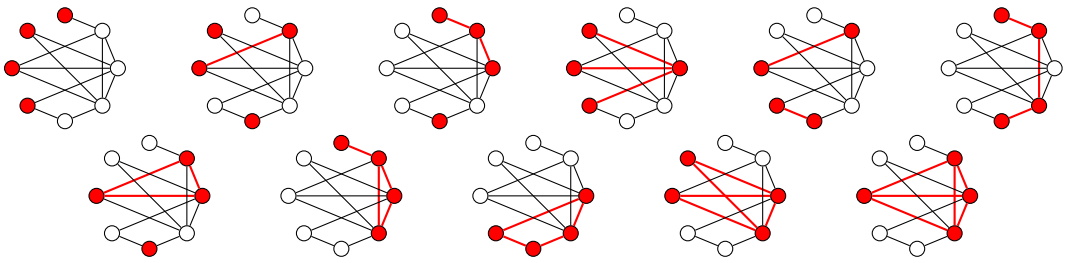

Figure 1 A smallest induced 4-universal graph and how all 11 graphs of order 4 embeds into it.

Figure 2 A smallest 4-universal tournament and how all 4 tournaments of order 4 embeds into it.

1 Introduction

A graph G is *induced k-universal* if it contains all k-vertex graphs as induced subgraphs; in other words, any k-vertex graph can be obtained from G by deleting some of G's vertices. This notion of a universal graph, which extends naturally to directed graphs and tournaments (orientations of complete graphs) was introduced by Rado in 1964 [25] and has since then attracted much attention in combinatorics. Figures 1 and 2 provide examples of a universal graph and a universal tournament, respectively. The fundamental question is to determine for each integer $k \geq 1$ the smallest n such that an induced k-universal graph with n vertices exists; this number is denoted $f(k) = n$. In 1965, Moon [21] obtained the bounds $2^{(k-1)/2} \leq f(k) \leq O(k \cdot 2^{k/2})$; Alstrup et al. [2] recently improved the upper bound to $f(k) \leq 16 \cdot 2^{k/2}$. The tight asymptotic upper bound of $f(k) \leq (1 + o(1))2^{(k-1)/2}$ is due to Alon [1].

In contrast to these general bounds and asymptotic results, very little is known about exact $f(k)$ values, even for small k. Trimble [26] carried out a brute-force search, expanding an approach suggested by Preen [24], and could determine the values of $f(k)$ for $k \leq 6$ and the interval $16 \leq f(7) \leq 18$. The brute-force search enumerates all candidate n-vertex graphs up to isomorphism and tests for each of them whether it contains all the k-vertex graphs as induced subgraphs. We refer to the k-vertex graphs that need to be checked as induced subgraphs as *pattern graphs*. Tools exist for the isomorph-free enumeration of all connected n-vertex graphs (we can assume that a candidate graph is connected), but the number of such graphs exceeds eleven million for $n = 10$, and one billion for $n = 11$ [23, A001349]. Thus, the feasibility of the brute-force approach quickly hits a rigid boundary. In particular, we cannot tighten the gap $16 \leq f(7) \leq 18$ in this way, as enumerating all graphs with 16 vertices is far out of reach, although we can easily enumerate all the 7-vertex pattern graphs; the exact number is 1044 [23, A000088].

This paper proposes new methods that allow us to break this boundary. The idea is to formulate the problem as a *synthesis problem* for the universal graph rather than the easier problem of testing a candidate graph for being universal. This way, we can avoid the bottleneck of enumerating all candidate graphs. We formulate the synthesis problem in propositional logic to harness the power of solvers for the propositional satisfiability problem (SAT) [8]. Suppose the sought-for universal graph G has the numbers $1, \ldots, n$ as its vertices. For each pair $1 \leq i < j \leq n$, we introduce a propositional variable $e_{i,j}$ (an *edge variable*) whose truth value determines whether there is an edge between vertices i and j in G.

We propose and compare various strategies and techniques of encoding the property of G being induced k-universal. Each of the encodings must combine two fundamental properties: (i) to ensure that the found graph is indeed induced k-universal and (ii) to break symmetries to minimize the enormous search space. We have two fundamentally different approaches for properties (i) and (ii).

Our first approach for ensuring that the found graph is indeed induced k-universal uses a *direct SAT encoding*. For the relevant cases, the number of pattern graphs is reasonably small, and we can enumerate them up to isomophism. For each pattern graph, we obtain

a formula which constrains the edge variables in such a way that the sought-for universal graph contains an induced subgraph that is isomorphic to the pattern graph. A conjunction over the formulas, one for each pattern graph, yields the SAT encoding.

Our second approach for ensuring universality utilizies the *Glasgow subgraph solver (GSS)* [19], a powerful constraint-based tool for testing subgraphs. However, during the SAT solver's search we don't have a candidate graph available for testing whether it accommodates all the required induced subgraphs. A partial truth assignment instead sets a subset of the edge variables to true or false, with some edge variables remaining undecided. Thus, a partial truth assignment gives rise to a *partially defined graph*. Our objective is to find out already for a partially defined graph whether it could be extended to a fully defined induced k-universal graph. Indeed, we could achieve this by accessing GSS's internal functionality.

Our first approach to breaking symmetries utilizes the *SAT modulo Symmetries (SMS)* framework [17]. Also SMS works on partially defined graphs as represented by a partial truth assignment to edge variables. Whenever the solver determines an edge, the partially defined graph represented by the current partial truth assignment is sent to an external propagator, which determines whether the partially defined graph can be extended to a fully defined graph that is canonical (i.e., a unique graph within its ismorphism class). If not, a clause is sent back to the solver.

Our second approach to symmetry breaking is based on the concept of *templates* which serve two purposes: symmetry breaking and search space partition. In an offline phase, we fix a small collection of highly symmetric pattern graphs and compute all possible ways these pattern graphs can interact as induced subgraphs in the universal graph, up to isomorphism. Each possible interaction gives rise to a template. By hardcoding a template, we get a formula that is satisfiable if and only if there exists an induced k-universal graph on n vertices in which the fixed collection of pattern graphs interact as prescribed in the template. $f(k) \leq n$ if an only if for at least one of the templates, the corresponding formula is satisfiable. In addition to the symmetry breaking aspect, the templates approach allows us to parallelize the search, running SAT calls for various templates independently.

We also evaluate our SAT-based approach to determining smallest *k-universal tournaments*. A tournament is a directed graph that can be obtained from a complete undirected graph by orienting its edges. The name tournament originates from such a directed graph's interpretation as the outcome of a round-robin tournament in which every player encounters every other player exactly once, and in which no draws occur. Tournaments are a central combinatorial object whose properties have been intensively studied. The notion of universality applies to tournaments in a natural way: we say that a tournament is *k-universal* if it contains each tournament on k-vertices as a subgraph. Let $t(k)$ denote the smallest number of vertices a k-universal tournament can have. The study of k-universal tournaments goes back more than 50 years to Moon's book (entirely dedicated to tournaments [22]) where he determined the bounds $2^{(k-1)/2} \leq t(k) \leq O(k \cdot 2^{k/2})$. The improvements on the bounds parallel the ones for induced k-universal graphs: $f(k) \leq 16 \cdot 2^{\lceil k/2 \rceil}$ by Alstrup et al. [2] and $f(k) \leq (1 + o(1))2^{(k-1)/2}$ by Alon [1].

The situation regarding exact values for $t(k)$ is even worse than for $f(k)$. While the number of tournaments up to isomorphism is for all $k \leq 19$ [23, A000568], very little can be found on the value of $t(k)$ in the literature. The only obvious values are $t(1) = 1$ and $t(2) = 2$. In Moon's book [22], it is left as an exercise to the reader to determine the value of $t(3)$ and $t(4)$, indicating that these numbers might be easily deduced.

We implemented our SAT-based approaches to obtain new results on smallest induced k-universal graphs and smallest k-universal tournaments. In particular, we could improve the known lower bound for the order of an induced 7-universal graph by showing that no induced

Table 1 Overview of results along the order k of the pattern graphs or tournaments. $\mathcal{G}(k)/\mathcal{T}(k)$ give the number of pattern graphs/tournaments of order k, $f(k)/t(k)$ give the order of a smallest (induced) k-universal graph or tournament, and $F(k)/T(k)$ their number modulo ismomorphism, respectively. New results are indicated in bold face.

	Graphs			Tournaments		
k	$\mathcal{G}(k)$	$f(k)$	$F(k)$	$\mathcal{T}(k)$	$t(k)$	$T(k)$
1	1	1	1	1	1	1
2	2	3	2	1	2	1
3	4	5	5	2	4	**3**
4	11	8	438	4	5	**1**
5	34	10	22	12	**8**	**1643**
6	156	14	>**36294**	56	**10**	**1088**
7	1044	[**17**,18]	–	456	[**13**,**15**]	–

7-universal graph with 16 vertices exists, leaving the possibilities 17 and 18 for the smallest solution, and we could determine the exact order of smallest k-universal tournaments up to $k = 6$ and leaving the possibilities 13,14, and 15 for the smallest solution for $k = 7$. We also determined the precise number of optimal solutions for k-universal tournaments for $k \leq 6$, up to isomorphism, which is of independent interest for combinatorial research. Our main results are summarized in Table 1, smallest k-universal tournaments that we identified are shown in Figures 6 and 7 in the appendix.

2 Preliminaries

We denote the set $\{1, \ldots, n\}$ by $[n]$ and the set $\{m, m+1, \ldots, n\}$ by $[m, n]$ for $m \leq n$.

2.1 Graphs

We consider simple undirected graphs G, denoting the vertex set and the edge set of G by $V(G)$ and $E(G)$, respectively. The *order* or a graph is the number $|V(G)|$ of its vertices. An edge between vertices $u, v \in V(G)$ is denoted uv or equivalently vu. A graph G' is a *subgraph* of a graph G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. A subgraph G' of G is *induced* if for any $u, v \in V(G')$, $uv \in E(G)$ implies $uv \in E(G')$. Thus, an induced subgraph is determined by its set of vertices.

The *neighborhood* of a vertex $u \in V(G)$ is denoted $N_G(u)$, i.e., $N_G(u) := \{\, x \in G \mid ux \in E(G) \,\}$. Two vertices $v, v' \in G$ are *twins* if they have the exact same neighbors excluding each other, i.e., $N_G(u) \setminus \{v\} = N_G(v) \setminus \{u\}$.

The *complement graph* $\overline{G}$ of a graph G has $V(\overline{G}) = V(G)$ and $E(\overline{H}) := \{\, uv \mid u \neq v \in V(H), uv \notin E(H) \,\}$.

We denote the complete graph of order k by K_k, the complete bipartite graph by $K_{l,r}$.

A *monomorphism* from H to G is an injective mapping $\phi : V(H) \to V(G)$ such that for all $u, v \in V(H)$ we have $\phi(u)\phi(v) \in E(G)$ if $uv \in E(H)$. Such a monomorphism ϕ is *faithful* if for all $u, v \in V(H)$ we have $uv \in E(H)$ if $\phi(u)\phi(v) \in E(G)$ [11]. We say H *embeds into* G if there is a faithful monomorphism from H to G. A *bijective* faithful monomorphism is an *isomorphism*. We refer to H as the *pattern graph* and G as the *target graph*. For a monomorphims ϕ from H to G we put $\phi(V(H)) := \{\, \phi(v) \mid v \in V(H) \,\}$.

Let $\mathcal{G}(k)$ denote the class of all graphs of order k up to isomorphism. A graph G is *induced k-universal* if all $H \in \mathcal{G}(k)$ embed into G. $f(k)$ is the smallest order of an induced k-universal graph, and $F(k)$ is the number of non-isomorphic k-universal graphs of order $f(k)$.

2.2 Partially Defined Graphs

The notion of partially defined graphs was introduced in the context of SMS [17], for capturing the combinatorial object represented by a partial truth assignment on the edge variables. In this paper, we utilize this concept for two further purposes: to check with an external subgraph solver whether the current branch of the search has the potential of success (Section 4.5) and for the formulation of our template method for symmetry breaking and search partition (Section 4.6).

A *partially defined graph* is a graph where some edges are undefined in the sense that their presence in the graph is open. Formally, a partially defined graph G is a graph whose edge set $E(G)$ is split into disjoint sets $E_d(G)$ and $E_u(G)$, the sets of defined and undefined edges, respectively. G is *fully defined* if $E_u(G) = \emptyset$. For a partially defined graph G, $\mathcal{X}(G)$ denotes the set of all fully defined graphs G can be extended to, i.e., $G' \in \mathcal{X}(G)$ if and only if $V(G') = V(G)$, $E_d(G) \subseteq E(G') \subseteq E(G)$.

We extend the notion of a faithful monomorphism and an isomorphism to partially defined graphs. Let H, G be partially defined graphs. An injective mapping $\phi : V(H) \to V(G)$ is a *faithful monomorphism* from H to G if for all $u, v \in V(H)$ it holds that

1. $uv \in E_d(H)$ if and only if $\phi(u)\phi(v) \in E_d(G)$, and
2. $uv \in E_u(H)$ if and only if $\phi(u)\phi(v) \in E_u(G)$.

If there is a faithful monomorphism form H to G we say that H embeds into G. We say that H and G are *isomorphic* if there is a *bijective* faithful monomorphism from H to G.

2.3 Directed Graphs and Tournaments

We denote a directed graph (or digraph) D by its vertex set $V(D)$ and arc set $A(D) \subseteq V(D) \times V(D)$. A digraph D is an *oriented graph* if it has no directed digon $\{(u, v), (v, u)\}$. (Induced) subdigraphs, isomorphisms, and faithful monomorphisms are defined for digraphs analogously to graphs. A *tournament* is an oriented graph with a maximal number of arcs, i.e., adding any further arc creates a digon. A tournament is *transitive* if it contains no directed cycles. A tournament D' is a subtournament of tournament D if it is a subdigraph of D. If a tournament H is isomorphic to a subtournament of D, then we say that H embeds into D. Hence H embeds into D if and only if there exists a faithful monomorphism $\phi : V(H) \to V(D)$.

Let $\mathcal{T}(k)$ denote the class of all tournaments of order k up to isomorphism. A tournament D is *k-universal* if all $H \in \mathcal{T}(k)$ embed into D. Let $T(k)$ denote the number of non-isomorphic tournaments of order $t(k)$ that are induced k-universal tournaments.

2.4 SAT

We consider propositional formals in conjunctive normal form (CNF). A CNF formula is a conjunction of *clauses*, each clause is a disjunction of *literals*, a literal is a propositional variable x or its negation $\neg x$. A propositional formula F is *satisfiable* if there exits a mapping τ that assigns each variable a truth value $\in \{0, 1\}$ such that each clause contains a literal x with $\tau(x) = 1$ or a literal $\neg x$ with $\tau(x) = 0$. A truth assignment is *partial* for a CNF

formula if it is defined only for a subset of the formula's variables. SAT solvers are tools that decide whether a given CNF formula is satisfiable or not [8]. Today's most powerful complete SAT solvers follow the conflict driven clause learning (CDCL) paradigm. Modern CDCL SAT solvers like Cadical [4] can certify the unsatisfiability of a CNF formula by producing DRAT proofs (deletion, reverse asymmetric tautology) that can be independently checked and verified [12, 27].

3 A Lower Bound for Induced k Universal Graphs

Trimble [26] showed that there is no graph of order less than $2k+2$ that is universal for $\{K_k, \overline{K_k}, K_{3,3}, \overline{K_{3,3}}\}$, for all $k \geq 6$. We extend this argument to show the following more general proposition, which gives a strictly tighter lower bound for $k \geq 8$.

▶ **Proposition 1.** *$2k + 2\lfloor \frac{k}{2} \rfloor - 4 \leq f(k)$ for all $k \geq 2$.*

Proof. Suppose G is an induced k-universal graph. Graphs K_k and $\overline{K_k}$ are both elements of $\mathcal{G}(k)$, and therefore there exist faithful monomorphisms $\phi_K : V(K_k) \to V(G)$ and $\phi_I : V(\overline{K_k}) \to V(G)$. Let $S_1 := \{\, \phi_K(v) \mid v \in V(K_k) \,\}$, $S_2 := \{\, \phi_I(v) \mid v \in V(\overline{K_k}) \,\}$ and $S3 := V(G)/(S_1 \cup S_2)$. Clearly, the subgraphs induced by S_1 and S_2 are a clique and an independent set, respectively, and therefore may overlap by no more than one vertex. This implies that $|S_1 \cup S_2| \geq 2k - 1$ and $|S_3| \leq n - 2k + 1$. Define $H := K_{\lfloor \frac{k}{2} \rfloor, \lfloor \frac{k}{2} \rfloor + (k \bmod 2)}$. Let G_1 be an induced subgraph of G that is isomorphic to H. Since there are no edges between the two cliques of G_1, it must be the case that at least one of the two cliques of G_1 does not intersect S_1. Denote this clique by C. Since S_2 is an independent set in G, C can only intersect S_2 by at most one vertex. Since C has at least $\lfloor \frac{k}{2} \rfloor$ vertices, it follows that C intersects S_3 by at least $\lfloor \frac{k}{2} \rfloor - 1$ vertices. In other words, the subgraph induced by S_3 contains a clique D of size $\lfloor \frac{k}{2} \rfloor - 1$.

Now consider the graph $\overline{H}$. Since $\overline{H}$ is an induced subgraph of G, it follows that $H = \overline{\overline{H}}$ is an induced subgraph of $\overline{G}$. We can repeat the argument of the previous paragraph with the roles of S_1 and S_2 reversed to show that the subgraph induced by S_3 contains an independent set of size $\lfloor \frac{k}{2} \rfloor - 1$. Since this independent set can overlap with the clique D in at most one vertex, it follows that the minimal size of S_3 is $2(\lfloor \frac{k}{2} \rfloor - 1) - 1$. In other words, for any k-universal graph of order n where $S_1 \cap S_2 \neq \emptyset$, $n - 2k + 1 \geq 2(\lfloor \frac{k}{2} \rfloor - 1) - 1$, which implies that $f(k) \geq 2k + 2\lfloor \frac{k}{2} \rfloor - 4$. Hence, the proposition is shown. ◀

Proposition 1 gives us $f(2) \geq 2$, $f(3) \geq 4$, $f(4) \geq 8$, $f(5) \geq 10$, $f(6) \geq 14$ and $f(7) \geq 16$. Note that for $k = 4, 5, 6$, the lower bound given by the proposition is also the exact value of $f(k)$. Even though it does not give a tighter lower bound for $f(7)$, it does improve on the existing knowledge for $k \geq 8$.

▶ **Corollary 2.** *Let G be a k-universal graph. If there exist faithful monomorphisms $\phi_K : V(K_k) \to V(G)$ and $\phi_I : V(\overline{K_k}) \to V(G)$ with $\phi_K(V(K_k)) \cap \phi_I(V(\overline{K_k})) = \emptyset$, then the order of G is at least $2k + 2\lfloor \frac{k}{2} \rfloor - 3$.*

4 Encodings for Induced k universal Graphs

Throughout this section, we fix an integer n and consider a potential universal graph G with $V(G) = [n]$ whose edges are represented by *edge variables* $e_{u,v}$, $1 \leq u < v \leq n$.

4.1 Encoding Universality

First, we define a CNF formula $M(H,n)$ that ensures that $(m_{u,v})_{u\in V(H),v\in V(G)}$ encodes an injective mapping from $V(H)$ to $V(G)$.

$$M(H,n) := \bigwedge_{u\in V(H)} \Big(\bigvee_{x\in[n]} m_{u,x}\Big) \wedge \bigwedge_{\substack{u\neq v\in V(H)\\ x\in[n]}} (\neg m_{u,x} \vee \neg m_{v,x}) \wedge \bigwedge_{\substack{u\in V(H)\\ x\neq y\in[n]}} (\neg m_{u,x} \vee \neg m_{u,y})$$

Now, we specify a CNF formula $F(H,n)$ which is satisfiable if and only if H embeds into G. $F(H,n)$ encodes the existence of a faithful monomorphism ϕ from H to G in terms of variables $m_{u,v}$, $u \in V(H)$, $v \in V(G)$, which are true if and only if $\phi(u) = v$.

$$F(H,n) := M(H,n) \wedge \bigwedge_{\substack{uv\in E(H)\\ 1\leq x<y\leq n}} (\neg m_{u,x} \vee \neg m_{u,y} \vee e_{x,y}) \wedge \bigwedge_{\substack{u\neq v\in V(H)\\ \text{s.t. } uv\notin E(H)\\ 1\leq x<y\leq n}} (\neg m_{u,x} \vee \neg m_{v,y} \vee \neg e_{x,y}).$$

The second and third conjunct ensure that the mapping preserves edges and non-edges, respectively. The formula

$$U(k,n) = \bigwedge_{H\in\mathcal{G}(k)} F(H,n)$$

is satisfiable if and only if there exists a k-universal graph on n vertices. From a satisfying assignment we can read off a k-universal graph.

4.2 Symmetry-breaking Based on Twins

A basic symmetry-breaking can already be achieved by observing the symmetries of a pattern graph $H \in \mathcal{G}(k)$. Specifically, if $v, v' \in V(H)$ are twins and ϕ is a faithful monomorphism from H to G, then $\phi' : V(H) \to V(G)$ defined as

$$\phi'(x) := \begin{cases} \phi(v') & \text{if } x = v, \\ \phi(v) & \text{if } x = v', \\ \phi(x) & \text{otherwise,} \end{cases}$$

is also a faithful monomorphism. This means that we can always require that the faithful monomorphism encoded by $(m_{u,v})_{u\in V(H),v\in V(G)}$ preserves the order of vertices (seen as integers) for twins. The following CNF formula encodes this additional requirement.

$$\mathrm{Twin}(k) := \bigwedge_{\substack{H\in\mathcal{G}(k)\\ u<v\in V(H)\\ \text{s.t. } u,v \text{ are twins}\\ 1\leq x<y\leq n}} \neg m_{v,x} \vee \neg m_{u,y}$$

4.3 Embedding K_k and $\overline{K_k}$

From preliminary experiments we observed that fixing (or "hardcoding") some edges/non-edges of the target graph can drastically speed up the solving process. Trimble [26] hardcodes K_k and $\overline{K_k}$ in his experiments, and we agree that it is a good place to start. Since the two subgraphs can have at most one overlapping vertex, we have the following two formulas, each hardcoding one of the two possibilities. We assume $n \geq 2k$.

$$\mathrm{KI}_{\mathrm{disjoint}}(k) := \bigwedge_{1\leq x<y\leq k} e_{x,y} \wedge \neg e_{x+k,y+k}, \quad \mathrm{KI}_{\mathrm{overlap}}(k) := \bigwedge_{1\leq x<y\leq k} e_{x,y} \wedge \neg e_{x+k-1,y+k-1}.$$

In Section 4.6, we will extend this idea by hardcoding more pattern graphs.

To sum up, the following two encodings are shared by all methods.

$$U_d(k,n) := U(k,n) \wedge \mathrm{Twin}(k) \wedge \mathrm{KI}_{\mathrm{disjoint}}(k),$$
$$U_o(k,n) := U(k,n) \wedge \mathrm{Twin}(k) \wedge \mathrm{KI}_{\mathrm{overlap}}(k).$$

Note that $U(k,n)$ is satisfiable if and only if $U_d(k,n)$ or $U_o(k,n)$ is satisfiable.

4.4 SAT Modulo Symmetries (SMS)

If a SAT solver checks the satisfiability of $U_d(k,n)$ or $U_o(k,n)$, it does not break symmetries and considers isomorphic copies of graphs implicitly represented by the edge variables, making the approach impractical. Fortunately, the SAT modulo Symmetries (SMS) framework [17] offers a solution.

For two (fully defined) graphs G_1, G_2, let $G_1 \preceq G_2$ if and only the adjacency matrix of G_1 is lexicographically smaller or equal to the adjacency matrix of G_2 (we consider the matrix as the string obtained by concatenating the rows of the matrix).

SMS works as follows. Whenever the SAT solver decides on an edge variable $e_{u,v}$, the partially defined graph G represented by the current truth assignment on the edge variables is sent to an external propagator. With a certain pre-determined probability the propagator checks a necessary condition for $\mathcal{X}(G)$ containing a $\preceq$-minimal graph. If the condition is not met, then this branch of the search can be terminated: a clause that excludes G is learned and sent back to the SAT solver.

In our context, while the satisfiability of $U(k,n)$ and $\mathrm{Twin}(k)$ are unaffected by our choice of the $\preceq$-minimal graph as the representative target graph for each isomorphic class, $\mathrm{KI}_{\mathrm{disjoint}}(k)$ and $\mathrm{KI}_{\mathrm{overlap}}(k)$ might cause an otherwise satisfying assignment to fail the minimality check.

The solution to this is to take advantage of the following functionality of SMS. When SMS checks the $\preceq$-minimality of a partially defined graph G, one can specify a partition P of the vertex set such that $\preceq$-minimality is only checked among all partially defined graphs that can be obtained from G by permuting vertices within an equivalence class of P. For $U_d(k,n)$, we give SMS the partition $\{[k],[k+1,2k],[2k+1,n]\}$, and for $U_o(k,n)$, we give the partition $\{[k-1],[k,k],[k+1,2k-1],[2k,n]\}$. The disadvantage of this adaption is that we are not able to trim down search branches as efficiently, but the advantage is that we can enjoy the speed-up brought by hardcoding a part of the target graph.

4.5 External Subgraph Isomorphism Testing

Next we lay out a lazy encoding strategy where we check during the SAT solver's run whether the partially defined graph represented by the current partial assignment to the edge variables can be extended to a k-universal graph. We utilize the *Glasgow subgraph solver (GSS)* [19] to achieve this. The GSS is a state-of-the-art solver for the subgraph isomorphism problem based on constraint programming. Given a pattern graph H and a target graph G, the subgraph isomorphism problem asks whether there exists a monomorphism from H to G. One of the key concepts underlying the GSS's functionality is to generate auxiliary graphs $H_1,\ldots,H_m$ with $V(H) = V(H_i), 1 \le i \le m$, and $G_1,\ldots,G_m$ with $V(G) = V(G_i), 1 \le i \le m$, such that a mapping from $V(G)$ to $V(H)$ is a (faithful) monomorphisms from H to G if and only if it is a monomorphisms from H_i to G_i, for every $1 \le i \le m$. Now the solver searches for a mapping from $V(G)$ to $V(H)$ that is a monomorphism from H_i to G_i, for every $1 \le i \le m$. For instance, the GSS can decide whether H is isomorphic to an *induced* subgraph of G by setting $H_1 = H$, $G_1 = G$, $H_2 = \overline{H}$, and $G_2 = \overline{G}$.

We utilize this functionality of the GSS to check whether a partially defined graph can be extended to an inducted k-universal graph.

For a partially defined graph G we define two fully defined graphs $G_{\neg 0}$ and $G_{\neg 1}$ by setting $V(G_{\neg 0}) = V(G_{\neg 0}) = V(G)$,

$$\begin{aligned} E(G_{\neg 0}) &= \{\, uv \mid u \neq v \in V(G), uv \in E_d(G) \cup E_u(G) \,\}, \text{ and} \\ E(G_{\neg 1}) &= \{\, uv \mid u \neq v \in V(G), uv \notin E_d(G) \,\}. \end{aligned}$$

▶ **Proposition 3.** *Let H be a graph and G a partially defined graph G. H embeds into some graph in $\mathcal{X}(G)$ if and only if there is a mapping from $V(H)$ to $V(G)$ that is a monomorphism both from H to $G_{\neg 0}$ and from $\overline{H}$ to $G_{\neg 1}$,*

Thus, we can use the GSS with a minimal change to its interface to check whether H embeds into some graph in $\mathcal{X}(G)$. We integrate the subgraph solver into the external propagator of the SMS framework. Whenever the SAT solver decides on an edge variable $e_{u,v}$, the partially defined graph G represented by the current truth assignment on the edge variables is sent to the propagator. The propagator checks whether all graphs from a set of randomly selected pattern graphs of a certain size are embeddable into G. If the check fails, this branch of the search can be terminated: a clause that excludes G is learned and sent back to the SAT solver. We set up three parameters for this process to control the usage of the subgraph solver: *Sample size* specifies the number of randomly selected pattern graphs. *Threshold* specifies the minimal number of determined edges/non-edges that is required to present in G for the embedability check to be performed. Finally, *frequency* specifies the likelihood the embedability check is performed and is indicated as an integer f. Everytime the SAT solver decides on an edge variable $e_{u,v}$, if the threshold is met, then there is a $1/f$ chance the emdedability check will be performed.

4.6 Templates

Next we propose a different approach, based on the concept of a template, which extends the simple method of hardcoding edges or non-edges for speeding up the solving process discussed in Section 4.3.

As it turned out, templates can be defined as partially defined graphs, but with a different purpose than the partially defined graphs used in SMS. Fix a small collection of graphs from $\mathcal{G}(k)$. We want each template to provide a distinct way in terms of how this small collection of graphs can be embedded at simultaneously in the universal graph. Ideally, these templates should allow us to hardcode as many edges/non-edges into the universal graph as possible for the benefit of speed-up. At the same time, there should not be too many of them, as we set a separate SAT instance of each template and need to show that all of them are unsatisfiable to establish a lower bound. In the end, we also want them to be easy to generate.

Let $\mathcal{H} \subseteq \mathcal{G}(k)$. An *$(n, \mathcal{H})$-template* is a partially defined graph G such that there exists a family of faithful monomorphism $(\delta_H : V(H) \to V(G))_{H \in \mathcal{H}}$ with the additional condition that for all $x, y \in V(G)$, if there is no $H \in \mathcal{H}$ such that $x, y \in \delta_H(V(H))$, then $xy \in E_u(G)$. Let $\mathcal{H}^{(n)}$ denote the set of all $(n, \mathcal{H})$-templates modulo ismomorphism.

▶ **Proposition 4.** *Let $k < n$ integers and $\mathcal{H} \subseteq \mathcal{G}(k)$. Then $f(k) \leq n$ if and only if $\mathcal{X}(G)$ contains a k-universal graph for some $G \in \mathcal{H}^{(n)}$.*

According to this proposition, we can limit the search for an induced k-universal graph to the graphs in $\mathcal{X}(G)$ for some $G \in \mathcal{H}^{(n)}$. It remains to generate $\mathcal{H}^{(n)}$ for suitable sets $\mathcal{H}$ for which

$\mathcal{H}^{(n)}$ remains reasonably small. We accomplish that by representing templates (i.e., partially defined graphs) as directed graphs and utlizing SMS for directed graphs as introduced by Kirchweger et al. [16], as follows.

Let us define a mapping $\mathcal{D}$ from the set of partially defined undirected graphs to the set of (fully defined) directed graphs: for each partially defined graph G, we put $V(\mathcal{D}(G)) := V(G)$ and $A(\mathcal{D}(G)) := \{\, uv \in E_d(G) \mid u > v \,\} \cup E_u(G)$. We also define the reverse mapping $\mathcal{D}'$ from directed graphs to partially defined undirected graphs: for each directed graph D, we put $V(\mathcal{D}'(D)) := V(D)$, $E_d(\mathcal{D}'(D)) := \{\, uv \mid uv \in A(D), vu \notin A(D) \,\} \cup \{\, uv \mid vu \in A(D), uv \notin A(D) \,\}$ and $E_u(\mathcal{D}'(D)) := \{\, uv \mid uv, vu \in A(D) \,\}$. For any partially defined graph G we have that $\mathcal{D}'(\mathcal{D}(G))$ and G are isomorphic.

We encode into CNF the condition for a directed graph D to be $\mathcal{D}(G)$ for some template G of $\mathcal{H}^{(n)}$, use SMS for digraphs to enumerate all such D modulo isomorphism, and then apply $\mathcal{D}'$ to recover the corresponding G. We encode the condition for a directed graph D to be $\mathcal{D}(G)$ for some template G of $\mathcal{H}$ as follows.

$$D(\mathcal{H}, n) := \bigwedge_{H \in \mathcal{H}} \Big(M(H, n) \wedge \bigwedge_{\substack{uv \in E(H) \\ 1 \le x < y \le n}} (m_{u,x} \wedge m_{v,y}) \to (\neg a_{x,y} \wedge a_{y,x}) \wedge \bigwedge_{\substack{uv \notin E(H) \\ 1 \le x < y \le n}} (m_{u,x} \wedge m_{v,y}) \to (\neg a_{x,y} \wedge \neg a_{y,x}) \Big)$$
$$\wedge \bigwedge_{1 \le x < y \le n} \Big((a_{x,y} \wedge a_{y,x}) \vee \bigvee_{H \in \mathcal{H}} \big(\bigvee_{i \in V(H)} m_{i,x} \wedge \bigvee_{i \in V(H)} m_{i,y} \big) \Big).$$

The first three conjuncts in $D(\mathcal{H}, n)$ enforce that H is an induced subgraph of G, and the last conjunct ensures that for all pairs u, v of $V(G)$ such that u, v are not both in the range of δ_H for all $H \in \mathcal{H}$, where δ_H refers to the faithful monomorphism from H to G, we have $uv \in U(G)$.

By trial and error, we determined that an ideal set of pattern graphs to generate templates for the task of determining $f(7)$ is $\mathcal{H}_4 := \{\, K_7, \overline{K_7}, K_{3,4}, \overline{K_{3,4}} \,\}$. Running SMS on $D(\mathcal{H}_4, 16)$ and $D(\mathcal{H}_4, 17)$ gives us 350 and 2772 solutions, respectively. We recover the templates from these solutions by applying $\mathcal{D}'$. Then, we filter out isomorphic templates using Nauty with the method explained in the Nauty and Traces User's Guide [20], which converts detecting isomorphisms between graphs with edge-colorings to that between graphs with vertex-colorings. This leaves us with 40 and 359 templates, respectively. Hence we have established the following result.

▶ **Proposition 5.** *$|\mathcal{H}_4^{(16)}| = 40$ and $|\mathcal{H}_4^{(17)}| = 359$.*

5 Encodings for k Universal Tournaments

5.1 Basic Encoding

For a fixed integer n, we consider a potential k-universal tournament G with $V(G) = [n]$ whose arcs are represented by *arc variables* $a_{u,v}$, $u \neq v \in [n]$. The truth value of $a_{u,v}$ indicates whether there is an arc from u to v.

First, we specify a CNF formula $F'(H, n)$ which is satisfiable if and only if H embeds into G, i.e., if H is isomorphic to a subtournament of G. The following CNF formula encodes the existence of a faithful monomorphism ϕ from H to G, in a similar way $F(H, n)$ encodes

the universality condition for universal graphs.

$$F'(H,n) := M(H,n) \wedge \bigwedge_{x \neq y \in [n]} \big((\neg a_{x,y} \vee \neg a_{y,x}) \wedge (a_{x,y} \vee a_{y,x})\big) \wedge \bigwedge_{\substack{uv \in A(H) \\ 1 \leq x < y \leq n}} (\neg m_{u,x} \vee \neg m_{v,y} \vee a_{x,y}).$$

$M(H,n)$ ensure that $(m_{u,v})_{u \in V(H), v \in V(G)}$ encodes an injective mapping from $V(H)$ to $V(G)$. The second conjunct ensures that there is exactly one arc between any two distinct vertices. The third conjunct ensures that the mapping preserves arcs. The formula

$$U'(k,n) = \bigwedge_{H \in \mathcal{T}(k)} F'(H,n)$$

is satisfiable if and only if there exists a k-universal tournament on n vertices. From a satisfying assignment we can read off a k-universal tournament.

5.2 Approaches to Universal Tournaments

For the SMS approach, we use a recent extension of the framework to directed graphs [16].

For the SAT approach, there are two more features that we add to the basic encoding. The first feature is to hardcode an embedding of the transitive tournament onto $[k]$.

$$\mathrm{TT}(k) = \bigwedge_{1 \leq i < j \leq k} a_{i,j}$$

The second feature is to utilize the perfect symmetry-breaking clauses from Lohn et al.'s work [18] who defined a series of CNF formulas $I(t)$, $1 \leq t \leq 8$, which they call *isolators*. These isolators restrict arc variables to achieve a perfect symmetry-breaking for tournaments in the sense that the satisfying assignments to $I(t)$ are in 1-1 correspondence with the tournaments of order t modulo isomorphism. Since the size of $I(t)$ grows quickly in t, we only utilize isolators for $t \leq 6$.

Given $l \leq r$, we write $I(l,r)$ to denote the isolator resulting from considering $[l,r]$ instead of $[r-l+1]$ to be the underlying vertex set. To obtain the overall encoding $\mathrm{TTI}(k,n)$, we start with $\mathrm{TT}(k)$ and see how many vertices there are in $[k+1,n]$. If $n-k \leq 6$, then we add $I(k+1,n)$ to the encoding and finish. If $n-k > 6$, then we add $I(k+1,k+6)$ to the encoding and repeat this process with $[k+7,n]$. For example, $\mathrm{TTI}(6,10) = \mathrm{TT}(6) \wedge I(7,10)$ and $\mathrm{TTI}(7,16) = \mathrm{TT}(7) \wedge I(8,13) \wedge I(14,16)$.

6 Certificates

There is a long tradition of computer-assisted proofs in mathematics; a famous example is the proof of the Four-Color Theorem by Apple and Haken [3]. There are mathematical statements whose proof seems to require extensive brute-force search, which is impossible for a human to check [13]. For trusting mathematical statements established by such computational methods, certifying the computational result with a formal proof that can be independently checked is highly desirable. Proof checking should ideally be accomplished through a simple algorithm whose correctness can be trusted or even fully verified. Most of the SAT-based methods described in our paper can be independently verified. In particular, the runs of a CDCL SAT solver on our encodings can produce a DRAT proof, for which a verified proof checker DRAT-trim is available [12, 27]. This approach extends to SMS as follows [15]. During the SMS run, we collect all the generated symmetry-breaking clauses and add them to the clauses generated from the encoding. We can run a plain CDCL SAT solver on this extended set of clauses that generates a DRAT proof. The validity of the symmetry-breaking clauses can also be checked offline with a simple tool.

One could extend the our current verification tool chain in various ways. One can formally verify SAT encodings themselves [5, 6, 9] to provide additional trust in the obtained results. To certify the correctness of auxiliary computations provided by the Glasgow subgraph solver, one can utilize a Cutting Planes proof logging format [10]. What remains is the usage of Nauty [20] for generating pattern graphs and occasionally for filtering isomorphic copies, which does not support independent certification. The former usage, the generation of pattern graphs, can easily be replaced by SMS. For the latter, if an isomorphism between two objects has been found, it is easy to verify whether this is indeed a correct isomorphism. On the other hand, if Nauty fails to identify two objects as being isomorphic, this would only make our overall approach less efficient but would not alter the final results.

7 Experiments

7.1 Configurations

Based on the techniques discussed in Sections 4 and 5, we consider several configurations that we test in our experiments. We use a naming convention for configurations, where a name starting with UG indicates a configuration for universal graphs and a name starting with UT indicates a configuration for universal tournaments. All UG configurations contain the following basic encoding: the encoding for universality (Section 4.1), the symmetry breaking for twins (Section 4.2), and the hardcoded $K_k, \overline{K_k}$ (Section 4.3). We distinguish between the use of SMS and the use of SAT without SMS (plain SAT).

UG-SAT	basic encoding with plain SAT.
UG-SMS	basic encoding with SMS.
UG-SMS-GSS	basic encoding with SMS extended with the Glasgow Subgraph Solver.
UG-SAT-Temp	basic encoding together with templates from Section 4.6 with plain SAT.
UT-SMS	encoding for universality from Section 5.1 with SMS.
UT-SAT-TTI	encoding for universality with fixed transitive tournament and isolators from Section 5.2 and plain SAT.

Theoretically, it is also possible to run SMS extended with GSS without the basic encoding. However, the preliminary experiments we conduct show that this is impractical given the drastically increased running time. Hence, we do not use this method in our main experiments.

7.2 Setup

The setup for the experiments is as follows. We generate $\mathcal{G}(k)$ and $\mathcal{T}(k)$ with Nauty 2.8.6 [20]. For configurations based on SMS, we use a recent SMS version that invoke a recent version of the Cadical SAT solver through the new IPASIR-UP interface [7]. For configurations that do not use SMS we employ Cadical as provided by the PySAT package [14]. We made minor changes to GSS [19] to access its internal functionality. All non-standard tools can be found in the supplementary material.

The experiments are run on a Sun Grid Engine (SGE) with Ubuntu 18.04.6 LTS. The architecture of the nodes in the SGE are among the following: 2× Intel Xeon E5540 with 2.53 GHz Quad Core, 2× Intel Xeon E5649 with 2.53 GHz 6-core, 2× Intel Xeon E5-2630 v2 with 2.60GHz 6-core, 2× Intel Xeon E5-2640 v4 with 2.40GHz 10-core and 2× AMD EPYC 7402 with 2.80GHz 24-core.

7.3 Universal Graphs

7.3.1 Verifying $f(k)$ and $F(k)$ for $1 \leq k \leq 5$

We run the configurations UG-SMS and UG-SAT-Temp to verify values $f(k)$ and $F(k)$ for $1 \leq k \leq 5$ as obtained by Trimble [26]. To verify the values of $f(k)$, we run both configurations on $n = f(k) - 1$ and $n = f(k)$ to see whether the former gives a negative answer (unsat) and the latter a positive (sat). To verify the values of $F(k)$, we solve incrementally, i.e., every time a universal graph is discovered, a clause that prohibits the generation of this particular graph is added to the solver and the solver continues. For UG-SMS, we do not hardcode K_k and $\overline{K_k}$ and start with $([n])$ as the initial partition. This way, the minimality check is enough to filter out non-isomorphic solutions. For UG-SAT-Temp, we detect and exclude isomorphic copies among the found universal graphs using Nauty. It turns out, that both configurations can verify each of the vaules within a couple of seconds.

7.3.2 Comparing the Efficiency of Computing $f(6)$

The case of $k = 6$ serves as a good benchmark for testing the efficiency of different methods, since it is neither too hard nor too easy. The value $f(6) = 14$ was determined by Trimble [26]. Hence, we test various configurations on $k = 6, n = 13$ and $k = 6, n = 14$. In the experiments described below, we run all instances 5 times and take the average running time.

For the case of $k = 6, n = 13$, we only test UG-SAT and UG-SMS since this case is computationally easy. The configurations use $0.61s$ and $1.39s$, respectively.

For the case of $k = 6, n = 14$, we test UG-SAT, UG-SMS, and UG-SMS-GSS. The first two configurations use $13.9s$ and $521s$, respectively. With UG-SMS-GSS, we try all possible combinations of the following parameter values: *threshold* $\in \{40, 50, 60, 70, 80\}$, *frequency* $\in \{5, 20, 50, 100, 500, 1000\}$, and *sample size* $\in \{5, 20, 60, 100, 156\}$. We record the total time used in solving and the time used by the embedability check.

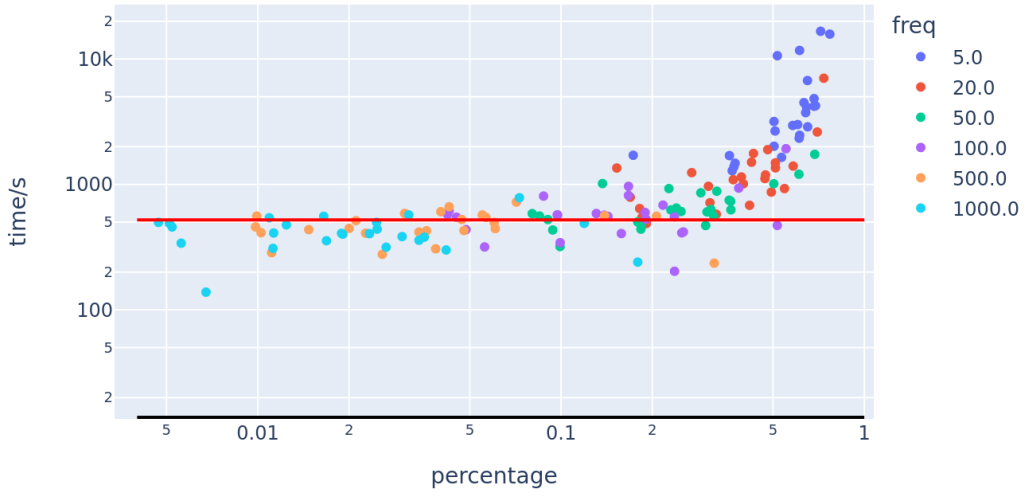

Figure 3 Data points for UG-SMS-GSS on $k = 6$, $n = 14$, grouped according to the value of *frequency*. We cast the data points in a two-dimensional space where the x-axis represents the percentage of time taken by the embedability check, and the y-axis represents the total time used. We draw a red horizontal line standing for the case of SMS, and a black horizontal line standing for the case of pure SAT.

Table 2 Running times for UG-SMS-GSS on the unsatisfiable case with $k = 7$ and $n = 16$.

case	sample size	frequency	threshold	total time/h	time used in embedability check/h
1	1044	500	100	73.63	0.12
2	1044	1000	80	71.34	3.40
3	1044	500	70	82.36	11.63
4	1044	1000	110	76.34	0

To understand the parameters' influence on the running time, and how the three configurations compare with each other, we present our findings in Figure 3. We first observe that UG-SAT uses significantly less time than UG-SMS or UG-SMS-GSS. Second, the introduction of the embedability check slightly decreases the running time in some cases and drastically increases the running time in others. Third, the solving time positively correlates with the percentage of the time used by the embedability check. In general, UG-SAT is clearly the most efficient among the configurations and the benefit of UG-SMS-GSS compared to UG-SMS is not obvious.

In Figure 3, data points of different *frequency* are shown in different colors. We see that data points of the same color form loose clusters. A further check into the distribution in terms of sample size and threshold within each group does not yield any meaningful pattern. We also tried grouping the same data points according to *sample size* and *threshold*, but did not obvious any obvious patterns. This indicates that the *frequency* parameter might be the only useful one among the three in terms of tuning for efficiency, and giving it a reasonably large value (which corresponds to the infrequent invocation of the embedability check) is preferred.

We also tried to determine the value of $F(6)$ incrementally with UG-SAT. After a few days we terminated the program. At that state, it had generated 36294 non-isomorphic 6-universal graphs of order 14.

7.3.3 $f(7) > 16$

To show $f(7) > 16$, we use UG-SAT, UG-SMS, UG-SMS-GSS, and UG-SAT-Temp. All four configurations conclude that a 7-universal graph with 16 vertices does not exist. UG-SAT takes 39 hours and UG-SMS takes 68 hours. For UG-SMS-GSS, we narrow down the choices of the possible values for the parameters based on previous experiments, and test all combinations with *sample size* $= 1044$, *frequency* $\in \{500, 1000\}$ and *threshold* $\{70, 80, 90, 100, 110\}$. Only 4 test cases out of 50 terminate within 97 hours. Table 2 provides some details. For UG-SAT-Temp, there are $|\mathcal{H}_4^{(16)}| = 40$ templates to consider. The easiest template takes $0.4s$, the hardest 38 minutes. The total time spent on all 40 templates is 3.6 hours, making the average time spent on each template $322.4s$.

Since UG-SAT-Temp seems to be the most effective among the considered configurations, we also tried using it to determine whether there exists a 7-universal graph of order 17. Here we have $|\mathcal{H}_4^{(17)}| = 359$ templates to consider. Unfortunately, this approach does not seem to be able to bring the problem down to the reach of modern SAT solvers. By the time of the writing, 25 out of the 359 cases still have not terminated. The 334 cases that have terminated spent 4237 CPU days, all returning with a negative answer. Figure 4 shows the time spent on each template in an ascending order. Even though we have no definitive result, the facts that (i) most cases could be shown unsatisfiable, and (ii) the SAT solver was not able to find a counter example on the remaining cases even after a long runtime, provide some evidence that a 7-universal graph of order 17 might not exist.

Table 3 Overview of running times in seconds for finding one or all solutions up to isomorphism.

				$t(k)-1$		$t(k)$			
				UT-SAT-TTI	UT-SMS	UT-SAT-TTI		UT-SMS	
k	$\mathcal{T}(k)$	$t(k)$	$T(k)$			one	all	one	all
5	12	8	1643	0.03	0.17	0.00	48.28	0.06	3.27
6	56	10	1088	6.41	30.68	11.83	N/a	105.36	47033.06

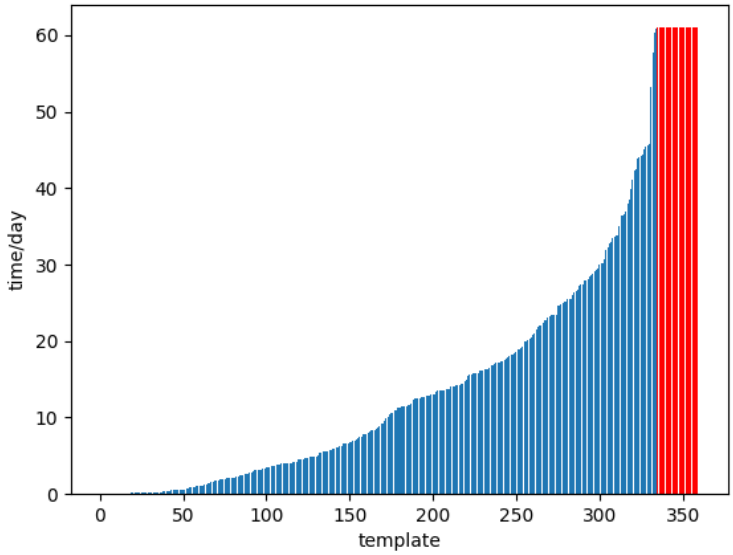

Figure 4 Time spent on each template ordered by their value. The blue bars represent the time spent by templates that have terminated, and the red bars show the time spent on templates that have not terminated by the time of writing.

7.4 Universal Tournaments

7.4.1 $t(k)$ and $T(k)$ for $1 \leq k \leq 6$

We run the configurations UT-SMS and UT-SAT-TTI to calculate $t(k)$ and $T(k)$ for $1 \leq k \leq 6$. To calculate the value of $f(k)$, we start with the lower bound

$$t(k) \geq \max(\{k, t(k-1), \min\{\, x \mid C_k^x \geq |\mathcal{T}(k)|\,\}\})$$

where C_k^x denotes the number of k-combinations from a given set of x elements, i.e., $C_k^x = \frac{x!}{k!(x-k)!}$. We test TTI$(k, n)$ while increasing n until we find a universal tournament. To calculate $T(k)$, we modify the two methods similarly as we did for verifying $F(k)$. Table 3 shows the time spent on the case $n = t(k) - 1$ and $n = t(k)$. The time consumed in all applicable cases for $0 \leq k \leq 4$ are under $0.005s$, so we omit them in the table. The process of finding out all non-isomorphic 6-universal tournaments of order 10 with UT-SAT-TTI does not terminate within a couple of days.

7.4.2 $13 \leq t(7) \leq 15$

An obvious lower bound for $t(7)$ is 11 since $C_7^{10} = 120 < 456 = |\mathcal{T}(7)|$. Neither UT-SMS nor UT-SAT-TTI gives a result on $k = 7, n = 11$ within half an hour. We generate separate SAT instances where we hardcode each of the four non-isomorphic tournaments of order 4 onto $[8, 11] \subseteq V(G)$ in addition to TTI$(7, 11)$. The four instances terminate with a negative answer in $14.55s$, $458.32s$, $1399.09s$, and $1212.89s$, respectively. This way, we have shown that $t(7) > 11$. To show that $t(7) > 12$, we generate and test the following SAT instance for each $\vec{a} = (a_i)_{i\in[7]}, a_i \in [4]$,

$$\text{TTI}(7, 12) \wedge \bigvee_{i\in[7]} \psi_{a_i}(i)$$

where

$$\psi_1(i) := |\{\, j \in [8, 12] \mid e_{i,j} = 1 \,\}| \leq 1,\ \psi_2(i) := |\{\, j \in [8, 12] \mid e_{i,j} = 1 \,\}| = 2$$
$$\psi_3(i) := |\{\, j \in [8, 12] \mid e_{i,j} = 1 \,\}| = 3, \text{ and } \psi_4(i) := |\{\, j \in [8, 12] \mid e_{i,j} = 1 \,\}| \geq 4.$$

This gives us 16384 instances in total and all of them return with a negative answer. Thus, $t(7) > 12$. The sum of the time spent is around 2002 CPU days, making the average running time for each case 2.9 hours. More than half of the cases terminate within 40 minutes, and the hardest case takes almost 4 days.

The upper bound $t(7) \leq 15$ is obtained with UT-SAT-TTI, which finds a 7-universal tournament of order 15 within $57.95s$. As for the cases of TTI$(7, 13)$ and TTI$(7, 14)$, UT-SAT-TTI does not terminates within a few days, and it is not practical to generate separate SAT instances in the way we did for TTI$(7, 12)$, given the large number of instances and the estimated time per case from our preliminary experimentation.

8 Conclusion

We have proposed several methods to compute the smallest order of induced k-universal graphs and tournaments as a synthesis problem that does not require the enumeration of all potential candidates. Our approaches make use of state-of-the-art CDCL SAT solvers. For the SAT encoding, we proposed a direct encoding and a lazy encoding that utilizes the Glasgow subgraph solver (GSS); for symmetry breaking, we proposed using the SAT modulo Symmetries (SMS) framework and a new templates approach.

We tested and compared these methods, which let us understand their strengths and weaknesses. The template approach was the fastest for unsatisfiable instances; SMS was particularly strong for enumerating all solutions; the GSS was able to speed up the SMS approach. We could improve the known lower bound and show that no induced 7-universal graph with 16 vertices exists, leaving possibilities 17 and 18 for the smallest solution.

We could determine the exact order of smallest k-universal tournaments up to $k = 6$ and the interval [13,15] for $k = 7$. We determined the precise number of optimal solutions, which is of independent interest for combinatorial research.

As future work, determining the exact order of a smallest induced 7-universal graph might be within reach by refining our methods and additional cubing. It would be interesting to see whether it is possible to combine SMS with the templates approach, as this might give a significant boost to possibly allow us to attack the k-universality problems for even larger values of k.

References

1. Noga Alon. Asymptotically optimal induced universal graphs. *Geom. Funct. Anal.*, 27(1):1–32, 2017. doi:10.1007/s00039-017-0396-9.
2. Stephen Alstrup, Haim Kaplan, Mikkel Thorup, and Uri Zwick. Adjacency labeling schemes and induced-universal graphs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 625–634. ACM, 2015. doi:10.1145/2746539.2746545.
3. Kenneth Appel and Wolfgang Haken. The solution of the four-color-map problem. *Sci. Amer.*, 237(4):108–121, 152, 1977. doi:10.1038/scientificamerican1077-108.
4. Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In Tomas Balyo, Nils Froleyks, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020. URL: https://tuhat.helsinki.fi/ws/files/142452772/sc2020_proceedings.pdf.
5. Cayden R. Codel. Verifying SAT encodings in lean. Master's thesis, Computer Science Department School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213, May 2022. URL: http://reports-archive.adm.cs.cmu.edu/anon/2022/CMU-CS-22-106.pdf.
6. Luís Cruz-Filipe, João Marques-Silva, and Peter Schneider-Kamp. Formally verifying the solution to the boolean Pythagorean triples problem. *Journal of Automated Reasoning*, 63(3):695–722, 2019. doi:10.1007/s10817-018-9490-4.
7. Katalin Fazekas, Aina Niemetz, Mathias Preiner, Markus Kirchweger, Stefan Szeider, and Armin Biere. IPASIR-UP: User propagators for CDCL. In Meena Mahajan and Friedrich Slivovsky, editors, *The 26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023), July 04-08, 2023, Alghero, Italy*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.
8. Johannes K. Fichte, Daniel Le Berre, Markus Hecher, and Stefan Szeider. The silent (r)evolution of SAT. *Communications of the ACM*, 66(6):64–72, June 2023. doi:10.1145/3560469.
9. Sofia Giljegård and Johan Wennerbeck. Puzzle solving with proof—writing a verified SAT encoding chain in HOL4. Master's thesis, Chalmers University of Technology and University of Gothenburg, 2021. URL: https://hdl.handle.net/20.500.12380/304104.
10. Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Subgraph isomorphism meets cutting planes: Solving with certified solutions. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1134–1140. ijcai.org, 2020. doi:10.24963/ijcai.2020/158.
11. Geňa Hahn and Claude Tardif. Graph homomorphisms: structure and symmetry. In Geňa Hahn and Gert Sabidussi, editors, *Graph Symmetry*, pages 107–166. Kluwer Academic Publishers, Dortrecht, 1997.
12. Marijn Heule, Warren A. Hunt Jr., Matt Kaufmann, and Nathan Wetzler. Efficient, verified checking of propositional proofs. In Mauricio Ayala-Rincón and César A. Muñoz, editors, *Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings*, volume 10499 of *Lecture Notes in Computer Science*, pages 269–284. Springer Verlag, 2017. doi:10.1007/978-3-319-66107-0_18.
13. Marijn J. H. Heule and Oliver Kullmann. The science of brute force. *Communications of the ACM*, 60(8):70–79, 2017. doi:10.1145/3107239.
14. Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018. doi:10.1007/978-3-319-94144-8_26.
15. Markus Kirchweger, Manfred Scheucher, and Stefan Szeider. A SAT attack on Rota's Basis Conjecture. In *Theory and Applications of Satisfiability Testing - SAT 2022 - 25th International Conference, Haifa, Israel, August 2-5, 2022, Proceedings*, 2022. doi:10.4230/LIPIcs.SAT.2022.4.

16 Markus Kirchweger, Manfred Scheucher, and Stefan Szeider. SAT-based generation of planar graphs. In Meena Mahajan and Friedrich Slivovsky, editors, *The 26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023), July 04-08, 2023, Alghero, Italy*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

17 Markus Kirchweger and Stefan Szeider. SAT modulo symmetries for graph generation. In Laurent D. Michel, editor, *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 39:1–39:17. Dagstuhl, 2021. doi:10.4230/LIPIcs.CP.2021.34.

18 Evan Lohn, Chris Lambert, and Marijn J. H. Heule. Compact symmetry breaking for tournaments. In Alberto Griggio and Neha Rungta, editors, *22nd Formal Methods in Computer-Aided Design, FMCAD 2022, Trento, Italy, October 17-21, 2022*, pages 179–188. IEEE, 2022. doi:10.34727/2022/isbn.978-3-85448-053-2_24.

19 Ciaran McCreesh, Patrick Prosser, and James Trimble. The Glasgow subgraph solver: Using constraint programming to tackle hard subgraph isomorphism problem variants. In Fabio Gadducci and Timo Kehrer, editors, *Graph Transformation - 13th International Conference, ICGT 2020, Held as Part of STAF 2020, Bergen, Norway, June 25-26, 2020, Proceedings*, volume 12150 of *Lecture Notes in Computer Science*, pages 316–324. Springer, 2020. doi:10.1007/978-3-030-51372-6_19.

20 Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *J. Symbolic Comput.*, 60:94–112, 2014. doi:10.1016/j.jsc.2013.09.003.

21 Jhon W. Moon. On minimal n-universal graphs. *Proc. Glasgow Math. Assoc.*, 7:32–33 (1965), 1965. doi:10.1017/S2040618500035139.

22 John W. Moon. *Topics on tournaments*. Holt, Rinehart and Winston, New York-Montreal, Que.-London, 1968.

23 OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences. Published electronically at http://oeis.org.

24 James Preen. What is the smallest graph that contains all non-isomorphic 4-node and 5-node connected graphs as induced subgraphs? online, October 2011. Mathematics Stack Exchange, https://math.stackexchange.com/q/75513, last edited 2020-04-25.

25 Richard Rado. Universal graphs and universal functions. *Acta Arithmetica*, 9(4):331–340, 1964. URL: http://eudml.org/doc/207488.

26 James Trimble. Induced universal graphs for families of small graphs. *CoRR*, abs/2109.00075, 2021. doi:10.48550/arXiv.2109.00075.

27 Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing – SAT 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer Verlag, 2014. doi:10.1007/978-3-319-09284-3_31.

A Appendix

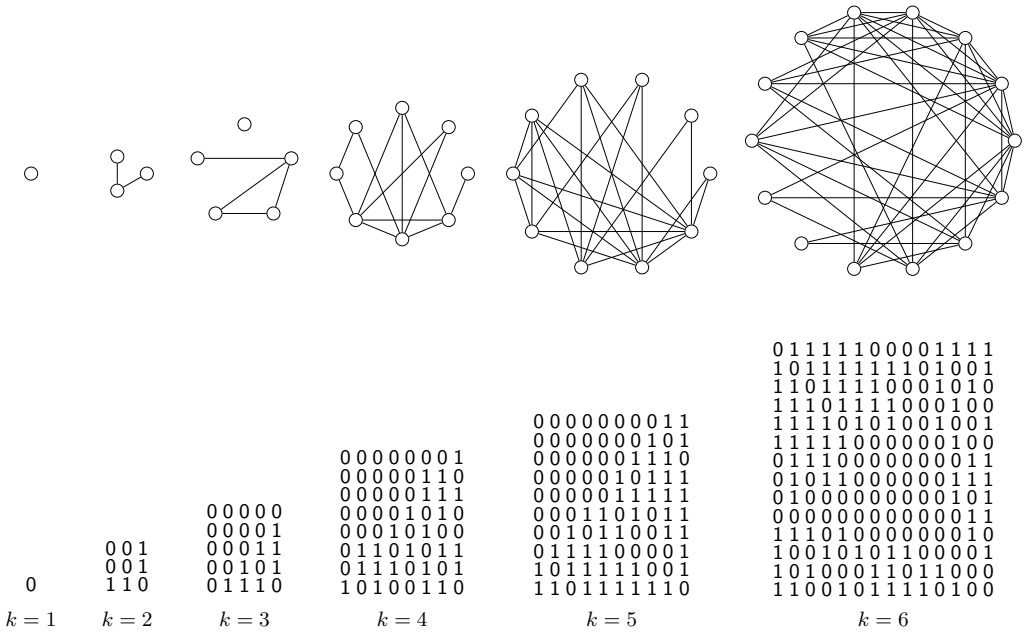

Figure 5 Smallest induced k-universal graphs and their incidence matrices.

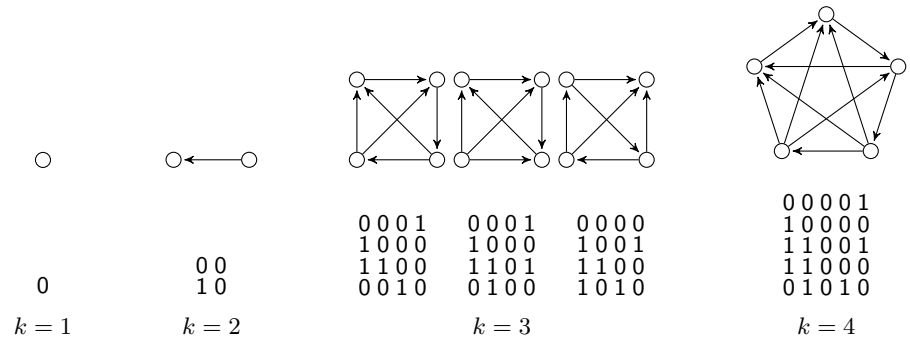

Figure 6 All smallest k-universal tournaments for $k = 1, 2, 3, 4$, and their incidence matrices.

Figure 7 The three displayed tournaments are examples of k-universal tournaments of order 8, 10, and 15 for $k = 5$, $k = 6$, and $k = 7$, respectively. We found many more with this property. We also show their incidence matrices. For $k = 5, 6$ we know that there are no k-universal tournaments of smaller order, for $k = 7$ we cannot rule out that there is one of order 14.