Compact Symmetry Breaking for Tournaments

Evan Lohn*, Chris Lambert*, Marijn Heule*
*Carnegie Mellon University. *elohn@andrew.cmu.edu, chrislambert@cmu.edu, marijn@cmu.edu*

***Abstract*—Isolators are a useful tool for reducing the computation needed to solve graph existence problems via SAT. We extend techniques for creating isolators for undirected graphs to the tournament (complete, directed) case, noting several parallels in properties of isolators for the two classes. We further present an algorithm for constructing n-vertex tournament isolators with $\Theta(n \log n)$ unit clauses. Finally, we show the utility of our new isolators in computations of tournament Ramsey numbers.**

***Index Terms*—Satisfiability, Symmetry-breaking, Directed-graphs, Tournaments, Isolators.**

I. Introduction

In recent years, SAT solvers have been used to solve several difficult combinatorial problems [1]–[3]. However, naive encodings of SAT problems often include undesired symmetries, i.e. certain matching subsets of variables that result in equivalent subproblems when given equivalent assignments. To prove the original formula unsatisfiable, in the worst case a solver must search through all possible symmetric parts of the problem space, which slows the generation of unsatisfiable proofs unnecessarily. Similarly, while the solver tries to find a satisfying assignment, symmetries in the input formula may cause the solver to effectively re-explore the same part of the search space even after proving the lack of a solution in a symmetric part of the problem.

The most common way of reducing the impact of symmetries in a given formula is by adding a set of new clauses called a Symmetry-Breaking Predicate (SBP) to the formula before solving. The goal of a SBP is to preserve the satisfiability of the formula while removing from consideration any regions of the search space known to be symmetric to other regions. In this work we focus on SBP's for *graph existence problems*, which are problems that can be solved by checking if a graph with particular structure exists. Solving such problems is an active area of research [4]–[6]. A large class of problem symmetries in graph existence problems naturally results from the existence of isomorphic labeled graphs. These symmetries exist independent of any desired graph property related to graph structure. Rather, they occur because SAT solvers must search the space of labeled graphs in order to prove the (non-)existence of an unlabeled graph. A SBP that targets graph isomorphisms is known as an isolator. Isolators that break many symmetries with few clauses are most useful in practice, as SAT solvers generally take longer to solve formulas with more clauses. Such isolators are often described as "short", "small", or "compact."

Prior work has shown that it is possible to generate small isolators for undirected graphs [7]. The present work instead handles the generation of short isolators for tournaments: complete, directed graphs. There are several mathematically interesting questions one can ask about tournaments that motivate the generation of tournament isolators. For example, Sumner's conjecture and various election models in social choice theory rely on tournament properties [8], [9]. However, the most well-known question about tournament structure is the Tournament Ramsey number problem, an analog to Ramsey numbers [10] that asks the question of "in what size tournament n must a transitive sub-tournament of size k exist." A (sub)tournament is *transitive* if it contains no cycles.

The first contribution of this work is the generation of compact tournament isolators that asymptotically match the search space reduction of a perfect isolator. Second, we present a methodology for the generation of compact isolators for small tournaments that extends prior work on undirected tournaments [7]. Finally, we demonstrate the practical usage of our small isolators for finding larger graphs relevant to the search for tournament Ramsey numbers.

II. Preliminaries

We define the following common concepts from SAT literature: A *literal* is either a variable or a negated variable. We use $\neg$ to denote negation. A *clause* is a disjunction of literals. A *unit clause* (sometimes referred to as simply a *unit*) is a clause containing exactly one literal. A *Conjunctive Normal Form (CNF) formula* is a conjunction of clauses. Unless otherwise specified, "formula" refers to "CNF formula." An *assignment* α is a function from variables to truth values (True/False). α satisfies a formula F if the boolean function denoted by F returns True given the inputs specified by α.

We also define several graph-theoretic concepts. A tournament $G = (V, E)$ is a complete directed graph; more formally, $\forall (v_1, v_2) \in V \times V, v_1 \neq v_2 \rightarrow ((v_1, v_2) \in E) \oplus ((v_2, v_1) \in E)$ and $\forall v \in V, (v, v) \notin E$, where $\oplus$ is the XOR operation. The phrase "G is an n-vertex tournament" means $|V| = n$. Given an n-vertex tournament $G = (V, E)$ and a permutation π on V, $\pi(G)$ is defined as $\pi(G) = (V, \{(\pi(v_1), \pi(v_2)) | (v_1, v_2) \in E\})$ and is colloquially referred to as applying π to G. Two n-vertex tournaments G_1, G_2 are *isomorphic* (written $G_1 \simeq G_2$) exactly when there exists a permutation π on the vertices of G_1 such that $\pi(G_1) = G_2$. When any such π exists, it is referred to as an *isomorphism* between G_1 and G_2. The *isomorphism class* (also, *equivalence class*) I_G of a tournament G is defined as $I_G = \{G' | G \simeq G'\}$. An *automorphism* π on a tournament G is any permutation π such that $\pi(G) = G$. The set of automorphisms of G form a group under function composition. This group is referred to as $Aut(G)$.

III. Isolator Notation and Concepts

To search for a tournament G satisfying some structural property, we define variables with the semantics "edge e exists in graph G" for use in a formula F. We say that F *admits* a graph G' exactly when there exists a satisfying assignment to the conjunction of F and the set of unit clauses semantically implied by the edges of G'. An *isolator* for n-vertex tournaments is a formula F that admits at least one tournament from each equivalence class on n-vertex tournaments. A *perfect* isolator is an isolator that admits *exactly* one tournament from each equivalence class. A perfect isolator F is *optimal* if there does not exist a perfect isolator with fewer non-unit clauses than F. A *compact* or *short* isolator is not rigorously defined. Rather, it describes an isolator with few enough non-unit clauses to be of practical use in solving SAT problems.

In this work, vertices will be denoted with lowercase letters $a, b, c, \ldots$ or with $v_1, v_2, \ldots, v_n$ when an ordering of the vertices is relevant. Arcs (directed edges) will be referred to with (u, v), meaning "there is an arc from u to v." In our construction of isolators, each variable is written in the form uv and has the semantics "arc (u, v) exists in the graph." Note that the literal $\neg uv$ therefore means "arc (v, u) exists in the graph."

A. Short Isolator Examples

Consider the following two labeled three-vertex tournaments.

These tournaments represent the only two equivalence classes for $n = 3$ tournaments: a cycle and a transitive tournament. While any combination of a cycle and transitive tournament would suffice to represent both equivalence classes, the tournaments chosen above have the interesting property of sharing two edges ab and bc (colored red). This property allows us to produce a short formula that admits both graphs:

$$ab \wedge bc.$$

This formula admits exactly one of the two labeled cycles and one of the six labeled transitive tournaments on 3 vertices, and does so with the fewest number of clauses. Therefore, $ab \wedge bc$ is a perfect, optimal isolator for $n = 3$ tournaments.

Figure 1 displays all 4 canonical representatives for $n = 4$ tournaments. We note that once again all highlighted edges have the same edge labels across graphs, and all permutations of non-highlighted edges are present. So, a short formula that admits exactly the set of graphs in the figure is

$$ab \wedge bc \wedge cd \wedge ad.$$

While the optimal isolators for $n = 3, 4$ are comprised entirely of unit clauses, this pattern does not hold for $n = 5$.

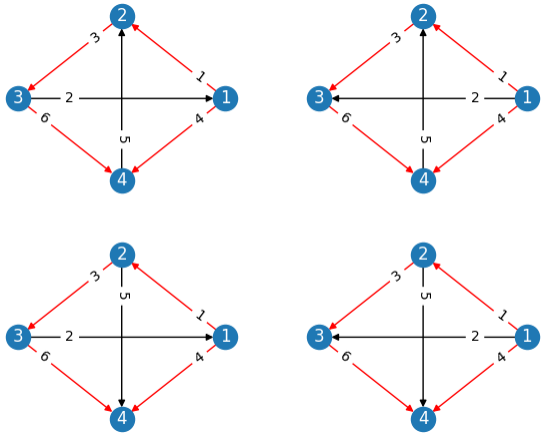


Fig. 1. Isomorphism class representatives admitted by a perfect, optimal isolator for 4-vertex tournaments. Red edges are edges fixed by unit clauses of the isolator, and the isolator has only unit clauses. Edge labels are the integers returned by the idx function from the Arc Literal Numbering section.

Table II contains the number of unit and non-unit clauses for $n \leq 7$.

B. Comparison of undirected graph and tournament isolators

Although the majority of this work focuses on tournament isolators, there are many interesting parallels between undirected and tournament isolators. In an undirected context, the *existence* of edge (u, v) is denoted by the literal uv, while its nonexistence is denoted by the literal's negation $\neg uv$. Because edgeless and complete graphs are isomorphism classes for any n in the undirected case, every clause of an undirected graph isolator containing only arc literals must contain at least one positive and one negative literal. These two graphs do not exist in the case of tournaments; the closest parallel is transitive tournaments. Unlike the set of n-vertex undirected graphs which contains exactly one empty graph and one complete graph with $n!$ automorphisms each, there are $n!$ isomorphic transitive tournaments on n vertices. It is possible to select the particular transitive tournament TT that an isolator admits by ensuring that at least one edge from TT is present in each clause of the isolator. A simple way to do so is ensure each clause contains at least one edge uv s.t. $u < v$ in vertex numbering.

One consequence of undirected isolators requiring at least one positive and one negative literal per clause is that undirected isolators have no unit clauses. However, while negating all literals in an undirected isolator produces another undirected isolator (because the existence and non-existence of an edge is symmetric), there is no direct parallel to be found in tournaments as edge directionality does not have this property.

Another interesting difference between undirected graphs and tournaments is the low number of isomorphism classes for tournaments when n is small (see table II). Intuitively, this happens because it is "easier" for tournaments to be isomorphic. The two options for the edge between vertices u and v in the undirected case are uv existing or not existing.

Crucially, an undirected graph G will never be isomorphic to G' constructed by adding or removing an edge of G, which is an operation that can be seen as "flipping" an edge to its other possibility. However, "flipping" an edge of a tournament T by changing the edge's direction will produce $T' \simeq T$ iff the two vertices u and v of the flipped edge had the same edges to the rest of the graph (the isomorphism is via the permutation that swaps u and v). Although this discrepancy exists for small n, the numbers of isomorphism classes for undirected graphs and tournaments are remarkably close for larger n (see OEIS A000088, A00056 [11]). Therefore, we expect that perfect, optimal isolators for undirected graphs and tournaments will have similar numbers of clauses for larger n.

C. Arc Literal Numbering

Each uv must be assigned a corresponding integer to conform to the commonly used DIMACS CNF format. To do so, we specify a function $idx_n(u, v)$ to map each possible arc (u, v) in an n-vertex tournament to a unique integer identifier. Because exactly one of (u, v) and (v, u) must exist for any two vertices u, v, idx must satisfy $idx_n(u, v) = -idx_n(v, u)$. To facilitate isolator comparisons across different n, idx also should satisfy the property that $idx_n(u, v) = idx_{n+1}(u, v)$. We therefore drop the subscript n when referring to $idx_n(u, v)$ in the future, as its value does not depend on n.

In particular, idx is inductively defined as follows for an $n + 1$-vertex tournament with vertices $v_1, v_2, , ...v_n, v_{n+1}$. Let $K = n(n-1)/2$ be the largest output of idx for an n-vertex tournament (implying base case $idx(v_1, v_2) = 1$ when $n = 2$). Applying idx to each of the arcs $(v_1, v_{n+1}), (v_2, v_{n+1}), ...(v_n, v_{n+1})$ yields $K + 1, K + 2, ..., K + n$, respectively. All arcs not included in this definition are of the form (v_w, v_u) where $w > u$, and are defined by the earlier mentioned constraint of $idx(u, v) = -idx(v, u)$.

IV. Unit Clauses

In practice, SAT solvers immediately reduce formulas with unit clauses to shorter formulas without units via unit propagation. Additionally, each unit clause reduces the size of the search space by a factor of 2. Therefore, it is practically useful to create isolators with as many units as possible. The following sections detail and analyze our various methods for creating isolators with many unit clauses.

A. Provable Units

While constructing smaller isolators using the techniques above, we opted to manually inspect our results and see what patterns they shared. In doing so, we rediscovered a well-known fact from graph theory literature; every tournament contains a Hamiltonian path [12]. Proof sketch: inductively consider a length n Hamiltonian path $v_1, v_2, ...v_n$ in an $n+1$-vertex tournament $G = (V, E)$. For the vertex v_{n+1} not part of the path, in the case that either (v_{n+1}, v_1) or (v_n, v_{n+1}) is in E, a length $n + 1$ Hamiltonian path is formed. Otherwise, (v_1, v_{n+1}) and (v_{n+1}, v_n) are in E and thus there must exist consecutive vertices v_i, v_{i+1} in the Hamiltonian path such that arcs (v_i, v_{n+1}) and (v_{n+1}, v_{i+1}) are in E. In this case, the sequence $v_1, v_2, ...v_i, v_{n+1}, v_{i+1}, ...v_n$ forms a length $n + 1$ Hamiltonian path. As a result of this property, a set of unit clauses describing a Hamiltonian path on an n-vertex tournament is always a valid n-vertex isolator.

Given the utility of unit clauses in isolators, it is natural to ask how many units there can possibly be in an n-vertex isolator. As it turns out, there is a long-known result from graph theory that implies that asymptotically there are at most $O(n \log n)$ units possible. By the orbit-stabilizer theorem, the size of the equivalence class of a graph G on n vertices is $\frac{n!}{|Aut(G)|}$, where $Aut(G)$ is the set of distinct automorphisms of G. In 1963 Erdős and Rényi proved that as n approaches infinity, the proportion of tournaments of size n with with nontrivial automorphisms approaches 0 [13]. Therefore, a proportion of graphs approaching 1 have equivalence classes of size $n!$, so the asymptotic number of equivalence classes is

$$\frac{2^{\binom{n}{2}}}{n!} \in \Theta(\frac{2^{\binom{n}{2}}}{2^{n \log n}}) = \Theta(2^{\binom{n}{2} - n \log n}).$$

An isolator with k unit clauses for n-vertex graphs admits at most $2^{\binom{n}{2}-k}$ equivalence class representatives, so in order to admit at least one member of each equivalence class (by the definition of an isolator), the number of units in an isolator must also be asymptotically upper-bounded by $n \log n$.

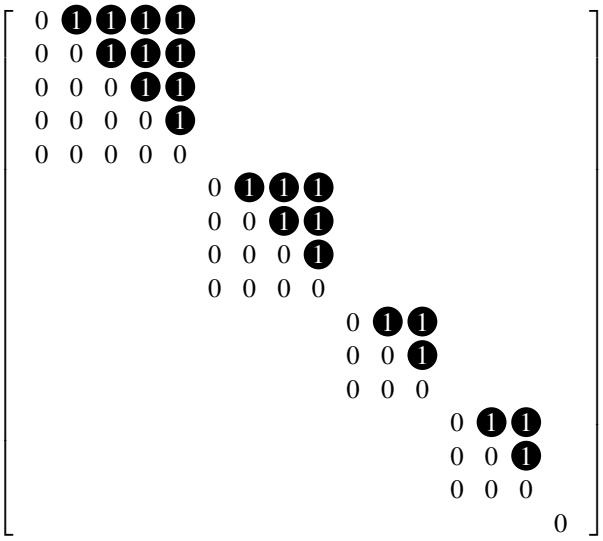
In the next section, we provide a procedure that achieves this bound.

B. TT-fixing

In situations where we know that every member of the class of n-vertex tournaments contains a TT_k (a transitive tournament of size k), we also know that every equivalence class must contain a member with the tournament fixed in some arbitrary position and orientation (i.e. vertices 1 through k in ascending order). Therefore, any formula that *fixes* (i.e. asserts the existence of) a TT_k on the class of n-vertex tournaments is a valid isolator. Because the remaining subset of $n - k$ non-fixed vertices also forms a tournament, further knowledge about the existence of a transitive tournament within the remaining $n-k$ vertices can be used to fix (via units) another transitive tournament within the $n - k$ vertex subtournament. This procedure can be repeated until all vertices of the original tournament are part of some fixed transitive subtournament. Tournament Ramsey numbers provide exactly the required information about the existence of a transitive subtournament. In fact, tournament Ramsey numbers (when known) provide the *largest* transitive subtournament guaranteed to exist in a tournament of a given size. Therefore, tournament Ramsey numbers (as well as upper bounds, which exist for arbitrarily large n) can be used to iteratively construct large sets of unit clauses for tournament isolators: we will refer to this process as TT-fixing.

C. TT-fixing gives $\Theta(n \log n)$ units

Let $units(n)$ be the function that returns the number of units that can be added to an isolator when using the TT-fixing method on n-vertex tournaments. Our goal is to prove



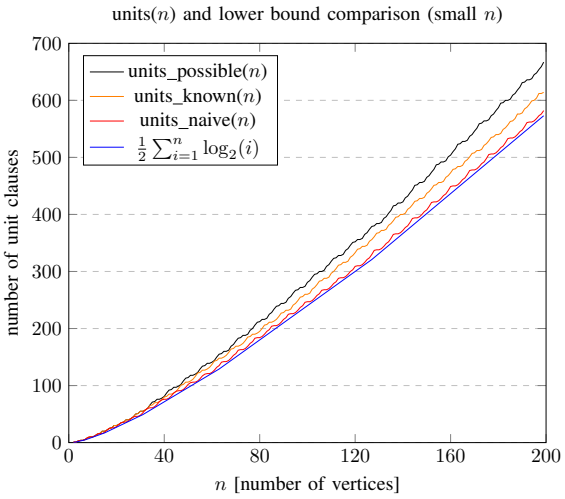


Fig. 2. A visual depiction of how the number of unit clauses produced by TT-fixing grows under different assumptions about Ramsey numbers.

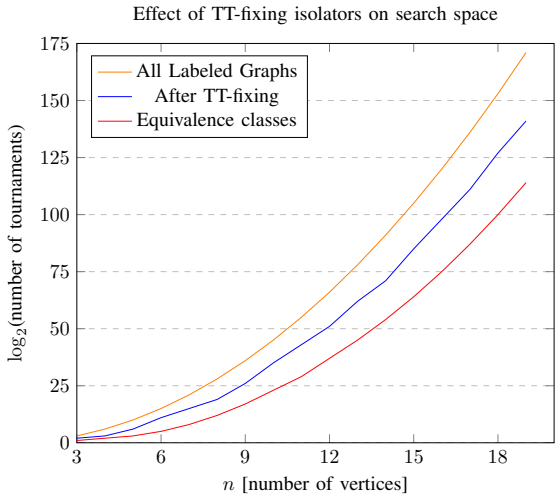


Fig. 3. A depiction of the search space reduction provided by TT-fixing using best-known Ramsey number bounds up to $n = 18$ in $\log_2$ space.

a lower bound on $units(n)$. Unfortunately, exact tournament Ramsey numbers are non-trivial to calculate (only up to $R(6) = 28$ is known). However, from Erdős and Moser we have that $R(k) \leq 2^{k-1}$ [14], i.e. that a TT_k must exist when considering any tournament on 2^{k-1} or more vertices. Erdős and Moser's bound can thus be used with TT-fixing to lower-bound $units(n)$.

We claim that $units(n) \geq \sum_{i=1}^{n} \frac{1}{2}\lfloor \log_2(i) \rfloor$. We proceed via induction, with step n depending on step $n-k$, with $k = \lfloor \log_2(n) \rfloor + 1$. The proposition is true for $n = 1$ because $0 \geq 0$, $n = 2$ because $1 \geq 0.5$. By definition of TT-fixing, for a graph with n vertices we have

$$units(n) = \frac{k(k-1)}{2} + units(n-k). \tag{1}$$

By the inductive hypothesis,

$$units(n-k) \geq \sum_{i=1}^{n-k} \lfloor \log_2(i) \rfloor / 2. \tag{2}$$

Next, we have that

$$\begin{aligned} k\frac{k-1}{2} &= k\lfloor \log_2(n) \rfloor / 2 \\ &= \sum_{i=n-k+1}^{n} \lfloor \log_2(n) \rfloor / 2 \\ &\geq \sum_{i=n-k+1}^{n} \lfloor \log_2(i) \rfloor / 2. \end{aligned} \tag{3}$$

Combining lower bounds (2) and (3) for the terms of eq. (1) completes the proof:

$$\begin{aligned} units(n) &= \frac{k(k-1)}{2} + units(n-k) \\ &\geq \sum_{i=n-k+1}^{n} \lfloor \log_2(i) \rfloor / 2 + \sum_{i=1}^{n-k} \lfloor \log_2(i) \rfloor / 2 \qquad (4) \\ &= \sum_{i=1}^{n} \lfloor \log_2(i) \rfloor / 2. \qquad (5) \end{aligned}$$

This inequality result directly implies the asymptotic $n \log n$ bound, because $\log_2(n!) \in \Theta(n \log n)$.

D. Practical vs Theoretical TT-fixing units

We first note a useful recurrence relation on tournament Ramsey numbers: $R(k) \leq 2R(k-1)$.

Proof. Consider an arbitrary vertex v in an arbitrary tournament G on $2R(k-1)$ vertices. v must have either an out-degree or an in-degree of at least $R(k-1)$. In either case, consider the subset of at least $R(k-1)$ vertices pointed to/at by v. This subset must contain some TT_{k-1} as a subgraph by definition of $R(k-1)$. However, v points to or at all vertices in this TT_{k-1}, which demonstrates that a TT_k comprised of the TT_{k-1} vertices and v exists in G. □

In Figure 2 the bottom two lines depict the strict lower bound used in the $n \log n$ units proof (blue), as well as the actual number of units TT-fixing would provide if we only used the $R(k) \leq 2^{k-1}$ bound from the proof (red). Above that (orange) is the number of units TT-fixing provides given the best currently known Ramsey number bounds. The best known bound on $R(7)$ is $34 \leq R(7) \leq 47$ [15], so the black line describes the best case for how many unit clauses TT-fixing could provide if $R(7) = 34$ was proven. The recurrence relation $R(k) \leq 2R(k-1)$ is what allows even improvements to small Ramsey number bounds to impact the efficacy of TT-fixing for large n.

In Figure 3, the top (orange) line is the total number of graphs a SAT solver must search in a tournament existence problem in the absence of an isolator. The bottom (red) line is the number of unlabeled tournaments on n vertices; this is the minimum number of graphs that any brute-force solver must search to solve a tournament existence problem. This data was taken from OEIS sequence A000568 [11], which limits the size of n for which we can make this comparison to $n = 19$. The middle (blue) line shows how many graphs are admitted by an isolator created by TT-fixing with the best known bounds on tournament Ramsey numbers. As n grows large, the gap between the bottom two lines should grow small relative to the gap between the top two lines as per the $n \log n$ units upper bound proof.

E. Undirected Isolators: Clique-fixing

As mentioned earlier, undirected isolators cannot have unit clauses. Therefore, undirected isolators cannot directly benefit from units via TT-fixing. However, a crossover result for undirected graphs does exist for binary clauses that uses the same ideas as TT-fixing; we term this process *clique-fixing*. Undirected Ramsey number guarantee the existence of a red or blue colored k-clique for graphs with more than $R_u(k)$ vertices (R_u used here for undirected Ramsey numbers). Clique-fixing uses the same iterative process as TT-fixing, but replaces the generation of units for TT_k with the following clauses:

$\{r \vee e, \neg r \vee \neg e | e \in Edges(K_k)\}$

where r is an auxiliary variable representing the concept "the k-clique is red" and $Edges(K_k)$ is the set of edge literals for the complete graph on k vertices. We note that these clauses are "almost" units in the sense that after a solver makes a decision about whether to set r to true or false, an entire $k(k-1)/2$ edges are set by unit propagation. Each step of clique-fixing therefore loses a single unit clause worth of search space reduction when compared to TT-fixing. Although not the focus of this work, it is plausible that a similar asymptotic optimality analysis could be done for clique-fixing given this small discrepancy. However, it is notable that undirected Ramsey numbers (necessary for clique-fixing) empirically grow much faster than tournament Ramsey numbers despite equal theoretical worst-case scaling, which implies that clique-fixing may not be as practically useful as TT-fixing.

V. Perfect, Optimal Isolator SAT encoding

Unit-based techniques scale to arbitrary n, and TT-fixing is "asymptotically perfect" in the sense that for large tournaments, no isolator generation technique can provide more than a non-constant factor of search space reduction over TT-fixing. However, no known perfect isolators for $n > 4$ consist solely of unit clauses. Additionally, it can be practically useful to have an *optimal* perfect isolator for small tournaments to allow searching via SAT solver for only non-isomorphic (sub-)graphs as efficiently as possible. The practical utility of compact perfect isolators is demonstrated in our own experiments in the later "Tournament Ramsey Graphs" section. In the following sections, we describe our technique for creating perfect, optimal isolators for $n \leq 6$.

A. Basic SAT encoding

We re-implemented and modified the perfect isolator encoding for undirected graphs [7] to be used for tournaments. Formally, we encoded the question "Is there a set of k clauses $C_1, C_2, ...C_k$ that is a perfect isolator for n-vertex tournaments." Decoding a solution to this formula allowed us to produce an n-vertex isolator with k clauses.

For the ith isolator clause C_i and arc literal l, we defined the variable $In(C_i, l)$ to represent "l is in C_i". Then, for each tournament G on n vertices, we define variables $Excludes(G, C_i)$ for $1 \leq i \leq k$ to mean "clause C_i does not admit G." This specification is implemented as follows with a Tseitin encoding [16] to handle the equality and conjunctions:

$$Excludes(G, C_i) \leftrightarrow \bigwedge_{l \in A_G} \neg In(C_i, l). \tag{6}$$

Here A_G is the set of arc literals corresponding to the arcs present in graph G. We also define the variable $Canon(G)$ for all graphs G, meaning "Graph G is the canonical representative of its isomorphism class I_G." We implement this as follows (again using Tseitin):

$$Canon(G) \leftrightarrow \bigwedge_{i=1}^{k} \neg Excludes(G, C_i). \tag{7}$$

Finally, for each isomorphism class I, we add the following clauses representing "exactly one graph in I is canonical" to the formula for each isomorphism class I:

$$ExactlyOne(\{Canon(G) | G \in I\}) \tag{8}$$

Here $ExactlyOne$ is implemented with an At Most One operation via Sinz encoding [17] and an At Least One via disjunction. Therefore, a satisfying assignment to this formula corresponds to a perfect isolator on k clauses. If the formula is unsatisfiable for k and satisfiable for $k+1$, then the perfect isolator with $k+1$ clauses is optimal for the n in question.

B. Symmetry Breaking

One symmetry in the above encoding is the order of the isolator clauses, as reordering clauses of an expression in CNF does not affect its satisfying assignments. To break this symmetry, we added clauses that ensured a lexicographic ordering of the clauses in the resulting isolator. For every adjacent pair of clauses C_i and C_{i+1}, we fixed some ordering of every literal that may appear in them $l_1, l_2, \ldots, l_n$, and then created variables $e_0, e_1, \ldots, e_n$ where e_j represents that clauses C_i and C_{i+1} are equivalent when considering only the first j literals. e_0 is always true, and to maintain the semantics of the other e_j we added the clauses

$$e_j \leftrightarrow (e_{j-1} \wedge (In(C_i, l_j) \leftrightarrow In(C_{i+1}, l_j)))$$

via the Tseitin transformation for every $1 \leq i < k$ and $1 \leq j \leq n$. Then, we enforced a lexicographic ordering by requiring that for every j such that C_i and C_{i+1} were equal up to j, that if clause C_i contained l_j then C_{i+1} must also contain l_j. Explicitly, we added the following requirement via the Tseitin transformation for every $1 \leq i < k$ and $1 \leq j \leq n$:

$$e_{j-1} \wedge In(C_i, l_j) \implies In(C_{i+1}, l_j)$$

and furthermore we required that e_n is false to ensure a strict ordering. When searching for an isolator with k clauses, this reduces the search space by a factor of $k!$ as only one of the $k!$ permutations of a given distinct set of clauses will be considered.

There is another symmetry in the vertex labeling. For a given isolator, for each literal l corresponding to arc index I_{ab}, we can change l to correspond to arc index $I_{\pi(a),\pi(b)}$ where π is a permutation of vertex labels. The resulting isolator accepts the same graphs that the original did, but under vertex permutation π. To break this symmetry, note that any tournament isolator must admit exactly one transitive tournament. So, we choose to admit only the canonical transitive graph with edges of the form $(v_i, v_j), i < j$. Note that because every edge in this graph goes from a lower numbered vertex to a higher numbered vertex, the corresponding literals in our encoding are all positive. As such, we know that for any isolator, there is a permuted isolator such that every clause has at least one positive literal in each clause. We may add this to our encoding by requiring for all clauses C

$$\bigvee_{l \in A_p} In(C, l)$$

with A_p being the set of all positive literals. When trying to find an isolator for n vertices, this reduces the search space by a factor of $n!$ since the solver is guaranteed to only consider isolators for which the canonical transitive graph is the one described above.

C. Encoding Unit Propagation

Under the encoding described above, our solver finds isolators with many large clauses. However, by applying unit propagation it was often possible to reduce clause sizes. This indicated that not only was the solver generating solutions that needed postprocessing, but candidate isolators that were equivalent under unit propagation were being considered multiple times — a sort of symmetry in this problem. To resolve this, we added variables $Unit(l)$ representing "literal l is a unit clause." We then required that the isolator be already unit-propagated with respect to these literals by adding the requirement

$$\neg In(C, l) \vee \neg Unit(l)$$

for all clauses C and literals l. We also had to account for these units excluding graphs in the $Canon$ clauses, which were updated to

$$Canon(G) \leftrightarrow \bigwedge_{i=1}^{k} \neg Excludes(G, C_i) \wedge \bigvee_{l \in A_G} \neg Unit(\neg l)$$

Finally, we considered whether to count these special unit literals towards the clause count in determining isolator optimality. As mentioned in the preliminaries, we chose not to do so. When an isolator with units is used in a SAT solver, the units will be instantly eliminated through unit propagation and thus will reduce the complexity of the resulting problem. Therefore, we consider an optimal isolator to not just have the minimal number of clauses, but the minimal number of non-unit clauses. Since units cannot exist in undirected graph isolators (because an undirected graph isolator must admit both the complete and empty graph), this definition of optimality is consistent with the prior work on the undirected case [7]. Note that we only needed to consider positive unit literals as per the vertex-labeling symmetry breaking, which drastically reduced the search space.

VI. Additional Isolator Generation Techniques

The following sections describe several miscellaneous techniques, ranging from practical ways to gain slight improvements on prior techniques to possible directions for future research.

A. Incremental Isolators

Prior work has already shown that any isolator for n-vertex tournaments is also an isolator for $n+k$-vertex tournaments for any positive k, and that combining an isolator on m vertices with an isolator on n vertices by applying each isolator to a disjoint subset of vertices creates a new isolator on $m+n$ vertices [18]. Therefore, it is possible to construct perfect isolators for $n+k$-vertex tournaments by adding clauses to any isolator for n-vertex tournaments. In particular, our SAT encoding pipeline had the option to ignore graphs that are not admitted by a given set of units. Including the maximal set of units from an n-vertex isolator when generating an encoding for $n+1$-vertex isolators reduces the number of tournaments to generate $Canon$ clauses for by a factor of at least 2^n

because each isolator has at least the units corresponding to a Hamiltonian path. It is worth noting that we do not have any proofs that any of our non-perfect or non-optimal isolators can be extended to an *optimal* isolator, even when the isolator being extended from is comprised of only unit clauses. However, extending an isolator from an initial set of units can make searching for compact isolators much more efficient.

The technique of combining disjoint isolators is practically useful for creating compact isolators for large n. Although TT-fixing guarantees asymptotic optimality, when the procedure reaches the point of generating units for $k = 8$ or fewer remaining vertices, more units can be added by using the units from a known compact isolator for k vertices. For example, TT-fixing will generate 9 units when processing 8-vertex (sub)tournaments, while our isolator for $n = 8$ contains 11 units.

B. Probing

In addition to the SAT encoding approach to isolator generation, we also generated isolators using a method from prior work called "random probes" [7]. On a high level, this approach starts with an empty set of clauses and adds randomly generated clauses that preserve at least one member of each equivalence class until the isolator is perfect. There were only two non-superficial changes needed to adapt the prior work on random probes for undirected graphs to the directed case; allowing unit clauses and allowing clauses with only positive literals. While not guaranteed to generate optimal isolators, the strength of this approach is the relative speed with which isolators are generated. This approach also benefited in efficiency from the technique of disallowing clauses with all negative literals and extending isolators from the unit clauses of smaller isolators.

C. Canonical set techniques

One major criticism of our approaches for producing *perfect* isolators is that they do not scale to $n > 9$. Already at n=9 there are 2^{36} labeled tournaments on 9 vertices; this number can be reduced by enforcing all 13 known possible unit clauses during the generation of the CNF file, but the CNF produced by our SAT-based method is not solvable in several days, and probes for $n = 8$ take ~ 2 days compared to several seconds for $n = 7$. As such, we also experimented with a canonical-set-based approach that scales with the number of equivalence classes.

The canonical-set-based approach uses the well-known idea that one way to create an isolator is to generate one canonical representative for each equivalence class, then construct a formula that admits exactly those graphs [19]. Our approach differed slightly from [19], as instead of using lexicographic ordering constraints from a carefully constructed "canonizing set" on the graph's adjacency matrix, we express any given set of canonical representatives in disjunctive normal form (DNF), then convert the DNF to CNF using Bica [20]. Because this approach may lead to much larger isolators than the optimal ones, we first used the canonical sets induced by our optimal isolators as proofs of concept. As seen in the "optimal" column of table I, the resulting CNFs were equal in number of clauses to our optimal isolators up to n=6 (our isolators for n=7 and above are not optimal, and thus were not considered). Next, we tried the naive approach of using the canonical set produced by the NAUTY [21] program. The size of the isolators generated was very suboptimal, as the canonical form used by NAUTY does not enforce common edges (column "arbitrary" in table I below).

A more promising approach involves modifying a given arbitrary canonical set to have certain properties. Given a canonical set, we detect a Hamiltonian path in each member of the set via an algorithm modeled on the proof of Hamiltonian path existence, then permute each member of the canonical set to have the path $(v_1, v_2), (v_2, v_3), ...(v_{n-1}, v_n)$. The motivation of this procedure is to cause the canonical set members to have more edges in common, which will lead to more unit clauses in the final CNF. These experiments gave better results than the naive approach, but did not succeed in matching the true optimal isolator length (see "heuristic" column of table I). A possible direction for future work is the development of heuristics for permuting canonical set representatives to produce canonical sets admitting short isolators.

TABLE I
ISOLATORS FOUND WITH CANONICAL SET TECHNIQUES

Vertices	Isomorphism classes	optimal	heuristic	arbitrary
4	4	4	6	8
5	12	8	16	24
6	56	14	42	62

VII. RESULTS

Our experimental results include the sizes of known perfect isolators for small n, as well as experiments showing the practical utility of small $n = 6, 7$ perfect isolators for solving a tournament existence problem.

A. Experimental Setup

Our SAT-based approach to generating isolators rely on the creation of "map" files: text files associating each tournament of size n with a label representing that graph's isomorphism class. In order to generate a map file for tournaments on n vertices, we began by enumerating all $2^{n(n-1)/2}$ graphs of size n. We converted each graph into an adjacency matrix and then into the ".d6" format specified in the NAUTY handbook, then fed the resulting graphs into the labelg script bundled with the NAUTY tool for graph isomorphisms [21]. labelg produced a file where each graph was converted to the canonical form used by nauty. We gave each canonical form a unique label and outputted the arc (directed edge) indices of each original graph alongside its canonical form.

B. Small Optimal Isolators

Our SAT encoding allowed us to compute optimal isolators up to $n = 6$. The SAT solver CaDiCaL [22] solves the

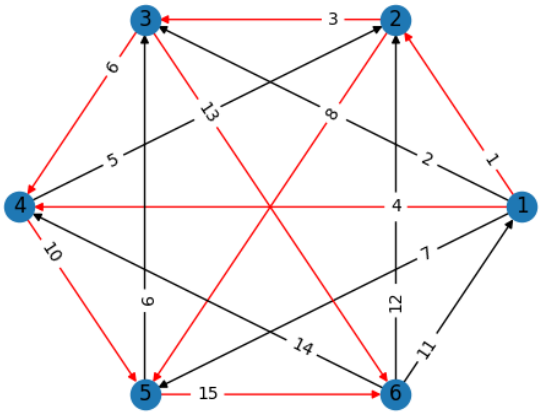


Fig. 4. One of the 56 Isomorphism class representatives admitted by a particular isolator for 6-vertex tournaments. Red edges are edges fixed by unit clauses of the isolator.

two instances required to prove optimality ($k = 6, 7$ non-unit clauses) within 24 hours. Figures 1 and 5 graphically display optimal, perfect isolators for $n = 4, 5$ by displaying a graph from each isomorphism class. Figure 4 presents the same image for one of the 56 isomorphism classes for $n = 6$. Most of the structure of these isolators can be seen from their unit clauses, which are depicted via red edges in the figures.

For $n = 7$, solving the SAT instance directly became clearly infeasible (taking several days without any signs of progress). However, random probing allowed us to find a perfect isolator. Each probe ran in around 10 seconds when restricted to force a positive literal in each clause with the map file reduced by the unit clauses from $n = 6$. Table II describes the best (fewest non-unit clauses) isolator found for $1 \leq n \leq 7$. Several thousand probes were required to find our best known isolator for $n = 7$.

TABLE II
SHORTEST ISOLATORS FOUND FOR $n \leq 7$

Vertices	Isomorphism classes	Best units	Best non-units
1	1	0	0
2	1	1	0
3	2	2	0
4	4	4	0
5	12	6	2
6	56	8	6
7	456	9	47

C. Tournament Ramsey Graphs

Let $TR(n, k)$ be the set of unlabeled (equivalently, labeled but up to isomorphism) tournaments on n vertices that contain no TT_k as a subgraph. The tournament Ramsey problem asks for the smallest n such that $|TR(n, k)| = 0$ given some fixed k. To improve a tournament Ramsey number lower bound, it suffices to find a tournament without a TT_k larger than any known tournament without a TT_k. One method for doing so is attempting to extend one of the largest known tournaments without a TT_k, because any larger tournament with a TT_k necessarily has at least one smaller such tournament as a subgraph (up to isomorphism).

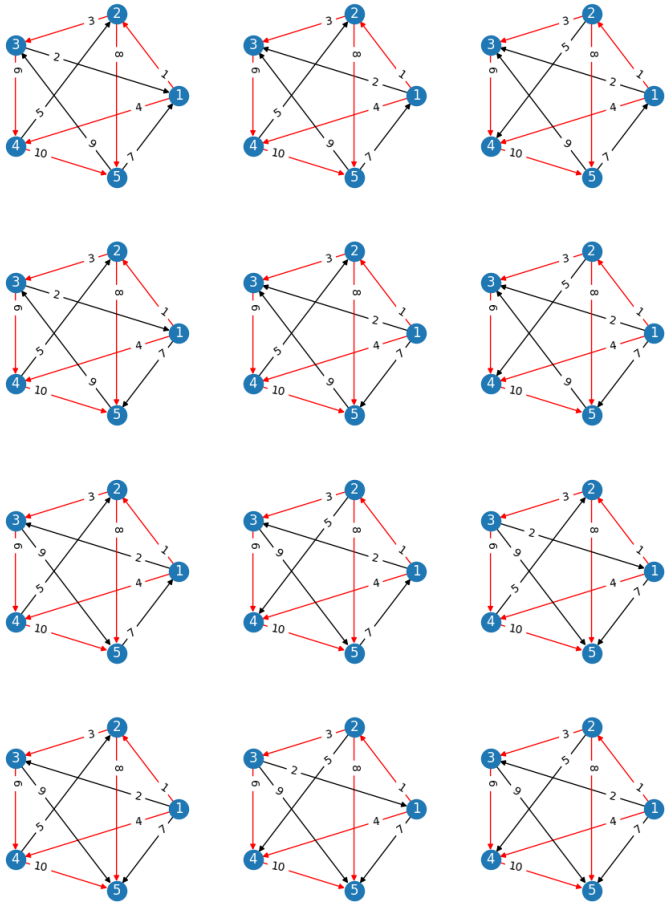


Fig. 5. Isomorphism class representatives admitted by a particular isolator for 5-vertex tournaments. Red edges are edges fixed by unit clauses of the isolator. The two non-unit clauses in the isolator are $2 \vee -5 \vee 9$ and $2 \vee 7 \vee -9$. **TODO: generate on ubuntu, it looks nicer**

Therefore, one direct method for incremental progress towards improving tournament Ramsey number bounds is to expand the known portion of $TR(n, k)$ for the maximal known n for some fixed k. In particular, we sought to use our isolator techniques in conjunction with the methods of earlier work on tournament Ramsey numbers [15] to find more members of $TR(33, 7)$. All isomorphism checking and automorphism generation referred to below was done with the NAUTY [21] tool.

For certain n, k it is the case that $|TR(n, k)| = 1$. In such cases, the graph STn refers to the single member of $TR(n, k)$. For example, ST25, ST26, and ST27 are the only tournaments on $25, 26, 27$ vertices respectively without a TT_6 [23]. Out of the 5303 members of $TR(33, 7)$ found prior to this work [15], all contained ST25 and ST26 as a subgraph, while all but 352 members contained ST27. While every tournament in the known portion of $TR(33, 7)$ contained exactly 0 or 1 subgraph isomorphic to ST27, each tournament had between 2 and 39 (inclusive) subgraphs isomorphic to ST26, with the

vast majority having 27 due the existence of ST27 in the graph (removing any vertex from ST27 yields ST26).

Our main experimental setup for finding new elements of $TR(33,7)$ involved finding new members containing ST26 but not ST27 by solving a CNF formula with a SAT solver. The formula can be described as the union of the following sets of clauses:

1) $26 * 25/2 = 325$ unit clauses requiring that ST26 be present in vertices v_1 through v_{26}
2) The perfect isolator for $n = 7$ on the seven remaining vertices v_{27} through v_{33}
3) A clause blocking each of the 5303 known solutions for each vertex permutation that caused the solution to have ST26 in vertices v_1 through v_{26} and a graph admitted by the $n = 7$ isolator in vertices v_{27} through v_{33}
4) clauses enforcing the "no TT_7" condition from [15]

While this formula does not disallow all ST27s (i.e. a solution might include an extension from ST26 that was not present in the original solution set), it disallows all currently known extensions, including the most common by far 1-vertex extension from ST26 to ST27. Additionally, the $n = 7$ perfect isolator plays a crucial role for finding new solutions in that without it, the SAT solver could find any tournament equivalent to one of the previously known $TR(33,7)$ elements except for some non-automorphic permutation of the last 7 vertices (which would thus be isomorphic to the previously known solution). We claim that all solutions to our formula must be non-isomorphic to the original 5303 solutions.

After several hours, CaDiCaL found 1 new member of $TR(33,7)$ with this method. After finding this solution, we updated the formula to include the blocking clauses for the new solution and its isomorphisms. CaDiCaL did not produce further solutions of the updated CNF for several days, so it is possible that the formula is simply unsatisfiable. In conclusion, the formula we produced allows for a relatively efficient (compared to brute force) search over the space of "new" members of $TR(33,7)$.

REFERENCES

[1] J. Brakensiek, M. Heule, J. Mackey, and D. Narváez, "The resolution of Keller's conjecture," in *Automated Reasoning*, N. Peltier and V. Sofronie-Stokkermans, Eds. Cham: Springer International Publishing, 2020, pp. 48–65.

[2] M. J. H. Heule, "Schur number five," in *AAAI*, 2018.

[3] M. J. H. Heule, O. Kullmann, and V. W. Marek, "Solving and verifying the boolean pythagorean triples problem via cube-and-conquer," in *Theory and Applications of Satisfiability Testing – SAT 2016*, N. Creignou and D. Le Berre, Eds. Cham: Springer International Publishing, 2016, pp. 228–245.

[4] T. Blankenship, J. Cummings, and V. Taranchuk, "A new lower bound for van der Waerden numbers," *European Journal of Combinatorics*, vol. 69, pp. 163–168, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S019566981730166X

[5] M. Codish, M. Frank, A. Itzhakov, and A. Miller, "Computing the Ramsey number R(4,3,3) using abstraction and symmetry breaking," *Constraints*, vol. 21, no. 3, p. 375–393, jul 2016. [Online]. Available: https://doi.org/10.1007/s10601-016-9240-3

[6] N. Komarov and J. Mackey, "On the number of 5-cycles in a tournament," *Journal of Graph Theory*, vol. 86, 2017.

[7] M. J. Heule, "Optimal symmetry breaking for graph problems," *Mathematics in Computer Science*, vol. 13, no. 4, pp. 533–548, 2019.

[8] D. Kühn, R. Mycroft, and D. Osthus, "A proof of Sumner's universal tournament conjecture for large tournaments," *Proceedings of the London Mathematical Society*, vol. 102, no. 4, pp. 731–766, 2011. [Online]. Available: https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/plms/pdq035

[9] W. Suksompong, "Tournaments in computational social choice: Recent developments," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 4611–4618, survey Track. [Online]. Available: https://doi.org/10.24963/ijcai.2021/626

[10] S. P. Radziszowski, "Small Ramsey numbers," *Electronic Journal of Combinatorics*, vol. 1000, 2011.

[11] N. J. A. Sloane and T. O. F. Inc., "The on-line encyclopedia of integer sequences," 2020. [Online]. Available: http://oeis.org/

[12] J. W. Moon, *Topics on Tournaments*. Holt, Rinehart and Winston, 1968, p. 28.

[13] P. L. Erdős and A. Rényi, "Asymmetric graphs," *Acta Mathematica Academiae Scientiarum Hungarica*, vol. 14, pp. 295–315, 1963.

[14] P. Erdős and L. Moser, "On the representation of directed graphs as unions of orderings," in *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, vol. 9, 1964, pp. 125–132.

[15] D. Neiman, J. Mackey, and M. Heule, "Tighter bounds on directed Ramsey number R(7)," *arXiv preprint arXiv:2011.00683*, 2020.

[16] G. S. Tseitin, "On the complexity of derivation in propositional calculus," in *Automation of reasoning*. Springer, 1983, pp. 466–483.

[17] C. Sinz, "Towards an optimal CNF encoding of boolean cardinality constraints," in *International conference on principles and practice of constraint programming*. Springer, 2005, pp. 827–831.

[18] M. Codish, A. Miller, P. Prosser, and P. Stuckey, "Breaking symmetries in graph representation," in *International Joint Conference on Artificial Intelligence*, 08 2013, pp. 510–516.

[19] A. Itzhakov and M. Codish, "Breaking symmetries in graph search with canonizing sets," *Constraints*, vol. 21, no. 3, p. 357–374, jul 2016. [Online]. Available: https://doi.org/10.1007/s10601-016-9244-z

[20] A. Ignatiev, A. Previti, and J. Marques-Silva, "SAT-based formula simplification," in *SAT*, 2015.

[21] B. D. McKay and A. Piperno, "Practical graph isomorphism, ii," *Journal of symbolic computation*, vol. 60, pp. 94–112, 2014.

[22] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger, "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020," in *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, ser. Department of Computer Science Report Series B, T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, Eds., vol. B-2020-1. University of Helsinki, 2020, pp. 51–53.

[23] A. Sánchez-Flores, "On tournaments free of large transitive subtournaments," *Graphs and Combinatorics*, vol. 14, pp. 181–200, 1998.