

# Optimal Symmetry Breaking for Tournaments

Chris Lambert Open Access   

Dummy University Computing Laboratory, [optional: Address], Country

My second affiliation, Country

Evan Lohn<sup>1</sup>  

Department of Informatics, Dummy College, [optional: Address], Country

Marijn Heule<sup>2</sup>  

Department of Informatics, Dummy College, [optional: Address], Country

---

## 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  $\Omega(n \log n)$  unit clauses. Finally, we show the utility of our new isolators in computations of tournament Ramsey numbers.

**2012 ACM Subject Classification** Replace `ccsdsc` macro with valid one

**Keywords and phrases** Satisfiability Symmetry-breaking Directed-graphs

**Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

**Funding** *Chris Lambert Open Access:* (Optional) author-specific funding acknowledgements

*Evan Lohn:* [funding]

*Marijn Heule:* [funding]

**Acknowledgements** I want to thank ...

## 1 Introduction

Our main goal was to extend prior work [?] with undirected graphs and generate optimal, perfect isolators for tournaments (complete, directed graphs). There were several places in which perfect isolator generation differed for tournaments.

### 1.1 Global Clause Properties

Because edgeless and complete graphs are isomorphism classes for any  $n$  in the undirected case, every clause of an undirected graph isolator 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 satisfying  $AB \wedge BC \rightarrow AC$  for every three vertices  $A, B, C$  in the tournament. Unlike the undirected case (with exactly 1 empty graph and 1 complete graph), there are  $n!$  isomorphic transitive tournaments on  $n$  vertices. It is possible to select the particular transitive tournament  $TT$  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  $AB$  s.t.  $A < B$  in vertex numbering. However, this approach is not yet proven to admit an optimal isolator.

---

<sup>1</sup> Optional footnote, e.g. to mark corresponding author

<sup>2</sup> Optional footnote, e.g. to mark corresponding author



## 1.2 Unit Clauses

Another consequence of undirected isolators requiring at least one positive and one negative literal per clause is that undirected isolators have no unit clauses. However, tournament isolators have no such restriction and are thus allowed to have unit clauses. For  $n < 7$ , the majority of clauses are unit in the optimal tournament isolators we present below.

## 1.3 Ease of Isomorphism

Another interesting difference between undirected graphs and tournaments is the low number of isomorphism classes for tournaments when  $n$  is small (see table ??). Intuitively, this happens because it is “easy” for tournaments to be isomorphic. The two options for the edge between vertices  $A$  and  $B$  in the undirected case are  $AB$  existing or not existing. Crucially, an undirected graph  $G$  will never be isomorphic to  $G'$  constructed by adding or removing an edge of  $G$  (an operation that can be seen as “flipping” an edge to its other possibility). However, “flipping” an edge of a tournament  $T$  (changing the edge’s direction) will produce  $T' \simeq T$  iff the two vertices  $A$  and  $B$  of the flipped edge had the same edges to the rest of the graph (the isomorphism is via the permutation that swaps  $A$  and  $B$ ).

# 2 Methodology for Generating Isolators

## 2.1 Generation of Map Files

Both of our approaches to generating isolators relied on the creation of “map” files: text files associating each graph 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 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 [?]. labelg produced a file where each graph was converted to its canonical form. We gave each canonical form a unique label and output the arc (directed edge) indices of each original graph alongside its canonical form. Let  $I_{ab}^n$  be the index of the arc from  $a$  to  $b$  in an  $n$ -vertex tournament.

The output format of the original graph arcs was extensible for higher  $n$ , satisfying the property that  $I_{ab}^n = I_{ab}^{n+1}$ . In particular, with  $K = n(n-1)/2$  as the largest edge index for graphs of size  $n$ , the edges  $(1, n+1), (2, n+1), \dots, (n, n+1)$  for  $n+1$ -vertex tournaments were labeled  $K+1, K+2, \dots, K+n$ , respectively. We will drop the superscript  $n$  when referring to  $I_{ab}^n$  in the future, as its value does not depend on  $n$ .

The arc indices described thus far only refer to the arcs in the direction of increasing vertex index, i.e. arc index 2 corresponds to  $(1, 3)$  and not  $(3, 1)$ . We further define  $I_{ab} = -I_{ba}$ . However, when outputting graphs for the map file we simply exclude all negative arc indices to reduce file size, as the negative edges can be inferred with knowledge of  $n$  because tournaments are complete graphs.

## 2.2 Optimal, Perfect Isolator SAT encoding

We re-implemented the basic structure of the perfect isolator encoding for undirected graphs [?] with several modifications that mostly stem from the differences between undirected graphs and tournaments. The specific problem we encoded was “Is there a set of  $k$  non-unit clauses and a set of unit clauses whose union is a perfect isolator for  $n$ -vertex tournaments?”. This first section will explain the encoding of the simpler problem: “Is there a set of  $k$  clauses

whose union is a perfect isolator for  $n$ -vertex tournaments”, which will be modified to the final problem definition in following sections.

For each isolator clause  $c$  and arc literal  $a$ , we defined the variable  $In(c, a)$  to represent “ $a$  is in  $c$ ”. Then, for each graph  $G$  from the map file for  $n$  vertices, we define a variable  $Kills(G, c)$  to mean “clause  $c$  does not admit graph  $G$ ”. This specification is implemented as follows with a Tseitin encoding [?] to handle the equality:

$$Kills(G, c) = \bigvee_{a \in A_G} In(c, a) \quad (1)$$

where  $A_G$  is the set of arc literals corresponding to the arcs present in graph  $G$ . We also define the variable  $Canon(G)$  for all graphs  $G$  from the map file, meaning “Graph  $G$  is the canonical representative of its isomorphism class.” We implement this as follows (again using Tseitin to handle the equality and conjunctions):

$$Canon(G) = \bigwedge_c \neg Kills(G, c) \quad (2)$$

Finally, for each isomorphism class  $E$  extracted from the mapfile as a set of graphs, let  $E_1$  be the set  $Canon(G) | G \in E$ . We add the following clauses to the formula for each isomorphism class  $E$ :

$$AMO(E_1) \wedge \bigvee_{c \in E_1} c \quad (3)$$

Where  $AMO$  is the “at most one” operation on a set of clauses, implemented via a Sinz encoding [?]. In total, a satisfying assignment to this problem corresponds to a perfect isolator on  $k$  non-unit clauses. If the problem is UNSAT for  $k$  and SAT for  $k + 1$  non-units, this proves that the perfect isolator with  $k + 1$  non-units is optimal.

## 2.3 Breaking Symmetries in the Encoding

An easy symmetry in the encoding was the order of the isolator clauses. Clearly, reordering clauses of an expression in CNF does not affect its behavior. To resolve this, we added clauses that ensured a lexicographic ordering of the clauses in the resulting isolator. For every adjacent pair of clauses  $c_1$  and  $c_2$ , we fixed some ordering of every literal that may appear in them  $a_1, a_2, \dots, a_n$ , and then created variables  $e_0, e_1, \dots, e_n$  where  $e_i$  represents that clauses  $c_1$  and  $c_2$  are equivalent when considering only the first  $i$  literals.  $e_0$  is always true, and to maintain the others we added the clauses

$$e_i = e_{i-1} \wedge (In(c_1, a_i) = In(c_2, a_i))$$

via the Tseitin transformation for every  $1 \leq i \leq n$ . Then, we enforced a lexicographic ordering by requiring that for every  $i$  such that  $c_1$  and  $c_2$  were equal up to  $i$ , that if clause  $c_1$  contained  $a_i$  then  $c_2$  must also contain  $a_i$ . Explicitly, we added the following requirement via the Tseitin transformation for every  $1 \leq i \leq n$  and pair of adjacent clauses  $c_1, c_2$

$$e_{i-1} \wedge In(c_1, a_i) \implies In(c_2, a_i)$$

101 and furthermore we required that  $e_n$  is false to ensure a strict ordering. When trying to  
 102 find an isolator with  $C$  clauses, this reduces the search space by a factor of  $C!$  as only one  
 103 permutation of a given set of clauses should be considered as opposed to all  $C!$  of them.

We can find another symmetry in the ordering of the vertices. We can reorder the vertices, and relabel the edge literals in an isolator accordingly, and this will result in another isolator that accepts the same graphs that the original did, but with the vertices permuted which keeps it in the same isomorphism class. To break this, recall from the introduction that any tournament isolator must admit exactly one transitive tournament. We may apply a permutation to this transitive graph (and the isolator) that turns it into a canonical transitive graph of  $1 \rightarrow 2 \rightarrow 3 \rightarrow \dots \rightarrow n$ . Note that every edge in this graph goes from a lower numbered vertex to a higher numbered vertex, and so it would be a positive literal in our encoding and requires that this permuted isolator must have at least one positive literal in every clause. 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  $c$

$$\bigvee_{a \in A_p} In(c, a)$$

104 with  $A_p$  being the set of all positive literals. When trying to find an isolator for  $n$  vertices,  
 105 this reduces the search space by a factor of  $n!$  since the solver is guaranteed to only consider  
 106 isolators for which the canonical transitive graph is the  $1 \rightarrow 2 \rightarrow \dots \rightarrow n$  described above.

## 107 2.4 Encoding Unit Propagation

Under the still rather naive encoding described above, the solver tended to find isolators where every clause was very large, but after applying unit propagation resulted in a much smaller, cleaner isolator that consisted of many units. 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 ‘unit clause’ variables  $Unit(a)$  representing “literal  $a$  is a unit clause”. We then required that the isolator be already unit-propagated with respect to the ‘unit clause’ literals by adding the requirement

$$\neg In(c, a) \vee \neg Unit(a)$$

We also had to account for these units killing graphs in the *Canon* clauses, so those were updated to be

$$Canon(G) = \bigwedge_c \neg Kills(G, c) \wedge \bigvee_{a \in A_G} \neg Unit(\neg a)$$

108 Finally, we have to count these special unit literals towards the clause count in determining  
 109 optimality. However, in the practical sense units are essentially zero additional cost to an  
 110 isolator: when used in a SAT solver they will be instantly eliminated through unit propagation  
 111 and can only serve to reduce the complexity of the resulting problem. From this line of  
 112 reasoning, we might consider an ‘optimal isolator’ to not just have the minimal number of  
 113 clauses, but the minimal number of non-unit clauses. Note that since units cannot exist in  
 114 the undirected case, this alternative definition of optimality is consistent with the prior work  
 115 on the undirected case.

116 We decided to not count units towards the clause count for use in determining optimality  
 117 because of this. This is not only a more useful definition, but results in a less complex  
 118 encoding as we do not need to count objects of two different types, and we do not need to

ensure that the non-unit clauses are actually not units. The last is due to the fact that since units are ‘free’ in our setup and non-units are costly, a solution to our encoding that uses a non-unit clause position to encode a unit will have wasted that position, and a solution that holds for fewer allowed non-units will simply move that clause to a unit literal position.

Note that using the direction symmetry breaking above, we only need to consider positive unit literals, which drastically reduces the search space in this step.

## 2.5 Provable Units

While constructing smaller isolators using the techniques above, we opted to manually inspect our results and see what patterns they shared. Specifically, in the isolators up through 6 vertices there were some very noticable patterns in the unit clauses they contained, at least after permuting the isolators’ vertices to line them up with each other. The easiest structure to notice was that all of these isolators contained an  $n - 1$ -length path, or a directed path that touches every vertex exactly once. In graph theory literature, this is known as a Hamiltonian path.

As it turns out this pattern of units admits a proof, and so without worry we can include it in the basis for any isolator.

► **Theorem 1.** *All tournament graphs admit a Hamiltonian path.*

**Proof.** Proceed by induction. The statement is trivial for 1- and 2- vertex tournaments, since in the former the empty path is Hamiltonian and in the latter the sole edge is the path. Now, given an  $m$ -length path in a tournament for  $2 \leq m < n - 1$ , argue that an  $m + 1$ -length path exists.

Label the vertices on the given path by 1 through  $m + 1$ , and since  $m + 1 < n$  there exists some other vertex  $a$ . If there is an edge from  $a$  to 1 or  $m + 1$  to  $a$ , then we can trivially extend the path to include  $a$ , so take the other case where there is an edge from 1 to  $a$  and an edge from  $a$  to  $m + 1$ . Consider the smallest  $1 \leq i \leq m + 1$  such that there is an edge from  $a$  to  $i$ . Clearly  $i > 1$  as there is an edge from 1 to  $a$ , and clearly  $i$  is well-defined since the set of  $i$  where there is an edge from  $a$  to  $i$  is nonempty as it contains  $m + 1$ . Then, we have that  $i - 1 \geq 1$ , and furthermore there is no edge from  $a$  to  $i - 1$ , so there is an edge from  $i - 1$  to  $a$ . Therefore, we can take  $i - 2$  steps on the original path from vertex 1 to vertex  $i - 1$ , one step from  $i - 1$  to  $a$ , one step from  $a$  to  $i$ , and  $m + 1 - i$  steps on the original path from vertex  $i$  to vertex  $m + 1$ . This new path has length  $(i - 2) + (1) + (1) + (m + 1 - i) = m + 1$ , completing the induction step and thus the proof. ◀

It is worth noting that we do not have a proof that fixing a Hamiltonian path in an isolator is optimal. However, getting a large set of units for free is extremely helpful, especially since our map-file approach does not need to consider graphs that are made impossible by a basis set of units. Including these units reduces the map file size and generation time by a factor of  $2^{n-1}$ , which makes map file-based approaches still viable for up to  $n = 8$ , and the risk of it being suboptimal (especially by a large factor) seems intuitively very low.

## 2.6 Incremental Isolators

One way to view the inclusion of a Hamiltonian path in our problem definition for each  $n$  is that the unit clauses used to encode the isolator search problem for  $n + 1$  vertices build off the set of isolators used in the  $n$  case (by simply adding the positive unit corresponding to the arc  $(n, n + 1)$ ). We further generalize this idea by showing that it is always possible

to extend an isolator for  $n$ -vertex tournaments (and, incidentally, undirected graphs) to an isolator on  $n + 1$  vertices. Let  $P_n$  be an isolator, defined here as a predicate on  $n$ -vertex tournaments that admits at least one tournament from each isomorphism class. Further,  $[G]_n$  for a tournament  $G$  on  $n + 1$  vertices refers to the tournament formed by removing the  $n + 1$ th vertex of  $G$ .

► **Theorem 2.** *For any  $P_n$ , a predicate  $P_{n+1}$  that admits the  $n + 1$ -vertex  $G$  iff  $P_n([G]_n)$  holds is an isolator on  $n + 1$ -vertex tournaments.*

**Proof.** Consider any canonical set containing representatives  $R$  from each isomorphism class of  $n + 1$ -vertex tournaments. For each  $R$ ,  $[R]_n$  is an  $n$ -vertex tournament and therefore a member of one of the isomorphism classes on  $n$ -vertex tournaments. By definition, the arbitrary given  $P_n$  admits some  $r$  such that  $r \simeq [R]_n$ . Any two isomorphic  $n$ -vertex tournaments are related by some vertex permutation  $v_n$ , where we define  $v_n(r)$  for a tournament  $r$  to be the tournament constructed by the application of  $v_n$  to each vertex of  $r$ . Therefore we have  $v_n(r) = [R]_n$  for some  $v_n$ .

This  $v_n$  can be extended to tournaments on  $n + 1$  vertices via  $v_{n+1}(x) = v_n(x)$  when vertex  $x$  is not vertex  $n + 1$ , and  $v_{n+1}(x) = n + 1$  otherwise. Applying each  $v_{n+1}^{-1}$  (the inverse permutation of  $v_{n+1}$ ) to its corresponding  $R$  yields a new canonical set  $S$  for  $n + 1$ -vertex tournaments because tournament isomorphism classes are closed under vertex permutation. We claim that  $P_{n+1}$  admits all members of  $S$ , making it an isolator. To check this, note that each new canonical representative is of the form  $v_{n+1}^{-1}(R)$  for some  $R$ . By definition of  $P_{n+1}$ , we must confirm that  $P_n([v_{n+1}^{-1}(R)]_n)$  holds. However,  $[v_{n+1}^{-1}(R)]_n = v_n^{-1}([R]_n) = r$ , and  $P_n(r)$  holds by the definition of  $r$ , completing the proof. ◀

The idea for incremental isolators and its following proof fell out of the observation that the optimal isolator for  $n = 5$  had striking similarities to the optimal isolator for  $n = 4$ . In fact, it is possible to build directly off the entire  $n = 4$  isolator when constructing an optimal isolator for  $n = 5$ . However, in general a perfect  $n + 1$ -vertex isolator constructed incrementally from an optimal, perfect  $n$ -vertex isolator is not optimal, with a counterexample appearing at  $n = 6$ . We note that it is also possible to build  $n + 1$ -vertex incremental isolators from any subset of clauses from an  $n$ -vertex isolator. The most natural choice of subset when pursuing optimal isolators would be the unit clauses, because it kills the most non-canonical graphs. We were not able to prove that isolators built from incremental units are optimal or suboptimal in general, but it produces optimal isolators in practice up to  $n = 6$ . The incremental units approach is a generalization to the approach of fixing the hamiltonian path units, and allows a further reduction of the space of considered isolators.

## 2.7 Probing

In addition to the SAT encoding approach to isolator generation, we also generated isolators using a method from prior work called “random probes” [?]. 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.

## 2.8 $n \log n$ proof

If a  $TT_k$  is known to exist within every member of the class of  $n$ -vertex tournaments, each 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 isolator that fixes a  $TT_k$  on the class of  $n$ -vertex tournaments is valid, i.e. will not remove any equivalence classes. 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. Therefore, tournament Ramsey numbers can be used to construct sets of unit clauses for tournament isolators: we will refer to this as TT-fixing.

Let  $units(n)$  be the function that returns the maximum possible 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 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 **CITE** we have that  $R(k) \leq 2^{k-1}$ , 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

$$units(n) = \frac{k(k-1)}{2} + units(n-k) \quad (4)$$

By the inductive hypothesis,

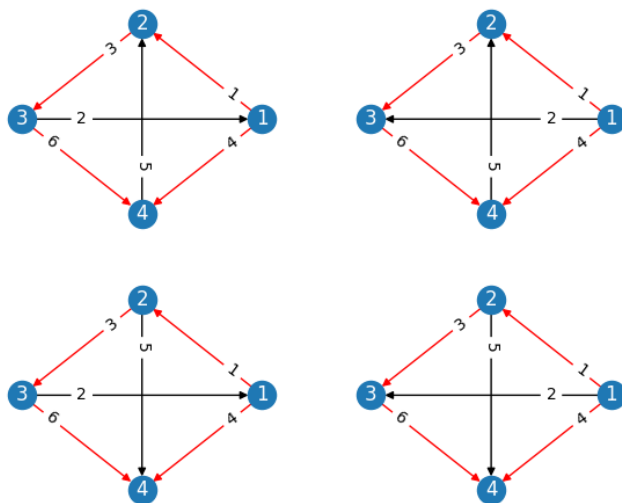
$$units(n-k) \geq \sum_{i=1}^{n-k} \lfloor \log_2(i) \rfloor / 2 \quad (5)$$

Next, we have that

$$k \frac{k-1}{2} = \sum_{i=n-k+1}^n \lfloor \log_2(n) \rfloor / 2 \geq \sum_{i=n-k+1}^n \lfloor \log_2(i) \rfloor / 2 \quad (6)$$

Combining the two lower bounds ((?) and (??)) for the terms of ?? completes the proof:  $units(n) = \frac{k(k-1)}{2} + units(n-k) \geq \sum_{i=n-k+1}^n \lfloor \log_2(i) \rfloor / 2 + \sum_{i=1}^{n-k} \lfloor \log_2(i) \rfloor / 2 = \sum_{i=1}^n \lfloor \log_2(i) \rfloor / 2$  as desired.

Additionally, there is a long-known result from graph theory that implies that there are at most  $O(n \log n)$  units possible. By the orbit-stabilizer theorem **need to cite? or just explain more why it applies probably**, 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 Renyi **CITE** proved that as  $n$  approaches infinity, the proportion of graphs of size  $n$  with multiple automorphisms approaches 0. Therefore, a proportion of graphs approaching 1 have equivalence classes of size  $n!$ , so the number of equivalence classes is



■ **Figure 1** Isomorphism class representatives admitted by a particular isolator for 4-vertex tournaments. Red edges are edges fixed by unit clauses of the isolator, and the isolator has only unit clauses.

approximately  $\frac{2^{\binom{n}{2}}}{n!} \simeq \frac{2^{\binom{n}{2}}}{2^{n \log n}} = 2^{\binom{n}{2} - n \log n}$ . An isolator with  $k$  unit clauses for  $n$ -vertex  
 graphs admits at most  $2^{\binom{n}{2} - k}$  equivalence class representatives, so the number of units is  
 also upper-bounded by  $n \log n$ .

### 3 Results

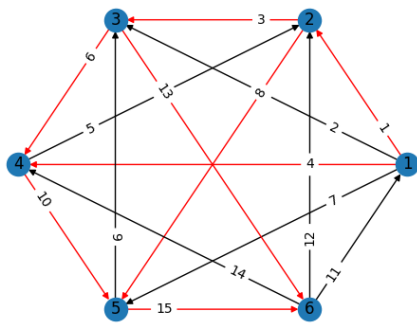
#### 3.1 Small Optimal Isolators

Our SAT encoding allowed us to compute optimal isolators up to  $n = 6$ . Figures ?? and ?? graphically display optimal, perfect isolators for  $n = 4, 5$  by displaying a graph from each isomorphism class. Figure ?? 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  $n = 4, 5$  we also computed the number of different perfect optimal isolators up to isomorphism, where we define two isolators to be isomorphic if they are equivalent under some vertex permutation. There is only a single perfect optimal isolator for  $n = 4$ , while there are 5 different (perfect optimal) isolators for  $n = 5$ . Interestingly, each of the 5 can be expressed such that they have the same 6 unit clauses, and differ only in the 2 non-unit clauses present in optimal  $n = 5$  isolators.

#### 3.2 Larger Isolators

In scaling up to larger graphs for isolators, we quickly run into barriers in our ability to solve problems in our encoding after  $n = 6$ . For  $n = 7$ , solving the SAT instance directly became clearly infeasible (taking several days without any signs of progress). Random probing was the only method that allowed us to find an 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 ?? describes the best isolator found during the semester for  $n = 1$  through 7. Several thousand probes were required to find our best known isolator for  $n = 7$ .





■ **Figure 2** 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.

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

■ **Table 1** The smallest isolators found by number of non-unit clauses for all vertex counts up to 7, as well as the units in that isolator. These are known optimal through 6 vertices.

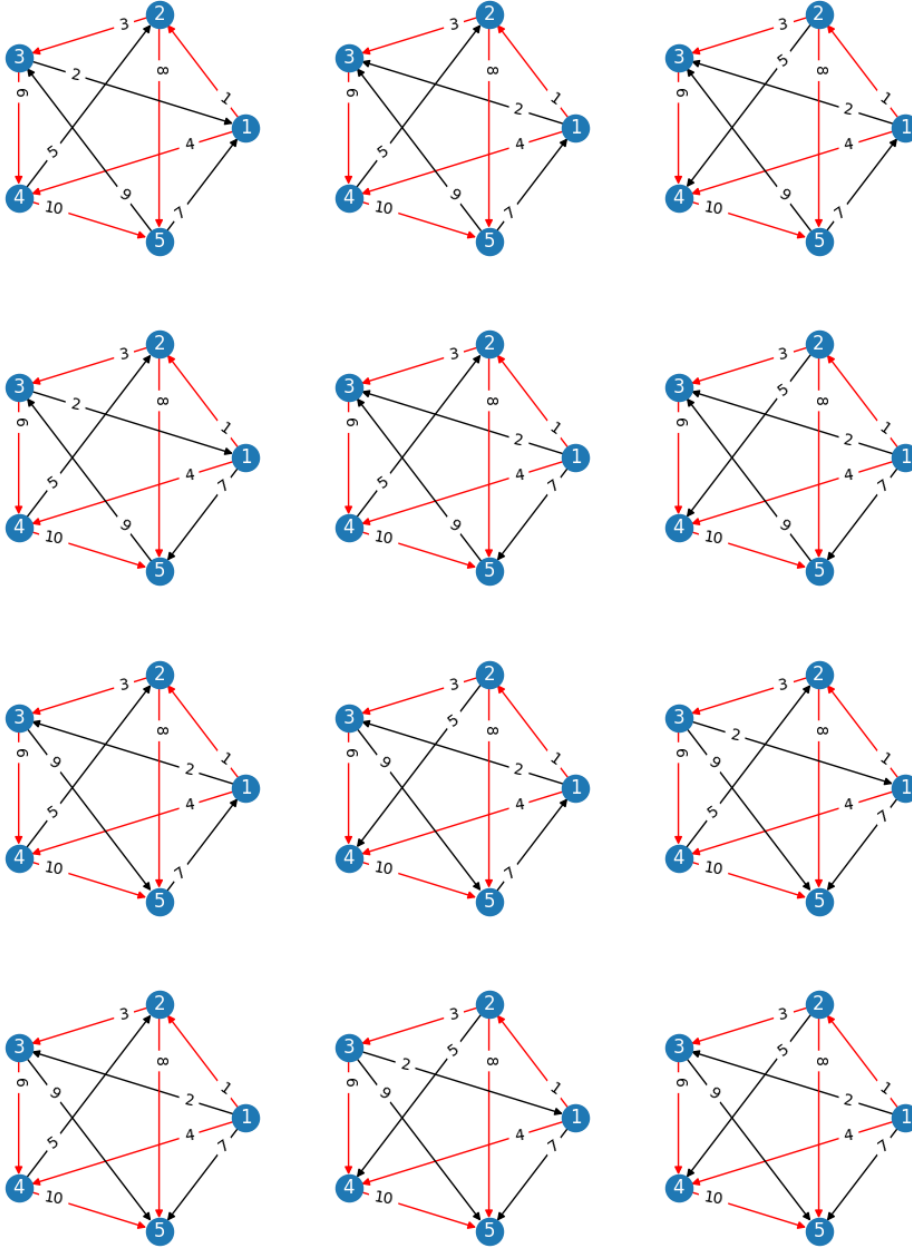
268 **4 Future Work**

269 **4.1 Removing the Map File**

270 In our current setup, generation of the SAT problems involves mapping all possible tournament  
271 graphs to their isomorphism class. Since this grows at a rate of  $2^{O(n^2)}$ , this becomes  
272 prohibitively large very quickly. Even only considering graphs that satisfy a basis set of units,  
273 such as the Hamiltonian path or the prior isolator’s units, these approaches quickly become  
274 infeasible as soon as  $n = 9$ . By only generating the graphs that satisfy the previous isolator  
275 (and not just the units), this approach could potentially be extended for slightly longer, albeit  
276 at the cost of almost certainly suboptimal isolators. A more clever encoding, or some way of  
277 encoding some of the ideas of graph isomorphism into the problem itself, could potentially  
278 drastically reduce the size of the problems and find isolators for much larger tournaments.

279 **4.2 Isolator Applications**

280 Graph existence problems, or problems that relate to whether a graph satisfying some  
281 conditions exist, are notoriously difficult to solve. The Ramsey numbers, Conway’s 99-graph  
282 problem, and many other problems can be phrased (or are most directly phrased) in this  
283 manner remain open problems in their fields even for relatively small instances. For this  
284 reason computational approaches are tempting since they might be able resolve smaller  
285 instances of these problems, at least with an efficient enough algorithm. Isolators provide a  
286 mechanism by which SAT solvers might be able to go fast enough to help in this area, since  
287 they drastically reduce the search space of possible graphs and leave only the non-redundant



■ **Figure 3** 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  $2 \vee 7 \vee -9$ .

288 cases to check.

289     The main motivation towards finding these tournament graph isolators is their potential  
290 uses in computing tighter bounds on tournament Ramsey numbers — the directed analog  
291 of Ramsey numbers for tournaments. The tournament Ramsey number at  $n$  is defined to  
292 be the smallest  $m$  such that all tournaments on  $m$  vertices contain a transitive (cycle-free)  
293 subtournament of  $n$  vertices. SAT approaches have been taken with this problem before to  
294 tighten bounds on the directed Ramsey number  $R(7)$  [?]. From this it seems very likely that  
295 directed isolators — boosted by their simplicity / number of unit clauses in them — should  
296 be able to be applicable on this problem and potentially tighten those bounds further.

## 297 — References —

- 298 1 Marijn JH Heule. Optimal symmetry breaking for graph problems. *Mathematics in Computer Science*, 13(4):533–548, 2019.
- 299 2 Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of symbolic computation*, 60:94–112, 2014.
- 300 3 David Neiman, John Mackey, and Marijn Heule. Tighter bounds on directed ramsey number  $r$   
301 (7). *arXiv preprint arXiv:2011.00683*, 2020.
- 302 4 Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In  
303 *International conference on principles and practice of constraint programming*, pages 827–831.  
304 Springer, 2005.
- 305 5 Grigori S Tseitin. On the complexity of derivation in propositional calculus. In *Automation of reasoning*, pages 466–483. Springer, 1983.
- 306
- 307
- 308