TOURNAMENTS IN DIRECTED GRAPHS: A REVIEW

Kousshik Raj, Robin Babu

Indian Institute of Technology, Kharagpur

ABSTRACT

Pairwise comparison is the process of establishing a relation among a set of entities when comparing them a pair at a time. A tournament is a representation of such pairwise comparison data. Formally, in graph theory, a tournament on n-vertices is a directed graph with every pair of vertices having a directed edge between them. We can gather a lot of information on the structural relation between the entities by analyzing the tournament graph. In this paper, we will first look at some of the interesting properties of tournaments and then focus on a specific type of tournament called transitive tournament. Later, we will present algorithms for Hamiltonian Path, Feedback Vertex Set, Dominating Set in tournaments. Though tournaments have a more ordered structure than a generic directed graph, most of these problems still do not have a polynomial time solution, though we will see that in some cases we have a better algorithm than the generic directed graph.

Index Terms— Tournament, Directed Graph, Hamiltonian Path, Feedback Vertex Set, Dominating Set

1. INTRODUCTION

A tournament on n-vertices is an orientation of a complete graph on n-vertices (K_n), i.e, each edge is assigned a direction. Fig. 1 shows a tournament on four vertices. Usually, tournaments are used to represent pairwise comparison data between entities. Tournaments are one of the interesting graphs researched in graph theory. Though ironically, most of the important properties of tournaments were first investigated in a work of biophysics, where dominance relation in flocks of chickens were analysed. Currently, tournaments are used extensively in the study of voting theory and social theory among several other things.

The name *tournament* comes from the term *round-robin tournament*. As we can see, tournaments can also be seen as a representation of the outcome of a round-robin tournament where each player encounters every other player exactly once and the winner is determined every time (no draws). In the tournament digraph, each vertex represents a player, and there is a directed edge from a winner to a loser. If a player a beats player b, we say that a dominates b. This is generally denoted by $a \rightarrow b$. The score of a vertex v is the number of wins this vertex enjoys, or we can equivalently say, the number of outward going arcs from this vertex. In Fig. 1, the scores of vertices $1, 2, 3, 4$, are $2, 1, 1, 2$, respectively.

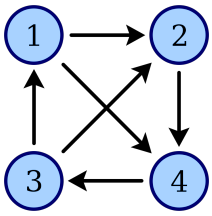

Fig. 1. A tournament on 4 vertices

We will first look at some common notations involved while handling tournaments. Let T be a tournament. Then $V(T)$ represents the set of vertices of the tournament and $E(T)$ denotes the set of directed edges of the tournament, with each edge represented by (a, b) where there is an edge from a to b. Let, $X \subset V(T)$, be a subset of vertices of the tournament. Then, the subtournament $T[X]$ is the tournament whose set of vertices is X, and whose edge set consists of all directed edges in $E(T)$ whose both endpoints lie in X. Here we say that, $T[X]$ is induced in the tournament T by X. Furthermore, for a vertex $v \in V(T)$, we denote $T - v$ as the induced subtournament of T resulting from deleting vertex v (and all edges with v as an endpoint). We can define $T - X$ similarly, where $X \subset V(T)$.

2. RELATED WORKS

One of the foremost works on tournaments can be attributed to Landau [1], in which he established necessary and sufficient conditions for a score set of a tournament. Winston et al. [2] further proceeded to prove an asymptotic bound for the number of score sequences for a tournament.

Camion [3] focused on strongly connected tournaments and the existence of Hamiltonian cycle in it. He also presented an algorithm for finding Hamiltonian cycle in a strongly connected tournament in $O(n^3)$. Soroker [4] improved the algorithm to $O(n^2 \log n)$ which was further optimised to $O(n^2)$ by Manoussaki [5]. Morrow et al. [6] implemented an efficient algorithm to find the largest cycle in a tournament which runs in $O(n^3)$ time.

Bar-noy et al. [7] presented an algorithm for finding a

Thanks to Prof. Bhargab B. Bhattacharya for his guidance.

Hamiltonian path in tournaments as a translation of sorting algorithm based on the one to one correspondence between the set of minimal feedback vertex sets and the set of Hamiltonian paths and thus achieving a runtime of $O(n \log n)$. Furthermore, the algorithm maintains the tradeoff between processors and run time for sorting, thus running in time $O(\log n)$ when $O(n)$ processors are used.

Thomassen [8] characterised weakly Hamiltonian connected tournaments and weakly pan-connected tournaments completely and established various results on connectivity of tournaments. Guo et al. [9], generalised some of these results for tournaments in terms of its minimum in-degree δ^-, minimum out-degree δ^+ and irregularity i_g.

On the note of k-paradoxical tournaments, Erdos [10] established a lower bound on the number of vertices for which the existence of a k-paradoxical tournaments is guaranteed, whereas Szekeres et al. [11] gave a necessary requirement on the number of vertices for the existence of a k-paradoxical tournament. Graham et al [12] presented an explicit construction method for a k-paradoxical tournament, which is known as Paley tournament.

3. PROPERTIES

We shall visit some important theorems and properties of Tournaments. These are important for the algorithms that are stated further in the paper and reveal important information that can be gained due to the structure of a Tournament. Some problems that are considered NP-Hard for normal graphs can be in fact calculated for tournaments in polynomial time.

3.1. Redei's Theorem

Redei's Theorem [13] states that ever tournament has at least one directed Hamiltonian path.

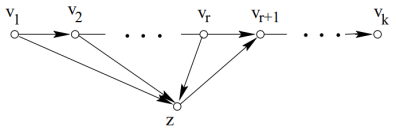

Fig. 2. Redei's Theorem proof

Proof. Assume the existence of a path of k vertices $v_1, v_2, ...v_k$. Let v_{new} be a new vertex not lying on this path. First we check if $v_{new} \rightarrow v_1$ is an edge. If so then we insert v_{new} at the beginning of the path, and thus $v_{new}, v_1, ...v_k$ is a path of $k+1$ vertices.

If $v_{new} \rightarrow v_1$ is not an edge, then we check with v_2. If $v_{new} \rightarrow v_2$ is an edge then we can insert v_{new} between v_1 and v_2. If not we continue in a similar manner for the first vertex v_r such that $v_{new} \rightarrow v_r$ is an edge and we insert v_{new} in the corresponding place to get a new path of size $k+1$. However if no such v_r is present then that means $v_k \rightarrow v_{new}$ is an edge, thus we can place v_{new} at the end of the path.

3.2. Camion-Moon Theorem

Camion-Moon theorem [3] states that every strongly connected tournament has a directed Hamiltonian cycle. We will prove this by induction.

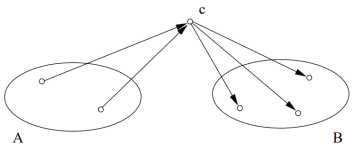

Fig. 3. Base case: Camion Moon Theorem

Base Case. Let c be an arbitrary vertex of G, we divide the remaining vertices into two sets A and B. A vertex a is in A if and only if there is an arc (a, c) from a to c but no arc (c, a) from c to a. Similarly, a vertex b is in B if and only if there is an arc (c, b) from c to b but no arc (b, c) from b to c.

Now, $|V| \geq 3$, so that both A and B must be nonempty. If $B = \phi$, for example, then there would be no directed path from c to any vertex in A, which contradicts the fact that G is strongly connected. Also, if all arcs between vertices in A and vertices in B go from a vertex in A to a vertex in B, then again there would be no directed path from any vertex in B to any vertex in A, again contradicting the fact that G is strongly connected. Therefore, there is an arc from some vertex $b \in B$ to a vertex $a \in A$, and we have a directed cycle of length 3 given by $a \rightarrow c \rightarrow b \rightarrow a$.

Induction. Now, suppose that the digraph G has a directed cycle of length k for some k with $3 \leq k < |V|$, we will show that this implies that G has a directed cycle of length $k+1$. We divide the $|V| - k$ vertices that are not in the cycle into three sets A, B and C, where vertices in A have no arcs coming from the cycle, vertices in B have no arcs going to the cycle, and vertices in C have arcs coming from the cycle and arcs going to the cycle. We consider two cases.

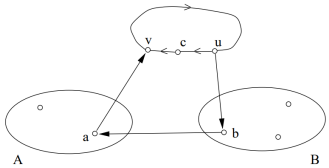

Fig. 4. Case 1: Camion Moon Theorem

1. Suppose C is empty, then both A and B must be nonempty, otherwise G would not be strongly connected. As in the base case, there is an arc going from some vertex $b \in B$ to some vertex $a \in A$. If c is any

vertex in the cycle, we let u and v be the vertices immediately preceding and immediately following c on the cycle, respectively. If we replace the portion of the cycle $u \to c \to v$ by the directed path $u \to b \to a \to v$, we obtain a directed cycle of length $k+1$ as desired.

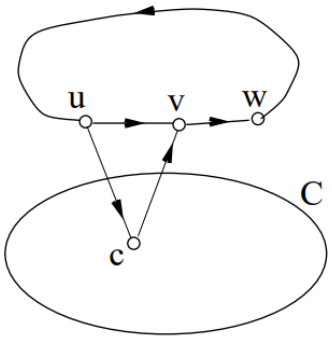

Fig. 5. Case 2: Camion Moon Theorem

2. Suppose that C is not empty, and let $c \in C$. Let u be a vertex on the cycle such that (u, c) is an arc, and let v be the vertex immediately following u on the cycle. If (c, v) is an arc, then we can insert c into the cycle between u and v. If (c, v) is not an arc, then since we have a tournament, (v, c) is an arc, and we look at the vertex w immediately following v on the cycle. If (c, w) is an arc, then we can insert c into the cycle between v and w. If not, we can proceed as in the proof of Redei's theorem, and we will eventually find a place to insert c into the cycle between consecutive vertices in the cycle. Thus, in this case also, we obtain a directed cycle of length $k+1$ as desired .

From the principle of mathematical induction for each $3 \leq k \leq V$, the directed graph G has a directed cycle of length k. Thus a strongly connected tournament graph has a Hamiltonian cycle. It also follows that the graph is pan-cyclic.

3.3. Transitive Tournament

A Transitive Tournament is a tournament in which all edges follow the transitive property. That is if there exists edges of the form $a \to b$ and $b \to c$, then there must exist an edge $a \to c$. Transitive tournaments have a lot of interesting properties. We shall discuss them in the following subsections.

3.3.1. Transitive Tournaments are Acyclic

Let us assume that a tournament is both transitive and cyclic in nature. Let $v_1, v_2....v_n, v_1$ represent the cycle present in the Tournament. since $v_1 \to v_2$ is an edge and $v_2 \to v_3$ is an edge, and since it is transitive in nature, $v_1 \to v_3$ must also be present in the tournament. Now, since we have edges $v_1 \to v_3$ and $v_3 \to v_4$ we must have the edge $v_1 \to v_4$. Thus we can go on to show that we must also have the edge $v_1 \to v_n$. This however is a contradiction, as we know $v_n \to v_1$ is an edge to complete the cycle. Thus, transitive tournaments cannot by cyclic in nature.

3.3.2. Transitive Tournaments have a strict total ordering

A Transitive Tournament G is a totally ordered set, where a vertex u is greater than another vertex v if there exists an arc from u to v. We prove by induction on the number of vertices n. The cases $n \in 1, 2, 3$ are immediate. Suppose now that G is a tournament on n vertices with $n > 3$ such that G is transitive.

We first claim that there exists a vertex without an incoming arc or without an outgoing arc, as if every vertex does contain an incoming arc and an outgoing arc then there must exist a cycle in the tournament, however we have already shown in 3.3.1 that transitive tournaments cannot have a cycle.

Let u be the vertex with an outgoing arc to every other vertex in the graph. Thus u is 'greater' than all other vertices. Now, we remove a vertex u to get a tournament G'. It follows that G' is also transitive in nature. We continue this procedure ,and we observe that by induction hypothesis, G' has a total ordering of its vertices, as we are capable of obtaining vertices at every stage that are 'greater' than all other vertices.

3.3.3. Transitive Tournaments have a unique Hamiltonian path

Let us consider a transitive tournament T. We assume that the Hamiltonian path is not unique in T, and we have two paths H_1 and H_2 such that: $H_1 = v_1, v_2, ...v_n$, $H_2 = v_{\pi^{-1}(1)}, v_{\pi^{-1}(2)}, ...v_{\pi^{-1}(n)}$, where π is a non identity permutation.

Since the tournament is transitive, there is an arc from v_i to v_j whenever $i > j$. As π is a non-identity permutation, there must be some i, j such that $i > j$, but $\pi(i) < \pi(j)$. This leads to a contradiction because there cannot exist an arc from j to i, as this would lead to a cyclic graph. Thus, the Hamiltonian path H_1 must be a unique Hamiltonian path. It also follows that the Hamiltonian path is the strict ordering of vertices.

4. ALGORITHMS

In this section, we will tackle three famous graph theoretical problems restricted to tournaments, namely Hamiltonian Paths, Feedback Vertex Set, and Dominating Set. In Hamiltonian Path problem, we are required to find a path that covers all vertices in the graph exactly once. This problem is NP-hard in generic graphs, but due to the tournaments having an

extra underlying structure, we are able to solve it in polynomial time for tournaments. This algorithm was first published in [7].

In the case of Feedback Vertex Set problem, we need to find a set of vertices of size at most k, the removal of which results in an acyclic tournament. This problem is NP-hard even for tournaments, unlike Hamiltonian Path. So, we will take a step back, and resort to an algorithm for the Feedback Vertex Set in Tournaments problem whose running time is of the form $O(f(k).n^c)$, where c is a constant and f is some computable function. The algorithm is presented in [14]. Such algorithms are called parameterized algorithms and are recently gaining increasing attention. In this case, the problem is parameterized by the solution size.

Finally, for the Dominating Set problem, we are required to find a set of vertices of minimum size which dominates the rest of the tournament. Even though this problem is NP-hard, when restricted to tournaments we do not whether this is NP-hard or has a polynomial time solution. But, although we do not know of a polynomial time algorithm, we do know there exists a sub-exponential algorithm for it, which will be our focus. This algorithm is a consequence of the results presented in [15].

4.1. Hamiltonian Path

4.1.1. Algorithm

We know that finding the Hamiltonian path in a directed graph is an NP Hard problem. However in the case of tournaments this is not true, due to the underlying structure which involves every node to be connected to every other node by a directional edge. We have also shown in the properties section that every tournament has at least one Hamiltonian path. Here we provide a polynomial time algorithm for finding one Hamiltonian path in a tournament.

Algorithm 1 Finding a Hamilton Path

```
V = Vertex set of graph
HP = φ
for all v_new in V do
    Let current Hamilton Path HP = v_1, v_2, ...v_n
    start = 1
    end = n
    while Position for v_new not found do
        if start − end ≤ 1 then          ▷ Ending conditions
            Insert v_new in correct position in HP
            break
        mid = (start + end)/2            ▷ Binary Search
        if v_new ← v_mid then
            start = mid + 1
        else
            end = mid − 1
```

Essentially in this algorithm we perform binary search to find a location to insert the new vertex to increase the path length by one. As shown in Algorithm 1, we compare the edge structure between the middle most element in the current sequence of vertices and the new vertex. If there is an edge from v_{new} to v_{mid}, then we search for a position for v_{new} in the left half of the current vertex list. Otherwise we compare the edge between v_{new} and v_{mid+1}. If it is an outgoing edge from v_{new}, then we stop here and insert v_{new}, else we continue searching in the right half of the vertex list.

The correctness for the algorithm directly follows from the proof of Redei's theorem (3.1). That is, for every path of vertices v_1 to v_k we will always be able to add a new vertex from the tournament to this sequence of vertices such that we get a new path of length $k+1$. This holds for when we divide the path and search a single half of the path for a position to insert. Hence the correctness of the algorithm. We inductively do this until we have finally created a path comprising of all the nodes, which ultimately is our Hamiltonian path.

Time Complexity. As we perform binary search for every k sized path length from 1 to n where n is the total number of vertices, our algorithm runs in $\Sigma_{k=1}^{n} O(\log k) = O(n \log n)$ time.

We observe that we cannot do better than this time complexity. For the intuition behind this, let's consider a transitive tournament, where there exists a unique Hamiltonian path and a strict ordering of vertices. Since both are equivalent, finding a Hamiltonian path in transitive tournaments is same as the problem of comparison based sorting. Hence, we cannot do better than $O(n \log n)$, as otherwise it would imply there is also a better algorithm for comparison based sorting than $O(n \log n)$, which we know is not possible.

4.1.2. Sensitivity study of edge reversal on Hamiltonian Path

We show the following properties on the reversal of an edge in a Hamiltonian path of a tournament:

- A Hamilton path of the new graph can be found using the older hamilton path in $O(n)$ time.
- The subparts of the older Hamilton path divided by the reversed edge maintain the same individual ordering in the new hamilton path.

We demonstrate an algorithm to find the new Hamiltonian path. Let the ordering of vertices $v_1, v_2, \ldots v_n$ be a Hamiltonian path in a tournament of n vertices. Let us consider that the edge between v_r and v_{r+1} is reversed. Now we have two portions of the older Hamiltonian path :

$P_1 : v_1, v_2, \ldots v_r$

$P_2 : v_{r+1}, v_{r+2}, ...v_n$

Now we iterate with two pointers which start from the ends of P_1, P_2 individually. First, we move from v_n backwards in P_2 looking for a position to insert v_r from P_1. Once we find this location, we insert v_r and then continue the procedure for v_{r-1}. Observe that since we know there exists an

edge $v_{r-1} \rightarrow v_r$ we can continue from there to insert v_{r-1} without any incompatibility issues (similar to the ideology behind the binary search conditioning for finding the Hamiltonian path). We continue until all nodes from P_1 are inserted into P_2, and finally P_2 becomes our new Hamiltonian Path. Every node is iterated through once and hence the algorithm runs in $O(n)$.

We know that P_1 and P_2 originally maintain different portions of the older Hamiltonian path. While generating the newer Hamiltonian path we do not disturb the ordering of P_2's vertices, and we continually insert vertices from P_1 from v_r to v_1. Thus we also maintain the ordering of vertices from P_1. Hence we have shown that the final Hamiltonian path consists of all the vertices from P_1 and P_2 maintaining their individual ordering.

4.2. Feedback Vertex Set of size k

As seen previously, in this problem we are give an tournament T and an integer k, and we are required to find $X \subseteq V(T)$, $|X| \leq k$ such that $T - X$ is acyclic or determine that no such X exists. In this algorithm, we will use a powerful algorithmic technique often used for parameterized algorithms called Iterative Compression, first introduced by Reed et. al [16].

Algorithm 2 Iterative Compression

```
1:  function ITERATIVE-COMPRESS(T)
2:      V' ← φ
3:      X ← φ
4:      for all v ∈ V(T) do
5:          V' ← V' ∪ {v}
6:          X ← X ∪ {v}
7:          X ← COMPRESS (T[V'], X)
8:          if |X| > k then
9:              return NO
10:     return X
```

In Algorithm 2, the general idea for the iterative compression algorithm that uses the Compress subroutine is given. The Compress routine takes a tournament T and a Feedback Vertex Set (FVS) X of T of size $k+1$ as input and either outputs a FVS of size at most k or conclude that no such FVS exists. Now, we take an arbitrary ordering $(v_1, v_2, ..., v_n)$ of $V(T)$ and we define $V_i = \{v_1, v_2, ..., v_i\}$ and $T_i = T[V_i]$. We note that V_k is a FVS of T_k, and also if X is a FVS of T_i, then $X \cup \{v_{i+1}\}$ is a FVS of T_{i+1}. From this fact, and the help of the Compress routine, we always try to maintain a FVS (X in the algorithm) of size at most k. If for any T_i, $|X| > k$ we stop the algorithm and conclude that a FVS of size at most k doesn't exist for the tournament T, because there is no such FVS even for T_i which is an induced subtournament of T.

Compress Routine. The main aim of this routine is to try to find a FVS whose size is smaller than the given FVS of the tournament. This routine solves this problem by reducing it to a bounded number of instances of the Disjoint-Compress problem. In this problem, you are given a tournament T, a FVS S of T, and we are required to find a FVS X disjoint from S ($X \cap S = \phi$) such that $|X| < |S|$. The basic idea is depicted in Algorithm 3. Assume there exists a FVS Y of T such that $|Y| < |X|$. Now we guess the intersection of X with Y, that is, we guess the the set $Z = X \cap Y$. Let $S = X - Z$. If our guess is correct, Z will be present in the final solution and hence it is alright if we just delete these vertices from T. Now we should try to find $S' = Y - Z$. But, we know that $S' \cap S = \phi$. Thus, it can be seen that we now have an instance of the Disjoint-Compress problem with the input tournament being $T - Z$ and its FVS being S, and need to find a FVS S' such that $S' \cap S = \phi$ and $|S'| < |S|$. If such a set exists, then we can happily return $S' \cup Z$ as the required output. Else, if no such S' exists for all possible guesses of Z, then we conclude that we cannot compress the given problem instance. Finally, note that, since $|X| \leq k$, the number of guesses of Z is at most 2^k.

Algorithm 3 Compress Routine

```
1:  function COMPRESS(T, X)
2:      for all S ⊆ X do
3:          Z ← X − S
4:          if T[S] is transitive then
5:              T' ← T[V(T) − Z]
6:              S' ← DISJOINT-COMPRESS (T', S)
7:              if |S'| < |S| then
8:                  return S' ∪ Z
9:      return X
```

Disjoint-Compress Routine. To solve this problem, we need to use two properties of acyclic tournaments:

i. A tournament T is acyclic if and only if it doesn't contain any triangles.

ii. If a tournament T is ayclic, then there exists a unique ordering of $V(T)$, such that for every directed edge (u, v), u appears before v in the ordering.

Now, let the input tournament be T and its FVS be S. Let the optimal FVS disjoint from S be X. Define, $A = V(T) - S$. By i), as long as we can ensure there are no triangles in T, we are done. Fig. 6 shows the different triangles that are possible. But, since we cannot remove any vertex from S, we can conclude that triangle **a)** cannot occur. The same goes for triangle **d)** as S is a FVS of T. Thus, we can say that $T[S]$ and $T[A]$ are acyclic tournaments. Moreover, triangle **c)** can only be removed by adding the vertex that lies in A to the FVS X, as the other two vertices cannot be added. Thus, all remaining triangles are only of type **b)**.

Let the unique topological ordering for $T[S]$ and $T[A]$ be $\sigma = (s_1, s_2, ..., s_q)$ and ρ, respectively. We also know that $T[S \cup (A - X)]$ is acyclic, and its unique topological ordering

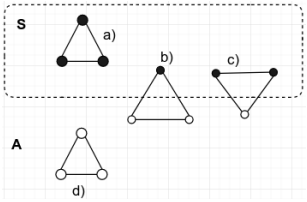

Fig. 6. Different types of triangles

is such that when we restrict it to S and $A - X$, we get σ and ρ, respectively. So, we just have to find the maximum number of vertices from A we can insert in σ.

Consider the tournament $T[S \cup \{v\}]$, where $v \in A$. It will be transitive since there is no triangles of type **c)** (Fig. 6) in T. The unique topological ordering of this tournament will be formed by inserting v in σ at position, say, $p[v]$. Moreover, $p[v]$, is defined for all $v \in A$. Now, we construct a new ordering of A, π. In π, u occurs before v iff $p[u] < p[v]$, or $p[u] = p[v]$ and u occurs before v in ρ.

Now, the main observation is that, in $T - X$, the topological ordering of $T[A - X]$ needs to be a restriction of π, because S is still present in $T - X$, and also needs to be a restriction of ρ as that is the original topological ordering of $T[A]$. Combining these two facts, we can see that the maximum number of vertices from A that can be inserted in σ is the size of the Longest Common Subsequence (LCS) of π and ρ, which can be found in $O(n^2)$ (it can even be done in $O(n \log n)$ as the two sequences are permutations). Define, Z as the set of vertices in the LCS of π and ρ. It is obvious that the optimal disjoint FVS for T is $X = A - Z$. If $|X| < |S|$, we can return this FVS, else conclude that no disjoint FVS of smaller size exists. With this we conclude our algorithm.

Time Complexity. In Disjoint Compress, detecting triangles of type **c)** can be done in $O(k^2 n \log n)$, creating the ordering π needs $O(kn)$ time and LCS of π and ρ can be found in $O(n \log n)$. Thus, Disjoint Compress takes $O(k^2 n \log n)$ time. Merging the complexities of Compress and Iterative-Compress, we get the promised running time of $O(k^2 2^k n^2 \log n)$, which is of the form $O(f(k).n^c)$.

4.3. Dominating Set

In this problem, we are given a tournament T, and we need to find $X \subseteq V(T)$ of minimum size such that for every $v \in V(T) - X$, $\exists v_0 \in X$ such that the directed edge $(v_0, v) \in E(T)$. We call such a X, the minimum dominating set of T. The algorithm mainly hinges on the following claim.

Claim. For every tournament T on n vertices, there exists a dominating set of size at most $1 + \log n$ vertices.

Proof. We will prove this by induction on n. For $n = 1$, it is trivial. Hence, we will assume $n \geq 2$. We know that the sum of out-degrees of the vertices in T is $\binom{n}{2} = \frac{n(n-1)}{2}$. This means that, $\exists v_0$ such that its out degree is at least $\frac{n-1}{2}$. Let U^+ and U^- be the set of out-neighbours and in-neighbours of v_0, respectively. We observe that $|U^-| \leq \frac{n-1}{2} \leq \left\lfloor \frac{n}{2} \right\rfloor$. As $\left\lfloor \frac{n}{2} \right\rfloor \leq n$, by induction hypothesis, there exists a dominating set X' of size at most $1 + \log \left\lfloor \frac{n}{2} \right\rfloor \leq \log n$ for $T[U^-]$. Therefore, it can be seen that $X = X \cup \{v_0\}$ is a dominating set for T of size at most $1 + \log n$. This concludes the proof. □

With this claim, we now have an easy algorithm. We start from $i = 1$ and increase it by 1 in every step. For every i, we enumerate all $X \subseteq V(T), |X| = i$. This is done until we find a dominating set. From the above claim, we are guaranteed to find a dominating set for $i \leq 1 + \log n$. This simple brute force algorithm is, surprisingly, the best algorithm we have for dominating set in tournaments.

Time Complexity. In the worst case, we enumerate $\sum_{i=1}^{1+\log n} \binom{n}{i} = O(n^{(1+\log n)})$ sets. And for every set, the verification takes $O(n^2)$ time. So, overall, we have a run time of $O(n^{(3+\log n)}) = O(n^{O(\log n)}) = O(2^{O(\log^2 n)})$. Thus, we have a sub-exponential algorithm.

Size	Avg.	Min.	Max.	Limit
5	1.656	1	2	3
8	1.953	1	2	4
12	2.009	1	3	4
16	2.148	2	3	5
20	2.385	2	3	5
23	2.604	2	3	5
27	2.802	2	3	5
32	2.929	2	3	6

Table 1. Average, Minimum and Maximum size of Minimum Dominating Set in randomly generated tournaments

Though, we have said that there always exists a dominating set of size $1 + \log n$ for T_n, practically, for a random tournament, the size of the minimum dominating set almost never reaches this theoretical limit, as demonstrated by Table 1. This table is generated by sampling $\min(2^n, 1000)$ random tournaments of size n, and calculating the size of their minimum dominating set. We can also see that the average size is always much lesser than the upper bound we have established earlier, which leads us to believe that in most of the cases, the size of minimum dominating set is quite small, thereby leading to an increased performance of our algorithm in practical scenarios.

5. CONCLUSION

In this term paper, we have observed a special type of directed graphs known as tournaments. We have gone through many of the major properties and theorems regarding tournaments and in the process highlighted some of the most popular problems in the domain of tournaments. We have also further gone on to discuss some of the most efficient solutions to these problems, which in general for directed graphs are much more

difficult. But, we are capable of alleviating a lot of the complexities due to the properties observed in tournaments, and thus we have more efficient solutions than the ones we have for a generic directed graph. With that, we conclude the paper by encouraging the reader to explore more interesting problems and properties in the domain of tournaments.

6. REFERENCES

[1] H.G. Landau, "On dominance relations and the structure of animal societies: The condition for a score structure," *Bulletin of Mathematical Biophysics*, vol. 15, pp. 143–148, 1953.

[2] Kenneth J. Winston and Daniel J. Kleitman, "On the asymptotic number of tournament score sequences," *Journal of Combinational Theory, Series A*, vol. 35, pp. 208–230, 1983.

[3] P. Camion, "Chemins et circuits hamiltoniens des graphs complets," *C.R. Acad. Sci. Paris (A)*, vol. 249, pp. 2151–2152, 1959.

[4] D. Soroker, "Fast parallel algorithms for finding hamiltonian paths and cycles in a tournament," *Journal of Algorithms*, vol. 9, pp. 276–286, 1988.

[5] Y. Manoussakis, "A linear-time algorithm for finding hamiltonian cycles in tournaments," *Discrete Applied Mathematics*, vol. 36, pp. 199 – 201, 1992.

[6] C. Morrow and S. Goodman, "An efficient algorithm for finding a longest cycle in a tournament," in *7th Southeastern Conference on Combinatorics, Graph Theory and Computing*. Utilitas Math., 1976, pp. 453–462.

[7] Amotz Bar-Noy and Joseph Naor, "Sorting, minimal feedback sets and hamilton paths in tournaments," *SIAM Journal on Discrete Mathematics*, vol. 3, pp. 7–20, 1990.

[8] C. Thomassen, "Hamiltonian-connected tournaments," *Journal of Combinational Theory, Series B*, vol. 28, pp. 142–163, 1980.

[9] Y. Guo, Andreas Holtkamp, and Sebastian Milz, "Connectivity of local tournaments," *Australian Journal of Combinatorics*, vol. 57, pp. 271–279, 2013.

[10] P. Erdos, "On a problem in graph theory," *The Mathematical Gazette*, vol. 47, pp. 220–223, 1963.

[11] E. Szekeres and G. Szekeres, "On a problem of schütte and erdős," *The Mathematical Gazette*, vol. 49, pp. 290–293, 1965.

[12] R. L. Graham and J. H. Spencer, "A constructive solution to a tournament problem," *Canadian Mathematical Bulletin*, vol. 14, pp. 45–48, 1971.

[13] L. Redei, "Die neue theorie der endlichen abelschen gruppen und verallge- meinerung des hauptsatzes von hajs," *Acta Math. Acad. Sci. Hung.*, vol. 16, pp. 329–373, 1965.

[14] Marek Cygan et al., *Iterative Compression*, pp. 77–98, Springer International Publishing, 2015.

[15] Marek Cygan et al., *Fixed-parameter intractability*, pp. 421–465, Springer International Publishing, 2015.

[16] Bruce Reed, Kaleigh Smith, and Adrian Vetta, "Finding odd cycle transversals," *Operations Research Letters*, vol. 32, no. 4, pp. 299 – 301, 2004.