Proposition 4 *With probability tending to 1 as n tends to infinity, a random tournament on n nodes has an anchor of size $O(\log n)$.*

XXX Moni wanted an exact statement here giving the constant in the Big Oh XXX and a proof of the statement. I have slogged on this paper too long XXX and I am out of patience to do that. Converting the paper from troff to XXX latex was a very unrewarding activity.

The proof is immediate and is omitted.

5 Open Problems

The problem of proving an upper bound of $\lfloor n/2 \rfloor$ on the size of the minimum anchor in any n node tournament is the foremost open problem arising from this paper. Other questions include analyzing a more relaxed definition of anchors which allows vertices to be distinguished based on degree as well and using ideas such as these to come up with a tournament isomorphism algorithm with a good bound on running time.

References

[1] L. Babai, *Moderately Exponential Bound for Graph Isomorphism*, FCT **3** , 1981.

[2] L. Babai, W.M. Kantor, and E.M. Luks, *Computational Complexity and the Classification of Finite Simple Groups*, Proc. FOCS 1983, 162–171.

[3] L. Babai, *Isomorphism Testing and Symmetry of Graphs*, Combinatorics 79 (M. Deza and I.G. Rosenberg, eds.) Ann. Discr. Math. **8**, 1980, 101–109.

[4] E.M. Luks, *Isomorphism of Graphs of Bounded Valence can be Tested in Polynomial Time*, J. of Comp. Systems Sciences, **25** No. 1, 1982, 42–65.

have G-nodes u_T and v_T be able to distinguish between them. Thus the arc between $(u,v)_1$ and u_T will be oriented into u_T while the arc between $(u,v)_2$ and u_T will be oriented out of u_T. Similarly we will have an arc from $(u,v)_1$ to v_T and an arc from v_T to $(u,v)_2$. Finally it remains to specify the arcs to the G'-nodes. The purpose of these nodes is to distinguish every pair of nodes besides the twin-nodes. They correspond to "garbage collection units" in typical NP-Completeness constructions. We know that a vertex cover in G' must pick one of x_i or y_i for each i. We would like to claim that if a set of nodes S forms a minimum vertex cover in G' the corresponding set of nodes in T will form an anchor. We will use the G'-nodes of T to ensure that if one of x_{i_T} and y_{i_T} is picked for each i, then we would have distinguished between all pairs of nodes in T except for the twin-nodes.

To achieve this, we create sufficiently many partitions of the nodes of T into two halves in order that every pair of nodes aside from the twin-nodes occur in different halves of some partition. These partitions will also ensure that twin-nodes are never put in different halves. Since the number of nodes in T is $O(n^2)$ we need no more than about $2\log n$ partitions. To be safe we will assume that we have fewer than $4\log n$ partitions. Let P_1 be the first partition and let A_1 and B_1 be the two halves of this partition. We put in the arcs to x_{1_T} and y_{1_T} on the basis of P_1. All nodes in A_1 will have arcs into x_{1_T} and y_{1_T}. All nodes in B_1 will have arcs out of these nodes. In general we handle partition P_i by using nodes x_{i_T} and y_{i_T}. There appears to be a small problem here since the prescription above might call for an arc from a node to itself. However this is not really a problem because nodes in the anchor do not have to be distinguished from any other node.

We will show that the instance of vertex cover with G' and k' is a YES-instance if and only if the instance of anchor with T (constructed as above) and k' is a YES-instance.

If the instance of vertex cover is a YES-instance let S be a set of nodes in G' which forms a vertex cover such that $|S| < k'$. We claim that the corresponding nodes in T form an anchor for T. S must contain one of x_i and y_i for each i. Our construction ensures that the corresponding nodes in T distinguish between every pair of nodes except for the twin-nodes. But since S is a vertex cover in G' it contains at least one of the end points of each edge in G'. Since twin-nodes are distinguished at their "end points", the corresponding set of nodes also distinguishes between twin-nodes in T. Hence it is an anchor.

Now suppose that T has an anchor of cardinality less than k'. We will prove that G' has a vertex cover of cardinality less than k'. A pair of twin-nodes $(u,v)_1$ and $(u,v)_2$ are distinguished by a set iff one of the four nodes: $(u,v)_1, (u,v)_2, u$, or v is in the set. So the anchor must contain one of these nodes. The choice of $(u,v)_1$ or $(u,v)_2$ does not help in distinguishing between other pairs of twin-nodes. Hence, if we assumed that the only nodes to distinguish between were twin-nodes then we can assume without loss of generality that one of u or v was picked. Since an anchor must distinguish between all pairs of nodes in T a lower bound on the size of the anchor is the minimum cardinality of a set that distinguishes between twin-nodes. It is clear that any such set must contain a vertex cover in G'. This completes the reduction proving the NP-Completeness of the anchor problem. ∎

Definition 3 *Define a random tournament as one where the directions of the arc are chosen independently with a probability of half for either direction.*

Proof: Consider the regular rotational tournament on $2n+1$ nodes defined as: The arc between node i and node j goes from i to j iff $j-i \pmod{2}n+1 < i-j \pmod{2}n+1$. Since $2n+1$ is odd this defines a tournament. To distinguish between i and $i+1$ we need to pick one of $i, i+1, n+i+1 \pmod{2}n+1$. In fact each node distinguishes at most three pairs of consecutive vertices. Since there are n such pairs we need at least $n/3$ nodes in the anchor. ▮

We make the following conjecture about anchor sizes based only on empirical evidence.

Conjecture 1 *Every n node tournament has an anchor of size at most $\lfloor \frac{n}{2} \rfloor$.*

4.2 NP-Hardness of Finding Minimum Anchors

Theorem 3 *Given a tournament, T, and an integer l it is NP-Complete to determine if T has an anchor of size less than l.*

Proof: The reduction is from vertex cover. Let $G=(V,E)$ and k be an instance of vertex cover. We make a preliminary transformation to a new instance of vertex cover as follows. If G has n nodes we create a graph G' which is obtained by adjoining $4\log n$ pairs of vertices, x_i, y_i, $i = 1, 2, \ldots, 4\log n$ to G. Each x_i is connected to the corresponding y_i by an edge. Thus we add $4\log n$ disjoint components each of which is a copy of K_2. Of course, k must be suitably modified. So we set $k' = k + 4\log n$ since we know that any minimum vertex cover will pick exactly one vertex out of each new component that was added. Now we transform this new instance of vertex cover into an instance of the anchor problem.

We create a tournament T from G' as follows: For each node $v_{G'} \in G'$ in G' we create a corresponding node v_T in T. For each edge (u,v) in G' we create two nodes $(u,v)_1$ and $(u,v)_2$ in T. The arcs of the tournament will be so chosen as to make the two nodes $(u,v)_1$ and $(u,v)_2$ distinguishable only at u_T and v_T.

We now specify the arcs in T. Choose an arbitrary order for the nodes in G' where the nodes in G precede the nodes added to create G'. Also choose an arbitrary order on the edges of G'. To facilitate the description of the arcs of T we divide the nodes in T into three classes — namely

1. Nodes in T corresponding to edges in G'. These will be called the *edge-nodes*.

2. Nodes in T corresponding to the nodes in G. These will be called the *G-nodes*.

3. Nodes in T corresponding to the nodes in $G' - G$. These will be called the *G'-nodes*.

Two edge-nodes which correspond to the same edge in G' will be called *twin-nodes*.

We choose an arbitrary order on the edge-nodes such that twin-nodes occur consecutively. Similarly we choose arbitrary orders on G-nodes and on G'-nodes. Arcs between two nodes within the same class will be transitively oriented according to the order chosen for that class. Arcs between edge-nodes and G-nodes will be oriented from the edge-nodes to the G-nodes with the following exception. For twin-nodes $(u,v)_1$ and $(u,v)_2$ we will arrange to

Proof: By lemma 2 we know that if $w, z \in S$ are both of type u then the arcs between w and u and between z and u are oppositely oriented. Hence w and z are distinguished by u. ▮

At this point we know that the only kinds of nodes in S that could resemble each other over $V - S$ are nodes of different types.

Lemma 7 *No three nodes in S can resemble each other over $V - S$.*

Proof: Let if possible three nodes in S resemble each other over $V - S$. Let v_1 and v_2 be two such nodes. We know that v_1 and v_2 are of different types. Let v_1 be of type u_1 and v_2 be of type u_2. Without loss of generality assume that we have an arc from v_1 to u_1. Since v_1 and v_2 resemble each other over $V - S$ there must be an arc from v_2 to u_1. Since v_1 is of type u_1 there is also an arc from v_2 to v_1. Since v_2 is of type u_2 there is an arc from u_2 to v_1. Finally, since v_1 and v_2 resemble each other over $V - S$ there is an arc from u_2 to v_2. This proves that if v_1 and v_2 resemble each other the arcs to their type nodes are oppositely oriented. Hence, at most two nodes in S can resemble each other over $V - S$. ▮

Now we are ready to complete the proof of the upper bound.

Theorem 2 *In any n-node tournament there is an anchor of size at most $2n/3$.*

Proof: Construct an arbitrary minimal anchor S and suppose that $|S| > 2n/3$. We will create a new anchor S' as follows. S' will consist of all the nodes of $V - S$ and one node out of every pair of nodes in S which resemble each other over $V - S$. By the previous lemmas the size of S' is no more than $|V - S| + |S|/2$. If $|S| = k$, $|S'| \leq n - k + k/2$. $|S'| \leq n - k/2$. Since $k > 2n/3$ by assumption $|S'| \leq 2n/3$. ▮

In fact our proof of theorem 2 gives us an efficient algorithm for constructing such an anchor.

The next proposition shows that the upper bound in Theorem 2 is not too far off. There are tournaments with no anchors of smaller size than $\Omega(n)$. This means that the notion of anchor cannot be used directly to solve tournament isomorphism for all tournaments.

Proposition 2 *There exists a tournament on n nodes where the size of the minimum anchor is $\lfloor n/2 \rfloor$.*

Proof: let T be the transitive tournament on n nodes, i.e. a tournament where the vertices are ordered from 1 to n and all the arcs are directed from the lower numbered vertices to the higher numbered ones. It follows from the definition of an anchor that no two consecutive vertices can be left out of any anchor. This proves that any anchor must include at least $\lfloor n/2 \rfloor$ vertices. ▮

A slightly more interesting fact is that even regular tournaments can have large anchors.

Proposition 3 *There are regular n-node tournaments where the minimum anchor is of size $n/3$.*

Proof: With each pair-node, u, in S we can asssociate a pair of nodes in $V-S$ such that u is the unique node in S distinguishing this pair. If no such pair exists u would be redundant and the anchor would not be minimal. These associations naturally induce a graph on the nodes in $V-S$, with edges going between every pair associated with a pair-node. We claim that this graph is acyclic. Suppose to the contrary that u_1, u_2, $\ldots$, u_l is a cycle in the graph. Let $v_1, v_2, \ldots, v_l$ be the pair-nodes in S such that v_i is the unique node distinguishing u_i from u_{i+1} and v_l is the unique node distinguishing u_l from u_1. At v_{l-1}, u_{l-1} resembles u_{l-2} for otherwise v_{l-2} would not be a pair-node. Similarly u_{l-2} resembles u_{l-3} since otherwise v_{l-3} would not be a pair-node. Continuing thus and using transitivity we find that u_1 resembles u_{l-1} at v_{l-1}. Since v_{l-1} distinguishes u_{l-1} from u_l it follows that it distinguishes u_1 from u_l and thus v_l is not a pair-node. Thus we have a contradiction proving our claim. ▮

The two previous lemmas immediately imply the following lemma:

Lemma 4 *Every minimal anchor in an n-node tournament is of size $< 3n/4$.*

Proof:

Let $|V-S| = k$. Then there are at most $2k$ type-nodes and at most $k-1$ pair-nodes. Thus the anchor is of size less than $3k$ and this implies that the size of *every* minimal anchor in an n-node tournament is less than $3n/4$. ▮

Remark: The largest minimal anchor we have been able to find in an n-node tournament is of size $2n/3$. The tournament is the transitive tournament (defined later in this paper). If the nodes of the tournament are numbered $1 \,..\, n$ in transitive order, the minimal anchor consists of all nodes whose numbers are 1 or 0 mod 3.

We now show that there exists an efficiently constructible minimal anchor of size at most $2n/3$.

The construction works as follows: We first construct an arbitrary minimal anchor S. If $|S| \leq 2n/3$ we are done. Therefore we will assume that $|S| > 2n/3$. In this case, we construct a superset of $V-S$ which will be an anchor of size at most $2n/3$.

The following lemmas establish how far $V-S$ deviates from being an anchor.

Lemma 5 *Every pair-node in S is distinguished from every other node in S by some node in $V-S$.*

Proof: Each pair-node $w \in S$ is the unique node distinguishing a pair of nodes, $u, v \in V-S$. Thus the arcs between w and u and between w and v are oppositely oriented, while for every other node $z \in S$ these arcs are similarly oriented. Hence, $\forall z \in S$ w is distinguished from z at one of u or v. ▮

Lemma 6 *Two nodes of the same type are distinguished from each other at some node in $V-S$.*

Proof: If $x_1, x_2, \ldots, x_k$ is the list of k nodes that form an anchor we can associate a k-bit vector with each node outside the anchor. For node u this k-bit vector has 1 in the i^{th} place if the arc between u and x_i is directed into x_i and 0 otherwise. By definition of anchors these k-bit vectors are distinct for distinct nodes outside the anchor. Hence, there must be at least $n-k$ k-bit vectors. This implies the required bound. ∎

4.1 An Upper Bound on Anchor Size

We will prove in this subsection that every tournament on n nodes has an anchor of size at most $2n/3$.

We make the following definitions. Two nodes u and v are said to *resemble* each other at node w if (u, w) is an arc iff (v, w) is an arc. If two nodes do not resemble each other at w we say that they are *distinguished* at w. Two nodes are distinguished by a set A if $\exists w \in A$ which distinguishes them.

Let $T = (V, E)$ be a tournament on n nodes and let S be a minimal anchor in T. Since S is minimal whenever we consider $S - v$ for any $v \in S$ one of the following two things must happen:

1. Two nodes $u_1, u_2 \in V - S$ resemble each other at every node in $S - v$. In this case v was crucial in distinguishing between u_1 and u_2. Such a node v will be designated a *pair-node*.

2. v resembles $u_1 \in V - S$ at every node in $S - v$. In this case removing v from S makes u_1 and v indistinguishable. Such a node v will be designated a *type-node* of *type* u_1.

Note that it is possible for a node to be both a type-node and a pair-node.

We prove the main theorem by proving various facts about type-nodes and pair-nodes.

Lemma 2 *There are at most two type-nodes of any one type in S.*

Proof: Let u be a node in $V - S$ and suppose there are more than 2 nodes of type u in S. Let v_1 and v_2 be two such nodes. Suppose, without loss of generality that the arc between u and v_1 is directed from u to v_1. Then because v_2 is of type u we must have an arc directed from v_2 to v_1. Now, since v_1 is also of type u we must have an arc directed from v_2 to u. Thus we have proved that if two nodes are of type u they must have oppositely oriented arcs between u and themselves. This proves also that there can be at most two nodes of type u. ∎

Lemma 3 *For any minimal anchor S, if there are r nodes in $V - S$ there are at most $r - 1$ pair-nodes in S.*

guarantees that at the end we will have the required size bound on the label classes; all we have to do is bound the number of nodes that we pick.

We represent the selection process above by a tree, H. H represents how the label classes are split as more and more nodes of T_1 are selected. The internal nodes of H correspond to label classes of T_1 at various stages of refinement. Thus for example, the root of H represents all the nodes of T_1 since all nodes are in one label class initially. The internal nodes of H also stand for the node of T_1 that is selected at a particular stage. Thus if the first node selected by our selection process is v in T_1, then the root of H also represents the node v. The children of the root in H are the label classes that result from the creation of a singleton class for v and stabilization. (This tree does not capture the full extent of stabilization. It does not represent splits in label classes resulting from selection of a node in another label class. However, it will do for our purpose.) Finally, the leaves of H correspond to label classes whose size is bounded by $2n/k$.

In order to bound the number of nodes selected, we can bound the number of internal nodes of H. We will use the fact that if some internal node a represents a label class of size l, each of its children represents a label class of size at most $l/2$. We adopt the following system of accounting. Each internal node v of H corresponds to a set of nodes S of the tournament. If $x \in S$ we charge x an amount of $1/|S|$ towards v. Thus the total charge towards v is $|S| * 1/|S|$ which is 1. Thus to count the total number of internal nodes we can add up all the charges. For each node x in the tournament its charge towards its lowest level internal node is $< k/2n$ (since each internal node corresponds to a label class of size greater than $2n/k$). Subsequent charges on x drop at least by a factor of two for each step up in the tree. Thus the total charge accruing to x is at most k/n and thus the total over all nodes in the tournament of the charges is at most k. This establishes that the number of internal nodes is at most k. ∎

The following theorem now follows easily.

Theorem 1 *The above algorithm works in* $O(n^{4\sqrt{n}+O(1)})$ *time.*

Proof: For the case where the label classes are of size bounded by l the running time of the deterministic algorithm in [3] is $O((nl!)^4)$. There is also a Las Vegas version in [3] which runs in time $O((nl!)^2)$. For the purpose of this analysis we assume we are using the Las Vegas algorithm. With appropriate bounds for the factorial, the overall running time of the algorithm is in $O(n^{k+2}l^{2l})$. We choose k so as to minimize this running time. The choice of $k = 2\sqrt{n}$ gives us the best time bound of $n^{4\sqrt{n}+O(1)}$. ∎

The sequence of nodes found in the course of this algorithm is like an anchor except that we also allow distinctions between nodes based on (in and out) degrees to already distinguished nodes.

4 Results on Anchors

Proposition 1 *Any anchor in a tournament with* n *nodes must have size at least* $\lceil \log n \rceil - 1$.

Intuitively an anchor is a subset of the nodes such that for all pairs of nodes outside it there is some node in it which "distinguishes" the pair. However the notion of 'distinguishing' here is very restrictive — we only consider two nodes v and w distinguished by a node u if $(v, u) \in E \Leftrightarrow (u, w) \in E$.

3 An Algorithm for Tournament Isomorphism

We present a very elementary algorithm for tournament isomorphism which runs in $n^{O(\sqrt{n})}$ time. The algorithm is presented mainly as a way of motivating notions such as anchors.

Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be the two tournaments for which we want to test isomorphism. Let $n = |V_1| = |V_2|$.

The algorithm works by choosing an appropriate sequence S_1 of k vertices (k to be specified later) in T_1 as distinguished vertices and using the asymmetry introduced to partition V_1 into an ordered set of equivalence classes of size no more than $2n/k$. We will let $l = 2n/k$. We will refer to these equivalence classes as *label classes* to emphasize the order on these classes. Any automorphism of T_1 that fixes S_1 pointwise must respect the label classes. If T_1 and T_2 are isomorphic and S_1 and S_2 are chosen so that there is an isomorphism between T_1 and T_2 that maps the sequence S_1 to S_2, then the i^{th} label class in T_1 must map to the i^{th} label class in T_2. In order to find S_2 the algorithm tries all possible sequences of vertices of T_2 of cardinality $|S_1|$.

Once we have a partition into label classes we apply the algorithm due to Babai[3] for isomorphism of graphs where the vertices have been partitioned into label classes.

We now describe the process by which the partition is obtained. This process will be called *stabilization*. Stabilization works as follows: Initially all nodes are in one label class. If there are two nodes, u and v in a label class L such that they have different outdegrees (or indegrees) to some label class (possibly L itself), then we break up L into two or more classes based on this difference. This process must recursively be applied to each of the new label classes we create and in fact to the old label classes as well. We have a stabilized labeling if no further splitting is possible. In this case, if some of the label classes are too large we create a further asymmetry by picking a suitable node to be in a label class by itself and stabilizing again.

Lemma 1 *For any k such that $1 \leq k \leq n$, there are k nodes in a tournament which when picked to form individual classes, split the other nodes into classes no bigger than $2n/k$.*

Proof: We will give a constructive proof of the lemma. By definition of stabilization above, after stabilization each label class is a regular tournament, i.e one where all the nodes have the same indegree and outdegree. Thus selecting any one node v, in a label class to be a singleton class causes that label class to be broken up into classes no one of which is more than half the size of the original class.

We will pick the nodes by the following simple strategy: As long as there are label classes whose size is greater than $2n/k$, pick a node in one such class and stabilize. This strategy

1 Introduction

The complexity of tournament isomorphism was shown to be $n^{O(\log n)}$ by Luks[4]. However, the proof in [4] depends on non-elementary theorems about finite groups such as the fact that groups of odd order are solvable. In this paper we are motivated by the question, 'Are there algorithms that run in time $n^{O(\log n)}$ or better whose running time bounds can be obtained purely combinatorially?'

In a paper on graph isomorphism, Babai[1] describes a technique of Zemlyachenko which attempts to break the 'symmetry' between vertices of a graph by picking a subset, S, of distinguished vertices and partitioning the rest of the vertices based on degree to the distinguished vertices and recursively on degrees to sets in the partition. If we can partition the vertex set into constant sized classes using a set S then we have an $O(n^{|S|})$ algorithm for the isomorphism question since we simply try all possible sets of size upto $|S|$ in both graphs and try to establish an isomorphism based on a 1–1 correspondence between vertices in the two sets. For the case of arbitrary graphs the technique in [1] yields a bound of $\exp(n^{2/3+o(1)})$ for the isomorphism of n-node graphs. (Subsequent papers have improved upon this time bound using other techniques. See for example [2].)

In this paper we show that a very simple algorithm for tournament isomorphism along the lines of [1] has an $n^{O(\sqrt{n})}$ time bound. We also define the notion of an *anchor* in a tournament to be a set of nodes which distinguishes every other pair of nodes at distance 1. This is a major strengthening of the requirements on a distinguishing set of vertices as in [1] and we cannot expect to be able to prove very good upper bounds on the sizes of anchors. However anchors are a clean combinatorial notion and their study seems to be of interest in its own right. An understanding of anchors might lead to appropriate relaxations of the definition which are tractable and useful with respect to algorithms for tournament isomorphism. Our results about anchors in n-node tournaments are the following:

1. Every tournament has an anchor of size at most $2n/3$.
2. There are tournaments that require $n/2$ nodes in any anchor.
3. Every anchor in every tournament must be of size $\Omega(\log n)$.
4. Determining an anchor of minimum size is NP-Complete.

2 Definitions

Definition 1 *A tournament, $T = (V, E)$ is a directed graph such that for every pair of vertices u, v, $(u, v) \in E \Leftrightarrow (v, u) \notin E$.*

Definition 2 *An anchor in a tournament, $T = (V, E)$ is a subset $S \subseteq V$ such that $\forall u, v \in V - S\ \exists w \in S$ such that exactly one of (u, w) and (v, w) is an arc of T.*

Anchors in Tournaments

Sampath Kannan[1] Moni Naor[2] Steven Rudich[3]

TR 92-08

Abstract

We define and study the graph-theoretic notion of an *anchor* in a tournament. An anchor in a tournament is a subset of the nodes of the tournament such that every pair of nodes outside the anchor is 'distinguished' at one of the nodes in the anchor. We show that every n-node tournament has an anchor of size at most $2n/3$ and that there are n-node tournaments where the anchor size is at least $n/2$. We also show that finding a minimum-sized anchor in a tournament is NP-Complete. Finally we point out potential applications of anchors and related concepts to the design of efficient algorithms for tournament isomorphism.

Department of Computer Science
The University of Arizona
Tucson, AZ 85721

[1]University of Arizona
[2]IBM Research Division, Almaden Research Center
[3]Carnegie-Mellon University