Discrete Applied Mathematics 322 (2022) 240–252

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Finding kings in tournaments☆

Arindam Biswas [a], Varunkumar Jayapaul [b,*], Venkatesh Raman [a], Srinivasa Rao Satti [c]

[a] The Institute of Mathematical Sciences, Chennai 600113, India
[b] Indian Institute of Technology Mandi, Kamand 175075, India
[c] Norwegian University of Science and Technology, Norway

ARTICLE INFO

Article history:
Received 4 April 2022
Received in revised form 21 June 2022
Accepted 23 August 2022
Available online 15 September 2022

Keywords:
Tournament
King
d-cover
Dominating set

ABSTRACT

A tournament is an orientation of a complete graph. It is well-known that any tournament has a vertex from which every other vertex can be reached by a path of length at most 2. Such a vertex is called a king or a 2-king. It is also known that to find such a vertex, $\Omega\left(n^{4/3}\right)$ queries (to the adjacency matrix) are necessary and $O\left(n^{3/2}\right)$ probes are sufficient. It is a long standing open problem to narrow this gap between the upper and lower bound.

We first show that

– the adversary Ajtai et al. (2016) and Shen et al. (2003) used to prove the known $\Omega\left(n^{4/3}\right)$ lower bound cannot be used to prove a better lower bound, by giving an algorithm that achieves the bound against the same adversary;

Clearly in any tournament there is a vertex from which every other vertex is reachable by a path of length at most d for any $d \geq 2$ and such a vertex is called a d-king. The bounds for finding a 2-king have been generalized (Ajtai et al., 2016) to obtain generalized upper and lower bounds to find a d-king. We show that

– our algorithm against the weak adversary works against such an adversary for finding d-kings too. More generally, if we can find a 2-king in $O\left(n^{4/3}\right)$ time, then we can find a d-king in asymptotically optimal time for any $d \geq 2$. This was conjectured in an earlier paper.

Then we address the complexity of finding a set of d-kings, i.e. a small subset of vertices such that every vertex is reachable from one of them by a path of length at most d. Such a set is called a d-cover.

– We generalize the lower bound for finding a d-king to give a lower bound for finding k sized d-covers. We complement it with an algorithm matching this bound for $k \in \Omega(\lg n)$. For $d = 1$ for example, our results imply that we can find a $(\lg n - \lg\lg n + k)$-sized dominating set in $O\left(n^2/k\right)$ time and that this bound is optimal.

Finally we develop a dynamic data structure so that whenever a new vertex is added to the tournament, we can find a king of the new tournament in $O(\sqrt{n})$ time.

☆ A preliminary version of this paper appeared in the proceedings of 11th international frontiers in Algorithms workshop (FAW) 2017 (Biswas et al., 2017) [4].

* Corresponding author.
E-mail addresses: barindam@imsc.res.in (A. Biswas), varunkumar@iitmandi.ac.in (V. Jayapaul), vraman@imsc.res.in (V. Raman), srinivasa.r.satti@ntnu.no (S.R. Satti).

https://doi.org/10.1016/j.dam.2022.08.014

1. Introduction and motivation

A *tournament* is a directed graph in which there is exactly one directed edge between every pair of vertices. As they model many practical scenarios (game tournaments, voting strategies), tournaments are well-studied in structural and algorithmic graph theory. Various structural and algorithmic properties of tournaments are known and Moon's [15] early monograph on this subject lead to much subsequent work in the area.

It is well-known that every tournament has a 'center' or a *king* vertex, i.e. a vertex from which every other vertex can be reached by a directed path of length at most 2 (a maximum outdegree vertex of the tournament is one such vertex). A king is also called a 2-king.

There are two different papers written several years apart with almost identical upper bound and lower bound strategies for finding a king [1,18]. Here basically one counts the number of probes (or edge-queries) to the adjacency matrix of the tournament. Both these papers show an upper bound of $O(n\sqrt{n})$ time and a lower bound of $\Omega(n^{4/3})$ time to find a king in a tournament. Closing the gap between the upper and lower bound has been mentioned as an interesting open problem. We address this problem in the paper. While we do not quite close the gap, we make several advances.

Our first result is that to improve the lower bound, we need a different adversary than used in these papers. The lower bound adversary essentially answers an edge query by favoring the vertex (i.e. increasing its outdegree) with lower outdegree till that time. We give an $O(n^{4/3})$ algorithm against this adversary. As the adversary can answer arbitrarily when the outdegree between the two vertices are the same, we need to play against this adversary carefully to prove our upper bound.

Ajtai et al. [1] generalized the notion of a king to a d-king (for any $d \geq 2$). A d-king is a vertex from which every vertex is reachable by a path of length at most d. Ajtai et al. showed that $\Omega(n^{1+1/(2^d-1)})$ probes are necessary and $O(n^{1+1/(3\cdot 2^{d-2}-1)})$ are sufficient to find a d-king in an n vertex tournament. Our algorithm against the adversary for finding a 2-king also works to give an optimal algorithm against the same adversary for finding a d-king. More generally we show that if a 2-king can be found in $O(n^{4/3})$ time, then a d-king can be found in optimal $O(n^{1+1/(2^d-1)})$ time.

Then we address the complexity of finding not just one king, but a small subset of vertices such that every other vertex is reachable by a path of length at most 2 from one of these vertices. More generally, a d-cover is a set of vertices S such that every other vertex in the tournament can be reached from some vertex in S by a directed path of length at most d. A 1-cover is simply a *dominating set*, and d-covers are referred to as *distance-d* dominating sets.

It is well-known that every tournament has a dominating set of size at most $\lceil \lg n \rceil$ and can be found in $O(n^2)$ time. In fact, one can guarantee existence of a slightly smaller dominating set that can also be found in the same time. There exists a dominating set of size $g(n) = \lg n - \lg\lg n + 2$ in a tournament on n vertices [12] (see also Section 2.2). Thus, the minimum dominating set can be found in $n^{O(\lg n)}$ time and hence it is unlikely to be *NP*-complete. Papadimitriou and Yannakakis [16] show that finding the smallest dominating set in a tournament is complete for a complexity class *LOGSNP*. It is also known that there are tournaments where the minimum dominating set size (also called the *domination number*) is $\Omega(\lg n)$ [2,10] (see also [14]) using which it has been shown that finding a k-sized dominating set is complete for the parameterized complexity class $W[2]$ (where the parameter is the solution size) and hence, even an $f(k)n^{O(1)}$ time algorithm is unlikely for any computable function f [9].

We consider the complexity of finding dominating sets of size more than $\lg n - \lg\lg n + 2$ and more generally, the complexity of finding d-covers. Using the lower bound adversary that favors the lower degree vertex, we can show a lower bound for finding k-sized d-covers. We also provide a matching upper bound for finding such d-covers for $k \in \Omega(\lg n)$. More specifically, we show that $\lg n - \lg\lg n + 2 + k$ sized d-covers can be found in $O(k(n/k)^{1+1/(2^d-1)})$ time for any $d \geq 1$ and $k \geq 1$. We also prove a matching adversary based lower bound to show that this bound is optimal for $k > \epsilon \lg n$ for any $\epsilon > 0$. This, in particular, implies that a dominating set of size $k > \lg n$ can be found in optimal $O(n^2/k)$ time.

We also discuss the complexity of finding a special king vertex called a Banks vertex which has applications in social choice theory [6]. A Banks vertex is a source of a transitive subtournament of vertices, who dominate the entire tournament. I.e. let S be a subset of vertices of the tournament such that the induced tournament on S is acyclic, and S is a dominating set of the tournament. Clearly the source vertex of the induced subtournament on S is a king, and that is called a Banks vertex. We observe in this paper that the known $O(n\sqrt{n})$ algorithm to find a king, actually finds a Banks vertex.

In social choice theory, the set of all kings in a tournament is referred to as an *uncovered set*. See [8] for a recent result that shows that finding the set of all kings or Banks points in a tournament requires $\Omega(n^2)$ probes. A similar, but a slightly different kind of result has been known for finding a sorted sequence of kings. A sorted sequence of kings is in a tournament on n vertices is a sequence of vertices $v_1, v_2 \ldots v_n$, such that v_i dominates v_{i+1} and v_i is a king in sub-tournament $T_{v_i} = \{v_i, v_{i+1} \ldots v_{v_n}\}$. It has been known [18] that to find a sorted sequence of kings, $\Theta(n\sqrt{n})$ probes are necessary and sufficient.

Finally we discuss the problem of finding a king or a Banks vertex in an incremental setting. I.e. given a tournament we can maintain some invariants of the tournament so that when a new vertex is added to it, we can find a king or a Banks vertex in $O(\sqrt{n})$ time.

Organization of the paper

In Section 2, we describe the necessary terminology and some basic results on tournaments that we use in the paper. In Section 3, we describe our optimal algorithm against the known lower bound adversary. In Section 4, we look at

d-covers and describe several upper and lower bounds for finding d-kings and d-covers in a tournament. Section 5 shows an adversarial strategy which forces $\Omega(n^2)$ time to verify whether a vertex is a king. In Section 6, we give an incremental (vertices can be added to the tournament, but not deleted) dynamic algorithm to find a king or a Banks point in tournament. We conclude with pointers to some open problems in Section 7.

2. Preliminaries

2.1. Definitions and notation

Let $T = (V, E)$ be a tournament (complete directed graph). A *dominator* in T is a vertex $u \in V$ such that for any other vertex $v \in S$, $(u, v) \in E$. A *king* in T is a vertex $u \in V$ such that for any other vertex $v \in V$ there is a directed path of length at most 2 from u to v. We denote by $V(T)$ the vertex set of T, and by $E(T)$, the edge set of T. $|T|$ denotes the order of T, i.e. the size of $V(T)$, and for a subset $S \subseteq V(T)$, $T[S]$ denotes the subtournament of T induced by S. A tournament or a subtournament is transitive if it has no directed cycle; i.e. there is an ordering of the vertices such that every edge is going from a smaller vertex to a larger vertex.

Let $d \in \mathbb{N}$ and $S \subseteq V$. S is called a *d-cover* of T if for every vertex $v \in V \setminus S$, there is vertex $u \in S$ such that there is a directed path of length at most d from u to v.

A 1-cover is also called a *dominating set*. A king by itself is a 2-cover. If a d-cover is of size 1, we call the unique element in the set a d-king. Let $u \in V$ be a vertex. An out-neighbor of u is a vertex $v \in V$ such that $(u, v) \in E$; u is said to *dominate* v. Similarly, an in-neighbor of u is a vertex $v \in V$ such that $(v, u) \in E$. For any $v \in V(T)$, $N^+(v)$ denotes the set of out-neighbors of v, and $N^-(v)$ denotes the set of its in-neighbors. We also define $\deg^+(v) = |N^+(v)|$, and $\deg^-(v) = |N^-(v)|$.

Given a subset S of vertices, *performing a round robin tournament* on S refers to the process of querying all possibles edges between every pair of vertices in S to know their orientations. Using the same tournament analogy, we sometimes refer to the out-neighbors of a vertex v as vertices that *lose* to v, and to its in-neighbors as vertices that *win* against it. In other words, if there is an edge from u to v, we say that u wins against v, and v loses to u.

We assume that the vertex set of the tournament is $\{1, 2, \ldots, n\}$ and is given in the form of an adjacency matrix where the (i, j)th entry denotes the direction of the edge between i and j. By an edge query or an edge probe, we mean a probe to the adjacency matrix.

To keep the notation simple, we sometimes omit ceilings and floors on fractions when we actually mean integers, typically on the sizes of the vertex subsets we handle. This does not affect the asymptotic analysis. When the term “degree" is used in context of directed graphs we refer to the out-degree of a vertex.

2.2. Some known results we use

In this section, we capture some elementary results about the nature and the complexity of finding kings, that are used later in the paper.

Lemma 1 ([13]). *Any tournament has a king vertex (a vertex from which every other vertex is reachable by a path of length at most 2). In particular, a maximum degree vertex of the tournament is a king.*

Lemma 2 ([13]). *Let T be a tournament and $v \in V(T)$. If a vertex u in $N^-(v)$ is a king in $T[N^-(v)]$, then u is a king in T.*

Proof. The vertex u reaches every vertex in $N^+(v)$ through v. Every other vertex in T is reachable from u by a directed path of length at most 2 as it is a king in $T[N^-(v)]$. □

The above lemma can be used to give a simple algorithm to find a king.

A simple algorithm. Pick any vertex $v \in V$ and find all its out-neighbors in V. Remove v, as well as all its out-neighbors in V from V. Repeat this process till V becomes empty, in which case the last picked vertex is a king. The algorithm takes $O(n^2)$ time.

Lemma 3. *For a tournament of order n a king can be found in $O(n^2)$ time.*

Shen et al. [18] (see also [1]) gave a better algorithm taking $O(n\sqrt{n})$ time which uses the following simple lemma. The lemma follows from the fact that in a tournament, the sum of the outdegrees is $n(n-1)/2$.

Lemma 4 ([12]). *In any tournament T of order n, there is a vertex with outdegree at least $(n-1)/2$ and a vertex with indegree at least $(n-1)/2$ and such vertices can be found in $O(n^2)$ time.*

Lemma 5 ([1,18]). *For a tournament of order n, a king can be found in $O(n\sqrt{n})$ time.*

Proof. The algorithm first picks an arbitrary set of vertices of size $\sqrt{n}$ and performs a round-robin tournament on it. Then, it picks a vertex (say v) of maximum outdegree (which is at least $\sqrt{n}/2$ by Lemma 4) in the set, and finds its out-neighbors in the entire tournament. Next, it removes v and all its outneighbors from V, and repeats the process until V has less than $\sqrt{n}$ vertices in it. At this point, the algorithm uses the routine of Lemma 3 and finds a king in $O(n)$ time. It is easy to see that the king in the final set is a king for the entire tournament by Lemma 2 and that the algorithm takes $O(n\sqrt{n})$ time. □

Banks point

A Banks point is a king vertex with some special properties and has been studied extensively in social choice theory, where they are useful in determining winners in certain types of competition [6].

Definition 1 (*Banks Point, Banks Set [6]*)**.** Let $T = (V, E)$ be a tournament. A vertex $v \in V$ is called a *Banks* point if there is a sequence of vertices $(v_1, v_2, \ldots, v_k = v)$ such that

1. v_i dominates v_j, for all $1 \leq j < i \leq k$, and
2. $\{v_1, \ldots, v_k\}$ is a dominating set for G.

The Banks set for T is the set of all Banks points in T.

From the above definition, it is clear that $S = \{v_1, \ldots, v_k\}$ induces a transitive subtournament of T. Also, $v_k = v$ dominates all other vertices in S. Since S is a dominating set, all vertices in V are reachable from v_k by a directed path of length at most 2. Thus, v_k is a king. However, not every king is a Banks point [17].

Note that the algorithm for finding a king in Lemma 3 actually finds a Banks vertex as the last vertex dominates all the vertices picked up in the previous iterations and together will all of them, it dominates the entire tournament. It is also not hard to see that Lemma 5 also actually finds a Banks point as the sequence of vertices found in the phases of elimination form a transitive subtournament. These vertices which form the transitive tournament are of size $O(\sqrt{n})$. Then in the last subtournament (which is also of size $O(\sqrt{n})$), where every vertex dominates the vertices picked earlier, we apply Lemma 3 to find a king vertex, which actually finds a Banks vertex, which gives us the following lemma.

Lemma 6. *For a tournament of order n, a Banks vertex can be found in* $O(n\sqrt{n})$ *time, and the number of vertices forming the transitive tournament is* $O(\sqrt{n})$.

Dominating sets

The following result is immediate from Lemma 4. We simply find a vertex in the tournament that dominates at least $(n-1)/2$ other vertices, include it in the dominating set and recurse on the in-neighbors of the vertex.

Theorem 1 (*[14]*)**.** *For any tournament of order n, a dominating set of size at most* $\lceil \lg n \rceil$ *can be found in time* $O(n^2)$.

There is also a tighter upper bound which is probably not well-known [12], we give the proof here for completeness.

Theorem 2. *In any tournament* $T = (V, E)$ *of order n, a dominating set of size at most* $\lg n - \lg\lg n + 2$ *exists, and can be found in time* $O(n^2)$.

Proof. We proceed as in the case of Theorem 1, but bail out of the recursion after $\lceil \lg n - \lg\lg n \rceil + 1$ steps. At this point, the partial dominating set D found has at most $\lceil \lg n - \lg\lg n \rceil + 1$ vertices, and the remaining subtournament R (whose vertices are yet to be dominated) has at most $n/2^{\lg n - \lg\lg n + 1} < (\lg n)/2$ vertices. By definition, R dominates all of D.

Now, by looking at the out-neighborhood of $V(R)$ in $V \setminus D$ we check (in $O(n^2)$ time) whether $V(R)$ dominates all of $V \setminus D$. If so, we output $V(R)$, whose size is at most $(\lg n)/2$. Otherwise, there is a vertex x in $V \setminus (D \cup V(R))$ that dominates the vertices of R. In this case, we output $D \cup \{x\}$ as the dominating set. Either way, the size of the dominating set output is at most $\lg n - \lg\lg n + 2$. □

Central to the lower bound result in [1,18] for finding a king, is, the adversary that answers according to the pro-low strategy defined below.

Definition 2 (*Pro-Low Strategy*)**.** Let T be a tournament whose edge directions are determined by the following adversary strategy for an algorithm that queries edges: when the edge uv is queried, it is assigned the direction $u \rightarrow v$ if $\deg^+(u) \leq \deg^+(v)$, and $u \leftarrow v$ otherwise. Here $\deg^+(u)$ and $\deg^+(v)$ denote the outdegree of u and v before the edge query is made.

The following lemma was used in the lower bound arguments.

Lemma 7 (*[1,18]*)**.** *Let T be a tournament and* $S \subseteq V(T)$. *Suppose an adversary answers edge queries involving vertices* $u, v \in S$ *using the pro-low strategy, i.e., it compares out-degrees within* $T[S]$. *If a vertex v in S achieves* $\deg^+_S(v) = t$, *then at least* $t(t+1)/2$ *edge queries must have been made by the algorithm.*

3. Finding a king against a pro-low adversary

As mentioned in the introduction, it is known that $O(n^{3/2})$ probes are sufficient and $\Omega(n^{4/3})$ probes are necessary to find a king in a tournament on n vertices. Narrowing this gap between upper and lower bound has been mentioned as an open problem in literature [1,18]. In this subsection, we take a look at the adversary strategy used in [1,18] and show that there exists an algorithm which works specifically against their strategy and can find a king using $O(n^{4/3})$ edge queries, matching the lower bound given by their adversary.

Theorem 3. *If the adversary follows a pro-low strategy as described in Definition 2 then we can find a king in $O(n^{4/3})$ time.*

Proof. Take a sample of $2n^{1/3}$ vertices from the input V and find a vertex v that dominates at least half of these vertices (such a vertex exists from Lemma 4) using a round-robin tournament on the sample. Remove v and all vertices that lose to v (in that sample) from V, and add v to a set D. Repeat this process of sampling $2n^{1/3}$ elements and discarding elements, and adding elements to D, till there are at least $2n^{2/3}+1$ and at most $3n^{2/3}$ elements remaining which we call R. The number of edge queries made up to this point is $O(n^{4/3})$.

Notice that the vertices in D form a dominating set for all vertices in the input, except those in R. Let $|D| = f$ and $|R| = r$. Clearly $f < n^{2/3} < r/2$. Arrange the vertices of D in non-decreasing order of their out-degrees. Let $v_1, v_2, \ldots, v_f$ be the vertices of D in that order. We shall probe edges between all vertices in D and all vertices in R in a certain order which shall be described below. This step also uses $O(n^{4/3})$ queries, since D and R are both of size $O(n^{2/3})$. The aim is to demonstrate a vertex in R that wins against all vertices in D. Once we find such a vertex, we will be done as the following lemma shows.

Lemma 8. *Let C be a non-empty set of vertices in R that dominate all vertices in D. Then a king of $T[C]$, the induced subtournament on C, is a king for the entire tournament.*

Proof. Let x be a king of $T[C]$. Clearly x can reach all vertices of C in at most two steps. As x dominates every vertex of D, x can reach every vertex of $V \setminus R$ in at most two steps. Now consider a vertex v in $R \setminus C$. As $v \notin C$, v loses to some vertex y of D. So x can reach v by a path of length 2 by passing through y. □

So the rest of the algorithm probes edges between R and D in such a way that the pro-low adversary is made to give a vertex in R that dominates all vertices in D.

Recall that $v_1, v_2, v_3 \ldots v_f$ is the set of vertices in D in non-decreasing order of their degrees. As we probe edges between vertices in D and those of R, some of the degrees of D will change, and if that happens, we rearrange the vertices so that the vertices continue to be in non-decreasing order of their degrees.

We define a *free win point* in D as the maximum integer value p such that for all $i \leq p$, $deg(v_i) \geq i$. By this definition, we can deduce the following.

Lemma 9. *If p is the free win point, then $deg(v_{p+1}) = p$.*

Proof. As the vertices in D are in non-decreasing order of their degrees, $deg(v_{p+1}) \geq deg(v_p) \geq p$ which implies that $deg(v_{p+1}) \geq p$. But $p+1$ is not a free win point and so $deg(v_{p+1}) < p+1$ from which it follows that $deg(v_{p+1}) = p$. □

The following claim gives the importance of the notion of free win point.

Lemma 10. *If p is the free win point, then any vertex v in R will, by default, win at least up to the vertex v_p (if queried in that order) when played against a pro-low adversary.*

Proof. When $i = 1$, it is true, since all vertices in D have outdegree at least $n^{1/3} \geq 1$ and v has 0 wins before querying with v_1. When edge (v, v_i), $i < p$ is probed v has exactly $i-1$ wins since it has won all the $i-1$ probes up to this point, and $deg(v_i) \geq i$. Hence, $\forall i \leq p$, v will dominate v_i. □

The following lemma is immediate from the lemma above.

Lemma 11. *If the free win point p is $|D|$, then all vertices in R which have not queried their edges with any vertices in D will win all vertices of D whenever the queries are made.*

We probe every vertex of D with every vertex in R. After completing with one vertex of R, we simply choose another vertex of R arbitrarily. However, we probe that vertex with vertices in D in a careful order. Now we explain the order in D in which edges are probed between a vertex v in R. We first probe v up to the free win point p, thus making v win against all these vertices and obtain a degree of p. The queries after this point, depend on which of the following cases happen.

- There exists a vertex v' in D such that $degree(v') > degree(v_{p+1}) = p$.
  In this case, we play v with v' to get degree $p+1$ (as degree of v is p and degree of v' is more than p). Then v plays with all vertices with degree p and v would lose to all of them while all of them including v_{p+1} will attain a degree of $p+1$. Then v plays with all the remaining vertices in D.
- The vertex v_{p+1} is a maximum degree vertex in D. In this case, we simply query v' with v_{p+1} and all other vertices with the same degree. Note that as $degree(v)$ would also be p when the first (and possibly even subsequent) queries are made, the adversary may answer either way as the adversary answers arbitrarily in case of ties in the degrees.

Now we show the following about the sequence in D by which we probe vertices of R.

Lemma 12. *Let $x_1, x_2, \ldots, x_r$ be the vertices in R in the order in which they are probed against vertices in D. If the free win point is p before x_i has started probing with vertices in D, then by the time x_{i+1} has completed its round, the free win point will be at least $p+1$ or we would have found a vertex that dominates all vertices of D.*

Proof. The vertex x_i will win with all vertices up to $v_p \in R$ by Lemma 10. If there is no vertex beyond v_p, then x_i is the vertex that dominates all of D.

Otherwise, we argue based on various cases. If there is a vertex whose degree is more than $deg(v_{p+1}) = p$, then the algorithm probes with such a higher degree vertex, wins (due to the pro-low adversary) and then plays with all those with degree p and loses to them (as it has a higher degree when it started playing with them). So all vertices with degree p including v_{p+1} gains a degree and hence the free win point moves to at least $p+1$ as $deg(v_{p+1}) = p+1 \geq p+1$ now.

If v_{p+1} is the only maximum degree vertex, then one of two cases arises as the adversary can answer arbitrarily as there is a tie in the degree of x_i and v_{p+1}. If x_i wins, then it is a vertex that dominates all vertices of D, and otherwise v_{p+1} gains an extra degree moving the free win point.

So the only case we are left to argue is that $degree(v_{p+1})$ has the maximum degree but there is a set M of $m > 1$ vertices with the maximum degree. Here also the adversary can answer arbitrarily due to the tie between the degree of x_i and of the m vertices. If x_i loses to all of them, then all vertices with degree p in D now have degree $p+1$. Thus $degree(v_{p+1})$ changes to $p+1$ and thus the free win point also changes to $p+1$. Otherwise, observe that if x_i wins any one of the m vertices, then it gains a higher degree and hence will lose to subsequent vertices of M due to the pro-low strategy. Hence x_i can win at most one vertex of M and lose to rest of the vertices in M. After x_i completes its round, one of the m vertices of M will gain degree $p+1$ and all others remain with degree p. Now as there is a vertex with degree more than p, due to the cases argued above, x_{i+1} from R will play with such a higher degree vertex before playing vertices with degree p, and hence those with degree p including v_{p+1} will gain a degree and the free win point will advance to at least $p+1$. □

The following claim follows from Lemmas 11 and 12 and the fact that $|R| > 2|D|$ and every vertex in D starts off with outdegree at least $n^{1/3}$ and every vertex in R starts with outdegree 0.

Lemma 13. *There exists a vertex in R that wins against all vertices of D.*

Now the algorithm simply probes all edges between vertices in R and vertices in D, identifies the set of vertices that dominate all of D (and the set is non-empty by Lemma 13) and returns a king among them. The fact that the returned vertex is the king for the entire tournament follows from Lemma 8. The total number of edge queries made during the entire course of the algorithm is clearly $O(n^{4/3})$.

This concludes the proof of Theorem 3. □

4. Finding d-kings and d-covers

As described before a set $S \subseteq V$ is called a *d-cover* of T if for every vertex $v \in V \setminus S$, there is vertex $u \in S$ such that there is a directed path of length at most d from u to v.

While we showed in the last section that the pro-low adversary is weak to provide an improved lower bound for finding a king, here we show that it is still useful to find a lower bound for finding a set of vertices such that there is a directed path of length at most 2 (or more generally d) to every other vertex from one of these vertices. We complement the lower bound by providing an algorithm that achieves this bound for finding $\Omega(\lg n)$ sized d-cover.

4.1. Lower bounds

Generalizing Lemma 7, Ajtai et al. [1] show that

Lemma 14 ([1])**.** *When playing against a pro-low adversary, if an algorithm finds a vertex that reaches at least t vertices by a path of length at most d for $d \geq 1$, then at least $\Omega(t^{1+1/(2^d-1)})$ edge queries must have been made.*

Setting $t = n$ gives the lower bound for finding a d-king. We generalize the above lemma to show

Lemma 15. *If an algorithm playing against the pro-low adversary finds a k-sized d-cover for some k, $d \geq 1$, then it must have made at least $k(n/k)^{1+1/(2^d-1)}$ edge queries.*

Proof. Suppose the algorithm returns a set of vertices $C = \{v_1, v_2, v_3 \ldots, v_k\}$ as the k sized d-cover of the entire tournament. Perform a breadth-first search from the vertices of C with all vertices in C at level 0, those that have direct edges from them at level 1 etc. The breadth-first search tree has d levels. Now, for each vertex in level i (from d to 1) assign a unique vertex at level $i-1$ from which there is a direct edge to it. This partitions the vertices into k sets, where the ith set contains the vertex v_i, and all vertices assigned to it from level 1, and all vertices assigned to those level 1 vertices from level 2 and so on. Thus we have k subtournaments $T_1, T_2, \ldots T_k$ induced on the partition, and let s_i be the number of vertices in the sub-tournament T_i. Note that v_i has an at most d-length path to every vertex in T_i and hence by Lemma 14, the number of edge queries made in the tournament T_i is at least $(s_i)^{1+1/(2^d-1)}$. Thus the total number of edge queries done by all the vertices is at least $\sum_{i=1}^{k}[(s_i)^{1+1/(2^d-1)}]$. As $\sum_{i=1}^{k} s_i = n$, we have that the number of edge queries made is at least $k(n/k)^{1+1/(2^d-1)}$ using Jensen's inequalities [11] (as $\sum_{i=1}^{k}[(s_i)^{1+1/(2^d-1)}]$ attains its minimum when all the s_i's are the same as their average value which is n/k). □

For $d = 1$, we give a slightly different argument which maybe of independent interest.

Theorem 4. *Let $1 \leq k \leq n$. If an algorithm finds a dominating set of size at most k in a tournament of order n, then it makes $\Omega(n^2/k)$ edge queries to its input in the worst case.*

Proof. We prove the result by an adversary argument. The adversary dynamically builds the input instance $T = (V, E)$, a tournament of order n. At all times, it maintains a partition $V = S \cup S'$ of the vertex set.

Initially $S = V$, $S' = \emptyset$ and $\deg^+(v) = 0$ for all $v \in V$. Consider the following invariant which the adversary maintains:

– For each $v \in V$, $\deg_S^+(v) < (n/3k) - 1$.

Lemma 16. *If $|S| \geq n/2$ and $D \subseteq V$ is a dominating set of T, then $|D| > k$.*

Let $v \in D$. Then, by the invariant, v dominates at most $n/3k$ vertices in S (as a vertex in S also dominates itself). Since $|S| \geq n/2$, we have

$$|D| \geq \frac{n/2}{n/3k} > 3k/2 > k.$$

This proves the claim.

The adversary uses the following strategy. When a new edge uv is queried, if $u, v \in S$, the adversary uses the pro-low strategy and directs the edge from u to v if $\deg_S^+(u) \leq \deg_S^+(v)$, and from v to u otherwise. If $u, v \in S'$, the adversary arbitrarily directs the edge. If $u \in S$ (resp. $v \in S$) and $v \in S'$ (resp. $u \in S'$), then the adversary directs the edge from v to u (respectively from u to v). This ensures that there are no edges directed from a vertex in set S to a vertex in set S'.

After a query is answered, if a vertex $v \in S$ satisfies $\deg_S^+(v) = \lceil n/3k \rceil - 1$, v is moved with all its out-neighbors into S'. This ensures that the invariants always hold. It now suffices to show that to reach a stage where $|S| < n/2$, $\Omega(n^2/k)$ queries must be made. Note that $|S|$ shrinks in steps of $\lceil n/3k \rceil$, i.e., its size only halves after $O(3k)$ steps. At each step, a vertex v with $\deg_S^+(v) = \lceil n/3k \rceil - 1$ is moved into S'. For a vertex in S to acquire a degree (in S) of $\lceil n/3k \rceil$, $\Omega((n/3k)^2)$ queries must be made (by Lemma 7 as we use pro-low strategy inside S). So the number of queries needed to halve the size of S is $\Omega(3k(n/3k)^2) = \Omega(n^2/k)$.

Therefore in the worst case, any algorithm that finds dominating sets of size at most k makes $\Omega(n^2/k)$ edge queries. □

Corollary 1. *If an algorithm finds dominating sets of size $O(\log n)$ in tournaments of order n, then it makes $\Omega(n^2/\log n)$ edge queries to its input in the worst case.*

Theorem 5. *Any algorithm that finds minimum dominating sets in tournaments makes $\Omega(n^2/\log n)$ edge queries to its input in the worst case.*

Proof. Since a minimum dominating set has the smallest size among all dominating sets, any algorithm that finds minimum dominating sets in tournaments finds dominating sets of size at most $\log n$, because of Theorem 1. By Corollary 1, this implies that the algorithm makes $\Omega(n^2/\log n)$ edge queries in the worst case.

4.2. Upper bounds

In this subsection we provide algorithms to find d-covers that match the lower bounds proved in the last section, as long as the size of the d-covers is $\Omega(\lg n)$.

Finding a dominating set (1-cover) of size $(k + \lg n - \lg\lg n + 2)$

We start with the following result for 1-cover (dominating set).

Theorem 6. *A* $(k + \lg n - \lg\lg n + 2)$ *sized 1-cover in a tournament on n vertices can be found in* $O(n^2/k)$ *time.*

Proof. Let V be the input set of vertices. If $k \leq 2$ apply Theorem 2. Otherwise, pick a subset of vertices of size $2(n/k)+1$ and find a vertex (say u) which dominates at least (n/k) vertices inside this subset using Lemma 4. This vertex can be found using $O(n^2/k^2)$ time. Delete u's out-neighbors add u to the dominating set. Now recurse on the remaining tournament as long as $|V| \geq 2n/k + 1$. Once $|V| \leq 2n/k$, apply Theorem 2 and find a 1-cover of size $\lg n - \lg\lg n + 2$ for the remaining tournament in $O((n/k)^2)$ time, and add these vertices to the output cover. The output cover is a 1-cover for the tournament, of size at most $k + \lg n - \lg\lg n + 2$. The total time taken is $O((k-2)(n/k)^2 + (n/k)^2) = O(n^2/k)$. □

Finding d-covers $(d \geq 2)$ *of size* $(k + \lg n - \lg\lg n + 2)$

For $d \geq 2$, the lower bound on the size of the d-cover is 1. We first make the following simple generalization of the $O(n^{1+1/(3*2^{d-2}-1)})$ algorithm [1] to find a d-king to show the following.

Theorem 7. *In any tournament of order n, a d-cover of size k can be found in* $O(k(n/k)^{1+1/3(2^{d-2}-1)})$ *time.*

Proof. Partition the vertex set into k parts to get k sub-tournaments on roughly n/k vertices each. Now use the d-king finding algorithm in [1] in each sub-tournament to find a d-king. The set of all these d-kings together form a k-sized d-cover. Time taken to find a d-king in n/k size tournament is $O((n/k)^{1+1/(3*2^{d-2}-1)})$, thus the time taken to find a d-cover of size k is $O(k(n/k)^{1+1/(3*2^{d-2}-1)})$. □

In what follows, we give an improved algorithm for finding d-covers of size more than $\lceil \lg n \rceil$. We start with a lemma similar to Lemma 4 for 2-covers.

Lemma 17. *In any tournament of order n, a vertex that reaches at least* $(n-1)/2$ *other vertices via directed paths of length at most 2 can be found in time* $O(n^{4/3})$.

Proof. Let V be the vertex set of the tournament. Pick a subset of vertices of size $2n^{1/3}+1$ and find a vertex (say u) which dominates at least $n^{1/3}$ of those vertices. Take exactly $n^{1/3}$ of u's losers and remove them from V. Also add u to a set of vertices called C. Repeat this process on the resulting V as long as $|V| \geq 2n^{1/3}+1$. Once $|V| < 2n^{1/3}+1$, perform a round-robin tournament on all the vertices in C and find a vertex k that dominates at least $(|C|-1)/2$ vertices using Lemma 4. The size of C is at least $n^{2/3} - n^{1/3} - 1$.

Thus vertex k can reach at least $n^{1/3}(n^{2/3} - n^{1/3} - 1)/2 = n/2 - (n^{2/3} - n^{1/3})/2$ vertices with path of length exactly two, and can reach $(n^{2/3} - n^{1/3} - 1)/2$ vertices with path of length 1. Thus the vertex k is king of at least $(n-1)/2$ vertices.

The total number of edge queries made is $O(n^{4/3})$, since round robin on set C costs $O(n^{4/3})$ edge queries, as well as the time taken to create the set C is $O(n^{2/3}(n^{1/3})^2) = O(n^{4/3})$. □

It is interesting to note that Yao [19] conjectured that finding a partial order S_k^m (an element which has k elements less than itself and m elements greater than itself) in a total order of size n cannot be done faster even if extra elements are given. Lemma 17 shows that finding a king of n elements becomes substantially easier if $2n$ elements are given. As we do not see any connection at the problem level, we are not sure whether our algorithm provides any insight to address Yao's conjecture.

Corollary 2. *A* $(\lg n - \lg\lg n + 2)$ *sized 2-cover in a tournament on n vertices can be found in* $O(n^{4/3})$ *time.*

Proof. Use Lemma 17 to find a vertex $v \in V$ which has a path of length at most 2, to at least $(n-1)/2$ vertices (say to the set C). Remove v and the vertices in C from V after adding v to the 2-cover set S and adding C to S', and repeat this process on the remaining vertices, till the number of vertices in S is $\lg n - \lg\lg n + 1$. At this point the number of vertices remaining in V is at most $(\lg n)/2$. Find the relation of all vertices with these $(\lg n)/2$ with the vertices in S' using $O(n \lg n)$ queries. One of the two cases happen. Either some vertex v' in S' dominates all vertices in V, in which case we add v' to S and output it as the 2-cover, or no vertex can dominate all vertices in V, in which case V forms a dominating set of size $(\lg n)/2$ and we output as 2-cover, since any 1-cover is a valid 2-cover. This would give a $\lg n - \lg\lg n + 2$ sized 2-cover. □

Theorem 8. *A* $(k + \lg n - \lg\lg n + 2)$ *sized 2-cover in a tournament on n vertices can be found in* $O(k(n/k)^{4/3})$ *time.*

Proof. Take a subset of input vertices of size $2n/k$ from V. Using Lemma 17 find a 2-king u of at least half of these vertices using $O((n/k)^{4/3})$ edge queries. Remove u from V and add it to the output set C, and remove all vertices in V that can be reached by a path of length at most 2 from u. Recurse on the remaining tournament as long as $|V| \geq 2n/k$. When

$|V| < 2n/k$, find a $\lg n - \lg\lg n + 2$ sized 2-cover, by using $O((n/k)^{4/3})$ edge queries using Corollary 2, and add this 2-cover to the set C. The set C is a 2-cover for all the vertices in the tournament of size at most $k + \lg n - \lg\lg n + 2$. The total time spent is $O(k(n/k)^{4/3} + (n/k)^{4/3}) = O(k(n/k)^{4/3})$. □

Now we generalize the above theorem for $d \geq 2$.

Lemma 18. *Let $d \geq 2$ and $k \geq 1$ be an integer. If a k-sized 2-cover can be found in $O(k(n/k)^{4/3})$ time, then a k-sized d-cover can be found in $O(k(n/k)^{1+1/(2^d-1)})$ time.*

Proof. We prove this by induction on d. For $d = 2$, there is nothing to prove. Let $d \geq 3$. Assume that the lemma is true for all integers from 2 to $d - 1$.

Let V be the set of vertices and $s = (n/k)^{1/(2^d-1)}$. As $d \geq 3$, $n/s = n^{1-1/(2^d-1)}k^{1/(2^d-1)} > 3 + \lg n$ (for large enough n), and so using Theorem 6 find a (n/s)-sized 1-cover of V using $O(n^2/(n/s)) = O(ns)$ time. Now find a k-sized $(d-1)$-cover for these n/s vertices using $O(k(n/ks)^{1+1/(2^{d-1}-1)}) = O(ns)$ time using induction hypothesis. The resulting set is a k-sized d-cover of entire input, which is found in $O(ns) = O(k(n/k)^{1+1/(2^d-1)})$ time. □

Thus we have from Theorem 8, Lemma 18 and Theorem 6,

Theorem 9. *A $(k + \lg n - \lg\lg n + 2)$-sized d-cover in a tournament on n vertices can be found in $O(k(n/k)^{1+1/(2^d-1)})$ time for any $k \geq 1$ and $d \geq 1$.*

By setting $k = 1$ in Lemma 18, we have

Corollary 3. *If a 2-king can be found in $O(n^{4/3})$ time, then we can find a d-king in $O(n^{1+1/(2^d-1)})$ time.*

The above corollary was mentioned as a likely possibility (without proof) in the conclusion of [1].

The following corollary follows from Lemma 18 and Theorem 3.

Corollary 4. *If the adversary follows a pro-low strategy as described in Definition 2 then we can find a d-king in $O(n^{1+1/(2^d-1)})$ time.*

The above lemma implies that we need a completely different adversary strategy to improve the lower bounds for finding d-kings.

5. Verification of kings

In this section, we deal with problem of verifying whether a given vertex is a king or not. One can easily verify in $O(n^2)$ time whether a given vertex is a king by running a standard breadth-first-search starting at the vertex and checking whether every vertex is reachable by a path of length at most 2. We give an $\Omega(n^2)$ lower bound for this problem. It is interesting to note that it is NP-complete [5] to verify whether a given vertex is a Banks point.

Lemma 19. *Let T be a tournament of order n. Given a vertex $v \in V(T)$, it takes $\Omega(n^2)$ edge queries to decide whether v is a king in T.*

Proof. Consider the following adversary strategy. The adversary arbitrarily fixes subsets $A, B \subseteq V(T) \setminus \{v\}$ such that $|A| = \lfloor (n-1)/2 \rfloor$ and $|B| = \lceil (n-1)/2 \rceil$, and then assigns edge directions such that $N^+(v) = B$ and $N^-(v) = A$. The remaining edge directions are assigned dynamically.

Suppose $u \in A$, $w \in B$, and the edge uw is queried. The following cases arise.

- There are other non-queried u–B edges: the adversary assigns the direction $u \to w$.
- All other u–B edges have been queried: the adversary assigns the direction $w \to u$.

For edges uw where $u \in A$ and $w \in A$ or $u \in B$ and $w \in B$, the adversary answers arbitrarily. Note that v directly reaches each vertex in $N^+(v)$ by a directed edge, and it can only reach vertices in $N^-(v)$ through vertices in $N^+(v)$. For any vertex $u \in A$, an algorithm cannot decide whether v can reach u until it queries all u–B edges, since the status of v depends on how the adversary answers when the last u–B edge query is made. By the same token, the algorithm cannot decide whether v can reach all vertices of A unless it asks all edges between A and B.

Thus for every $u \in A$, $\lceil (n-1)/2 \rceil$ queries must be made, i.e. a total of $\Omega(n^2)$ queries must be made to determine whether v is a king. □

6. Finding kings in the incremental dynamic setting

In the incremental dynamic setting, we start with a tournament, and the adversary provides a vertex at a time, and the task is to find a king of this new tournament after a new vertex is added to it without starting from scratch. We give an $O(\sqrt{n})$ algorithm to determine a king under vertex additions. We assume that no vertex is deleted.

The algorithm maintains the following invariants/data:

1. A transitive subtournament on a subset $D = \{v_1, v_2, \ldots, v_k\}$ of vertices. For all $j > i$, v_j dominates v_i. Let V' be the set consisting of D and all vertices dominated by D. Then vertex v_k is a Banks point of $T[V']$.
2. Let $R = V \setminus V'$, and by definition of V', every vertex in R dominates all vertices of D.
3. $|R| \leq 2|D| + 1$.
4. The outdegree of every vertex in R in the subtournament $T[R]$.

When a new vertex v is added, it is probed with vertices in D, and if any vertex in D dominates v, then v is added to V' and we return the king declared earlier as the king for this new tournament too. Otherwise, v is added to R after probing its edges with every vertex in R and updating the degree of every vertex in R including that of v. If R is non-empty, then we output a maximum degree vertex in R as a king, otherwise, we return v_k as the king. The correctness is obvious as a maximum degree vertex x in R reaches all vertices in R by a path of length at most two by Lemma 1. The vertex x also dominates all vertices of D and as D is a dominating set of V', x reaches all vertices in V' by a path of length at most two, and hence x is a king of the entire tournament. If R is empty, then as D is a dominating set and as v_k is the source of D, clearly v_k is a king. Since every vertex in R dominates every vertex in D, we maintain the transitive order when x is added as the last vertex in D.

Clearly the first two invariants on D, V' and R are maintained but we need to maintain the (third) size constraint on R. So when $|R| = 2|D| + 1$, we delete a maximum degree vertex x in R from R and add it to D, make it as v_{k+1} and moving all vertices dominated by x from R to V'. This step of moving vertices from R to D reduces the size of R by at least $|D|$ and increases the size of D by 1, thereby ensuring that the third invariant is always maintained.

Now to analyze the number of probes made, we argue that $|D|$ is maintained as $O(\sqrt{n})$. As $|R| \leq 2|D| + 1$ and as the probes of the new vertex are made only with vertices in D and R, the claim that the number of probes made by the algorithm is $O(\sqrt{n})$ will follow.

Lemma 20. $|D| \leq 2\sqrt{n} + 1$.

Proof. Let $D = \{v_1, v_2, \ldots, v_k\}$ be the vertices ordered according to the order at which they are added to D. We first claim that v_k dominates at least $k - 1$ vertices (other than itself).

We prove this by induction on k. This is true for v_1. Assume that $k \geq 2$ and the claim is true for values up to $k - 1$. Noticing the manner in which the algorithm proceeds, just before v_k was added to D, $|R| = 2(k-1)+1$ and v_k is a maximum degree vertex in R and hence by Lemma 4, v_k dominates at least $(k-1)$ unique vertices (which are not dominated by any vertex v_j such that $j < k$ as they were in the set R before being moved to set V') other than itself and the claim follows.

Thus $(n - k) \geq |V'| \geq \sum_{i=1}^{k}(i - 1)$ from which it follows that $(n - k) \geq (k - 1)k/2$ from which it follows that $k \leq 2\sqrt{n} + 1$. □

As a preprocessing step, we can simply find a Banks vertex using the algorithm of Lemma 6 and store the dominating set D of V.

Now we count the time for other bookkeeping operations (beyond the number of edge probes). We notice that we find a maximum outdegree vertex in R whenever R is non-empty. But as we maintain the outdegrees of every vertex in R, this can be found in $O(|R|) = O(\sqrt{n})$ time.

When $|R| = 2|D| + 1$, we delete a maximum outdegree vertex from R to D and delete from R all vertices dominated by that vertex. Note that $\Omega(\sqrt{n})$ vertices are deleted from R and due to that the outdegree of every other vertex in R may need to be updated. This bookkeeping step could take $O(n)$ time as every vertex in R now needs to recount its degree in R. Notice that immediately after the step, $|R|$ decreases by at least half its size while $|D|$ increases by 1. Hence, this expensive step which takes $O(n)$ time will not happen for another $\Omega(\sqrt{n})$ steps. No new edges are queried in R when this split happens, and only the running time gets affected. Thus, the overall amortized time over a long sequence of insertions for updating a king is $O(\sqrt{n})$. Thus we have

Theorem 10. *Given a tournament on n vertices, we can maintain a data structure in $O(n\sqrt{n})$ time so that when a new vertex v is added to the tournament, we can output a king of the new tournament using $O(\sqrt{n})$ edge probes in the worst case and $O(\sqrt{n})$ other bookkeeping operations in the amortized case.*

Note that the algorithm may not return a Banks vertex, as the maximum degree vertex of R (which is output as a king) may not be a Banks point of R.

In what follows, we give a modification to the above algorithm where one can actually find a king in the new tournament in $O(\sqrt{n})$ *worst* case time using some standard tricks. Basically when R is split, instead of moving the highest

degree vertex to D immediately, we store that in a temporary set, and move it over roughly $\sqrt{n}$ steps by which time, we will be able to update the outdegrees in the remaining vertices in R.

As before the algorithm maintains

1. A Banks vertex v_k of a subtournament $T[V']$ ($V' \subseteq V$) of the given tournament, along with the vertices in the transitive subtournament $v_1, v_2, \ldots, v_k$ realizing the Banks vertex where v_k is the source vertex, and the set $D = \{v_1, v_2, \ldots, v_k\}$ that dominates V'.
2. We partition $R = V \setminus V'$ into three disjoint sets R_1, R_2 and R_3 and maintain the invariant that $|R| \leq 3|D| + 1$ and $|R_1| \leq 2|D| + 1$.
3. Whenever $R_2 \neq \emptyset$, $|R_2| \geq |D| + 1$ and it contains a vertex m that dominates all other vertices in R_2. Whenever $R_3 \neq \emptyset$, every vertex of R_3, dominates m, the dominator vertex of R_2.
4. We maintain the outdegree of every vertex in R with respect to the subtournament on vertices of R.
5. We maintain the outdegree of every vertex in R_3 with respect to the subtournament on vertices of R_3.

Initially $R = R_1$ and $R_2 = R_3 = \emptyset$.

When a new vertex v is added to the existing tournament, it is probed as before with vertices in D, and if any vertex in D dominates v, then v is added to V', in which case, the king of the tournament just prior to addition of v in the tournament also is a king of the tournament after v is added and the algorithm terminates.

Otherwise, if v dominates all vertices in D, then v is probed with all vertices in R and the count of outdegrees of all vertices in R (with respect to tournament on R) is updated.

If v is added to R it is added to exactly one of the sets R_1, R_2 or R_3 using the following rules

- If $R_2 = \emptyset$ and $R_1 < 2|D| + 1$, then v is added to R_1. The outdegree of vertices in R is updated.
- If $R_2 = \emptyset$ and $|R_1| = 2|D| + 1$, then we find the maximum degree vertex in R_1 and label it as m. Move m and all vertices dominated by m from R_1 to R_2, making R_1 of size at most $|D|$ (and thereby making R_2 of size at least $|D|+1$). If the new vertex v dominates m, v is added to R_3, else it is added to R_2. If $R_1 \neq \emptyset$, an arbitrary vertex is moved from R_1 to R_3. The outdegree of vertices in R is updated. The outdegree of vertices in R_3 (with respect to tournament on R_3) is maintained.
- If $R_2 \neq \emptyset$ and $R_1 \neq \emptyset$, then we probe v with m. If m dominates v, then we add v to R_2, otherwise we add v to R_3 after updating the outdegree of all vertices of R_3 within the subtournament induced on R_3. Regardless of whether we add v to R_2 or R_3), we also remove an arbitrary vertex of R_1 and move it to R_3 and update the outdegrees of vertices in R_3(with respect to tournament on R_3).
- If $R_2 \neq \emptyset$ and $R_1 = \emptyset$, then we add m to D as $v_{|D|+1}$, move all other vertices in R_2 to V' and move all vertices in R_3 to R_1. Now R contains only the vertices which were earlier in R_3. The outdegrees of R_3 were already maintained anyway, and thus the outdegrees of every vertex in R is also maintained.

It is easy to see that the invariants 1, 3, 4, 5 are maintained. Now we prove that the invariant 2 is also maintained.

Notice that whenever $R_2 = \emptyset$, $|R_1| < 2|D| + 1$ and R_3 are empty. Thus $|R| \leq 3|D| + 1$ and $|R_1| \leq 2|D| + 1$.

When $|R_1| = 2|D| + 1$ vertices, R_1 gets split and atleast $|D| + 1$ vertices move from R_1 to R_2, and a new vertex is added to either R_2 or R_3. Suppose $d \leq |D|$ is size of R_1 at this point. Thus, in this case also $R \leq 3|D| + 1$ and $|R_1| \leq 2|D| + 1$.

When $R_1, R_2 \neq \emptyset$, it means that R_1 has been split, and everytime a new vertex is added to R, it is either added to R_2/R_3 and exactly one vertex in R_1 moves to R_2/R_3. Before R_1 becomes empty, at most d vertices have moved from R_1 to R_3 and d new vertices are added to R. Thus, whenever $R_2 \neq \emptyset$ and $R_1 = \emptyset$, the maximum size of R is d(number of new vertices added to R)+d (number of vertices moved from R_1 to R_2/R_3)+$(2|D| + 1 - d)$ (number of vertices in $R_2 \cup R_3$ when R_1 was last split) $= 2|D| + 1 + d$. Since $d \leq |D|$, the maximum size of R is $3|D| + 1$. No vertices are added to R_1 whenever $R_1, R_2 \neq \emptyset$. Thus $|R_1| \leq d \leq |D|$. When $R_2 \neq \emptyset$ and $R_1 = \emptyset$, at least $|D| + 1$ vertices are moved from R to V', which ensures that $|R| < 3|D| + 1$.

When $R_2 \neq \emptyset$ and $R_1 = \emptyset$, all vertices of R_2 are moved to V', and then vertices in R_3 are moved to R_1. Thus at least $|D| + 1$ vertices are moved from R to V', thus $R \leq 2|D|$, and then the sets R_2, R_3 become empty. So $|R| \leq 3|D| + 1$, and $|R_1| \leq 2|D| + 1$.

This proves that invariant 2 is always maintained.

Now to output a king vertex of the updated tournament, the algorithm checks if R is empty or not. Whenever R is empty, the Banks vertex of V' is the Banks point of entire tournament and thus it is also a king. Whenever R is non-empty, a maximum degree vertex in R (which happens to be a king of R) is also a king of entire tournament, which is always maintained by the invariant 4. This proves the correctness of the algorithm.

The running time (including the number of queries made) for incrementally maintaining a king is clearly $O(|D|) = O(\sqrt{n})$ in the worst case. Thus, we have

Theorem 11. *Given a tournament on n vertices, we can build a data structure in $O(n\sqrt{n})$ time so that when a new vertex v is added to the tournament, we can output a king of the new tournament in $O(\sqrt{n})$ time in the worst case.*

The algorithms in Theorems 10 and 11 always return a king, but they do not necessarily return a Banks point at every step. This is because the maximum outdegree vertex of R may not dominate all vertices in R. We could maintain a Banks

point within R and return such a Banks point whenever R is non-empty. And when the size invariant on R is violated, it makes sense to move the Banks point along with the dominated vertices to D, but then we move too many vertices to D and this requires care.

First recall that a Banks point is simply the source vertex of a transitive subtournament whose vertices dominate the entire tournament. So if we simply maintain the transitive subtournament (which maybe of large size) and compare the newly added vertex to the vertices of the subtournament, we can find a Banks point. If the new vertex dominates all vertices of the transitive subtournament, then it is a Banks point and otherwise the earlier Banks point is the Banks point for the new tournament. But as the size of the transitive subtournament can be $\Theta(n)$ this results in a linear time algorithm to report a new Banks point.

Lemma 21. *Given a tournament on n vertices, we can build a data structure in* $O(n^2)$ *time so that when a new vertex* v *is added to the tournament, we can output a Banks point of the new tournament in* $O(n)$ *time in the worst case.*

We shall now use this lemma together with the strategy in Theorem 11 to improve the bound for finding a Banks point to $O(\sqrt{n})$. In particular, we continue to maintain the invariants 1, 2 and 3 of Theorem 11. The invariants 4 and 5 are changed to the following conditions.

- Maintain a Banks point over all vertices in R in addition to maintaining the relation between every pair of vertices in R.
- Maintain a Banks point over vertices R_3 with respect to the subtournament on vertices of R_3.

First, we notice that the algorithm in Theorem 11 always returns a Banks point whenever the set R is empty. Whenever R is non-empty it only contains vertices which dominate all vertices in D. Thus any Banks point of R is a Banks point of entire tournament.

Initially $R = R_1$ and $R_2 = R_3 = \emptyset$.

When a new vertex v is added to the existing tournament, it is probed with vertices in D, and if any vertex in D dominates v, then v is added to V', in which case, the Banks point of the tournament just prior to addition of v in the tournament also is declared as a Banks point.

Otherwise, if v dominates all vertices in D, then v is probed with all vertices in R and a Banks point of all vertices in R (with respect to tournament on R) is updated in $O(|R|) = O(\sqrt{n})$ time using Lemma 21.

If v is added to R it is added to exactly one of the sets R_1, R_2 or R_3 using the exact same rules as used in Theorem 11. But there is one more step, which is performed in addition to the procedure followed in Theorem 11 while a vertex is added to R. Whenever R and R_3 are not empty, a Banks point with respect to R and a Banks point with respect to R_3 are also maintained. This takes an additional $O(|R| + |R_3|) = O(\sqrt{n})$ time/edge queries by the use of Lemma 21.

The arguments for maintaining invariants, correctness and query complexity are similar to the one showed in Theorem 11, which gives us the following theorem.

Theorem 12. *Given a tournament on n vertices, we can build a data structure in* $O(n\sqrt{n})$ *time so that when a new vertex* v *is added to the tournament, we can output a Banks point of the new tournament in* $O(\sqrt{n})$ *time in the worst case.*

7. Conclusions and open problems

We have investigated the complexity of finding Kings, Banks points and d-dominating sets in tournaments. While our algorithms are not very involved, they are strengthened by the fact that the algorithms to find above $\Omega(\lg n)$ sized d-dominating sets are provably optimal. We have also provided some additional insights into the complexity of finding a d-king which may help narrow the gap between upper and lower bound for the complexity of the problem. We believe that our work will spur further work on improving the bounds for finding a king in tournament. We end with some specific problems from the work on tournaments.

- There is still a gap in the complexity of finding a d-king for $d \geq 2$. We have shown in Corollary 3 that to improve the upper bound, it is sufficient to improve the upper bound for finding 2-kings. Is the converse true? I.e. can we improve the upper bound for finding d-kings ($d > 2$) even if a 2-king cannot be found in $o(n\sqrt{n})$ time?
- Are the bounds in Theorem 9 optimal for $k < \epsilon \lg n$, for $d \geq 2$? For $d = 1$, we know that finding a k-sized dominating set is $W[2]$-complete.
- For vertices u and v, let $b(u, v)$ denote the number of vertices through which u can reach v by a directed path of length 2. Then, a king z is strong [7] if the following condition is satisfied: $b(z, v) > b(v, z)$ whenever v dominates z. I.e. if v dominates z, the number of ways to reach v from z is more than the number of ways to reach x from v through a directed path of length at most 2. It can be shown that a maximum degree vertex in a tournament is a strong king. Can a strong king be found in $o(n^2)$ time? It is also known [3] that we need $\Omega(n^2)$ time to find a maximum degree vertex in a tournament.
- Can we show that the lower bound for finding a Banks point is $\Omega(n\sqrt{n})$?

References

[1] M. Ajtai, V. Feldman, A. Hassidim, J. Nelson, Sorting and selection with imprecise comparisons, ACM Trans. Algorithms 12 (2) (2016) 19:1–19:19.
[2] N. Alon, J. Spencer, The Probabilistic Method, John Wiley, 1992.
[3] R. Balasubramanian, V. Raman, G. Srinivasaragavan, Finding scores in tournaments, J. Algorithms 24 (2) (1997) 380–394.
[4] A. Biswas, V. Jayapaul, V. Raman, Improved bounds for poset sorting in the forbidden-comparison regime, in: Algorithms and Discrete Applied Mathematics - Third International Conference, CALDAM 2017, Sancoale, Goa, India, February 16-18, 2017, Proceedings, 2017, pp. 50–59.
[5] F. Brandt, F.A. Fischer, P. Harrenstein, Recognizing members of the tournament equilibrium set is NP-hard, 2007, CoRR abs/0711.2961.
[6] F. Brandt, F.A. Fischer, P. Harrenstein, The computational complexity of choice sets, Math. Log. Q. 55 (4) (2009) 444–459, http://dx.doi.org/10.1002/malq.200810027.
[7] A.H. Chen, J.M. Chang, Y. Cheng, Y.L. Wang, The existence and uniqueness of strong kings in tournaments, Discrete Math. 308 (12) (2008) 2629–2633.
[8] P. Dey, Query complexity of tournament solutions, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA, 2017, pp. 2992–2998, URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14180.
[9] R. Downey, M. Fellows, Parameterized Complexity, Springer New York, 1999.
[10] R.L. Graham, J.H. Spencer, A constructive solution to a tournament problem, Canad. Math. Bull. 14 (1971) 45–48.
[11] G.H. Hardy, J.E. Littlewood, G. Pólya, Inequalities, Cambridge University Press, Cambridge, 1997, p. xii, 324.
[12] X. Lu, D. Wang, C.K. Wong, On the bounded domination number of tournaments, Discrete Math. 220 (1–3) (2000) 257–261.
[13] S.B. Maurer, The king chicken theorems, Math. Mag. 53 (2) (1980) 67–80.
[14] N. Megiddo, U. Vishkin, On finding a minimum dominating set in a tournament, Theoret. Comput. Sci. 61 (1988) 307–316.
[15] J. Moon, Topics on Tournaments, in: Athena Series: Selected Topics in Mathematics, Holt, Rinehart and Winston, 1968.
[16] C.H. Papadimitriou, M. Yannakakis, On limited nondeterminism and the complexity of the V-C dimension, J. Comput. System Sci. 53 (2) (1996) 161–170, http://dx.doi.org/10.1006/jcss.1996.0058.
[17] E.M. Penn, The banks set in infinite spaces, Soc. Choice Welf. 27 (3) (2006) 531–543, http://dx.doi.org/10.1007/s00355-006-0144-9.
[18] J. Shen, L. Sheng, J. Wu, Searching for sorted sequences of kings in tournaments, SIAM J. Comput. 32 (5) (2003) 1201–1209.
[19] F.F. Yao, On Lower Bounds For Selection Problems, Tech. rep, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.