The Power of the Score Sequence of a Tournament

Prantar Ghosh
Tennessee Technological University, Cookeville, TN, USA

Sahil Kuchlous
Unaffiliated, Irvine, CA, USA

Shravan Mehra
University of Birmingham, UK

Sagnik Mukhopadhyay
University of Birmingham, UK

Abstract

What problems can one solve on a tournament if only its score sequence is known?

Tournaments are oriented complete graphs that form an extensively-studied class of directed graphs (digraphs), both from combinatorial and algorithmic perspectives. Over the years, researchers have identified multiple classical digraph problems that can be solved on a tournament from only its score sequence (indegree sequence). These problems include *acyclicity testing* and *topological sorting* [Chakrabarti, Ghosh, McGregor, and Vorotnikova; SODA'20], s,t*-reachability*, *strong connectivity*, and decomposition into *strongly connected components (SCC)* [Ghosh and Kuchlous; ESA'24], and vertex-ordering problems such as *cutwidth* and *optimal linear arrangement* [Barbero, Paul, and Pilipczuk; ICALP'17]. These prior works showed the sufficiency of the score sequence by designing distinct algorithms for the individual problems. In this work, we give a simple unified framework that solves all these problems using only indegrees and, in fact, completely characterises the class of problems that is determined by the indegree information: problems whose answers are invariant under cycle reversals.

As a byproduct of our results, we obtain algorithms for a variety of connectivity-based, cut-based, and vertex-ordering problems on tournaments and almost-tournaments in the streaming, the two-player communication, and the cut-query models of computation. Some of these algorithms match existing optimal bounds and others provide new bounds improving the state of the art. Specifically, our polynomial-time algorithms for almost-tournaments improve upon the exponential-time algorithms of Ghosh and Kuchlous and have much simpler analysis.

The said characterisation is a special case of a much more general result that we establish: for any *arbitrary* digraph, the knowledge of its skeleton (underlying undirected graph) and the vertex indegrees completely determines its properties that are invariant under cycle reversal. In particular, this gives us an $O(n^2)$-cut-query algorithm to solve directed minimum cut on n-node graphs in polynomial time, a significant result that has been observed in the literature but not concretely stated in this form. Our results also unveil interesting general connections between two well-studied sublinear models for graph problems: semi-streaming and cut-query.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms; Mathematics of computing → Graph theory

Keywords and phrases tournaments, score sequence, cycle reversal, streaming algorithms, graph connectivity, cut queries, min-cut

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.156

Related Version *Full Version*: https://arxiv.org/abs/2607.15260

Funding *Prantar Ghosh*: Funded in part by startup support provided by the Department of Computer Science and the College of Engineering at Tennessee Tech. Part of this work was done while the author was visiting the University of Birmingham, UK, hosted by Sagnik Mukhopadhyay.
Sagnik Mukhopadhyay: Partially supported by UKRI New Investigator Award EP/X03805X/2.

Acknowledgements We thank Deeparnab Chakrabarty for helpful discussions and for pointing us to relevant prior works by Cunningham. We are also grateful to an anonymous reviewer for referring us to the paper "How to orient the edges of a graph?" by Frank and Gyárfás. Finally, we thank anonymous ESA 2026 reviewers for their suggestions and comments that helped in improving the presentation of the paper.

1 Introduction

Tournaments are digraphs where each pair of nodes shares exactly one directed edge. They have a rich literature that dates back more than half a century (see the book on tournaments by Moon [56]). In particular, the *score sequence* of a tournament, which is the sequence of its indegrees, has been a topic of extensive study [57, 59, 61, 51, 65, 21]. It has been noted sporadically in the literature that multiple classical digraph problems can be solved on a tournament from only its indegree sequence. For instance, Chakrabarti, Ghosh, McGregor, and Vorotnikova [13] observed that the acyclicity of a tournament and the topological ordering of an acyclic tournament are completely determined by its indegree sequence: for the former, check if the indegrees are a permutation of $\{0, \ldots, n-1\}$, and for the latter, sort the nodes by indegree. Again, it is known that sorting by indegree yields optimal solutions for two widely-studied vertex ordering problems: *cutwidth* (as shown by Fradkin [31]) and *optimal linear arrangement* (shown by Barbero, Paul, and Pilipczuk [8]); see Section 2.3 for the definition of these problems. A recent and more surprising result of Ghosh and Kuchlous [35] says that the indegree information also determines s,t-reachability, strong connectivity, and, in general, all strongly connected components (SCCs) of a tournament. This raises an interesting combinatorial question:

Can we characterise the class of problems determined by the score sequence of a tournament?

We answer this question in the affirmative and show that this class is *precisely* the set of all digraph problems whose answers are invariant under cycle reversal, i.e., whose answers remain the same even if we flip any number of cycles in the graph. Then, given the score sequence, we can always have an algorithm to solve such a problem, even if using exponential runtime: brute force over all possible tournaments on the given number of nodes to find one with that score sequence and solve the problem on that input using a classical algorithm. Can we, however, always guarantee a *polynomial-time* algorithm? We answer this in the affirmative as well, provided that the problem admits a polynomial-time solution in the classical model. Indeed, we achieve this by designing a polynomial-time algorithm to generate a tournament satisfying a given indegree sequence. We thus have the following theorem.

▶ **Theorem 1.** *The class of graph problems determined by the indegree sequence of a tournament is precisely the class of problems whose answers are invariant under cycle reversal. Moreover, each problem can be determined in polynomial time from the indegree sequence (provided that it has a polynomial-time algorithm when the entire input graph is given).*

Since all problems mentioned in the first paragraph can be easily shown to belong in this class, our result generalises all of them. To solve those problems using the indegree information, prior work proposed different algorithms to process the indegrees, tailored according to the respective problems. We give a simple unified polynomial-time framework for all those problems as described above.

Generalisation for Arbitrary Digraphs. Since tournaments form a rather narrow subclass of digraphs, it is natural to ask: *can we generalise the result to a broader class of graphs?* Theorem 1 is indeed a special case of our more general theorem: all problems that are invariant under cycle reversal can be solved on any general digraph only from the knowledge of its underlying undirected graph a.k.a. *skeleton* and indegree sequence.

▶ **Theorem 2.** *The class of graph problems determined by the skeleton and indegree sequence of a directed graph is precisely the class of problems whose answers are invariant under cycle reversal. Moreover, each problem can be determined in polynomial time from the skeleton and the indegree sequence (provided that it has a polynomial-time algorithm when the entire input graph is given).*

Since the skeleton of a tournament is the complete graph, its score sequence suffices.

Applications: Sublinear Algorithms. In the modern world of massive graphs, a natural goal is to analyse them by extracting as little information as possible from the graphs. Tournaments, which have the maximum possible edges, are prime examples of such graphs. They appear in the real world when, for example, there is a pairwise ranking or preference (represented by directed edges) among all elements in a dataset [33, 34, 62]. Common questions that arise here include detecting whether the rankings are consistent (acyclicity testing), and if so, ordering the elements in increasing order of preference (topological sorting), whether an element t is (directly or indirectly) preferred over another element s (s,t-reachability), or partitioning the elements into groups such that there is a linear ordering of preference among the groups (SCC decomposition).

This motivates the study of *streaming algorithms* for tournaments [13, 9, 35, 19]. These algorithms make one or a few passes over the edges while maintaining a small summary and then process the summary to generate an output. They usually aim for *semi-streaming* space, i.e., $\widetilde{O}(n) := O(n \operatorname{poly}(\log n))$ space for n-node graphs.[1] Observe that problems completely determined by the vertex indegrees admit single-pass semi-streaming algorithms: while processing the stream, we just keep count of the indegrees using $O(n \log n)$ space. We thus get semi-streaming algorithms for all cycle-reversal-invariant problems on tournaments, and they also carry over to a couple of other popular models for sublinear graph algorithms: (a) the *cut-query model* where we solve a problem by optimizing queries to an oracle that returns the (directed) cut value for any subset of nodes and (b) the *two-player communication model* where the input graph is split between two players who need to communicate (possibly in multiple rounds of interaction) as few bits as possible to solve the problem. For (a), we need n (singleton) cut queries to find all vertex indegrees. For (b), standard streaming-to-communication simulations imply $O(n \log n)$ communication protocols for the problems. Hence, we have the following theorem.

▶ **Theorem 3.** *Given any digraph problem $\mathcal{P}$ whose answer is invariant under cycle reversal, there is a single-pass $O(n \log n)$-space streaming algorithm, an $O(n)$-cut-query algorithm, and an $O(n \log n)$-communication protocol for the problem $\mathcal{P}$ on tournaments. Each algorithm runs in polynomial-time, provided $\mathcal{P}$ has a polynomial-time classical algorithm on tournaments.*

[1] Most graph problems turn out to be intractable when restricted to smaller memory [27].

In fact, Chakrabarti et al. [13] and Ghosh and Kuchlous [35] discovered their aforementioned results while studying streaming algorithms for tournaments and consequently obtained semi-streaming algorithms for acyclicity testing, topological sorting, s,t-reachability, strong connectivity, and SCC decomposition. Our result implies such algorithms for all these problems, and also the first semi-streaming algorithms for the fundamental problems of directed minimum cut and minimum s,t-cut on tournaments. Notably, although there is a substantial body of work on undirected min-cut [71, 3, 58, 4, 26, 47, 50], the streaming and communication complexities of directed (global or s,t) min-cut was unknown, even for tournaments. In particular, we obtain the following corollary.

▶ **Corollary 4.** *There is a single-pass $O(n \log n)$-space streaming algorithm, an $O(n)$-cut-query algorithm, and an $O(n \log n)$-communication protocol for the following problems on tournaments, each of which runs in polynomial time.*

- **(i)** *Connectivity-based problems: s,t-reachability, strong connectivity, SCC decomposition (or more generally finding the SCC-graph);*
- **(ii)** *DAG-based problems: acyclicity testing, topological sorting;*
- **(iii)** *Vertex ordering problems: optimal linear arrangement (OLA), cutwidth;*
- **(iv)** *Directed cut problems: minimum cut, minimum s,t-cut.*

All these bounds are optimal (up to a $\log n$ factor).

We remark that having the same "sketch" of the tournament – the indegree-count – for a large variety of problems is very convenient from a practical perspective: the sketch is deterministic, can be computed very easily and quickly, and rather than storing separate summaries based on the respective problems at hand, we can just post-process this single sketch accordingly upon receiving the respective queries.

Extension to Almost Tournaments. Again, a natural question about generalisability arises: *can we extend these streaming algorithms to a broader class of graphs?* Indeed, we cannot fully generalise them since most of these problems are hard in streaming (i.e., require $\Omega(n^2)$ space for a single pass) for general digraphs [27, 13]. However, Ghosh and Kuchlous [35] showed that they are tractable when the digraph is "close to" a tournament: for digraphs that can be obtained by deleting at most k edges from a tournament, they gave single-pass $\widetilde{O}(n+k)$-space streaming algorithms for the connectivity-based problems mentioned above. Our general theorem for arbitrary digraphs (Theorem 2) implies these algorithms and extends them to any cycle-reversal-invariant problem. This is because we can recover the skeleton of such a graph in $O(k \operatorname{poly}(\log n))$ space: as shown by the said work, we can recover the k non-edges using this space. Again, in the cut-query model, they can be retrieved using $O(k \log n)$ queries by standard techniques. We thus get the following corollary.

▶ **Corollary 5.** *Given any digraph problem $\mathcal{P}$ whose answer is invariant under cycle reversal, there is a single-pass $\widetilde{O}(n+k)$-space streaming algorithm, an $\widetilde{O}(n+k)$-cut-query algorithm, and an $\widetilde{O}(n+k)$-communication protocol for $\mathcal{P}$ on digraphs that are k-close to tournaments. Each algorithm runs in polynomial-time, provided $\mathcal{P}$ has a polynomial-time classical algorithm on such digraphs.*

Furthermore, we get an *exponential improvement in runtime* over [35]'s algorithm for almost tournaments. Their runtime for s,t-reachability and strong connectivity is $\widetilde{O}(2^{\min(k,n)})$ and that for SCC decomposition or SCC-graph construction is $\widetilde{O}(2^n)$. Our algorithm is not only polynomial time for any k, but it also has much simpler analysis, especially for the SCC decomposition algorithm where their analysis is somewhat intricate.

Other Applications. We remark that our general theorem (Theorem 2) for arbitrary digraphs implies the following results in the cut-query model.

▶ **Corollary 6.** *For any directed graph, global- and s,t-minimum cut can be solved with $O(n^2)$ directed cut queries and polynomial runtime.*

Indeed, the directed global-minimum cut problem in the cut-query model is a special instance of the fundamental submodular function minimisation (SFM) problem since the cut function (directed or undirected) is submodular.[2] The SFM problem asks to minimise a submodular function $f : 2^{\mathcal{U}} \to \mathbb{R}$ when given access to an oracle that, upon query $S \subseteq \mathcal{U}$, returns the value of $f(S)$. There has been a long line of work [54, 39, 23, 28, 43, 64, 42, 44, 60, 69, 17, 53, 7, 46, 15, 16] to pin down the exact complexity of SFM. The current best algorithm due to [45] makes $\widetilde{O}(n^2)$ queries but has has exponential runtime. The best-known (strongly) polynomial-time algorithm has query complexity $\widetilde{O}(n^3)$, also by [45]. Therefore, these algorithms for SFM do not imply Corollary 6. Again, while the undirected analogue of Corollary 6 is trivial [63], the directed version is not; since unlike undirected graphs, it is provably impossible to recover a directed graph exactly using $O(n^2)$ cut queries.[3]

Corollary 6, however, follows from some observations in prior work ([23] Section 2) but was not stated in this form, which we believe is worth stating explicitly in light of the discussion above. We also remark that our proof technique and algorithm seem simpler than that of [23].

Finally, in Section 5, we show that our results establish an interesting connection between the streaming and the cut-query models. For tournament problems, existence of a cut-query algorithm (irrespective of the number of queries) implies the existence of a semi-streaming algorithm (Lemma 14). Thus, strong streaming lower bounds can be a useful tool for proving intractability in the cut query model. Again, since this implies that the semi-streaming model is at least as (algorithmically) strong as the cut-query model for tournament problems, a natural question is whether it is *strictly* stronger: is there a tournament problem that is undecidable by cut queries, but solvable by a semi-streaming algorithm? We answer this question in the affirmative (Proposition 15).

Related Work and Comparisons

Streaming and Communication. Apart from the results mentioned above, Chakrabarti et al. [13] obtained algorithms and lower bounds for approximate feedback arc set in tournaments (FAST) and for source/sink-finding, both in adversarial-order and random-order streams. Baweja, Jia, and Woodruff [9] then improved [13]'s results for feedback arc set. They also gave $(p+1)$-pass $O(n^{1+1/p})$-space algorithms for SCC decomposition (also implying an algorithm for reachability), strong connectivity, and hamiltonian path on tournaments. Ghosh and Kuchlous [35] improved upon their results for the connectivity-based problems by designing single-pass semi-streaming algorithms for each of them. They also gave new algorithms for hamiltonian cycle and FAST, and established lower bounds for multiple problems including (exact and approximate) FAST, s,t-distance, source/sink finding, and the connectivity and DAG-based problems. All these works considered the respective tournament

[2] A function $f : 2^{\mathcal{U}} \to \mathbb{R}$ is *submodular* iff for any $S, T \subseteq \mathcal{U}$, we have $f(S) + f(T) \geqslant f(S \cup T) + f(S \cap T)$.
[3] In fact, the impossibility holds for any number of cut queries; to wit, a directed cycle and its reversal are distinct labelled graphs, but all cut queries have identical answers on both of them.

problems in the two-party communication model in order to establish lower bounds. Mande, Paraashar, Sanyal, and Saurabh [55] recently studied the communication complexity of multiple tournament problems including king finding, source finding, and maximum degree, giving both upper and lower bounds.

The connectivity- and DAG-based problems on *general digraphs* have been studied in the streaming and communication models by multiple works [27, 40, 13, 6, 20, 5, 9]. These works mostly studied lower bounds for the problems, with the best-known ones being $\Omega(n^2)$ space for a single pass, $\Omega(n^{2-o(1)})$ space for $O(\sqrt{\log n})$ passes, and $\Omega(n^{1+\Omega(1/p)})$ space for p passes, thereby needing at least $\Omega(\log n/\log\log n)$ passes for semi-streaming algorithms. The s,t-reachability problem is the only one among them (to the best of our knowledge) that has a non-trivial pass upper bound for semi-streaming algorithms: $O(n^{1/2+o(1)})$ passes suffice [5]. Another notable work is by Laura and Santaroni [52] who studied SCC decomposition on arbitrary graphs under W-streams (allowing outputs to be streamed), obtaining an $O(n)$-pass $O(n\log n)$-space algorithm.

Cut query. Graur et al. [38] showed that a directed graph can be fully learned up to cycle reversals via $O(n^2)$ cut queries,[4] which implies exponential-time $O(n^2)$-cut-query algorithms for directed global- and s,t-minimum cut. They, however, did not mention a concrete polynomial-time algorithm that generates such a digraph, which would have then implied our Corollary 6. Cunningham [23] proved a statement similar to Corollary 6 using general techniques for submodular function minimisation; we use simpler techniques specific to graphs. We are not aware of any prior work that studied tournaments in the cut query model. The *undirected-cut query* model has been recently studied extensively for solving undirected-cut related problems [63, 58, 1, 38, 53, 12, 18, 47, 48, 49, 50].

Other settings. Frank and Gyárfás [32] studied the problem of orienting the edges of an undirected graph so as to satisfy target upper and lower bounds on the vertex in- and out-degrees, a generalized version of our main problem (Lemma 9). They mentioned an algorithm for this problem that flips directed paths and also remarked that the problem can be solved using network-flows, but did not specify the algorithm. While they might have alluded to an algorithm similar to ours, the final result we get for general digraphs (Theorem 2) is novel. Coppersmith et al. [22] proved that the indegree sequence of a tournament suffices in achieving a 5-approximation to FAST. The problems of optimal linear arrangement (OLA) and cutwidth have been widely studied from the perspective of computational complexity as well as fixed parameter tractability (FPT) [31, 30, 8, 25, 10].

2 Preliminaries

Notation and Terminology. We state some notation and terminology that we use throughout the paper. The notation $\widetilde{O}(f)$ hides factors polylogarithmic in f. For $a \in \mathbb{N}$, we have $[a] := \{1,\ldots,a\}$. Unless otherwise specified, a general digraph is denoted by $G=(V,E)$ where $|V|=n$ and $|E|=m$. The *skeleton* of a digraph is its underlying undirected graph, and is denoted by $\mathsf{S}=(V,\widetilde{E})$ where $\widetilde{E}=\{\{u,v\} : (u,v)\in E\}$ is a multiset – allowing S to contain up to two parallel edges (corresponding to directed 2-cycles in G). For a vertex v, we use $d_{\text{in}}(v,G)$ and $d_{\text{out}}(v,G)$ to denote the indegree and outdegree of v respectively in

[4] We came to know via personal communication that an independent unpublished work by Chakrabarty, Chen, and Khanna [14] also obtained a similar result.

the digraph G. We simply write $d_{\text{in}}(v)$ and $d_{\text{out}}(v)$ if the graph G is clear from the context. The vector $\mathbf{d}$ is an n-length sequence of non-negative integers and may denote the indegree sequence of a graph, given by $\langle d_{\text{in}}(1), \ldots, d_{\text{in}}(n)\rangle$ where the vertex set is $[n]$. We use $\mathcal{G}[\mathsf{S}, \mathbf{d}]$ to denote the equivalence class of all graphs that have skeleton S and indegree sequence $\mathbf{d}$. We say that two graphs G_1 and G_2 are related, i.e., $G_1 \sim G_2$ iff both of them have the same skeleton and indegree sequence. Unless otherwise specified, a *cut* refers to a directed cut. Given a vertex subset U of a digraph, the cut size of U, i.e., the number of edges incoming to U, is given by $\overleftarrow{\mathcal{C}}_G(U) := |\{(v,u) \in E(G) : u \in U\}|$. We often refer to cut sizes by simply "cuts". For a directed graph G, we define its *cut profile* as the set $\{(U, \overleftarrow{\mathcal{C}}(U)) : U \subseteq V\}$. The number of edges outgoing from U is given by $\overrightarrow{\mathcal{C}}_G(U) := |\{(u,v) \in E(G) : u \in U\}|$

2.1 Models

We describe the algorithmic models that appear in this work.

The Directed Cut-Query Model. The *undirected* cut-query model was introduced by Rubinstein, Schramm, and Weinberg [63]. Here, we define the analogous *directed* cut-query model [38, 14].

We are given query access to an oracle that holds a digraph $G = (V, E)$, where we know V but not E. Upon a query $U \subseteq V$, the oracle returns $\overleftarrow{\mathcal{C}}_G(U)$, the number of edges incoming to U in G. The goal is to solve a problem on G by making as few queries as possible.

Equivalently, we could define the model as returning $\overrightarrow{\mathcal{C}}_G(U)$, the number of edges outgoing from U, but since $\overleftarrow{\mathcal{C}}(U) = \overrightarrow{\mathcal{C}}(\overline{U})$, WLOG and incurring at most a factor of 2 in the number of queries, we can assume that we have access to just one type of queries. We drop the subscript G if the graph is clear from the context.

The Graph Streaming Model. In this paper, we only focus on the insert-only streaming setting and define only this variant. Here, we have an input digraph $G = (V, E)$, where the vertex set $V = [n]$ and the elements of the edge set E arrive one by one. The input graph and the stream order are chosen adversarially (worst case). The goal is to make one or a few passes over the stream elements, store as few bits of information as possible, and after the final pass, run a post-processing algorithm on the stored data to generate an output. The goal is to minimise the space usage and the number of passes. The semi-streaming model [27] restricts the space usage to $\widetilde{O}(n)$. In this work, we only deal with single-pass algorithms.

Two-Party Communication. We study graph problems in the two-player communication model of Yao [70]. Here, players Alice and Bob both know the vertex set V and receive disjoint edge sets $E_A, E_B \subseteq V \times V$ respectively. They send messages to each other (possibly over multiple rounds) to compute $\mathcal{F}(G)$ for some relation $\mathcal{F}$ defined on the graph $G = (V, E_A \sqcup E_B)$. The *communication cost* of a protocol for $\mathcal{F}$ is defined as the maximum number of bits exchanged by the players, over all possible inputs (E_A, E_B). If the protocol is randomized, then the maximum is also taken over all possible random strings used by the players. The *communication complexity* of a relation $\mathcal{F}$ is the minimum communication cost over all protocols that return a valid output for $\mathcal{F}$ on all inputs. If the protocol is randomized, we require the answer to be correct on all inputs with probability at least $2/3$ over the randomness of the players' random strings. All our protocols in this paper are single-round, i.e., Alice sends a single message to Bob, following which he returns the output. Our lower bounds hold even for the stronger model allowing unlimited rounds of interaction.

2.2 Basic Tools

▶ **Fact 7** (Sparse recovery [36, 24]). *Given streaming updates to a vector* $\mathbf{x} \in [-M, M]^N$*, there is a deterministic* $O(k \cdot \text{polylog}(M, N))$*-space algorithm that, at the end of the stream, recovers* $\mathbf{x}$ *in polynomial time if* $\mathbf{x}$ *has at most* k *non-zero entries.*

▶ **Fact 8** (Undirected Graph Recovery [63]). *Given cut-query access to an undirected graph* G *on* n *nodes and* m *edges, there is a deterministic* $O(\min(m \log n, n^2))$*-query algorithm that recovers* G*.*

2.3 Problems

We formally define some problems considered in this paper. The input to each of them is a digraph $G = (V, E)$.

- s, t*-Reachability:* Given $s, t \in V$, is there a directed path from s to t?
- *Strong connectivity:* Does each node have a directed path to every other node?
- *SCC decomposition:* Partition of the vertex set V into $V_1, \ldots, V_\ell$ such that each V_i induces a strongly connected component (SCC), i.e., a maximal subgraph that is strongly connected.
- *Acyclicity testing:* Is there a directed cycle in G?
- *Topological sorting:* Given that G is a DAG, output an ordering of V such that for each edge $(u, v) \in E$, u appears (somewhere) before v in the ordering.
- *Minimum cut:* Find the size of the smallest directed cut, i.e., $\min_{U \subseteq V} \overleftarrow{\mathcal{C}}(U)$.
- *Minimum* s, t*-cut:* Find the size of the smallest directed cut whose removal makes s unreachable from t, i.e., $\min\{\overleftarrow{\mathcal{C}}(U) : U \subseteq V : s \in U, t \notin U\}$.
- For an ordering σ of the digraph nodes, the *width* of σ is defined as the maximum cut value over the prefixes of σ. The *cutwidth* of the digraph is the minimum width over all vertex orderings, i.e.,

$$\text{cutwidth}(G) := \min_{\sigma \in S_n} \left\{ \max_{\text{prefix } P \text{ of } \sigma} \overleftarrow{\mathcal{C}}(P) \right\}.$$

- *Optimal Linear Arrangement* (OLA) is a vertex ordering that minimizes the sum (rather than maximum) of the prefix-cut values, i.e.,

$$\text{OLA}(G) := \text{argmin}_{\sigma \in S_n} \left\{ \sum_{\text{prefix } P \text{ of } \sigma} \overleftarrow{\mathcal{C}}(P) \right\}.$$

  Equivalently, it is an ordering that minimizes the sum of the *stretches* of the induced back edges [8].
  For cutwidth and OLA, we refer to the problem versions that ask for the value and the ordering as their "value version" and "ordering version" respectively.

3 Graph Characterisation and Computation Based on Skeleton and Indegrees

Recall that $\mathcal{G}[\mathsf{S}, \mathbf{d}]$ is the class of all graphs with skeleton S and indegree sequence $\mathbf{d}$ and two graphs G_1 and G_2 are related, i.e., $G_1 \sim G_2$ iff both of them have the same skeleton and indegree sequence. We prove the following general lemma that implies our main theorems.

▶ **Lemma 9.**

(1) *For any two digraphs G_1 and G_2, we have $G_1 \sim G_2$ iff there is a set of cycles in G_1, flipping the directions of which yields G_2.*

(2) *Given an undirected graph S on n nodes and m edges and a sequence $\mathbf{d}$ of n non-negative integers, we have an $O(m^{1+o(1)})$-time deterministic algorithm that outputs a directed graph from $\mathcal{G}[\mathsf{S}, \mathbf{d}]$ if $\mathcal{G}[\mathsf{S}, \mathbf{d}]$ is non-empty, and otherwise outputs $\perp$.*

Characterisation. We first prove Lemma 9 (1), which characterises graphs based on skeleton and indegree sequence.

Proof of Lemma 9 (1). The "if" direction is straightforward. Given any $v \in V$, each cycle in G_1 has one edge outgoing from v and one edge incoming to v. Therefore, reversing a cycle preserves the indegree of vertex v. Also, reversing the direction of a cycle does not change the skeleton of the graph. Hence, if G_2 is obtained by reversing cycles in G_1, the graphs G_1 and G_2 must have the same skeleton and indegree sequence, i.e., $G_1 \sim G_2$.

For the "only if" direction, let us first fix some notation. For a subset of undirected edges H in the skeleton of a digraph G, let $H[G]$ denote the directed subgraph of G induced by H (i.e., edges in H oriented as in G).

Here, G_1 and G_2 have the same skeleton, say S. Let F be the set of (undirected) edges in S that have opposite directions in G_1 and G_2. Let $\overline{F} := \mathsf{S} \setminus F$. Fix any vertex $v \in V$. We have

$$d_{\text{in}}(v, G_2) = d_{\text{in}}(v, F[G_2]) + d_{\text{in}}(v, \overline{F}[G_2]) = d_{\text{out}}(v, F[G_1]) + d_{\text{in}}(v, \overline{F}[G_1]) \tag{1}$$

The second equality follows since precisely the edges in F have opposite directions in G_1 and G_2, and the edges in $\overline{F}$ have the same direction in G_1 and G_2. Again,

$$d_{\text{in}}(v, G_1) = d_{\text{in}}(v, F[G_1]) + d_{\text{in}}(v, \overline{F}[G_1]) \tag{2}$$

But since G_1 and G_2 have the same indegree sequence, $d_{\text{in}}(v, G_1) = d_{\text{in}}(v, G_2)$. Hence, from Equation (1) and Equation (2), we get $d_{\text{out}}(v, F[G_1]) = d_{\text{in}}(v, F[G_1])$. Since this is true for all $v \in V$, $F[G_1]$ is an Eulerian circuit, a collection of edge-disjoint cycles. We can obtain G_2 from G_1 by precisely reversing $F[G_1]$. ◀

Therefore, if a graph property is invariant under cycle reversal, then the skeleton and the indegree sequence suffice in determining it.

Completeness of characterisation. We note that if a property is *not* invariant under cycle reversal, then the skeleton and the degrees alone cannot decide the property. Assume to the contrary that there is a property $\mathcal{P}$ that is not invariant under cycle reversal, but there is an algorithm that decides it from only the skeleton and the indegree information. By definition of $\mathcal{P}$, there must exist graphs G and G' such that G' is obtained by reversing cycles of G, but $\mathcal{P}$ has different answers on G and G'. By Lemma 9 (1), $G \sim G'$. Therefore, they have the same skeleton, say S, and the same indegree sequence, say $\mathbf{d}$. Our algorithm outputs an answer that is a function of only S and $\mathbf{d}$. Thus, it outputs the same answer for $\mathcal{P}$ on both G and G', and hence, must be incorrect on one of them. This shows that the class of cycle-reversal-invariant properties *completely* characterises the set of properties determined by the skeleton and the indegrees.

▶ **Corollary 10.** *The set of well-defined properties for the equivalence under skeleton and indegree sequence is exactly the class of properties that are invariant under reversal of cycles.*

▶ Remark 11. The characterisation can be equivalently stated with the flipped cycles being "edge-disjoint." This is because the effect of reversing two overlapping cycles C_1 and C_2 with $C_1 \cap C_2 = F$ can be obtained by reversing the cycle $(C_1 \cup C_2) \setminus F$.

Efficient computation. Lemma 9 (1) only implies an exponential-time algorithm for any cycle-reversal-invariant problem when given the skeleton and indegree sequence of the input digraph: we brute force over all orientations of the given skeleton and find one that satisfies the given indegree sequence. The invariance guarantees that solving the problem on this digraph would give us the solution for the original digraph. Now we focus on generating such a feasible digraph efficiently in *polynomial time*.

Proof of Lemma 9 (2). We describe a polynomial-time algorithm to assign directions to edges in S so as to obtain a digraph $G' \in \mathcal{G}[\mathsf{S}, \mathbf{d}]$. Let $w(u, v)$ be the number of edges between u and v in S. If $w(u, v) = 2$, G' must have both $\overrightarrow{uv}$ and $\overrightarrow{vu}$. Therefore, we can assign direction to all such edges.

Ignore all edges whose directions have been marked. Therefore, for remaining edges $\{u, v\}$ in S, we have $w(u, v) = 0$ or 1. Now, we find a flow f which satisfies the following equations:

$$\forall \{u, v\} \in E(\mathsf{S}): \quad f(u, v) = -f(v, u) \tag{3}$$

$$\forall \{u, v\} \in E(\mathsf{S}): \quad |f(u, v)| \leq w(u, v) \qquad\qquad w(u, v) = 0 \text{ or } 1 \tag{4}$$

$$\forall u \in V: \quad \sum_{v \in N_{\mathsf{S}}(u)} f(u, v) = d_{out}(u) - d_{in}(u) \tag{5}$$

where $E(\mathsf{S})$ are the edges in S and $N_{\mathsf{S}}(u)$ is the neighbourhood of u in the graph S.

We check for a feasible flow satisfying these constraints. This is the well-studied *circulation problem with vertex demands* [66, 67, 37] and can be solved using any polynomial-time (s, t)-maxflow algorithm in the following way: let us define the demand of a vertex u to be $\mathsf{dem}(u) := d_{out}(u) - d_{in}(u)$. We create a source node s and a sink node t. For each node u with $\mathsf{dem}(u) > 0$, we put an edge from s to u with capacity $\mathsf{dem}(u)$. Similarly, for every node u with $\mathsf{dem}(u) < 0$, we put an edge from u to t with capacity $-\mathsf{dem}(u)$. Note that this augmentation makes the flow conservation constraints to be $\forall u \in V : \sum_{v \in N_{\mathsf{S}}(u)} f(u, v) = 0$, which means now we can run a (s, t)-max-flow algorithm on this augmented graph, and the corresponding flow on the original graph will maintain all original constraints. Using the recent breakthrough result of [68], we can solve it deterministically in $O\left(m^{1+o(1)}\right)$ time. If there is no feasible flow, then we return $\perp$, announcing that $\mathcal{G}[\mathsf{S}, \mathbf{d}] = \emptyset$. Observe that if $\mathcal{G}[\mathsf{S}, \mathbf{d}] \neq \emptyset$ and $G = (V, E) \in \mathcal{G}[\mathsf{S}, \mathbf{d}]$, then the following flow is feasible.

$$f(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E, \\ -1 & \text{if } (v, u) \in E, \end{cases}$$

Thus, we shall always find a feasible flow in this case. The contrapositive says that if we do not find a feasible flow, then $\mathcal{G}[\mathsf{S}, \mathbf{d}]$ must be empty. Hence, we return the correct answer in this case.

Suppose that we do find a feasible flow. Then we describe a strategy to orient the remaining edges to construct a graph G' that we output. As $w(u, v)$ and $\mathsf{dem}(u)$ are integers, we know that the flow f is also integral (therefore, $f(u, v) > 0 \implies f(u, v) = 1$). Now for all $u, v \in V$ such that $f(u, v) = 1$, we assign direction $\overrightarrow{uv}$ to edges uv in S. Consider the subgraph S' of S induced by the edges that have not been assigned a direction, i.e., the edges $\{u, v\} \in \mathsf{S}$ that have flow $f(u, v) = 0$.

Because the demand constraint of each node $v \in V$ has been satisfied, we know that $d_{\text{in}}(v, \mathsf{S}'[G']) = d_{\text{out}}(v, \mathsf{S}'[G'])$ for all $v \in V$. This is equivalent to the fact that the remaining edges in the graph G, i.e., the subgraph S' must form an Euler circuit. If S' is not an Euler tour, then no Euler circuit can be formed, implying that $\mathcal{G}[\mathsf{S}, \mathbf{d}] = \emptyset$.

We can now find a Euler tour in linear time [29] and orient the edges in S' along the direction of the tour. This ensures that for every vertex u, the in- and out-degree of u restricted to edges in S' is equal. Therefore, we have managed to assign direction to each edge in S without violating the indegree sequence $\mathbf{d}$. ◀

The proof of Lemma 9 is now complete. From Lemma 9 and Corollary 10, we have our main theorem.

▶ **Theorem 2.** *The class of graph problems determined by the skeleton and indegree sequence of a directed graph is precisely the class of problems whose answers are invariant under cycle reversal. Moreover, each problem can be determined in polynomial time from the skeleton and the indegree sequence (provided that it has a polynomial-time algorithm when the entire input graph is given).*

Applications to Cut Query. Since for any subset of nodes U of a digraph, the cut size $\overleftarrow{\mathcal{C}}(U)$ is invariant under cycle reversal, we have the following corollary of Theorem 2 that is significant from the perspective of cut-query algorithms.

▶ **Corollary 12.** *Given an undirected graph S and a sequence $\mathbf{d}$ of non-negative integers, the cut profiles of all graphs in $\mathcal{G}[\mathsf{S}, \mathbf{d}]$ are identical.*

Although it follows immediately from Theorem 2, we give an independent direct and simple proof of the statement.

Proof. If $\mathcal{G}[\mathsf{S}, \mathbf{d}] = \emptyset$, then this is vacuously true. Hence, we assume that $\mathcal{G}[\mathsf{S}, \mathbf{d}] \neq \emptyset$. Let G be any graph in $\mathcal{G}[\mathsf{S}, \mathbf{d}]$. For any $T \subseteq V$, let $\mathcal{C}_{\mathsf{S}}(T)$ denote the undirected cut value of T in S. Observe that

$$\overrightarrow{\mathcal{C}}_G(T) + \overleftarrow{\mathcal{C}}_G(T) = \mathcal{C}_{\mathsf{S}}(T), \tag{6}$$

$$\overrightarrow{\mathcal{C}}_G(T) - \overleftarrow{\mathcal{C}}_G(T) = \sum_{v \in T} (d_{\text{out}}(v) - d_{\text{in}}(v)). \tag{7}$$

Equation (7) follows since each edge (u, w) inside T (i.e., both $u, w \in T$) contributes to both $d_{\text{out}}(u)$ and $d_{\text{in}}(w)$ and cancels. Now, for each $T \subseteq V$, the values $\overleftarrow{\mathcal{C}}_G(T)$ and $\overrightarrow{\mathcal{C}}_G(T)$ are determined by the linearly independent equations (6) and (7). The RHS of Equations (6) and (7) are completely determined by S and $\mathbf{d}$. Hence, for fixed S and $\mathbf{d}$, $\overleftarrow{\mathcal{C}}_G(T)$ and $\overrightarrow{\mathcal{C}}_G(T)$ are identical for all $G \in \mathcal{G}[\mathsf{S}, \mathbf{d}]$. ◀

We restate Corollary 6 that captures an important implication as discussed in Section 1.

▶ **Corollary 6.** *For any directed graph, global- and s,t-minimum cut can be solved with $O(n^2)$ directed cut queries and polynomial runtime.*

Proof. For an input digraph G, we find the indegree profile $\mathbf{d}$ using n queries. We recover its skeleton S using $O(n^2)$ queries (Fact 8). Then using Lemma 9 (2), we generate $G' \sim G$ in polynomial time and then find the global- or s,t-min-cut of G' using any polynomial-time classical algorithm for the problem. By Corollary 12, the answer for G must be the same. ◀

4 Applications to Algorithms for Tournaments and Almost Tournaments

Recall that a *tournament* graph is one where each pair of vertices shares exactly one directed edge. In other words, a tournament is an oriented complete graph. The indegree sequence of a tournament is also called its *score sequence*. We obtain the following characterisation of tournaments based on their score sequences.

▶ **Corollary 13** (score sequence characterisation)**.** *Two tournaments T and T' have the same indegree sequence iff T can be obtained from T' by only reversing a set of cycles of T.*

Proof. Since T and T' have the same skeleton (the complete graph), the statement immediately follows from Lemma 9 (1). ◀

Again, we have the following implication of Theorem 2.

▶ **Theorem 1.** *The class of graph problems determined by the indegree sequence of a tournament is precisely the class of problems whose answers are invariant under cycle reversal. Moreover, each problem can be determined in polynomial time from the indegree sequence (provided that it has a polynomial-time algorithm when the entire input graph is given).*

Theorem 1 now implies our algorithmic results for tournaments in various computational models.

▶ **Theorem 3.** *Given any digraph problem $\mathcal{P}$ whose answer is invariant under cycle reversal, there is a single-pass $O(n \log n)$-space streaming algorithm, an $O(n)$-cut-query algorithm, and an $O(n \log n)$-communication protocol for the problem $\mathcal{P}$ on tournaments. Each algorithm runs in polynomial-time, provided $\mathcal{P}$ has a polynomial-time classical algorithm on tournaments.*

Proof. Given an input tournament, we can find its indegree sequence in the cut-query model using n singleton cut queries; in the streaming model, indegree count takes $O(n \log n)$ space and a single pass; in the communication model, Alice can send Bob $O(n \log n)$ bits that encode the indegree sequence induced by her edges E_A, from which Bob can compute the indegree sequence induced by all edges $E_A \sqcup E_B$. Then, by Theorem 1, we can determine the property $\mathcal{P}$ in polynomial time. ◀

▶ **Corollary 4.** *There is a single-pass $O(n \log n)$-space streaming algorithm, an $O(n)$-cut-query algorithm, and an $O(n \log n)$-communication protocol for the following problems on tournaments, each of which runs in polynomial time.*

- **(i)** *Connectivity-based problems: s,t-reachability, strong connectivity, SCC decomposition (or more generally finding the SCC-graph);*

(ii) *DAG-based problems: acyclicity testing, topological sorting;*
(iii) *Vertex ordering problems: optimal linear arrangement (OLA), cutwidth;*
(iv) *Directed cut problems: minimum cut, minimum s,t-cut.*

All these bounds are optimal (up to a $\log n$ factor).

Proof. It is easy to see that reachability, strong connectivity, or in general, any strongly connected component of a graph does not change when cycles in the graph are flipped. Since a DAG has no cycle, the DAG-based properties indeed remain intact. For the vertex ordering problems and cut problems, we note that they are all certain functions of the cuts of a graph. *All* cuts in the graph are preserved under cycle reversal, and hence all these properties are, in turn, preserved. Therefore, all these problems are invariant under cycle reversal, and the results follow from Theorem 3.

The optimality of the bounds up to a factor of $\log n$ in each model–including streaming and cut query – follows from $\Omega(n)$ communication complexity of the problems. By standard streaming-to-communication simulations, an S-bit communication lower bound for a problem implies an S-space single-pass lower bound for the same problem in the streaming model. Again, an $\Omega(n/\log n)$ cut-query lower bound for each problem follows from its $\Omega(n)$ communication lower bound: a C-cut-query algorithm for the problem implies an $O(C \log n)$-communication protocol for it, since the players can communicate to each other the answer for each query using $O(\log n)$ bits.

We now justify the $\Omega(n)$ communication lower bound for each problem. [35] showed such lower bounds for the connectivity-based and the DAG-based problems. The lower bounds on the value versions of the vertex-ordering problems follow via trivial reductions from the acyclicity problem: the cutwidth or the OLA-value of a tournament is zero if and only if it is acyclic. Along similar lines, the lower bounds for their ordering versions follow from the topological ordering problem: the cutwidth-ordering and OLA of an acyclic tournament is also a valid topological ordering of the tournament.

Again, the lower bounds for global- and s,t-minimum-cut follow via easy reductions from the strong connectivity and s,t-reachability problems respectively: observe that in any digraph, the minimum cut value is non-zero if and only if the graph is strongly connected. Again, the minimum s,t-cut value is non-zero if and only if s is reachable from t. ◀

Define a digraph to be *k-close to tournament* if it is obtained by deleting at most k edges from a tournament. [35] showed that these graphs are some generalizations of tournaments that still admit sublinear-space streaming algorithms for connectivity-based problems. Our results imply a much more general statement.

▶ **Corollary 5.** *Given any digraph problem $\mathcal{P}$ whose answer is invariant under cycle reversal, there is a single-pass $\widetilde{O}(n+k)$-space streaming algorithm, an $\widetilde{O}(n+k)$-cut-query algorithm, and an $\widetilde{O}(n+k)$-communication protocol for $\mathcal{P}$ on digraphs that are k-close to tournaments. Each algorithm runs in polynomial-time, provided $\mathcal{P}$ has a polynomial-time classical algorithm on such digraphs.*

Proof. For a digraph that is k-close to tournament, its k non-edges can be recovered deterministically in the single-pass streaming model using $O(k \cdot \text{polylog } n)$ space via sparse recovery (Fact 7), and the degree information takes an additional $O(n \log n)$ space. This streaming algorithm can be easily simulated in the communication model along standard lines: Alice runs the algorithm on her edges and sends Bob the algorithm's memory state at the end of her stream. Given this state, Bob continues the stream with his edges and finds the solution for the entire graph at the end of the stream. The total communication is at most the space usage of the algorithm, which is $\widetilde{O}(n+k)$.

Finally, in cut-query model, we recover the complement $\bar{G}$ of the input graph. Note that $\bar{G}$ is an undirected graph on at most k edges. For each $U \subseteq V$, we need 2 queries to the directed cut oracle on G to recover the undirected cut $\mathcal{C}_{\bar{G}}(U)$ since $\mathcal{C}_{\bar{G}}(U) = |V \setminus U| - \overleftarrow{\mathcal{C}}_G(U) - \overleftarrow{\mathcal{C}}_G(\bar{U})$. Thus, by Fact 8, we need $O(k \log n)$ directed cut queries to recover $\bar{G}$. Once we do that, we know the skeleton of G (which is the undirected complement of $\bar{G}$). The indegree sequence is obtained by n additional queries. The results then follows from Theorem 2. ◀

This gives us an improvement over a prior result of [35]. Previously, we only knew $\widetilde{O}(2^{\min(k,n)})$-time algorithms for reachability and $\widetilde{O}(2^n)$-time algorithms for SCC decomposition and constructing the SCC-graph. This is because they go over every possible "completion" of the almost tournament and on each completion, call a blackbox for tournaments. Our results improve the runtime to polynomial time (in fact, almost-linear $O(m^{1+o(1)})$ time).

5 Connections Between the Streaming and the Cut Query Models

Finally, we observe the following interesting connection between the semi-streaming and cut-query models with respect to tournament problems.

▶ **Lemma 14.** *If a tournament problem $\mathcal{P}$ is decidable/solvable by cut queries (regardless of the number of queries), then it has a single-pass $O(n \log n)$-space algorithm. More generally, if a problem $\mathcal{P}$ on a digraph that is k-close to tournament, is solvable by cut queries, then it has a single-pass $\widetilde{O}(n+k)$-space streaming algorithm.*

Proof. Given a tournament T, we can store its indegrees in a single-pass using $O(n \log n)$ space. By Lemma 9 (2), this would let us generate in polynomial time a tournament T' in the same equivalence class as T. Then Corollary 12 says that T' has the same cut profile as T, and so we can now access all cut queries on T for free.

The "more general" statement follows along similar lines since we can recover the skeleton of a digraph that is k-close to tournament in $\widetilde{O}(n+k)$ space (as in the proof of Corollary 5). ◀

The contrapositive is also interesting: stronger than semi-streaming lower bounds for a tournament problem would imply its undecidability in the directed cut query model.

A separation between semi-streaming and cut queries. By Lemma 14, the single-pass semi-streaming model is at least as strong as the cut query model for tournament problems. Can we show that it is *strictly* stronger? That is, can we exhibit a tournament problem that is solvable in single-pass semi-streaming but not in the cut query model? The answer is yes, as long as the model allows randomness.

The FAST problem asks to find the minimum Feedback Arc Set in a Tournament, a set of arcs whose removal deletes all cycles. We are interested in the FAST-size.

▶ **Proposition 15.** *The problem of finding a $(1+\varepsilon)$-approximation to FAST-size has a single-pass semi-streaming (randomized) algorithm, but even a $o(n)$-approximation is not possible using cut queries.*

Proof. The single-pass semi-streaming algorithm was given by [13].

For the impossibility, let T be the tournament on vertex set $[n]$ defined as follows. Add the edge $n \to 1$. For remaining pairs $(i,j) \in [n] \times [n]$ with $i < j$, add the edge $i \to j$. Let T' be the tournament obtained by reversing the cycle $1 \to 2 \to \ldots \to n \to 1$ in T. Hence, T and T' have the same cut profile and any cut-query algorithm must output the same answer for both of them.

In T, the FAST-size is 1 (deleting the edge $n \to 1$ makes it acyclic). But in T', the FAST-size $= \Omega(n)$ since T' has $\Omega(n)$ edge-disjoint triangles: for each odd $i \leq n-2$, we have the triangle $i \to i+2 \to i+1 \to i$. Hence, α-approximate FAST-size for $\alpha = o(n)$ is undecidable by cut queries. ◀

6 Conclusions & Future Directions

In this paper, we characterised the class of all digraph problems decidable by only its skeleton and indegrees, and consequently the class of all tournament problems decidable by only its score sequence. We also developed a polynomial-time algorithm to generate a feasible digraph satisfying a given skeleton and indegree sequence. We then exhibited a variety of applications of these results in several domains such as streaming algorithms, cut-query algorithms, and communication complexity. The complete characterisation should be influential in future research on these properties, from either an algorithmic or a combinatorial point of view.

One of the immediate future directions is to show that directed minimum cut can be solved by $O(n)$ cut queries or, towards hardness, requires $\Omega(n^2)$ queries. Any of these results will be a breakthrough as directed minimum cut is a great candidate for proving a superlinear SFM lower bound – an outstanding question that has remained open for the last five decades [17, 11, 41, 16]. Because cut queries cannot fully specify edge directions, we conjecture that any improvement of the form $O(n^{2-\epsilon})$ will require designing a minimum cut algorithm that does not crucially use the individual edge directions. This will be a departure from the standard combinatorial ways of solving this problem and will be of interest.

Another intriguing future direction is designing streaming algorithms for tournament problems that are *not* invariant under cycle reversal. We showed that one can obtain the entire cut profile of a tournament in semi-streaming space. Combining this strong tool with other useful sketching and sampling techniques known in the streaming literature, can we resolve the streaming complexities of important tournament problems that remain open? A couple of these problems are (approximate) feedback arc set in tournaments (FAST) and finding Hamiltonian paths and cycles. For the former problem, we showed a separation between cut-query and semi-streaming algorithms: $o(n)$-approximation is undecidable by directed cut queries but a $(1+\varepsilon)$-approximate semi-streaming algorithm exists. However, the said algorithm by [13] is exponential time. The best-known polytime single-pass semi-streaming algorithm achieves a 5-approximation simply by sorting with respect to indegrees [22]. Can we improve upon this approximation factor in polytime and a single pass?

Finally, we remark that the complexity of each digraph problem studied in this paper is unsettled for general digraphs in the multipass streaming model. In fact, even after a series of works [40, 13, 6, 20, 5] on reachability, there is an exponential gap in the pass-complexity of semi-streaming algorithms for the problem, and it is one of the biggest open problems in multipass streaming (see the survey [2]). Our theorem for arbitrary digraphs provides new insights on the problem, and coupled with sophisticated ideas, might lead to improved multipass algorithms for s, t-reachability and related problems on general digraphs.

References

1 Simon Apers, Yuval Efron, Pawel Gawrychowski, Troy Lee, Sagnik Mukhopadhyay, and Danupon Nanongkai. Cut query algorithms with star contraction. In *FOCS*, pages 507–518. IEEE, 2022. doi:10.1109/FOCS54457.2022.00055.

2 Sepehr Assadi. Recent advances in multi-pass graph streaming lower bounds. *ACM SIGACT News*, 54(3):48–75, September 2023. doi:10.1145/3623800.3623808.

3 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Polynomial pass lower bounds for graph streaming algorithms. In *STOC*, pages 265–276. ACM, 2019. doi:10.1145/3313276.3316361.

4 Sepehr Assadi and Aditi Dudeja. A simple semi-streaming algorithm for global minimum cuts. In *4th Symposium on Simplicity in Algorithms, SOSA 2021, Virtual Conference, January 11-12, 2021*, pages 172–180. SIAM, 2021. doi:10.1137/1.9781611976496.19.

5 Sepehr Assadi, Arun Jambulapati, Yujia Jin, Aaron Sidford, and Kevin Tian. Semi-streaming bipartite matching in fewer passes and optimal space. In *SODA*, pages 627–669. SIAM, 2022. doi:10.1137/1.9781611977073.29.

6 Sepehr Assadi and Ran Raz. Near-quadratic lower bounds for two-pass graph streaming algorithms. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 342–353. IEEE, 2020. doi:10.1109/FOCS46700.2020.00040.

7 Brian Axelrod, Yang P. Liu, and Aaron Sidford. Near-optimal approximate discrete and continuous submodular function minimization. In *SODA*, pages 837–853. SIAM, 2020. doi:10.1137/1.9781611975994.51.

8 Florian Barbero, Christophe Paul, and Michal Pilipczuk. Exploring the complexity of layout parameters in tournaments and semi-complete digraphs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 70:1–70:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.70.

9 Anubhav Baweja, Justin Jia, and David P. Woodruff. An efficient semi-streaming PTAS for tournament feedback arc set with few passes. In *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 16:1–16:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.16.

10 Matthias Bentert, Fedor V. Fomin, Tanmay Inamdar, and Saket Saurabh. Exponential-Time Approximation (Schemes) for Vertex-Ordering Problems. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, volume 325 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2025.15.

11 Joakim Blikstad. Breaking o(nr) for matroid intersection. In *ICALP*, volume 198 of *LIPIcs*, pages 31:1–31:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.31.

12 Joakim Blikstad, Jan van den Brand, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Nearly optimal communication and query complexity of bipartite matching. In *FOCS*, pages 1174–1185. IEEE, 2022. doi:10.1109/FOCS54457.2022.00113.

13 Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1786–1802. SIAM, 2020. doi:10.1137/1.9781611975994.109.

14 Deeparnab Chakrabarty, Yu Chen, and Sanjeev Khanna. Notes on directed cut oracle. *Unpublished*, 2022.

15 Deeparnab Chakrabarty, Andrei Graur, Haotian Jiang, and Aaron Sidford. Improved lower bounds for submodular function minimization. In *FOCS*, pages 245–254. IEEE, 2022. doi:10.1109/FOCS54457.2022.00030.

16 Deeparnab Chakrabarty, Andrei Graur, Haotian Jiang, and Aaron Sidford. Parallel submodular function minimization. In *FOCS*. IEEE, 2023.

17 Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, Sahil Singla, and Sam Chiu-wai Wong. Faster matroid intersection. In *FOCS*, pages 1146–1168. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00072.

18 Deeparnab Chakrabarty and Hang Liao. A query algorithm for learning a spanning forest in weighted undirected graphs. In *ALT*, volume 201 of *Proceedings of Machine Learning Research*, pages 259–274. PMLR, 2023. URL: https://proceedings.mlr.press/v201/chakrabarty23a.html.

19 Ho-Lin Chen, Tsun Ming Cheung, Peng-Ting Lin, and Meng-Tsung Tsai. Independence-number parameterized space complexity for directed connectivity certificate. *CoRR*, abs/2602.12668, 2026. doi:10.48550/arXiv.2602.12668.

20 Lijie Chen, Gillat Kol, Dmitry Paramonov, Raghuvansh R. Saxena, Zhao Song, and Huacheng Yu. Almost optimal super-constant-pass streaming lower bounds for reachability. In *STOC*, pages 570–583. ACM, 2021. doi:10.1145/3406325.3451038.

21 Anders Claesson, Mark Dukes, Atli Franklín, and Sigurður Stefánsson. Counting tournament score sequences. In *Proceedings of the American Mathematical Society*, pages 290–297, January 2023. doi:10.5817/CZ.MUNI.EUROCOMB23-040.

22 Don Coppersmith, Lisa Fleischer, and Atri Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM Trans. Algorithms*, 6(3):55:1–55:13, 2010. doi:10.1145/1798596.1798608.

23 William H. Cunningham. On submodular function minimization. *Comb.*, 5(3):185–192, 1985. doi:10.1007/BF02579361.

24 Abhik Kumar Das and Sriram Vishwanath. On finite alphabet compressive sensing. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5890–5894, 2013. doi:10.1109/ICASSP.2013.6638794.

25 Matt DeVos and Kathryn Nurse. A maximum linear arrangement problem on directed graphs, 2025. arXiv:1810.12277.

26 Matthew Ding, Alexandro Garces, Jason Li, Honghao Lin, Jelani Nelson, Vihan Shah, and David P. Woodruff. Space Complexity of Minimum Cut Problems in Single-Pass Streams. In *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, volume 325 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:23, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ITCS.2025.43.

27 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2–3):207–216, 2005. Preliminary version in *Proc. 31st International Colloquium on Automata, Languages and Programming*, pages 531–543, 2004. doi:10.1016/J.TCS.2005.09.013.

28 Lisa Fleischer and Satoru Iwata. A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discret. Appl. Math.*, 131(2):311–322, 2003. doi:10.1016/S0166-218X(02)00458-4.

29 H. Fleischner. *Eulerian Graphs and Related Topics*. Number pt. 1, v. 2 in Annals of discrete mathematics. North-Holland, 1990. URL: https://books.google.com/books?id=6Df1xQEACAAJ.

30 Fedor V. Fomin and Michał Pilipczuk. Subexponential parameterized algorithm for computing the cutwidth of a semi-complete digraph. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms – ESA 2013*, pages 505–516, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

31 Alexandra Ovetsky Fradkin. *Forbidden structures and algorithms in graphs and digraphs*. PhD thesis, Princeton University, 2011.

32 András Frank and András Gyárfás. How to orient the edges of a graph? *Combinatorics*, 18:353–364, 1978.

33 S I Gass. Tournaments, transitivity and pairwise comparison matrices. *Journal of the Operational Research Society*, 49(6):616–624, 1998. doi:10.1057/palgrave.jors.2600572.

34 Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, and Heung-Yeung Shum. Query dependent ranking using k-nearest neighbor. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 115–122. Association for Computing Machinery, 2008. doi:10.1145/1390334.1390356.

35 Prantar Ghosh and Sahil Kuchlous. New algorithms and lower bounds for streaming tournaments. In *32nd Annual European Symposium on Algorithms, ESA 2024, September 2-4, 2024, Royal Holloway, London, United Kingdom*, volume 308 of *LIPIcs*, pages 60:1–60:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ESA.2024.60.

36 Anna C. Gilbert and Piotr Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010. doi:10.1109/JPROC.2010.2045092.

37 Andrew V. Goldberg and Robert E. Tarjan. Finding minimum-cost circulations by successive approximation. *Math. Oper. Res.*, 15(3):430–466, 1990. doi:10.1287/MOOR.15.3.430.

38 Andrei Graur, Tristan Pollner, Vidhya Ramaswamy, and S. Matthew Weinberg. New query lower bounds for submodular function minimization. In *ITCS*, volume 151 of *LIPIcs*, pages 64:1–64:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ITCS.2020.64.

39 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988. doi:10.1007/978-3-642-97881-4.

40 Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 287–298. IEEE Computer Society, 2013. doi:10.1109/CCC.2013.37.

41 Nicholas J. A. Harvey. Matroid intersection, pointer chasing, and young's seminormal representation of S_n. In *SODA*, pages 542–549. SIAM, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347142.

42 Satoru Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM J. Comput.*, 32(4):833–840, 2003. doi:10.1137/S0097539701397813.

43 Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, 2001. doi:10.1145/502090.502096.

44 Satoru Iwata and James B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *SODA*, pages 1230–1237. SIAM, 2009. doi:10.1137/1.9781611973068.133.

45 Haotian Jiang. Minimizing convex functions with integral minimizers. In *SODA*, pages 976–985. SIAM, 2021. doi:10.1137/1.9781611976465.61.

46 Haotian Jiang. Minimizing convex functions with rational minimizers. *J. ACM*, 70(1):5:1–5:27, 2023. doi:10.1145/3566050.

47 Yonggang Jiang, Danupon Nanongkai, and Pachara Sawettamalya. Minimum s–t cuts with fewer cut queries. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*, pages 258–296. SIAM, 2026. doi:10.1137/1.9781611978971.12.

48 Yotam Kenneth-Mordoch and Robert Krauthgamer. Cut-query algorithms with few rounds. In *33rd Annual European Symposium on Algorithms, ESA 2025, Warsaw, Poland, September 15-17, 2025*, volume 351 of *LIPIcs*, pages 100:1–100:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ESA.2025.100.

49 Yotam Kenneth-Mordoch and Robert Krauthgamer. All-pairs minimum cut using $\widetilde{O}(n^{7/4})$ cut queries. In Kasper Green Larsen and Barna Saha, editors, *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2026, Vancouver, BC, Canada, January 11-14, 2026*, pages 4077–4095. SIAM, 2026. doi:10.1137/1.9781611978971.150.

50 Yotam Kenneth-Mordoch and Robert Krauthgamer. Faster all-pairs minimum cut: Bypassing exact max-flow. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing, STOC 2026, Salt Lake City, UT, USA, June 22-26, 2026*, pages 1254–1265. ACM, 2026. doi:10.1145/3798129.3800836.

51 Daniel J. Kleitman and Kenneth J. Winston. Forests and score vectors. *Combinatorica*, 1:49–54, 1981. doi:10.1007/BF02579176.

52 Luigi Laura and Federico Santaroni. Computing strongly connected components in the streaming model. In *Theory and Practice of Algorithms in (Computer) Systems*, pages 193–205. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-19754-3_20.

53 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *FOCS*, pages 1049–1065. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.68.

54 László Lovász. Submodular functions and convexity. In *ISMP*, pages 235–257. Springer, 1982. doi:10.1007/978-3-642-68874-4_10.

55 Nikhil S. Mande, Manaswi Paraashar, Swagato Sanyal, and Nitin Saurabh. On the communication complexity of finding a king in a tournament. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, August 28-30, 2024, London School of Economics, London, UK*, volume 317 of *LIPIcs*, pages 64:1–64:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.APPROX/RANDOM.2024.64.

56 John W. Moon. *Topics on tournaments*. Athena series; selected topics in mathematics. Holt, Rinehart and Winston, 1968.

57 J.W. Moon. On the score sequence of an n-partite tournament. *Canadian Mathematical Bulletin*, 5(1):51–58, 1962. doi:10.4153/CMB-1962-008-9.

58 Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: sequential, cut-query, and streaming algorithms. In *STOC*, pages 496–509. ACM, 2020. doi:10.1145/3357713.3384334.

59 T.V. Narayana and D.H. Bent. Computation of the number of score sequences in round-robin tournaments. *Canadian Mathematical Bulletin*, 7(1):133–136, 1964. doi:10.4153/CMB-1964-015-1.

60 James B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Math. Program.*, 118(2):237–251, 2009. doi:10.1007/S10107-007-0189-2.

61 John Riordan. The number of score sequences in tournaments. *Journal of Combinatorial Theory*, 5(1):87–89, 1968. doi:10.1016/S0021-9800(68)80032-8.

62 Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. Combining outputs from multiple machine translation systems. In Candace Sidner, Tanja Schultz, Matthew Stone, and ChengXiang Zhai, editors, *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 228–235, Rochester, New York, April 2007. Association for Computational Linguistics. URL: https://aclanthology.org/N07-1029/.

63 Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In *ITCS*, volume 94 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.39.

64 Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory, Ser. B*, 80(2):346–355, 2000. doi:10.1006/JCTB.2000.1989.

65 Paul K. Stockmeyer. Counting various classes of tournament score sequences, 2023. arXiv:2202.05238.

66 Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Comb.*, 5(3):247–256, 1985. doi:10.1007/BF02579369.

67 Robert E. Tarjan. Efficiency of the primal network simplex algorithm for the minimum-cost circulation problem. *Math. Oper. Res.*, 16(2):272–291, 1991. doi:10.1287/MOOR.16.2.272.

68 Jan van den Brand, Li Chen, Richard Peng, Rasmus Kyng, Yang P. Liu, Maximilian Probst Gutenberg, Sushant Sachdeva, and Aaron Sidford. A deterministic almost-linear time algorithm for minimum-cost flow. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 503–514. IEEE, 2023. doi:10.1109/FOCS57990.2023.00037.

69 Jens Vygen. A note on schrijver's submodular function minimization algorithm. *J. Comb. Theory, Ser. B*, 88(2):399–402, 2003. doi:10.1016/S0095-8956(02)00047-3.

70 Andrew C. Yao. Some complexity questions related to distributive computing. In *Proc. 11th Annual ACM Symposium on the Theory of Computing*, pages 209–213, 1979.

71 Mariano Zelke. Intractability of min- and max-cut in streaming graphs. *Information Processing Letters*, 111(3):145–150, 2011. doi:10.1016/J.IPL.2010.10.017.