List rankings and on-line list rankings of graphs

Daniel C. McDonald

Department of Mathematics, University of Illinois, Urbana, IL, USA

Abstract

A k-ranking of a graph G is a labeling of its vertices from \( \{1, \ldots, k\} \) such that any path on at least two vertices whose endpoints have the same label contains a larger label. The least \( k \) for which \( G \) has a \( k \)-ranking is the ranking number of \( G \), also known as tree-depth. The list ranking number of \( G \) is the least \( k \) such that if each vertex of \( G \) is assigned a set of \( k \) potential labels, then \( G \) can be ranked by labeling each vertex with a label from its assigned list. Rankings model a certain parallel processing problem in manufacturing, while the list ranking version adds scheduling constraints. We compute the list ranking number of paths, cycles, and trees with many more leaves than internal vertices. Some of these results follow from stronger theorems we prove about on-line versions of list ranking, where each vertex starts with an empty list having some fixed capacity, and potential labels are presented one by one, at which time they are added to the lists of certain vertices; the decision of which of these vertices are actually to be ranked with that label must be made immediately.

Keywords: List ranking, on-line list ranking, vertex ranking, ranking number, tree-depth

2000 MSC: 05C05, 05C15, 05C38, 05C78

1. Introduction

We consider a special type of proper vertex coloring using positive integers, called “ranking.” As with proper colorings, there exist variations on the original ranking problem. In this paper we consider the list ranking problem,
first posed by Jamison in 2003 [10]. For a graph $G$, we denote its vertex set by $V(G)$ and its edge set by $E(G)$. For $S \subseteq V(G)$ we let $G - S$ denote the subgraph of $G$ obtained by deleting all vertices in $S$ as well as any edges incident to at least one vertex in $S$, and we call $G - (V(G) - S)$ the subgraph of $G$ induced by $S$. We also set $\mathbb{N} = \{0, 1, 2, \ldots\}$.

**Definition 1.1.** Let $G$ be a finite simple graph, and let $f : V(G) \to \mathbb{N}$. An $f$-ranking $\alpha$ of $G$ labels each $v \in V(G)$ with an element of $\{1, \ldots, f(v)\}$ in such a way that if $u \neq v$ but $\alpha(u) = \alpha(v)$, then every $u, v$-path contains a vertex $w$ satisfying $\alpha(w) > \alpha(u)$. For a positive integer $k$, a $k$-ranking of $G$ is an $f$-ranking for the constant function $f = k$. The ranking number of $G$, denoted here by $\rho(G)$ (though in the literature often as $\chi_r(G)$), is the minimum $k$ such that $G$ has a $k$-ranking.

See Figure 1 for an example of a ranking. Note that $\alpha$ is a ranking of $G$ if and only if every path contains a unique vertex with largest label, or, equivalently, for $j \geq 1$ each component of $G - \{v : \alpha(v) > j\}$ contains a unique vertex with largest label. Since each edge in $G$ is a path with no internal vertices, adjacent vertices in $G$ receive distinct colors in any vertex ranking of $G$, so every vertex ranking of $G$ is also a proper coloring of $G$. Hence $\rho(G) \geq \chi(G)$, where $\chi(G)$ denotes the chromatic number of $G$, the least number of colors needed to properly color $G$.

![Figure 1: A 3-ranking of a graph $G$.](image)

Vertex rankings of graphs were introduced in [7], and results through 2003 are surveyed in [10]. Their study was motivated by applications to VLSI layout, cellular networks, Cholesky factorization, parallel processing, and computational geometry. Vertex rankings are sometimes called ordered colorings, and the ranking number of a graph is trivially equal to its “tree-depth,” a term introduced by Nešetřil and Ossona de Mendez in 2006 [15] in developing their theory of graph classes having bounded expansion, which has been the topic of much further study [12, 13, 14, 16].

In general, rankings are used to design efficient divide-and-conquer strategies for minimizing the time needed to perform interrelated tasks in parallel
The most basic example concerns a complex product being assembled in stages from its individual parts, where each stage of construction consists of individual parts being attached to previously assembled components in such a way that no component ever has more than one new part. Here, the complex product is represented by the graph $G$ whose vertices are the individual parts and whose edges are the connections between those parts; assuming all parts require the same amount of time to be installed, the fewest number of stages needed to complete construction is $\rho(G)$, achieved by finding some $\rho(G)$-ranking $\alpha$ of $G$ and installing each part $v$ in stage $\alpha(v)$. Similarly, rankings can be used to optimize the disassembly of a product into parts, where each stage of deconstruction consists of individual parts being detached from remaining components in such a way that no component loses multiple parts at the same time; $G$ can be disassembled in $\rho(G)$ stages by removing each part $v$ in stage $\rho(G) - \alpha(v) + 1$.

A ranking of a graph $G$ can also be viewed as a (successful) search strategy [5], wherein stationary searchers and an agile fugitive occupy vertices of $G$. The searchers place themselves one by one onto the vertices of $G$, where they remain permanently, until one searcher is placed on the vertex currently occupied by the fugitive, with the placement of each new searcher operating under the following procedure. First, with the previously placed searchers permanently occupying some set $S \subset V(G)$, the location of the fugitive is revealed as some vertex $v \in V(G) - S$. The new searcher then announces some vertex $u \in V(G) - S$ she will occupy (for the rest of the search), but before the new searcher can occupy $u$, the fugitive is allowed to move any distance along any path in $G - S$ (so the new searcher is placed only with the knowledge that the fugitive must stay in the same component $G - S$). Given a $k$-ranking of $G$, $k$ searchers can always catch the fugitive by placing a new searcher on the highest ranked vertex in the component of $G - S$ containing the fugitive; conversely, $k$ searchers suffice to guarantee capture of the fugitive only if $G$ is $k$-rankable.

The vertex ranking problem has spawned multiple variations, including edge ranking [4, 9], on-line ranking [2, 6, 11], and list ranking, introduced in [10] and studied here. The list ranking problem is to vertex ranking as the list coloring problem is to ordinary vertex coloring.

**Definition 1.2.** A function $L$ that assigns each vertex of $G$ a finite set of positive integers is an $f$-list assignment for $G$ if $|L(v)| = f(v)$ for each $v \in V(G)$. If $|L(v)| = k$ for all $v$, then $L$ is $k$-uniform. An $L$-ranking of $G$ is
a ranking $\alpha$ such that $\alpha(v) \in L(v)$ for each $v$. Say that $G$ is $f$-list rankable if $G$ has an $L$-ranking whenever $L$ is an $f$-list assignment for $G$, and say that $G$ is $k$-list rankable if $G$ is $f$-list rankable for $f = k$. Let the list ranking number of $G$, denoted $\rho_l(G)$, be the least $k$ such that $G$ is $k$-list rankable.

Note that $G$ is $f$-rankable if $G$ is $f$-list rankable, since an $f$-ranking is just an $L$-ranking if $L$ is the $f$-list assignment defined by $L(v) = \{1, \ldots, f(v)\}$ for each $v \in V(G)$. Furthermore, $\rho_l(G) \geq \chi_l(G)$, where $\chi_l(G)$ denotes the list chromatic number of $G$, the smallest $k$ such that $G$ can always be properly colored from a list assignment $L$ giving each vertex a set of $k$ potential colors. In terms of our application of assembling a product from parts in stages, obtaining an $L$-ranking corresponds to finding a feasible schedule for assembling the product when predetermined scheduling constraints limit each individual part $v$ to being attached during a stage listed in $L(v)$. In terms of our searching problem, obtaining an $L$-ranking corresponds to constructing a strategy for the searchers guaranteed to lead to the capture of the fugitive when predetermined scheduling constraints allow a vertex $v$ to accept a new searcher at time $t$ only if $t = m - j + 1$ for some $j \in L(v)$, where $m$ is the maximum label appearing in any list assigned by $L$.

The complexity of determining whether a given list assignment admits a ranking of a graph was considered by Dereniowski in 2008 [3], who produced a polynomial time reduction from the set cover decision problem (given a set $S$, $j$ of its subsets, and a positive integer $t$, is $S$ the union of $t$ of these subsets?) to the list ranking decision problem for certain graphs. This reduction showed the list ranking decision problem to be NP-complete for several classes of trees and their line graphs, including full binary trees. It was also shown that, given a list assignment $L$ for a path, the problem of finding an $L$-ranking minimizing the maximum label used, or determining that no $L$-ranking exists, is polynomially solvable.

In Section 2 we introduce three on-line versions of list ranking, which relate to list ranking similarly to the way on-line list coloring (also known as paintability) relates to list coloring [17]. We shall define these on-line versions of list ranking as games between adversaries Taxer and Ranker to be played on a predetermined graph $G$. At the beginning of the game each vertex is assigned a size for its list of potential labels, but no actual labels. Taxer in effect fills out these lists in real time by iterating through the possible labels one by one, stipulating which vertices have the given label in their lists (the order in which Taxer iterates through the labels depends on the version of
the game, and once a label comes up it cannot be revisited). Ranker must decide immediately which of those vertices just selected by Taxer are to receive the given label, extending a partial ranking of \( G \) (Taxer is allowed to use knowledge of the partial ranking already created by Ranker when deciding which vertices are to have a given label in their lists). Ranker wins by creating a ranking of \( G \) before any vertex has its list filled with unused labels, and Taxer wins otherwise.

Complete descriptions of the on-line ranking games are postponed until Section 2, though we do introduce relevant terminology and notation here. Given a graph \( G \) and function \( f : V(G) \to \mathbb{N} \), we say \( G \) is on-line \( f \)-list rankable if Ranker has a winning strategy for the game we shall denote \( R^\pm(G, f) \), and we define the on-line list ranking number of \( G \), denoted \( \rho^\pm(G) \), to be the least \( k \) such that \( G \) is on-line \( f \)-list rankable for the constant function \( f = k \). We similarly define the on-line list low-ranking number \( \rho^-(G) \) and on-line list high-ranking number \( \rho^+(G) \) based on the games \( R^-(G, f) \) and \( R^+(G, f) \), respectively.

Just as we have seen that finding an \( f \)-ranking of \( G \) is equivalent to finding an \( L \)-ranking of \( G \) for a special \( f \)-list assignment \( L \), we will see that any \( f \)-list assignment can be modeled by special strategies for Taxer in the games \( R^-(G, f) \) and \( R^+(G, f) \), and these games in turn can be modeled by special Taxer strategies in \( R^\pm(G, f) \). Thus our parameters will satisfy

\[
\rho(G) \leq \rho^-(G) \leq \min\{\rho^-(G), \rho^+(G)\} \leq \max\{\rho^-(G), \rho^+(G)\} \leq \rho^+(G).
\]

In Section 3 we investigate how ranking a graph relates to ranking its minors in our various versions of the ranking problem.

In Sections 4 and 5 we consider paths and cycles. As stated in [1], \( \rho(P_n) = \lceil \log(n + 1) \rceil \): the largest label can appear only once, so rankings are achieved by individually ranking the subpaths on either side of the vertex receiving the largest label. For instance, \( P_7 \) can be ranked by labeling vertices from left to right with 1, 2, 1, 3, 1, 2, 1. Furthermore, \( \rho(C_n) = 1 + \lfloor \log n \rfloor \), as stated in [2]: the largest label can appear only once, so rankings are achieved by ranking the copy of \( P_{n-1} \) left by deleting the vertex receiving the largest label.

The first main result of this paper is proved in Section 4.

**Theorem 1.3.** \( \rho^\pm(P_n) = \rho(P_n) \).

This theorem is proved using a more general result. For a nonnegative integer valued function \( f \) whose domain includes some finite set \( V \) of vertices,
define $\sigma_f(V) = \sum_{v \in V} 2^{-f(v)}$; we prove that $P_n$ is on-line $f$-list rankable if $\sigma_f(V(P_n)) < 1$. The result is sharp, because $P_{2^k}$ is not even $k$-rankable; the construction extends to all $n$. On the other hand, for $n \geq 5$ there are functions $f$ satisfying $\sigma_f(V(P_n)) > 1$ such that $P_n$ is on-line $f$-list rankable.

The second main result of this paper is proved in Section 5 and relies heavily on the work of the previous section.

**Theorem 1.4.** $\rho^{\pm}_\ell(C_n) = \rho(C_n)$.

Turning our attention toward more complicated graphs, we note the following lower bound for $\rho_\ell(G)$.

**Proposition 1.5.** If $q$ is the maximum number of leaves in a subtree $T$ of a graph $G$, then $\rho_\ell(G) \geq q$.

**Proof.** Construct a $(q - 1)$-uniform list assignment $L$ by giving each vertex that is not a leaf of $T$ the list \(\{1, \ldots, q - 1\}\) and each leaf of $T$ the list \(\{q, \ldots, 2q - 2\}\). If $G$ has an $L$-ranking, then two leaves $u$ and $v$ of $T$ must receive the same label, but no internal vertex of the $u,v$-path through $T$ can contain a larger label, contradicting the definition of a ranking. \(\Box\)

Thus $\rho_\ell(G) - \rho(G)$ can be made arbitrarily large, since for the star $K_{1,n}$ having one internal vertex and $n$ leaves, we have $\rho(K_{1,n}) = 2$ (label the leaves with 1 and the center with 2) but $\rho_\ell(K_{1,n}) \geq n$. Section 6 finds a class of trees for which the bound of Proposition 1.5 is equality, yielding our third main result.

**Theorem 1.6.** For any positive integer $p$, there is a positive integer $q_p$ such that for any tree $T$ with $p$ internal vertices and at least $q_p$ leaves, $\rho_\ell(T)$ equals the number of leaves of $T$.

The proof of Theorem 1.6 gives a specific value of $q_p$ that is exponential in $p$, but this does not appear to be anywhere near sharp.

**Conjecture 1.7.** If $T$ is a tree with $p$ internal vertices and $q$ leaves, and $p < q$, then $\rho_\ell(T) = q$.

It also seems likely that Theorem 1.6 can be strengthened in another way.

**Conjecture 1.8.** For any positive integer $p$, there is a positive integer $q_p$ such that for any tree $T$ with $p$ internal vertices and at least $q_p$ leaves, $\rho^{\pm}_\ell(T)$ equals the number of leaves of $T$. 

6
2. On-line list ranking games

In this section we introduce the various on-line versions of list ranking, presented as games between Taxer and Ranker. Let $G$ be a graph and $f : V(G) \rightarrow \mathbb{N}$. Each $v \in V(G)$ starts the game possessing $f(v)$ tokens (representing the open slots in the list of labels available to $v$), and each round of the game corresponds to a label to be used by Ranker to rank $G$. Taxer starts the round corresponding to the label $c$ by taking tokens from a set $T$ of unranked vertices in $G$ (in effect inserting $c$ into the list of each vertex in $T$). Ranker responds by removing some subset $R$ of $T$ from $G$ and potentially adding certain edges to the resulting graph (in effect labeling vertices in $R$ with $c$, with restrictions placed on $R$ according to $c$, and the set of edges added determined by $c$ and $R$). Taxer’s goal is to bankrupt some vertex without Ranker being able to assign it any of its possible labels.

The three on-line list ranking games we introduce differ in the order the potential labels are introduced. We start with the game in which the labels are introduced in increasing order. In terms of the application of list ranking from Section 1 concerning product assembly, this variation corresponds to determining the feasibility of assembly when the list of individual parts that can be attached at each stage of construction is not known until that stage is reached.

**Definition 2.1.** $G$ is on-line $f$-list low-rankable if Ranker has a winning strategy over Taxer in the following game $R^{-}(G, f)$, which starts by setting $G_1 = G$ and allotting $f(v)$ tokens to each vertex $v$. During round $i$ for $i \geq 1$, Taxer begins play by taking a single token from each element of a nonempty set $T_i$ of vertices of $G_i$. Ranker responds by creating the graph $G_{i+1}$ from $G_i$ by first selecting a subset $R_i$ of $T_i$ that is independent in $G_i$, then adding an edge between any nonadjacent vertices that have a common neighbor in $R_i$, and finally deleting $R_i$. Taxer wins after round $i$ if $G_{i+1}$ contains any vertex without a token, and Ranker wins after round $i$ if $G_{i+1}$ is empty (since Taxer takes at least one token each round, $R^{-}(G, f)$ ends after at most $\sum_{v \in V(G)} f(v)$ rounds). Say $G$ is on-line $k$-list low-rankable if $G$ is on-line $f$-list low-rankable for the constant function $f = k$. Let the on-line list low-ranking number of $G$, denoted $\rho^{-}_{\ell}(G)$, be the least $k$ such that $G$ is on-line $k$-list low-rankable.

See Figure 2 for an example of a round of $R^{-}(G, f)$. Note that if Ranker wins the game after round $j$, then $G$ can be given a $j$-ranking by labeling
each vertex in \( R_i \) with \( i \): if \( u, v \in R_i \) then \( u \) and \( v \) are not adjacent in \( G_i \), so each \( u, v \)-path in \( G \) has an internal vertex \( w \) that appears in \( G_i \) but not \( R_i \), meaning \( w \) receives a larger label than do \( u \) and \( v \). Further note that \( G \) is \( f \)-list rankable if \( G \) is on-line \( f \)-list low-rankable: finding an \( L \)-ranking from an \( f \)-list assignment \( L \), whose set of labels we may presume to be precisely \( \{1, \ldots, m\} \), is equivalent to finding a winning strategy for Ranker in \( R^-(G, f) \) where Taxer must declare before the game that any vertex \( v \) remaining in round \( i \) will be put in \( T_i \) if and only if \( i \in L(v) \).

![Figure 2: A possibility for round \( i \) of \( R^-(G, f) \) (or of \( R^+(G, f) \), if the round is low), where the number on each vertex counts its remaining tokens.](image)

We now introduce the on-line list ranking game in which the labels are introduced in decreasing order (should Ranker win after round \( j \), the actual value of the first label introduced will be \( j \)). In terms of application of list ranking from Section 1 concerning product disassembly, this variation corresponds to determining the feasibility of disassembly when the list of individual parts that can be detached at each stage of deconstruction is not known until that stage is reached.

**Definition 2.2.** \( G \) is on-line \( f \)-list high-rankable if Ranker has a winning strategy over Taxer in the following game \( R^+(G, f) \), which starts by setting \( G_1 = G \) and allotting \( f(v) \) tokens to each vertex \( v \). During round \( i \) for \( i \geq 1 \), Taxer begins play by taking a single token from each element of a nonempty set \( T_i \) of vertices of \( G_i \). Ranker responds by creating the induced subgraph \( G_{i+1} \) of \( G_i \) by deleting from \( G_i \) a subset \( R_i \) of \( T_i \) no two vertices of which lie in the same component. Taxer wins after round \( i \) if \( G_{i+1} \) contains any vertex without a token, and Ranker wins after round \( i \) if \( G_{i+1} \) is empty (since Taxer takes at least one token each round, \( R^+(G, f) \) ends after at most \( \sum_{v \in V(G)} f(v) \) rounds). Say \( G \) is on-line \( k \)-list high-rankable if \( G \) is on-line \( f \)-list high-rankable for the constant function \( f = k \). Let the on-line list
high-ranking number of $G$, denoted $\rho^+_k(G)$, be the least $k$ such that $G$ is on-line $k$-list high-rankable.

See Figure 3 for an example of a round of $R^+(G, f)$. Note that if Ranker wins the game after round $j$, then $G$ can be given a $j$-ranking by labeling each vertex in $R_i$ with $j + 1 - i$: if $u, v \in R_i$ then $u$ and $v$ are in different components of $G_i$, so each $u, v$-path in $G$ has an internal vertex $w$ that does not make it to $G_i$, meaning $w$ receives a larger label than do $u$ and $v$. Further note that $G$ is $f$-list rankable if $G$ is on-line $f$-list high-rankable: finding an $L$-ranking from an $f$-list assignment $L$, whose set of labels we may presume to be precisely $\{1, \ldots, m\}$, is equivalent to finding a winning strategy for Ranker in $R^+(G, f)$ where Taxer must declare before the game that any vertex $v$ remaining in round $i$ will be put in $T_i$ if and only if $m + 1 - i \in L(v)$.

![Figure 3: A possibility for round $i$ of $R^+(G, f)$ (or of $R^\pm(G, f)$, if the round is high), where the number on each vertex counts its remaining tokens.](image)

Our final on-line list ranking game is basically a mixture of the first two, in that at the beginning of each round Taxer gets to decide whether the label to be assigned to vertices in that round is either the least or greatest label yet to be used.

**Definition 2.3.** $G$ is on-line $f$-list rankable if Ranker has a winning strategy over Taxer in the following game $R^\pm(G, f)$, which starts by setting $G_1 = G$ and allotting $f(v)$ tokens to each vertex $v$. During round $i$ for $i \geq 1$, Taxer begins play by declaring the round to be either low or high; low rounds are played like rounds of $R^-(G, f)$ and high rounds are played like rounds of $R^+(G, f)$. As in $R^-(G, f)$ and $R^+(G, f)$, Taxer wins $R^\pm(G, f)$ after round $i$ if $G_{i+1}$ contains any vertex without a token, and Ranker wins after round $i$
if \( G_{i+1} \) is empty (since Taxer takes at least one token each round, \( R^\pm(G, f) \) ends after at most \( \sum_{v \in V(G)} f(v) \) rounds). Say \( G \) is on-line \( k \)-list rankable if \( G \) is on-line \( f \)-list rankable for the constant function \( f = k \). Let the on-line list ranking number of \( G \), denoted \( \rho^\pm(G) \), be the least \( k \) such that \( G \) is on-line \( k \)-list rankable.

Note that if Ranker wins the game after round \( j \), then \( G \) can be given a \( j \)-ranking by labeling each vertex in \( R_i \) with \( i' \) if round \( i \) was the \( i' \)th low round and with \( j + 1 - i'' \) if round \( i \) was the \( i'' \)th high round. Further note that \( G \) is on-line \( f \)-list low-rankable and on-line \( f \)-list high-rankable if \( G \) is on-line \( f \)-list rankable: a winning strategy for Ranker in \( R^-(G, f) \) is a winning strategy for Ranker in \( R^\pm(G, f) \) where Taxer is required to declare each round low, and a winning strategy for Ranker in \( R^+(G, f) \) is a winning strategy for Ranker in \( R^\pm(G, f) \) where Taxer is required to declare each round high.

We summarize our observations about the relationships among the parameters we have introduced.

**Proposition 2.4.** For any graph \( G \), \( \rho(G) \leq \rho^\ell(G) \leq \min\{\rho^-\ell(G), \rho^+(G)\} \leq \max\{\rho^-\ell(G), \rho^+(G)\} \leq \rho^\pm(G) \).

Currently we have no example of a graph \( G \) satisfying \( \rho^\ell(G) < \rho^+(G) \); it would be especially interesting to find a construction that could make \( \rho^+(G) - \rho^\ell(G) \) arbitrarily large. Furthermore, we know of no \( G \) and function \( f \) such that \( G \) is on-line \( f \)-list high-rankable but not on-line \( f \)-list rankable.

We can, however, present a graph \( G \) and function \( f \) such that \( G \) is on-line \( f \)-list low-rankable but not on-line \( f \)-list high-rankable. We use a lemma (which we will use several times more in later sections) that provides list ranking and on-line list ranking analogues of the following observation. If the vertices of a graph \( G \) can be labeled \( v_1, \ldots, v_n \) so that for some index \( k \), the subgraph \( G' \) of \( G \) induced by \( \{v_1, \ldots, v_k\} \) is \( f \)-rankable, and \( f(v_i) \geq i \) for \( k < i \leq n \), then we can construct an \( f \)-ranking of \( G' \) by giving \( G' \) a \( k \)-ranking (which is possible since at most \( k \) labels can be used in a ranking of \( G' \)) and labeling \( v_i \) with \( i \) for \( k < i \leq n \).

**Lemma 2.5.** Let \( G \) be a graph with vertices \( v_1, \ldots, v_n \), and for some index \( k \) let \( G' \) be the subgraph of \( G \) induced by \( \{v_1, \ldots, v_k\} \). Suppose that for every component \( C \) of \( G - V(G') \), the set of vertices in \( G' \) adjacent to vertices in \( C \) is a (possibly empty) clique. Let \( f : V(G) \to \mathbb{N} \) satisfy \( f(v_i) \geq i \) for
Proof. We first prove the lemma for off-line list ranking, and then we prove it for all three on-line list ranking variations.

**Claim.** If \( G' \) is \( f \)-list rankable and \( L \) is an \( f \)-list assignment for \( G \), then \( G \) has an \( L \)-ranking.

**Proof of claim.** We create an \( L \)-ranking of \( G \) in the following manner: first rank \( G' \) from \( L \), then delete the at most \( k \) labels used on \( G' \) from the remaining lists, and finally label \( v_{k+1}, \ldots, v_n \) in order from their remaining lists, deleting the label given to \( v_i \) from the list of each unlabeled vertex. To see that this completes a ranking of \( G \), we note that any path \( P \) in \( G \) between vertices with the same label must have endpoints in \( G' \), in which case \( P \) could be modified into a path \( P' \) in \( G' \) by replacing each maximal subpath of \( P \) in \( G - V(G') \) with an edge in \( G' \) (due to the hypothesis that there is an edge joining any two vertices in \( G' \) that are themselves adjacent to vertices from the same component of \( G - V(G') \)). Since \( G' \) is ranked, and the endpoints of \( P' \) have the same label, an internal vertex of \( P' \) must contain a larger label, and this vertex is also an internal vertex of \( P \). □

**Claim.** If Ranker can win \( R^*(G', f) \) for some \(* \in \{-, +, \pm\} \), then Ranker can win \( R^*(G, f) \).

**Proof of claim.** We prove the claim by using induction on \( n \); the base case of \( n = 1 \) is trivial, so we assume that \( n > 1 \) and the statement holds if \( G \) has fewer than \( n \) vertices. Let \( T \subseteq V(G) \) be the set of vertices from which Taxer takes a token in the first round of \( R^*(G, f) \), let \( R \subseteq T \) be the set of vertices to be removed by Ranker in response, let \( H \) be the graph to be played on in the second round (\( H = G - R \) if the first round was high, and \( H \) loses \( R \) but gains edges completing the remaining neighborhood of each vertex from \( R \) if the first round was low), and let \( H' \) be the subgraph of \( H \) induced by \( V(G') - R \). Let \( h : V(H) \to \mathbb{N} \) be defined by \( h(v_i) = f(v_i) - |T \cap \{v_i\}| \) and \( g : \mathbb{N} \to \mathbb{N} \) be defined by \( g(i) = i - |R \cap \{v_1, \ldots, v_{i-1}\}| \).

We complete the proof by showing Ranker can always choose \( R \) so that \( h(v_i) \geq g(i) \) for \( v_i \in V(H) - V(H') \), Ranker has a winning strategy on \( R^*(H', h) \), and the set \( S \) of vertices in \( H' \) adjacent to any component \( C \) of \( H - V(H') \) is a (possibly empty) clique. Fix such a component \( C \), and define
S as above. We separate the rest of the proof into cases based on whether T includes any vertices from G'.

**Case 1.** \( T \cap V(G') = \emptyset \).

Let \( R = \{v_j\} \), where \( j \) is the least index of any vertex in \( T \) (note that \( j > k \)); clearly this is a legal move by Ranker. In this case, \( H' = G' \), \( h(v_i) = f(v_i) \) for \( 1 \leq i < j \) (with \( f(v_i) \geq g(v_i) \) for \( k < i < j \)), and \( h(v_i) \geq f(v_i) - 1 = g(i) \) for \( j < i \leq n \). Since Ranker has a winning strategy on \( R^*(G', f) \), Ranker also has a winning strategy on \( R^*(H', h) \). Since all vertices in \( S \) are adjacent in \( G \) to vertices in the component of \( G - V(G') \) containing \( C \), \( S \) is a subset of a clique and thus a clique itself.

**Case 2.** \( T \cap V(G') \neq \emptyset \).

Let \( R \) consist of the vertices that Ranker would remove in a winning strategy on \( R^*(G', f) \) in response to Taxer removing tokens from \( T \cap V(G') \). This is a legal move by Ranker because if the first round is low, then \( R \) is independent in \( G' \) and thus also in \( G \), and if the first round is high, then any two vertices in \( R \) are in different components of \( G' \) and thus also in different components of \( G \) (since otherwise some component of \( G - V(G') \) would have nonadjacent neighbors in \( G' \)). Clearly, \( h(v_i) \geq f(v_i) - 1 \geq g(i) \) for \( v_i \in V(H) - V(H') \), Ranker has a winning strategy on \( R^*(H', h) \), and \( C \) is a component of \( G - V(G') \). We only need to show that \( S \) is a clique.

If the first round is high, then \( H = G - R \), so all vertices in \( S \) are adjacent in \( G \) to vertices in \( C \), making \( S \) a subset of a clique and thus a clique itself. If the first round is low, then \( H \) is obtained from \( G \) by deleting each vertex in \( R \) after completing its neighborhood. Any vertex in \( S \) is either adjacent in \( G \) to a vertex in \( C \) or adjacent in \( G \) to a vertex in \( R \) adjacent to a vertex in \( C \), so \( S \) is a clique since the set of vertices in \( G' \) adjacent to vertices in \( C \) is a clique, and the neighborhood of any vertex in \( R \) is completed when forming \( H \) from \( G \).

Hence if \( G' \) is \( f \)-list rankable, then so is \( G \), and for \( * \in \{-, +, \pm\} \), if Ranker wins \( R^*(G', f) \), then Ranker also wins \( R^*(G, f) \).

We are now ready to present a graph \( G \) and function \( f \) such that \( G \) is on-line \( f \)-list low-rankable but not on-line \( f \)-list high-rankable.

**Proposition 2.6.** If the vertices of \( P_n \) are \( v_1, \ldots, v_n \) in order, then \( P_n \) is on-line \( f \)-list low-rankable but not on-line \( f \)-list high-rankable for \( f = (3, 1, 2, 3, 5, 6, \ldots, n) \).
Proof. We show $P_n$ is on-line $f$-list low-rankable by giving a winning strategy for Ranker on $R^-(P_n, f)$, and we show $P_n$ is not on-line $f$-list high-rankable by giving a winning strategy for Taxer on $R^+(P_n, f)$.

Claim. Ranker wins $R^-(P_n, f)$.

Proof of claim. By Lemma 2.5 it suffices to exhibit a winning strategy for Ranker on $R^-(P_4, (3, 1, 2, 3))$. If Taxer takes a token from $v_2$ in the first round, let Ranker remove $v_2$, and also remove $v_4$ if Taxer selects that as well. Then, assuming Taxer also removed tokens from $v_1$ and $v_3$ (the worst-case scenario for Ranker in this situation), the game reduces to either $R^-(P_2, (2, 1))$ ($v_2$ and $v_4$ removed by Ranker) or $R^-(P_3, (2, 1, 3))$ ($v_2$ removed), both of which lead to victory for Ranker, by Lemma 2.5.

If Taxer does not remove a token from $v_2$ in the first round, let Ranker remove $v_1$ if selected, $v_3$ if selected, and $v_4$ if selected and $v_3$ is not selected. Then, assuming Taxer removed a token from $v_4$ if one was also taken from $v_3$ (again, the worst-case scenario for Ranker in this situation), the game reduces to either $R^-(P_2, (1, 2))$ ($v_1$ and either $v_3$ or $v_4$ removed), $R^-(P_3, (1, 2, 3))$ ($v_1$ removed), or $R^-(P_3, (3, 1, 2))$ (either $v_3$ or $v_4$ removed), each of which leads to victory for Ranker, by Lemma 2.5.

Claim. Taxer wins $R^+(P_n, f)$.

Proof of claim. Let Taxer begin play by removing tokens from $v_1$ and $v_4$. If Ranker responds by removing $v_1$, let Taxer next remove tokens from $v_2$, $v_3$, and $v_4$, leaving $v_2$ with no tokens and $v_3$ and $v_4$ with one each. Then Ranker must remove $v_2$, and if Taxer selects $v_3$ and $v_4$, Ranker can remove just one of these. The other vertex is left with no token, so Taxer wins.

If Ranker responds to Taxer’s initial selection by removing $v_4$, let Taxer next remove tokens from $v_1$ and $v_3$, leaving each remaining vertex with a single token. Ranker cannot remove both $v_1$ and $v_3$, leaving $v_2$ adjacent to another vertex, each with a single token. If Taxer selects both of these vertices then Ranker can remove just one, leaving the other with no token. Thus Taxer wins $R^+(P_n, f)$.

Hence Ranker wins $R^-(P_n, f)$ while Taxer wins $R^+(P_n, f)$, so $P_n$ is on-line $f$-list low-rankable but not on-line $f$-list high-rankable.
3. List ranking and online list ranking graph minors

We now examine how ranking a graph relates to ranking one of its minors in our various versions of the ranking problem. We first recall the definition of a graph minor and introduce some notation to be used in this section.

Definition 3.1. A **minor** of a graph $G$ is any graph $G'$ for which there exists a mapping that sends each vertex $w \in V(G')$ to a connected subgraph $G_w$ of $G$ induced by some set $U(w) \subseteq V(G)$ having the following properties: for distinct vertices $w_1$ and $w_2$ of $G'$, $U(w_1) \cap U(w_2) = \emptyset$, and if $w_1$ and $w_2$ are adjacent in $G'$, then there exist $u_1 \in U(w_1)$ and $u_2 \in U(w_2)$ that are adjacent in $G$.

Equivalently, $G'$ is a minor of $G$ if $G'$ can be obtained by performing zero or more edge contractions on a subgraph $F$ of $G$, where an edge $uv$ is contracted from $F$ by deleting $u$ and $v$ and adding a new vertex $w$ adjacent to all former neighbors of $u$ and $v$ in $F$.

For the rest of this section, fix a graph $G$ and a function $f : V(G) \to \mathbb{N}$ as well as a minor $G'$ of $G$ and a function $f' : V(G') \to \mathbb{N}$. For notational convenience, we say that $G$ being $f$-list rankable is equivalent to Ranker having a winning strategy for the game $R^\ell(G, f)$ (with Taxer’s goal in this game being to present an $f$-list assignment $L$ that admits no ranking of $G$); let $* \in \{\ell, -, +, \pm\}$. Our results in this section give sufficient conditions for Ranker to have a winning strategy on $R^* (G', f')$. Often one of our conditions is that Ranker has a winning strategy in $R^* (G, f)$; to prove Ranker can beat any Taxer strategy in $R^* (G', f')$, we translate the strategy taken by Taxer in the game $R^* (G', f')$ to a strategy by Taxer in the game $R^* (G, f)$, then translate Ranker’s winning strategy against Taxer in $R^* (G, f)$ back to a winning strategy for Ranker in $R^* (G', f')$. We first employ this technique to prove the following statement on list ranking minors.

Proposition 3.2. If $G$ is $f$-list rankable, but $G_w$ is not $(f - f'(w))$-list rankable for each $w \in V(G')$, then $G'$ is $f'$-list rankable.

Proof. We show $G'$ is $f'$-list rankable by constructing an $L'$-ranking $\alpha'$ from an arbitrary $f$-list assignment $L'$ for $G'$. For each $w \in V(G')$, let $L_w$ be an $(f - f'(w))$-list assignment for $G_w$ such that $G_w$ has no $L_w$-ranking. Without loss of generality assume that the smallest label in any list assigned by $L'$ is larger than the largest label in any list assigned by any $L_w$ (we may assume this since $G'$ has an $L'$-ranking if and only if $G'$ has an $L''$-ranking, where $L''$
is any list assignment obtained from \( L' \) by adding the same integer constant to each label in each list assigned by \( L' \). Let \( L \) be the \( f \)-list assignment for \( G \) obtained from \( L' \) by setting \( L(u) = L'(u) \cup L_w(u) \) for each \( w \in V(G') \) and \( u \in V(G_w) \), and \( L(t) = \{1, \ldots, f(t)\} \) for each \( t \in V(G) - \bigcup_{w \in V(G')} V(G_w) \).

By hypothesis \( G \) has an \( L \)-ranking \( \alpha \). We modify \( \alpha \) into an \( L' \)-ranking \( \alpha' \) of \( G' \) by letting \( \alpha'(w) = \max\{\alpha(u) : u \in V(G_w)\} \). Note that \( \alpha'(w) \in L'(w) \) for all \( w \in V(G') \) since \( \alpha \) cannot assign every vertex in \( G_w \) a label from \( L_w \) (or else \( G_w \) would have an \( L_w \)-ranking), and the smallest label in \( L'(w) \) is larger than the largest label anywhere in \( L_w \). We prove that \( \alpha' \) is in fact a ranking of \( G' \) by showing that if \( w_1, w_2, \ldots w_m \) is a path \( P' \) in \( G' \) satisfying \( w_1 \neq w_m \) and \( \alpha'(w_1) = \alpha'(w_m) \), then \( \alpha'(w_i) > \alpha'(w_1) \) for some \( 1 < i < m \).

By the definition of \( \alpha' \), for \( 1 \leq i \leq m \) there exists \( u_i \in V(G_{w_i}) \) such that \( \alpha(u_i) = \alpha'(w_i) \). Since \( w_i \) and \( w_{i+1} \) are adjacent in \( G' \) for \( 1 \leq i < m \), for \( 1 \leq i \leq m \) there exist (not necessarily distinct) vertices \( x_i, y_i \in V(G_{w_i}) \) such that \( x_i = u_1, y_i = u_m, m_1 \) and \( y_i \) is adjacent to \( x_{i+1} \) in \( G \) for \( 1 \leq i < m \). Since each \( G_{w_i} \) is connected, there exists a \( x_i, y_i \)-path \( P^i \) in \( G_{w_i} \) for \( 1 \leq i \leq m \). Hence concatenating the paths \( P^1, P^2, \ldots, P^m \) forms a \( u_1, u_m \)-path \( P \) in \( G \).

Since \( \alpha \) is a ranking of \( G \) and \( \alpha(u_1) = \alpha'(w_1) = \alpha'(w_m) = \alpha(u_m) \), for some \( i \) there exists \( z \in V(P^i) \) satisfying \( \alpha(z) > \alpha(u_1) \), in which case \( \alpha'(w_i) \geq \alpha(z) > \alpha(u_1) = \alpha'(w_1) \). \( \square \)

Suppose \( G' \) happens to be a subgraph of a graph \( G \), and \( f'(v) \geq f(v) \) for each \( v \in V(G') \). Clearly \( G' \) is \( f' \)-rankable if \( G \) is \( f \)-rankable: every \( f \)-ranking of \( G \) is also an \( f' \)-ranking of \( G' \). Since \( G' \) is also a minor of \( G \), with \( G_v = v \) and \( f(v) - f'(v) \leq 0 \) for each \( v \in V(G') \) (so \( G_v \) is not \( (f - f'(v)) \)-list rankable), Proposition 3.2 yields that \( G' \) is \( f' \)-list rankable if \( G \) is \( f \)-list rankable. Lemma 3.3 extends this statement to the on-line list ranking games.

**Lemma 3.3.** Fix \( \ast \in \{-, +, \pm\} \). If Ranker wins \( R^*(G, f) \), \( G' \) is a subgraph of \( G \), and \( f'(v) \geq f(v) \) for each \( v \in V(G') \), then Ranker wins \( R^*(G', f') \).

**Proof.** We perform induction on \( |V(G)| \). The base case of \( |V(G)| = 1 \) is trivial, so we may assume that \( |V(G)| > 1 \) and the statement holds for smaller graphs. Taxer begins \( R^*(G', f') \) by declaring the first round high or low and then taking tokens from certain vertices of \( G' \); let Taxer declare the first round of \( R^*(G, f) \) to be of the same type before taking tokens from the same vertices of \( G \). Whichever of these vertices Ranker removes from \( G \) to create the graph \( F \) in a winning strategy on \( R^*(G, f) \), let Ranker respond in
$R^*(G', f')$ by removing the same set of vertices from $G'$ to create the graph $F'$.

Since $f'(v) \geq f(v)$ for each $v \in V(G')$, and every vertex in $F'$ that lost a token during the first round of $R^*(G', f')$ also lost one during the first round of $R^*(G, f)$, every vertex in $F'$ has at least as many tokens remaining as its corresponding vertex in $F$. Furthermore, $F'$ is a subgraph of $F$, with vertices in $F'$ corresponding to the same vertices in $F$ that they did in $G$: $G'$ was a subgraph of $G$, and if the first round was low, then vertices selected by Ranker are deleted but their neighborhoods are completed, and if the first round was high, then vertices selected by Ranker are simply deleted without adding any new edges. Since Ranker wins $R^*(G, f)$, Ranker can win the new game on $F$, so by the inductive hypothesis Ranker can win the new game on $F'$ as well. Thus Ranker wins $R^*(G', f')$.

**Proposition 3.4.** Fix $* \in \{-, \pm\}$. If Ranker wins $R^*(G, f)$ but Taxer wins $R^-(G_w, f - f'(w))$ for each $w \in V(G')$, then Ranker wins $R^*(G', f')$.

**Proof.** Based on Taxer’s strategy on $R^*(G', f')$, we define a strategy for Taxer on an auxiliary game $R^*(G, f)$, and use Ranker’s winning strategy on $R^*(G, f)$ to define a winning strategy for Ranker on $R^*(G', f')$. Let Taxer begin the auxiliary game $R^*(G', f')$ by isolating play to each $G_w$ one-at-a-time and copying a winning strategy from $R^-(G_w, f - f'(w))$ until some $u \in V(G_w)$ is left with at most $f'(w)$ tokens; once this happens say that $w$ and $u$ are partners. Let Taxer continue the auxiliary game $R^*(G, f)$ by declaring low rounds and taking tokens from vertices of $G$ not partnered with vertices of $G'$ until Ranker has removed all such vertices (which Ranker must do since Ranker is playing a winning strategy on $R^*(G, f)$ and thus will never leave any vertex without a token).

Since each round so far has been low, the neighborhood of each vertex removed by Ranker has been completed before the next round, so the partnership between the vertices of $G'$ and the vertices of the altered graph $H$ of $R^*(G, f)$ is a graph isomorphism from $G'$ to a spanning subgraph of $H$. Letting $h(u)$ count the tokens on each $u \in V(H)$, we note that $h(u) \leq f'(w)$ if $u$ is partnered with $w \in V(G')$. The auxiliary game $R^*(G, f)$ has been reduced to $R^*(H, h)$, for which Ranker has a winning strategy since Ranker wins $R^*(G, f)$. By Lemma 3.3, Ranker has a winning strategy for $R^*(G', f')$. □

**Proposition 3.5.** If Ranker wins $R^+(G, f)$ but Taxer wins $R^+(G_w, f - f'(w))$ for each $w \in V(G')$, then Ranker wins $R^+(G', f')$. □
Proof. We show Ranker wins $R^+(G', f')$ by performing induction on $|V(G)|$, with the base case of $|V(G)| = 1$ being trivial. Now assume $|V(G)| > 1$ and the statement holds for smaller graphs. Let Taxer begin $R^+(G', f')$ by taking tokens from vertices in the set $T' \subseteq V(G')$. Based on $T'$, we decide from which set $T \subseteq V(G)$ Taxer will take tokens in the first round of the auxiliary game $R^+(G, f)$: let $T = \bigcup_{w \in T'} V(G_w)$. After Ranker responds as part of a winning strategy in the auxiliary game $R^+(G, f)$ by removing the vertices of some $R \subseteq T$ to create the graph $F$, we decide which set $R' \subseteq T'$ Ranker will remove from $G'$ to create the graph $F'$ in $R^+(G', f')$: let $R' = \{ w \in T' : V(G_w) \cap R \neq \emptyset \}$.

The games $R^+(G, f)$ and $R^+(G', f')$ have been reduced to $R^+(F, h)$ and $R^+(F', h')$, respectively, where $F = G - R$ and $F' = G' - R'$, with the functions $h$ and $h'$ defined by $h(v) = f(v) - 1$ if $v \in T - R$ and $h(v) = f(v)$ otherwise, and $h'(w) = f'(v) - 1$ if $w \in T' - R'$ and $h'(w) = f'(w)$ otherwise. To complete the proof, we show that Ranker wins $R^+(F', h')$. Because Ranker wins $R^+(F, h)$ (since Ranker is playing a winning strategy on $R^+(G, f)$), and $|V(F)| < |V(G)|$, by the inductive hypothesis we only need to show that for $w \in V(F')$, $G_w$ survives to $F$, with $F'$ a minor of $F$ according to the mapping $F_w = G_w$ and Taxer having a winning strategy on $R^+(G_w, h - h'(w))$.

Claim. For each $w \in V(F')$ and $v \in V(G_w)$, $v$ survives as a vertex in $F$, and $h(v) - h'(w) = f(v) - f'(w)$.

Proof of claim. Let $w \in V(F')$ and $v \in V(G_w)$. Either $w \in V(G') - T'$ or $w \in T' - R'$ (since $F' = G' - R'$ and $R' \subseteq T' \subseteq V(G')$). If $w \in V(G') - T'$, then $h'(w) = f'(w)$ (since $w \in V(F') - T'$), and for $v \in V(G_w)$ we have $v \in V(F)$ as well as $h(v) = f(v)$ (since $V(G_w) \cap R \subseteq V(G_w) \cap T = \emptyset$ for $w \notin T'$). If $w \in T' - R'$, then $h'(w) = f'(w) - 1$, and for $v \in V(G_w)$ we have $v \in V(F)$ as well as $h(v) = f(v) - 1$ (since $V(G_w) \subseteq T - R \subseteq V(F)$ for $w \in T' - R'$, as $V(G_w) \subseteq T$ for $w \in T'$ and $V(G_w) \cap R = \emptyset$ for $w \notin R'$).

Claim. For each $w \in V(F')$, $G_w$ survives as a subgraph of $F$, with Taxer having a winning strategy on $R^+(G_w, h - h'(w))$.

Proof of claim. Since $F$ and $F'$ are induced subgraphs of $G$ and $G'$, respectively, and $V(G_w) \subseteq V(F')$ for each $w \in V(F')$, $G_w$ survives as a subgraph of $F$ for each $w \in V(F')$. We have shown $h(v) - h'(w) = f(v) - f'(w)$ for each $v \in V(G_w)$, so Taxer wins $R^+(G_w, h - h'(w))$ because Taxer wins $R^+(G_w, f - f'(w))$ by our original hypothesis.
**Claim.** \( F' \) is a minor of \( F \) according to the mapping \( F_w = G_w \) for \( w \in V(F') \).

**Proof of claim.** Let \( v_1 \) and \( v_2 \) be distinct vertices in \( F \), and let \( w_1 \) and \( w_2 \) be distinct vertices in \( F' \). \( G_{w_1} \) is connected and \( V(G_{w_1}) \cap V(G_{w_2}) = \emptyset \) since \( G' \) is a minor of \( G \), so \( F_{w_1} \) is connected and \( V(F_{w_1}) \cap V(F_{w_2}) = \emptyset \) too. Furthermore, \( v_1 \) is adjacent to \( v_2 \) in \( F \) if and only if \( v_1 \) is adjacent to \( v_2 \) in \( F \) since \( F \) is an induced subgraph of \( G \), and \( w_1 \) is adjacent to \( w_2 \) in \( F' \) if and only if \( w_1 \) is adjacent to \( w_2 \) in \( G' \) since \( F' \) is an induced subgraph of \( G' \). Thus if \( w_1 \) and \( w_2 \) are adjacent in \( F' \), then they are also adjacent in \( G' \), so some \( u_1 \in V(G_{w_1}) \) is adjacent to some \( u_2 \in V(G_{w_2}) \) in \( G \) since \( G' \) is a minor of \( G \), and thus \( u_1 \) and \( u_2 \) are also adjacent in \( F \). Therefore \( F' \) is a minor of \( F \). ■

Hence Ranker wins \( R^+(F', h') \) and thus \( R^+(G', f') \).

We now produce some corollaries of Propositions 3.2, 3.4, and 3.5, framed by statements concerning the original ranking problem. Recall that we say that \( G \) being \( f \)-list rankable is equivalent to Ranker having a winning strategy for the game \( R^\ell(G, f) \). Fix \( \ast \in \{\ell, -, +, \pm\} \).

**Corollary 3.6.** Let \( G' \) be a minor of a graph \( G \), and suppose \( f : V(G) \to \mathbb{N} \) and \( f' : V(G') \to \mathbb{N} \) satisfy \( f'(w) \geq \min_{u \in V(G_w)} f(u) \) for all \( w \in V(G') \). If Ranker wins \( R^\ast(G, f) \), then Ranker also wins \( R^\ast(G', f') \).

**Proof.** The statement follows from applying either Proposition 3.2, 3.4, or 3.5, since clearly Taxer wins \( R^\ast(G_w, f - f'(w)) \) for each \( w \in V(G') \) such that \( f(u) - f'(w) \leq 0 \) for some \( u \in V(G_w) \). ■

Corollary 3.6 cannot be extended to the original ranking problem, however. Indeed, let \( n \geq 3 \) and consider \( G = P_n \) with vertices \( v_1, \ldots, v_n \) in order, and let \( G' \) be the minor of \( G \) consisting of a single edge \( xy \). Let \( f(v_1) = f(v_n) = f'(x) = f'(y) = 1 \) and \( f(v_i) = n \) for \( 1 < i < n \). We can give \( G \) an \( f \)-ranking by labeling \( v_1 \) and \( v_n \) with 1 and \( v_i \) with \( i \) for \( 1 < i < n \), but we cannot give \( G' \) an \( f' \)-ranking because \( x \) and \( y \) would both have to be labeled with 1 even though they are adjacent.

Clearly a graph is \( f \)-rankable if and only if each of its components is \( f \)-rankable, and this statement also holds for the list versions of ranking.

**Corollary 3.7.** Ranker wins \( R^\ast(G, f) \) if and only if Ranker wins \( R^\ast(G', f) \) for each component \( G' \) of \( G \).
Proof. If Ranker wins $R^*(G, f)$, then Ranker wins $R^*(G', f)$ for each component $G'$ of $G$, by setting $f' = f$ in Corollary 3.6. Now suppose Ranker wins $R^*(G', f)$ for each component $G'$ of $G$. If $* = \ell$, then Ranker can win $R^*(G, f)$ since each component can be dealt with individually. If $* \in \{-, +, \pm\}$, then Ranker can win $R^*(G, f)$ by treating each move by Taxer on $G$ as a collection of separate moves on the games $R^*(G', f)$ and playing winning strategies for each of those games.

We shall see that the following statement does not hold for the original ranking problem.

**Corollary 3.8.** Suppose $G'$ is obtained from $G$ by contracting an edge $uv$ into a vertex $w$, and $f(u) = f(v) = f'(w) + 1$ while $f = f'$ elsewhere. If Ranker wins $R^*(G, f)$, then Ranker wins $R^*(G', f')$.

**Proof.** Apply either Proposition 3.2, 3.4, or 3.5, since clearly Taxer wins $R^*(G_w, f - f'(w))$ if $G_w$ consists of an edge $uv$ such that $f(u) - f'(w) = f(v) - f'(w) = 1$. 

To see why Corollary 3.8 cannot be extended to the original ranking problem, let $n \geq 6$ and consider $G = P_n$ with vertices $v_1, \ldots, v_n$ in order and the minor $G'$ of $G$ obtained by contracting the edge $v_{n-1}v_{n-2}$ into the vertex $w$. Let $f(v_i) = f'(v_i) = n$ for $1 \leq i \leq n - 3$ while $f(v_{n-2}) = f'(v_{n-2}) = f'(w) = 1$ and $f(v_{n-1}) = f(v_n) = 2$. We can give $G$ an $f$-ranking by labeling $v_i$ with $i$ for $1 \leq i \leq n - 3$, $v_{n-2}$ and $v_n$ with 1, and $v_{n-1}$ with 2, but we cannot give $G'$ an $f'$-ranking because $v_{n-2}$ and $w$ would both have to be labeled with 1 even though they are adjacent.

### 4. Paths

To prove our first main result, that $\rho^\pm_{f}(P_n) = \lceil \log(n+1) \rceil = \rho(P_n)$, we instead prove the stronger statement that $P_n$ is on-line $f$-list rankable if $\sigma_f(V(P_n)) < 1$, where $\sigma_f(V) = \sum_{v \in V} 2^{-f(v)}$ for a nonnegative integer-valued function $f$ defined on a set $V$ of vertices.

Throughout this section, we will refer to the vertices of $P_n$ be $v_1, \ldots, v_n$ from left to right. Recall that $R^\pm(P_n, f)$ starts with each $v_i$ having $f(v_i)$ tokens, with Taxer beginning play by declaring the round to be low or high and then taking one token from each element of a nonempty set $T$ of vertices of $P_n$. If the round is low, then Ranker responds by choosing an independent
Proof. Let \( R \subseteq T \) to remove from \( P_n \), replacing each removed vertex with an edge between its neighbors to get a path on \( n - |R| \) vertices. If the round is high, then Ranker responds by choosing a vertex \( v \in T \) to delete from \( P_n \) to get two path components on a total of \( n - 1 \) vertices.

We isolate the following argument as a lemma because it alone is enough to show that \( P_n \) is on-line \( f \)-list low-rankable (and thus \( f \)-list rankable) if \( \sigma_f(V(P_n)) < 1 \).

**Lemma 4.1.** Suppose \( \sigma_f(V(P_n)) < 1 \), and Ranker has a winning strategy on \( R^\pm(P_m, g) \) if \( m < n \) and \( \sigma_g(V(P_m)) < 1 \). If Taxer declares the first round of \( R^\pm(P_n, f) \) low, then Ranker can win.

**Proof.** Let \( B = \{v_{2i-1} : 1 \leq i \leq \lfloor n/2 \rfloor \} \) and \( C = \{v_{2i} : 1 \leq i \leq \lfloor n/2 \rfloor \} \). Let \( R = B \cap T \) if \( \sigma_f(B \cap T) \geq \sigma_f(C \cap T) \) and \( R = C \cap T \) otherwise. Then \( R \) is independent and \( \sigma_f(R) \geq \sigma_f(T - R) \). The game is reduced to \( R^\pm(P_n - |R|, g) \), where \( g = f - 1 \) on \( T - R \) and \( g = f \) elsewhere. Since \( \sigma_f(R) \geq \sigma_f(T - R) \), we have

\[
\sigma_g(V(P_n) - R) = \sigma_f(V(P_n)) + \sigma_f(T - R) - \sigma_f(R) \leq \sigma_f(V(P_n)) < 1,
\]

so Ranker wins this game and thus the original, by hypothesis. \( \square \)

**Theorem 4.2.** Ranker wins \( R^\pm(P_n, f) \) if \( \sigma_f(V(P_n)) < 1 \).

**Proof.** We use induction on \( n \). Clearly Ranker wins \( R^\pm(P_1, f) \) when \( f = 1 \), so we assume \( n > 1 \) and that Ranker has a winning strategy on \( R^\pm(P_m, g) \) when \( m < n \) and \( \sigma_g(V(P_m)) < 1 \). By Lemma 4.1 we may also assume Taxer declares the first round high, so Ranker must remove one vertex \( v \) from the set \( T \) of vertices from which Taxer removes a token.

We will visit the vertices of \( P_n \) in some order \( v_{p_1}, \ldots, v_{p_n} \) such that \( \{v_{p_1}, \ldots, v_{p_i} \} \) induces a path (not necessarily with the vertices in that order) for \( 1 \leq i \leq n \). We will let \( V^{<p_i} = \{v_1, \ldots, v_{p_i-1} \} \) and \( P^{<p_i} \) be the subgraph of \( P_n \) induced by \( V^{<p_i} \), and we will let \( V^{>p_i} = \{v_{p_i+1}, \ldots, v_n \} \) and \( P^{>p_i} \) be the subgraph of \( P_n \) induced by \( V^{>p_i} \). For \( 1 \leq i \leq n \), set \( g_i = f \) on \( v_{p_1}, \ldots, v_{p_{i-1}} \) and \( g_i = f - 1 \) elsewhere. As we construct our ordering of \( V(P_n) \), we will require \( \sigma_{g_i}(V^{<p_i}) < 1 \) and \( \sigma_{g_i}(V^{>p_i}) < 1 \) for \( 1 \leq i \leq n \), properties we will show guarantee lead to a winning strategy for Ranker if he selects \( v = v_{p_i} \) in the first round for the least index \( i \) such that \( v_{p_i} \in T \).

We construct our ordering \( v_{p_1}, \ldots, v_{p_n} \) of \( V(P_n) \) inductively. Select \( p_1 \) as the least index such that \( \sigma_f(V^{<p_1} \cup \{v_{p_1} \}) \) reaches at least 1/2, unless
Thus we may assume

$$\sigma_f(V(P_n)) < 1/2,$$

in which case set $p_1 = n$. Since $g_1 = f - 1$, we have

$$\sigma_{g_1}(V^{<p_1}) = 2\sigma_f(V^{<p_1}) < 1$$

and

$$\sigma_{g_1}(V^{>p_1}) = 2\sigma_f(V^{>p_1}) = 2(\sigma_f(V(P_n)) - \sigma_f(V^{<p_1} \cup \{v_{p_1}\})) < 2(1 - 1/2) = 1,$$

as desired. Now assume that $k < n$ and $p_1, \ldots, p_k$ have been found such that

$$\{v_{p_1}, \ldots, v_{p_k}\}$$

induces a path and such that $\sigma_{g_1}(V^{<p_i}) < 1$ and $\sigma_{g_1}(V^{>p_i}) < 1$

for $1 \leq i \leq k$. Let $P_n - \{v_{p_1}, \ldots, v_{p_k}\}$ consist of the paths induced by

$$\{v_1, \ldots, v_s\}$$

and

$$\{v_t, \ldots, v_n\},$$

where we set $s = 0$ or $t = n + 1$ if $v_1$ or $v_n$ is in

$$\{v_{p_1}, \ldots, v_{p_k}\},$$

respectively.

**Claim.** Ranker can choose $p_{k+1} \in \{s, t\} - \{0, n+1\}$ so that $\sigma_{g_{k+1}}(V^{<p_{k+1}}) < 1$

and $\sigma_{g_{k+1}}(V^{>p_{k+1}}) < 1$.

**Proof of claim.** If $p_{k+1} = s \geq 1$, then

$$\sigma_{g_{k+1}}(V^{<p_{k+1}}) \leq \sigma_{g_k}(V^{<p_k}) - 2^{1-g_k(v_s)} < 1,$$

and if $p_{k+1} = t \leq n$, then

$$\sigma_{g_{k+1}}(V^{>p_{k+1}}) \leq \sigma_{g_k}(V^{>p_k}) - 2^{1-g_k(v_t)} < 1.$$

Thus it only remains to show that Ranker can either choose $p_{k+1} = s \geq 1$ such

that $\sigma_{g_{k+1}}(V^{>p_{k+1}}) < 1$ or choose $p_{k+1} = t \leq n$ such that $\sigma_{g_{k+1}}(V^{<p_{k+1}}) < 1$.

If $t = n + 1$, then $s = n - k \geq 1$, and setting $p_{k+1} = s$ yields

$$\sigma_{g_{k+1}}(V^{>p_{k+1}}) = \sigma_f(V^{>p_{k+1}}) < \sigma_f(V(P_n)) < 1.$$

If $s = 0$, then $t = k + 1 \leq n$, and setting $p_{k+1} = t$ yields

$$\sigma_{g_{k+1}}(V^{<p_{k+1}}) = \sigma_f(V^{<p_{k+1}}) < \sigma_f(V(P_n)) < 1.$$

Thus we may assume $s \geq 1$ and $t \leq n$. Note that

$$\sum_{i=1}^{s} 2^{1-f(v_i)} + 2^{t-1-s+1} 2^{-f(v_s)} + \sum_{i=t}^{n} 2^{1-f(v_i)} = 2\sigma_f(V(P_n)) < 2,$$

so at least one of the following holds:

$$\sum_{i=s+1}^{t-1} 2^{-f(v_i)} + \sum_{i=t}^{n} 2^{1-f(v_i)} < 1,$$
in which case Ranker can set \( p_{k+1} = s \) to get \( \sigma_{g_{k+1}}(V^{>p_{k+1}}) < 1 \), or
\[
\sum_{i=1}^{s} 2^{1-f(v_i)} + \sum_{i=s+1}^{t-1} 2^{-f(v_i)} < 1.
\]
in which case Ranker can set \( p_{k+1} = t \) to get \( \sigma_{g_{k+1}}(V^{<p_{k+1}}) < 1 \).

**Claim.** For \( 1 \leq i \leq n \), if \( i \) is the least index such that \( v_{p_i} \in T \), then selecting \( v = v_{p_i} \) in the first round leads to a winning strategy for Ranker.

**Proof of claim.** Let \( g = f - 1 \) on \( T \) and \( g = f \) elsewhere. Setting \( v = v_{p_i} \) reduces \( R^\pm(P_n, f) \) to separate games of \( R^\pm(P^{<p_i}, g) \) and \( R^\pm(P^{>p_i}, g) \), both of which Ranker wins by our inductive hypothesis: \( P^{<p_i} \) and \( P^{>p_i} \) each have fewer than \( n \) vertices, and \( g \geq g_i \) since \( v_{p_j} \notin T \) for \( 1 \leq j < i - 1 \), yielding
\[
\sigma_g(V^{<p_i}) \leq \sigma_{g_i}(V^{<p_i}) < 1
\]
as well as
\[
\sigma_g(V^{>p_i}) \leq \sigma_{g_i}(V^{>p_i}) < 1.
\]

Hence Ranker wins \( R^\pm(P_n, f) \). □

Naturally we want to see what happens when \( \sigma_f(V(P_n)) \geq 1 \). To show the sharpness of Theorem 4.2, we exhibit a function \( f \) such that \( \sigma_f(V(P_n)) = 1 \) but \( P_n \) is not even \( f \)-rankable, much less on-line \( f \)-list rankable.

**Proposition 4.3.** For \( n \geq 1 \), there is a function \( f \) such that \( \sigma_f(V(P_n)) = 1 \) and \( P_n \) is not \( f \)-rankable.

**Proof.** Fix \( n \), and set \( k = \lceil \log n \rceil \). Define \( f(v_i) = k \) if \( n - 2^k < i \leq 2^k \) and \( f(v_i) = k + 1 \) otherwise. Recall that \( \rho(P_m) = \lceil \log(m+1) \rceil \) (as stated in [1]), so \( \rho(P_m) = k + 1 \) for \( 2^k \leq m \leq n \). If \( n = 2^k \), then \( P_n \) is not \( f \)-rankable because \( f = k \) everywhere and \( \rho(P_n) = k + 1 \). If \( n > 2^k \), then only one vertex \( v_i \) can be labeled \( k+1 \), and \( i \) must satisfy \( 1 \leq i \leq n - 2^k \) or \( 2^k < i \leq n \) (or else \( f(v_i) = k \)). Thus one component \( P \) of \( P_n - v_i \) is a path on at least \( 2^k \) vertices which must be given a \( k \)-ranking, which is impossible since \( \rho(P) = k + 1 \). □

From Proposition 4.3 it may seem hopeful that a perfect converse of Theorem 4.2 holds, but unfortunately one does not.
Proposition 4.4. For \( n \geq 4 \), there is a function \( f \) such that \( \sigma_f(V(P_n)) = 1 \) and Ranker wins \( R^\pm(P_n, f) \).

Proof. Let \( f(v_1) = 2, f(v_2) = 3, f(v_3) = 1, f(v_i) = i \) for \( 4 \leq i \leq n-1 \), and \( f(v_n) = n-1 \). Then \( \sigma_f(V(P_n)) = 1 \). Suppose the first round is low. If \( n \geq 5 \) and Taxer removes tokens from \( v_{n-1} \) and \( v_n \), then Ranker can remove \( v_n \) to reduce the game to \( R^\pm(P_{n-1}, f') \), where \( f'(v_{n-1}) = n-2 \) and \( f' = f \) elsewhere, so we may assume \( n = 4 \) or \( T \neq \{v_{n-1}, v_n\} \). If Ranker removes an independent set \( R \subseteq T \) that maximizes \( \sigma_f(R) \) then the game will reduce to \( R^\pm(P_{n-|R|}, g) \) where \( \sigma_g(V(P_{n-|R|})) < 1 \), so Ranker wins by Theorem 4.2.

Now suppose the first round is high. We continue use of the method and notation from the proof of Theorem 4.2, visiting the vertices of \( P_n \) in some sequence \( v_{p_1}, \ldots, v_{p_n} \). Let \( p_1 = 3, p_2 = 2, p_3 = 1 \), and \( p_i = i \) for \( 4 \leq i \leq n \). For \( 1 \leq i \leq n \), if \( i \) is the least index such that \( v_{p_i} \in T \), then \( \sigma_{g_i}(V^{<p_i}) < 1 \) and \( \sigma_{g_i}(V^{>p_i}) < 1 \). By Theorem 4.2, Ranker can win \( R^\pm(P_n, f) \) by removing \( v_{p_i} \) in the first round and then winning \( R^\pm(P^{<p_i}, g_i) \) and \( R^\pm(P^{>p_i}, g_i) \). \( \square \)

Corollary 4.5. For \( n \geq 5 \), there is a function \( f \) such that \( \sigma_f(V(P_n)) > 1 \) and Ranker wins \( R^\pm(P_n, f) \).

Proof. By the Proposition 4.4, if \( f(v_1) = 2, f(v_2) = 3, f(v_3) = 1 \), and \( f(v_4) = 3 \), then Ranker wins \( R^\pm(P_4, f) \). Let \( f(v_i) = i \) for \( 5 \leq i \leq n \), so \( \sigma_f(V(P_n)) = 17/16 - 2^{-n} \). By Lemma 2.5, Ranker wins \( R^\pm(P_n, f) \). \( \square \)

The natural remaining question concerns what happens when \( \sigma_f(V(P_n)) \) is large.

Conjecture 4.6. There exists a real number \( \alpha \) such that for any positive integer \( n \), if \( \sigma_f(V(P_n)) > \alpha \), then Taxer wins \( R^\pm(P_n, f) \).

5. Cycles

We now turn our attention to cycles, for which the following results are proved using many of the techniques and results of Section 4. To prove our second main result, that \( \rho_f^c(C_n) = 1 + \lceil \log n \rceil = \rho(C_n) \), we once again prove a stronger statement, though we begin with a technical lemma.

Let \( X \) be a set of vertices and \( f \) be a function \( f : X \to \mathbb{N} \), with the elements \( x_1, \ldots, x_n \) of \( X \) named so that \( f(x_1) \leq \cdots \leq f(x_n) \). Set \( k = n \) if \( f(x_1) < \cdots < f(x_n) \) and otherwise let \( k \) satisfy \( f(x_1) < \cdots < f(x_k) = f(x_{k+1}) \). For \( X' = \{x_i \in X : 1 \leq i < k \} \) and \( X'' = \{x_i \in X : k < i \leq n \} \), define \( \tau_f(X) = \sigma_f(X') + 2\sigma_f(X'') \) (recalling that \( \sigma_f(V) = \sum_{v \in V} 2^{-f(v)} \)).
Lemma 5.1. If $\tau_f(V(C_n)) < 1$, then Ranker wins $R^\pm(C_n, f)$ whenever Taxer declares the first round high.

Proof. Recall that $R^\pm(C_n, f)$ starts with each $v \in V(C_n)$ having $f(v)$ tokens. If Taxer begins play by declaring the first round high, then Taxer takes one token from each element of a nonempty set $T \subseteq V(C_n)$, and Ranker responds by choosing some $u \in T$ to delete from $C_n$ to get a path on $n - 1$ vertices. Set $Y = V(C_n) - T$ and $Z = T - \{u\}$, and define $\tau' = \sigma_f(Y) + 2\sigma_f(Z)$. Thus after the first round the game is reduced to $R^\pm(P_{n-1}, g)$, where $V(P_{n-1}) = V(C_n) - \{u\}$ and $g(v) = f(v)$ for $v \in Y$ and $g(v) = f(v) - 1$ for $v \in Z$. Note that $\sigma_g(V(P_{n-1})) = \tau'$, so by Theorem 4.2 Ranker wins $R^\pm(C_n, f)$ if Ranker can always choose some $u \in T$ so that $\tau' < 1$.

Letting $V(C_n) = \{v_1, \ldots, v_n\}$, named so that $f(v_1) \leq \cdots \leq f(v_n)$, we note that if $f(v_i) < f(v_{i+1})$ for $1 \leq i < n$, then Ranker wins $R^\pm(C_n, f)$ by Lemma 2.5. Thus we may assume $f$ is not injective, and let $k$ be the least index such that $f(v_k) = f(v_{k+1})$. We complete the proof by showing that the minimum value Ranker can make $\tau'$ (given Taxer’s choice of $T$) is maximized when $T = \{v_k, \ldots, v_n\}$. Indeed, for that choice of $T$, Ranker can set $u = v_k$ to get $\tau' = \tau_f(V(C_n)) < 1$, so for any other $T$ Ranker could choose some $u \in T$ so that $\tau' < 1$.

Claim. The minimum value Ranker can make $\tau'$ (given Taxer’s choice of $T$) is maximized when $T = \{v_k, \ldots, v_n\}$.

Proof of claim. Suppose Taxer has chosen $T$ to maximize the minimum value Ranker is able to make $\tau'$ through the selection of $u$. For any choice of $T$, Ranker minimizes $\tau'$ by selecting $u$ as the vertex in $T$ having the fewest tokens. Thus we may assume Taxer selects $T = \{v_j, \ldots, v_n\}$ for some $j$ satisfying either $j = 1$ or $f(v_{j-1}) < f(v_j)$, since elements of $Z$ get counted double in $\tau'$. Given this choice of $T$, Ranker will select $u = v_j$ to get $\tau' = \sigma_f(Y) + 2\sigma_f(Z)$ for $Y = \{v_1, \ldots, v_{j-1}\}$ and $Z = \{v_{j+1}, \ldots, v_n\}$. We prove the claim by showing Taxer maximizes $\tau'$ by setting $j = k$.

We first show that $j \leq k$ by showing that if $f(v_i) = f(v_{i+1}) < f(v_j)$, then Taxer could increase $\tau'$ by $2^{1-f(v_i)}$ by adding $v_i$ and $v_{i+1}$ to $T$. Indeed, Ranker would choose $u = v_i$, with $\tau'$ losing $2^{1-f(v_i)}$ by removing $v_i$ and $v_{i+1}$ from $Y$ but gaining $2^{1-f(v_i)} + 2^{1-f(v_j)}$ by adding $v_{i+1}$ and $v_j$ to $Z$.

We now show that if $j < k$, then Taxer would not decrease $\tau'$ by removing $v_j, \ldots, v_{k-1}$ from $T$. Indeed, $\tau'$ would gain $2^{-f(v_j)}$ by adding $v_j$ to $Y$ and
only lose $2^{1-f(v_k)} + \sum_{d=j+1}^{k-1} 2^{-f(v_d)}$ by removing $v_k$ from $Z$ and switching $v_{j+1}, \ldots, v_{k-1}$ from $Z$ to $Y$, and we have

$2^{1-f(v_k)} + \sum_{d=j+1}^{k-1} 2^{-f(v_d)} \leq 2^{1-f(v_{k-j}+1)} + \sum_{d=1}^{k-j-1} 2^{-f(v_{j-d})}$

$= 2^{-f(v_j)} (2^{-k+j} + \sum_{d=1}^{k-j} 2^{-d})$

$= 2^{-f(v_j)}$

since $f(v_d) \geq f(v_j) + d - j$ for $j \leq d \leq k$.

Hence Ranker can guarantee $\tau' < 1$, so Ranker wins. \hfill ■

**Corollary 5.2.** If $\sigma_f(V(C_n)) < 1/2 + 2^{-\lceil \log n \rceil}$, then Ranker wins $R^\pm(C_n, f)$ whenever Taxer declares the first round high.

**Proof.** By Lemma 5.1 it suffices to show $\tau_f(V(C_n)) < 1$. Assume without loss of generality that $f(v_1) < \cdots < f(v_k) = f(v_{k+1}) \leq \cdots \leq f(v_n)$. Set $Y = \{v_1, \ldots, v_{k-1}\}$ and $Z = \{v_{k+1}, \ldots, v_n\}$. If $f(v_k) \leq \lceil \log n \rceil$, then

$\tau_f(V(C_n)) \leq 2\sigma_f(V(C_n)) - 2^{1-f(v_k)} < 1 + 2^{1-\lceil \log n \rceil} - 2^{1-\lceil \log n \rceil} = 1$.

If $f(v_k) \geq 1 + \lceil \log n \rceil$, then we have

$\tau_f(V(C_n)) = \sigma_f(Y) + 2\sigma_f(Z)$

$= \sigma_f(V(C_n)) - 2^{-f(v_k)} + \sigma_f(Z)$

$< 1/2 + 2^{-\lceil \log n \rceil} - 2^{-f(v_k)} + (n-1)2^{-f(v_k)}$

$= 1/2 + 2^{-\lceil \log n \rceil} + (n-2)2^{-f(v_k)}$

$\leq 1/2 + 2^{-\lceil \log n \rceil} + (n-2)2^{-1-\lceil \log n \rceil}$

$= 1/2 + n2^{-1-\lceil \log n \rceil}$

$\leq 1$.

\hfill \Box

**Theorem 5.3.** If $\sigma_f(V(C_n)) < 1/2 + 2^{-\lceil \log n \rceil}$, then Ranker wins $R^\pm(C_n, f)$. 

25
Proof. By Corollary 5.2 it suffices to show Ranker wins whenever Taxer declares the first round low, which we do using induction on $n$. Let $V(C_n) = \{v_1, \ldots, v_n\}$ and $E(C_n) = \{v_i v_{i+1} : 1 \leq i \leq n\}$. Note that $C_3$ is online $f$-list rankable when $\sigma_f(V(C_3)) < 3/4$: if $f(v_1) \leq f(v_2) \leq f(v_3)$ and $\sigma_f(V(C_3)) < 3/4$, then $f(v_i) \geq i$ for each $i$, so by Lemma 2.5 Ranker can win $R^\pm(C_3, f)$. Now suppose Ranker wins $R^\pm(C_m, g)$ for $3 \leq m < n$ and $\sigma_g(V(C_m)) < 1/2 + 2^{-[\log m]}$.

Recall that $R^\pm(C_n, f)$ starts with each $v_i$ having $f(v_i)$ tokens. If Taxer begins play by declaring the first round low, then Taxer takes one token from each element of a nonempty set $T \subseteq V(C_n)$, and Ranker responds by choosing an independent set $R \subseteq T$ to remove from $C_n$, replacing each removed vertex with an edge between its neighbors to get a cycle on $n - |R|$ vertices (or an edge if $|R| = n - 2$). Let $B = \{v_{2i-1} : 1 \leq i \leq \lceil n/2 \rceil\}$ and $C = \{v_{2i} : 1 \leq i \leq \lfloor n/2 \rfloor\}$.

Claim. If $T = V(C_n)$ and $n$ is odd, then Ranker can win.

Proof of claim. Without loss of generality assume $f(v_1) \geq f(v_i)$ for $1 \leq i \leq n$. Hence $f(v_n) \geq 1 + [\log n]$, since otherwise we would have

$$\sigma_f(T) \geq (n - 1)2^{-[\log n]} + 2^{-[\log n]} \geq 1/2 + 2^{-[\log n]}.$$  

Set $R = B - \{v_n\}$ if $\sigma_f(B - \{v_n\}) \geq \sigma_f(C)$ and $R = C$ otherwise, so $R$ is independent and the game reduces to $R^\pm(C_{(n+1)/2}, f - 1)$, where $V(C_{(n+1)/2}) = T - R$ and $2\sigma_f(R) \geq \sigma_f(T) - 2^{-f(v_n)}$. By our inductive hypothesis Ranker wins, due to the fact that

$$\sigma_{f-1}(T - R) = 2\sigma_f(T) - 2\sigma_f(R) \leq \sigma_f(T) + 2^{-f(v_n)} < 1/2 + 2^{-[\log n]} + 2^{-1-[\log n]} < 1/2 + 2^{1-[\log n]} = 1/2 + 2^{-[\log((n+1)/2)]}.$$ 

Claim. If $T \neq V(C_n)$ or $n$ is even, then Ranker can win.

Proof of claim. Without loss of generality assume $v_n \notin T$ if $n$ is odd (unlike in the proof of the previous claim, we are no longer assuming $f(v_n) \geq f(v_i)$
for \(1 \leq i \leq n\). Set \(R = B \cap T\) if \(\sigma_f(B \cap T) \geq \sigma_f(C \cap T)\) and \(R = C \cap T\) otherwise, so \(R\) is independent and \(\sigma_f(R) \geq \sigma_f(T - R)\). The game reduces to \(R^\pm(C_{n-|R|}, g)\), where \(V(C_{n-|R|}) = V(C_n) - R\) and \(g(v) = f(v) - |T \cap \{v\}|\).

By our inductive hypothesis Ranker wins, as

\[
\sigma_g(V(C_n) - R) = \sigma_f(V(C_n)) + \sigma_f(T - R) - \sigma_f(R) \\
\leq \sigma_f(V(C_n)) \\
< 1/2 + 2^{-\lfloor \log n \rfloor} \\
< 1/2 + 2^{-\lfloor \log(n-|R|) \rfloor}.
\]

Hence Ranker wins for all possible \(T\) and \(n\). \(\square\)

Once again we want to explore the boundary case.

**Proposition 5.4.** For \(n \geq 3\), there is a function \(f\) such that \(\tau_f(V(C_n)) = 1\) and \(\sigma_f(V(C_n)) = 1/2 + 2^{-\lfloor \log n \rfloor}\) but \(C_n\) is not \(f\)-list rankable.

**Proof.** Note that if \(f(v) = k+1\) for each \(v \in V(C_{2k+1})\), then \(\tau_f(V(C_{2k+1})) = 1\) and \(\sigma_f(V(C_{2k+1})) = 1/2 + 2^{-\lceil \log(2k+1) \rceil}\), but \(C_{2k+1}\) is not even \(f\)-rankable (as stated in [2]). By Corollary 3.8, if \(C_n\) is not \(f\)-list rankable and \(C_{n+1}\) has the same vertices as \(C_n\) except for replacing some vertex \(w \in V(C_n)\) with adjacent vertices \(u\) and \(v\) in \(C_{n+1}\), then setting \(f(u) = f(v) = f(w) + 1\) precludes \(C_{n+1}\) from being \(f\)-list rankable. If we choose \(w\) to maximize \(f\) on \(V(C_n)\), then \(\tau_f(V(C_n)) = \tau_f(V(C_{n+1}))\) and \(\sigma_f(V(C_n)) = \sigma_f(V(C_{n+1}))\), so the proposition follows. \(\square\)

**Proposition 5.5.** For \(n \geq 6\), there is a function \(f\) such that \(\tau_f(V(C_n)) = 1\) and \(C_n\) is \(f\)-list rankable.

**Proof.** Let \(V(C_n) = \{v_1, \ldots, v_n\}\) and \(E(C_n) = \{v_i, v_{i+1} : 1 \leq i \leq n\}\), with \(f(v_1) = 1, f(v_2) = 3, f(v_3) = 4, f(v_4) = 2, f(v_i) = i\) for \(5 \leq i < n\), and \(f(v_n) = n - 1\). Let \(L\) be an \(f\)-list assignment, and over \(\bigcup_{i=1}^n L(v_i)\) let \(m\) be the largest value, \(m'\) be the second largest value, \(b\) be the smallest value, and \(b'\) be the second smallest value found. We wish to find a ranking of \(C_n\) such that each \(v_i\) is labeled with \(a_i \in L(v_i)\).

**Case 1.** \(L(v_1) \cap \{m, m'\} \neq \emptyset\).

27
We can create an $L$-ranking of $C_n$ by letting $a_1 \in L(v_1) \cap \{m, m'\}$ and then ranking the path $P$ induced by $\{v_2, \ldots, v_n\}$ so that $a_i \in L(v_i) - \{a_1\}$ for $2 \leq i \leq n$. Indeed, it was shown in Proposition 4.4 that $P$ is $(f - 1)$-list rankable, and our ranking of $P$ labels no vertex with $a_1$ and at most one vertex with a label greater than $a_1$.

**Case 2.** $L(v_n) \neq \bigcup_{i=1}^{n-1} L(v_i)$.

Let $j$ be the least index such that $L(v_j) - L(v_n) \neq \emptyset$. We create an $L$-ranking by choosing labels $a_i \in L(v_i)$ in the order $i = 1, 4, 2, 3, 5, \ldots, n$ such that each $a_i$ is distinct from its predecessors and $a_j \notin L(V_n)$. We can do this for $1 \leq i < n$ since by the time $a_i$ is to be chosen only $f(v_i) - 1$ previous labels will have been used, and we can choose $a_n \in L(v_n) - \{a_1, \ldots, a_{n-1}\}$ since $|L(v_n)| = n - 1$ and $a_j \notin L(v_n)$.

**Case 3.** $b \in \bigcup_{i=1}^{n-2} L(v_i)$.

We can choose $a_1, \ldots, a_{n-2}$ from $L(v_1), \ldots, L(v_{n-2})$ to be distinct and contain $b$, leaving some $a \in L(v_{n-1}) - \{a_1, \ldots, a_{n-2}\}$ with $a > b$. We may assume $a \in L(v_n)$ (otherwise Case 2 applies), so we can complete an $L$-ranking by either setting $a_{n-1} = a$ and $a_n = b$ if $a_1 \neq b$ or setting $a_{n-1} = b$ and $a_n = a$ if $a_{n-2} \neq b$.

**Case 4.** $L(v_1) = \{b'\}$.

We may assume $b \notin L(v_2)$ (otherwise Case 3 applies), so finding an $L$-ranking of $C_n$ reduces to finding an $L'$-ranking of the cycle $C_{n-1}$ created by deleting $v_1$ and adding the edge $v_n v_2$, where $L'(v_i) = L(v_i) - \{b'\}$ for $i \in \{2, n-1, n\}$ and $L'(v_i) = L(v_i)$ for $3 \leq i \leq n-2$. Indeed, we would have $a_2 > b'$ (since $L'(v_2) \cap \{b, b'\} = \emptyset$) and $\max\{a_{n-1}, a_n\} > b'$ (since $b' \notin L'(v_{n-1}) \cup L'(v_n)$ and $a_{n-1} \neq a_n$), so the $L'$-ranking of $C_{n-1}$ could be turned into an $L$-ranking of $C_n$ by setting $a_1 = b'$. If $f'(v_i) = f(v_i) - 1$ for $i \in \{2, n-1, n\}$ and $f'(v_i) = f(v_i)$ for $3 \leq i \leq n-2$, then $|L'(v_i)| \geq f'(v_i)$
for each $i$, and we have

$$
\tau'_f(V(C_{n-1})) = 2(1/4 + \left(\sum_{i=4}^{n-2} 2^{-i}\right) + 2^{2-n} + 2^{2-n})
$$

$$
\leq 2(3/8 + 2^{2-n})
$$

$$
\leq 2(3/8 + 1/16)
$$

$$
= 7/8
$$

$$
< 1.
$$

By Lemma 5.1, $C_{n-1}$ is $f'$-list rankable, giving us an $L'$-ranking of $C_{n-1}$ and thus an $L$-ranking of $C_n$.

**Case 5.** $n = 6$.

Without loss of generality assume $L(v_5) = L(v_6) = \{1, 2, 3, 4, 5\}$ (since $f(v_5) = f(v_6) = 5$ and Case 2 applies if $L(v_6) \neq \bigcup_{i=1}^{5} L(v_i)$), so $b = 1$, $b' = 2$, $m' = 4$, and $m = 5$. Since Case 3 applies if $1 \in \bigcup_{i=1}^{4} L(v_i)$, we may assume that $L(v_1) = \{3\}$ (Case 1 applies if $L(v_1) \cap \{4, 5\} \neq \emptyset$, and Case 4 applies if $L(v_1) = \{2\}$), $L(v_2)$ contains 2 or 3 as well as 4 or 5, $L(v_3) = \{2, 3, 4, 5\}$, and either $L(v_4) = \{4, 5\}$ or $L(v_4)$ contains 2 or 3. If $2 \in L(v_4)$, let $a_1 = 3$, $a_2 = 4$ (or $a_2 = 5$, if $4 \notin L(v_2)$), $a_3 = 3$, $a_4 = 2$, $a_5 = 5$ (or $a_5 = 4$, if $4 \notin L(v_2)$), and $a_6 = 1$. If $3 \in L(v_4)$, let $a_1 = 3$, $a_2 = 4$ (or $a_2 = 5$ if $4 \notin L(v_2)$), $a_3 = 2$, $a_4 = 3$, $a_5 = 5$ (or $a_5 = 4$, if $4 \notin L(v_2)$), and $a_6 = 1$. If $L(v_4) = \{4, 5\}$, let $a_1 = 3$, $a_2 = 5$ (or $a_2 = 4$, if $4 \notin L(v_2)$), $a_3 = 2$, $a_4 = 5$ (or $a_4 = 4$, if $4 \notin L(v_2)$), $a_5 = 2$, and $a_6 = 1$.

**Case 6.** $n \geq 7$.

We perform induction on $n$, using Case 5 as the base case. Thus we assume the cycle $C_{n-1}$ created by deleting $v_{n-2}$ and adding the edge $v_{n-3}v_{n-1}$ is $f'$-list rankable, where $f'(v_i) = f(v_i) - 1$ for $i \in \{n-1, n\}$ and $f'$ elsewhere. We may also assume $b \in (L(v_{n-1}) \cap L(v_n)) - (L(v_1) \cup L(v_{n-2}))$ and $b' \in (L(v_{n-1}) \cap L(v_n)) - L(v_1)$ (otherwise Case 3 or 4 applies). If $a_1, \ldots, a_{n-2}$ can be chosen from $L(v_1), \ldots, L(v_{n-2})$ to be distinct such that $a_{n-2} \neq b'$, then we can complete an $L$-ranking of $C_n$ by setting $a_{n-1} = b'$ and $a_n = b$.

If $a_1, \ldots, a_{n-2}$ cannot be chosen from $L(v_1), \ldots, L(v_{n-2})$ to be distinct such that $a_{n-2} \neq b'$, then the only lists containing $b'$ are $L(v_{n-2})$, $L(v_{n-1})$, and $L(v_n)$. By our inductive hypothesis, if $L'(v_i) = L(v_i)$ for $1 \leq i \leq n-3$ and $L'(v_i) = L(v_i) - \{b'\}$ for $i \in \{n-1, n\}$, then the cycle $C_{n-1}$ created by
deleting \(v_{n-2}\) and adding the edge \(v_{n-3}v_{n-1}\) has an \(L'\)-ranking, which we can extend to an \(L\)-ranking of \(C_n\) by setting \(a_{n-2} = b'\).

**Corollary 5.6.** For \(n \geq 7\), there is a function \(f\) such that \(\tau_f(V(C_n)) > 1\) and \(C_n\) is \(f\)-list rankable.

**Proof.** By the above proposition, if \(f = (1, 3, 4, 2, 5, 5)\) then \(C_6\) is \(f\)-list rankable. Let \(f(v_i) = i\) for \(7 \leq i \leq n\); then \(\tau_f(V(C_n)) = 33/32 - 2^{1-n}\) and \(C_n\) is \(f\)-list rankable by Lemma 2.5.

Conjecture 4.5 says that if \(\sigma_f(V(P_n))\) is large enough, then \(P_n\) is not \(f\)-list rankable. Since deleting an edge of \(C_n\) leaves a copy of \(P_n\), and \(\tau_f(V(C_n)) < 2\sigma_f(V(C_n))\), Conjecture 4.5 would also imply that \(C_n\) is not \(f\)-list rankable for large enough \(\tau_f(V(C_n))\) or \(\sigma_f(V(C_n))\).

6. Trees With Many Leaves

We now prove our third main result, that \(\rho_\ell(T) = q\) if \(T\) is a tree having \(p\) internal vertices and \(q\) leaves, where \(q \geq 2^{p+2} - 2p - 4\). Since Proposition 1.5 implies that \(\rho_\ell(T) \geq q\) for any tree \(T\) with \(q\) leaves, we need only prove the upper bound. We consider separately trees with two or fewer internal vertices. Recall that a **star** is a tree having at most one internal vertex.

**Proposition 6.1.** If \(S\) is a star with \(q\) leaves, then \(\rho^+(S) = q\).

**Proof.** We use induction on \(q\) to show Ranker has a winning strategy for the game \(R^+(S, f)\), where \(f = q\) everywhere. The statement is obvious if \(S \in \{P_1, P_2\}\), and it follows from Theorem 4.2 if \(S = P_3\). Thus we may assume \(q \geq 3\) and Ranker wins for stars having fewer than \(q\) leaves.

If Taxer takes a token from the internal vertex in the first round, let Ranker respond by removing it; then \(q\) isolated vertices remain, each with at least \(q - 1\) tokens, so Ranker can win the game. If Taxer takes tokens from only leaves in the first round, then let Ranker respond by removing a leaf, leaving a star with \(q - 1\) leaves and at least \(q - 1\) tokens on each vertex. By the inductive hypothesis Ranker can win the game.

A **double star** is a tree having exactly two internal vertices.

**Proposition 6.2.** If \(T\) is a double star with \(q\) leaves \((q \geq 3)\), then \(\rho^+_\ell(T) = q\).
Proof. We show Ranker has a winning strategy for the game $R^+(T, f)$, where $f = q$ everywhere. Let the (adjacent) internal vertices of $T$ be $x$ and $y$, with $x$ adjacent to leaves $x_1, \ldots, x_m$ and $y$ adjacent to leaves $y_1, \ldots, y_n$ (so $q = m + n \geq 3$). We may assume $m \leq n$, so by hypothesis $n \geq 2$. If Taxer selects an internal vertex in the first round, let Ranker respond by removing an internal vertex, leaving behind isolated vertices and a star with at most $q - 1$ leaves. Each remaining vertex still has at least $q - 1$ tokens, so by Proposition 6.1 Ranker can win the game.

Thus we may assume that Taxer selects only leaves in the first round, with Ranker responding by removing a selected leaf. We use induction on $q$ to show Ranker has a winning strategy. If $q = 3$, then $m = 1$ and $n = 2$. If Ranker removes $y_1$ or $y_2$, then a path on four vertices remains, with the internal vertices having three tokens each and the leaves having at least two tokens each; by Theorem 4.2 Ranker can win the game. If Taxer only selects $x_1$ and Ranker removes $x_1$, then a star with three leaves remains, with each vertex having three tokens; by Proposition 6.1 Ranker can win this game.

Now assume $q \geq 4$ and Ranker wins for trees having two internal vertices and between three and $q - 1$ leaves. If Ranker removes some $y_i$, then a tree with $q - 1$ leaves and two internal vertices remains, with each vertex having at least $q - 1$ tokens; by the inductive hypothesis Ranker can win this game. If Taxer only selects $x_i$ and Ranker removes $x_i$, leaving behind either a tree with two internal vertices and $q - 1$ leaves, with each vertex having at least $q - 1$ tokens, or a star with $q$ leaves, with each vertex having $q$ tokens. Either way Ranker can win this game. □

**Theorem 6.3.** For any tree $T$ having $p$ internal vertices and $q$ leaves, if $q \geq 2^{p+2} - 2p - 4$, then $\rho_\ell(T) = q$.

Proof. If $p < \min\{3, q\}$, then $\rho_\ell(T) \leq \rho_\ell^+(T) = q$ by Propositions 6.1 and 6.2, so we may assume $p \geq 3$. If $T$ is a tree with $p$ internal vertices and $q$ leaves, with $q \geq 2^{p+2} - 2p - 4$, then $T$ has a vertex of degree at least 3. For an internal vertex $u$, if $u$ is a vertex of degree at least 3, or is adjacent to one, or is located on a path whose endpoints each have degree at least 3 in $T$, let $T_u$ be a component of $T - u$ containing the most leaves of $T$. If $u$ is any other internal vertex, let $T_u = T_w$, where $w$ is the unique vertex nearest to $u$ that has degree 2 and is adjacent to a vertex of degree at least 3.

For each internal vertex $u$, say $T_u$ has $p_u$ internal vertices and $q_u$ leaves, $q_u' \leq q_u < q$. Let $v$ be an internal vertex such that $q_u'$ is minimum. For any internal vertex $u$ besides
we have \( q'_v \geq q/2 \) since either \( q'_v \geq q/2 \), in which case \( q'_u \geq q/2 \) by the minimality of \( q'_v \), or \( q'_v < q/2 \), in which case any subtree of \( T \) obtained by deleting from \( T \) a component of \( T - v \) must have more than \( q/2 \) leaves of \( T \). Thus \( v \) has degree at least 3, and \( T_u \) contains the subtree of \( T \) obtained by deleting the component of \( T - v \) containing \( u \), giving \( T_u \) more than \( q/2 \) leaves of \( T \).

Let \( L \) be a \( q \)-uniform list assignment to \( T \), and for any internal vertex \( u \) let \( m_u \) denote the largest element of \( L(u) \). Call an internal vertex \( u \) special if \( q'_u \geq q/2 \) (that is, if \( u \neq v \) or \( u = v \) and \( q'_v \geq q/2 \)) and there are vertices \( u_1, \ldots, u_p \) that are leaves of both \( T \) and \( T_u \) and satisfy the following properties: each internal vertex in \( T_u \) has at least two neighbors in \( T_u \) that are not one of these leaves, and from each \( L(u_i) \) we can select some \( e_i \) such that \( e_1 < \cdots < e_p < m_u \). We classify \( L \) by whether \( L \) admits a special vertex, and in each case we show how to give \( T \) an \( L \)-ranking.

**Case 1.** \( L \) admits no special vertex.

If \( q'_v < q/2 \) and \( m_v \) is the largest label in any list, label \( v \) with it; since no component of \( T - v \) can have \( q/2 + p - 1 \) vertices, and \( q/2 + p - 1 \leq q - 1 \), a ranking can be completed by deleting \( m_v \) from the list of any unlabeled vertex, and then for each component of \( T - v \) giving distinct labels to each vertex. Now assume \( q'_v \geq q/2 \) or \( m_v \) is not the largest label in any list. Give each leaf a separate label (which is possible since there are \( q \) leaves and each vertex receives a list of size \( q \)), making sure to give some leaf a label larger than \( m_v \) if possible.

Since for each internal vertex \( u \) besides \( v \), \( T_u \) contains fewer than \( p \) internal vertices of \( T \) and at least \( q/2 \) leaves of \( T \), we can fix a set \( S_u \) of at least \( q/2 - p + 1 \) leaves of both \( T \) and \( T_u \) such that each internal vertex in \( T_u \) has at least two neighbors in \( T_u \) that are not in \( S_u \) (to get \( S_u \), start with the set of leaves in both \( T \) and \( T_u \), and delete one leaf adjacent to each of the at most \( p - 1 \) internal vertices of \( T_u \) adjacent to a leaf of \( T \) in \( T_u \), and delete an additional leaf if \( T_u \) is a star).

For each internal vertex \( u \) besides \( v \), \( L(u) \) contains at most \( p \) of the labels used on \( S_u \), since otherwise \( u \) would be a special vertex (with \( u_1, \ldots, u_p \) being the elements of \( S_u \) receiving the smallest labels). Then \( L(u) \) contains at most \( q - ((q/2 - p + 1) - p) \), or \( q/2 + 2p - 1 \), of the labels used on the leaves of \( T \), so deleting from each \( L(u) \) the labels used on the leaves of \( T \) yields a list of size at least \( q - (q/2 + 2p - 1) \), or \( q/2 - 2p + 1 \), which is greater than \( p \)
for \( p \geq 3 \). If \( q'_v \geq q/2 \), then the same holds for \( L(v) \), and we can complete a ranking by giving distinct labels to each of the \( p \) internal vertices.

If \( q'_v < q/2 \), then by hypothesis \( m_w \) is not the largest label in any list. In this case the largest label must be in the list of some leaf (or else the internal vertex \( u \) containing that label would be a special vertex, with \( u_1, \ldots, u_p \) being any \( p \) elements of \( S_u \), and we assigned that label to such a leaf. Thus we can complete a ranking by labeling \( v \) distinctly from the leaves (possible since \( |L(v)| = q \) and one of the \( q \) leaves was given a label not in \( L(v) \)) and then labeling the remaining internal vertices distinctly.

**Case 2.** \( L \) admits a special vertex \( u \).

We use induction on \( p \); assume \( p \geq 3 \). If \( u \) has degree 2 and is not adjacent to any vertex of degree at least 3, and some component of \( T - u \) is a path, then let \( w \) be the vertex nearest to \( u \) that has degree 2 and is adjacent to a vertex of degree at least 3, and without loss of generality assume no vertex between \( u \) and \( w \) is special (if such a special vertex existed then we could just use the closest one to \( w \) instead of using \( u \)). Label \( u \) with \( m_u \) and each \( u_i \) with \( e_i \), so by the positioning of \( u \) and the size of its label, no label given to a vertex separated from \( T_u \) by \( u \) can cause a conflict with the label of any \( u_i \). Let \( A \) be the set of vertices separated from \( T_u \) by \( u \), and let \( A' \) be the (possibly empty) set of vertices strictly between \( u \) and \( T_u \); see Figure 4.

![Figure 4: A possibility for \( T \), if \( u \) is a special vertex and \( A' \neq \emptyset \).](image)

We complete the proof via a sequence of three claims that show a ranking of \( T \) can be completed in the following way: first label the rest of \( T_u \) without using any of the previously used labels, then label the vertices of \( A \) without using \( m_u \) or any labels larger than \( m_u \) that were used on \( T_u \) (these labels
cannot come into conflict with any labels given to $T_u$ since they are separated from each other by the label $m_u$ given to $u$), and finally label any possible vertices in $A'$ with labels unused on the rest of $T$.

**Claim.** We can finish ranking $T_u$ without using the labels $m_u, e_1, \ldots, e_p$.

**Proof of claim.** If we delete $m_u, e_1, \ldots, e_p$ from the list of each of the $p_u + q_u - p$ unlabeled vertices in $T_u$, then each such list must still have size at least $q - p - 1$, and thus we can finish ranking $T_u$ if the subtree induced by its unlabeled vertices is $(q - p - 1)$-list rankable (since no vertex already labeled could be part of a path in $T_u$ between vertices with the same label, once the remaining vertices are labeled from their truncated lists). By our inductive hypothesis this subtree is in fact $(q - p - 1)$-list rankable, since it has $p_u$ internal vertices and $q_u - p$ leaves, with $p_u \leq p - 1$ and

$$q_u - p \geq q/2 - p \geq (2^{p+2} - 2p - 4)/2 - p = 2^{p+1} - 2p - 2 = 2^{(p-1)+2} - 2(p-1) - 4.$$  

\[\blacksquare\]

**Claim.** We can rank $A$ without using $m_u$ or any labels larger than $m_u$ that were used on $T_u$.

**Proof of claim.** Let $b$ denote the minimum size of a list assigned to a vertex in $A$ after deleting $m_u$ as well as any labels larger than $m_u$ that were used on $T_u$. We prove the claim by showing $|A| \leq b$, since then a ranking of the vertices of $A$ can be completed trivially by giving them distinct labels from their truncated lists (these labels cannot come into conflict with any labels given to $T_u$ since they are separated from each other by the label $m_u$ given to $u$). We have $|A| = p + q - p_u - q_u - |A'| - 1$, since $T$ has $p + q$ vertices and $A$ does not include the $p_u + q_u$ vertices of $T_u$, nor the vertices of $A'$, nor $u$. We also have $b \geq q - (p_u + q_u - p + 1)$, since each list assigned to an unlabeled vertex started out with $q$ elements, and the only ones that could have been deleted were $m_u$ as well as any of the at most $p_u + q_u - p$ labels given to $T_u$ that exceeded $m_u$. Thus $|A| \leq b - |A'|$.  

\[\blacksquare\]

**Claim.** We can rank $A'$ using labels previously unused on the rest of $T$.

**Proof of claim.** Let $b'$ denote the minimum size of a list assigned to a vertex in $A'$ after deleting any labels used on the rest of $T$. We prove the claim by showing $|A'| \leq b'$, since then a ranking of $T$ can be completed trivially by giving each of the vertices in $A'$ distinct and previously unused labels. Since
for each \( z \in A' \), \( T_z \) contains fewer than \( p \) internal vertices of \( T \) and exactly \( q - 1 \) leaves of \( T \), we can fix a set \( S_z \) of at least \( q - p \) leaves of both \( T \) and \( T_z \) such that each internal vertex in \( T_z \) has at least two neighbors in \( T_z \) that are not in \( S_z \) (to get \( S_z \), start with the set of leaves in both \( T \) and \( T_z \), and delete one leaf adjacent to each of the at most \( p - 1 \) internal vertices of \( T_z \) adjacent to a leaf of \( T \) in \( T_z \), and delete an additional leaf if \( T_z \) is a star). For each \( z \in A' \), \( L(z) \) contains at most \( p \) of the labels used on \( S_z \), since otherwise \( z \) would be a special vertex (with \( z_1, \ldots, z_p \) being the elements of \( S_z \) receiving the smallest labels). Then \( L(z) \) contains at most \( 2p - 1 \) of the labels used on the leaves of \( T_z \), so deleting from each \( L(z) \) the labels used on the leaves of \( T_z \) or on any of the other \( p \) vertices of \( T \) besides \( z \) (i.e., the leaf of \( T \) not in \( T_z \) along with any of the \( p - 1 \) internal vertices of \( T \) besides \( z \)) yields a list of size at least \( q - 3p + 1 \). Thus we can finish ranking \( T \) because \( |A'| \leq p - 2 \leq q - 3p + 1 \leq b' \) for \( p \geq 3 \).

Hence \( T \) has an \( L \)-ranking regardless of whether \( L \) admits a special vertex \( u \).

We conclude by noting that no statement similar to Theorem 6.3 can be applied to graphs in general.

**Proposition 6.4.** For \( p \geq 3 \) and \( q \geq 0 \), there is a graph \( G \) with \( p \) internal vertices and \( q \) leaves such that \( \rho_L(G) > q \).

**Proof.** Let \( G \) be obtained by connecting \( q \) leaves to \( K_p \), the complete graph on \( p \) vertices, such that at least one internal vertex \( v \) is not adjacent to any leaf. Then \( G \) contains a spanning subtree of which \( v \) is a leaf; this tree has \( q + 1 \) leaves, so \( \rho_L(G) > q \) by Proposition 1.5.

7. Acknowledgements

Many thanks to Doug West for running the combinatorics REGS program and assisting with this project, to Robert Jamison for introducing the list ranking problem and providing initial guidance, and to the two anonymous referees for their helpful and clarifying suggestions.

The author acknowledges support from National Science Foundation grant DMS 0838434 EMSW21MCTP: Research Experience for Graduate Students.

[1] Bodlaender, H. L.; Deogun, J. S.; Jansen, K.; Kloks, T.; Kratsch, D.; Muller, H.; Tuza, Z. Rankings of graphs, Siam J. Discrete Math, Vol. 11, No. 1 (1998), 168–181.
[2] Bruoth, E.; Horňáč, M. Online-ranking numbers for cycles and paths, Discuss. Math. Graph Theory, 19 (1999), 175–197.

[3] Dereniowski, D. The complexity of list ranking of trees, Ars Combin. 86 (2008), 97–114.

[4] Dereniowski, D. Edge ranking and searching in partial orders, Discrete Appl. Math. 156(13) (2008), 2493–2500.

[5] Giannopoulou, A. C.; Hunter, P.; Thilikos, D. LIFO-search: a min-max theorem and a searching game for cycle-rank and tree-depth, Discrete Appl. Math. 160(15) (2012), 2089–2097.

[6] Isaak, G.; Jamison, R.; Narayan, D. Greedy rankings and arank numbers, Inform. Process. Lett. 109 (2009), no. 15, 825–827.

[7] Iyer, A. V.; Ratliff, H. D.; Vijayan, G. Optimal node ranking of trees, Inform. Process. Lett. 28 (1988), 225–229.

[8] Iyer, A. V.; Ratliff, H. D.; Vijayan, G. Parallel assembly of modular products, Tech. Rept. 88–06, Production and Distribution Research Center, Georgia Institute of Technology, Atlanta, GA (1988).

[9] Iyer, A. V.; Ratliff, H. D.; Vijayan, G. On an edge ranking problem of trees and graphs, Discrete Appl. Math., Vol. 30, Issue 1 (1991), 43–52.

[10] Jamison, R. E. Coloring parameters associated with rankings of graphs, Congr. Numer. 164 (2003), 111–127.

[11] McDonald, D. C. On-line vertex ranking of trees, SIAM J. Discrete Math. 29(1) (2015), 145–156.

[12] Nešetřil, J.; Ossona de Mendez, P. Grad and classes with bounded expansion I. Decompositions, European J. Combin. 29(3) (2008), 760–776.

[13] Nešetřil, J.; Ossona de Mendez, P. Grad and classes with bounded expansion II. Algorithmic aspects, European J. Combin. 29(3) (2008), 777–791.
[14] Nešetřil, J.; Ossona de Mendez, P. Grad and classes with bounded expansion III. Restricted graph homomorphism dualities, European J. Combin. 29(4) (2008), 1012–1024.

[15] Nešetřil, J.; Ossona de Mendez, P. Tree-depth, subgraph coloring and homomorphism bounds, European J. Combin. 27(6) (2006), 1022–1041.

[16] Nešetřil, J.; Ossona de Mendez, P.; Wood, D. R. Characterisations and examples of graph classes with bounded expansion, European J. Combin. 33(3) (2012), 350–373.

[17] Schauz, U. Mr. Paint and Mrs. Correct, Electron. J. Combin., Vol. 16, No. 1 (2009), R77.