Relaxing the Irrevocability Requirement for Online Graph Algorithms*

Joan Boyar
University of Southern Denmark

Lene M. Favrholdt
University of Southern Denmark

Michal Kotrbčík
The University of Queensland

Kim S. Larsen
University of Southern Denmark

February 2022

Abstract

Online graph problems are considered in models where the irrevocability requirement is relaxed. We consider the Late Accept model, where a request can be accepted at a later point, but any acceptance is irrevocable. Similarly, we consider a Late Reject model, where an accepted request can later be rejected, but any rejection is irrevocable (this is sometimes called preemption). Finally, we consider the Late Accept/Reject model, where late accepts and rejects are both allowed, but any late reject is irrevocable. We consider four classical graph problems: For Maximum Independent Set, the Late Accept/Reject model is necessary to obtain a constant competitive ratio, for Minimum Vertex Cover the Late Accept model is sufficient, and for Minimum Spanning Forest the Late Reject model is sufficient. The Maximum Matching problem admits constant competitive ratios in all cases. We also consider Maximum Acyclic Subgraph and Maximum Planar Subgraph, which exhibit patterns similar to Maximum Independent Set.

*This work was supported in part by the Danish Council for Independent Research, Natural Sciences, grant DFF-1323-00247, the Independent Research Fund Denmark, Natural Sciences, grants DFF-7014-00041 and DFF-0135-00018B, and the Villum Foundation, grant VKR023219. A preliminary version of the paper appeared in the 15th International Algorithms and Data Structures Symposium (WADS), volume 10389 of Lecture Notes in Computer Science, pages 217–228. Springer, 2017. This full version is to appear in Algorithmica.

1 Introduction

For an online problem, the input is a sequence of requests. For each request, the algorithm has to make some decision without any knowledge about possible future requests. Often (part of) the decision is whether to accept or reject the request and the decision is usually assumed to be irrevocable. However, many online problems have applications for which total irrevocability is not inherent or even realistic. Furthermore, when analyzing the quality of online algorithms, relaxations of the irrevocability constraint often result in dramatically different results, especially for graph problems. This has already been realized and several papers study various relaxations of the irrevocability requirement. In this paper we initiate a systematic study of the nature of irrevocability and of the implications for the performance of the algorithms. Our aim is to understand whether it is the lack of knowledge of the future or the irrevocability restrictions on the manipulation of the solution set that makes an online problem difficult.

We consider graph problems and focus on four classical ones, *Maximum Independent Set*, *Maximum Matching*, *Minimum Vertex Cover*, and *Minimum Spanning Forest*. It turns out that the techniques used for analyzing Maximum Independent Set can easily be generalized to similar problems like *Maximum Acyclic Graph* and *Maximum Planar Subgraph*. Maximum Independent Set and Minimum Vertex Cover are studied in the vertex arrival model. In this model, vertices arrive one by one together with all the edges between the newly arrived vertex and previous vertices. Maximum Matching and Minimum Spanning Forest are studied in the edge arrival model, but the results hold in the vertex arrival model as well. In the edge arrival model, edges arrive one by one, and if a vertex incident with the newly-arrived edge was not seen previously, it is also revealed.

As is customary for these problems, we consider simple graphs, i.e., undirected graphs without multi-edges or self-loops.

1.1 Relaxed irrevocability

For the problems considered in this paper, the online decision is whether to accept or reject the current request. In the *standard* model of online problems, this decision is irrevocable and has to be made without any knowledge about possible future requests. We relax the irrevocability requirement by allowing the algorithm to perform two additional operations, namely *late accept* and *late reject*. Late accept allows the algorithm to accept not only the current request but also requests that arrived earlier (at any time). Thus,

late accept relaxes irrevocability by not forcing the algorithm to discard the items that are not used immediately. Late reject allows the algorithm to remove items from the solution being constructed (at any time), relaxing the irrevocability of the decision to accept an item. When the algorithm is allowed to perform late accept or late reject, but not both, we speak of a *Late Accept model* and *Late Reject model*, respectively. Note that, in these two models, the late operations are irrevocable. We also consider the situation where the algorithm is allowed to perform *both* late accepts and late rejects, focusing on the *Late Accept/Reject model*, where any item can be late-accepted and late-rejected, but once it is late-rejected, this decision is irrevocable. In other words, if the algorithm performs both late accept and late reject on a single item, the late accept has to precede the late reject.

We believe that the Late Accept, Late Reject, and Late Accept/Reject models are appropriate modeling tools corresponding to many natural settings. Maximum Matching, for example, in the context of online gaming or chats, functions in the Late Accept model. Indeed, the users are in the pool until assigned, allowing the late accept, but once the users are paired, the connection should not be broken by the operator. Note that the matching problem is a maximization problem. For minimization problems, accepting a request may correspond to establishing a resource at some cost. Often there is no natural reason to require the establishment to happen at a specific time. Late acceptance was considered for the dominating set problem in [3], which also contains further feasible practical applications and additional rationale behind the model.

When the knapsack problem is studied in the Late Reject model, items are usually called *removable*; see for example [19, 16, 15, 7, 17]. For most other problems, late rejection is usually called *preemption* and has been studied in variants of many online problems, for example call control [2, 12], maximum coverage [33, 31], weighted matching problems [9, 10], and submodular maximization [6]. Preemption was also previously considered for one of the problems we consider here, independent set, in [25], but in a model where advice is used, presenting lower bounds on the amount of advice necessary to achieve given competitive ratios in a stated range. Also focusing on advice results, [32] investigates graph problems where node or edge deletions are carried out to keep the graph in some fixed graph class. These deletions can be deferred but are irrevocable when executed.

Online Minimum Vertex Cover was studied in [8], where the authors considered the possibility of swapping some of the accepted vertices for other vertices at the very end, at some cost depending on the number of vertices involved.

A similar concept has been studied in, for example, [18, 30, 13, 14] for online Steiner tree problems, MST, and TSP. Here, replacing an accepted edge with another is allowed, and the objective is to minimize the number of times this occurs while obtaining a good competitive ratio. The problem is said to allow *rearrangements* or *recourse*.

TSP has also been studied [20] in a model where the actual acceptances and rejections (rejections carry a cost) are made at any time.

In [1], which is a continuation of the work on matching from the present paper (the conference version [4]), the authors consider more levels of late accept/reject.

1.2 Competitive analysis

For each graph problem, we study online algorithms in the standard, Late Accept, Late Reject, and Late Accept/Reject models using the standard tool of competitive analysis, where the performance of an online algorithm is compared to the optimum algorithm OPT via the competitive ratio [34, 22] or the competitive function [5]. For any algorithm (online or offline), A, we let $A(\sigma)$ denote the value of the objective function when A is applied to the input sequence σ.

For minimization problems, an algorithm ALG is *c-competitive* if there exists a constant α such that, for all inputs σ,

$$\text{ALG}(\sigma) \leq c \cdot \text{OPT}(\sigma) + \alpha\,.$$

Similarly, for maximization problems, ALG is c-competitive, if there exists a constant α such that, for all inputs σ,

$$\text{OPT}(\sigma) \leq c \cdot \text{ALG}(\sigma) + \alpha\,.$$

In both cases, if the inequality holds for $\alpha = 0$, the algorithm is *strictly* c-competitive. The *(strict) competitive ratio* of ALG is the infimum over all c such that ALG is (strictly) c-competitive. The competitive ratio of a problem P is the infimum over the competitive ratios of all online algorithms for the problem.

For algorithms that do not have a constant competitive ratio, we use the competitive function as defined in [5], to emphasize the high order term in the function and the constant in front of it.

Most often in the area (including textbooks), people define the competitive ratio for constants only, but then use it for functions of n as well. Given the focus of this paper, we find it necessary to be precise about the mix of constants and asymptotic notation.

For minimization problems, an algorithm ALG is $f(n)$-*competitive*, if for all inputs σ of length n,

$$\text{ALG}(\sigma) \leq (f(n) + o(f(n)) \cdot \text{OPT}(\sigma)\,.$$

Similarly, for maximization problems, an algorithm ALG is $f(n)$-*competitive*, if for all inputs σ of length n,

$$\text{OPT}(\sigma) \leq (f(n) + o(f(n)) \cdot \text{ALG}(\sigma)\,.$$

An algorithm ALG has competitive function $f(n)$, if ALG is $f(n)$-competitive, and for any $g(n)$ such that ALG is $g(n)$-competitive,

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq 1\,.$$

A problem P has competitive function $f(n)$, if the problem has an $f(n)$-competitive algorithm, and for any $g(n)$-competitive algorithm for the problem,

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq 1\,.$$

For most combinations of the problem and the model, we obtain matching lower and upper bounds on the competitive ratio/function. For results on the competitive ratio, the upper bounds hold for the strict competitive ratio, and the lower bounds hold for the (non-strict) competitive ratio. For convenience, when stating results containing both an upper bound on the strict competitive ratio and a lower bound on the competitive ratio, we use the term "competitive ratio".

For ease of notation for our results, we adopt the following convention (see Theorem 2, for example). We say that a problem has competitive function $n/\Theta(1)$ if

- for any $f(n)$-competitive algorithm, there is a constant $b > 0$ such that

$$\lim_{n\to\infty} \frac{f(n)}{n/b} \geq 1\,,$$

  and

- for any constant b, there is an $f(n)$-competitive algorithm for graphs with at least b vertices, such that

$$\lim_{n\to\infty} \frac{f(n)}{n/b} \leq 1\,.$$

1.3 Our results

The paper shows that for some problems the Late Accept model allows for algorithms with significantly better competitive ratios, while for others it is the Late Reject model which does. For a third class of problems, the Late Accept/Reject model is necessary to get these improvements. Table 1 illustrates these varying patterns. The results for Maximum Independent Set are repeated in Table 2 together with the results for the two related problems. Note that only deterministic algorithms are considered, not randomized algorithms.

Our results on Minimum Spanning Forest follow from previous results. Thus, they are mainly included to give an example where late rejects bring down the competitive ratio dramatically. In fact, for any cost function, the problem of finding an independent set of minimum (maximum) cost for any matroid has competitive ratio 1 in both the Late Reject and the Late Accept/Reject models. The technical highlights of the paper are the results for Maximum Independent Set in the Late Accept/Reject model (Theorems 8 and 9) and Maximum Planar Subgraph in the Late Reject model (Theorems 5 and 6).

Table 1: Competitive ratios/functions of the four main problems in each of the four models under the standard model, as well as Late Accept, Late Reject, and Late Accept/Reject. W is the ratio of the largest weight to the smallest, and $\frac{3\sqrt{3}}{2} \approx 2.598$.

Problem	Std.	Late A	Late R	Late A/R
Maximum Independent Set	n	$\frac{n}{\Theta(1)}$	$\frac{n}{2}$	$\frac{3\sqrt{3}}{2}$
Maximum Matching	2	2	2	$\frac{3}{2}$
Minimum Vertex Cover	n	2	n	2
Minimum Spanning Forest	W	W	1	1

We consider only undirected graphs $G = (V, E)$. Throughout the paper, G will denote the graph under consideration, and V and E will denote its vertex and edge set, respectively. Moreover, $n = |V|$ will always denote the number of vertices in G. We use uv for the undirected edge connecting vertices u and v, so vu denotes the same edge.

Table 2: Comparison of the competitive ratios/functions of Maximum Independent Set and the two related problems under the standard model, as well as Late Accept, Late Reject, and Late Accept/Reject.

Problem	Std.	Late A	Late R	Late A/R
Maximum Independent Set	n	$\frac{n}{\Theta(1)}$	$\frac{n}{2}$	$\frac{3\sqrt{3}}{2}$
Maximum Acyclic Subgraph	$\frac{n}{2}$	$\frac{n}{\Theta(1)}$	$\frac{n}{3}$	$[\frac{3\sqrt{3}}{2}; 3]$
Maximum Planar Subgraph	$\frac{n}{4}$	$\frac{n}{\Theta(1)}$	$\frac{5n}{28}$	$[\frac{3\sqrt{3}}{2}; 3]$

2 Maximum Independent Set and Friends

In this section, we study the graph problems Maximum Independent Set, Maximum Acyclic Subgraph, and Maximum Planar Subgraph defined below.

An *independent set* for a graph $G = (V, E)$ is a subset $I \subseteq V$ such that no two vertices in I are connected by an edge. For the problem called Maximum Independent Set, the objective is to find an independent set of maximum cardinality. Similarly, Maximum Acyclic Subgraph (Maximum Planar Subgraph) is the problem of finding a maximum cardinality subset V' of the vertices in the input graph, such that V' induces an acyclic (planar) subgraph.

We consider the three problems in the vertex arrival model.

2.1 Standard model

It turns out that some quite general observations can be used to obtain results for three specific problems. In the below, we consider graph problems where the objective is to select a largest possible, feasible subset of the vertices.

Theorem 1 Let P be a graph problem and assume there exists an integer c_{P} such that P has the following properties.

- Any independent set is a valid solution to P.
- Any set of at most c_{P} vertices is a valid solution to P.
- No set of size $c_{\mathrm{P}} + 1$ inducing a clique is a valid solution to P.

For P in the standard model, the competitive function is n/c_{P}.

Proof The greedy algorithm, which accepts whenever the inclusion of the next vertex is still a valid solution, will accept at least the first c_{P} vertices. Since Opt can accept at most n vertices, the greedy algorithm is n/c_{P}-competitive, giving the upper bound.

For the lower bound, we use the following adversarial strategy. For each new vertex v, we define its neighbor set, $N(v)$, to be equal to S, where S is the set of vertices accepted by the online algorithm. Note that S is always a clique, and $V \setminus S$ is an independent set. Thus, $|S| \leq c_{\mathrm{P}}$ and $\textsc{Opt} \geq n - |S|$, so no algorithm can be better than n/c_{P}-competitive. □

For completeness, we show below how this theorem implies results in the standard model. The results themselves are trivial and have certainly been observed directly many times before.

Corollary 1 For the following three problems in the standard model, the competitive function is as follows.

- Maximum Independent Set: n.
- Maximum Acyclic Subgraph: $n/2$.
- Maximum Planar Subgraph: $n/4$.

Proof These results follow from Theorem 1 by observing that in any simple graph, any single vertex is an independent set ($c_{\mathrm{P}} = 1$), the induced graph of any two vertices is acyclic ($c_{\mathrm{P}} = 2$), and the induced graph of any four vertices is planar ($c_{\mathrm{P}} = 4$). □

2.2 Late Accept model

Theorem 2 Let P be a graph problem and assume there exists an integer c_{P} such that P has the following properties.

- Any independent set is a valid solution to P.
- No set of size $c_{\mathrm{P}} + 1$ inducing a clique is a valid solution to P.

For P in the Late Accept model, the competitive function is $n/\Theta(1)$.

Proof For any integer c, we define an n/c-competitive algorithm: The algorithm does not accept any vertex until the partial input presented so far contains a solution of size at least c and then accepts any such set of

vertices. Note that if ALG does not accept any vertices, then OPT $\leq c-1$. Hence, OPT $\leq n/c \cdot$ ALG $+c-1$ for any input.

Conversely, let ALG be any algorithm for P in the Late Accept model. The adversarial strategy is the same as in the proof of Theorem 1: For each new vertex v, the neighborhood of v is exactly the set of vertices accepted by ALG just before the arrival of v.

After each request, ALG has the possibility of (late-)accepting one or more vertices. A set of vertices accepted at the same time is called an accept set. Since all vertices in any given accept set are connected to all vertices in accept sets accepted earlier, picking one vertex from each accept set forms a clique. Thus, there can be at most c_{P} accept sets; otherwise, ALG has not produced a valid solution. Let m be the size of the largest of these at most c_{P} accept sets. Then, ALG accepts at most $a = m \cdot c_{\mathrm{P}}$ vertices, whereas OPT accepts at least $n-a$. Consequently, OPT $\geq (n/a-1) \cdot$ ALG. □

Corollary 2 For all of the problems Maximum Independent Set, Maximum Acyclic Subgraph, and Maximum Planar Subgraph in the Late Accept model, the competitive function is $n/\Theta(1)$.

2.3 Late Reject model

In this section, we prove that, in the Late Reject model, the competitive functions of Maximum Independent Set, Maximum Acyclic Subgraph, and Maximum Planar Subgraph are $n/2$, $n/3$, and $5n/28$, respectively.

2.3.1 Maximum Independent Set

Theorem 3 For Maximum Independent Set in the Late Reject model, the competitive function is $n/2$.

Proof For the lower bound, whenever there is at least one vertex v in the current independent set constructed by ALG, the adversary presents a vertex incident only to v. The only vertex which can be accepted when ALG rejects v is the vertex which just arrived, so ALG will never have more than one accepted vertex. On the other hand, the graph the adversary produces is bipartite, so OPT can accept at least half of its vertices.

For the upper bound, consider the following algorithm, ALG: If the presented vertex v can be added to the independent set I being constructed, then accept it. Otherwise, if v is adjacent to only one vertex u in I, then remove u from I and add v to I.

By definition, ALG accepts the first vertex. If ALG ever has two accepted vertices, it will also have at least two accepted vertices at the end, and the result holds. Otherwise, consider some vertex u accepted at some point by ALG. If the adversary presented a vertex not adjacent to u, ALG would accept it without rejecting u, which would be a contradiction. Thus, each vertex presented by the adversary is connected to the unique vertex currently in I. By definition of the algorithm, in every step, the currently accepted vertex is rejected and the new one accepted. Thus, considering the vertices in the order they are presented, they form a path. No algorithm can accept more than every second vertex from a path to form an independent set, and since all vertices are on the path, OPT accepts at most $\lceil n/2 \rceil$ vertices, and the result follows. □

2.3.2 Maximum Acyclic Subgraph

Theorem 4 For Maximum Acyclic Subgraph in the Late Reject model, the competitive function is $n/3$.

Proof For the lower bound, we use the same adversarial strategy as in the proofs of Theorems 1 and 2.

Consider any algorithm, ALG. By the adversarial strategy, at any point when ALG changes from having one to having two accepted vertices, those vertices are connected. Also, when the algorithm has two accepted vertices, accepting a new one requires rejecting one of the currently accepted vertices, and if it has two accepted vertices after such a procedure, they are connected. Thus, ALG cannot accept more than two vertices.

Figure 1 depicts an example of the resulting graph when six vertices are given, and the algorithm under consideration always accepts the current vertex and, if forced to by the acyclicity requirement, late-rejects the oldest vertex it currently has in its set. The last two, the red vertices, are the ones it accepts at the end. Note that OPT can accept vertices 1, 2, 4, and 5, for example.

We now give an upper bound on the number of vertices that OPT can accept from such an adversarial graph. To see that each graph G constructed by the adversary is chordal, consider a cycle C of length at least 4 and consider the point in time where the last vertex v of C arrived. Since v establishes the cycle, it was adjacent to two other vertices in the cycle, u and w. This means that u and w were already accepted at the point where v was given and, as argued above, also connected, so uw is a chord of C.

The graphs constructed by the adversary do not contain cliques of size

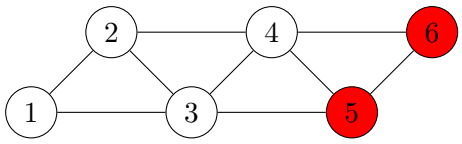

Figure 1: Possible adversarial graph with 6 vertices against example algorithm, having the red vertices as the accepted ones at the end.

more than three, since each arriving vertex is adjacent to at most two previous vertices, and the last vertex of a clique has to be adjacent to all other vertices of the clique when it arrives. It is well known that all chordal graphs are perfect, so all graphs constructed by the adversary are colorable by three colors, the size of their maximum cliques [21].

Consider any 3-coloring of one of these graphs, G. The subgraph induced by the vertices colored by any two of these three colors is acyclic, since any induced cycle in G of length at least four has a chord, and any 3-cycle must have three different colors. One of the colors must be used on at most $\lfloor \frac{n}{3} \rfloor$ vertices, so there must exist a subset of at least $\lceil \frac{2n}{3} \rceil$ of the vertices having only two of the colors. Thus, Opt may accept at least $\lceil \frac{2n}{3} \rceil$ vertices, and the competitive function is then at least $\lceil \frac{2n}{3} \rceil /2$.

For the upper bound, we define an algorithm, Alg, as follows: Alg always accepts the newest vertex, and rejects the oldest of its accepted vertices if this is necessary to break a cycle. Note that if Alg ever accepts three vertices, we are below the claimed bound on the competitive function and therefore done.

Alg accepts the first vertex. If the next vertex is not adjacent to the first, Alg will accept it and will be able to accept any third vertex given, and we would be done. Thus, we may assume that the first two vertices given are connected. Alg accepts both and will never reduce the number of accepted vertices it has.

If any future vertex given by the adversary is not connected to both of the accepted vertices at the time, Alg gets up to three accepted vertices. Thus, we may assume that in every step from the third step, the adversary gives vertices that are connected to the two vertices accepted by Alg at the given time, plus possibly to older, rejected vertices.

Thus, ignoring the edges possibly given to older, rejected vertices, the adversary is giving a graph as the one illustrated in Figure 1 for $n = 6$. We give an upper bound on how large an acyclic subgraph Opt can accept in such a graph (ignoring some edges only makes it easier for Opt).

To be acyclic, OPT must reject at least one vertex of any triangle, of which there are $n-2$. Since no vertex is incident to more than three triangles, it must reject at least $\lceil \frac{n-2}{3} \rceil$ vertices. Thus, it can accept no more than $n - \lceil \frac{n-2}{3} \rceil = \lfloor \frac{2n+2}{3} \rfloor = \lceil \frac{2n}{3} \rceil$.

We have shown that ALG is $\lceil \frac{2n}{3} \rceil$ /2-competitive and have matching upper and lower bounds on the competitive function. □

2.3.3 Maximum Planar Subgraph

For Maximum Planar Subgraph in the Late Reject model, we first give a general lower bound of $5n/28$ (Theorem 5) and then present an algorithm with a matching upper bound (Theorem 6). For both results, we use Kuratowski's Theorem [36], stating that a graph is planar if and only if it does not contain a (subdivision of) K_5 or $K_{3,3}$. We start by briefly outlining our approach.

For the lower bound, we use the following adversary strategy: Each new vertex has edges to exactly the vertices that are currently accepted by the online algorithm, ALG, considered. This ensures that, at any time during execution, the set, S, of vertices accepted by ALG induces a clique. Thus, at any time, $|S| \leq 4$.

Note that the *adversary graph* constructed in this way is an interval graph. If a vertex, v, is ever in S, the corresponding interval is the time interval in which it is in S. Thus, if v is late-rejected at the arrival of another vertex, u, the interval corresponding to v has its endpoints at the arrival times of the vertices v and u. If v never belongs to S, i.e., is not accepted by ALG, then the corresponding interval consists of just one point: the point in time where v arrives. For any point in time, we define the *depth* of the adversary graph to be the number of intervals containing this point. The depth is at most 5 at any point. Furthermore, for any time interval containing more than one point, the minimum depth of G within the interval is at most 4.

Among a set of vertices, the *last-living* vertex is the vertex that is in S at the latest point in time. If more vertices are late-rejected at the same time, ties are broken arbitrarily.

To give a lower bound on OPT for adversary graphs, we define an offline algorithm, OFF. The vertices of the adversary graph are partitioned into groups, each consisting of 7, 11, or 14 vertices, and OFF processes the vertices group by group. For each group of size 7, it rejects the two last-living vertices of the group. For each group of size 11, it rejects three vertices, two of which are the two last-living vertices of the group. For each group of size 14, it

rejects four vertices, two of which are the two last-living vertices of the group.

Note that OFF accepts at least $5n/7$ vertices, and ALG accepts at most 4 vertices, resulting in a ratio of at least $5n/28$. We prove, by induction on the number of groups, that the set of vertices accepted by OFF indeed induces a planar graph.

We define the algorithm OFF in detail in Algorithm 1, but first we prove some basic properties of the adversary graph (Lemmas 1–4 and Corollary 3).

Lemma 1 For any adversary graph, G, the following holds. Let G' be a subgraph of G containing exactly six vertices. Assume that G' contains a $K_{3,3}$ as a subgraph and denote the two partitions by X and Y. Then the vertices of X or Y induce a triangle.

Proof Let x_1, x_2, x_3 and y_1, y_2, y_3 be the vertices of X and Y, respectively, numbered according to their relative order of arrival. Assume without loss of generality that x_1 arrives before y_1. If Y induces a triangle, we are done. Thus, assume that the vertices of Y do not form a triangle.

By the definition of X and Y, x_1 is still in S at the arrival of y_3. Since Y does not form a clique, at least one vertex $y' \in \{y_1, y_2\}$ is rejected before the arrival of y_3, and hence, before x_1 is rejected. Since each vertex of X arrives no later than the rejection of y' and is rejected no earlier than the arrival of y_3, there must be a point in time (in fact, every point between the rejection of y' and the arrival of y_3), where x_1, x_2, and x_3 are all in S. Hence, X induces a clique. □

Corollary 3 For any adversary graph, G, the following holds. Let G' be a subgraph of G containing exactly six vertices. Assume that G' contains a $K_{3,3}$ as a subgraph. Then each vertex in G' is contained in a 4-clique in G'.

Proof By Lemma 1, one of the partitions, Z, of G' induces a clique. Since any vertex, v, of the other partition is adjacent to all vertices in Z, $Z \cup \{v\}$ induces a 4-clique. □

Lemma 2 Let G be an adversary graph. If G contains (a subdivision of) a $K_{3,3}$ as a subgraph, then it contains (a subdivision of) a $K_{3,3}$ for which the vertices of one partition constitute a triangle.

Proof Please refer to Fig. 2 for an illustration of the subgraph discussed in this proof.

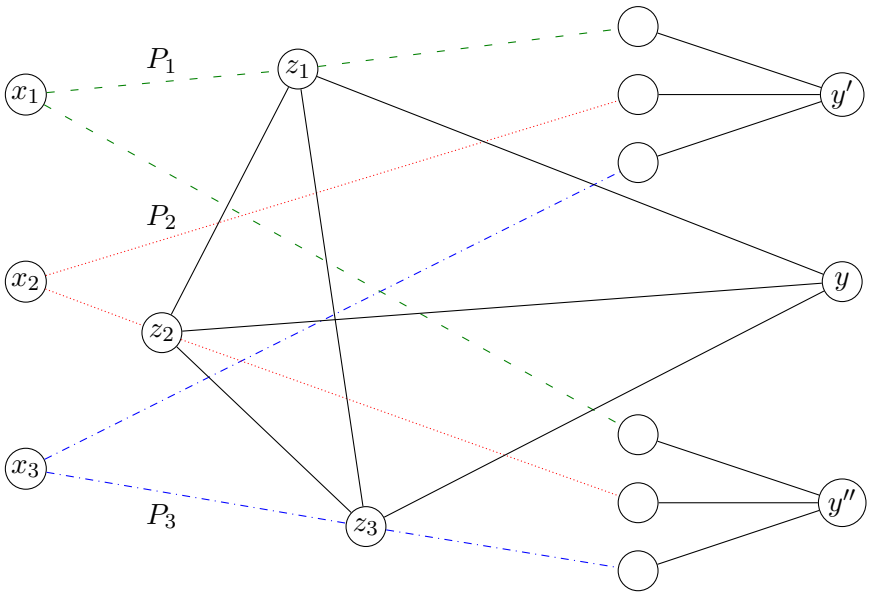

Figure 2: Illustration of the subgraph discussed in Lemma 2.

Let X and Y denote the two partitions of the (subdivision of a) $K_{3,3}$. If Y induces a triangle, we are done. Otherwise, a vertex $y' \in Y$ is rejected before the last vertex $y'' \in Y$ arrives. If there are two vertices in Y that are rejected before the arrival of y'', we let y' be the one which is rejected first (breaking ties arbitrarily).

For each vertex $x_i \in X$, we let P_i' be a path in (the subdivision of) the $K_{3,3}$, possibly consisting only of x_i, between x_i and a vertex adjacent to y'. Similarly, we let P_i'' be a path between x_i and a vertex adjacent to y''. We choose these six paths such that they are pairwise vertex-disjoint, except that they all contain x_i. This is possible since there are vertex-disjoint paths (except for the endpoints) between any pair of vertices $x \in X$ and $y \in Y$.

For each x_i, we let P_i be the union of P_i' and P_i''. Thus, P_i is a path connecting a neighbor of y' to a neighbor of y'' and containing x_i.

At any point in the time interval starting with the rejection of y' and ending with the arrival of y'', S contains a vertex from each of the three paths, P_1, P_2, P_3. Moreover, there must be a point, t, in this time interval at which the vertex $y \in Y \setminus \{y', y''\}$ is rejected or contained in S. Consider the three vertices, $z_1 \in P_1$, $z_2 \in P_2$, $z_3 \in P_3$, contained in S at time t. Note that, for each i, z_i could be equal to x_i. Since S always induces a clique, it

follows that each of these three vertices is adjacent to y and also that they are pairwise adjacent. Moreover, each of them has a path to both y' and y'' and all of these paths are disjoint. Thus, letting $Z = \{z_1, z_2, z_3\}$, (Y, Z) forms the bipartition of a, possibly different, (subdivision of a) $K_{3,3}$, and Z induces a triangle. □

The idea behind the first part of the following proof is that the contraction of an edge in an adversary graph G amounts to merging the corresponding (overlapping) intervals in the interval representation of G. Consequently, if G' is obtained from an adversary graph G by contracting an edge, the depth of any point in G' is at most as large as the corresponding depth in G.

Lemma 3 Any nonplanar adversary graph contains a K_5 or a $K_{3,3}$ as a subgraph.

Proof By Kuratowski's Theorem, a graph is nonplanar if and only if it contains a (subdivision of) K_5 or $K_{3,3}$. Thus, we just need to show that any adversary graph containing a subdivision of a K_5 or a $K_{3,3}$ also contains a K_5 or a $K_{3,3}$.

Consider an arbitrary adversary graph, G. First, assume that G has a subgraph, G', which is a subdivision of a K_5. Let $v_1, v_2, \ldots, v_5$ be the vertices corresponding to the vertices of K_5. G' contains disjoint paths between every pair v_i, v_j, $1 \leq i, j \leq 5$. Assume that we contract each of these paths into one edge, resulting in the graph G''. In any point, the depth of G' is at least as large as that of G'', and in some points it is larger. Since G'' is a 5-clique, there is at least one point where it has depth 5. In this point, the depth of G' is also 5. Hence, G' also contains a 5-clique.

Now, assume that G has a subgraph which is a subdivision of a $K_{3,3}$. According to Lemma 2, there exists (another) subgraph which is a subdivision of a $K_{3,3}$ for which the vertices of one partition constitute a triangle, Z. Each vertex, x, in the other partition, X, has three disjoint paths to the three vertices in Z, giving a total of nine disjoint paths. Let I be the interval where all three of the vertices in Z are in S, and let z'' be the last of these three vertices to arrive and z' be the first of the three to leave S. The third vertex in Z, z, must arrive no later that z'' and leave no earlier than z'. Each of the disjoint paths from $x \in X$ to the vertices in Z must have overlapping intervals for successive vertices on the paths. If x is in S during the interval I, then X has edges to all three vertices in Z. If x arrives before I, then the path from x to z'' must contain a vertex overlapping I. If x arrives after I,

then the path from z' to x must contain a vertex overlapping I. In both of these cases, let y_x be that vertex overlapping I. Each vertex $x \in X$ and the three paths connecting it to Z can be replaced by this y_x and the three edges connecting it to the vertices of Z. These three y_x vertices, together with Z, form a $K_{3,3}$. □

Call the consecutive groups of vertices produced by OFF from the original input sequence I, $U_1, U_2, \ldots$, to indicate the order in which they are processed. The vertices in U_i will arrive before those in U_{i+1}. We show later that the set of vertices OFF produces does not contain the two last-living vertices from each group. This allows us to use the following lemma. The proof uses the fact that $|S| \leq 4$. Because of the two vertices missing from S from each group U_i, there are at most two vertices in both S and the set being created just before the first vertex from U_{i+1} arrives. These are the only vertices from U_i in OFF's output graph that any vertex from U_{i+1} could be adjacent to.

Lemma 4 Consider an arbitrary adversary graph, G. Assume that the vertices of G are partitioned into groups, $U_1, U_2, \ldots$, such that for any pair of vertices, $u \in U_i$ and $v \in U_j$, accepted by ALG, if u arrives before v, then $i \leq j$. Consider the graph G' obtained by deleting the two last-living vertices from each group. If G' contains a K_5 or a $K_{3,3}$ as a subgraph, then this subgraph cannot contain vertices from more than one group.

Proof Let V' denote the vertex set of G'. Consider the set S at the time just before the arrival of the first vertex of some group, U_j. If $S \cap V' = \emptyset$ at this time, then no vertices of U_j will be connected to any vertices of V' from previous groups. Otherwise, we argue that $S \setminus V'$ contains two vertices, x and y, that both stay in S until all vertices currently in $S \cap V'$ are rejected. Consider any vertex $v \in S \cap V'$. Let U_i be the group that v belongs to. Then, by the assumption that V' is created by deleting the two last-living vertices of each group, $U_i \setminus V'$ contains two vertices that both stay in S at least until v is rejected.

Note that a K_5 can only be created in G' if, at the arrival of some vertex, S contains four vertices of V'. Since S contains at most four vertices at a time, it follows from the above that all vertices from previous groups must be rejected before S can contain four vertices of V' at the same time.

Now assume for the sake of contradiction that, after processing the vertices of U_j, G' contains a $K_{3,3}$ containing at least one vertex, $u \in U_j$, and at least one vertex, v, from a previous group. By Corollary 3, u and v are

each contained in a 4-clique contained in the $K_{3,3}$. When the last vertex of the 4-clique containing v arrives, the first three vertices of the clique are all in S. Since x and y stay in S at least until v is rejected, this must be before the start of U_j. Thus, none of these four vertices belong to U_j. Similarly, the vertices of the 4-clique containing u must all belong to U_j. Since both of these disjoint 4-cliques are contained in the $K_{3,3}$, there are at least eight vertices in the $K_{3,3}$, giving a contradiction. □

The input to OFF (see Algorithm 1) is a sequence I containing the vertices of an adversary graph in the order ALG receives them.

Lemma 5 When presented with an adversary graph, OFF produces a planar subgraph.

Proof Each set U, created in line 5, 11, 15, or 20, is called a *group*. If the vertices of I are sorted by order of arrival, the vertices of each group are consecutive vertices. Hence, by Lemmas 3 and 4, it is sufficient to prove the following invariant:

- Within each group, OFF rejects the two last-living vertices.
- No single group contains a K_5 or a $K_{3,3}$.

Consider an arbitrary group, U.

If the first five vertices, $u_1, \ldots, u_5$, of U do not induce a K_5, it follows trivially from the definition of OFF that it rejects the two last-living vertices of U (see lines 4–7 of Algorithm 1). It is not difficult to see that removing these two vertices ensures that the remaining five vertices in the group induce a planar graph: If $\{u_1, \ldots, u_6\}$ induces a nonplanar graph, the problem can be solved by rejecting u_6, since $\{u_1, \ldots, u_5\}$ induces a planar graph. Since the last-living vertex in $\{u_1, \ldots, u_6\}$ arrives no later than u_6, it is in S all the time that u_6 is there. Hence, rejecting the last-living vertex also solves the problem, even if it is not u_6. Similarly, if a problem occurs at the arrival of u_7, it can be solved by deleting the last-living remaining vertex. Finally, note that the subgraph induced by $U \setminus \{u, u'\}$ does not induce a $K_{3,3}$, since that would require six vertices.

For the remaining part of the proof, we assume that $u_1, \ldots, u_5$ induce a K_5 and let u denote the last-living vertex among $u_1, \ldots, u_5$. We let v_1, v_2, v_3, v_4 denote the vertices in $\{u_1, \ldots, u_5\} \setminus \{u\}$, sorted according to when they leave S (that is, last-living vertices appear last in the sorted sequence). We consider three cases. Note that lines 9– 13 of the algorithm are treated in both Case 1 and Case 2.

Algorithm 1: The algorithm OFF

Input: I

Result: A subsequence V_{OFF} of I, where the vertices of V_{OFF} induce a planar graph

$V_{\text{OFF}} \leftarrow \emptyset$
while $|I| \geq 14$ **do**
  Let $u_1, u_2, \ldots, u_{14}$ be the first 14 vertices of I, numbered according to their relative order of arrival
  if $\{u_1, \ldots, u_5\}$ does *not* induce a K_5 **then**
    $U \leftarrow \{u_1, \ldots, u_7\}$
    $u, u' \leftarrow$ the two last-living vertices in U
    $U_{\text{OFF}} \leftarrow U \setminus \{u, u'\}$
  else
    $u \leftarrow$ last-living vertex in $\{u_1, \ldots, u_5\}$
    if the subgraph induced by $\{u_1, \ldots, u_9\} \setminus \{u\}$ contains a $K_{3,3}$ **then**
      $U \leftarrow \{u_1, \ldots, u_7\}$
      $u' \leftarrow$ last-living vertex in $U \setminus \{u\}$
      $U_{\text{OFF}} \leftarrow U \setminus \{u, u'\}$
      if the last-living vertex in $U \setminus \{u, u'\}$ is rejected later than u **then**
        $U \leftarrow \{u_1, \ldots, u_{14}\}$
        $v \leftarrow$ last-living vertex in $\{u_8, \ldots, u_{13}\}$
        $v' \leftarrow$ last-living vertex in $\{u_8, \ldots, u_{14}\} \setminus \{v\}$
        $U_{\text{OFF}} \leftarrow U_{\text{OFF}} \cup \{u_8, \ldots, u_{14}\} \setminus \{v, v'\}$
    else
      $U \leftarrow \{u_1, \ldots, u_{11}\}$
      $u', u'' \leftarrow$ the two last-living vertices in $U \setminus \{u\}$
      $U_{\text{OFF}} \leftarrow U \setminus \{u, u', u''\}$
  From I, remove the prefix consisting of vertices in U
  $V_{\text{OFF}} \leftarrow V_{\text{OFF}} \cup U_{\text{OFF}}$

Case 1 [The subgraph induced by $\{u_1, \ldots, u_7\} \setminus \{u\}$ contains a $K_{3,3}$.] In this case, $U = \{u_1, \ldots, u_7\}$ (see lines 10–11). By Lemma 1, $U \setminus \{u\}$ contains a set, Z, of three vertices inducing a clique. Each vertex in Z is also adjacent to the three other vertices in $U \setminus \{u\}$. Before the arrival of u_6, v_1 must be rejected, since $|S| \leq 4$ at all times. Hence, $v_1 \notin Z$, since it cannot be adjacent to five other vertices from $U \setminus \{u\}$. Similarly, $u_6, u_7 \notin Z$, since vertices in Z must be in S at times overlapping with all other vertices of the $K_{3,3}$ and u_6 and u_7 do not overlap in time with v_1. In other words, $Z = \{v_2, v_3, v_4\}$. Thus, u_7 is adjacent to each of v_2, v_3, v_4. Since u stays in S at least until all of these three vertices are rejected, u_7 can be adjacent to all three only if u_6 is not in S when u_7 arrives. Thus, u_7 is the only vertex in U that possibly stays in S after u is rejected. This means that the two vertices deleted from U in line 13 are indeed the two last-living vertices of U.

The set $\{u_1, \ldots, u_5\} \setminus \{u\}$ has size four and therefore does not induce a K_5, and since u is the last-living vertex in $\{u_1, \ldots, u_5\}$, a K_5 cannot be created in $U \setminus \{u\}$ before all vertices in $\{u_1, \ldots, u_5\}$ have left S and four new vertices have entered. Clearly, the subgraph induced by $U \setminus \{u, u'\}$ does not contain a $K_{3,3}$, since that would require six vertices.

Case 2 [The subgraph induced by $\{u_1, \ldots, u_7\} \setminus \{u\}$ does not contain a $K_{3,3}$, but $\{u_1, \ldots, u_9\} \setminus \{u\}$ does contain a $K_{3,3}$.] In this case, we have $U = \{u_1, \ldots, u_7\}$ or $U = \{u_1, \ldots, u_{14}\}$ (see lines 10–11 and 14–15). Let U' be the set containing the six vertices of this $K_{3,3}$.

Since $\{v_1, v_2, v_3, v_4, u\}$ induces a K_5, $\{v_1, \ldots, v_4\}$ also forms a clique. Since the subgraph induced by $\{v_1, v_2, v_3, v_4, u_6, u_7\}$ does not contain a $K_{3,3}$, this means that u_6 and u_7 are not both adjacent to all of the vertices v_2, v_3, v_4. Since u_7 arrives after u_6, it means in particular that u_7 is not adjacent to v_2, which in turn implies that none of the vertices u_7, u_8, u_9 is adjacent to any of the vertices v_1, v_2.

By Lemma 1, at least one partition, Z, of the $K_{3,3}$ induces a clique. We note that $v_1, v_2 \notin Z$, since neither v_1 nor v_2 has five neighbors in $U \setminus \{u\}$. This means that if v_1 and v_2 are both in U', they are in the same partition. As argued before, v_1 is not adjacent to u_6 because it must leave S before u_6 arrives to keep $|S| \leq 4$, so now v_1 cannot have three neighbors in $U \setminus \{u, v_2\}$, and therefore, $v_1 \notin U'$.

If $v_2 \in U'$, v_3, v_4, and u_6 must all be in the partition not containing v_2, since v_2 has no neighbors in $U' \setminus \{v_3, v_4, u_6\}$ (recall that $v_1 \notin U'$ and that v_2 leaves S before the arrival of u_7). The vertices v_3, v_4, and u_6 (and hence, u) all stay in S at least until the arrival of the last vertex of U'. Thus, u_7

cannot enter S. This means that u_6 is the only vertex in U that may stay in S after u is rejected. We conclude that the two vertices deleted from U are the two last-living vertices.

Having accomplished this, we may now assume that $v_2 \notin U'$, i.e.,

$$U' = \{v_3, v_4, u_6, u_7, u_8, u_9\}.$$

If v_3 and v_4 are in the same partition, they must both stay in S until the three vertices of the other partition have arrived. Similarly, the remaining vertex, x, of the partition containing v_3 and v_4 must stay in S until the last vertex of the other partition has arrived, or the three vertices of the other partition must stay in S until x has arrived. Since u stays in S at least until v_3 and v_4 are rejected, there is only room for one more vertex in S. This must be the third vertex of the partition containing v_3 and v_4, which must then be u_6. This means that none of the vertices u_7 and u_8 can be accepted by the algorithm. In particular, this means that u_6 is the only vertex in $\{u_1, \ldots, u_7\}$ that can possibly stay in S after u is rejected. Hence, deleting the last-living vertex in $\{u_1, \ldots, u_7\} \setminus \{u\}$ together with u ensures that the two last-living vertices are deleted.

If v_3 and v_4 are in opposite partitions, v_3 must stay in S until all vertices of v_4's partition have arrived, and vice versa. Since u stays in S at least until v_3 and v_4 are rejected, the two vertices of v_4's partition must arrive before any of the two vertices of v_3's partition, and the algorithm must reject v_3 at the arrival of the last vertex of v_4's partition. Hence, the two vertices in the same partition as v_4 must be u_6 and u_7. The three vertices v_4, u_6, and u_7 must all stay in S until both u_8 and u_9 have arrived. This means that u_8 cannot enter S.

Since, among the vertices in U, only u_6 and u_7 can possibly stay in S after u is rejected, deleting u and the last-living vertex, u', in $\{u_1, \ldots, u_7\} \setminus \{u\}$ results in deleting the last-living vertex and the second or third last-living vertex of $\{u_1, \ldots, u_7\}$. If the two last-living vertices are deleted, we are done. Otherwise, the group is extended with the vertex set $\{u_8, \ldots, u_{14}\}$ (see lines 14–15). In this case, we let x denote the vertex in $\{u_1, \ldots, u_7\} \setminus \{u'\}$ that stays in S after u is rejected and note that $\{x, u'\} = \{u_6, u_7\}$.

Since u and u' are deleted and u_8 does not enter S, no K_5 can occur before five vertices after u_8 have arrived, i.e., before the arrival of u_{13}. Hence, deleting the last-living vertex, v, in $\{u_8, \ldots, u_{13}\}$ ensures that no K_5 will be created during the arrival of vertices $u_8, \ldots, u_{14}$.

Assume that a $K_{3,3}$ is created during the arrival of vertices $u_8, \ldots, u_{14}$. This $K_{3,3}$ cannot contain any of the vertices v_1, v_2, v_3, since v_3 is rejected

at the arrival of u_7. The vertex u_8 cannot be contained in this $K_{3,3}$ either, since u_8 does not enter S and therefore is adjacent only to u_4, u_6, and u_7.

Since u and u' stay in S at least until u_4 is rejected, u_4 cannot be part of a 4-clique in the subgraph induced by $\{u_4, \ldots, u_{14}\} \setminus \{u, u'\}$. Hence, by Corollary 3, u_4 cannot be part of the $K_{3,3}$. We conclude that the vertex set of the $K_{3,3}$ must be contained in $\{x, u_9, \ldots, u_{14}\}$. For the remaining part of the case, see lines 16–17 for the choice of v and v' discussed below. Recall that u_8 does not enter into S, so $v \neq u_8$, and, thus, $|\{x, u_9, \ldots, u_{14}\} \setminus \{v\}| = 6$. This means that choosing any vertex $v' \in \{u_9, \ldots, u_{14}\} \setminus \{v\}$ will ensure that $\{u_9, \ldots, u_{14}\} \setminus \{v, v'\}$ does not contain a $K_{3,3}$.

Case 3 [$\{u_1, \ldots, u_9\} \setminus \{u\}$ does not contain a $K_{3,3}$.] In this case, removing the two last-living vertices from $\{u_1, \ldots, u_{11}\} \setminus \{u\}$ ensures that the resulting set does not contain a K_5 or a $K_{3,3}$. In this case, it follows directly from the definition of Off that the two last-living vertices in U are rejected.

□

Theorem 5 For Maximum Planar Subgraph in the Late Reject model, the competitive function is at least $5n/28$.

Proof We have described an adversary strategy ensuring that any online algorithm accepts at most 4 vertices. We have also described an offline algorithm, Off, accepting at least 5/7 of the vertices in such an adversarial input. By Lemma 5, Off is a valid offline algorithm for the problem. The result follows. □

We now define a $5n/28$-competitive algorithm for Maximum Planar Subgraph in the Late Reject model, proving that the above lower bound is tight. The pseudocode is given as Algorithm 2.

The algorithm maintains a set, S, of accepted vertices. It always accepts the first four vertices, and the number of accepted vertices never decreases.

If the number of accepted vertices gets up to six, the algorithm keeps the six accepted vertices and rejects all future vertices. In this case, a ratio of at most $n/6 < 5n/28$ is obtained.

The input vertices are divided into groups, each consisting of seven or eight consecutive vertices. When the last vertex of a group has been handled, the group is closed and a new group is opened. At any given time, the current open group is called U, and in the analysis, the vertices of U are called $u_1, u_2, u_3, \ldots$, numbered according to their relative arrival times.

We prove that, as long $|S| = 4$, Opt will reject at least two vertices in each group, and as long as $|S| = 5$, Opt will reject at least one vertex in each group.

As long as $|S| < 6$, the algorithm accepts each new vertex v, if $S \cup \{v\}$ induces a planar subgraph. Otherwise, if $S \setminus U \neq \emptyset$, it tries to swap a vertex in $S \setminus U$ for v. Otherwise, depending on the sizes of S and U, it may swap a vertex in S for v. Note that as long as S contains vertices outside U, no vertex in U is late-rejected.

This ends the informal description of the algorithm. The details are in the pseudocode of Algorithm 2.

Algorithm 2: Algorithm for Maximum Planar Subgraph in the Late Reject model.

Result: A valid solution vertex set S

```
1  S ← ∅
2  U ← ∅
3  while a vertex v is presented do
4      if |S| < 6 then
5          U ← U ∪ {v}
6          if S ∪ {v} induces a planar graph then
7              S ← S ∪ {v}
8          else if S \ U ≠ ∅ then
9              Among the vertices in S \ U, let x be a vertex with a
                maximum number of neighbors in S ∪ {v}
10             if (S \ {x}) ∪ {v} induces a planar graph then
11                 S ← (S \ {x}) ∪ {v}
12         else if |S| = 4 and |U| ∈ {5, 7} then
13             Among the vertices in S, let u be a vertex with a
                maximum number of neighbors in S ∪ {v}
14             S ← (S \ {u}) ∪ {v}
15         if (|U| = 7 and |S| = 4) or |U| = 8 then
16             U ← ∅
```

Throughout the analysis, we will use the fact that the number of accepted vertices is monotonically nondecreasing.

Lemma 6 Consider the time of arrival of a vertex v during the execution of Algorithm 2. Assume that $|S| \leq 5$ and $S \setminus U \neq \emptyset$. Let $x \in S \setminus U$ be a vertex such that, in the subgraph induced by $S \cup \{v\}$, x has maximum degree among the vertices in $S \setminus U$. Then $(S \setminus \{x\}) \cup \{v\}$ induces a planar subgraph, if at least one of the following conditions hold:

(a) $|S| \leq 4$.

(b) There is a vertex $y \in S \setminus U$ such that $(S \setminus \{y\}) \cup \{v\}$ induces a planar subgraph.

(c) $|S \setminus U| \geq 3$.

Proof Condition (a) is trivially sufficient, since any nonplanar graph has at least five vertices.

Assume for the sake of contradiction that Condition (b) holds, but $(S \setminus \{x\}) \cup \{v\}$ does not induce a planar subgraph. Then $(S \setminus \{x\}) \cup \{v\}$ induces a K_5, but there is a vertex $y \in (S \setminus U) \setminus \{x\}$ such that $(S \setminus \{y\}) \cup \{v\}$ does not induce a K_5. This contradicts the fact that, in the subgraph induced by $S \cup \{v\}$, the degree of x is at least as high as that of y.

Finally, consider Condition (c). Since Condition (a) covers the case $|S| \leq 4$, we can assume that $|S| = 5$. Note that at least two vertices $a, b \in S$ are not neighbors, since otherwise S would contain a K_5. Since $|S \setminus U| \geq 3$, there is a vertex $z \in (S \setminus U) \setminus \{a, b\}$, and since a and b are not neighbors, $(S \setminus z) \cup \{v\}$ induces a planar graph. Hence, Condition (b) is fulfilled, and it follows that $(S \setminus \{x\}) \cup \{v\}$ induces a planar subgraph. □

Note that by Lemma 6(a), the check in line 10 of Algorithm 2 is successful as long as $|S| \leq 4$.

We now prove that the first three or four vertices of a group are always accepted:

Corollary 4 For the first four vertices u_1, u_2, u_3, u_4 of any group U during the execution of Algorithm 2, the following conditions hold.

(a) If $|S| \leq 5$ at the arrival of u_3, the vertices u_1, u_2, and u_3 are all accepted.

(b) If $|S| \leq 4$ at the arrival of u_4, then u_4 is also accepted.

(c) If $|S| = 5$ after u_3 is accepted, none of the vertices u_1, u_2, u_3 is late-rejected until after U is closed.

Proof For Condition (a), we consider three cases:

If $|S| \leq 3$ at the arrival of u_3, then $|S| \leq 3$ at the arrival of u_1 and u_2 as well. Hence, since any nonplanar graph has at least five vertices, all three vertices are accepted.

If $|S| = 4$ at the arrival of u_3, then $|S| \leq 4$ and $|S \setminus U| \geq 2$ at the arrival of each of u_1, u_2, and u_3. Hence, by Lemma 6(a), all three vertices are accepted.

If $|S| = 5$ at the arrival of u_3, then $|S \setminus U| \geq 3$. Since $|S \setminus U|$ can only decrease as vertices from U arrive, $|S \setminus U| \geq 3$ at the arrival of u_1 and u_2 as well. Hence, by Lemma 6(c), all three vertices are accepted.

For Condition (b), we consider two cases:

If $|S| \leq 3$ at the arrival of u_4, u_4 is accepted, since any nonplanar graph has at least five vertices.

If $|S| = 4$ at the arrival of u_4, $|S \setminus U| \geq 1$, and it follows from Lemma 6(a) that u_4 is accepted.

For Condition (c), note that except for line 14, the algorithm never late-rejects a vertex of U. Thus, Condition (c) follows from the fact that line 14 can be reached only if $|S| = 4$. □

Note that no vertices of U are late-rejected before the arrival of u_5. Moreover, as long as $S \setminus U \neq \emptyset$, no vertices of U are late-rejected. Thus, u_1, u_2, and u_3 stay in S at least until the arrival of u_5.

Lemma 7 For any group U, if $|S| = 4$ just after u_7 has been handled, then Opt rejects at least two vertices from U.

Proof Since $|S| = 4$ after the arrival of u_7, we have $|S| \leq 4$ when u_4 arrives. Hence, it follows from Corollary 4 that the vertices $u_1, \ldots, u_4$ are accepted. Additionally, $u_1, \ldots, u_5$ induce a K_5, since otherwise $|S| \geq 5$ after the arrival of u_5. Thus, when u_5 arrives, since the conditions in line 12 hold, Algorithm 2 late-rejects a vertex to accept u_5. Let u_i denote the vertex which is late-rejected. Then, u_6 is adjacent to all vertices in $\{u_1, ..., u_5\} \setminus \{u_i\}$, since otherwise the algorithm would accept it, and $|S| \geq 5$. S is not changed when u_6 arrives, since $|U| = 6$, so the condition in line 12 does not hold. Similarly to u_6, u_7 is adjacent to all vertices in $\{u_1, ..., u_5\} \setminus \{u_i\}$.

If Opt keeps all vertices in $\{u_1, ..., u_5\} \setminus \{u_i\}$, it must reject u_6 and u_7. Thus, we assume that it rejects one of the four vertices in $\{u_1, ..., u_5\} \setminus \{u_i\}$. Call this vertex u_j. However, $\{u_1, \ldots, u_7\} \setminus \{u_j\}$ induces a $K_{3,3}$, with u_i, u_6 and u_7 in one partition, since u_1, u_6, and u_7 each forms a K_5 with $\{u_1, ..., u_5\} \setminus \{u_i\}$. Thus, Opt must also reject one of these six vertices. □

Lemma 8 For any group U, if $|S| = 5$ just after u_7 has been handled, then either $|S| = 6$ when U is closed or Opt rejects at least one vertex from U.

Proof Consider the time just after u_7 has been handled and assume that $|S| = 5$. By Corollary 4(c), $\{u_1, u_2, u_3\} \subseteq S$ when U is closed. Thus, we only need to consider the cases $|S \cap U| = 3$, $|S \cap U| = 4$, and $|S \cap U| = 5$.

If $|S \cap U| = 3$, $S \cap U = \{u_1, u_2, u_3\}$ and all of the vertices $u_4, \ldots, u_7$ have been rejected. This means that rejecting one of the two vertices in $S \setminus U$ was not enough to enable the algorithm to accept any of the vertices $u_4, \ldots, u_7$. Let $x \in S \setminus U$ and $u_i \in \{u_4, \ldots, u_7\}$. Then, by Lemma 6(b), $\{x, u_1, u_2, u_3, u_i\}$ is nonplanar. Thus, any of $u_4, \ldots, u_7$ would form a K_5 together with u_1, u_2, u_3 and any accepted vertex outside U. In particular, this means that each of $u_4, \ldots, u_7$ is connected to each of u_1, u_2, u_3. Thus, U induces a $K_{3,3}$, and Opt must reject at least one vertex from U.

Similarly, if $|S \cap U| = 4$, exactly one of the vertices $u_4, \ldots, u_7$ is accepted. Hence, at least three of the vertices $u_4, \ldots, u_7$ must be connected to each of u_1, u_2, u_3. Again, U induces a $K_{3,3}$.

Finally, if $|S \cap U| = 5$, then $S \subseteq U$. Hence, if u_8 is not accepted by the algorithm, then U is not planar, and Opt has to reject at least one vertex from U. □

Theorem 6 The competitive function of Algorithm 2 is at most $5n/28$.

Proof Clearly, we can assume that $|S|$ never reaches 6, since that would result in a ratio of at most $n/6 < 5n/28$. Thus, since the first four vertices are accepted, and since $|S|$ never decreases, we only need to consider the cases $|S| = 4$ and $|S| = 5$. The result now follows from Lemmas 7 and 8: If the algorithm accepts only four vertices, then Opt accepts at most 5/7 of the vertices, resulting in a ratio of at most $5n/28$. If the algorithm accepts five vertices, then Opt accepts at most 7/8 of the vertices, resulting in a ratio of at most $7n/40 < 5n/28$. □

Corollary 5 For Maximum Planar Subgraph in the Late Reject model, the competitive function is $5n/28$.

2.4 Late Accept/Reject model

In the remainder of the section, we consider the Late Accept/Reject model. We show that there exists a $3\sqrt{3}/2$-competitive algorithm for Maximum Independent Set and 3-competitive algorithms for both Maximum Acyclic Subgraph and Maximum Planar Subgraph. After this, we give a lower bound of $3\sqrt{3}/2$ for all three problems. Thus, the result for Maximum Independent Set is tight. The upper bounds come from a variant of the greedy algorithm, Algorithm 3, rejecting a set of vertices if it can be replaced by a set at least r times as large. For Maximum Independent Set, we use $r = \sqrt{3}$, and for Maximum Acyclic Subgraph and Maximum Planar Subgraph, we use $r = 2$. The algorithmic idea is natural and has been used before in [31, 33], for

example. Thus, the challenge lies in deciding the parameter and proving the resulting competitive ratio. Pseudocode for Algorithm 3 is given below.

For Algorithm 3, we introduce the following notation. Let S be the current set of vertices that have been accepted and not late-rejected. Let R be the set of vertices that have been late-rejected, and let P denote the set $V \setminus (R \cup S)$ of vertices that have not been accepted (and, hence, not late-rejected).

For any $r > 1$, a pair (T, Q) of vertex sets is called an *r-admissible* pair if all the following conditions are satisfied:

1) $(S \setminus Q) \cup T$ is a valid solution;

2) $T \subseteq P$;

3) $|T| \geq r\,|Q|$.

A set $T \subseteq S$ is called *r-admissible*, if there exists a $Q \subseteq S$ such that (T, Q) is an r-admissible pair.

Algorithm 3: Algorithm for maximization graph problems in the Late Accept/Reject model.

Result: A valid solution vertex set S

```
1 S ← ∅
2 while a vertex v is presented do
3     if S ∪ {v} is a valid solution then
4         S ← S ∪ {v}
5     else
6         while there exists an r-admissible pair do
7             Let (T, Q) be an admissible pair minimizing |Q|
8             S ← (S \ Q) ∪ T
```

For the analysis of Algorithm 3, we partition S into the set, A, of vertices accepted in line 4 and the set, B, of vertices accepted in line 8. We let O be the valid solution constructed by Opt. For any set, U, of vertices, we let $U^+ = U \cap O$ and $U^- = U \setminus O$. Thus, $O = P^+ \cup S^+ \cup R^+ = P^+ \cup A^+ \cup B^+ \cup R^+$.

Lemma 9 When Algorithm 3 terminates, $|B| \geq (r-1)|R|$.

Proof Clearly, the inequality is true before the first vertex is presented. Each time a set, X, of vertices is moved from $S = A \cup B$ to R, a set at least r times as large as X is added to B. Thus, each time the size of R increases by some number x, the size of B increases by at least $rx - x$. The result follows inductively. □

Lemma 10 When Algorithm 3 terminates, $|P^+| < r\,|S^-|$.

Proof When the algorithm terminates, there are no r-admissible sets. This means, in particular, that P^+ is not r-admissible. Trivially, P^+ does not violate 2). Furthermore, since $P^+ \cup S^+ \subseteq O$, (P^+, S^-) fulfills 1). Thus, we conclude from 3) that $|P^+| < r|S^-|$. □

Theorem 7 For any graph problem with the objective of maximizing the number of vertices in the solution, using $r = 2$, Algorithm 3 is strictly 3-competitive.

Proof Since the algorithm accepts $|S|$ vertices and OPT accepts $|O|$ vertices, the result follows from the following calculations.

$$\begin{aligned} |O| &= |P^+| + |S^+| + |R^+| \\ &< 2|S^-| + |S^+| + |R|, \text{ by Lemma 10} \\ &\leq 2|S^-| + |S^+| + |B|, \text{ by Lemma 9} \\ &\leq 3|S| \end{aligned}$$

□

Corollary 6 For Maximum Acyclic Subgraph and Maximum Planar Subgraph in the Late Accept/Reject model, there exists a strictly 3-competitive algorithm.

Using Lemmas 11 and 12 below, we prove that for Maximum Independent Set, using $r = \sqrt{3}$, Algorithm 3 has competitive ratio at most $3\sqrt{3}/2 \approx 2.598$.

Lemma 11 For Maximum Independent Set, when Algorithm 3 terminates, $|B^-| + |R^-| \geq r\,|R^+|$.

Proof Consider a set, T, added to B in line 8. Note that $Q = N(T) \cap S$. We prove that

$$|T^-| \geq r|Q^+| \tag{1}$$

If $|Q^+| = 0$, this is trivially true. Thus, we can assume that Q^- is a proper subset of Q. Since T is r-admissible, it follows that

$$|T| \geq r|Q| \tag{2}$$

Note that $(S \setminus Q^-) \cup T^+$ is independent, since $(S \setminus Q) \cup T$ is independent and there are no edges between Q^+ and T^+. Since the algorithm chooses T such that $|Q|$ is minimized, this means that

$$|T^+| < r|Q^-| \tag{3}$$

Subtracting Ineq. (3) from Ineq. (2), we obtain Ineq. (1).

Let $T_1, T_2, \ldots, T_k$ be all the r-admissible sets that are chosen in line 8 during the run of the algorithm, and let $Q_1, Q_2, \ldots, Q_k$ be the corresponding sets that are removed from S. Then, $\cup_{i=1}^k T_i \subseteq B \cup R$, and thus, $\cup_{i=1}^k T_i^- \subseteq B^- \cup R^-$. Furthermore, $R = \cup_{i=1}^k Q_i$. Hence,

$$|B^-| + |R^-| \geq \sum_{i=1}^{k} |T_i^-| \geq \sum_{i=1}^{k} r|Q_i^+| = r|R^+|,$$

where the second inequality follows from Ineq. (1). □

Lemma 12 For Maximum Independent Set, when Algorithm 3 terminates, $|B^+| + |R^+| \leq r\left(\frac{|B^+|}{r+1} + \frac{|B|}{r^2-1}\right)$.

Proof Since $|B^+| = |B| - |B^-|$ and $|R^+| = |R| - |R^-|$, we obtain the following.

$$\begin{aligned}
(r+1)(|B^+| + |R^+|) &= r\big(|B^+| + |R^+|\big) + \big(|B| + |R| - (|B^-| + |R^-|)\big) \\
&\leq r|B^+| + r|R^+| + |B| + |R| - r|R^+|, \text{ by Lemma 11} \\
&= r|B^+| + |B| + |R| \\
&\leq r|B^+| + |B| + \frac{1}{r-1}|B|, \text{ by Lemma 9} \\
&= r|B^+| + \frac{r}{r-1}|B|
\end{aligned}$$

The result now follows by dividing both sides by $r+1$. □

Theorem 8 For Maximum Independent Set in the Late Accept/Reject model, using $r = \sqrt{3}$, Algorithm 3 is strictly $3\sqrt{3}/2$-competitive.

Proof We prove that $|O| \leq \frac{3\sqrt{3}}{2}|S|$, establishing the result.

$$\begin{aligned}
\textsc{Opt} &= |P^+| + |A^+| + |B^+| + |R^+| \\
&\leq \sqrt{3}(|A^-| + |B^-|) + |A^+| + |B^+| + |R^+|, \text{ by Lemma 10} \\
&\leq \sqrt{3}(|A| + |B^-|) + |B^+| + |R^+|, \text{ since } |A| = |A^+| + |A^-| \\
&\leq \sqrt{3}\left(|A| + |B^-| + \frac{1}{\sqrt{3}+1}|B^+| + \frac{1}{2}|B|\right), \text{ by Lemma 12} \\
&\leq \sqrt{3}\left(|A| + |B| + \frac{1}{2}|B|\right), \text{ since } \frac{1}{\sqrt{3}+1} < 1 \\
&\leq \frac{3\sqrt{3}}{2}(|A| + |B|), \text{ since } \frac{1}{2}|B| \leq \frac{1}{2}(|A| + |B|)
\end{aligned}$$

□

We prove a matching lower bound:

Theorem 9 For Maximum Independent Set in the Late Accept/Reject model, the competitive ratio is at least $\frac{3\sqrt{3}}{2}$.

Proof Assume that Alg is strictly c-competitive for some $c > 1$. We first show that c is at least $3\sqrt{3}/2$ and then lift the strictness restriction. Assume for the sake of contradiction that $c < 3\sqrt{3}/2$.

Incrementally, we construct an input graph consisting of a collection of bags, where each bag is an independent set. Whenever a new vertex, v, belonging to some bag B is given, we make it adjacent to every vertex not in B, except vertices that have been late-rejected by Alg. Thus, if Alg accepts v, it cannot hold any vertex in any other bag. This implies that the currently accepted vertices of Alg always form a subset of a single bag, which we refer to as Alg*'s bag*, and this is the crucial invariant in the proof. We say that Alg *switches* when it rejects the vertices of its current bag and accepts vertices of a different bag.

For the incremental construction, the first bag is special in the sense that Alg cannot switch to another bag. We discuss later when we decide to create the second bag, but all we will need is that the first bag is large enough. From the point where we have created a second bag, Alg has the option of switching. Whenever Alg switches from a bag, B, to a bag, B', we start the next bag, B'', and Alg must late-reject the vertices it has accepted in bag B since the vertices in B' are adjacent to those in B that have not already been late-rejected. The vertices we give from this point on and until Alg switches bag again belong to B'', and Alg never holds vertices in this

newest bag. However, since it is possible that ALG did not accept all of the vertices in B before it accepted some vertices in B', theoretically, at some later point other vertices in B could be accepted. This is also a switch, and a new bag is started. Note that new vertices are never added to an earlier bag.

Now we argue that as long as we keep giving vertices, ALG will repeatedly have to switch bags in order to be c-competitive. Choose some $\varepsilon > 0$, let B be ALG's bag, B' be the new bag, and let s be the number of vertices which are not adjacent to any vertices in B'. The s vertices are from earlier bags that were late-rejected by ALG before vertices from the successor bag were given, as explained above. Thus, they can be spread out over many earlier bags, but have no edges to vertices in later bags. If ALG has accepted a vertices of B, after $(c+\varepsilon)a - s$ vertices of the new bag B' have been given, ALG has to accept at least one additional vertex to be c-competitive, since at this point OPT could accept all of the vertices in B' and the s additional vertices. Since B' is the new bag, B has reached its final size, so eventually ALG will have to switch to a different bag.

For the proof, we keep track of relevant parts of the behavior of ALG using a tree structure. The first bag is the root of the tree. Recall that whenever ALG switches to a bag, say X, we start a new bag Y. In our tree structure we make Y a child of X.

Since ALG is c-competitive and always holds vertices only from a single bag B, the number a of vertices held in B satisfies $a \geq |B|/c$. Since, by assumption, $c < 3$, it follows that ALG can accept and then reject disjoint sets of vertices of B at most twice, or equivalently, that each bag in the tree has at most two children. As we proved above, ALG will have to keep switching bags, so if we keep giving vertices, this will eventually lead to leaves arbitrarily far from the root.

Consider a bag B_m that ALG holds after a sufficiently long sequence has been presented. Label the bags from the root to ALG's bag by $B_1, \ldots, B_m$, where B_{i+1} is a child of B_i for each $i = 1, \ldots, m-1$. Let a_j, $1 \leq j < m$, be the number of vertices of B_j held by ALG immediately before it rejected already accepted vertices from B_j for the first time and let a_m be the number of vertices currently accepted in B_m. Let $n_j = |B_j|$, $1 \leq j \leq m$.

Furthermore, for each $0 \leq j \leq m$, if j is even, let $s_j = a_2 + a_4 + \cdots + a_j$, and if j is odd, let $s_j = a_1 + a_3 + \cdots + a_j$. Note that our choice of adjacencies between bags implies that OPT can hold at least s_j vertices in bags $B_1, B_2, \ldots, B_j$.

Thus, just before ALG rejects the vertices in B_{j-1} (just before the n_jth

vertex of B_j is given), we must have

$$ca_{j-1} \geq n_j - 1 + s_{j-2}, \tag{4}$$

by the assumption that ALG is c-competitive. We want to introduce the arbitrarily small ε chosen above and eliminate the "-1" in this inequality: Since OPT can always hold the a_1 vertices from the root bag,

$$ca_j \geq a_1 \tag{5}$$

must hold for all j. Since $a_1 \geq (n_1 - 1)/c$, we get that

$$a_j \geq \frac{n_1 - 1}{c^2}. \tag{6}$$

We want to establish that $\varepsilon a_{j-1} \geq 1$. By Ineq. (6), we can make a_{j-1} as large as desired by increasing n_1. Thus, we give sufficiently many independent vertices in the beginning of the input sequence for the first bag, making n_1 large enough such that a_j becomes large enough. Now, using $\varepsilon a_{j-1} \geq 1$ and Ineq. (4), we establish that $(c + \varepsilon)a_{j-1} \geq n_j + s_{j-2}$. Trivially, $n_j \geq a_j$, so

$$(c + \varepsilon)a_{j-1} - s_{j-2} \geq a_j. \tag{7}$$

Next, we want to show that for any $1 \leq c < 3\sqrt{3}/2$, there exists an m such that

$$s_m > ca_m, \tag{8}$$

contradicting the assumption that ALG was c-competitive. To accomplish this, we repeatedly strengthen Ineq. (8) by replacing a_j with the bound from Ineq. (7), eventually arriving at an inequality which can be proven to hold, and then this will imply all the strengthened inequalities and, finally, Ineq. (8).

We first rewrite Ineq. (8), using that $s_m = a_m + s_{m-2}$ and collecting the a_m terms to get

$$s_{m-2} > (c - 1)a_m \tag{9}$$

We then replace a_m by the left-hand side of Ineq. (7) with $j = m$ to obtain the following stronger inequality:

$$s_{m-2} > (c - 1)(c + \varepsilon)a_{m-1} - (c - 1)s_{m-2}. \tag{10}$$

Note that, combined with Ineq. (7), Ineq. (10) implies Ineq. (9), which is equivalent to Ineq. (8). Collecting s_{m-2} terms in Ineq. (10) gives

$$cs_{m-2} > (c - 1)(c + \varepsilon)a_{m-1}. \tag{11}$$

Using the left-hand side of Ineq. (7) again, we get a stronger inequality:

$$cs_{m-2} > (c-1)(c+\varepsilon)^2 a_{m-2} - (c-1)(c+\varepsilon)s_{m-3}, \tag{12}$$

and, after moving s_{m-3},

$$cs_{m-2} + (c-1)(c+\varepsilon)s_{m-3} > (c-1)(c+\varepsilon)^2 a_{m-2}. \tag{13}$$

We proceed by labeling the coefficients of s_j and a_j in these inequalities of the form

$$f_i s_{m-i} + f_{i+1} s_{m-(i+1)} \geq g_i a_{m-i} \tag{14}$$

by sequences $\{f_k\}_{k=0}^{m}$, respectively $\{g_k\}_{k=0}^{m-1}$. We reverse the order of f and g so the index k corresponds to k applications of Ineq. (7), and simplifications as in the above. Therefore, f_k and g_k are the coefficients of s_{m-k} and a_{m-k}, respectively (recall that m is fixed).

We alternate between replacing s_{m-i} by $a_{m-i} + s_{m-i-2}$ and replacing a_{m-i} by the left-hand side of Ineq. (7), until we reach

$$f_{m-1} s_1 + f_m s_0 \geq g_{m-1} a_1 \tag{15}$$

The above calculations show that

$$\begin{aligned} f_0 &= 1 \\ f_1 &= 0 \\ f_2 &= c \\ f_3 &= (c-1)(c+\varepsilon) \\ g_0 &= c \\ g_1 &= (c-1)(c+\varepsilon) \\ g_2 &= (c-1)(c+\varepsilon)^2 \end{aligned}$$

Our aim is to show that the coefficients f_k and g_k satisfy

$$g_{i+1} = (c+\varepsilon)(g_i - f_i) \tag{16}$$

$$f_{i+2} = g_i \tag{17}$$

With the derived constants for f and g with small indices given above as the base case, we proceed by induction, assuming that for i, the coefficients of Ineq. (14) have the values claimed in Eq. (16) up to index i and in Eq. (17) up to index $i+1$.

We emphasize that we are still strengthening inequalities, so it is the inequality for the $(i+1)$-version of Ineq. (14) that we derive that will imply

Ineq. (14) for i. The induction is used to show that the coefficients in the inequalities fulfill the recurrence equations given in Eqs. (16) and (17).

From Ineq. (14), we replace S_{m-i} by $a_{m-i} + s_{m-2}$ and collect the a_{m-i} terms to obtain

$$f_i s_{m-(i+2)} + f_{i+1} s_{m-(i+1)} \geq (g_i - f_i) a_{m-i},$$

and replace a_{m-i} by the left-hand side of Ineq. (7) with $j = m - i$, to obtain the stronger inequality

$$f_i s_{m-(i+2)} + f_{i+1} s_{m-(i+1)} \geq (g_i - f_i)(c+\varepsilon) a_{m-(i+1)} - (g_i - f_i) s_{m-(i+2)}, \quad (18)$$

which can be rewritten as

$$g_i s_{m-(i+2)} + f_{i+1} s_{m-(i+1)} \geq (g_i - f_i)(c + \varepsilon) a_{m-(i+1)}. \quad (19)$$

Reordering on the left-hand side and inserting the claimed Eqs. (16) and (17), we arrive at

$$f_{i+1} s_{m-(i+1)} + f_{i+2} s_{m-(i+2)} \geq g_{i+1} a_{m-(i+1)},$$

which is the $(i + 1)$-version of Ineq. (14), proving Eqs. (16) and (17).

This concludes the strengthening of the inequalities and the sequence of implications. Thus, to prove Ineq. (8), it is sufficient to prove that there exist m and i such that Ineq. (14) holds. This, in turn, will follow if g_j is negative. Indeed, if g_j eventually becomes negative, we can choose j to be the smallest such index and Ineq. (14) then implies the desired Ineq. (8). This inequality contradicts the assumption of the algorithm being c-competitive, and we will have established the theorem.

Plugging Eq. (17) into Eq. (16) gives us

$$g_{i+1} = (c + \varepsilon)(g_i - g_{i-2}), \quad (20)$$

and this is the recurrence we use to show that g_i becomes negative.

According to [23, Theorem 1.2.1], every solution to a linear homogeneous difference equation oscillates (around zero, and thus has negative values infinitely often) if and only if its characteristic equation has no positive roots. For a direct proof of the subcase of the above theorem that we need, see [28].

The characteristic equation of Eq. (20) is $r^3 - (c + \varepsilon)r^2 + (c + \varepsilon) = 0$, which, for the interval $1 \leq c + \varepsilon < \frac{3\sqrt{3}}{2}$, has two imaginary roots and one real root. Letting $d = c + \varepsilon$ and

$$s = -\sqrt[3]{108d - 8d^3 - 12\sqrt{-12d^4 + 81d^2}},$$

this third root is $\frac{s}{6} + \frac{2d^2}{3s} + \frac{d}{3}$, which is negative for $1 \leq d \leq \frac{3\sqrt{3}}{2}$. (Note that s is no longer real when $-12d^4 + 81d^2$ becomes negative, which occurs for $d > \frac{3\sqrt{3}}{2}$. At $d = \frac{3\sqrt{3}}{2}$, in addition to the negative real root, there is a double root at $\sqrt{3}$, and the solution to the recurrence never becomes negative.)

Thus, from [23, Theorem 1.2.1], any solution to the recurrence equation oscillates, implying, in particular, that it becomes negative at some point, giving the desired contradiction.

Finally, we return to the assumption of strictness, which can easily be removed. There are only two places we use the relation that holds due to the assumption of ALG being strictly c-competitive. One place is in the claim $ca_j \geq a_1$. However, we use this only to lead to Ineq. (7), and just as we used a large enough first bag to make $\varepsilon a_{j-1} \geq 1$, we can increase the size of the first bag to eliminate any additive constant. The other place was in the argument that $ca_{j-1} \geq n_j - 1 + s_{j-2}$. Also in this case, if the bags are large enough, no additive constant makes a difference: Using the technique described immediately following Ineq. (6), the minimum size of all bags can be increased by increasing n_1, since that increases the lower bound on a_1, and the algorithm can never hold fewer vertices in any bag than that. □

The previous two theorems give us:

Corollary 7 For Maximum Independent Set in the Late Accept/Reject model, the competitive ratio is $\frac{3\sqrt{3}}{2}$.

Making small adjustments to the proof of Theorem 9 for Maximum Independent Set, we can prove the same lower bound for Maximum Acyclic Subgraph and Maximum Planar Subgraph.

Theorem 10 For Maximum Acyclic Subgraph and Maximum Planar Subgraph in the Late Accept/Reject model, the competitive ratio is at least $\frac{3\sqrt{3}}{2}$.

Proof We use the same proof as for Theorem 9, with the modifications described below.

For Maximum Acyclic Subgraph, the algorithm may keep one vertex from a previous bag. However, since each vertex in the current bag is adjacent to all vertices in previous bags that have not been late-rejected by the algorithm, keeping two vertices from previous bags would create a cycle. Similarly, for Maximum Planar Subgraph, keeping three vertices from previous bags would create a $K_{3,3}$. Thus, for both problems, we can assume that the algorithm keeps at most two vertices from previous bags.

Taking into consideration that the algorithm may keep up to two vertices from previous bags leads to the following adjustments:

As long as a vertex of an old bag is kept by the algorithm, it is connected to all vertices of the new bag. Thus, Opt may lose up to two vertices from each bag. Hence, to ensure that our lower bounds on Opt are still valid, we redefine s_j as

$$s_j = \begin{cases} a_2 + a_4 + \ldots + a_j - j, \text{ if } j \text{ is even} \\ a_1 + a_3 + \ldots + a_j - j - 1, \text{ if } j \text{ is odd} \end{cases}$$

Eqs. (4), (5), and (6) are changed to

$$c(a_{j-1} + 2) \geq n_j - 1 + s_{j-2}, \tag{21}$$

$$c(a_j + 2) \geq a_1 \geq \frac{n_1 - 1}{c},$$

and

$$a_j \geq \frac{n_1 - 1}{c^2} - 2.$$

We choose n_1 large enough that

$$\varepsilon a_{j-1} \geq \frac{2c+2}{c-1} + 2c + 1,$$

instead of $\varepsilon a_{j-1} \geq 1$ as in the proof of Theorem 9. Thus, using Ineq. (21), Ineq. (7) can be replaced by

$$(c + \varepsilon)a_{j-1} - s_{j-2} - \frac{2c+2}{c-1} \geq a_j.$$

We replace Ineq. (8) by

$$s_m > c(a_m + 2)$$

Now, using $s_m = s_{m-2} + a_m - 2$ instead of $s_m = s_{m-2} + a_m$, we obtain the following instead of Ineq. (9).

$$s_{m-2} > (c-1)a_m + 2c + 2$$

Substituting $(c + \varepsilon)a_{m-1} - s_{m-2} - \frac{2c+2}{c-1}$ for a_m, we obtain Ineq. (11). Substituting $(c + \varepsilon)a_{m-2} - s_{m-3} - \frac{2c+2}{c-1}$ for a_{m-1} in Ineq. (11), we obtain

$$cs_{m-2} > (c-1)(c+\varepsilon)^2 a_{m-2} - (c-1)(c+\varepsilon)s_{m-3} - (c+\varepsilon)(2c+2),$$

which is implied by Ineq. (12).

Each time we substitute s_{m-i} by $s_{m-i-2}+a_{m-i}-2$, we subtract $2f_i$ from the left-hand side of the inequality, and substituting a_{m-i} by $(c+\varepsilon)a_{m-i-1}-s_{m-i-2}-\frac{2c+2}{c-1}$, we subtract $\frac{2c+2}{c-1}(g_i-f_i)$ from the right-hand side. Note that

$$\begin{array}{rcl} 2f_i & < & \frac{2c+2}{c-1}(g_i-f_i) \\ \Updownarrow & & \\ 4cf_i & < & (2c+2)g_i \\ \Updownarrow & & \\ 4cg_{i-2} & < & (2c+2)g_i \\ \Updownarrow & & \\ 4c\frac{g_i}{(c+\varepsilon)^2} & < & (2c+2)g_i \\ \Updownarrow & & \\ 2c & < & (c+1)(c+\varepsilon)^2 \\ \Uparrow & & \\ c & \geq & 1 \end{array}$$

Thus, we subtract more on the right-hand side than on the left-hand side, and hence, we keep getting inequalities that are weaker than the corresponding ones in Ineq. (14). Hence, the strongest inequality (the one obtained by setting $i=m-1$), is implied by the strongest inequality in Ineq. (14). Since Ineq. (14) was proven to hold in the proof of Theorem 9, this completes the proof. □

3 Maximum Matching

A *matching* in a graph $G=(V,E)$ is a subset of E consisting of pairwise non-incident edges. For the problem called Maximum Matching, the objective is to find a matching of maximum cardinality. We study online Maximum Matching in the edge arrival model, but note that the results hold in the vertex arrival model as well: For the upper bounds, an algorithm in the vertex arrival model can process the edges incident to the current vertex in any order. For the lower bounds, all adversarial sequences used in this section consist of paths, and hence, exactly the same input can be given in the vertex arrival model.

It is well known and easy to prove that the greedy algorithm which adds an edge to the matching whenever possible is 2-competitive and this is optimal in the standard model. The first published proof of this is perhaps in the classical paper of Korte and Hausmann [26]. The paper shows that in any graph, the ratio of the minimum size of a maximal matching to the

size of a maximum matching is at least $\frac{1}{2}$, and there are graphs where it is no more than $\frac{1}{2}$. Since the greedy algorithm produces a maximal matching, the claim follows.

Late accept or late reject alone does not help:

Theorem 11 For Maximum Matching in the Late Accept model, the competitive ratio is 2.

Proof The upper bound follows from the standard model. For the lower bound, we can use the same sequence as in the standard model: The adversary presents m mutually non-incident edges to some algorithm, ALG. For every edge uv accepted by ALG at any point, the adversary presents edges xu and vy, which ALG cannot accept. Thus, there will be m connected components such that in each component, OPT accepts at least twice as many edges as ALG. □

Theorem 12 For Maximum Matching in the Late Reject model, the competitive ratio is 2.

Proof The upper bound follows from the standard model. For the lower bound, the adversary presents m mutually non-incident edges to some algorithm, ALG. For each edge, uv, accepted by ALG, the adversary presents an edge vx. If ALG late-rejects uv, then the adversary presents an edge xy. ALG can only accept one of the edges vx and xy and it cannot accept uv again, but OPT accepts both uv and xy. Otherwise, ALG must reject vx. In this case, the adversary presents an edge, zu. ALG can only keep zu or uv, but OPT accepts both zu and vx. This adversarial strategy results in m connected components such that in each component, OPT accepts at least twice as many edges as ALG. □

Theorem 13 For Maximum Matching in the Late Accept/Reject model, the competitive ratio is at least 3/2.

Proof The adversary presents a number of mutually non-incident edges to some algorithm, ALG. If, for some such edge uv, during the entire processing of the input, ALG does not accept uv, then OPT will, and no more edges incident to u or v will be presented. The ratio is then unbounded on the subconstruction containing uv.

If ALG accepts an edge uv, the adversary presents xu and vy. If ALG never rejects uv, no more edges incident to any of these vertices will be presented, and the ratio is 2 on this subconstruction.

If ALG late-rejects uv at some point, then the adversary presents $x'x$ and yy'. The algorithm cannot accept uv again, so it cannot accept more than two edges from this subconstruction, while OPT can accept three, giving a ratio of 3/2. □

To prove a matching upper bound, we give an algorithm, Algorithm 4, which is strictly $\frac{3}{2}$-competitive in the Late Accept/Reject model.

Recall that for a matching M, a path $P = e_1, \ldots, e_k$ is *alternating* with respect to M, if for all $i \in \{1, \ldots, k\}$, e_i belongs to M if and only if i is even. Moreover, an alternating path P is called *augmenting* if neither endpoint of P is incident to a matched edge. Note that the symmetric difference of a matching M and an augmenting path with respect to M is a matching of size larger than M. We focus on local changes, called short augmentations in [37]. We use a result which implies that if a maximal matching M does not admit augmenting paths of length 3, then $3|M| \geq 2|\,\text{OPT}\,|$. This fact is part of the folklore and its proof can be found for example in [11, Lemma 2].

Algorithm 4: Algorithm for Maximum Matching in the Late Accept/Reject model.

Result: Maximum Matching M

$M \leftarrow \emptyset$
while an edge e is presented **do**
  if $M \cup \{e\}$ is a matching **then**
    $M \leftarrow M \cup \{e\}$
  if there is an augmenting path $xuvy$ of length 3 **then**
    $M \leftarrow (M \cup \{ux, vy\}) \setminus \{uv\}$

Theorem 14 For Maximum Matching in the Late Accept/Reject model, Algorithm 4 is strictly 3/2-competitive.

Proof We first show that Algorithm 4 is a Late Accept/Reject algorithm, i.e., no edge which is late-rejected is later late-accepted. Suppose an edge $e = uv$ is late-rejected in some step i. For it to be accepted again, there must later be an augmenting path consisting of three edges, where e is one of the outer edges, and one endpoint of e must not be incident to any edges of the matching at that point. However, once a vertex is incident to an edge in some matching, it is always incident to some edge in all later matchings, due to the augmentation. Thus, e cannot be late-accepted later, so after any edge is late-rejected, it will never be accepted again.

Next, we prove that the algorithm is strictly 3/2-competitive. Let M be the matching constructed by Algorithm 4 on a graph G. Algorithm 4 ensures

that G does not contain any augmenting paths of length at most three with respect to M by augmenting on them when they do exist. To prove that this fact implies the bound, we present a proof similar to [11, Lemma 2]. Let M' be any maximum matching in G. Consider the symmetric difference N of M and M'. Since M and M' are matchings, any path in N contains alternatingly edges from M and M'. Since each augmenting path with respect to M in N contains one more edge of M' than of M, we get that N contains $|M'| - |M|$ augmenting paths. Clearly, no augmenting path in N consists of a single edge, since all edges which either are not accepted or are late-rejected by Algorithm 4 are incident to at least one edge accepted by the algorithm. Since there is no augmenting path of length at most three with respect to M in G, it follows that all of the $|M'| - |M|$ augmenting paths in N have length at least five. At least two edges of an augmenting path in N of length at least five are contained in M, and thus accepted by the algorithm, so we get that $|M| \geq 2(|M'| - |M|)$, giving that $|M'| \leq \frac{3}{2}|M|$. □

4 Minimum Vertex Cover

A *vertex cover* for a graph $G = (V, E)$ is a subset $C \subseteq V$ such that for any edge, $uv \in E$, $\{u, v\} \cap C \neq \emptyset$. For the problem called Minimum Vertex Cover, the objective is to find a vertex cover of minimum cardinality. We study online Minimum Vertex Cover in the vertex arrival model.

Theorem 15 For Minimum Vertex Cover in the standard model, the competitive function is n.

Proof For the lower bound, consider any online algorithm, Alg. For each n, the adversary presents independent vertices until Alg rejects some vertex or n vertices have been presented. If n vertices are presented, Opt accepts none of them and the ratio is unbounded. If the algorithm rejects some vertex v, then the remainder of the n vertices will be adjacent only to v. Opt will only accept v and the result follows.

For the upper bound, the algorithm only accepts a new vertex v if at least one edge incident with v is not already covered. Thus, it rejects the first vertex and therefore accepts at most $n - 1$ vertices. Opt accepts at least one vertex unless there are no edges, in which case the algorithm does not accept any vertices either. □

The situation improves dramatically if we can accept vertices at a later stage.

Theorem 16 For Minimum Vertex Cover in the Late Accept model, the competitive ratio is 2.

Proof The best known offline 2-approximation algorithm for Minimum Vertex Cover greedily maintains a maximal matching, repeatedly covering both endpoints of an edge and removing all edges incident to these two endpoints. In the Late Accept model, a 2-competitive online algorithm can be obtained by mimicking the offline approximation algorithm. The online algorithm does not accept any vertex until it sees the second vertex incident to an uncovered edge; then it accepts both endpoints of that edge.

For the lower bound, consider any algorithm ALG. The adversary presents isolated pairs of vertices, each pair connected by an edge. After the second vertex of a pair has arrived, ALG must have accepted at least one of them, or the adversary could stop the input there, and ALG's output would not be a vertex cover. If ALG accepts both vertices, then no further vertices adjacent to the pair arrive, and OPT could have covered the edge with only one vertex. If ALG accepts only one vertex u from a pair $\{u, v\}$, then an additional vertex adjacent only to v arrives, and OPT could cover both edges with only v, but ALG must accept at least two of the three vertices. □

Allowing both late accept and late reject does not improve the situation further.

Theorem 17 For Minimum Vertex Cover in the Late Accept/Reject model, the competitive ratio is 2.

Proof The upper bound follows from Theorem 16. The lower bound follows from the observation that no algorithm that ever late-rejects a vertex can be c-competitive for any constant c. Indeed, if ALG late-rejects a vertex v, then the adversary can present arbitrarily many vertices adjacent only to v. Therefore, to be c-competitive for any constant c, ALG can never late-reject a vertex and the lower bound from Theorem 16 applies. □

Theorem 18 For Minimum Vertex Cover in the Late Reject model, the competitive function is n.

Proof For the lower bound, the adversary keeps giving independent vertices until the algorithm rejects at least one vertex, v. Since OPT does not have to accept any vertices if they are all independent, the algorithm must eventually reject at least one vertex to avoid an unbounded competitive function. We let b denote the number of vertices presented at the time the first vertex is

rejected. After this point, all new vertices are adjacent to v, so the algorithm has to accept all of them. In total, at least $n - b$ vertices are accepted, and OPT accepts only v.

For the upper bound, consider the following algorithm, ALG_b: The first $b + 1$ vertices are accepted (if they arrive). After that an optimal vertex cover, C, for the edges seen so far is calculated. Each vertex not included in C is rejected. After this, each new vertex is accepted only if necessary. Note that the size of C is a lower bound on OPT. Thus, for any input sequence I of length at least $b + 1$, either $\text{OPT}(I) = \text{ALG}(I) = 0$ or

$$\frac{\text{ALG}_b(I)}{\text{OPT}(I)} = \frac{|C| + n - (b+1)}{\text{OPT}(I)} \leq \frac{\text{OPT}(I) + n - (b+1)}{\text{OPT}(I)} \leq n - b\,.$$

□

5 Minimum Spanning Forest

A *spanning forest* for a graph $G = (V, E)$ is a subset $T \subseteq E$ which forms a spanning tree on each of the connected components of G. Given a weight function $w\colon E \to \mathbb{R}^+$, the objective of the Minimum Spanning Forest problem is to find a spanning forest of minimum total weight. We let W denote the ratio between the largest and the smallest weight of any edge in the graph.

We study online Minimum Spanning Forest in the edge arrival model, but the results also hold in the vertex arrival model: For the upper bounds, an algorithm in the vertex arrival model can process the edges incident to the current vertex in any order. In the lower bound sequences presented here, all edges from a new vertex to all previous vertices are presented together in an arbitrary order.

As also noted in [24], the competitive ratio of any deterministic algorithm in the standard model can be arbitrarily high, depending only on the edge weights:

Theorem 19 For Minimum Spanning Forest in the standard model, the competitive ratio is W.

Proof Since all spanning forests have the same number of edges, the ratio cannot be worse than W. A matching lower bound can be realized by the adversary first presenting a tree consisting of edges of weight W, and then presenting edges of weight 1 from a new vertex v to each of the vertices seen so far. The ratio is $\frac{(n-2)W+1}{n-1}$, giving an asymptotic lower bound of W. □

Since the adversary can end the input at any time, an online algorithm must always have a forest spanning all the vertices seen so far, so in moving from the standard model to the Late Accept model, we do not gain any advantage:

Theorem 20 For Minimum Spanning Forest in the Late Accept model, the competitive ratio is W.

Proof We show that we can never perform a late accept. Assume to the contrary that an edge uv was late-accepted and added to a solution F' for the current graph $G' = (V', E')$. Since uv was late-accepted, both vertices u and v were seen earlier and thus contained in V'. By our requirement that the algorithms maintain a spanning forest on the set of vertices presented so far, F' is a spanning forest of G'. Therefore, adding uv created a cycle, contradicting that the algorithm finds a forest. □

On the other hand, in the Late Reject model, the greedy online algorithm mentioned by Tarjan in [35] can be used. We detail the algorithm in the proof.

Theorem 21 For Minimum Spanning Forest in the Late Reject model, the competitive ratio is 1.

Proof No algorithm can be better than 1-competitive. For the upper bound, we note that the greedy online algorithm mentioned by Tarjan in [35] works in the Late Reject model: Assume that the current forest is F' when an edge $e = uv$ arrives. If at least one of the two endpoints of e is a vertex not seen earlier, accept e. Otherwise, the greedy algorithm constructs the unique cycle C_e in $F' \cup \{e\}$. If e is not the heaviest edge in C_e, then the algorithm late-rejects the heaviest edge f in C_e and replaces it by e, obtaining F''. Otherwise, it rejects e. It is easy to see that this produces an optimal spanning forest; it only uses the so-called red rule [35]. □

Since the Late Reject model leads to an optimal spanning tree, any model allowing that possibility inherits the result.

Theorem 22 For Minimum Spanning Forest in the Late Accept/Reject model, the competitive ratio is 1.

The Minimum Spanning Forest problem is one of the original examples of a matroid [38]. The last two results hold for any matroid, since the red rule [35, Chapter 6] can be used for any matroid. A proof of this can be found in [27, Section 3.2], which credits Lawler [29] for the statement without a proof.

Future Work

One could reasonably consider late operations as a resource to be used sparingly, as for the rearrangements in [18, 30, 13, 14], for example. Thus, an interesting continuation of our work would be a study of trade-offs between the number of late operations employed and the quality of the solution (in terms of competitiveness). Obviously, one could also investigate other online problems and further model variations. Work in the direction of the latter has been initiated in [1], where the authors investigate further levels of late accept/reject operations.

References

[1] Spyros Angelopoulos, Christoph Dürr, and Shendan Jin. Online maximum matching with recourse. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, 2018.

[2] Yair Bartal, Amos Fiat, and Stefano Leonardi. Lower bounds for on-line graph problems with application to on-line circuit and optical routing. In *28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 531–540. ACM, 1996.

[3] Joan Boyar, Stephan J. Eidenbenz, Lene M. Favrholdt, Michal Kotrbčík, and Kim S. Larsen. Online Dominating Set. *Algorithmica*, 81(5):1938–1964, 2019.

[4] Joan Boyar, Lene M. Favrholdt, Michal Kotrbčík, and Kim S. Larsen. Relaxing the irrevocability requirement for online graph algorithms. In *15th International Algorithms and Data Structures Symposium (WADS)*, volume 10389 of *Lecture Notes in Computer Science*, pages 217–228. Springer, 2017.

[5] Joan Boyar, Kim S. Larsen, and Abyayananda Maiti. The frequent items problem in online streaming under various performance measures. *International Journal of Foundations of Computer Science*, 26(4):413–439, 2015.

[6] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. In *26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1202–1216. SIAM, 2015.

[7] Marek Cygan, Łukasz Jeż, and Jiří Sgall. Online knapsack revisited. *Theory of Computing Systems*, 58(1):153–190, 2016.

[8] Marc Demange and Vangelis Th. Paschos. On-line vertex-covering. *Theoretical Computer Science*, 332:83–108, 2005.

[9] Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM Journal on Discrete Mathematics*, 25(3):1251–1265, 2011.

[10] Leah Epstein, Asaf Levin, Danny Segev, and Oren Weimann. Improved bounds for online preemptive matching. In *30th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 389–399. Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, 2013.

[11] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2–3):207–216, 2005.

[12] Juan A. Garay, Inder S. Gopal, Shay Kutten, Yishay Mansour, and Moti Yung. Efficient on-line call control algorithms. *Journal of Algorithms*, 23(1):180–194, 1997.

[13] Albert Gu, Anupam Gupta, and Amit Kumar. The power of deferral: Maintaining a constant-competitive Steiner tree online. *SIAM Journal on Computing*, 45(1):1–28, 2016.

[14] Anupam Gupta and Amit Kumar. Online Steiner tree with deletions. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 455–467. SIAM, 2014.

[15] Xin Han, Yasushi Kawase, and Kazuhisa Makino. Randomized algorithms for online knapsack problems. *Theoretical Computer Science*, 562:395–405, 2015.

[16] Xin Han, Yasushi Kawase, Kazuhisa Makino, and He Guo. Online removable knapsack problem under convex function. *Theoretical Computer Science*, 540:62–69, 2014.

[17] Xin Han and Kazuhisa Makino. Online minimization knapsack problem. *Theoretical Computer Science*, 609:185–196, 2016.

[18] Makoto Imase and Bernard M. Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.

[19] Kazuo Iwama and Shiro Taketomi. Removable online knapsack problems. In *29th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2380 of *Lecture Notes in Computer Science*, pages 293–305. Springer, 2002.

[20] Patrick Jaillet and Xin Lu. Online traveling salesman problems with rejection options. *Networks*, 64:84–95, 2014.

[21] Tommy R. Jensen and Bjarne Toft. *Graph Coloring Problems*. John Wiley & Sons, 1995.

[22] Aanna R. Karlin, Mark S. Manasse, Larry Rudolph, and Daniel D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

[23] V. L. Kocic and G. Ladas. *Global Behavior of Nonlinear Difference Equations of Higher Order with Applications*, volume 256 of *Mathematics and Its Applications*. Springer, 1993.

[24] Dennis Komm. *An Introduction to Online Computation: Determinism, Randomization, Advice*. Springer, 2016.

[25] Dennis Komm, Rastislav Královič, Richard Královič, and Christian Kudahl. Advice complexity of the online induced subgraph problem. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 58 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 59:1–59:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[26] Bernhard Korte and Dirk Hausmann. An analysis of the greedy heuristic for independence systems. *Annals of Discrete Mathematics*, 2:65–74, 1978.

[27] Dexter C. Kozen. *The Design and Analysis of Algorithms*. Springer, 1992.

[28] G. Ladas, Ch. G. Philos, and Y. G. Sficas. Necessary and sufficient conditions for the oscillation of difference equations. *Libertas Mathematica*, 9, 1989.

[29] Eugene Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, 1976.

[30] Nicole Megow, Martin Skutella, José Verschae, and Andreas Wiese. The power of recourse for online MST and TSP. *SIAM Journal on Computing*, 45(3):859–880, 2016.

[31] Dror Rawitz and Adi Rosén. Online budgeted maximum coverage. In *24th Annual European Symposium on Algorithms (ESA)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73:1–73:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, 2016.

[32] Peter Rossmanith. On the advice complexity of online edge- and node-deletion problems. In *Adventures Between Lower Bounds and Higher Altitudes*, volume 11011 of *Lecture Notes in Computer Science*, pages 449–462. Springer, 2018.

[33] Barna Saha and Lise Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *SIAM International Conference on Data Mining*, pages 697–708. SIAM, 2009.

[34] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[35] Robert Endre Tarjan. *Data Structures and Network Algorithms*, volume 44 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1983.

[36] Carsten Thomassen. Kuratowski's theorem. *Journal of Graph Theory*, 5(3):225–241, 1981.

[37] Doratha E. Drake Vinkemeier and Stefan Hougardy. A linear-time approximation algorithm for weighted matchings in graphs. *ACM Transactions on Algorithms*, 1(1):107–122, 2005.

[38] Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, 1935.