Relaxing the Irrevocability Requirement for Online Graph Algorithms*

Joan Boyar[1], Lene M. Favrholdt[1], Michal Kotrbčík[2], and Kim S. Larsen[1]

[1] University of Southern Denmark, Odense, Denmark,
{joan,lenem,kslarsen}@imada.sdu.dk
[2] The University of Queensland, Brisbane, Australia, m.kotrbcik@uq.edu.au

Abstract. Online graph problems are considered in models where the irrevocability requirement is relaxed. Motivated by practical examples where, for example, there is a cost associated with building a facility and no extra cost associated with doing it later, we consider the Late Accept model, where a request can be accepted at a later point, but any acceptance is irrevocable. Similarly, we also 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. For Independent Set, the Late Accept/Reject model is necessary to obtain a constant competitive ratio, but for Vertex Cover the Late Accept model is sufficient and for Minimum Spanning Forest the Late Reject model is sufficient. The Matching problem has a competitive ratio of 2, but in the Late Accept/Reject model, its competitive ratio is $\frac{3}{2}$.

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 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 absence 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 problems, *Independent Set*, *Matching*, *Vertex Cover*, and *Minimum Spanning Forest*. Independent

* Supported in part by the Danish Council for Independent Research, Natural Sciences, grant DFF-1323-00247, and the Villum Foundation, grant VKR023219.

Set and 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. 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.

Relaxed irrevocability

For the four 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. 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, 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. 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 [2], 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 [16, 13, 12, 4, 14]. For most other problems, late rejection is usually called *preemption* and has been studied in variants of many online problems, for example call control [1, 9], maximum coverage [23, 22], and weighted matching problems [6, 7]. Preemption was also previously considered for one of the problems we consider here, independent set, in [19], 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.

Online Vertex Cover was studied in [5], where they 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 is studied in, for example, [15, 21, 10, 11] 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 [17] in a model where the actual acceptances and rejections (rejections carry a cost) are made at any time.

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 [24, 18], where the performance of an online algorithm is compared to the optimum algorithm OPT via the competitive ratio. 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, we say that 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 ratio of all online algorithms for the problem. For all combinations of the problem and the model, we obtain matching lower and upper bounds on the competitive ratio.

For ease of notation for our results, we adopt the following conventions to express that a problem essentially has competitive ratio n, i.e., it is true up to an additive constant. We say that a problem has competitive ratio $n - \Theta(1)$ if (i) for any algorithm, there is a constant $b > 0$ such that the strict competitive ratio is at least $n - b$, and (ii) for any constant b, there is a strictly $(n - b)$-competitive algorithm for graphs with at least $b + 1$ vertices. Similarly, we say that a problem has competitive ratio $n/\Theta(1)$ if (i) for any algorithm, there is a constant $b > 0$ such that the strict competitive ratio is at least n/b, and (ii) for any constant b, there is an n/b-competitive algorithm for graphs with at least b vertices. This notation is used in Theorems 3 and 12. For all other results, the upper bounds hold for the 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" even though the result holds for the strict competitive ratio as well.

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 other problems, the Late Accept/Reject model is necessary to get these improvements. See Table 1. 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. The technical highlights of the paper are the results for Independent Set in the Late Accept/Reject model, where, in Theorems 4 and 5, we prove matching lower and upper bounds of $3\sqrt{3}/2$ on the competitive ratio.

Table 1. Competitive ratios of the four problems in each of the four models. W is the ratio of the largest weight to the smallest.

Problem	Standard	Late Accept	Late Reject	Late Accept/Reject
Independent Set	$n-1$	$\frac{n}{\Theta(1)}$	$\lceil \frac{n}{2} \rceil$	$\frac{3\sqrt{3}}{2} \approx 2.598$
Matching	2	2	2	$\frac{3}{2}$
Vertex Cover	$n-1$	2	$n-\Theta(1)$	2
Min. 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.

The missing proofs all appear in the full paper [3].

2 Independent Set

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 Independent Set, the objective is to find an independent set of maximum cardinality. We consider online Independent Set in the vertex arrival model.

In the standard model, no online algorithm can be better than $(n-1)$-competitive, since the adversary can give independent vertices, until the algorithm accepts a vertex, v, and then only give vertices adjacent to v. On the other hand, the greedy algorithm is $(n-1)$-competitive; if it ever rejects a vertex, the graph contains at least one edge and hence, OPT accepts at most $n-1$ vertices.

Theorem 1. *For Independent Set in the standard model, the strict competitive ratio is $n-1$.*

In the Late Reject model, the greedy algorithm becomes $n/2$-competitive with the following modification. If a new vertex is adjacent to exactly one accepted vertex, v, then v is rejected and the new vertex is accepted. If the algorithm accepts only one vertex, the graph has a path containing all n vertices and Opt can accept at most $\lceil \frac{n}{2} \rceil$ vertices. For the lower bound, the adversary can give a bipartite graph by always connecting the new vertex to the only vertex (if any) accepted by the algorithm.

Theorem 2. *For Independent Set in the Late Reject model, the strict competitive ratio is $\lceil n/2 \rceil$.*

Allowing late accepts helps further, but not enough to obtain a finite (constant) competitive ratio. For a given positive constant c, an algorithm which does not accept any vertex until the presented graph has an independent set of size at least c, and then accepts any such set, is n/c-competitive. For the lower bound, if the adversary starts the input sequence with isolated vertices, any algorithm with a bounded competitive ratio will have to accept a vertex, v, at some point. From this point on, the adversary can give vertices with only v as a neighbor. If v was the cth vertex in the input, the algorithm can accept at most the first c vertices.

Theorem 3. *For Independent Set in the Late Accept model, the competitive ratio is $n/\Theta(1)$.*

The following two theorems show that, in the Late Accept/Reject model, the optimal competitive ratio for Independent set is $3\sqrt{3}/2$. The upper bound comes from a variant of the greedy algorithm, Algorithm 1, rejecting a set of vertices if it can be replaced by a set at least $\sqrt{3}$ as large. The algorithmic idea is natural and has been used before (with other parameters than $\sqrt{3}$) in [22, 23], for example. Thus, the challenge lies in deciding the parameter and proving the resulting competitive ratio. Pseudocode for Algorithm 1 is given below.

For Algorithm 1, 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 $V - (R \cup S)$ of vertices that have not been accepted (and, hence, not late-rejected).

For a set U of vertices, let $N(U) = \cup_{v \in U} N(v)$, where $N(v)$ is the neighborhood of a vertex v (not including v). We call a set, T, of vertices *admissible* if all the following conditions are satisfied:

1) T is an independent set;
2) $T \subseteq P$;
3) $|T| \geq \sqrt{3}\,|N(T) \cap S|$.

Algorithm 1: Algorithm for Independent Set in the Late Accept/Reject model.

```
Result: Independent set $S$
1 $S = \emptyset$
2 while a vertex $v$ is presented do
3     if $S \cup \{v\}$ is independent then
4         $S = S \cup \{v\}$
5     else
6         while there exists an admissible set do
7             Let $T$ be an admissible set minimizing $|S \cap N(T)|$
8             $S = (S - N(T)) \cup T$
```

For the analysis of Algorithm 1, 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 independent set constructed by Opt. For any set, U, of vertices, we let $U^+ = U \cap O$ and $U^- = U - O$. Thus, $O = P^+ \cup S^+ \cup R^+ = P^+ \cup A^+ \cup B^+ \cup R^+$.

The following lemma follows from the fact that each time a set, X, of vertices is moved from S to R, a set at least $\sqrt{3}$ times as large as X is added to B.

Lemma 1. *When Algorithm 1 terminates,* $|B| \geq (\sqrt{3}-1)|R|$.

The next lemma follows from the facts that when the algorithm terminates, P^+ is not admissible and $P^+ \cup S^+$ is independent, since $P^+ \cup S^+ \subseteq O$.

Lemma 2. *When Algorithm 1 terminates,* $|P^+| < \sqrt{3}\,|S^-|$.

Lemma 3. *When Algorithm 1 terminates,* $|B^-| + |R^-| \geq \sqrt{3}\,|R^+|$.

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

$$|T^-| \geq \sqrt{3}|Q^+| \tag{1}$$

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

$$|T| \geq \sqrt{3}|Q| \tag{2}$$

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

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

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

Let $T_1, T_2, \ldots, T_k$ be all the 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,

$$|B^-| + |R^-| \geq \sum_{i=1}^{k} |T_i^-| \geq \sum_{i=1}^{k} \sqrt{3}|Q_i^+| = \sqrt{3}|R^+|,$$

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

Using $(\sqrt{3}+1)(|B^+|+|R^+|) = \sqrt{3}(|B^+|+|R^+|)+|B|+|R|-(|B^-|+|R^-|)$, we obtain the following lemma via simple calculations using Lemmas 1 and 3.

Lemma 4. *When Algorithm 1 terminates,* $|B^+| + |R^+| \leq \frac{\sqrt{3}}{\sqrt{3}+1}|B^+| + \frac{\sqrt{3}}{2}|B|$.

The upper bound now follows from simple calculations using Lemmas 2 and 4:

Theorem 4. *For Independent Set in the Late Accept/Reject model,* Algorithm 1 *is strictly* $3\sqrt{3}/2$*-competitive.*

We prove a matching lower bound:

Theorem 5. *For Independent Set in the Late Accept/Reject model, the competitive ratio is at least* $3\sqrt{3}/2$.

Proof (Sketch of a 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 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 to a bag, B', we start the next bag, B''. All that this means is that the vertices we give from this point on and until Alg switches bag again belong to B'', and Alg never holds vertices in the newest bag.

Now we argue that as long as we keep giving vertices, Alg will repeatedly have to switch bag in order to be c-competitive. Choose some $\varepsilon > 0$, let B be Alg's bag, B' be the new bag, and s be the number of vertices which are not adjacent to any vertices in B'. 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 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 "long enough" 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 j, 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 $ca_{j-1} \geq n_j - 1 + s_{j-2}$, 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, $ca_j \geq a_1$ must hold for all j. Since $a_1 \geq (n_1 - 1)/c$, we get that $a_j \geq (n_1 - 1)/c^2$. Thus, at the beginning of the input sequence, we can keep giving vertices for the first bag, making n_1 large enough such that a_j becomes large enough that $\varepsilon a_{j-1} \geq 1$. This establishes $(c + \varepsilon)a_{j-1} \geq n_j + s_{j-2}$. Trivially, $n_j \geq a_j$, so

$$(c + \varepsilon)a_{j-1} - s_{j-2} \geq a_j. \tag{4}$$

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

$$s_m > ca_m, \tag{5}$$

contradicting the assumption that ALG was c-competitive. To accomplish this, we repeatedly strengthen Ineq. (5) by replacing a_j with the bound from Ineq. (4), eventually arriving at an inequality which can be proven to hold, and then this will imply all the strengthened inequalities and, finally, Ineq. (5). □

3 Matching

A *matching* in a graph $G = (V, E)$ is a subset of E consisting of pairwise non-incident edges. For the problem called Matching, the objective is to find a matching of maximum cardinality. We study online 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 [20].

For late accept, we can use the same adversarial sequence as for the standard model: First a number of isolated edges are presented. Then, for each edge, uv, accepted by the algorithm, two edges, xu and vy, are presented.

Theorem 6. *For Matching in the Late Accept model, the competitive ratio is* 2.

Late rejects do not improve the competitive ratio either. This can be seen from a sequence starting with a number of isolated edges. For each accepted edge, uv, the adversary presents an edge vx. If the algorithm late-rejects uv (and thus accepts vx), an edge xy is presented. Otherwise, an edge zu is presented.

Theorem 7. *For Matching in the Late Reject model, the competitive ratio is* 2.

In the Late Accept/Reject model, the competitive ratio is 3/2. Again, the adversarial sequence starts with a number of isolated edges. If an edge uv is accepted at any point, the adversary presents edges xu and vy. If uv is then late-rejected, edges $x'x$ and yy' are presented.

Theorem 8. *For Matching in the Late Accept/Reject model, the competitive ratio is at least* 3/2.

To prove a matching upper bound, we give an algorithm, Algorithm 2, 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 [26].

Algorithm 2: Algorithm for maximal matching in the Late Accept/Reject model.

```
  Result: Matching $M$
1 $M = \emptyset$
2 while an edge $e$ is presented do
3     if $M \cup \{e\}$ is a matching then
4         $M = M \cup \{e\}$
5     if there is an augmenting path $xuvy$ of length 3 then
6         $M = (M \cup \{ux, vy\}) - \{uv\}$
```

The fact that Algorithm 2 is a Late Accept/Reject algorithm follows from the observation that no matched vertex ever becomes unmatched again. For the upper bound, we use that if a maximal matching M does not admit augmenting paths of length 3, then $3|M| \geq 2|OPT|$. This fact is easy to prove and can be found as Lemma 2 of [8], for example.

Theorem 9. *For Matching in the Late Accept/Reject model, Algorithm 2 is strictly* $3/2$*-competitive.*

4 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 Vertex Cover, the objective is to find a vertex cover of minimum cardinality. We study online Vertex Cover in the vertex arrival model.

In the standard model, no online algorithm can be better than $(n-1)$-competitive: The adversary can present isolated vertices until some vertex, v, is rejected, and then vertices that are adjacent only to v. On the other hand, this competitive ratio is obtained by the algorithm that accepts only the second endpoint of each uncovered edge.

Theorem 10. *For Vertex Cover in the standard model, the strict competitive ratio is* $n-1$.

Late accept changes the situation dramatically, since then the 2-approximation algorithm adding both endpoints of each uncovered edge can be used. Adding late rejects does not change the situation further; if the algorithm ever late-rejects a vertex, v, the adversary can add arbitrarily many neighbors of v.

Theorem 11. *For Vertex Cover in the Late Accept model and the Late Accept/Reject model, the competitive ratio is* 2.

In the Late Reject model, the competitive ratio is $n - \Theta(1)$. For the lower bound, the adversary can give isolated vertices until some vertex, v, is rejected, and then arbitrarily many neighbors of v. The upper bound is obtained by a family ALG_b of algorithms accepting the first $b+1$ vertices and then rejecting the vertices not part of an optimal vertex cover of the graph seen so far.

Theorem 12. *For Vertex Cover in the Late Reject model, the competitive ratio is* $n - \Theta(1)$.

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.

Even in the standard model, the competitive ratio cannot be higher than W, since all spanning forests contain the same number of edges. A matching lower bound follows from the sequence consisting of a tree of edges of weight W and then a vertex with edges of weight 1 to all previous vertices. Since an online algorithm, even in the Late Accept model model, does not know when the input ends, it must always have a forest spanning all the vertices seen so far:

Theorem 13. *For Minimum Spanning Forest in the standard model or the Late Accept model, the competitive ratio is W.*

On the other hand, in the Late Reject model, the greedy online algorithm mentioned by Tarjan in [25] can be used: Each new edge is accepted, and if this results in a cycle, the heaviest edge in the cycle is (late-)rejected. Since the Late Reject model leads to an optimal spanning tree, any model allowing that possibility inherits the result.

Theorem 14. *For Minimum Spanning Forest in the Late Reject model or the Late Accept/Reject model, the competitive ratio is* 1*.*

Future Work

Since we prove tight results for all combinations of problems and models considered, we leave no immediate open problems. However, one could reasonably consider late operations a resource to be used sparingly, as for the rearrangements in [15, 21, 10, 11], 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.

References

1. Y. Bartal, A. Fiat, and S. Leonardi. Lower bounds for on-line graph problems with application to on-line circuit and optical routing. In *28th STOC*, pages 531–540. ACM, 1996.
2. J. Boyar, S.J. Eidenbenz, L.M. Favrholdt, M. Kotrbčík, and K.S. Larsen. Online dominating set. In *15th SWAT*, volume 53 of *LIPIcs*, pages 21:1–21:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, 2016.
3. J. Boyar, L.M. Favrholdt, M. Kotrbčík, and K.S. Larsen. Relaxing the irrevocability requirements for online graph algorithms. Technical Report arXiv:1704.08835 [cs.DS], arXiv, 2017.

4. M. Cygan, Ł. Jeż, and J. Sgall. Online knapsack revisited. *Theor. Comput. Syst.*, 58(1):153–190, 2016.
5. M. Demange and V.Th. Paschos. On-line vertex-covering. *Theor. Comput. Sci.*, 332:83–108, 2005.
6. L. Epstein, A. Levin, J. Mestre, and D. Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011.
7. L. Epstein, A. Levin, D. Segev, and O. Weimann. Improved bounds for online preemptive matching. In *30th STACS*, volume 20 of *LIPIcs*, pages 389–399. Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, 2013.
8. J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2–3):207–216, 2005.
9. J.A. Garay, I.S. Gopal, S. Kutten, Y. Mansour, and M. Yung. Efficient on-line call control algorithms. *J. Algorithm.*, 23(1):180–194, 1997.
10. A. Gu, A. Gupta, and A. Kumar. The power of deferral: Maintaining a constant-competitive steiner tree online. *SIAM J. Comput.*, 45(1):1–28, 2016.
11. A. Gupta and A. Kumar. Online steiner tree with deletions. In *25th SODA*, pages 455–467, 2014.
12. X. Han, Y. Kawase, and K. Makino. Randomized algorithms for online knapsack problems. *Theor. Comput. Sci.*, 562:395–405, 2015.
13. X. Han, Y. Kawase, K. Makino, and H. Guo. Online removable knapsack problem under convex function. *Theor. Comput. Sci.*, 540:62–69, 2014.
14. X. Han and K. Makino. Online minimization knapsack problem. *Theor. Comput. Sci.*, 609:185–196, 2016.
15. M. Imase and B.M. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, 1991.
16. K. Iwama and S. Taketomi. Removable online knapsack problems. In *29th ICALP*, volume 2380 of *LNCS*, pages 293–305. Springer, 2002.
17. P. Jaillet and X. Lu. Online traveling salesman problems with rejection options. *Networks*, 64:84–95, 2014.
18. A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.
19. D. Komm, R. Královič, R. Královič, and C. Kudahl. Advice complexity of the online induced subgraph problem. In *41st MFCS*, volume 58 of *LIPIcs*, pages 59:1–59:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
20. B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. *Ann. Discrete Math.*, 2:65–74, 1978.
21. N. Megow, M. Skutella, J. Verschae, and A. Wiese. The power of recourse for online MST and TSP. *SIAM J. Comput.*, 45(3):859–880, 2016.
22. D. Rawitz and A. Rosén. Online Budgeted Maximum Coverage. In *24th ESA*, volume 57 of *LIIPCcs*, pages 73:1–73:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, 2016.
23. B. Saha and L. Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *9th SDM*, pages 697–708. SIAM, 2009.
24. D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
25. R.E. Tarjan. *Data Structures and Network Algorithms*, volume 44 of *CBMS-NSF regional conference series in applied mathematics*. SIAM, 1983.
26. D.E.D. Vinkemeier and S. Hougardy. A linear-time approximation algorithm for weighted matchings in graphs. *ACM T. Algorithms*, 1(1):107–122, 2005.