The Power of Amortized Recourse for Online Graph Problems

Alison Hsiang-Hsuan Liu(✉) and Jonathan Toole-Charignon

Department of Information and computing sciences, Utrecht University, Utrecht, The Netherlands
{h.h.liu,j.c.f.toole-charignon}@uu.nl

Abstract. In this work, we study online graph problems with monotone-sum objectives, where the vertices or edges of the graph are revealed one by one and need to be assigned to a value such that certain properties of the solution hold. We propose a general two-fold greedy algorithm that augments its current solution greedily and references yardstick algorithms. The algorithm maintains competitiveness by strategically aligning to the yardstick solution and incurring recourse. We show that our general algorithm achieves t-competitiveness while incurring at most $\frac{w_{\max}\cdot(t+1)}{t-1}$ amortized recourse for any monotone-sum problems with integral solution, where $w_{\max}$ is the largest value that can be assigned to a vertex or an edge. For fractional monotone-sum problems where each of the assigned values is between $[0,1]$, our general algorithm incurs at most $\frac{t+1}{w_{\min}\cdot(t-1)}$ amortized recourse, where $w_{\min}$ is the smallest non-negative value that can be assigned. We further show that the general algorithm can be improved for three classical graph problems. For Independent Set, we refine the analysis of our general algorithm and show that t-competitiveness can be achieved with $\frac{t}{t-1}$ amortized recourse. For Maximum Cardinality Matching, we limit our algorithm's greed to show that t-competitiveness can be achieved with $\frac{(2-t^*)}{(t^*-1)(3-t^*)}+\frac{t^*-1}{3-t^*}$ amortized recourse, where t^* is the largest number such that $t^*=1+\frac{1}{j}\leq t$ for some integer j. For Vertex Cover, we show that our algorithm guarantees a competitive ratio strictly smaller than 2 for any finite instance in polynomial time while incurring at most 3.33 amortized recourse. We beat the almost unbreakable 2-approximation in polynomial time by using the optimal solution as the reference without computing it. We remark that this online result can be used as an offline approximation result (without violating the unique games conjecture [20]) to partially improve upon the constructive algorithm of Monien and Speckenmeyer [23].

1 Introduction

Graph optimization problems serve as stems for various practical problems. A solution for such a problem can be described as an assignment from the elements of the problem (e.g. vertices of a graph) to non-negative real numbers such that the constraints between the elements are satisfied. In the online setting, the most considered models are the *vertex-arrival* and *edge-arrival* models. That is, the graph is revealed vertex-by-vertex or edge-by-edge, and once an element arrives, the *online algorithm* has to immediately make an irrevocable decision on the new element. The performance of an online algorithm is measured by *competitive ratio* against the optimal offline solution. Many graph optimization problems are non-competitive: the larger the input size, the larger the competitive ratio of any deterministic online algorithm. In other words, a non-competitive problem has no constant-competitive online algorithm.

The pure online model is pessimistic, in that altering decisions may be possible (albeit expensive) or limited knowledge about the future may be available in the real world. In this work, we investigate online graph optimization problems in the *recourse* model. That is, decisions made by the online algorithm can be revoked. In particular, we aim at finding out the amount of amortized recourse that is sufficient and/or necessary for attaining a desirable competitive ratio for a given problem.

Uncertainty and Amortized Recourse. The competitive ratio can be seen as quantification of how far the quality of an online algorithm's solution is from that of a conceptual optimal offline algorithm that has complete knowledge of the input and unlimited computational power. Therefore, the non-competitiveness of graph optimization problems suggests that uncertainty of the input is critical to these problems. However, the online algorithm may perform better when the irrevocability constraint is relaxed or knowledge about future inputs is available. It is intriguing to investigate to what extent these problems remain non-competitive under these conditions, in particular to determine how much revocability or knowledge the online algorithm needs in order to attain a desirable competitive ratio.

Beyond the practical motivation of relaxing irrevocability of online algorithms' decisions, amortized recourse also provides insight on how a given online problem is affected by uncertainty. In particular, it captures how rapidly the structure of the offline optimal solution can change: the fewer elements required to do so, the larger the amortized recourse. Furthermore, the impact of uncertainty is directly correlated with this idea: the faster the optimal solution can change, the more impact uncertainty on future inputs will have. Different problems may attain constant competitive ratios using different amounts of (amortized) recourse, which implies variability in the impact of uncertainty. For example, to attain a constant competitive ratio, one needs exactly $O(\log n)$ recourse per edge for min-cost bipartite matching [21], while one only needs a constant amount of recourse per element for maximum independent set and minimum vertex cover [10].

Online Monotone-Sum Problems. We study *online* graph problems in the vertex-arrival or the edge-arrival models. Along with the newly-revealed element, which can be a vertex or an edge according to the arrival model, there may be constraints imposed upon some subset of the currently-revealed elements that a feasible solution should satisfy. An algorithm aims at finding a feasible solution that maximizes (or minimizes) the objective. A problem is a *sum* problem if the objective is a sum of the values assigned to each element. If the value of the optimal solution of an instance is always greater than or equal to that of a subset of the instance, then the problem is a *monotone* problem.[1]

An online algorithm makes decisions upon arrival of each element. In the *recourse* model, the online algorithm can also revoke an earlier decision that it made and pay for the revocation. We aim to reduce the competitive ratio with as little total recourse (i.e. as few revocations) as possible.

Our Contribution. We propose a general online algorithm Target-and-Switch ($\texttt{TaS}_t$), which is parameterized by a target competitive ratio t and uses a yardstick algorithm as a reference. The yardstick algorithm can be ant exact optimal algorithm or an incremental (defined later) approximation algorithm if one aims at polynomial time online algorithms. Throughout the process, the $\texttt{TaS}_t$ algorithm compares itself to the yardstick algorithm's solution and strategically switches its solution to that of the yardstick algorithm. Overall, the $\texttt{TaS}_t$ algorithm provides a trade-off between amortized recourse and competitive ratio for arbitrary monotone-sum graph problems. In particular, we consider two measurements of recourse cost: number of reassigned elements, or the amount of change in the reassigned values. Our result works for both unweighted and weighted problems, and it even works for fractional optimization problems, where the smallest non-zero value assigned to a single element can be a real number between 0 and 1. The following is the main result of our work, where the bound of amortized recourse works for both measurements of recourse cost (Theorem 1 and Corollary 1).

Main result (informal). *Using an optimal algorithm (resp. an incremental α-approximation algorithm, defined formally in Sect. 2) as the yardstick, $\texttt{TaS}_t$ is t-competitive (resp. $(t \cdot \alpha)$-competitive) and incurs at most $\frac{w_{max}\cdot(t+1)}{\min\{1,w_{min}\}\cdot(t-1)}$ amortized recourse for any monotone-sum graph problem where w_{max} and w_{min} are the maximum and minimum non-zero values that can be assigned to an element.*[2]

$\texttt{TaS}_t$ is two-fold greedy. First, it assigns the value *greedily* once an element arrives. Second, the algorithm aligns its solution to the yardstick solution *completely* and incurs recourse when the current solution fails to be t-competitive against the yardstick solution.

In general, the $\texttt{TaS}_t$ algorithm works for any optimization problem. The challenge is to bound the amortized recourse that it incurs, as the complete alignment may require a vast amount of recourse. By looking closer at a specific problem, we can show a tighter bound on the amount of recourse needed. We use a sophisticated analysis for the INDEPENDENT SET problem and improve the recourse bound (Theorem 2).

[1] The DOMINATING SET and MATCHING WITH DELAYS problems are sum problems but not monotone. The COLORING PROBLEM is monotone but not a sum problem.

[2] The bound of amortized recourse $\frac{w_{\max}\cdot(t+1)}{w_{\min}\cdot(t-1)}$ is larger when the elements can be assigned minimum non-zero values smaller than 1. For example, the fractional VERTEX COVER problem in [24].

The two-fold greedy algorithm may perform better when the greediness is relaxed. Moreover, by choosing different yardstick algorithms and tuning the alignment to the yardstick carefully, the amortized recourse can be further reduced. We show that for the Maximum Cardinality Matching problem, partially aligning to the yardstick solution is more recourse-efficient (Theorem 5).

For the Vertex Cover problem, we show that a special version of $\texttt{TaS}_t$ with $t = 2 - \frac{2}{\texttt{OPT}}$ incurs a very small amount of amortized recourse (Theorem 8) and is $(2 - \frac{2}{\texttt{OPT}})$-competitive, where $\texttt{OPT}$ is the size of the optimal vertex cover[3] (Theorem 7). Our algorithm uses an optimal solution as a yardstick. The key to the polynomial time complexity is that instead of explicitly finding the yardstick assignment, we show that the yardstick cannot be too "far" from our solution at any moment if the target competitive ratio $2 - \frac{2}{\texttt{OPT}}$ is not already achieved. More specifically, by restricting the range of greedy choice, we can show that the yardstick solution can be aligned partially within a constant amount of amortized recourse. Thus, our result breaks the almost unbreakable 2-approximation for the Vertex Cover problem and improves upon that of Monien and Speckenmeyer [23] for a subset of the graphs containing odd cycles of length no less than $2k+3$ (for which $2 - \frac{2}{\texttt{OPT}} < 2 - \frac{1}{k+1}$), using an algorithm that is also constructive.

Our results are summarized in Table 1, which illustrates the power of amortization.

Table 1. Summary of our results. Note that t can be any real number larger than 1. For Maximum Matching, t^* is the largest number such that $t^* \leq t$ and $t^* = 1 + \frac{1}{j}$ for some integer j. The note P means that the algorithm is a polynomial-time online algorithm.

	(Competitive ratio, worst case recourse)	(Competitive ratio, amortized recourse)
Monotone-sum problems		$(t\alpha, \frac{w_{\max}\cdot(t+1)}{\min\{w_{\min},1\}\cdot(t-1)})$ (Theorem 1, Corollary 1, P with incremental α-approximation algorithms)
Maximum independent set	$(2.598, 2)$ [10]	$(t, \frac{t}{t-1})$ (Theorem 2) $(2.598, 1.626)$ (Theorem 2)
Maximum matching	$(k, O(\frac{\log k}{k}) + 1))$ [2] $(1.5, 2)$ [10]	$(t, \frac{(2-t^*)}{(t^*-1)(3-t^*)} + \frac{t^*-1}{3-t^*})$ (Theorem 5, P) $((1.5, 1)$ with $t^* = 1.5)$
Minimum vertex cover	$(2, 1)$ [10]	$(2 - \frac{2}{\texttt{OPT}}, \frac{10}{3})$ (Theorem 8, P)

Related Work. Typically, online graph problems such as maximum independent set, maximum cardinality matching, minimum vertex cover, minimum dominating set, can be modeled as inclusion/exclusion problems. In these problems, any individual element (vertex or edge as appropriate) is either included in or

[3] Note that over all instances, $\texttt{OPT}$ can be arbitrarily large. Thus, there is no $\varepsilon > 0$ for which $2 - \frac{2}{\texttt{OPT}} \leq 2 - \varepsilon$ over all instances. Therefore, our result does not violate the unique games conjecture [20].

excluded from the solution. In our terminology, the inclusion and exclusion of an element is assigning values 1 and 0 to it, respectively. Recourse is incurred each time the inclusion/exclusion status of an element is changed. The min-cost matching, Steiner tree, facility location, and routing problems are also inclusion/exclusion problems, but the cost of an edge/vertices inclusion is weighted. That is, the edges/vertices are assigned values 0 or 1, and the cost incurred by a value-1 edge/vertex is its weight. The scheme also works for non-monotone sum problems. For example, in coloring and bin-packing problems, the assignment of vertices/items to colors/bins can be modeled as assigning the vertices/items non-negative integral numbers, which are the indices of the colors/bins.

The closest previous result is the work by Boyar et al. [10]. The authors investigated the Independent Set, Maximum Cardinality Matching, Vertex Cover, and Minimum Spanning Forest problems, which are all non-competitive in the pure online model. The authors showed that the competitive ratio of these problems can be massively reduced to a constant by incurring at most 2 recourse for any single element. Note that the bounds of the worst case recourse are upper bounds of the amortized recourse. Moreover, the algorithms in [10] incur at least 1.5 amortized recourse for the Maximum Cardinality Matching problem and at least 0.5 amortized recourse for the Vertex Cover problem.

There is a line of research on online matching problems with recourse. Angelopoulos et al. [2] studied a more general setting for Maximal Cardinality Matching and showed that given that no element incurs more than k recourse, there exists an algorithm that attains a competitive ratio of $1 + O(1/\sqrt{k})$. Megow and Nölke [21] showed that for the Min-Cost Bipartite Matching problem, constant competitiveness is achievable with amortized recourse $O(\log n)$, where n is the number of requests. Bernstein et al. [6] showed that there exists an algorithm that achieves 1-competitiveness with $O(\log^2 n)$ amortized recourse for the Bipartite Matching problem, where n is the number of vertices inserted. The result also shows that to achieve 1-competitiveness for Vertex Cover, any online algorithm needs at least $\Omega(n)$ amortized recourse per vertex.

In addition, there has been extensive work on online algorithms in the recourse model for a variety of different problems. For amortized recourse, studied problems include online bipartite matching [6], graph coloring [9], minimum spanning tree and traveling salesperson [22], Steiner tree [12], online facility location [11], bin packing [13], submodular covering [14], and constrained optimization [3].

Graph problems model various real-world issues whose performance guarantees are often abysmal, as they are notoriously non-competitive in the pure online model. Prior work has shown curiosity about the conditions under which these problems become competitive, and these problems have been investigated under different models out of both practical and theoretical interests. Other than the recourse model, considered models include paying for a delay in the timing of decision making to achieve a better solution [5,7]. Another model for delayed decision making is the reordering buffer model [1], where the online algorithm can delay up to k decisions by storing the elements in a size-k buffer.

The impact of extra knowledge about the input has also been studied. For example, once a vertex arrives, the neighborhood is known to the algorithm [16]. In the lookahead model, an online algorithm is capable of foreseeing the next events [1]. Predictions provided by machine learning are also considered for graph problems [4]. Finally, there are also works where the integral assignment restrictions are relaxed for vertex cover and matching problems [24].

Another major area of related work for practically any problem considered in the online model is polynomial-time approximation algorithms for the equivalent problem in the offline setting. The link between the two is particularly salient when considering a polynomial-time online algorithm, as this online algorithm can also be run in polynomial time in the offline setting by processing the graph as if it were revealed in an online manner.

In the case of minimum vertex cover, assuming the unique games conjecture, it is not possible to obtain an approximation factor of $(2 - \varepsilon)$ for fixed $\varepsilon > 0$ [20]. However, results have been obtained for parameterized ε. In particular, Halperin [15] showed an approximation factor of $2 - (1 - o(1))\frac{2 \ln \ln \Delta}{\ln \Delta}$ on graphs with maximum degree Δ, and Karakostas [19] showed an approximation factor of $2 - \theta(\frac{1}{\sqrt{\log n}})$. Both of these results use semidefinite relaxations of the problem, whereas Monien and Speckenmeyer [23] had previously used a constructive approach to show an approximation factor of $2 - \frac{1}{k+1}$ for graphs without odd cycles of length at most $2k + 1$.

Paper Organization. Section 2 defines monotone-sum graph problems and the amortized recourse model. We propose a general algorithm $\texttt{TaS}_t$ for finding the trade-off between the desired competitive ratio and the amortized recourse needed. Section 3 provides a refined analysis on the $\texttt{TaS}_t$ algorithm on the Independent Set problem. Section 4 discusses an existing algorithm [2], which is a variant of $\texttt{TaS}_t$ algorithm, for the Maximum Cardinality Matching problem that is less greedy in aligning its solution and obtains a better trade-off. Section 5 introduces a polynomial-time version of $\texttt{TaS}_t$ algorithm for the Vertex Cover problem that limits both greedy aspects. This algorithm can also be used as a novel offline approximation algorithm for certain graph classes. Due to space constraints, we only provide proof ideas for the theorem. The detailed proof for all lemmas and theorems can be found in the full version of this paper.

2 Monotone-Sum Graph Problems and a General Algorithm

For an *online graph problem* Q on a graph $G = (V, E)$, which is unknown a priori, we consider either the *vertex-arrival* model or the *edge-arrival* model. In the vertex-arrival model (resp. edge-arrival model), the elements in V (resp. elements in E) arrive one at a time, and an algorithm has to assign each element a non-negative value in $[0, w_{\max}]$ such that the assignment satisfies certain properties associated with Q. Formally, the *assignment* is defined as $\mathcal{A} : \mathcal{X} \to \mathbb{R}^+$, where

$\mathcal{X}$ is V or E, such that $\mathcal{A}(\mathcal{X})$ satisfies a set of properties $\mathcal{P}_Q$. The *value* of a feasible assignment $\mathcal{A}$ is defined as a function *value* : $\mathcal{X} \times \mathcal{A}(\mathcal{X}) \rightarrow \mathbb{R}^+$, which should be minimized or maximized as appropriate. In this work, we focus on the problems with *sum* objectives, that is, $value(\mathcal{X}, \mathcal{A}(\mathcal{X})) = \sum_{x \in \mathcal{X}} \mathcal{A}(x)$. Moreover, we concern ourselves about the impact of lacking information on the optimality of the solution. Therefore, we consider monotone sum graph problems where given a feasible assignment and a newly-arrived element, there is always a value in $[0, w_{\max}]$ that can be assigned to the new element such that the new assignment is feasible.[4]

We denote the assignment on input $\mathcal{X}$ returned by the algorithm $\texttt{ALG}$ by $\texttt{ALG}(\mathcal{X})$. We abuse the notation $\mathcal{X}$ to denote the graph revealed by the input $\mathcal{X}$. We further abuse notation and denote the total value of the assignment by $\texttt{ALG}(\mathcal{X})$ as well. That is, $\texttt{ALG}(\mathcal{X}) = \sum_{x_i \in \mathcal{X}} \texttt{ALG}(x_i)$. When the context is clear, the parameter $\mathcal{X}$ is dropped.

We study the family of monotone-sum graph problems, which is defined as follows. Similarly, we define the family of incremental algorithms. Note that a monotone-sum problem can be a maximization or a minimization problem.

Definition 1. *The* projection *of an assignment $\mathcal{A}(G)$ on an induced subgraph H of G assigns to each element in H the same value that $\mathcal{A}(G)$ does in G.*

Definition 2. Monotone-sum graph problems. *A sum problem is* monotone *if for any graph G and any induced subgraph H of G, 1) the projection of any feasible assignment $\mathcal{A}(G)$ on H is also feasible, and 2) $\texttt{OPT}(H) \leq \texttt{OPT}(G)$, where $\texttt{OPT}$ is an optimal solution.*

Definition 3. Incremental algorithms. *An algorithm $\texttt{ALG}$ is* incremental *if for any graph G corresponding to the instance $\mathcal{X}$ and any induced subgraph H of G, $\texttt{ALG}(H) \leq \texttt{ALG}(G)$. Furthermore, the projection of $\texttt{ALG}(\mathcal{X})$ on a prefix $\mathcal{X}'$ of instance $\mathcal{X}$ does not have a better objective value than the assignment $\texttt{ALG}(\mathcal{X}')$.*[5]

In this work, the performance of an online algorithm is measured by the *competitive ratio*. An online algorithm $\texttt{ALG}$ attains a competitive ratio of t if $\max\{\frac{\texttt{ALG}(\mathcal{X})}{\texttt{OPT}(\mathcal{X})}, \frac{\texttt{OPT}(\mathcal{X})}{\texttt{ALG}(\mathcal{X})}\} \leq t$ for any instance $\mathcal{X}$, where $\texttt{OPT}$ is the optimal offline algorithm that knows all information necessary for solving the problem. In the recourse model, the online algorithm can revoke its decisions and incurs *recourse cost*. There are two types of recourse cost considered in this paper:

- **Type-1:** The recourse cost is defined as the *number* of elements which assignment values are changed. Formally, $\sum_{x_i \in \mathcal{X}} \mathbb{1}[A_1(x_i) \neq A_2(x_i)]$ when an assignment on instance $\mathcal{X}$ is changed from $A_1(\mathcal{X})$ to $A_2(\mathcal{X})$.
- **Type-2:** The recourse cost is defined as the *amount of change* of the assignment value. Formally, $\sum_{x_i \in \mathcal{X}} |A_1(x_i) - A_2(x_i)|$ when an assignment on instance $\mathcal{X}$ is changed from $A_1(\mathcal{X})$ to $A_2(\mathcal{X})$.

[4] Classical graph problems such as Independent Set, Maximum Cardinality Matching, and Vertex Cover all satisfy this property.

[5] For example, the Ramsey algorithm in [8] is an incremental algorithm. Also note that any online algorithm is an incremental algorithm.

We study the trade-off between the competitive ratio and the *amortized recourse.* That is, the total incurred recourse cost divided by the number of elements that should be assigned a value in the final instance. We define a family of algorithms for monotone-sum problems.

Target-and-Switch ($\texttt{TaS}_t$) Algorithm. The $\texttt{TaS}_t$ algorithm uses a yardstick algorithm REF as a reference, where the yardstick can be the optimal algorithm or an incremental α-approximation algorithm. Throughout the process, $\texttt{TaS}_t$ keeps track of the yardstick solution value. Once a new element arrives, $\texttt{TaS}_t$ greedily assigns a feasible value[6] to the newly-revealed element if this assignment remains t-competitive relative to the yardstick algorithm's solution. Otherwise, $\texttt{TaS}_t$ *switches* its assignment to the one by the yardstick algorithm and incurs recourse. (See Algorithm 1).

Algorithm 1. $\texttt{TaS}_t$ algorithm for monotone-sum graph problems

```
ALG ← 0
while new element $v$ arrives do
    $g$ ← the best value from $[0, w_{\max}]$ such that no feasibility constraint is violated
    if the new assignment will fail to be $t$-competitive then ▷ $\max\{\frac{\texttt{ALG}+g}{\texttt{OPT}}, \frac{\texttt{OPT}}{\texttt{ALG}+g}\} > t$
        SWITCH(OPT)
    else
        incorporate the greedy assignment
    end if
    ALG ← the value of $\texttt{TaS}_t$'s current assignment
end while

function SWITCH(assignment $A$)
    for every element $x$ do
        if $\texttt{TaS}_t(x) \neq A(x)$ then
            change the assignment of element $x$ into $A(x)$
        end if
    end for
end function
```

Now, we show that the $\texttt{TaS}_t$ algorithm achieves the desired competitive ratio t with at most polynomial of t amortized recourse. In our analysis, we use the following observation heavily (including for Theorem 1).

Observation 1. *For all $x_i \geq 0$ and $y_i > 0$, $\frac{\Sigma_i x_i}{\Sigma_i y_i} \leq \max_i \frac{x_i}{y_i}$.*

Theorem 1. *Using an optimal algorithm (resp. incremental α-approximation algorithm) as the yardstick, $\texttt{TaS}_t$ is t-competitive (resp. $(t \cdot \alpha)$-competitive) and incurs at most $\frac{w_{max} \cdot (t+1)}{t-1}$ **Type-2** amortized recourse for any monotone-sum graph problem where w_{max} is the maximum value that can be assigned to an element. The bound also works for **Type-1** amortized recourse.*

[6] Note that there always exists a value such that the new assignment is feasible since the problem is monotone.

Proof **(Ideas.)** We can show that any optimal solution satisfies the incremental property (see the full version) and thus can be seen as an incremental 1-approximation algorithm.

Since recourse is incurred only at the moments when a switch happens in the $\texttt{TaS}_t$ algorithm, we partition the process of the algorithm into *phases* according to the switches. Phase i consists all the events after the $(i-1)$-th switch until the i-th switch. By Observation 1, the amortized recourse for the whole instance is bounded by the maximum amortized recourse incurred within a phase. Therefore, we consider the amortized recourse incurred by the $(i+1)$-th switch for arbitrary $i \geq 0$.

Let $\texttt{REF}_i$ and $\texttt{TaS}_i$ denote the value of the yardstick algorithm's solution and the $\texttt{TaS}_t$ algorithm's solution right *after* the i-th switch, respectively. By construction, $\texttt{TaS}_i = \texttt{REF}_i$. Let $\texttt{ALG}$ denote the value of $\texttt{TaS}_t$'s solution right *before* the arrival of x, which triggers the $(i+1)$-th switch. The total **Type-2** recourse cost is at most $\texttt{ALG} + \texttt{REF}_{i+1}$ (where the $\texttt{TaS}_t$ algorithm changes the value on every element to zero and then changes it to the $\texttt{REF}$ assignment).

The main ingredients for the proof are:

– **Property 1:** Monotonicity of the problem and the incremental nature of $\texttt{REF}$ implies that $\texttt{REF}_i \leq \texttt{REF}_{i+1}$.
– **Property 2:** The incremental nature of $\texttt{TaS}_t$ during a phase implies that $\texttt{ALG} \geq \texttt{REF}_i$.
– **Property 3:** By the switching condition of $\texttt{TaS}_t$, $\texttt{ALG} < \texttt{REF}_{i+1}/t$ for maximization problems, and $\texttt{ALG} + w_{\max} > t \cdot \texttt{REF}_{i+1}$ for minimization problems.

Maximization Problems. By **Property** 1 and the fact that the assigned values are at most $w_{\max}$, we can show that the adversary needs to release at least $\frac{\texttt{REF}_{i+1} - \texttt{REF}_i}{w_{\max}}$ elements such that the yardstick assignment value increases enough to trigger the switch. By **Property** 2 and **Property** 3, $\frac{\texttt{REF}_{i+1} - \texttt{REF}_i}{w_{\max}} \geq \frac{(1-1/t) \cdot \texttt{REF}_{i+1}}{w_{\max}}$. By **Property** 3, the total recourse incurred by the $(i+1)$-th switch is at most $\texttt{ALG} + \texttt{REF}_{i+1} < (1+1/t) \cdot \texttt{REF}_{i+1}$. Hence, the **Type-2** amortized recourse incurred in phase $i+1$ is bounded by $\frac{w_{\max} \cdot (1+1/t) \cdot \texttt{REF}_{i+1}}{(1-1/t) \cdot \texttt{REF}_{i+1}} = \frac{w_{\max} \cdot (t+1)}{t-1}$.

Minimization Problems. In minimization problems, the $(i+1)$-th switch may be triggered by shifting the $\texttt{REF}$ assignment completely but without changing its value. In this case, a massive amount of recourse is incurred by a single input. However, we can show by **Property** 3 that in this case, the $\texttt{ALG}$ value must be large enough to trigger the switch. Thus, we can bound the number of elements released during phase $i+1$ by the change of $\texttt{TaS}_t$ assignment's total value. That is, it is at least $\frac{\texttt{ALG} - \texttt{TaS}_i}{w_{\max}} + 1 = \frac{\texttt{ALG} - \texttt{REF}_i}{w_{\max}} + 1$, where the 1 is the element which triggers the switching. By **Property** 1 and **Property** 2, the number is at least $\frac{(1-1/t) \cdot (\texttt{ALG} + w_{\max})}{w_{\max}}$. By **Property** 3, the total recourse incurred by the $(i+1)$-th switch is at most $\texttt{ALG} + \texttt{REF}_{i+1} < (1+1/t) \cdot \texttt{ALG} + w_{\max}/t$. Therefore, the **Type-2** amortized recourse incurred in phase $i+1$ is bounded by $\frac{w_{\max} \cdot ((1+1/t) \cdot \texttt{ALG} + w_{\max}/t)}{(1-1/t) \cdot (\texttt{ALG} + w_{\max})} \leq \frac{w_{\max} \cdot (t+1)}{t-1}$. $\square$

The yardstick algorithm can be the optimal offline algorithm. Since the problem is monotone, our algorithm can be t-competitive for arbitrary $t > 1$. Furthermore, if we apply a polynomial-time incremental α-approximation algorithm as the yardstick, then our algorithm also runs in polynomial time.

The results work for weighted versions of problems, and it also work for fractional assignment problems, where the value assigned to any element is in $[0, 1]$ (for example, the fractional Vertex Cover problem in [24]). In this case, the **Type-2** amortized recourse is bounded above by the **Type-1** amortized recourse:

Corollary 1. *For a fractional monotone-sum problem, $\texttt{TaS}_t$ is $(t\cdot\alpha)$-competitive and incurs at most $\frac{t+1}{w_{min}\cdot(t-1)}$* ***Type-1*** *amortized recourse using an incremental α-approximation algorithm as the yardstick. The bound also works for* ***Type-2*** *amortized recourse.*

The monotone-sum problem property captures many classical graph optimization problems such as Independent Set, Maximum Cardinality Matching, and Vertex Cover. The three problems can be interpreted as a special case of general monotone-sum problems as follows.

Independent Set Problem in Vertex-Arrival Model. Vertices arrive one at a time and should be assigned a value 0 or 1. Once a vertex is revealed, the edges between it and its previously-revealed neighbors are known. The goal is to find a maximum value assignment such that for any edge, the sum of values assigned to the two endpoints is at most 1.

Maximum Cardinality Matching Problem in Vertex-Arrival or Edge-Arrival Model. Edges or vertices arrive one at a time and each of the edges should be assigned a value 0 or 1. The goal is to find a maximum value assignment such that for any vertex, the sum of values assigned to its incident edges is at most 1.

Vertex Cover Problem in Vertex-Arrival Model. Vertices arrive one at a time and should be assigned a value 0 or 1. Once a vertex is revealed, the edges between it and its previously-revealed neighbors are known. The goal is to find a minimum value assignment such that for any edge, the sum of values assigned to its two endpoints is at least 1.

Since the available value for each element is either 0 or 1 in these three problems, we say that an element is *accepted* if it is assigned a value 1. Similarly, an element is *rejected* if it is assigned a value 0. An element is *late-accepted* if its value is changed from 0 to 1 after its arrival, and *late-rejected* if its value is changed from 1 to 0 after its arrival. Furthermore, since the value for any element only changes between 0 and 1, the **Type-1** recourse cost and **Type-2** recourse cost are equivalent in these three problems. Therefore, we have the following corollary.

Corollary 2. *The* $\texttt{TaS}_t$ *algorithm attains competitive ratio* $t > 1$ *while incurring at most* $\frac{t+1}{t-1}$ *(**Type-1** or **Type-2**) amortized recourse for* Independent Set, Maximum Cardinality Matching, *and* Vertex Cover *problems.*

3 Maximum Independent Set

For the maximum independent set problem in the vertex-arrival model, the algorithm proposed by Boyar et al. incurs at most 2 amortized recourse while maintaining a competitive ratio of 2.598 [10]. By Theorem 1, the general $\texttt{TaS}_t$ algorithm incurs at most $\frac{t+1}{t-1}$ amortized recourse and guarantees a competitive ratio of t. In this section, we show that the amortized recourse incurred by $\texttt{TaS}_t$ is even smaller by a more sophisticated analysis.

Lemma 1 (Instance reduction). *For any instance* (G, σ) *of the maximum independent set problem, there exists an instance* (G', σ') *for which any newly revealed vertex is either accepted by* $\texttt{TaS}_t$ *or is part of the optimal offline solution when* $\texttt{TaS}_t$ *incurs its next switch, but not both, such that the amortized recourse for* (G', σ') *is at least that for* (G, σ).

By Lemma 1, given any input (G, σ), its amortized recourse incurred by $\texttt{TaS}_t$ is bounded above by that of its reduced instance (G', σ'). In Theorem 2, we provide an upper bound of the amortized recourse incurred by $\texttt{TaS}_t$ against any reduced instance, and thus that this upper bound holds for any instance.

Theorem 2. *For the maximum independent set problem, given a target competitive ratio* $t > 1$, $\texttt{TaS}_t$ *is* t*-competitive while incurring at most* $\frac{t}{t-1}$ *amortized recourse.*

Proof **(Ideas).** We prove this theorem by using the same phase partition argument in the proof of Theorem 1 on any reduced instance (G', σ'). Assume that every vertex is released with some "budget" B, which is a function of t, for later recourse. Our attempt is to find a sufficient B such that the total recourse incurred by $\texttt{TaS}_t$ in one phase is no more than the total recourse budget carried by the vertices released in this phase. That is, the total recourse incurred by one switch can be "paid" by the recourse budget from the newly revealed vertices. By Lemma 1 and Observation 1, $\texttt{TaS}_t$ incurs at most B amortized recourse.

By Lemma 1, in the reduced instance, any newly revealed vertex is either accepted by $\texttt{TaS}_t$ or is part of $\texttt{OPT}_{i+1}$. Therefore, we can show that the number of vertices revealed in phase $i+1$ that are part of $\texttt{OPT}_{i+1}$ is bounded above by $\texttt{OPT}_{i+1} - \texttt{OPT}_{i-1}$, which implies that it is sufficient for the budget to satisfy $B \geq \frac{\texttt{ALG}+\texttt{OPT}_{i+1}}{\texttt{OPT}_{i+1}-\texttt{OPT}_{i-1}}$. Furthermore, we incorporate both the number of vertices in phase $i+1$ that are accepted by $\texttt{TaS}_t$ and the number of vertices revealed in phase i that are accepted by $\texttt{TaS}_t$ into our analysis and show that the lower bound on the required budget is largest when there are no such vertices.

We conclude that it is sufficient for each newly-revealed vertex to carry budget $B = \frac{t}{t-1}$. Thus, $\texttt{TaS}_t$ is t-competitive while incurring at most $\frac{t}{t-1}$ amortized recourse. □

Theorem 3. *For any $1 < t \leq 2$, $\varepsilon > 0$, and t-competitive deterministic online algorithm, there exists an instance for which the algorithm incurs at least $\frac{1}{t-1} - \varepsilon$ amortized recourse.*

Proof **(Ideas).** Consider any t-competitive online algorithm against an adversary that constructs a complete bipartite graph and only reveals new vertices in the partition which does not contain the algorithm's current solution. This means that the maximum number of vertices that the algorithm's solution can contain will only increase if the algorithm moves its solution from one partition to the other. Furthermore, in doing so, the algorithm is forced to late-reject all vertices in its old solution in order to late-accept all vertices in its new solution.

By the structure of this adversary, we can show that 1) each partition-changing switch will incur at least $\frac{1}{t}$ recourse amortized over the size of the revealed graph when the switch occurs, and 2) there are at most t times more vertices at switch $i+1$ than at switch i. Therefore, the recourse incurred by switches up to switch i amortized over the size of the revealed graph $f(i)$ satisfied the recurrence relation $f(i) = \frac{1}{t} + \frac{1}{t} \cdot f(i-1)$, where $f(1) = 1$. Solving this recurrence relation, we conclude that for any $1 < t \leq 2$, $\varepsilon > 0$, and t-competitive deterministic online algorithm, there exists an instance for which the algorithm incurs at least $\frac{1}{t-1} - \varepsilon$ amortized recourse. □

4 Maximum Cardinality Matching

The $\texttt{TaS}_t$ algorithm greedily aligns with the yardstick solution completely and incurs a lot of recourse. However, for some of the elements whose value is changed, the alignment may not contribute to the improvement of the competitive ratio as much as the alignment of other elements. This observation suggests that it may be possible to reduce the amount of amortized recourse while maintaining t-competitiveness by switching the solution only *partially* into the yardstick. In this section, we show that the L-`Greedy` algorithm by Angelopoulos et al. [2], which is in fact a $\texttt{TaS}_t$ algorithm that uses an optimal solution as the yardstick without aligning to it fully, incurs less amortized recourse for the Maximum Cardinality Matching problem.

L-`Greedy` **Algorithm** [2]**.** The algorithm is associated with a parameter L. Throughout the process, the L-`Greedy` algorithm partially switches its solution to the optimal once by eliminating all augmenting paths with length at most $2L+1$. That is, it late rejects all the edges selected by itself and late accepts all the edges in the optimal solution on the path.

After applying late operations on all augmenting paths with at most $2L+1$ edges, every remaining augmenting path has length at least $2L+3$, and the ratio of the `OPT` solution value to the L-`Greedy` solution value is $\frac{\texttt{OPT}(P)}{L\text{-}\texttt{Greedy}(P)} \leq \frac{L+2}{L+1}$ on the component P. Since the Maximum Cardinality Matching problem can be solved in $O(n^{2.5})$ time, the following theorem holds by selecting $L = \lceil \frac{1}{t-1} \rceil - 1$.

Theorem 4. *The L-`Greedy` algorithm returns a valid matching with competitive ratio $\frac{L+2}{L+1}$ in $O(n^{3.5})$ time, where n is the number of vertices in the final graph.*

Proof **(Ideas).** After applying the late operations on all the augmenting path with at most $2L+1$ edges, every remaining augmenting path P has length at least $2L+3 = (L+2)+(L+1)$, and the ratio of the OPT size to the L-Greedy size $\frac{\texttt{OPT}(P)}{L\text{-}\texttt{Greedy}(P)} \leq \frac{L+2}{L+1}$ on the component P. By Observation 1, the ratio of an optimal solution to that of L-Greedy is at most $\frac{L+2}{L+1}$.

When an edge/vertex arrives, the algorithm checks if there is an augmenting path. By the well-known Hopcroft-Karp algorithm [17], it can be done in $O(n \cdot n^{2.5}) = O(n^{3.5})$-time, where n is the number of vertices in the final graph. □

Since it was shown that to achieve 1.5-competitiveness, every vertex incurs at most 2 recourse, we consider a target competitive ratio $1 < t < 2$ and have the following theorem. Note that $1 < t^* < 2$, thus $0 < \frac{t^*-1}{3-t^*} < 1$.

Theorem 5. *For the* Maximum Cardinality Matching *problem in the vertex/edge-arrival model, the L-Greedy algorithm is t-competitive for any $1 < t < 2$ and incurs at most $\frac{(2-t^*)}{(t^*-1)(3-t^*)} + \frac{t^*-1}{3-t^*}$ amortized recourse, where t^* is the largest number such that $t^* \leq t$ and $t^* = 1 + \frac{1}{j}$ for some integer j.*

Proof **(Ideas).** Consider the connected components generated by the union of edges chosen by L-Greedy or by OPT. By Observation 1, we prove this theorem by showing that for any component in the graph, the total recourse incurred at this component divided by its size is at most $\frac{(2-t^*)}{(t^*-1)(3-t^*)} + \frac{t^*-1}{3-t^*}$.

By selecting $L = \lceil \frac{2-t}{t-1} \rceil$, the path eliminations only happen at odd-size components with length from 3 to $2 \cdot (\lceil \frac{1}{t-1} \rceil - 1) + 1$ (note that $\lceil \frac{1}{t-1} \rceil \geq 2$ since $1 < t < 2$). Moreover, for such a $(2k+1)$-edge augmenting path, the total recourse incurred by the $2k+1$ elements in the path is at most $1 + \sum_{k=1}^{\lceil \frac{1}{t-1} \rceil - 1} 2k$. Hence, the amount of amortized recourse incurred by this component is at most $\frac{(\lceil \frac{1}{t-1} \rceil - 1) \cdot \lceil \frac{1}{t-1} \rceil + 1}{2 \lceil \frac{1}{t-1} \rceil - 1}$. Thus, the theorem is proven if $t = 1 + \frac{1}{j}$ for some integer j.

For the case in which there is no integer j such that $t = 1 + \frac{1}{j}$, the proof can be adapted by rounding down t to the largest $t^* \leq t$ such that $t^* = 1 + \frac{1}{j}$ for some integer j. By eliminating all augmenting paths that have length at most $\frac{2}{t^*-1} - 1$, the amount of incurred amortized recourse is at most $\frac{2-t^*}{(t^*-1)(3-t^*)} + \frac{t^*-1}{3-t^*}$, and the algorithm attains a competitive ratio of $t^* \leq t$. □

Theorem 6. *No deterministic t-competitive online algorithm can incur amortized recourse less than $\frac{(2-t^*)}{(t^*-1)(3-t^*)}$ in the worst case.*

Proof **(Ideas).** Given that $\frac{n+2}{n+1} \leq t < \frac{n+1}{n}$ for some integer $n \geq 1$, consider the adversarial instance that releases a sequence of $2n+1$ edges that form a path. More specifically, given the current instance which is an ℓ-path, the adversary releases a new edge that is incident to one of the endpoints of the ℓ-path and forms a $(\ell+1)$-path. For any $1 \leq k \leq n$, the following invariants hold for any t-competitive algorithm:

(**I1**) For a path with length $2k+1$, a t-competitive algorithm has to accept $k+1$ edges.
(**I2**) For a path with length $2k$, a t-competitive algorithm has to accept k edges.
(**I3**) When an instance is increased from a $2(k-1)+1$ path to a $2k+1$ path, a t-competitive algorithm incurs at least $2k$ amount of recourse.

Given the invariants **I1**, **I2**, and **I3**, the $(2n+1)$-length path instance incurs recourse with total amount at least $\sum_{k=1}^{n}(2k) = n \cdot (n+1)$. Therefore, any t-competitive algorithm incurs at least $\frac{n\cdot(n+1)}{2n+1}$ amortized recourse for this instance. Let $t^* = \frac{n+2}{n+1} \leq t$. It follows that $n = \frac{2-t^*}{t^*-1}$. Therefore, the amortized recourse is at least $\frac{n\cdot(n+1)}{2n+1} = \frac{(2-t^*)}{(t^*-1)(3-t^*)}$. □

5 Minimum Vertex Cover

In this section, we propose a special version of the $\mathtt{TaS}_t$ algorithm, `Duo-Halve`, that attains a competitive ratio of $2 - \frac{2}{\mathtt{OPT}}$ for the Minimum Vertex Cover problem with optimal vertex cover size `OPT` in polynomial time.

The `Duo-Halve` algorithm uses an optimal solution as the yardstick with $t = 2 - \frac{2}{\mathtt{OPT}}$. However, the computation of the optimal solution of Vertex Cover is very expensive. Thus, we maintain a maximal matching greedily (as the well-known 2-approximation algorithm for Vertex Cover) on the current input graph and only select vertices that are saturated by the matching. Intuitively, if the `Duo-Halve` algorithm rejects two of these saturated vertices, the competitive ratio is at most $2 - \frac{2}{\mathtt{OPT}}$. We show that either we can remove up to 2 carefully chosen vertices from the matching-based solution without violating its feasibility, or the optimal solution contains at least one vertex more than the number of edges in the maximal matching (Lemma 3 and Theorem 7). In either case, the constructed solution is $(2 - \frac{2}{\mathtt{OPT}})$-competitive. We can refine the choice of vertices to be removed so that they are incident to one of the two last edges added to the matching. This restriction allows us to show that maintaining this constructed solution needs only a constant amount of amortized recourse, and polynomial time (Lemma 2).

In the following discussion, we use some terminology. Let ME1 and ME2 be the last and second-to-last edges added to the matching respectively. Note that ME1 and ME2 change over the course of the input sequence as more vertices are revealed and edges are added to the matching. Also, let $V_{M(\mathcal{X})}$ be the vertices saturated by the maximal matching $M(\mathcal{X})$. The `DH` algorithm partitions the vertices into three groups: **Group-1**: the endpoints of ME1 or ME2, **Group-2**: the vertices in V_M but not in Group-1, and **Group-3**: the vertices in $V \backslash V_M$.

`Duo-Halve` **Algorithm (**`DH`**).** When a new vertex v arrives, if an edge (p, v) is added to $M(\mathcal{X})$, then it introduces a new ME1 (namely (p, v)). The algorithm first accepts all **Group-2** vertices that are adjacent to v. Then, the algorithm decides the assignment of ME1 and ME2 and minimizes the number of accepted endpoints of ME1 and ME2. If there is a tie, we apply the one that accepts fewer endpoints in ME1 and/or incurs less recourse. (See Algorithm 2).

Algorithm 2. `Duo-Halve` algorithm (`DH`) for Minimum Vertex Cover Problem

```
ME1 $\leftarrow \varnothing$, ME2 $\leftarrow \varnothing$, $V_M \leftarrow \varnothing$
**while** new vertex $v$ arrives **do**
    **if** there is a vertex $p \in N(v) \cup (V \backslash V_M)$ **then**
        ME2 $\leftarrow$ ME1
        ME1 $\leftarrow (p, v)$      $\triangleright$ $(p, v)$ is a new matched edge. If there is more than one $p$, choose one arbitrarily.
        Add $p$ and $v$ into $V_M$
        LateAccept all rejected vertices in $(V_M \backslash \{\text{vertices in ME1 or ME2}\}) \cap N(v)$
        HalveBoth(ME1, ME2)
    **else**
        LateAccept all rejected vertices in $V_M \cap N(v)$
        HalveBoth(ME1, ME2)
    **end if**
**end while**

**Function** HalveBoth(matched edge ME1, matched edge ME2)
Among accept/reject configurations of ME1 and ME2 that yield a valid vertex cover, return one that maximizes the number of half edges among ME1 and ME2 with the minimum number of late operations. If there is a tie, prioritize ME1 (see Fig. 1 for details).
**end Function**
```

The `DH` algorithm returns a feasible solution in $O(n^3)$ time, where n is the number of vertices in the graph. Intuitively, the algorithm maintains a valid solution as it greedily covers edges using vertices in the maximal matching, with the exception of ME1 and ME2, where it carefully ensures that a feasible configuration is chosen. Furthermore, the most computationally-expensive component of the `DH` algorithm, which checks the validity of a constant number of configurations by looking at the neighborhoods of ME1 and ME2, runs in $O(n^2)$ time for each new element.

Lemma 2. *The `DH` algorithm always returns a valid vertex cover in $O(n^3)$ time, where n is the number of vertices in the graph.*

We first show that if `DH` fails to produce a solution where it accepts only one vertex of ME1, then $\texttt{OPT} \geq |M| + 1$. The intuition is that if `DH` has to accept both endpoints of ME1, there must be at least one **Group-3** vertex in each of the endpoints' neighborhoods. Therefore, the optimal solution has to cover the corresponding edges with at least two vertices.

Lemma 3. *In the assignment of `DH`, if both endpoints of ME1 are selected, then the optimal solution must contain at least two vertices in ME1 $\cup (V \backslash V_M)$, and $\frac{\texttt{DH}}{\texttt{OPT}} \leq 2 - \frac{2}{\texttt{OPT}}$.*

Theorem 7. *The `DH` algorithm is $(2 - \frac{2}{\texttt{OPT}})$-competitive.*

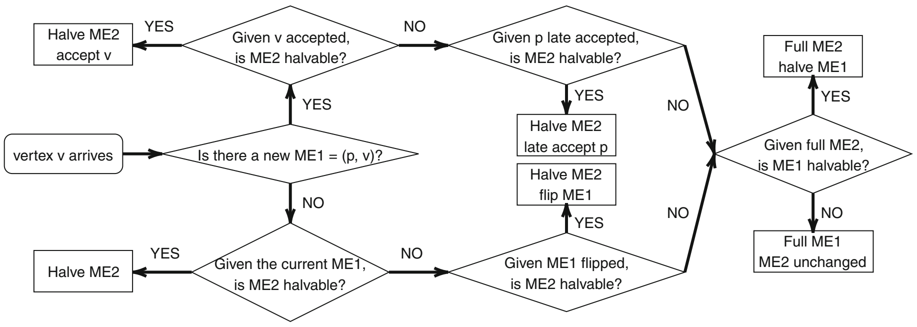
vertex v arrives
Is there a new ME1 = (p, v)?
YES
Given v accepted, is ME2 halvable?
YES
Halve ME2 accept v
NO
Given p late accepted, is ME2 halvable?
YES
Halve ME2 late accept p
NO
NO
Given the current ME1, is ME2 halvable?
YES
Halve ME2
NO
Given ME1 flipped, is ME2 halvable?
YES
Halve ME2 flip ME1
NO
Given full ME2, is ME1 halvable?
YES
Full ME2 halve ME1
NO
Full ME1 ME2 unchanged

Fig. 1. An illustration of the flow of `HalveBoth` (ME1, ME2).

Proof **(Ideas).** We define an edge as being *half* if exactly one of its endpoints is accepted by DH, and *full* if both of its endpoints are accepted by DH. In any possible solution provided by DH, there are three states based on the configuration of ME1 and ME2: 1) both ME1 and ME2 are half, 2) ME1 is half and ME2 is full, and 3) ME1 is full. In state 1, we can directly show that the bound holds since DH $\leq 2|M| - 2$. The bound holds for state 3 by Lemma 3.

State 2 requires more involved analysis. If an endpoint of ME2 has a rejected **Group-2** neighbor, then DH rejects at least two vertices in V_M (this **Group-2** neighbor and 1 ME1 vertex) and DH $\leq 2|M| - 2$. Otherwise, if at least one endpoint of ME2 has no **Group-3** neighbor, then we can show that there is no solution based on the maximal matching containing only 2 **Group-1** vertices. This means that OPT must contain either a **Group-3** vertex or 3 **Group-1** vertices, and thus OPT $\geq |M|+1$. Finally, if each endpoint of ME2 has a **Group-3** neighbor, then OPT must either select a **Group-3** vertex or both endpoints of ME2, and OPT $\geq |M| + 1$. □

For a single newly-revealed vertex, the amount of recourse incurred can be up to $O(n)$. Even if we restrict our consideration to ME1 and ME2, a single new vertex can incur recourse at most 4. However, this cannot happen at every input. We use a potential function to show that the amortized recourse incurred by DH is at most 3.33.

Theorem 8. *The amortized recourse incurred by DH is at most* $\frac{10}{3}$.

Proof **(Ideas).** We prove the theorem by using a potential function. To this end, we define an edge (u, v) as being *free* if there exist feasible assignments both by either accepting u or by accepting v. Also, we define a matched edge with only one endpoint selected as being *expired* if it is neither ME1 nor ME2. Finally, we define A as the set of vertices accepted by DH. Using these terms, we define the potential function Φ as

$$\Phi := |\{(u, v)|(u, v) \text{ expired}\}| + \frac{1}{3}|A \cap (\text{ME1} \cup \text{ME2})| + \frac{2}{3} \cdot \mathbb{1}[\text{ME2 is free}]$$

Furthermore, at any given moment in the input sequence where the matching constructed by DH contains at least 2 edges, the status of ME1 and ME2 is characterized by one of 6 states according to their possible combinations of selection statuses of their endpoints. We also differentiate between the two half possibilities for ME1, since the newly-revealed vertex in ME1 can be accepted without incurring a late operation when there is a new ME1.

We show that, for any possible state transition triggered by a newly-revealed vertex, the number of incurred late operations LO added to the change in potential $\Delta\Phi$ is bounded above by $\frac{10}{3}$. Note that, for any newly-revealed vertex v, v may be adjacent to $k \geq 0$ rejected vertices that are matched by some expired edge. This incurs k late operations, but also decreases Φ by k, so this may be ignored when computing LO $+ \Delta\Phi$. Since $\Phi_0 = 0$ and $\Phi_i \geq 0$, this allows us to conclude the statement of our theorem. □

Moreover, we can show a lower bound by constructing a family of instances that alternates between incurring a late accept on a **Group-2** vertex, and 4 late operations on ME1 and ME2. This is illustrated in Fig. 2.

Lemma 4. *For any $\varepsilon > 0$, there exists an instance such that* DH *incurs amortized recourse strictly greater than $\frac{5}{2} - \varepsilon$.*

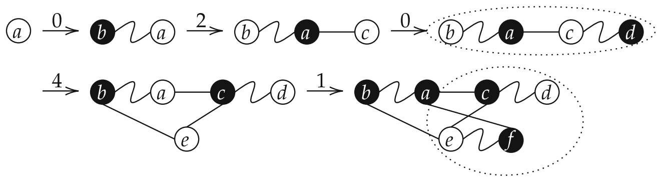

Fig. 2. Adversarial instance for Vertex Cover such that DH incurs asymptotic amortized recourse $\frac{5}{2}$. Each arrow's number denotes the number of late operations incurred by the next vertex's reveal. The dotted ovals highlight the repeating structure.

Finally, we show that the analysis in Theorem 7 is tight for a class of online algorithms where its solution only contains vertices saturated by the matching maintained throughout the process in an incremental manner. In other words, no online algorithm in this class achieves a lower competitive ratio, no matter how much amortized recourse it uses.

Definition 4. *An algorithm for vertex cover is* incremental matching-based *if it maintains a maximal matching throughout the process in an incremental manner, and its solution only contains vertices saturated by the matching.*

Theorem 9. *No deterministic incremental matching-based algorithm achieves a competitive ratio smaller than $2 - \frac{2}{\mathtt{OPT}}$.*

Proof. Consider the instance which first reveals k disconnected edges via their endpoints. For any incremental matching-based algorithm, each of these k edges will be added to the matching, and at least one vertex from each pair will be accepted. Then, the instance reveals a final vertex that is adjacent to all previously-revealed vertices (See Fig. 3). The incremental matching-based algorithm will not accept this vertex, as it is not matched, but must accept all other vertices for a vertex cover of size $2k$. However, the optimal solution consists of the last revealed vertex, and one endpoint from each of the k edges. Therefore, no incremental matching-based algorithm can achieve a competitive ratio smaller than $\frac{2k}{k+1} = 2 - \frac{2}{\texttt{OPT}}$. □

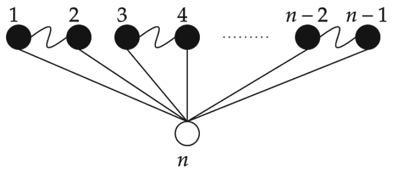

Fig. 3. Adversarial instance for Vertex Cover such that any incremental matching-based algorithm is exactly $(2 - \frac{2}{\texttt{OPT}})$-competitive. Vertices are labeled by their release order. Any such algorithm must accept $n - 1$ vertices, whereas the optimal solution contains $\frac{n-1}{2} + 1$ vertices.

6 Concluding Remarks

In this paper, we propose a general Target-and-Switch algorithm for online problems that allow recourse. We prove that for any monotone-sum graph problem, the algorithm attains a competitive ratio of $t > 1$ while incurring $w_{\max} \cdot (1 + \frac{1}{t-1})$ amortized recourse. Many interesting problems remain open. A major future direction is extending the analysis to non-monotone problems such as Dominating Sets or monotone-max problems such as Coloring.

Acknowledgement. We wish to thank the anonymous referees for their comments and suggestions on a previous version of this paper. In particular, we thank them for helping us complete the monotone problem definition.

References

1. Albers, S., Schraink, S.: Tight bounds for online coloring of basic graph classes. In: Pruhs, K., Sohler, C. (eds.) 25th Annual European Symposium on Algorithms, ESA 2017, 4–6 September 2017, Vienna, Austria. LIPIcs, vol. 87, pp. 7:1–7:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017). https://doi.org/10.4230/LIPIcs.ESA.2017.7
2. Angelopoulos, S., Dürr, C., Jin, S.: Online maximum matching with recourse. J. Comb. Optim. **40**(4), 974–1007 (2020). https://doi.org/10.1007/s10878-020-00641-w

3. Avitabile, T., Mathieu, C., Parkinson, L.H.: Online constrained optimization with recourse. Inf. Process. Lett. **113**(3), 81–86 (2013). https://doi.org/10.1016/j.ipl.2012.09.011
4. Azar, Y., Panigrahi, D., Touitou, N.: Online graph algorithms with predictions. In: Naor, J.S., Buchbinder, N. (eds.) Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference, Alexandria, VA, USA, 9–12 January 2022, pp. 35–66. SIAM (2022). https://doi.org/10.1137/1.9781611977073.3
5. Azar, Y., Touitou, N.: Beyond tree embeddings - a deterministic framework for network design with deadlines or delay. In: Irani [18], pp. 1368–1379 (2020). https://doi.org/10.1109/FOCS46700.2020.00129
6. Bernstein, A., Holm, J., Rotenberg, E.: Online bipartite matching with amortized $O(\log^2 n)$ replacements. J. ACM **66**(5), 37:1–37:23 (2019). https://doi.org/10.1145/3344999
7. Bienkowski, M., Kraska, A., Liu, H.-H., Schmidt, P.: A primal-dual online deterministic algorithm for matching with delays. In: Epstein, L., Erlebach, T. (eds.) WAOA 2018. LNCS, vol. 11312, pp. 51–68. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04693-4_4
8. Boppana, R.B., Halldórsson, M.M.: Approximating maximum independent sets by excluding subgraphs. BIT **32**(2), 180–196 (1992). https://doi.org/10.1007/BF01994876
9. Bosek, B., Disser, Y., Feldmann, A.E., Pawlewicz, J., Zych-Pawlewicz, A.: Recoloring interval graphs with limited recourse budget. In: Albers, S. (ed.) 17th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2020, 22–24 June 2020, Tórshavn, Faroe Islands. LIPIcs, vol. 162, pp. 17:1–17:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.SWAT.2020.17
10. Boyar, J., Favrholdt, L.M., Kotrbcík, M., Larsen, K.S.: Relaxing the irrevocability requirement for online graph algorithms. Algorithmica **84**, 1916–1951 (2022). https://doi.org/10.1007/s00453-022-00944-w
11. Cygan, M., Czumaj, A., Mucha, M., Sankowski, P.: Online facility location with deletions. In: Azar, Y., Bast, H., Herman, G. (eds.) 26th Annual European Symposium on Algorithms, ESA 2018, 20–22 August 2018, Helsinki, Finland. LIPIcs, vol. 112, pp. 21:1–21:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). https://doi.org/10.4230/LIPIcs.ESA.2018.21
12. Gu, A., Gupta, A., Kumar, A.: The power of deferral: Maintaining a constant-competitive steiner tree online. SIAM J. Comput. **45**(1), 1–28 (2016). https://doi.org/10.1137/140955276
13. Gupta, A., Guruganesh, G., Kumar, A., Wajc, D.: Fully-dynamic bin packing with limited repacking. CoRR abs/1711.02078 (2017). http://arxiv.org/abs/1711.02078
14. Gupta, A., Levin, R.: Fully-dynamic submodular cover with bounded recourse. In: Irani [18], pp. 1147–1157 (2020). https://doi.org/10.1109/FOCS46700.2020.00110
15. Halperin, E.: Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. SIAM J. Comput. **31**(5), 1608–1623 (2002). https://doi.org/10.1137/S0097539700381097
16. Harutyunyan, H.A., Pankratov, D., Racicot, J.: Online domination: the value of getting to know all your neighbors. In: Bonchi, F., Puglisi, S.J. (eds.) 46th International Symposium on Mathematical Foundations of Computer Science, MFCS 2021, 23–27 August 2021, Tallinn, Estonia. LIPIcs, vol. 202, pp. 57:1–57:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.MFCS.2021.57

17. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. **2**(4), 225–231 (1973). https://doi.org/10.1137/0202019
18. Irani, S. (ed.): 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, 16–19 November 2020. IEEE (2020). https://doi.org/10.1109/FOCS46700.2020
19. Karakostas, G.: A better approximation ratio for the vertex cover problem. ACM Trans. Algorithms **5**(4), 41:1–41:8 (2009). https://doi.org/10.1145/1597036.1597045
20. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within 2-epsilon. J. Comput. Syst. Sci. **74**(3), 335–349 (2008). https://doi.org/10.1016/j.jcss.2007.06.019
21. Megow, N., Nölke, L.: Online minimum cost matching with recourse on the line. In: Byrka, J., Meka, R. (eds.) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, 17–19 August 2020, Virtual Conference. LIPIcs, vol. 176, pp. 37:1–37:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2020.37
22. Megow, N., Skutella, M., Verschae, J., Wiese, A.: The power of recourse for online MST and TSP. SIAM J. Comput. **45**(3), 859–880 (2016). https://doi.org/10.1137/130917703
23. Monien, B., Speckenmeyer, E.: Ramsey numbers and an approximation algorithm for the vertex cover problem. Acta Inform. **22**(1), 115–123 (1985). https://doi.org/10.1007/BF00290149
24. Wang, Y., Wong, S.C.: Two-sided online bipartite matching and vertex cover: beating the greedy algorithm. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015, Part I. LNCS, vol. 9134, pp. 1070–1081. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47672-7_87