Learning-Augmented Online Bipartite Fractional Matching

Davin Choo[*] [§]
Harvard John A. Paulson School Of Engineering And Applied Sciences
Harvard University
Boston, Massachusetts, USA
davinchoo@seas.harvard.edu

Billy Jin[†] [§]
Daniels School of Business
Purdue University
West Lafayette, Indiana, USA
jin608@purdue.edu

Yongho Shin[‡] [§]
Institute of Computer Science
University of Wrocław
Wrocław, Poland
yongho@cs.uni.wroc.pl

Abstract

Online bipartite matching is a fundamental problem in online optimization, extensively studied both in its integral and fractional forms due to its theoretical significance and practical applications, such as online advertising and resource allocation. Motivated by recent progress in learning-augmented algorithms, we study online bipartite fractional matching when the algorithm is given advice in the form of a suggested matching in each iteration. We develop algorithms for both the vertex-weighted and unweighted variants that provably dominate the naïve "coin flip" strategy of randomly choosing between the advice-following and advice-free algorithms. Moreover, our algorithm for the vertex-weighted setting extends to the AdWords problem under the small bids assumption, yielding a significant improvement over the seminal work of Mahdian, Nazerzadeh, and Saberi (EC 2007, TALG 2012). Complementing our positive results, we establish a hardness bound on the robustness-consistency tradeoff that is attainable by any algorithm. We empirically validate our algorithms through experiments on synthetic and real-world data.

1 Introduction

Online bipartite matching is a fundamental problem in online optimization with significant applications in areas such as online advertising [MSVV07, FKM+09], resource allocation [DJSW19], and ride-sharing platforms [DSSX21, FNS24]. In its classical formulation [KVV90, AGKM11], the input is a bipartite graph where one side of (possibly weighted) *offline* vertices is known in advance, while the other side of *online* vertices arrives sequentially one at a time. When an online vertex v arrives, its incident edges are revealed, and the algorithm irrevocably decides whether to match v and, if so, to which currently unmatched neighbor. The objective is to maximize the total weight of the matched offline vertices. Algorithms for online bipartite matching are often evaluated by their *competitive ratio*: An algorithm is *ρ-competitive* if it always outputs a matching whose (expected) total weight is

[*]Part of work done while the author was affiliated with the National University of Singapore, Singapore.
[†]Work done while the author was at the University of Chicago Booth School of Business, USA.
[‡]Part of work done while the author was affiliated with Yonsei University, South Korea.
[§]Equal contribution.

at least ρ times the weight of the best matching in hindsight. In a seminal paper, [KVV90] proposed the RANKING algorithm and showed it is $(1 - 1/e)$-competitive for the unweighted setting. This competitive ratio is best-possible, and was later extended to the vertex-weighted case by [AGKM11].

Online bipartite matching has also been studied in the fractional setting, where edges can be fractionally chosen, provided that the total fractional value on the edges incident to any vertex does not exceed one [WW15, HPT+19, HTWZ20, HHIS24]. Fractional matching is important both theoretically and practically. It naturally models settings where online arrivals are divisible or offline vertices have large capacities [KP00, MSVV07, BJN07, FKM+09, MNS12, DHK+16, FN24], and it forms the basis for designing integral algorithms using rounding techniques [FSZ16, BNW23, NSW25]. For fractional vertex-weighted online bipartite matching, the BALANCE algorithm of [BJN07] gets a competitive ratio of $(1 - 1/e)$, which is best-possible and matches the ratio in the integral case.

The main challenge in online bipartite matching is that irrevocable decisions must be made without knowledge of future arrivals. Uncertainty in the arrival sequence is typically modeled either *adversarially* or *stochastically*. The adversarial model assumes no structure and measures worst-case performance, but can be overly pessimistic. On the other hand, the stochastic model assumes arrivals are drawn from a known distribution [FMMM09], but such distributions are often estimated and may be inaccurate. These models thus represent two extremes, each with practical limitations. A middle ground is offered by *algorithms with predictions*, or *learning-augmented algorithms* [MNS12, LV21], which incorporates advice – derived from data, forecasts, or experts – of unknown quality. The performance is typically measured in terms of its *robustness* (guaranteed performance regardless of advice quality) and *consistency* (performance when advice is accurate) [LV21, KPS18].[5] In online bipartite matching, an algorithm is r-robust if its competitive ratio is at least r, and c-consistent if it achieves at least a c-fraction of the total weight from following the advice (see Definition 4). A natural baseline is the COINFLIP algorithm, which randomly chooses between robustness- and consistency-optimal strategies. For matching, its tradeoff curve is the line segment between $(1 - 1/e, 1 - 1/e)$ and $(0, 1)$ in the vertex-weighted case, or $(1/2, 1)$ in the unweighted case [JM22].

This paper investigates the robustness-consistency tradeoff of online bipartite matching under the learning-augmented framework, building on prior work including [MNS07, MNS12, ACI22, JM22, SE23, CGLB24]. Particularly relevant are the works of Mahdian et al. [MNS07, MNS12] and Spaeh and Ene [SE23]. Mahdian et al. studied the AdWords problem (introduced in [MSVV07]), with advice in the form of a recommendation assigning each online impression to a specific offline advertiser. They proposed a learning-augmented algorithm under the *small bids* assumption that outperforms the naïve COINFLIP strategy, but only over part of the robustness range. Meanwhile, [SE23] generalized this result to Display Ads and the generalized assignment problem [FKM+09]. However, as shown in Fig. 1, neither of these algorithms dominate COINFLIP across the full robustness spectrum. This raises a natural question: *Does there exist a learning-augmented algorithm for online bipartite matching that dominates* COINFLIP *across the entire range of robustness?*

1.1 Our contributions

We answer the above question affirmatively by presenting learning-augmented algorithms for both vertex-weighted and unweighted online bipartite fractional matching whose robustness-consistency tradeoffs Pareto-dominate that of COINFLIP across the *entire* range of robustness (see Fig. 1).

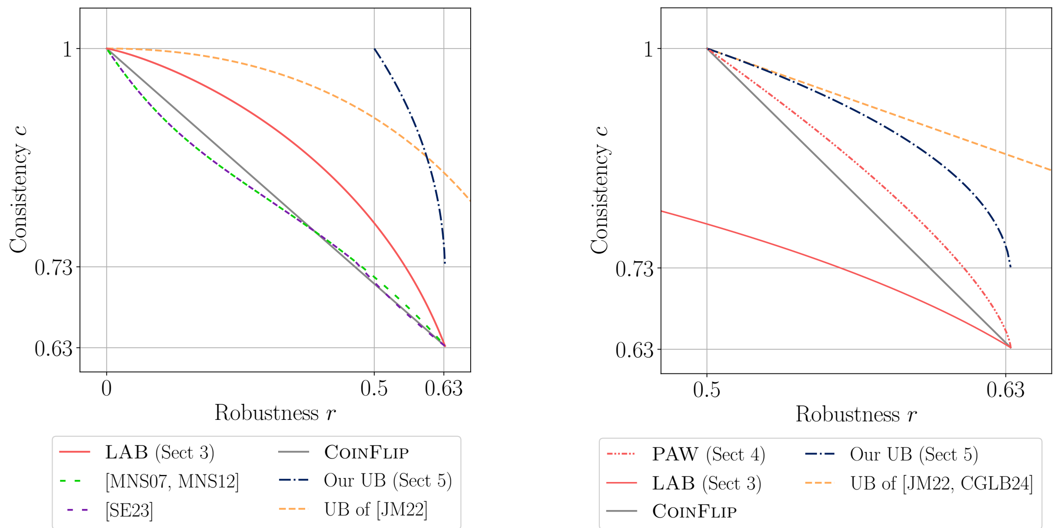
Motivated by [MNS07, MNS12, SE23], we take the advice to be a feasible fractional matching that is revealed in an online fashion: upon arrival of each online vertex v, the algorithm is given as advice fractional matching values for each neighboring edge of v. Moreover, as in [MNS07, MNS12, SE23], our algorithms are parameterized by a tradeoff parameter $\lambda \in [0, 1]$ that represents how closely we follow the advice. At the extremes, our algorithms blindly follow the advice when $\lambda = 1$ and revert to BALANCE when $\lambda = 0$.

For the vertex-weighted setting, we present an algorithm LEARNINGAUGMENTEDBALANCE (LAB) with the following guarantees:

Theorem 1. *For any tradeoff parameter $\lambda \in [0, 1]$,* LEARNINGAUGMENTEDBALANCE *is an $r(\lambda)$-robust and $c(\lambda)$-consistent algorithm for vertex-weighted online bipartite fractional matching, where*

$$r(\lambda) := 1 - e^{\lambda-1} - \left(e^{\lambda-1} - \lambda\right) \ln(1 - \lambda e^{1-\lambda}) - \lambda(1 - \lambda) \quad \textit{and} \quad c(\lambda) := 1 + \lambda - e^{\lambda-1}.$$

[5] A third property, *smoothness*, requires graceful degradation with advice quality [EADL24]. See Section 2.



(a) Vertex-weighted setting and AdWords under small bids assumption

(b) Unweighted setting

Figure 1: Robustness-consistency tradeoffs of previous works and our results.

This algorithm is based on BALANCE where the penalty function is modified to be *advice-dependent*. To analyze this algorithm, we adopt the standard primal-dual analysis of online bipartite matching and first prove its performance when the advice is integral. We then prove that the robustness and consistency are minimized when the advice is integral, yielding the same guarantees for the general fractional advice case.

We further show that LAB extends to the AdWords problem under the small bids assumption, yielding a significant improvement over [MNS07, MNS12].

Theorem 2. *Consider the small bids assumption where the maximum bid-to-budget ratio is bounded by some sufficiently small* $\varepsilon > 0$*. For any tradeoff parameter* $\lambda \in [0,1]$*, there exists an* $r(\lambda) \cdot (1 - 3\sqrt{\varepsilon \ln(1/\varepsilon)})$*-robust and* $c(\lambda) \cdot (1 - 3\sqrt{\varepsilon \ln(1/\varepsilon)})$*-consistent algorithm for AdWords with advice, where* $r(\lambda)$ *and* $c(\lambda)$ *are the same as in Theorem 1.*

To achieve this result, we first extend LAB to the *fractional* AdWords setting while preserving its robustness and consistency, and then employ a reduction from [FN24] to reduce the integral AdWords problem to the fractional problem with small loss under the small bids assumption.

Observe in Fig. 1 that the robustness-consistency tradeoff of LAB lies below the linear tradeoff of COINFLIP in the *unweighted* setting: the top-left endpoint of the tradeoff for LAB is $(r, c) = (0, 1)$, whereas in the unweighted setting COINFLIP can be implemented to be $1/2$-robust even when $c = 1$. This happens because any maximal matching in an unweighted graph is automatically $\frac{1}{2}$-robust. To beat COINFLIP in the unweighted setting, a tighter analysis of LAB would be required but this proved difficult using our current analysis framework for LAB, even when the advice is integral. Instead, we present another algorithm called PUSHANDWATERFILL (PAW) for the unweighted setting with integral advice that circumvents the aforementioned challenge in the analysis.

Theorem 3. *For any tradeoff parameter* $\lambda \in [0,1]$*,* PUSHANDWATERFILL *is* $r(\lambda)$*-robust and* $c(\lambda)$*-consistent for unweighted online bipartite fractional matching with integral advice, where*

$$r(\lambda) := 1 - \left(1 - \lambda + {\lambda^2}/{2}\right) e^{\lambda - 1} \;\; \text{and} \;\; c(\lambda) := 1 - (1 - \lambda)\, e^{\lambda - 1}.$$

PAW is based on the unweighted version of BALANCE, also known as WATERFILLING, with one additional step at each iteration where it first increases the fractional value of the currently advised edge until the "level" of the advised offline vertex reaches the tradeoff parameter λ. We analyze PAW using primal-dual but with a different construction of dual variables from LAB.

We complement our algorithmic results by presenting an upper bound on the robustness-consistency tradeoff of any learning-augmented algorithm for the unweighted setting with integral advice in

Section 5, improving upon the previous upper bound results [JM22, CGLB24] (see Fig. 1). Note that this result implies the same impossibility for more general problems including the vertex-weighted setting and the AdWords problem. To obtain our hardness result, we construct two adaptive adversaries — one for robustness and the other for consistency. The construction of these adversaries is inspired by the standard upper-triangular worst-case instances [KVV90], while we modify this construction to make the two adversaries have the same behavior until the first half of the online vertices are revealed. Due to this modification, the two adversaries are indistinguishable until the halfway point of the execution while inheriting the difficulty from the standard worst-case instances. We then identify a set of conditions characterizing the behavior of Pareto-optimal algorithms on our hardness instance and solve a factor-revealing LP to upper bound the best possible consistency subject to the constraint on the robustness to be r, for each $r \in [1/2, 1 - 1/e]$.

Lastly, we implemented and evaluated our proposed algorithms LAB and PAW in Section 6 against advice-free baselines on synthetic and real-world graph instances, for varying advice quality parameterized by a noise parameter γ, where larger γ indicates poorer advice quality. As predicted by our analysis, the attained competitive ratios of both LAB and PAW begin at 1 under perfect advice and smoothly degrades as the γ increases. Unsurprisingly, for sufficiently large γ, the worst case optimal advice-free algorithm BALANCE outperforms both LAB and PAW.

1.2 Related work

Learning-augmented algorithms for online matching. In addition to the works of [MNS07, MNS12] and [SE23], several other papers study learning-augmented algorithms for online bipartite matching. [AGKK20] studied the edge-weighted version under the random arrival model [KP09, KRTV13], where the advice estimates the edge-weight that each online vertex is assigned. [ACI22] considered a model where the advice predicts the degree of each offline vertex. They analyzed the performance of a greedy algorithm called MINPREDICTEDDEGREE that uses this advice, within the random graph model of [CLV03]. [JM22] considered the two-stage model of [FNS24] with the advice modeled as a feasible matching in the first stage. They developed optimal learning-augmented algorithms for the unweighted, vertex-weighted, edge-weighted, and AdWords variants. [CGLB24] showed that, in the unweighted setting under adversarial arrival, no algorithm that is 1-consistent can achieve robustness better than $1/2$. However, in the random arrival model, they designed a 1-consistent algorithm that achieves $(\beta - o(1))$-robustness using advice in the form of a histogram of arrival types, where β is the competitive ratio of the best advice-free algorithm in the same setting.

Learning-augmented algorithms more broadly. Since the seminal work of [LV21], there has been a surge of interest in incorporating unreliable advice into algorithm design and analyzing performance as a function of advice quality across various areas of computer science. This framework has been especially successful in online optimization, where the core challenge lies in handling uncertainty about future inputs. In this context, advice can serve as a useful proxy for the unknown future. Beyond online bipartite matching, a wide range of online optimization problems have been studied under the learning-augmented framework. Examples include caching and paging [LV21, JPS22, IKPP22, BCK$^+$22], ski rental [KPS18, WLW20, SLLA23, ZTCD24], covering problems [BMS20, GLS$^+$22], scheduling [LLMV20, ALT21, IKQP23, BP24], metric or graph problems [APT22, ACE$^+$23, SE24], causal graph learning [CGB23], and distribution learning [BCJG25a, BCJG25b]. For an overview of this growing area, we refer the reader to the survey by [MV22][6].

Online matching. The study of online matching began with the seminal work of [KVV90], who introduced the randomized RANKING algorithm and proved it to be $(1 - 1/e)$-competitive for online bipartite integral matching, the best possible in this setting. Due to its foundational importance, the analysis of RANKING has been revisited and extended in numerous subsequent works, including [GM08, BM08, DJK13, EFFS21]. The online matching problem has since been studied under a variety of extensions and settings, such as the edge-weighted case [FHTZ22, SA21, GHH$^+$22, BC22], ad allocation [MSVV07, BJN07, GM08, FKM$^+$09, HZZ24], random and stochastic arrivals [FMMM09, KP09, KMT11, KRTV13, JW21, HS21, HSY22], and two-sided or general arrival models [GKM$^+$19, HKT$^+$20, HTWZ20]. For a comprehensive overview of the field, we refer interested readers to the surveys by [Meh13] and [HTW24].

[6] See also `https://algorithms-with-predictions.github.io/`.

Paper outline. We begin with preliminaries such as definitions and notation used in the paper in Section 2. We then present our algorithmic results in the following two sections. In Section 3, we present the LAB algorithm for the vertex-weighted setting with fractional advice and then show that this algorithm extends to AdWords with small bids. Section 4 presents the PAW algorithm for the unweighted setting with integral advice. We then provide in Section 5 our upper bound result on the robustness-consistency tradeoff for the unweighted setting with integral advice. The experimental results are in Section 6, followed by concluding remarks in Section 7.

2 Preliminaries

Online bipartite matching. In the *vertex-weighted online bipartite fractional matching* problem, we have a bipartite graph $G = (U \cup V, E)$ and a weight $w_u \geq 0$ for each $u \in U$. If $w_u = 1$ for every $u \in U$, then the problem is called *unweighted*. The vertices in U are the *offline* vertices, and their weights are known to the algorithm from the very beginning. On the other hand, the vertices in V are the *online* vertices, and arrive one by one. Whenever $v \in V$ arrives, its neighborhood $N(v) := \{u \in U \mid (u, v) \in E\}$ is revealed. Since the online vertices arrive sequentially, we use the notation $t \prec v$ to mean that t arrives earlier than v. Similarly, for each offline vertex $u \in U$, we also use $N(u) := \{v \in V \mid (u, v) \in E\}$ to denote the neighborhood of u.

We use the analogy of *waterfilling* to describe the behavior of the algorithm. When $v \in V$ arrives and its neighborhood $N(v)$ is revealed, the algorithm decides at that moment the amount $x_{u,v}$ of water to send from v to each $u \in N(v)$ subject to the constraints that:

- the total amount of water supplied from v does not exceed 1, i.e., $\sum_{u \in N(v)} x_{u,v} \leq 1$;
- each offline vertex $u \in U$ can hold at most 1 unit of water, i.e., $\sum_{t \in N(u): t \preceq v} x_{u,t} \leq 1$.

This decision is irrevocable, meaning that, $\{x_{u,v}\}_{u \in N(v)}$ cannot be modified in the subsequent iterations. Let $x \in \mathbb{R}^E$ be the final solution of the algorithm. Note that x is a fractional matching in the hindsight graph G. The weight of this solution is defined to be $\sum_{(u,v) \in E} w_u x_{u,v}$. The objective of this problem is to maximize the weight of the solution.

Advice. Each online vertex $v \in V$ arrives with a *suggested allocation* $\{a_{u,v}\}_{u \in N(v)}$, where we assume $a = \{a_{u,v} : (u, v) \in E\} \in \mathbb{R}^E$ is a feasible fractional matching in the hindsight graph G.

AdWords. In the AdWords problem, the offline vertices U are called *advertisers*, and the online vertices V are called *impressions*. Each advertiser $u \in U$ starts with a budget $B_u \geq 0$. When an impression $v \in V$ arrives, each advertiser $u \in U$ submits a bid $b_{u,v} \geq 0$ for this impression. The algorithm then irrevocably assigns the impression to one of the advertisers.[7] If impression v is assigned to advertiser u, the algorithm earns a revenue of $b_{u,v}$, provided that u can afford the bid from its remaining budget. The objective is to maximize the total revenue earned by the algorithm.

We remark that vertex-weighted online bipartite matching is the special case of AdWords where $b_{u,v} = B_u$ if $(u, v) \in E$, and $b_{u,v} = 0$ if $(u, v) \notin E$.

Integral AdWords is commonly studied under the *small bids* assumption [MSVV07, BJN07, MNS12]. Under this assumption, the bid-to-budget ratio is assumed to be bounded by some small $\varepsilon > 0$. In other words, for all $u \in U$ and $v \in V$, we have $b_{u,v} \leq \varepsilon B_u$.

Performance measures. Denote the value of the final output of an algorithm by ALG, the value of an optimal solution in the hindsight instance by OPT, and the value obtained by the advice by ADVICE. We can then formally define the *robustness* and *consistency* of a learning-augmented algorithm.

Definition 4 (Robustness and Consistency)**.** For some $r \in [0, 1]$, we say an algorithm is *r-robust* if $\mathbb{E}[\mathsf{ALG}] \geq r \cdot \mathsf{OPT}$ for any instance of the problem. On the other hand, for some $c \in [0, 1]$, we say an algorithm is *c-consistent* if $\mathbb{E}[\mathsf{ALG}] \geq c \cdot \mathsf{ADVICE}$ for any instance of the problem.

Notice that, when we define the *error* of the advice to be $\eta := \mathsf{ADVICE}/\mathsf{OPT} \in [0, 1]$, the consistency implies the *smoothness* of the algorithm since we have $\mathbb{E}[\mathsf{ALG}] \geq c\eta \cdot \mathsf{OPT}$.

[7] By introducing an auxiliary advertiser who always bids 0 on every impression, we can assume without loss of generality that the algorithm assigns every impression.

Primal-dual analysis. To prove the robustness and consistency of our algorithms, we adopt the standard primal-dual analysis for online bipartite matching [DJK13]. Observe that, for vertex-weighted bipartite matching, the primal and dual LPs are formulated as follows:

$$\begin{aligned} \max \ & \textstyle\sum_{(u,v)\in E} w_u x_{u,v} & \\ \text{s.t.} \ & \textstyle\sum_{v\in N(u)} x_{u,v} \le 1, & \forall u \in U, \\ & \textstyle\sum_{u\in N(v)} x_{u,v} \le 1, & \forall v \in V, \\ & x_{u,v} \ge 0, & \forall (u,v) \in E; \end{aligned} \qquad \begin{aligned} \min \ & \textstyle\sum_{u\in U} \alpha_u + \sum_{v\in V} \beta_v & \\ \text{s.t.} \ & \alpha_u + \beta_v \ge w_u, & \forall (u,v) \in E, \\ & \alpha_u \ge 0, & \forall u \in U, \\ & \beta_v \ge 0, & \forall v \in V. \end{aligned}$$

The following lemma is the cornerstone of the primal-dual analysis.

Lemma 5 (see, e.g., [DJK13, FHTZ22])**.** *Let $x \in \mathbb{R}_+^E$ be a feasible fractional matching output by an algorithm. For some $\rho \in [0,1]$, if there exists $(\alpha, \beta) \in \mathbb{R}_+^U \times \mathbb{R}_+^V$ satisfying*

- *(reverse weak duality) $\sum_{(u,v)\in E} w_u x_{u,v} \ge \sum_{u\in U} \alpha_u + \sum_{v\in V} \beta_v$ and*
- *(approximate dual feasibility) $\alpha_u + \beta_v \ge \rho \cdot w_u$ for every $(u,v) \in E$,*

we have $\mathsf{ALG} \ge \rho \cdot \mathsf{OPT}$.

Proof. Observe that $(\alpha/\rho, \beta/\rho) \in \mathbb{R}^U \times \mathbb{R}^V$ is feasible to the dual LP. We thus have

$$\mathsf{ALG} = \sum_{(u,v)\in E} w_u x_{u,v} \ge \sum_{u\in U} \alpha_u + \sum_{v\in V} \beta_v = \rho \cdot \left[\sum_{u\in U} \alpha_u/\rho + \sum_{v\in V} \beta_v/\rho \right] \ge \rho \cdot \mathsf{OPT},$$

where the last inequality comes from the weak duality of LP. □

Similarly, the following are an LP relaxation of AdWords and its dual LP:

$$\begin{aligned} \max \ & \textstyle\sum_{u\in U} \sum_{v\in V} b_{u,v} x_{u,v} & \\ \text{s.t.} \ & \textstyle\frac{1}{B_u} \sum_{v\in V} b_{u,v} x_{u,v} \le 1, & \forall u \in U, \\ & \textstyle\sum_{u\in U} x_{u,v} \le 1, & \forall v \in V, \\ & x_{u,v} \ge 0, & \forall u \in U, v \in V; \end{aligned} \qquad \begin{aligned} \min \ & \textstyle\sum_{u\in U} \alpha_u + \sum_{v\in V} \beta_v & \\ \text{s.t.} \ & \textstyle\frac{b_{u,v}}{B_u} \alpha_u + \beta_v \ge b_{u,v}, & \forall u \in U, v \in V, \\ & \alpha_u \ge 0, & \forall u \in U, \\ & \beta_v \ge 0, & \forall v \in V. \end{aligned}$$

The below lemma is an adaptation of Lemma 5 to AdWords. The proof is omitted.

Lemma 6. *Let $x \in \mathbb{R}^{U\times V}$ be a feasible solution to the primal LP relaxation of AdWords. For some $\rho \in [0,1]$, if there exists $(\alpha, \beta) \in \mathbb{R}^U \times \mathbb{R}^V$ satisfying*

- *(reverse weak duality) $\sum_{u\in U} \sum_{v\in V} b_{u,v} x_{u,v} \ge \sum_{u\in U} \alpha_u + \sum_{v\in V} \beta_v$ and*
- *(approximate dual feasibility) $\frac{b_{u,v}}{B_u} \alpha_u + \beta_v \ge \rho \cdot b_{u,v}$ for every $u \in U$ and $v \in V$,*

we have $\sum_{u\in U} \sum_{v\in V} b_{u,v} x_{u,v} \ge \rho \cdot \mathsf{OPT}$.

Lambert W function. In the definition of LAB in Section 3, we use the (principal branch of) Lambert W function. Define $W : \left[-\frac{1}{e}, \infty\right) \to [-1, \infty)$ to be the inverse function of ye^y on $[-1, \infty)$, i.e., for any $z \in \left[-\frac{1}{e}, \infty\right)$, we have

$$W(z) \cdot e^{W(z)} = z. \tag{1}$$

It is known that W is increasing on $\left[-\frac{1}{e}, \infty\right)$. Its derivative can be written as follows:

$$W'(x) = \frac{W(x)}{x(1 + W(x))}. \tag{2}$$

Lastly, the next equality can easily be derived from Eq. (1):

$$\frac{W(z)}{z} = e^{-W(z)}. \tag{3}$$

3 Vertex-weighted matching with advice

We now present our algorithm LEARNINGAUGMENTEDBALANCE (LAB) for vertex-weighted online bipartite matching with advice and provide a proof sketch showing that it achieves the robustness-consistency tradeoff stated in Theorem 1. Detailed pseudocode is given in Appendix A and a full analysis is provided in the supplementary material.

Algorithm description. Given a tradeoff parameter $\lambda \in [0, 1]$, we define $f_0 : [0, 1] \to [0, 1]$ and $f_1 : [0, 1] \to [0, 1]$ as follows, where W is the Lambert W function:

$$f_0(z) := \min\{e^{z+\lambda-1}, 1\}, \quad \text{and} \quad f_1(z) := \begin{cases} \frac{e^{\lambda-1}-\lambda}{1-z}, & \text{if } z \in [0, \lambda e^{1-\lambda}), \\ \frac{-\lambda}{W(-\lambda e^{1-\lambda-z})}, & \text{if } z \in [\lambda e^{1-\lambda}, 1), \\ 1, & \text{if } z = 1, \end{cases} \tag{4}$$

Based on these functions, we define $f : [0, 1]^2 \to [0, 1]$ such that

$$f(A, X) := \begin{cases} f_1(X), & \text{if } A > X, \\ \max\{f_0(X - A), f_1(X)\}, & \text{if } A \leq X. \end{cases} \tag{5}$$

For clarity, let us describe LAB as a continuous process. Upon the arrival of each online vertex $v \in V$ along with the advice $\{a_{u,v}\}_{u \in N(v)}$, define $A_u := \sum_{t \in N(u): t \preceq v} a_{u,t}$ as the total advice-allocated amount to each offline vertex $u \in N(v)$, up to and including v. LAB then continuously pushes an infinitesimal unit of flow from v to the neighbor $u \in N(v)$ maximizing $w_u(1 - f(A_u, X_u))$, where X_u is the total amount allocated to u by the algorithm right before it starts pushing this infinitesimal unit of flow. This continues until v is fully matched (i.e. one unit of flow is pushed) or all neighbors are saturated.

Intuition behind the algorithm. First, we give intuition for the algorithm. For an online vertex v and an offline neighbor $u \in N(v)$, the amount allocated from v to u should depend on three factors. Firstly, a higher w_u should lead to larger $x_{u,v}$. Secondly, the more u is filled, the less desirable it is to allocate to it further, preserving capacity for future vertices. Thirdly, vertices favored by the advice should receive more allocation.

The classical BALANCE algorithm handles the first two factors by choosing the offline vertex with the highest potential value $w_u(1 - g(X_u))$ via a convex increasing penalty function $g(z) = e^{z-1}$. To incorporate the third factor, LAB introduces an advice-aware penalty function $f(A, X)$ that also depends on the total advice allocation A; see Fig. 2. This function is increasing in X (penalizing already-filled vertices) and decreasing in A (lower penalty for vertices recommended by the advice), thereby encouraging alignment with the advice.

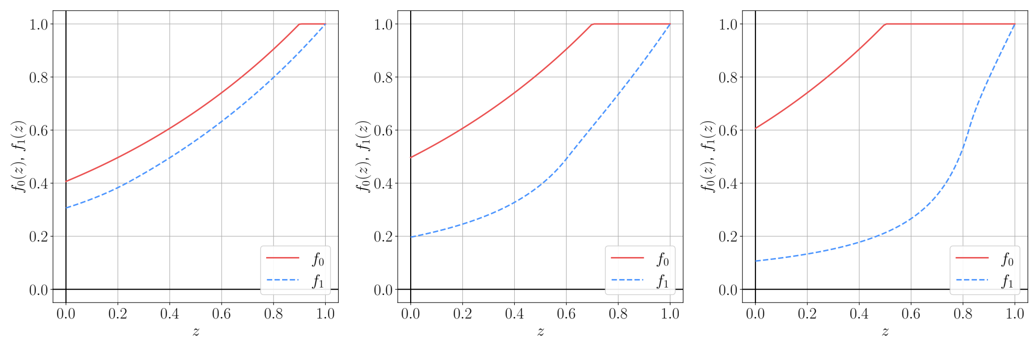
The penalty function f used by our algorithm is defined in Eq. (5) based on the functions f_0 and f_1 from Eq. (4). While f_0 and f_1 are derived from the primal-dual analysis, and their exact forms are not crucial for intuition, the structure of f admits a natural interpretation. Intuitively, if an offline vertex u has received less allocation than the advice suggests (i.e., $A_u > X_u$), then the penalty function treats u as if it were already saturated under the advice. Conversely, if u has been filled beyond the advised amount (i.e., $A_u \leq X_u$), then the penalty effectively treats the excess allocation $X_u - A_u$ as if it were added despite the advice indicating u should be unmatched.

The algorithm also takes as input a parameter $\lambda \in [0, 1]$, which determines how much it trusts the advice; this is directly reflected in the choice of f. When $\lambda = 0$, the penalty reduces to $f(A, X) = e^{X-1}$, and the algorithm recovers the classical BALANCE algorithm with a $(1 - 1/e)$-competitive guarantee. At the other extreme, when $\lambda = 1$, the penalty becomes

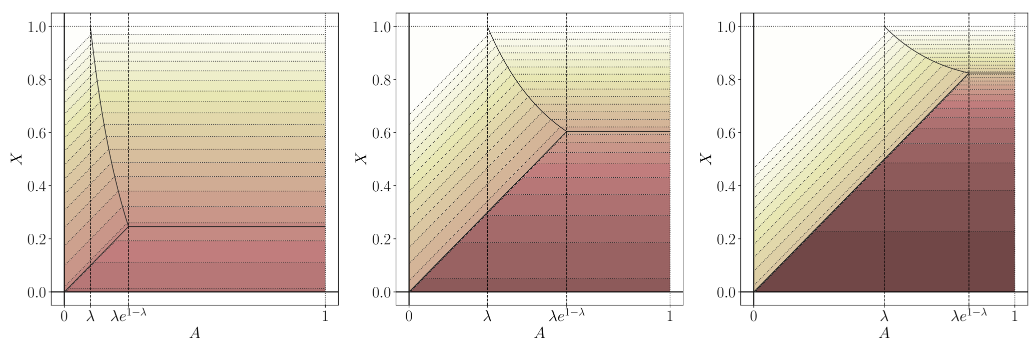
$$f(A, X) = \begin{cases} 0, & \text{if } A > X, \\ 1, & \text{if } A \leq X, \end{cases}$$

causing the algorithm to follow the advice exactly. As λ varies from 0 to 1, the algorithm achieves the robustness-consistency tradeoff described in Theorem 1. We illustrate the behavior of f_0, f_1, and f across different values of λ in Fig. 2.

We now restate and prove Theorem 1 in the remaining of this section:



(a) f_0, f_1 with $\lambda = 0.1$ (b) f_0, f_1 with $\lambda = 0.3$ (c) f_0, f_1 with $\lambda = 0.5$



(d) f with $\lambda = 0.1$ (e) f with $\lambda = 0.3$ (f) f with $\lambda = 0.5$

Figure 2: f_0, f_1, and f with $\lambda \in \{0.1, 0.3, 0.5\}$. (a)-(c) depict the function values of f_0 and f_1 with respect to $z \in [0, 1]$. (d)-(f) depict the contour plots with respect to $A \in [0, 1]$ and $X \in [0, 1]$: the brighter the color is, the closer to 1 the function value is.

Theorem 1. *For any tradeoff parameter $\lambda \in [0, 1]$,* LearningAugmentedBalance *is an $r(\lambda)$-robust and $c(\lambda)$-consistent algorithm for vertex-weighted online bipartite fractional matching, where*

$$r(\lambda) := 1 - e^{\lambda-1} - \left(e^{\lambda-1} - \lambda\right) \ln(1 - \lambda e^{1-\lambda}) - \lambda(1 - \lambda) \quad \text{and} \quad c(\lambda) := 1 + \lambda - e^{\lambda-1}.$$

Our analysis proceeds as follows. In Section 3.1, we perform a primal-dual analysis to bound robustness and consistency of LAB using expressions involving f. In Section 3.2, we analyze these expressions under integral advice to establish the guarantees in Theorem 1. In Section 3.3, we extend the results to fractional advice. Finally, in Section 3.4, we show how LAB extends to the AdWords problem under the small bids assumption.

3.1 Primal-dual analysis

We prove Theorem 1 using an online primal-dual analysis. Recall the primal and dual LPs for vertex-weighted bipartite matching in Section 2. We will maintain dual variables $(\alpha, \beta) \in \mathbb{R}^U \times \mathbb{R}^V$ online, such that the objective value of the dual is equal to the total weight obtained by the algorithm.

Properties of penalty functions. Before presenting the construction of our dual variables, we first argue that the penalty functions used in the algorithm indeed satisfy the properties that we previously discussed.

Lemma 7. *The functions f_0 and f_1 from Eq. (4) and f from Eq. (5) satisfy the following properties:*

1. *f_0 and f_1 are increasing with $f_0(1) = f_1(1) = 1$;*
2. *for any $z \in [0, 1]$, $f_0(z) \geq f_1(z)$;*
3. *for any $A \in [0, 1]$, $f(A, X)$ is increasing on $X \in [0, 1]$ with $f(A, 1) = 1$;*

4. *for any $X \in [0,1]$, $f(A,X)$ is decreasing on $A \in [0,1]$;*
5. *for any $X \in [0,1]$, $f(0,X) = f_0(X)$ and $f(1,X) = f_1(X)$.*

Proof. Let us first prove Property 1. It is trivial to see that f_0 is an increasing function with $f_0(1) = 1$. We can also see that f_1 is an increasing function since $z \mapsto -\lambda e^{1-\lambda-z}$ is increasing on $z \in [\lambda e^{1-\lambda}, 1]$, and $t \mapsto W(t)$ is also increasing on $t \geq -\lambda e^{1-\lambda-\lambda e^{1-\lambda}} \geq -1/e$. We have $f_1(1) = 1$ by definition.

For Property 2, we prove a stronger statement that $f_1(z) \leq e^{z-1}$ for any $z \in [0,1]$; note that $f_0(z) = e^{z+\lambda-1} \geq e^{z-1}$ since $\lambda \in [0,1]$. Indeed, for $z \in [0, \lambda e^{1-\lambda})$, observe that $f_1(z) \leq e^{z-1}$ is implied by

$$\frac{1-z}{e^{1-z}} > \frac{1-\lambda e^{1-\lambda}}{e^{1-\lambda e^{1-\lambda}}} \geq \frac{1-\lambda e^{1-\lambda}}{e^{1-\lambda}} = e^{\lambda-1} - \lambda,$$

where the first inequality comes from that $z \mapsto \frac{1-z}{e^{1-z}}$ is decreasing on $z \geq 0$, and the second from that $e^{1-\lambda e^{1-\lambda}} \leq e^{1-\lambda}$ for $\lambda \in [0,1]$. Meanwhile, for $\lambda \in [0,1]$ and $z \in [\lambda e^{1-\lambda}, 1)$, we have

$$-\lambda e^{1-z} \cdot e^{-\lambda e^{1-z}} > -\lambda e^{1-z} \cdot e^{-\lambda} \geq -\lambda e^{1-\lambda} \cdot e^{-\lambda e^{1-\lambda}} \geq -\frac{1}{e},$$

where the first inequality follows from that $-\lambda e^{1-z} < -\lambda$, and the second from that $z \geq \lambda e^{1-\lambda}$. We thus have

$$-\lambda e^{1-z} > W(-\lambda e^{1-\lambda-z}) \geq -\lambda e^{1-\lambda} \geq -1,$$

where the first inequality implies $f_1(z) \leq e^{z-1}$ while the last two inequalities show that f_1 is well-defined.

The rest of the properties can be easily shown by the above two properties. For Property 3, it is easy to see that $f(A,X)$ is increasing on X by the definition of f as well as Property 1. Note also that $f(A,1) = 1$ again due to Property 1. For Property 4, $f_0(X-A)$ is decreasing on $A \in [0,X]$ by Property 1 . The proof then follows from the definition of f. Lastly, for Property 5, notice that $f(0,X) = \max\{f_0(X), f_1(X)\} = f_0(X)$ for any $X \in [0,1]$ due to Property 2. It is trivial by definition to see $f(1,X) = f_1(X)$ for any $X \in [0,1]$. ☐

Construction of dual variables. We now describe the construction of the dual variables. First, initialize all dual variables to 0. Consider now an iteration when an online vertex $v \in V$ arrives. For each offline vertex $u \in U$, let $A_u^{(v)}$ denote the total amount allocated to u by the advice up to and including v. For each neighbor $u \in N(v)$ of v, let $x_{u,v}$ denote the amount that u is filled by the algorithm in this iteration, and let $X_u^{(v)}$ be the total amount allocated to u by the algorithm by the end of this iteration. We set

- $\alpha_u \leftarrow \alpha_u + x_{u,v} \cdot w_u f(A_u^{(v)}, X_u^{(v)})$ for every $u \in N(v)$, and
- $\beta_v \leftarrow \max_{u \in N(v)} \left\{ w_u (1 - f(A_u^{(v)}, X_u^{(v)})) \right\}$.

Note that the α_u variables are potentially increased in each iteration of the algorithm. On the other hand, each β_v variable is only updated once throughout the execution of the algorithm – in the iteration when v arrives.

Observation 8. For each $v \in V$, if $\sum_{u \in N(v)} x_{u,v} < 1$, we then have $\beta_v = 0$. Moreover, for every $u \in N(v)$ with $x_{u,v} > 0$, $w_u(1 - f(A_u^{(v)}, X_u^{(v)}))$ is constant.

Proof. For the first statement, notice that the algorithm pushes water from v until v is fully matched or its neighbors $N(v)$ are saturated. Therefore, $\sum_{u \in N(v)} x_{u,v} < 1$ implies that $X_u^{(v)} = 1$ for all $u \in N(v)$, and hence, $\beta_v = 0$ since $f(\cdot, 1) = 1$.

The second statement is because the algorithm always allocates an infinitesimal unit to a neighbor with highest potential, so all neighbors $u \in N(v)$ with $x_{u,v} > 0$ must have the same potential at the end of the iteration. ☐

Lemma 9. *The value of the algorithm is equal to the objective value of (α, β) in the dual LP.*

Proof. Let ALG and DUAL denote the total weight obtained by the algorithm and the objective value of (α, β) at any iteration. At the very beginning, we have $\mathsf{ALG} = \mathsf{DUAL} = 0$. We will show that $\Delta\mathsf{ALG} = \Delta\mathsf{DUAL}$ in each iteration of the algorithm. Consider some iteration when an online vertex v arrives. In that iteration, we have $\Delta\mathsf{ALG} = \sum_{u \in N(v)} w_u x_{u,v}$. Let us now calculate $\Delta\mathsf{DUAL}$. For clarity, let $A_u := A_u^{(v)}$ and $X_u := X_u^{(v)}$. We then have

$$\begin{aligned}
\Delta\mathsf{DUAL} &= \sum_{u \in U} \Delta\alpha_u + \beta_v \\
&\overset{(a)}{=} \sum_{u \in N(v)} x_{u,v} \cdot w_u f(A_u, X_u) + \sum_{u \in N(v)} x_{u,v} \cdot \beta_v \\
&= \sum_{u \in N(v)} x_{u,v} \cdot (w_u f(A_u, X_u) + \beta_v) \\
&\overset{(b)}{=} \sum_{u \in N(v)} x_{u,v} \cdot (w_u f(A_u, X_u) + w_u(1 - f(A_u, X_u))) \\
&= \sum_{u \in N(v)} w_u x_{u,v} \\
&= \Delta\mathsf{ALG},
\end{aligned}$$

where both (a) and (b) follow from Observation 8. ☐

We now analyze the robustness of the algorithm.

Lemma 10. *The algorithm is r-robust for any r satisfying that, for any $(u, v) \in E$,*

$$r \leq \int_0^{X_u^{(v)}} f(A_u^{(v)}, z)\, dz + (1 - f(A_u^{(v)}, X_u^{(v)})). \tag{6}$$

Proof. Due to Lemmas 5 and 9, it suffices to prove that that dual is approximately feasible, i.e.,

$$\alpha_u + \beta_v \geq r \cdot w_u \text{for all } (u, v) \in E.$$

Note that, for any $(u, v) \in E$, we have

$$\begin{aligned}
\alpha_u + \beta_v &\overset{(a)}{\geq} \sum_{t \preceq v} x_{u,t} \cdot w_u f(A_u^{(t)}, X_u^{(t)}) + \max_{u' \in N(v)} \left\{ w_{u'}(1 - f(A_{u'}^{(v)}, X_{u'}^{(v)})) \right\} \\
&\geq \sum_{t \preceq v} x_{u,t} \cdot w_u f(A_u^{(t)}, X_u^{(t)}) + w_u(1 - f(A_u^{(v)}, X_u^{(v)})) \\
&\overset{(b)}{\geq} \sum_{t \preceq v} x_{u,t} \cdot w_u f(A_u^{(v)}, X_u^{(t)}) + w_u(1 - f(A_u^{(v)}, X_u^{(v)})) \\
&\overset{(c)}{\geq} w_u \cdot \left[\int_0^{X_u^{(v)}} f(A_u^{(v)}, z)\, dz + (1 - f(A_u^{(v)}, X_u^{(v)})) \right],
\end{aligned}$$

where (a) is because α_u does not decrease throughout the execution, (b) is because $f(A, X)$ is decreasing in A, and (c) is because $f(A, X)$ is increasing in X. Therefore, the algorithm is r-robust whenever r satisfies that, for any $(u, v) \in E$,

$$r \leq \int_0^{X_u^{(v)}} f(A_u^{(v)}, z)\, dz + (1 - f(A_u^{(v)}, X_u^{(v)})).$$

☐

Next, we analyze the consistency of the algorithm.

Lemma 11. *The algorithm is c-consistent, for any value of c satisfying that, for every $u \in U$,*

$$\sum_{t \in N(u)} \left[x_{u,t} \cdot f(A_u^{(t)}, X_u^{(t)}) + a_{u,t} \cdot (1 - f(A_u^{(t)}, X_u^{(t)})) \right] \geq c \cdot A_u, \tag{7}$$

where $A_u := \sum_{t \in N(u)} a_{u,t}$ denotes the total amount that u is eventually filled by the advice.

Proof. Our goal here is to prove that $\mathsf{ALG} \geq c \cdot \mathsf{ADVICE}$, where ALG is the value earned by the algorithm, and ADVICE is the value earned by the advice. On the one hand, we have

$$\mathsf{ADVICE} = \sum_{u \in U} w_u A_u.$$

On the other hand, due to Lemma 9, we have

$$\begin{aligned}
\mathsf{ALG} &= \sum_{u \in U} \alpha_u + \sum_{t \in V} \beta_t \\
&\geq \sum_{u \in U} \alpha_u + \sum_{t \in V} \left(\beta_t \cdot \sum_{u \in N(t)} a_{u,t} \right) \\
&= \sum_{u \in U} \alpha_u + \sum_{u \in U} \sum_{t \in N(u)} a_{u,t} \beta_t \\
&= \sum_{u \in U} \left(\alpha_u + \sum_{t \in N(u)} a_{u,t} \beta_t \right),
\end{aligned}$$

where the inequality is due to the feasibility of the advice $\{a_e\}_{e \in E}$. Therefore, to show $\mathsf{ALG} \geq c \cdot \mathsf{ADVICE}$, it suffices to show

$$\alpha_u + \sum_{t \in N(u)} a_{u,t} \beta_t \geq c \cdot w_u A_u \text{for all } u \in U. \tag{8}$$

By construction, observe that the left-hand side of Eq. (8) is bounded by

$$\begin{aligned}
\alpha_u + \sum_{t \in N(u)} a_{u,t} \beta_t &\geq \sum_{t \in N(u)} x_{u,t} \cdot w_u f(A_u^{(t)}, X_u^{(t)}) + \sum_{t \in N(u)} a_{u,t} \cdot w_u (1 - f(A_u^{(t)}, X_u^{(t)})) \\
&= w_u \cdot \sum_{t \in N(u)} \left[x_{u,t} \cdot f(A_u^{(t)}, X_u^{(t)}) + a_{u,t} \cdot (1 - f(A_u^{(t)}, X_u^{(t)})) \right].
\end{aligned}$$

Therefore, Eq. (8) holds for any value of c satisfying that, for all $u \in U$,

$$\sum_{t \in N(u)} \left[x_{u,t} \cdot f(A_u^{(t)}, X_u^{(t)}) + a_{u,t} \cdot (1 - f(A_u^{(t)}, X_u^{(t)})) \right] \geq c \cdot A_u.$$

□

3.2 Integral advice

We first prove Theorem 1 when the advice is integral. This serves as a warm-up, and several ideas from this setting will carry over to the case of fractional advice. When the advice is integral, the expressions for robustness and consistency in Lemmas 10 and 11 simplify considerably.

Lemma 12. *When the advice is integral, the algorithm is r-robust and c-consistent where*

$$r = \min_{X \in [0,1]} \min \left\{ \int_0^X f_0(z)\, dz + (1 - f_0(X)),\ \int_0^X f_1(z)\, dz + (1 - f_1(X)) \right\} \text{ and}$$

$$c = \min_{X \in [0,1]} \min_{Y \in [0,X]} \left\{ \int_0^Y f_0(z)\, dz + (X - Y) \cdot f_1(X) + (1 - f_1(X)) \right\}.$$

Proof. Recall that $f(0, X) = f_0(X)$ and $f(1, X) = f_1(X)$ for any $X \in [0, 1]$. Note also that, for any $(u, v) \in E$, $A_u^{(v)}$ is either 0 or 1. The expression for robustness r then follows directly from Lemma 10.

Let us now turn to consistency. Consider any offline vertex $u \in U$. If u is exposed in the advice (i.e., $A_u = 0$), Eq. (7) in Lemma 11 trivially holds. On the other hand, if u is matched by the advice with an online vertex $v \in N(u)$ (i.e., $a_{u,v} = 1$), we can observe that $A_u^{(t)} = 0$ for any $t \prec v$, and $A_u^{(v)} = 1$. Therefore, Eq. (7) reduces to

$$\sum_{t \in N(u): t \prec v} x_{u,t} \cdot f_0(X_u^{(t)}) + x_{u,v} \cdot f_1(X_u^{(v)}) + (1 - f_1(X_u^{(v)}) \geq c,$$

which is implied by

$$\int_0^{X_u^{(v)} - x_{u,v}} f_0(z)\, dz + x_{u,v} \cdot f_1(X_u^{(v)}) + (1 - f_1(X_u^{(v)})) \geq c$$

due to the fact that f_0 is increasing. Hence, the algorithm is c-consistent for

$$c = \min_{X \in [0,1]} \min_{Y \in [0,X]} \left\{ \int_0^Y f_0(z)\, dz + (X - Y) \cdot f_1(X) + (1 - f_1(X)) \right\}.$$

□

We now prove Theorem 1 when the advice is integral. Recall that we defined f_0 and f_1 in Eq. (4) as follows:

$$f_0(z) := \min\{e^{z+\lambda-1}, 1\}, \text{ and } f_1(z) := \begin{cases} \frac{e^{\lambda-1}-\lambda}{1-z}, & \text{if } z \in [0, \lambda e^{1-\lambda}), \\ \frac{-\lambda}{W(-\lambda e^{1-\lambda-z})}, & \text{if } z \in [\lambda e^{1-\lambda}, 1), \\ 1, & \text{if } z = 1. \end{cases}$$

The below lemma determines the robustness of LAB.

Lemma 13. *We have*

$$\begin{aligned} r(\lambda) &:= 1 - e^{\lambda-1} - \left(e^{\lambda-1} - \lambda\right) \ln(1 - \lambda e^{1-\lambda}) - \lambda(1 - \lambda) \\ &= \min_{X \in [0,1]} \min \left\{ \int_0^X f_0(z)\, dz + (1 - f_0(X)), \int_0^X f_1(z)\, dz + (1 - f_1(X)) \right\}. \end{aligned}$$

Proof. We first consider the term defined by f_0. Observe that, for any $X \in [0, 1 - \lambda]$,

$$\int_0^X f_0(z)\, dz + (1 - f_0(X)) = 1 - e^{\lambda-1}.$$

Moreover, for any $X \in (1 - \lambda, 1]$, we can also see that

$$\int_0^X f_0(z)\, dz + (1 - f_0(X)) \geq \int_0^{1-\lambda} f_0(z)\, dz = 1 - e^{\lambda-1}.$$

We now turn to the other term defined by f_1. Let

$$I_1(X) := \int_0^X f_1(z)\, dz + (1 - f_1(X)).$$

For $X \in (0, 1)$, one can calculate the derivative of $I_1(X)$ as follows (see Eq. (2)):

$$\begin{aligned} \frac{d}{dX}\left[I_1(X)\right] &= f_1(X) - \frac{d}{dX} f_1(X) \\ &= \begin{cases} -\dfrac{(e^{\lambda-1} - \lambda)X}{(1 - X)^2}, & \text{for } X \in (0, \lambda e^{1-\lambda}), \\ -\dfrac{\lambda}{1 + W\left(-\lambda e^{1-\lambda-X}\right)}, & \text{for } X \in [\lambda e^{1-\lambda}, 1). \end{cases} \end{aligned}$$

Observe that $e^{\lambda-1} - \lambda \geq 0$ for any $\lambda \in [0,1]$. Note also that, for any $\lambda < 1$ and $X \geq \lambda e^{1-\lambda}$, we have $1 + W(-\lambda e^{1-\lambda-X}) > 0$ since W is an increasing function with $W(-\lambda e^{1-\lambda-\lambda e^{1-\lambda}}) = -\lambda e^{1-\lambda} > -1$. This implies that $I_1(X)$ is decreasing in on $(0,1)$, and hence, its minimum is attained at $X = 1$.

We now compute $I_1(1)$. Let $w(z) := W(-\lambda e^{\,1-\lambda-z})$. We then have

$$I_1(1) = \int_0^1 f_1(z)\,dz = \underbrace{\int_0^{\lambda e^{1-\lambda}} \frac{e^{\lambda-1}-\lambda}{1-z}\,dz}_{(A)} + \underbrace{\int_{\lambda e^{1-\lambda}}^1 \left(-\frac{\lambda}{w(z)}\right) dz}_{(B)}\,.$$

The first part is

$$(A) = \int_0^{\lambda e^{1-\lambda}} \frac{e^{\lambda-1}-\lambda}{1-z}\,dz = -\left(e^{\lambda-1}-\lambda\right)\ln\left(1-\lambda e^{1-\lambda}\right).$$

For the second part, since $we^w = -\lambda e^{1-\lambda-z}$, we have $dz = -\frac{1+w}{w}\,dw$. Observe also that $w(1) = W(-\lambda e^{-\lambda}) = -\lambda$ and $w(\lambda e^{1-\lambda}) = W(-\lambda e^{1-\lambda-\lambda e^{1-\lambda}}) = -\lambda\, e^{1-\lambda}$. We thus have

$$\begin{aligned}
(B) &= \int_{\lambda e^{1-\lambda}}^1 \left(-\frac{\lambda}{w(z)}\right) dz \\
&= \lambda \int_{-\lambda e^{1-\lambda}}^{-\lambda} \left(\frac{1}{w^2}+\frac{1}{w}\right) dw \\
&= \lambda \left[-\frac{1}{w} + \ln(-w)\right]_{-\lambda e^{1-\lambda}}^{-\lambda} \\
&= 1 - e^{\lambda-1} - \lambda(1-\lambda).
\end{aligned}$$

We can therefore conclude that

$$I_1(1) = (A) + (B) = 1 - e^{\lambda-1} - \left(e^{\lambda-1}-\lambda\right)\ln(1-\lambda e^{1-\lambda}) - \lambda(1-\lambda).$$

It remains to show that $I_1(1) \leq 1 - e^{\lambda-1}$ for all $\lambda \in [0,1]$. This is equivalent to showing that $G(\lambda) \geq 0$ for all $\lambda \in [0,1]$, where

$$G(\lambda) := \lambda(1-\lambda) + (e^{\lambda-1}-\lambda)\ln(1-\lambda e^{1-\lambda}).$$

To prove this, we need the following inequality:

Proposition 14. *For any* $t \in (0,1)$, *we have* $\ln(1-t) \geq \frac{-t}{\sqrt{1-t}}$.

Proof. Let us substitute $s := \sqrt{1-t}$ for simplicity. It then suffices to show that, for any $s \in (0,1)$,

$$\ln s^2 \geq \frac{s^2-1}{s} \iff s - 2\ln s - \frac{1}{s} \leq 0.$$

Let $h(s) := s - 2\ln s - \frac{1}{s}$. Observe that $h(1) = 0$. Note also that $h'(s) = 1 - \frac{2}{s} + \frac{1}{s^2} = \frac{(s-1)^2}{s^2} \geq 0$ for any $s \in (0,1)$. This implies that h is increasing on $(0,1)$ with $h(1) = 0$, completing the proof. $\square$

We are now ready to prove $G(\lambda) \geq 0$ for all $\lambda \in [0,1]$. Note that $G(0) = G(1) = 0$. For $\lambda \in (0,1)$, it suffices to show that

$$1 - \lambda \geq \left(1 - \frac{1}{\lambda e^{1-\lambda}}\right)\ln(1-\lambda e^{1-\lambda}). \tag{9}$$

Since $1 - \frac{1}{\lambda e^{1-\lambda}} < 0$ for $\lambda \in (0,1)$, due to Proposition 14 with $t := \lambda e^{1-\lambda} \in (0,1)$, we have

$$\left(1 - \frac{1}{\lambda e^{1-\lambda}}\right)\ln(1-\lambda e^{1-\lambda}) \leq \left(1 - \frac{1}{\lambda e^{1-\lambda}}\right)\cdot\frac{-\lambda e^{1-\lambda}}{\sqrt{1-\lambda e^{1-\lambda}}} = \sqrt{1-\lambda e^{1-\lambda}}.$$

We claim that

$$1 - \lambda \geq \sqrt{1-\lambda e^{1-\lambda}}.$$

Note that this claim immediately implies Eq. (9). For simplicity, let $t := 1 - \lambda$. It is then equivalent to showing that, for any $t \in (0, 1)$,

$$g(t) := t^2 + (1 - t)e^t \geq 1.$$

Since we have $g'(t) = t(2 - e^t)$, we can infer that $\inf_{t \in (0,1)} g(t) = \min\{g(0), g(1)\} = 1$. $\square$

The next lemma determines the consistency of LAB.

Lemma 15. *We have*

$$\begin{aligned} c(\lambda) &:= 1 + \lambda - e^{\lambda - 1} \\ &= \min_{X \in [0,1]} \min_{Y \in [0,X]} \left\{ \int_0^Y f_0(z)\, dz + (X - Y) \cdot f_1(X) + (1 - f_1(X)) \right\}. \end{aligned}$$

Proof. For simplicity of presentation, we define

$$J(X, Y) := \int_0^Y f_0(z)\, dz + (X - Y) f_1(X) + (1 - f_1(X)).$$

Notice that, by definition,

$$c(\lambda) = \min_{X \in [0,1]} \min_{Y \in [0,X]} J(X, Y). \tag{10}$$

Let us first calculate the inner minimizer $Y^\star(X)$ for each X fixed. Observe that

$$\frac{\partial}{\partial Y} J(X, Y) = f_0(Y) - f_1(X).$$

If $X < \lambda e^{1-\lambda}$, we have $f_1(X) = \frac{e^{\lambda-1} - \lambda}{1 - X} < e^{\lambda - 1} \leq f_0(Y)$ for any $Y \in [0, X]$, implying that $J(X, Y)$ is increasing on $Y \in [0, X]$. Therefore, the minimum is attained at $Y^\star(X) := 0$. On the other hand, if $X \geq \lambda e^{1-\lambda}$, since $f_1(X) \in [e^{\lambda-1}, 1]$ and $f_0(Y) = e^{\lambda - 1 + Y} \in [e^{\lambda-1}, 1]$ on $Y \in [0, 1 - \lambda]$, we can see that $J(X, Y)$ is minimized at $Y^\star(X)$ such that $f_0(Y^\star(X)) = f_1(X)$. We therefore have

$$Y^\star(X) = \begin{cases} 0, & \text{if } X < \lambda e^{1-\lambda}, \\ 1 - \lambda + \ln f_1(X), & \text{if } X \geq \lambda e^{1-\lambda}. \end{cases}$$

Note that $Y^\star(X) \in [0, 1 - \lambda]$.

Let us now compute $c(\lambda)$ from Eq. (10). We claim $J(X, Y^\star(X)) = 1 + \lambda - e^{\lambda - 1}$ for any $X \in [0, 1]$; note that this claim immediately implies the lemma. If $X < \lambda e^{1-\lambda}$, we know $Y^\star(X) = 0$. We therefore have

$$J(X, 0) = X \frac{e^{\lambda-1} - \lambda}{1 - X} + \left(1 - \frac{e^{\lambda-1} - \lambda}{1 - X}\right) = 1 + \lambda - e^{\lambda-1}.$$

We now consider the other case when $X \geq \lambda e^{1-\lambda}$. Since $Y^\star(X) = 1 - \lambda + \ln f_1(X) \leq 1 - \lambda$, we have

$$\int_0^{Y^\star(X)} f_0(z)\, dz = e^{\lambda-1}(e^{Y^\star(X)} - 1) = f_1(X) - e^{\lambda-1}.$$

Moreover, we can also derive that

$$\begin{aligned} X - Y^\star(X) &= X - 1 + \lambda - \ln f_1(X) \\ &= \ln\left(\frac{W(-\lambda e^{1-\lambda-X})}{-\lambda e^{1-\lambda-X}}\right) \\ &= -W(-\lambda e^{1-\lambda-X}), \end{aligned}$$

where the second equality is due to Eq. (3), implying that

$$(X - Y^\star(X)) f_1(X) = -W(-\lambda e^{1-\lambda-X}) \cdot \frac{-\lambda}{W(-\lambda e^{1-\lambda-X})} = \lambda.$$

From these equations, we finally have

$$\begin{aligned} J(X, Y^\star(X)) &= \int_0^{Y^\star(X)} f_0(z)\,dz + (X - Y^\star(X)) f_1(X) + (1 - f_1(X)) \\ &= f_1(X) - e^{\lambda - 1} + \lambda + 1 - f_1(X) \\ &= 1 + \lambda - e^{\lambda - 1}. \end{aligned}$$

□

3.3 Fractional advice

In this subsection, we show that even when the advice can be fractional, LAB achieves the same robustness and consistency ratios as in the integral case. Recall the definition of $f(A, X)$:

$$f(A, X) := \begin{cases} f_1(X), & \text{if } A > X, \\ \max\{f_0(X - A), f_1(X)\}, & \text{if } A \leq X. \end{cases}$$

where $f_1(\cdot)$ and $f_0(\cdot)$ are the penalty functions used in the integral case.

We first consider robustness. Together with Lemma 10, the below lemma shows that the algorithm is $r(\lambda)$-robust. Its proof easily follows from the integral case.

Lemma 16. *For any $A \in [0, 1]$ and $X \in [0, 1]$, we have*

$$\int_0^X f(A, z) dz + (1 - f(A, X)) \geq r(\lambda).$$

Proof. From the definition of $f(A, X)$, we know that either $f(A, X) = f_1(X)$ or $f(A, X) = f_0(X - A)$. First, suppose $f(A, X) = f_1(X)$. Then

$$\int_0^X f(A, z) dz + (1 - f(A, X)) \geq \int_0^X f_1(z) dz + (1 - f_1(X)) \geq r(\lambda),$$

where the first inequality comes from the fact that $f(A, z) \geq f_1(z)$ for every $z \in [0, 1]$ and the second from Lemma 13 from the integral case. On the other hand, if $f(A, X) \neq f_1(X)$ and hence $f(A, X) = f_0(X - A)$, this implies that $X \geq A$. Hence, we obtain

$$\begin{aligned} \int_0^X f(A, z) dz + (1 - f(A, X)) &\geq \int_A^X f(A, z) dz + (1 - f(A, X)) \\ &\geq \int_A^X f_0(z - A) dz + (1 - f_0(X - A)) \\ &\geq r(\lambda), \end{aligned}$$

where the second inequality follows from that $f(A, z) \geq f_0(z - A)$ for every $z \in [A, 1]$ and the last inequality by Lemma 13 from the integral case. □

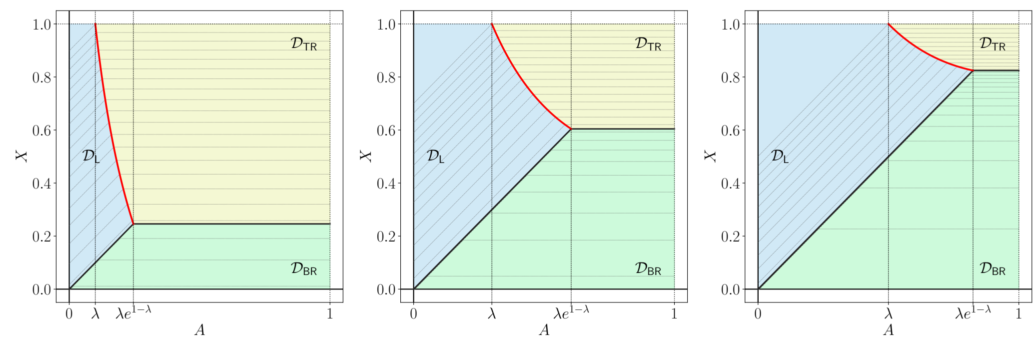
Let us now focus on the consistency. Recall the contour plot of f — see Fig. 3. We partition $[0, 1]^2$ into three regions as follows:

- $\mathcal{D}_{\mathsf{L}} := \{(A, X) \in [0, 1]^2 \mid A \leq X < A - \ln A + (1 - \lambda) + \ln \lambda\}$;
- $\mathcal{D}_{\mathsf{BR}} := \{(A, X) \in [0, 1]^2 \mid X < A \text{ and } X < \lambda e^{1 - \lambda}\}$; and
- $\mathcal{D}_{\mathsf{TR}} := \{(A, X) \in [0, 1]^2 \mid X \geq \lambda e^{1 - \lambda} \text{ and } X \geq A - \ln A + (1 - \lambda) + \ln \lambda\}$.

Observe that $(0, 0) \in \mathcal{D}_{\mathsf{L}}$. These three regions partition $[0, 1]^2$ into parts where f has a simple closed-form.

Lemma 17. *The definition of f in Eq. (5) is equivalent to*

$$f(A, X) := \begin{cases} f_0(X - A), & \text{if } (A, X) \in \mathcal{D}_{\mathsf{L}}, \\ f_1(X) = \frac{e^{\lambda - 1} - \lambda}{1 - X} & \text{if } (A, X) \in \mathcal{D}_{\mathsf{BR}}, \\ f_1(X) = \frac{-\lambda}{W(-\lambda e^{1 - \lambda - X})} & \text{if } (A, X) \in \mathcal{D}_{\mathsf{TR}}. \end{cases}$$



(a) $\lambda = 0.1$ (b) $\lambda = 0.3$ (c) $\lambda = 0.5$

Figure 3: Contour plots of $f(A, X)$ for $\lambda \in \{0.1, 0.3, 0.5\}$. We partition $[0, 1]^2$ into three regions: $\mathcal{D}_{\mathsf{L}}$ (Left, in blue), $\mathcal{D}_{\mathsf{BR}}$ (Bottom Right, in green), $\mathcal{D}_{\mathsf{TR}}$ (Top Right, in yellow). The red curve represents $X = A - \ln A + 1 - \lambda + \ln \lambda$ on $A \in [\lambda, \lambda e^{1-\lambda}]$ that separates $\mathcal{D}_{\mathsf{L}}$ and $\mathcal{D}_{\mathsf{TR}}$.

Proof. To begin, note that the boundaries of the regions intersect at the single point $(\lambda e^{1-\lambda}, \lambda e^{1-\lambda})$. If $X < A$, we have $f(A, X) = f_1(X)$ by the definition of f. Also, if $X < \lambda e^{1-\lambda}$, then $f_1(X) = \frac{e^{\lambda-1}-\lambda}{1-X}$, and if $X \geq \lambda e^{1-\lambda}$ then $f_1(X) = \frac{-\lambda}{W(-\lambda e^{1-\lambda-X})}$. Therefore, we conclude that

$$f(A, X) = \begin{cases} f_1(X) = \frac{e^{\lambda-1}-\lambda}{1-X}, & \text{if } (A, X) \in \mathcal{D}_{\mathsf{BR}}, \\ f_1(X) = \frac{-\lambda}{W(-\lambda e^{1-\lambda-X})}, & \text{if } (A, X) \in \mathcal{D}_{\mathsf{TR}} \cap \{(A, X) : X < A\}. \end{cases}$$

On the other hand, suppose $X \geq A$. To finish the proof, it suffices to show that

$$f_0(X - A) \geq f_1(X) \text{ if and only if } X \leq A - \ln A + (1 - \lambda) + \ln \lambda.$$

First, suppose $X < \lambda e^{1-\lambda}$. Then $(A, X) \in \mathcal{D}_{\mathsf{L}}$, and the analysis in Lemma 15 shows that $f_0(X - A) \geq f_1(X)$ in this case. On the other hand, suppose $X \geq \lambda e^{1-\lambda}$. Again, by the analysis in Lemma 15, we have

$$f_0(X - A) \geq f_1(X) \iff X - A \leq Y^\star(X) = 1 - \lambda + \ln f_1(X). \tag{11}$$

Here, $f_1(X) = -\lambda/w$ where $w := W(-\lambda e^{1-\lambda-X})$. Since $we^w = -\lambda e^{1-\lambda-X}$, we have

$$\ln\left(-\frac{\lambda}{w}\right) = w - 1 + \lambda + X.$$

Substituting this back into Eq. (11), we see that Eq. (11) is equivalent to $A \geq -w$. This is equivalent to

$$A - \ln A \geq -w - \ln(-w) = -\ln \lambda + X + \lambda - 1,$$

which is equivalent to the desired inequality

$$X \leq A - \ln A + (1 - \lambda) + \ln \lambda.$$

□

Define a *trajectory* π to be a sequence of $k + 1 \geq 2$ pairs

$$\pi := ((A_0, X_0), (A_1, X_1), \cdots, (A_k, X_k))$$

such that $0 = A_0 \leq \cdots \leq A_k \leq 1$ and $0 = X_0 \leq \cdots \leq X_k \leq 1$. The *cost* of π is defined as

$$\mathsf{cost}(\pi) := \sum_{i=1}^{k} \left[x_i f(A_i, X_i) + a_i(1 - f(A_i, X_i))\right],$$

where $a_i := A_i - A_{i-1} \geq 0$ and $x_i := X_i - X_{i-1} \geq 0$ for every $i \in \{1, \ldots, k\}$.

By Lemma 11, in order to show that the consistency remains the same as in the integral case, it suffices to prove the following lemma.

Lemma 18. *For any trajectory $\pi = ((A_0, X_0), \dots, (A_k, X_k))$, $\mathsf{cost}(\pi) \geq c(\lambda) \cdot A_k$.*

To this end, we first identify a class of trajectories that minimizes the cost. We say a trajectory $\pi = ((A_0, X_0), \dots, (A_k, X_k))$ is *irreducible* if one of the following is satisfied:

1. $(A_k, X_k) \in \mathcal{D}_{\mathsf{BR}}$ and $k = 1$; or
2. $(A_k, X_k) \in \mathcal{D}_{\mathsf{L}} \cup \mathcal{D}_{\mathsf{TR}}$, $A_0 = \dots = A_{k-1} = 0$, $X_{k-1} \leq 1 - \lambda$, and $f(0, X_{k-1}) = f(A_k, X_k)$.

Note that an irreducible trajectory indeed satisfies Lemma 18.

Lemma 19. *For any irreducible trajectory $\pi = ((A_0, X_0), \dots, (A_k, X_k))$, $\mathsf{cost}(\pi) \geq c(\lambda) \cdot A_k$.*

Proof. Suppose first that π satisfies Condition 1. Since $(A_1, X_1) \in \mathcal{D}_{\mathsf{BR}}$, we have $X_1 < \lambda e^{1-\lambda}$ and hence $f(A_1, X_1) = f_1(X_1) = \frac{e^{\lambda-1}-\lambda}{1-X_1} = \frac{1-c(\lambda)}{1-X_1} < f(0,0)$. We thus have

$$
\begin{aligned}
\mathsf{cost}(\pi) &= X_1 f_1(X_1) + A_1(1 - f_1(X_1)) \\
&= X_1 f_1(X_1) + (1 - f_1(X_1)) + (A_1 - 1)(1 - f_1(X_1)) \\
&\geq c(\lambda) + (A_1 - 1) \cdot (1 - f_1(X_1)) \\
&= c(\lambda) + (A_1 - 1) \cdot \left(1 - \frac{1 - c(\lambda)}{1 - X_1}\right) \\
&\geq c(\lambda) + (A_1 - 1) \cdot c(\lambda) \\
&= c(\lambda) \cdot A_1,
\end{aligned}
$$

where the first inequality follows from Lemma 15 in the integral case (with $Y = 0$) and the second inequality from the fact that $A_k \leq 1$ and $X_1 \geq 0$.

Let us now consider the case that π satisfies Condition 2. If $(A_k, X_k) \in \mathcal{D}_{\mathsf{L}}$, then $x_k = A_k$ since $f(0, X_{k-1}) = f(A_k, X_k)$ and the contour lines in $\mathcal{D}_{\mathsf{L}}$ have slope 1. This implies

$$
\begin{aligned}
\mathsf{cost}(\pi) &= \sum_{i=1}^{k-1} x_i f(0, X_i) + x_k f(A_k, X_k) + A_k(1 - f(A_k, X_k)) \\
&\geq A_k f(A_k, X_k) + A_k(1 - f(A_k, X_k)) \\
&= A_k \\
&\geq c(\lambda) \cdot A_k.
\end{aligned}
$$

On the other hand, if $(A_k, X_k) \in \mathcal{D}_{\mathsf{TR}}$, then we have $f(A_k, X_k) = f(1, X_k) = f_1(X_k)$, which implies

$$
\begin{aligned}
\mathsf{cost}(\pi) &= \sum_{i=1}^{k-1} x_i f_0(X_i) + x_k f_1(X_k) + A_k(1 - f_1(X_k)) \\
&\overset{(a)}{\geq} \int_0^{X_{k-1}} f_0(z) dz + x_k f_1(X_k) + A_k(1 - f_1(X_k)) \\
&= \int_0^{X_{k-1}} f_0(z) dz + x_k f_1(X_k) + (1 - f_1(X_k)) + (A_k - 1)(1 - f_1(X_k)) \\
&\overset{(b)}{\geq} c(\lambda) + (A_k - 1) \cdot (1 - f_1(X_k)) \\
&\overset{(c)}{\geq} c(\lambda) + (A_k - 1) \cdot (1 - e^{\lambda-1}) \\
&\overset{(d)}{\geq} c(\lambda) + (A_k - 1) \cdot c(\lambda) \\
&= c(\lambda) \cdot A_k,
\end{aligned}
$$

where (a) comes from the fact that f_0 is increasing, (b) is from Lemma 15 in the integral case (with $Y = X_{k-1}$ and $X = X_k$), (c) is because $X_k \geq \lambda e^{1-\lambda}$ and f_1 is an increasing function with $f_1(\lambda e^{1-\lambda}) = e^{\lambda-1}$, and (d) is because $c(\lambda) = 1 + \lambda - e^{\lambda-1} \geq 1 - e^{\lambda-1}$. $\square$

Let us now show that the cost of a trajectory is minimized when it is irreducible. In particular, we will prove the following lemma.

Lemma 20. *For any trajectory π, there exists an irreducible trajectory π' such that* $\mathsf{cost}(\pi) \geq \mathsf{cost}(\pi')$.

To prove this lemma, we fix an arbitrary trajectory π and apply a sequence of local modifications without ever increasing the cost of the trajectory. The following is a key technical lemma for the local modification.

Lemma 21. *Fix a trajectory $\pi = ((A_0, X_0), \ldots, (A_k, X_k))$ with $k \geq 2$. For some $i \in \{1, \ldots, k-1\}$, let π' be a new trajectory obtained by removing (A_i, X_i) from π. Then, we have* $\mathsf{cost}(\pi) \geq \mathsf{cost}(\pi')$ *if and only if one of the following holds:*

(I) $a_i \leq x_i$ and $f(A_i, X_i) \geq f(A_{i+1}, X_{i+1})$;

(II) $a_i \geq x_i$ and $f(A_i, X_i) \leq f(A_{i+1}, X_{i+1})$,

where $a_i = A_i - A_{i-1}$ and $x_i = X_i - X_{i-1}$.

Proof. Since π' is obtained by removing (A_i, X_i) from π, we have

$$
\begin{aligned}
&\mathsf{cost}(\pi) - \mathsf{cost}(\pi') \\
&= \Big[(X_i - X_{i-1}) f(A_i, X_i) + (A_i - A_{i-1})(1 - f(A_i, X_i)) \\
&\quad + (X_{i+1} - X_i) f(A_{i+1}, X_{i+1}) + (A_{i+1} - A_i)(1 - f(A_{i+1}, X_{i+1}))\Big] \\
&\quad - \Big[(X_{i+1} - X_{i-1}) f(A_{i+1}, X_{i+1}) + (A_{i+1} - A_{i-1})(1 - f(A_{i+1}, X_{i+1}))\Big] \\
&= (x_i - a_i) \cdot (f(A_i, X_i) - f(A_{i+1}, X_{i+1})).
\end{aligned}
$$

Therefore, $\mathsf{cost}(\pi) - \mathsf{cost}(\pi') \geq 0$ if and only if one of the conditions in the lemma holds. □

The following lemmas further identify pairs that can be removed from π without increasing the cost.

Lemma 22. *Fix a trajectory $\pi = ((A_0, X_0), \ldots, (A_k, X_k))$ with $k \geq 2$. For some $i \in \{1, \ldots, k-1\}$, let π' be a new trajectory obtained by removing (A_i, X_i) from π. If $(A_{i-1}, X_{i-1}) \in \mathcal{D}_{\mathsf{L}}$ and $(A_i, X_i) \in \mathcal{D}_{\mathsf{BR}}$, we have* $\mathsf{cost}(\pi) \geq \mathsf{cost}(\pi')$.

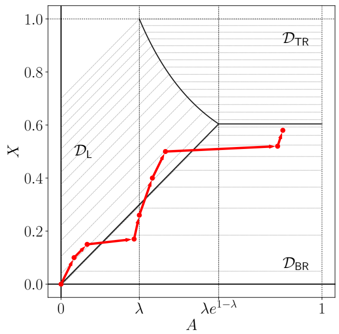
Proof. Since $(A_i, X_i) \in \mathcal{D}_{\mathsf{BR}}$ and $(A_{i-1}, X_{i-1}) \in \mathcal{D}_{\mathsf{L}}$, we have $a_i > x_i$. Moreover, since $(A_i, X_i) \in \mathcal{D}_{\mathsf{BR}}$, we also have $f(A_i, X_i) \leq f(A_{i+1}, X_{i+1})$. These together imply that i satisfies Condition (II) of Lemma 21. □

Lemma 23. *Fix a trajectory $\pi = ((A_0, X_0), \ldots, (A_k, X_k))$ with $k \geq 2$. Suppose there exists s and t such that $0 \leq s < t \leq k$, $(A_s, X_s) \in \mathcal{D}_{\mathsf{L}}$, $(A_t, X_t) \in \mathcal{D}_{\mathsf{L}} \cup \mathcal{D}_{\mathsf{TR}}$, and $(A_{s+1}, X_{s+1}), \ldots, (A_{t-1}, X_{t-1}) \in \mathcal{D}_{\mathsf{BR}}$. Let π' be the trajectory obtained by removing $(A_{s+1}, X_{s+1}), \ldots, (A_{t-1}, X_{t-1})$ from π. We have* $\mathsf{cost}(\pi) \geq \mathsf{cost}(\pi')$.

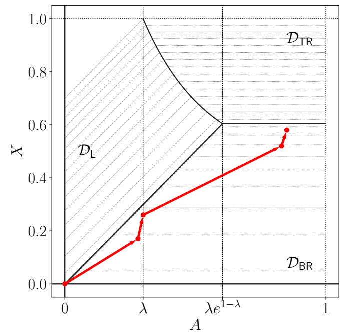
Proof. Note that Lemma 22 holds with π and $i := s + 1$. Repeated application of Lemma 22 leads to the proof of this lemma. □

Lemma 24. *Fix a trajectory $\pi = ((A_0, X_0), \ldots, (A_k, X_k))$ with $k \geq 2$. For some $i \in \{1, \ldots, k-1\}$, let π' be a new trajectory obtained by removing (A_i, X_i) from π. If $(A_i, X_i) \in \mathcal{D}_{\mathsf{L}}$, $a_i \leq x_i$, $(A_{i+1}, X_{i+1}) \in \mathcal{D}_{\mathsf{BR}} \cup \mathcal{D}_{\mathsf{L}}$, and $a_{i+1} \geq x_{i+1}$, we have* $\mathsf{cost}(\pi) \geq \mathsf{cost}(\pi')$.

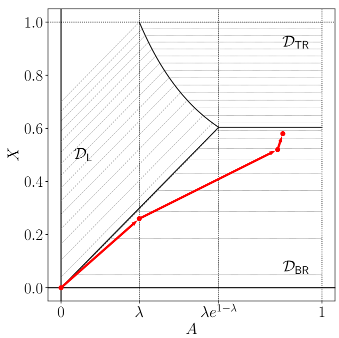
Proof. Note that we have $f(A_i, X_i) \geq f(A_{i+1}, X_{i+1})$. Indeed, if $(A_{i+1}, X_{i+1}) \in \mathcal{D}_{\mathsf{BR}}$, it trivially follows; otherwise if $(A_{i+1}, X_{i+1}) \in \mathcal{D}_{\mathsf{L}}$, it is implied by the condition that $a_{i+1} \geq x_{i+1}$. Therefore, Condition (I) of Lemma 21 is satisfied with this i on π. □



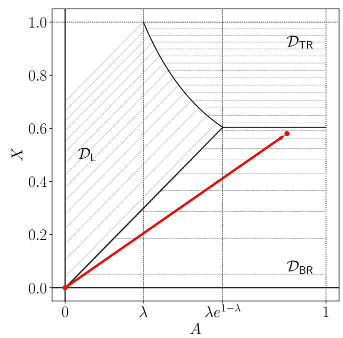
(a) Initial trajectory



(b) Due to Lemma Lemma 25



(c) Due to Lemma Lemma 22



(d) Irreducible trajectory

Figure 4: Illustration of the proof of Lemma 20 in the case where $(A_k, X_k) \in \mathcal{D}_{\mathsf{BR}}$.

Lemma 25. *Fix a trajectory $\pi = ((A_0, X_0), \ldots, (A_k, X_k))$ with $k \geq 2$. Suppose there exists s and t such that $0 \leq s < t \leq k$, $(A_s, X_s), (A_t, X_t) \in \mathcal{D}_{\mathsf{BR}}$, and $(A_{s+1}, X_{s+1}), \ldots, (A_{t-1}, X_{t-1}) \in \mathcal{D}_{\mathsf{L}}$. Let π' be the trajectory obtained by removing $(A_{s+1}, X_{s+1}), \ldots, (A_{t-1}, X_{t-1})$ from π. We have* $\mathsf{cost}(\pi) \geq \mathsf{cost}(\pi')$.

Proof. Since $(A_s, X_s) \in \mathcal{D}_{\mathsf{BR}}$ and $(A_{s+1}, X_{s+1}) \in \mathcal{D}_{\mathsf{L}}$, we have $a_{s+1} < x_{s+1}$. Similarly, we have $a_t > x_t$ due to the condition that $(A_{t-1}, X_{t-1}) \in \mathcal{D}_{\mathsf{L}}$ and $(A_t, X_t) \in \mathcal{D}_{\mathsf{BR}}$. Therefore, there must exist $i \in \{s+1, \ldots, t-1\}$ such that $a_i \leq x_i$ and $a_{i+1} \geq x_{i+1}$. Due to Lemma 24 with π and this i, removing this (A_i, X_i) from π does not increase the cost of the trajectory. This lemma then follows by repeatedly applying this process until all of $(A_{s+1}, X_{s+1}), \ldots, (A_{t-1}, X_{t-1})$ are removed from π. $\square$

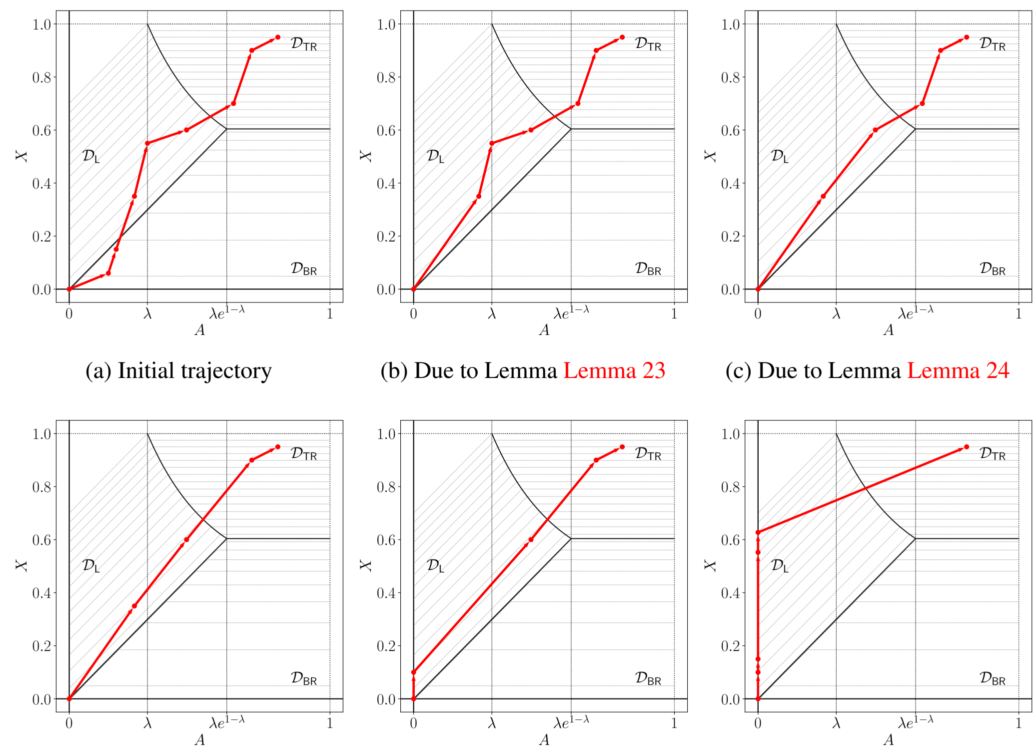
Lemma 26. *Fix a trajectory $\pi = ((A_0, X_0), \ldots, (A_k, X_k))$ with $k \geq 2$. For some $i \in \{1, \ldots, k-1\}$, let π' be a new trajectory obtained by removing (A_i, X_i) from π. If $(A_{i-1}, X_{i-1}) \in \mathcal{D}_{\mathsf{BR}} \cup \mathcal{D}_{\mathsf{L}}$, $(A_i, X_i) \in \mathcal{D}_{\mathsf{TR}}$, and $a_i \geq x_i$, we have* $\mathsf{cost}(\pi) \geq \mathsf{cost}(\pi')$.

Proof. Since $(A_{i+1}, X_{i+1}) \in \mathcal{D}_{\mathsf{TR}}$, we have $f(A_i, X_i) \leq f(A_{i+1}, X_{i+1})$, immediately completing the proof of this lemma due to Condition (II) of Lemma 21. $\square$

Equipped with the above lemmas, we can now prove Lemma 20.

Proof of Lemma 20. We break the proof into two cases depending on the region where the last pair of π is contained.

Case 1. $(A_k, X_k) \in \mathcal{D}_{\mathsf{BR}}$. See Fig. 4 for an illustration of the proof of this case. Due to Lemma 25, we can assume that every pair in π other than $(A_0, X_0) = (0, 0)$ is contained in $\mathcal{D}_{\mathsf{BR}}$. Note however that Lemma 22 is satisfied with $i := 1$ on π. We can therefore remove (A_1, X_1) from π without increasing the cost of the trajectory. By repeatedly applying this process, we end up with a trajectory



(a) Initial trajectory (b) Due to Lemma Lemma 23 (c) Due to Lemma Lemma 24

(d) Due to Lemma Lemma 26 (e) Intermediate trajectory π_1 (f) Irreducible trajectory π_k

Figure 5: Illustration of the proof of Lemma 20 in the case where $(A_k, X_k) \in \mathcal{D}_{\mathsf{L}} \cup \mathcal{D}_{\mathsf{TR}}$.

$\pi' := ((0,0), (A_1, X_1))$ with $(A_1, X_1) \in \mathcal{D}_{\mathsf{BR}}$. Recall that this π' is irreducible, completing the proof of Lemma 20 for this case.

Case 2. $(A_k, X_k) \in \mathcal{D}_{\mathsf{L}} \cup \mathcal{D}_{\mathsf{TR}}$**.** See Fig. 5 for an illustration of the proof of this case. Note that, due to Lemma 23, we can assume that every pair in π is contained in $\mathcal{D}_{\mathsf{L}} \cup \mathcal{D}_{\mathsf{TR}}$.

Let us now argue that we can further assume without loss of generality that the f values of every pair in π does not decrease, i.e.,

$$f(A_0, X_0) \leq \cdots \leq f(A_k, X_k). \tag{12}$$

Let i^* be the last index of a pair contained in $\mathcal{D}_{\mathsf{L}}$, i.e., $(A_0, X_0), \ldots, (A_{i^*}, X_{i^*}) \in \mathcal{D}_{\mathsf{L}}$ and $(A_{i^*+1}, X_{i^*+1}), \ldots, (A_k, X_k) \in \mathcal{D}_{\mathsf{TR}}$. We may have either $i^* = 0$ (i.e., no pairs other than $(0,0)$ are in $\mathcal{D}_{\mathsf{L}}$) or $i^* = k$ (i.e., no pairs are in $\mathcal{D}_{\mathsf{TR}}$). If $i^* > 0$, by Lemma 24 and the fact that $a_1 \leq x_1$, we can assume $a_i \leq x_i$ for every $i = 1, \ldots, i^*$, implying that $f(A_0, X_0) \leq \cdots \leq f(A_{i^*}, X_{i^*})$. Furthermore, if $i^* < k$, by Lemma 26, we can also assume $a_{i^*+1} < x_{i^*+1}$, yielding that $f(A_{i^*}, X_{i^*}) < f(A_{i^*+1}, X_{i^*+1})$. For the remaining indices, we can easily see that $f(A_{i^*+1}, X_{i^*+1}) \leq \cdots \leq f(A_k, X_k)$ since $(A_{i^*+1}, X_{i^*+1}), \ldots, (A_k, X_k) \in \mathcal{D}_{\mathsf{TR}}$.

For $i = 1, \ldots, k$, let X'_i be the value such that $f(0, X'_i) = f(A_i, X_i)$; if $f(A_i, X_i) = 1$, we set $X'_i := \lambda$. Due to Eq. (12) and the fact that $f(0, \cdot)$ is strictly increasing in $[0, \lambda]$, we have $0 \leq X'_1 \leq \cdots \leq X'_k \leq \lambda$. Moreover, by the definition of f, we also have

$$A_i \geq X_i - X'_i \quad \text{if } f(A_i, X_i) < 1. \tag{13}$$

For $i = 1, \ldots, k-1$, let π_i be the trajectory where $(A_1, X_1), \ldots, (A_i, X_i)$ in π are replaced by $(0, X'_1), \ldots, (0, X'_i)$, respectively, i.e.,

$$\pi_i := ((0,0), (0, X'_1), \ldots, (0, X'_i), (A_{i+1}, X_{i+1}), \ldots, (A_k, X_k)).$$

For consistency, let $\pi_0 := \pi$. We claim that, for every $i = 1, \ldots, k-1$, we have $\mathsf{cost}(\pi_{i-1}) \geq \mathsf{cost}(\pi_i)$. Indeed, since π_{i-1} and π_i differ only at index i, we have

$$\begin{aligned}
&\mathsf{cost}(\pi_{i-1}) - \mathsf{cost}(\pi_i) \\
&= \Big[(X_i - X'_{i-1})f(A_i, X_i) + A_i(1 - f(A_i, X_i)) \\
&\qquad + (X_{i+1} - X_i)f(A_{i+1}, X_{i+1}) + (A_{i+1} - A_i)(1 - f(A_{i+1}, X_{i+1}))\Big] \\
&\quad - \Big[(X'_i - X'_{i-1})f(0, X'_i) \\
&\qquad + (X_{i+1} - X'_i)f(A_{i+1}, X_{i+1}) + A_{i+1}(1 - f(A_{i+1}, X_{i+1}))\Big] \\
&= (f(A_{i+1}, X_{i+1}) - f(A_i, X_i)) \cdot (A_i - X_i + X'_i) \\
&\geq 0,
\end{aligned}$$

where the second equality comes from that $f(0, X'_i) = f(A_i, X_i)$ and the inequality from Eqs. (12) and (13). In particular, if $A_i < X_i - X'_i$, we have $f(A_i, X_i) = 1$ due to Eq. (13), and hence, $f(A_{i+1}, X_{i+1}) = 1$ due to Eq. (12).

Finally, let π_k be the trajectory where $(0, X'_k)$ is inserted between $(0, X'_{k-1})$ and (A_k, X_k) in π_{k-1}, i.e.,

$$\pi_k := ((0,0), (0, X'_1), \ldots, (0, X'_{k-1}), (0, X'_k), (A_k, X_k)).$$

Note that $\mathsf{cost}(\pi_k) = \mathsf{cost}(\pi_{k-1})$ since we have

$$\begin{aligned}
&\mathsf{cost}(\pi_{k-1}) - \mathsf{cost}(\pi_k) \\
&= \Big[(X_k - X'_{k-1})f(A_k, X_k) + A_k(1 - f(A_k, X_k))\Big] \\
&\quad - \Big[(X'_k - X'_{k-1})f(0, X'_i) + (X_k - X'_k)f(A_k, X_k) + A_k(1 - f(A_k, X_k))\Big] \\
&= 0
\end{aligned}$$

due to the fact that $f(0, X'_k) = f(A_k, X_k)$. Observe that π_k is irreducible, completing the proof of Lemma 20. ☐

The proof of Lemma 18 then immediately follows from Lemmas 19 and 20.

3.4 Extension to AdWords

In this subsection, we present that LEARNINGAUGMENTEDBALANCE naturally extends to AdWords under the small bids assumption, showing Theorem 2 restated below.

Theorem 2. *Consider the small bids assumption where the maximum bid-to-budget ratio is bounded by some sufficiently small $\varepsilon > 0$. For any tradeoff parameter $\lambda \in [0,1]$, there exists an $r(\lambda) \cdot (1 - 3\sqrt{\varepsilon \ln(1/\varepsilon)})$-robust and $c(\lambda) \cdot (1 - 3\sqrt{\varepsilon \ln(1/\varepsilon)})$-consistent algorithm for AdWords with advice, where $r(\lambda)$ and $c(\lambda)$ are the same as in Theorem 1.*

We first adapt LAB into an algorithm for *fractional* AdWords achieving the same robustness and consistency. We then argue that integral AdWords under the small bids assumption can be reduced to fractional AdWords with small loss.

Recall that, in AdWords, offline U and online V are corresponding to *advertisers* and *impressions*, respectively. Each advertiser $u \in U$ has a budget of B_u. Whenever an impression $v \in V$ is revealed, the algorithm also learns bids $\{b_{u,v}\}_{u \in U}$ from the advertisers; the advice is the advertiser to which this impression should be assigned. The algorithm then needs to assign each impression to an advertiser, making a revenue of its bid unless the advertiser has used up its budget. The objective is to maximize the total revenue. Recall from Section 2 the LP relaxation for AdWords and its dual LP:

$$\begin{array}{llr} \max & \sum_{u\in U}\sum_{v\in V} b_{u,v}x_{u,v} & \\ \text{s.t.} & \frac{1}{B_u}\sum_{v\in V} b_{u,v}x_{u,v} \le 1, & \forall u\in U, \\ & \sum_{u\in U} x_{u,v} \le 1, & \forall v\in V, \\ & x_{u,v} \ge 0, & \forall u\in U, v\in V; \end{array} \qquad \begin{array}{llr} \min & \sum_{u\in U}\alpha_u + \sum_{v\in V}\beta_v & \\ \text{s.t.} & \frac{b_{u,v}}{B_u}\alpha_u + \beta_v \ge b_{u,v}, & \forall u\in U, v\in V, \\ & \alpha_u \ge 0, & \forall u\in U, \\ & \beta_v \ge 0, & \forall v\in V. \end{array}$$

The fractional version of AdWords is corresponding to constructing online a feasible solution to the above primal LP relaxation. We also admit a fractional advice; we again denote by $a \in \mathbb{R}^{U\times V}$ the advice feasible to the primal LP.

Fractional algorithm. When an impression $v \in V$ arrives together with the advice $\{a_{u,v}\}_{u\in U}$, we denote by $A_u := \frac{1}{B_u}\sum_{t\preceq v} a_{u,t}$ for each advertiser $u \in U$ the fraction of u's budget spent by the advice up to and including v. Let f be the same function defined in Eq. (5) for vertex-weighted matching. The algorithm then continuously assigns an infinitesimal unit of impression v to an advertiser $u \in U$ with the highest value of $b_{u,v}(1 - f(A_u, X_u))$, where X_u denotes the fraction of u's budget spent by the algorithm right before this infinitesimal unit is assigned, until either impression v is integrally assigned or the budgets of all advertisers are used up.

Primal-dual analysis. Recall from Section 2 the key lemma of the primal-dual analysis for AdWords:

Lemma 27 (cf. Lemma 6)**.** *Let $x \in \mathbb{R}^{U\times V}$ be the output of a fractional algorithm for AdWords. For some $\rho \in [0, 1]$, if there exists $(\alpha, \beta) \in \mathbb{R}^U \times \mathbb{R}^V$ satisfying*

- *(reverse weak duality) $\sum_{u\in U}\sum_{v\in V} b_{u,v}x_{u,v} \ge \sum_{u\in U}\alpha_u + \sum_{v\in V}\beta_v$ and*
- *(approximate dual feasibility) $\frac{b_{u,v}}{B_u}\alpha_u + \beta_v \ge \rho\cdot b_{u,v}$ for every $u \in U$ and $v \in V$,*

we have $\mathsf{ALG} \ge \rho\cdot\mathsf{OPT}$.

The construction of dual variables is almost identical to the vertex-weighted setting. We first initialize $(\alpha, \beta) \leftarrow (\mathbf{0}, \mathbf{0})$. At the iteration when impression $v \in V$ is revealed, for each advertiser $u \in U$, let $x_{u,v}$ denote the fraction of v assigned to u. Let us also denote by $X_u^{(v)} := \frac{1}{B_u}\sum_{t\preceq v} b_{u,t}x_{u,t}$ and $A_u^{(v)} := \frac{1}{B_u}\sum_{t\preceq v} b_{u,t}a_{u,t}$ the fractions of u's budget spent by the algorithm and the advice, respectively, at the end of this iteration. We set

- $\alpha_u \leftarrow \alpha_u + x_{u,v}\cdot b_{u,v}f(A_u^{(v)}, X_u^{(v)})$ for every $u \in U$, and
- $\beta_v \leftarrow \max_{u\in U}\left\{b_{uv}(1 - f(A_u^{(v)}, X_u^{(v)}))\right\}$.

By the definition of the algorithm, observe that $b_{u,v}(1 - f(A_u^{(v)}, X_u^{(v)}))$ is constant for all $u \in U$ with $x_{u,v} > 0$ and that β_v is equal to this value. Also, note that if $\sum_{u\in U} x_{u,v} < 1$, then $\beta_v = 0$ due to the fact that $\sum_{u\in U} x_{u,v} < 1$ implies all the advertisers have used up their budgets.

We now show that Lemmas 9 to 11 can be easily adapted to fractional AdWords as follows.

Lemma 28 (cf. Lemma 9)**.** *The revenue of the algorithm is equal to the objective value of (α, β) in the dual LP.*

Proof. Let ALG and DUAL denote the revenue of the algorithm and the objective value of (α, β) at any iteration. As we have $\mathsf{ALG} = \mathsf{DUAL} = 0$ at the start, it suffices to show that $\Delta\mathsf{ALG} = \Delta\mathsf{DUAL}$ at each iteration. When impression v is revealed, we have $\Delta\mathsf{ALG} = \sum_{u\in U} b_{u,v}x_{u,v}$. Let us now calculate ΔDUAL. For clarity, let $A_u := A_u^{(v)}$ and $X_u := X_u^{(v)}$. We then have

$$\begin{aligned}\Delta\mathsf{DUAL} &= \sum_{u\in U} \Delta\alpha_u + \beta_v \\ &\overset{(a)}{=} \sum_{u\in U} x_{u,v}\cdot b_{u,v} f(A_u, X_u) + \sum_{u\in U} x_{u,v}\cdot\beta_v \\ &= \sum_{u\in U} x_{u,v}\cdot (b_{u,v} f(A_u, X_u) + \beta_v) \\ &\overset{(b)}{=} \sum_{u\in U} x_{u,v}\cdot (b_{u,v} f(A_u, X_u) + b_{u,v}(1 - f(A_u, X_u))) \\ &= \sum_{u\in U} b_{u,v} x_{u,v} \\ &= \Delta\mathsf{ALG},\end{aligned}$$

where (a) follows from that $\beta_v = 0$ if $\sum_{u\in U} x_{u,v} < 1$, and (b) from that $\beta_v = b_{u,v}(1 - f(A_u, X_u))$ if $x_{u,v} > 0$. ☐

For the counterparts of Lemmas 10 and 11, we define another notation: for every $u \in U$ and $v \in V$, let $\Delta X_u^{(v)} := \frac{b_{u,v} x_{u,v}}{B_u}$ and $\Delta A_u^{(v)} := \frac{b_{u,v} a_{u,v}}{B_u}$ denote the increase of fractions of u's budget spent by the algorithm and the advice, respectively, at iteration when v is revealed.

Lemma 29 (cf. Lemma 10)**.** *The algorithm is r-robust for any r satisfying that, for any $u \in U$ and $v \in V$,*

$$r \le \int_0^{X_u^{(v)}} f(A_u^{(v)}, z)dz + (1 - f(A_u^{(v)}, X_u^{(v)})).$$

Proof. Due to Lemmas 27 and 28, it suffices to prove a bound of the form

$$\frac{b_{u,v}}{B_u}\alpha_u + \beta_v \ge r\cdot b_{u,v} \text{for all } u \in U \text{ and } v \in V.$$

Note that, for any $u \in U$ and $v \in V$, we have

$$\begin{aligned}\frac{b_{u,v}}{B_u}\alpha_u + \beta_v &\overset{(a)}{\ge} \frac{b_{u,v}}{B_u}\sum_{t\preceq v} b_{u,t} x_{u,t}\, f(A_u^{(t)}, X_u^{(t)}) + \max_{u'\in U}\left\{b_{u',v}(1 - f(A_{u'}^{(v)}, X_{u'}^{(v)}))\right\} \\ &\ge \frac{b_{u,v}}{B_u}\sum_{t\preceq v} b_{u,t} x_{u,t}\, f(A_u^{(t)}, X_u^{(t)}) + b_{u,v}(1 - f(A_u^{(v)}, X_u^{(v)})) \\ &\overset{(b)}{\ge} b_{u,v}\sum_{t\preceq v}\Delta X_u^{(t)} f(A_u^{(v)}, X_u^{(t)}) + b_{u,v}(1 - f(A_u^{(v)}, X_u^{(v)})) \\ &\overset{(c)}{\ge} b_{u,v}\cdot\left[\int_0^{X_u^{(v)}} f(A_u^{(v)}, z)dz + (1 - f(A_u^{(v)}, X_u^{(v)}))\right],\end{aligned}$$

where (a) is because α_u does not decrease throughout the execution, (b) is because $f(A, X)$ is decreasing in A, and (c) is because $f(A, X)$ is increasing in X. ☐

Lemma 30 (cf. Lemma 11)**.** *The algorithm is c-consistent, for any value of c satisfying that, for every $u \in U$,*

$$\sum_{t\in N(u)}\left[\Delta X_u^{(t)}\cdot f(A_u^{(t)}, X_u^{(t)}) + \Delta A_u^{(t)}\cdot(1 - f(A_u^{(t)}, X_u^{(t)}))\right] \ge c\cdot A_u,$$

where $A_u := \sum_{t\in V} a_{u,t}$ denotes the fraction of the budget u eventually spends by the advice.

Proof. Our goal here is to prove that $\mathsf{ALG} \geq c \cdot \mathsf{ADVICE}$, where ALG is the revenue earned by the algorithm, and ADVICE is the revenue of the advice. On the one hand, we have

$$\mathsf{ADVICE} = \sum_{u \in U} B_u A_u.$$

On the other hand, due to Lemma 28, we have

$$\begin{aligned}
\mathsf{ALG} &= \sum_{u \in U} \alpha_u + \sum_{t \in V} \beta_t \\
&\geq \sum_{u \in U} \alpha_u + \sum_{t \in V} \left(\beta_t \cdot \sum_{u \in U} a_{u,t} \right) \\
&= \sum_{u \in U} \alpha_u + \sum_{u \in U} \sum_{t \in V} a_{u,t} \beta_t \\
&= \sum_{u \in U} \left(\alpha_u + \sum_{t \in V} a_{u,t} \beta_t \right),
\end{aligned}$$

where the inequality is due to the feasibility of the advice a. Therefore, to show $\mathsf{ALG} \geq c \cdot \mathsf{ADVICE}$, it suffices to show

$$\alpha_u + \sum_{t \in V} a_{u,t} \beta_t \geq c \cdot B_u A_u \text{for all } u \in U. \tag{14}$$

By construction, observe that the left-hand side of Eq. (14) is bounded by

$$\begin{aligned}
\alpha_u + \sum_{t \in V} a_{u,t} \beta_t &\geq \sum_{t \in V} x_{u,t} \cdot b_{u,t} f(A_u^{(t)}, X_u^{(t)}) + \sum_{t \in V} a_{u,t} \cdot b_{u,t} (1 - f(A_u^{(t)}, X_u^{(t)})) \\
&= \sum_{t \in V} \left[b_{u,t} x_{u,t} \cdot f(A_u^{(t)}, X_u^{(t)}) + b_{u,t} a_{u,t} \cdot (1 - f(A_u^{(t)}, X_u^{(t)})) \right].
\end{aligned}$$

Therefore, Eq. (14) holds for any value of c satisfying that, for all $u \in U$,

$$\sum_{t \in V} \left[\Delta X_u^{(t)} \cdot f(A_u^{(t)}, X_u^{(t)}) + \Delta A_u^{(t)} \cdot (1 - f(A_u^{(t)}, X_u^{(t)})) \right] \geq c \cdot A_u$$

□

We can thus show the robustness and consistency of this fractional algorithm for AdWords with fractional advice.

Theorem 31. *For any tradeoff parameter $\lambda \in [0, 1]$, this algorithm is an $r(\lambda)$-robust and $c(\lambda)$-consistent algorithm for fractional AdWords with a fractional advice, where $r(\lambda)$ and $c(\lambda)$ are defined in Theorem 1.*

Proof. Immediate from Lemmas 16, 18 and 27 to 30. □

Reducing integral AdWords to fractional AdWords. It remains to see that integral AdWords under the small bids assumption can be reduced to fractional AdWords with small loss. We remark that the reduction is inspired by [FN24].

Recall that, under the small bids assumption, there exists a sufficiently small $\varepsilon > 0$ such that $b_{u,v} \leq \varepsilon B_u$ for all $u \in U$ and $v \in V$. We will use the following concentration bound in the analysis of the reduction.

Proposition 32 (Bernstein's inequality for bounded independent variables)**.** *Let $Z_1, \ldots, Z_n$ be independent random variables satisfying that, for some constant $c > 0$,*

$$\mathbb{E}[Z_i] = 0 \quad \textit{and} \quad |Z_i| \leq c \ \textit{almost surely}$$

for all $i = 1, \ldots, n$. Define the aggregate variance $\sigma^2 := \sum_{i=1}^n \mathrm{Var}(Z_i) = \sum_{i=1}^n \mathbb{E}[Z_i^2]$. Then, for every $t > 0$,

$$\Pr\Big[\sum_{i=1}^{n} Z_i \ \geq t\Big] \ \leq \ \exp\Big(-\frac{t^2}{2\sigma^2 + \frac{2}{3} c\, t}\Big).$$

We now prove that the reduction is possible with small loss.

Theorem 33. *For sufficiently small $\varepsilon > 0$, given any algorithm for fractional AdWords, we can construct a randomized algorithm for integral AdWords satisfying*

$$\mathbb{E}[\mathsf{ALG}_{\mathsf{int}}] \geq \left(1 - 3\sqrt{\varepsilon \ln(1/\varepsilon)}\right) \mathsf{ALG}_{\mathsf{frac}},$$

where $\mathsf{ALG}_{\mathsf{frac}}$ and $\mathsf{ALG}_{\mathsf{int}}$ denote the revenue earned by the given fractional algorithm and the constructed randomized algorithm, respectively.

Proof. We construct the integral algorithm as follows. When an impression $v \in V$ arrives, let $\{x_{u,v}\}_{u \in U}$ be the (fractional) assignment of the given fractional algorithm at this moment. The integral algorithm then simply samples an advertiser $u \in U$ independently with probability $\gamma x_{u,v}$ for some $\gamma \in (0, 1-\varepsilon)$ to be chosen later, and assigns v to u only if u has enough budget remaining. Otherwise, if the algorithm samples no advertisers or the sampled u has insufficient budget, the algorithm does nothing and receives the next impression.

Observe that we have

$$\begin{aligned}
\mathbb{E}[\mathsf{ALG}_{\mathsf{int}}] &= \sum_{u \in U} \sum_{v \in V} b_{u,v} \Pr[\, v \text{ is assigned to } u \,] \\
&= \sum_{u \in U} \sum_{v \in V} b_{u,v} \cdot \left[\gamma x_{u,v} \cdot \Pr[\, u \text{ has enough budget to pay } b_{u,v} \text{ at } v\text{'s arrival} \,]\right] \\
&\geq \sum_{u \in U} \sum_{v \in V} b_{u,v} x_{u,v} \cdot \gamma \Pr[\, u \text{ has enough budget to pay } b_{u,v} \text{ at the end} \,] \\
&\geq \mathsf{ALG}_{\mathsf{frac}} \cdot \gamma \min_{u \in U} \Pr\left[u \text{ has enough budget to pay } \max_{v \in V} b_{u,v} \text{ at the end} \right].
\end{aligned} \tag{15}$$

We now claim that, for any advertiser $u \in U$,

$$\Pr\left[u \text{ has enough budget to pay } \max_{v \in V} b_{u,v} \text{ at the end} \right] \geq 1 - \exp\left(-\frac{(1-\gamma-\varepsilon)^2}{2\varepsilon} \right).$$

To this end, let us fix $u \in U$ and define

$$X_{u,v} := \mathbb{I}\{\, u \text{ is sampled at } v\text{'s arrival}\}$$

for every impression $v \in V$. Notice that $X_{u,v}$ is Bernoulli with mean $\gamma x_{u,v}$, and $\{X_{u,v}\}_{v \in V}$ are mutually independent. Moreover, since the fractional algorithm outputs a feasible fractional assignment $x \in \mathbb{R}^{U \times V}$, we have

$$\mathbb{E}\left[\sum_{v \in V} b_{u,v} X_{u,v}\right] = \gamma \sum_{v \in V} b_{u,v} x_{u,v} \leq \gamma B_u.$$

Let us further define, for every $v \in V$,

$$Y_v := \frac{b_{u,v} X_{u,v}}{B_u} \in [0, \varepsilon] \ \text{ and } \ Z_v := Y_v - \mathbb{E}[Y_v].$$

Observe that $\mathbb{E}[\sum_{v \in V} Y_v] \leq \gamma$, $\mathbb{E}[Z_v] = 0$, $|Z_v| \leq \varepsilon$, and $\mathrm{Var}(Z_v) = \mathrm{Var}(Y_v) = \frac{b_{u,v}^2}{B_u^2} \mathrm{Var}(X_{u,v})$. Moreover, note that $\{Z_v\}_{v \in V}$ are also mutually independent. We can therefore derive that

$$\mathrm{Var}\left(\sum_{v \in V} Z_v\right) = \sum_{v \in V} \frac{b_{u,v}^2}{B_u^2} \mathrm{Var}(X_{u,v}) \overset{(a)}{\leq} \sum_{v \in V} \frac{b_{u,v}^2}{B_u^2} \mathbb{E}[X_{u,v}] \overset{(b)}{\leq} \varepsilon\gamma \sum_{v \in V} \frac{b_{u,v} x_{u,v}}{B_u} \leq \varepsilon\gamma, \tag{16}$$

where (a) follows from the fact that $X_{u,v}$ is Bernoulli and (b) from the small bids assumption. We can then show the claim because

$$\begin{aligned}\Pr\left[u \text{ has enough budget to pay } \max_{v\in V} b_{u,v} \text{ at the end}\right] &\overset{(a)}{\geq} \Pr\left[\sum_{v\in V} Y_v \leq 1-\varepsilon\right] \\ &\overset{(b)}{\geq} \Pr\left[\sum_{v\in V} Z_v \leq 1-\gamma-\varepsilon\right] \\ &\overset{(c)}{\geq} 1-\exp\left(-\frac{(1-\gamma-\varepsilon)^2}{2\varepsilon\gamma+\frac{2}{3}\varepsilon(1-\gamma-\varepsilon)}\right) \\ &\geq 1-\exp\left(-\frac{(1-\gamma-\varepsilon)^2}{2\varepsilon}\right),\end{aligned}$$

where (a) follows from the small bids assumption, (b) from the fact that $\mathbb{E}[\sum_{v\in V} Y_v] \leq \gamma$, and (c) from Proposition 32 together with Eq. (16) and that $1-\gamma-\varepsilon > 0$.

Substituting this lower bound into Eq. (15), we have

$$\mathbb{E}[\mathsf{ALG}_{\mathsf{int}}] \geq \mathsf{ALG}_{\mathsf{frac}} \cdot \gamma \left(1-\exp\left(-\frac{(1-\gamma-\varepsilon)^2}{2\varepsilon}\right)\right).$$

Hence, for sufficiently small ε, choosing $\gamma := 1-\varepsilon-\sqrt{\varepsilon\ln(1/\varepsilon)} \in (0, 1-\varepsilon)$ gives

$$\begin{aligned}\frac{\mathbb{E}[\mathsf{ALG}_{\mathsf{int}}]}{\mathsf{ALG}_{\mathsf{frac}}} &\geq \left(1-\varepsilon-\sqrt{\varepsilon\ln(1/\varepsilon)}\right)\left(1-\exp\left(-\frac{1}{2}\ln\frac{1}{\varepsilon}\right)\right) \\ &= \left(1-\varepsilon-\sqrt{\varepsilon\ln(1/\varepsilon)}\right)\left(1-\sqrt{\varepsilon}\right) \\ &\geq 1-\varepsilon-\sqrt{\varepsilon}-\sqrt{\varepsilon\ln(1/\varepsilon)} \\ &\geq 1-3\sqrt{\varepsilon\ln(1/\varepsilon)}.\end{aligned}$$

□

4 Unweighted matching with integral advice

In this section, we introduce and analyze a new algorithm tailored to the unweighted setting with integral advice, which we call PUSHANDWATERFILL (PAW). To motivate why we need a new algorithm, it is worth noting that our theoretical guarantees of LAB are dominated by COINFLIP in the unweighted setting; see Fig. 1. This suggests that our previous analysis is not tight for unweighted instances. However, since that analysis was independent of the vertex weights, we find it challenging to improve it for the unweighted setting, even when we are given integral advice. As such, we propose PAW for the setting of unweighted matching with integral advice. In the following, we assume the advice is integral and represent it as a function $A : V \to U \cup \{\perp\}$, where $A(v)$ is the advised match for $v \in V$, and $A(v) = \perp$ indicates that v is advised to remain unmatched. Detailed pseudocode is given in Appendix A and a full analysis is provided in the supplementary material.

Algorithm description. As before, we describe PAW as a continuous-time process. Define the level of an offline vertex $u \in U$ as the total amount of water it has received so far. Upon arrival of online $v \in V$, with neighborhood $N(v)$ and advice $A(v)$, the algorithm proceeds in two phases:

Phase 1 (Push): Push flow into $A(v)$ until its level reaches λ.

Phase 2 (Waterfill): Distribute any remaining flow from v across $N(v)$ via the standard waterfilling.

4.1 Primal-dual analysis

We now analyze the performance of PAW, showing Theorem 3 restated below.

Theorem 3. *For any tradeoff parameter $\lambda \in [0, 1]$, PUSHANDWATERFILL is $r(\lambda)$-robust and $c(\lambda)$-consistent for unweighted online bipartite fractional matching with integral advice, where*

$$r(\lambda) := 1-\left(1-\lambda+{}^{\lambda^2}\!/_2\right)e^{\lambda-1} \quad \text{and} \quad c(\lambda) := 1-(1-\lambda)\,e^{\lambda-1}.$$

As in the analysis of LAB, we use a primal-dual framework to characterize the robustness and consistency of PAW. To begin, we adapt Lemma 5 to the unweighted setting:

Lemma 34 (cf. Lemma 5)**.** *Let $x \in \mathbb{R}_+^E$ be the algorithm's output. For some $\rho \in [0,1]$, if there exists $(\alpha, \beta) \in \mathbb{R}_+^U \times \mathbb{R}_+^V$ satisfying*

1. *(reverse weak duality) $\sum_{e \in E} x_e \geq \sum_{u \in U} \alpha_u + \sum_{v \in V} \beta_v$, and*
2. *(approximate dual feasibility) for every $(u, v) \in E$, $\alpha_u + \beta_v \geq \rho$,*

we have $\mathsf{ALG} \geq \rho \cdot \mathsf{OPT}$.

The dual variable construction differs from the vertex-weighted case and relies on a continuous and non-decreasing function $g : [0,1] \to [0,1]$ such that $g(1) = 1$. We call such a function a *splitting function*.

The dual variables (α, β) are initialized to zero, and are updated as follows. When an online vertex v sends an infinitesimal amount dz of flow to a neighbor $u \in N(v)$ whose current level is d_u, split this dz into $g(d_u)\, dz$ and $(1 - g(d_u))\, dz$. Then, we increase α_u by $g(d_u)\, dz$ and β_v by $(1 - g(d_u))\, dz$.

Since $g(d_u) \in [0, 1]$, both α and β remain nonnegative. Moreover, by construction, the reverse weak duality in Lemma 34 holds with equality: every infinitesimal unit of flow is split *exactly* into two values contributing to α_u and β_v, respectively.

Lemma 35. *For any splitting function g, the constructed dual variables satisfy the reverse weak duality of Lemma 34 with equality.*

Therefore, to analyze the robustness and consistency of the algorithm, it suffices to identify suitable splitting functions g that ensure approximate dual feasibility, with the goal of maximizing the parameter ρ in Lemma 34 for robustness and consistency, respectively.

We remark that the choice of splitting function g does not affect the behavior of PAW, in contrast to the vertex-weighted setting where the choice of penalty function f directly influences LAB's execution.

4.2 Robustness analysis

In this subsection, we prove that PAW is $r(\lambda)$-robust where $r(\lambda) := 1 - \left(1 - \lambda + \frac{\lambda^2}{2}\right) \cdot e^{\lambda - 1}$. To this end, we will prove the following lemma; observe that this lemma immediately implies the robustness result due to Lemmas 34 and 35.

Lemma 36. *There exists a splitting function g such that the resulting dual solution (α, β) satisfies $\alpha_u + \beta_v \geq r(\lambda)$ for every $(u, v) \in E$.*

We begin by giving a lower bound on β_v for every $v \in V$.

Lemma 37. *Fix an online vertex $v \in V$, and let ℓ be the minimum level of a neighbor of v at the end of the iteration when v arrives. For any splitting function g, we have*

$$\beta_v \geq \begin{cases} \displaystyle\int_{\ell}^{\lambda} (1 - g(z))\, dz + (1 - \lambda + \ell)(1 - g(\ell)), & \text{if } \ell \in [0, \lambda), \\ 1 - g(\ell), & \text{if } \ell \in [\lambda, 1]. \end{cases} \tag{17}$$

Proof. We break the proof into three cases.

Case 1. $\ell < 1$ and $A(v) \in N(v)$. Notice that v is saturated in this case, i.e., $\sum_{u \in N(v)} x_{u,v} = 1$. Let d be the level of $A(v)$ at the beginning of v's iteration. Recall that, in Phase 1, the algorithm pushes $\tau := (\lambda - d)_+ = \lambda - \min(d, \lambda)$ units along $(A(v), v)$; in Phase 2, the algorithm distributes the remaining $1 - \tau$ units to its neighborhood $N(v)$ in the waterfilling manner. We thus deduce that

$$\beta_v \geq \int_{\min(d,\lambda)}^{\lambda} (1 - g(z))\, dz + (1 - \lambda + \min(d, \lambda))(1 - g(\ell)). \tag{18}$$

If $\ell < \lambda$, the right-hand side is further bounded by

$$
\begin{aligned}
(\text{RHS of Eq. (18)}) &= \int_{\ell}^{\lambda} (1-g(z))\, dz + (1-\lambda+\ell)(1-g(\ell)) \\
&\quad + \int_{\min(d,\lambda)}^{\ell} (1-g(z))\, dz - (\ell - \min(d,\lambda))(1-g(\ell)) \\
&= \int_{\ell}^{\lambda} (1-g(z))\, dz + (1-\lambda+\ell)(1-g(\ell)) + \int_{\min(d,\lambda)}^{\ell} (g(\ell)-g(z))\, dz \\
&\geq \int_{\ell}^{\lambda} (1-g(z))\, dz + (1-\lambda+\ell)(1-g(\ell)), \qquad (19)
\end{aligned}
$$

where the inequality is satisfied due to the fact that, no matter whether $\ell \geq \min\{d,\lambda\}$ or $\ell < \min\{d,\lambda\}$, we have $\int_{\min(d,\lambda)}^{\ell}(g(\ell)-g(z))dz \geq 0$ since g is non-decreasing.

On the other hand, if $\ell \geq \lambda$, we have

$$
\begin{aligned}
(\text{RHS of Eq. (18)}) &= 1-g(\ell) + \int_{\min(d,\lambda)}^{\lambda} (1-g(z))\, dz - (\lambda - \min(d,\lambda))(1-g(\ell)) \\
&= 1-g(\ell) + \int_{\min(d,\lambda)}^{\lambda} (g(\ell)-g(z))\, dz \\
&\geq 1-g(\ell), \qquad (20)
\end{aligned}
$$

where the inequality is again due to that g is non-decreasing. This completes the proof for this case.

Case 2. $\ell < 1$ and $A(v) = \perp$. Observe that the algorithm does nothing in Phase 1 while it distributes 1 unit to its neighbor $N(v)$. We thus have

$$\beta_v \geq 1 - g(\ell),$$

which is equivalent to Equation (18) with $d = \lambda$. Hence, Equations (19) and (20) also follow for this case.

Case 3. $\ell = 1$. In this case, we have the following trivial bound that

$$\beta_v \geq 0 = 1 - g(\ell),$$

where the equality holds since $g(1) = 1$. $\square$

Let us define a splitting function g_r for robustness as follows:

$$
g_r(z) := \begin{cases} e^{\lambda-1}(z+1-\lambda), & \forall z \in [0,\lambda), \\ e^{z-1}, & \forall z \in [\lambda, 1]. \end{cases}
$$

Observe that g_r is indeed a splitting function, i.e., g_r is continuous and non-decreasing on $[0,1]$ with $g(0) \geq 0$ and $g(1) = 1$. Notice also that g_r is differentiable. Following is a technical lemma that will be used in the proof of Lemma 36.

Lemma 38. *Let $h : [0,1] \to [0,1]$ be a function defined as*

$$
h(\ell) := \begin{cases} \displaystyle\int_0^{\ell} g_r(z)dz + \int_{\ell}^{\lambda} (1-g_r(z))dz + (1-\lambda+\ell)(1-g_r(\ell)), & \text{if } \ell \in [0,\lambda), \\ \displaystyle\int_0^{\ell} g_r(z)dz + (1-g_r(\ell)), & \text{if } \ell \in [\lambda, 1]. \end{cases}
$$

Then, h is a constant function of value $r(\lambda) = 1 - \left(1 - \lambda + \frac{\lambda^2}{2}\right) \cdot e^{\lambda-1}$.

Proof. Note that the derivative of h is

$$
h'(\ell) = \begin{cases} g_r(\ell) - g_r'(\ell)(1-\lambda+\ell), & \text{if } \ell \in (0,\lambda), \\ g_r(\ell) - g_r'(\ell), & \text{if } \ell \in (\lambda, 1). \end{cases}
$$

By the definition of g_r, we have that $h'(\ell) = 0$ for all $\ell \in [0, 1]$. (Indeed, g_r was defined to make this true.) Therefore, h is a constant function. The value of this constant is equal to

$$h(\lambda) = \int_0^\lambda g_r(z)dz + (1 - g_r(\lambda)) = 1 - \left(1 - \lambda + \frac{\lambda^2}{2}\right) e^{\lambda - 1} = r(\lambda).$$

□

We are now ready to prove Lemma 36.

Proof of Lemma 36. Fix an edge $(u, v) \in E$, and let ℓ denote the minimum level of a neighbor of v at the end of the iteration of v. For any splitting function g, since α_u never decreases throughout the execution due to the definition of g, we have

$$\alpha_u \geq \int_0^\ell g(z)\, dz.$$

Together with Lemma 37, we can derive

$$\alpha_u + \beta_v \geq \begin{cases} \int_0^\ell g(z)dz + \int_\ell^\lambda (1 - g(z))dz + (1 - \lambda + \ell)(1 - g(\ell)), & \text{if } \ell \in [0, \lambda), \\ \int_0^\ell g(z)dz + (1 - g(\ell)), & \text{if } \ell \in [\lambda, 1]. \end{cases}$$

The proof of this lemma then immediately follows from Lemma 38 by choosing the splitting function $g := g_r$. □

4.3 Consistency analysis

We now show that PAW is $c(\lambda)$-consistent where $c(\lambda) := 1 - (1 - \lambda) \cdot e^{\lambda - 1}$. As in the previous robustness analysis, we will provide a good splitting function g that satisfies the approximate dual feasibility with $c(\lambda)$. However, contrary to the previous analysis, for the consistency, it suffices to have a *relaxed* notion of the approximate dual feasibility, formally stated as follows:

Lemma 39 (cf. Lemma 34)**.** *Let $x \in \mathbb{R}_+^E$ be the algorithm's output and $A \subseteq E$ be an integral advice. For some $\rho \in [0, 1]$, if there exists $(\alpha, \beta) \in \mathbb{R}_+^U \times \mathbb{R}_+^V$ satisfying*

- *(reverse weak duality) $\sum_{e \in E} x_e \geq \sum_{u \in U} \alpha_u + \sum_{v \in V} \beta_v$, and*
- *(relaxed approximate dual feasibility) for every $(u, v) \in A$, $\alpha_u + \beta_v \geq \rho$,*

we have $\mathsf{ALG} \geq \rho \cdot \mathsf{ADVICE}$.

Proof. Observe that

$$\mathsf{ALG} \geq \sum_{u \in U} \alpha_u + \sum_{v \in V} \beta_v \geq \sum_{(u,v) \in A} (\alpha_u + \beta_v) \geq \rho \cdot \mathsf{ADVICE},$$

where the second inequality is due to the fact that A is a matching. □

It thus suffices to prove the following lemma.

Lemma 40. *There exists a splitting function g such that resulting dual solution (α, β) satisfies $\alpha_u + \beta_v \geq c(\lambda)$ for any $(u, v) \in A$.*

We define a splitting function g_c as follows:

$$g_c(z) := \begin{cases} e^{\lambda - 1}, & \forall z \in [0, \lambda), \\ e^{z - 1}, & \forall z \in [\lambda, 1]. \end{cases}$$

It is easy to observe that g_c satisfies the conditions of a splitting function. We also remark that g_c is differentiable on $(0, \lambda) \cup (\lambda, 1)$. We need the following technical lemma in the proof of Lemma 40.

Lemma 41. *For any $\ell \in [0,1]$, we have*

$$\int_0^{\max(\ell,\lambda)} g_c(z)dz + (1 - g_c(\ell)) \geq c(\lambda).$$

Proof. Since g_c is non-decreasing, for any $\ell \in [0,\lambda)$, the left-hand side is bounded from below by

$$\int_0^{\max(\ell,\lambda)} g_c(z)dz + (1 - g_c(\ell)) \geq \int_0^{\lambda} g_c(z)dz + (1 - g_c(\lambda)).$$

Therefore, the infimum of the left-hand side is attained at $\ell \in [\lambda, 1]$. Let us denote $h(\ell) := \int_0^{\ell} g_c(z)dz + (1 - g_c(\ell))$, so that the left-hand side is equal to $h(\ell)$ on $[\lambda, 1]$. Note that by our choice of g_c, for any $\ell \in (\lambda, 1)$ we have

$$h'(\ell) = g(\ell) - g'(\ell) = e^{\ell-1} - e^{\ell-1} = 0.$$

Therefore, h is a constant function on $[\lambda, 1]$. The value of this constant is equal to

$$h(1) = \int_0^1 g(z)dz + (1 - g(1)) = 1 - (1-\lambda) \cdot e^{\lambda-1} = c(\lambda).$$

□

Proof of Lemma 40. Fix an edge $(u, v) \in A$, and let ℓ be the minimum level of a neighbor of v at the end of the iteration of v. For any splitting function g, we have

$$\beta_v \geq 1 - g(\ell)$$

due to the monotonicity of g. On the other hand, since u is matched by the advice A, the level of u must be at least $\max(\ell, \lambda)$ due to Phase 1 of PAW at this iteration, implying that

$$\alpha_u \geq \int_0^{\max(\ell,\lambda)} g(z)dz.$$

Therefore, choosing the splitting function $g := g_c$ immediately proves this lemma due to Lemma 41. □

5 Upper bound on robustness-consistency tradeoff

In this section, we present an upper bound result for the unweighted setting with integral advice. This upper bound also applies to vertex-weighted matching and AdWords with fractional advice. In Section 5.1, we define two adversaries, $\mathcal{R}$ and $\mathcal{C}$, which target robustness and consistency, respectively, against any fractional matching algorithm $\mathcal{M}$. Then, in Section 5.2, we formulate a factor-revealing linear program (LP) that provides an upper bound on the best possible consistency value c achievable against $\mathcal{C}$, subject to maintaining a robustness guarantee r against $\mathcal{R}$. The LP is constructed under a few assumptions about an algorithm's execution, which we later show in Section 5.3 to be without loss of generality.

5.1 Description of adversaries

We define two adversaries, $\mathcal{R}$ and $\mathcal{C}$, which target robustness and consistency, respectively, against any fractional matching algorithm $\mathcal{M}$. For a given integer $n \in \mathbb{Z}_+$, both adversaries construct a bipartite instance with a set U of $2n$ offline vertices and a set V of $2n$ online vertices. See Figure 6 for an illustration of the upper bound instance.

The two adversaries behave identically during the first n iterations, as follows: In the first iteration ($t = 1$), they present the first online vertex v_1 to $\mathcal{M}$, with v_1 connected to all offline vertices in U. The advice $A(v_1)$ is chosen arbitrarily. For each subsequent iteration $t = 2, \ldots, n$, the adversary presents online vertex v_t, which is adjacent to the neighbors $N(v_{t-1})$ of the previous vertex v_{t-1}, excluding two vertices: the previous advice $A(v_{t-1})$ and the offline vertex that has been filled the least so far by $\mathcal{M}$.

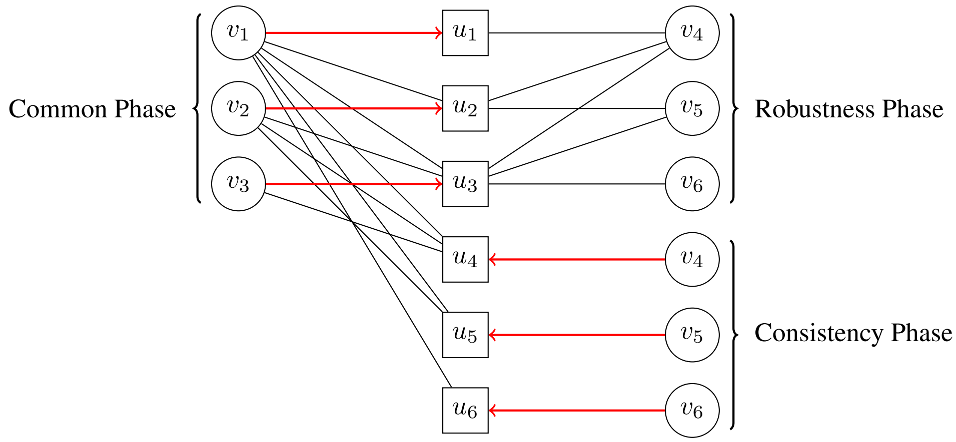


Figure 6: An illustration of the hardness construction. The instance begins with the common phase, which is the same in both adversaries. After the common phase, the instance can proceed in one of two ways, designed to be hard for robustness or consistency respectively. The robustness phase consists of an upper-triangular graph on the offline vertices $\{u_1, u_2, \ldots, u_n\}$, while the consistency phase forms a perfect matching to the offline vertices $\{u_{n+1}, u_{n+2}, \ldots, u_{2n}\}$.

Starting from iteration $t = n + 1$, the behaviors of the two adversaries diverge. The robustness adversary $\mathcal{R}$ continues on the vertices advised to be matched so far as in the classical setting of online fractional bipartite matching without advice: each online vertex is adjacent to the same neighbors as the preceding one, except for the offline vertex that has been filled the least so far by $\mathcal{M}$. In contrast, the consistency adversary $\mathcal{C}$ simply presents a matching to the offline vertices that were advised to be unmatched in the first n iterations, allowing the algorithm to fully saturate them.

The pseudocodes for $\mathcal{R}$ and $\mathcal{C}$ are given in Algorithms 1 and 2, respectively. Note that in both $\mathcal{R}$ and $\mathcal{C}$, the size of the maximum matching in hindsight is $2n$. For simplicity of presentation, we also allow the adversaries to reorder the indices of offline vertices based on the behavior of $\mathcal{M}$ over time.

5.2 Factor-Revealing LP

We now formulate a factor-revealing LP that upper bounds the consistency ratio c of any algorithm $\mathcal{M}$ against $\mathcal{C}$ while ensuring the algorithm is r-robust against $\mathcal{R}$, for any $r \in [1/2, 1 - 1/e]$. To this end, we assume that $\mathcal{M}$ satisfies the following conditions:

1. (Greediness) $\mathcal{M}$ saturates each online vertex unless its neighbors are all saturated.
2. (Monotonicity) $\mathcal{M}$ never makes $\mathcal{R}$ and $\mathcal{C}$ reorder the indices of U (e.g., Lines 12, 14, and 23 in Algorithm 1 and Line 12 in Algorithm 2).
3. (Uniformity) In the common phase, $\mathcal{M}$ pushes the same amount to the neighbors except the advised offline vertex at each iteration. In other words, for any $t \in \{1, \ldots, n\}$, the value x_{u,v_t} is the same for all $u \in N(v_t)$ with $u \neq A(v_t)$.

In Section 5.3, we argue that $\mathcal{M}$ satisfies these conditions without loss of generality.

Lemma 42. *For any algorithm $\mathcal{M}$, there exists an algorithm $\overline{\mathcal{M}}$ satisfying the above conditions (greediness, monotonicity, and uniformity), such that under both $\mathcal{R}$ and $\mathcal{C}$, the value of the matching returned by $\overline{\mathcal{M}}$ is at least that of $\mathcal{M}$,*

We now write the factor-revealing LP assuming these conditions are satisfied. For each iteration $t = 1, \ldots, n$ in the common phase, let

$$x_t := x_{u_t,v_t} \geq 0$$

denote the amount of water pushed by $\mathcal{M}$ to the suggested vertex $A(v_t) = u_t$, and let

$$\overline{x}_t := x_{u_{t+1},v_t} = \ldots = x_{u_{2n-t+1},v_t} \geq 0$$

Algorithm 1: Adversary $\mathcal{R}$ for robustness

Input: A fractional matching algorithm $\mathcal{M}$ and $n \in \mathbb{Z}_{\geq 1}$

Feed $U := \{u_1, \ldots, u_{2n}\}$ to $\mathcal{M}$
$d_u^{(0)} \leftarrow 0$ for every $u \in U$ // $d_u^{(t)}$ traces the level of u at the end of iteration t
// **Common phase**
for $t \leftarrow 1, \ldots, n$ **do**
    Feed v_t, $N(v_t) := \{u_t, u_{t+1}, \ldots, u_{2n-t+1}\}$, and $A(v_t) := u_t$ to $\mathcal{M}$
    Let $\{x_{u,v_t}\}_{u \in N(v_t)}$ be the output of $\mathcal{M}$
    for $u \in U$ **do**
        if $u \in N(v_t)$ **then**
            $d_u^{(t)} \leftarrow d_u^{(t-1)} + x_{u,v_t}$
        else
            $d_u^{(t)} \leftarrow d_u^{(t-1)}$
    Reorder $\{t+1, \ldots, 2n-t+1\}$ so that $d_{u_{t+1}}^{(t)} \geq \cdots \geq d_{u_{2n-t+1}}^{(t)}$
// **Robustness phase**
Reorder $\{1, \ldots, n\}$ so that $d_{u_1}^{(n)} \leq \cdots d_{u_n}^{(n)}$
for $t \leftarrow n+1, \ldots, 2n$ **do**
    Feed v_t, $N(v_t) := \{u_{t-n}, \ldots, u_n\}$, and $A(v_t) := \perp$ to $\mathcal{M}$
    Let $\{x_{u,v_t}\}_{u \in N(v_t)}$ be the output of $\mathcal{M}$
    for $u \in U$ **do**
        if $u \in N(v_t)$ **then**
            $d_u^{(t)} \leftarrow d_u^{(t-1)} + x_{u,v_t}$
        else
            $d_u^{(t)} \leftarrow d_u^{(t-1)}$
    Reorder $\{t-n, \ldots, n\}$ so that $d_{u_{t-n}}^{(t)} \leq \cdots \leq d_{u_n}^{(t)}$

Algorithm 2: Adversary $\mathcal{C}$ for consistency

Input: A fractional matching algorithm $\mathcal{M}$ and $n \in \mathbb{Z}_{\geq 1}$

Feed $U := \{u_1, \ldots, u_{2n}\}$ to $\mathcal{M}$
$d_u^{(0)} \leftarrow 0$ for every $u \in U$ // $d_u^{(t)}$ traces the level of u at the end of iteration t
// **Common phase**
for $t \leftarrow 1, \ldots, n$ **do**
    Feed v_t, $N(v_t) := \{u_t, u_{t+1}, \ldots, u_{2n-t+1}\}$, and $A(v_t) := u_t$ to $\mathcal{M}$
    Let $\{x_{u,v_t}\}_{u \in N(v_t)}$ be the output of $\mathcal{M}$
    for $u \in U$ **do**
        if $u \in N(v_t)$ **then**
            $d_u^{(t)} \leftarrow d_u^{(t-1)} + x_{u,v_t}$
        else
            $d_u^{(t)} \leftarrow d_u^{(t-1)}$
    Reorder $\{t+1, \ldots, 2n-t+1\}$ so that $d_{u_{t+1}}^{(t)} \geq \cdots \geq d_{u_{2n-t+1}}^{(t)}$
// **Consistency phase**
for $t \leftarrow n+1, \ldots, 2n$ **do**
    Feed v_t, $N(v_t) := \{u_t\}$, and $A(v_t) := u_t$ to $\mathcal{M}$
    Let $\{x_{u,v_t}\}_{u \in N(v_t)}$ be the output of $\mathcal{M}$
    for $u \in U$ **do**
        if $u \in N(v_t)$ **then**
            $d_u^{(t)} \leftarrow d_u^{(t-1)} + x_{u,v_t}$
        else
            $d_u^{(t)} \leftarrow d_u^{(t-1)}$

denote the amount pushed uniformly to the other neighbors $N(v_t) \setminus \{u_t\}$. Since $\mathcal{M}$ should output a fractional matching, we have

$$x_t + (2n - 2t + 1) \cdot \overline{x}_t \leq 1.$$

Let $d_t := d_{u_t}^{(t)}$ and $\overline{d}_t := d_{u_{2n-t+1}}^{(t)}$ denote the levels of u_t and u_{2n-t+1}, respectively, at the end of iteration t. Note that these levels remain unchanged until the end of the common phase by construction of our adversaries: both adversaries will never reveal edges adjacent to u_t and u_{2n-t+1} from iterations $t+1$ to n. Hence we have the constraints

$$d_t = \sum_{i=1}^{t-1} \overline{x}_i + x_t \leq 1 \text{ and } \overline{d}_t = \sum_{i=1}^{t} \overline{x}_i \leq 1,$$

from the capacity constraints of u_t and u_{2n-t+1} respectively. Since $\mathcal{M}$ is monotone, we also have

$$d_1 \leq d_2 \leq \ldots \leq d_n$$

due to Line 14 of $\mathcal{R}$.

Now consider the robustness adversary $\mathcal{R}$. For each $t \in \{1, \ldots, n\}$ and $i \in \{t, \ldots, n\}$, let $y_{i,t} := x_{u_i, v_{n+t}}$ be the amount pushed by $\mathcal{M}$ from v_{n+t} to u_i. Due to the degree constraint on v_{n+t}, we have

$$\sum_{i=t}^{n} y_{i,t} \leq 1.$$

Let $\ell_i^{(t)} := d_{u_i}^{(n+t)}$ be the level of u_i at the end of round $n+t$ against $\mathcal{R}$. Note that we have

$$\ell_i^{(t)} = d_i + \sum_{s=1}^{t} y_{i,s} \leq 1,$$

where the inequality is due to the degree constraint. Since $\mathcal{M}$ is monotone, we have

$$\ell_t^{(t)} \leq \ell_{t+1}^{(t)} \leq \ldots \leq \ell_n^{(t)}$$

due to Line 23 of $\mathcal{R}$.

Notice that, in order for $\mathcal{M}$ to have the robustness ratio of r, it should output a solution of size at least $2n \cdot r$ against $\mathcal{R}$, implying that we have

$$\sum_{t=1}^{n} (d_t + \overline{d}_t) + \sum_{t=1}^{n} \sum_{i=t}^{n} y_{i,t} \geq 2nr.$$

Under this situation, the solution quality output by $\mathcal{M}$ against $\mathcal{C}$ bounds from above the consistency ratio c of $\mathcal{M}$. We thus have

$$\sum_{t=1}^{n} d_t + n \geq 2nc.$$

Given a robustness r, our objective is to find the best-possible consistency c while satisfying all of the above constraints:

$$
\begin{aligned}
&\text{maximize } c \\
&\text{subject to } x_t + (2n - 2t + 1) \cdot \overline{x}_t \leq 1, && \forall t \in \{1, \ldots, n\}, \\
&\quad d_t = \textstyle\sum_{i=1}^{t-1} \overline{x}_i + x_t, && \forall t \in \{1, \ldots, n\}, \\
&\quad \overline{d}_t = \textstyle\sum_{i=1}^{t} \overline{x}_i, && \forall t \in \{1, \ldots, n\}, \\
&\quad d_t \leq d_{t+1}, && \forall t \in \{1, \ldots, n-1\}, \\
&\quad \textstyle\sum_{i=t}^{n} y_{i,t} \leq 1, && \forall t \in \{1, \ldots, n\}, \\
&\quad \ell_i^{(t)} = d_i + \textstyle\sum_{s=1}^{t} y_{i,s}, && \forall t \in \{1, \ldots, n\}, \forall i \in \{t, \ldots, n\}, \\
&\quad \ell_i^{(t)} \leq \ell_{i+1}^{(t)}, && \forall t \in \{1, \ldots, n\}, \forall i \in \{t, \ldots, n-1\}, \\
&\quad \textstyle\sum_{t=1}^{n} (d_t + \overline{d}_t) + \sum_{t=1}^{n} \sum_{i=t}^{n} y_{i,t} \geq 2nr, \\
&\quad \textstyle\sum_{t=1}^{n} d_t + n \geq 2nc, \\
&\quad 0 \leq x_t, \overline{x}_t, d_t, \overline{d}_t \leq 1, && \forall t \in \{1, \ldots, n\}, \\
&\quad 0 \leq y_{i,t}, \ell_i^{(t)} \leq 1, && \forall t \in \{1, \ldots, n\}, \forall i \in \{t, \ldots, n\}.
\end{aligned}
$$

We implemented this factor-revealing LP with PuLP[8] and computationally solved this LP using Gurobi 12.0.2 [Gur24] with $n = 1000$ and a finite set of values for robustness $r \in [0.5, 1 - 1/e]$. See Table 1 and Figure 1 for the computational results.

Table 1: Upper bound on the robustness-consistency tradeoff with $n = 1000$.

r	0.500	0.525	0.550	0.575	0.600	0.625	$1 - 1/e$
c	1.000	0.974	0.944	0.908	0.862	0.788	0.731

5.3 Proof of Lemma 42

This subsection is devoted to the proof of Lemma 42. In what follows, for any algorithm $\mathcal{M}$, let us denote by $\mathcal{M}(\mathcal{R})$ and $\mathcal{M}(\mathcal{C})$ the size of the output of $\mathcal{M}$ against $\mathcal{R}$ and $\mathcal{C}$, respectively.

Recall the conditions that we want to prove without loss of generality for an algorithm $\mathcal{M}$ to be Pareto-efficient:

1. (Greediness) $\mathcal{M}$ saturates each online vertex unless its neighbors are all saturated.
2. (Monotonicity) $\mathcal{M}$ never makes $\mathcal{R}$ and $\mathcal{C}$ reorder the indices of U.
3. (Uniformity) In the common phase, $\mathcal{M}$ pushes the same amount to the neighbors except the advised offline vertex at each iteration.

For greediness, it is folklore that this condition is without loss of generality for online bipartite matching (see, e.g., [KVV90]).

Let us now consider monotonicity. Observe that there are three places (Lines 12, 14, and 23) in Algorithm 1 and one place (Line 12) in Algorithm 2 where the adversaries may reorder the indices of U. Let us define the following.

- For every $t \in \{1, \ldots, n-1\}$, we say an algorithm is *monotone at iteration* t if $\mathcal{R}$ (and $\mathcal{C}$) does not reorder the indices at Line 12 (and Line 12, respectively) in iteration t of the common phase, i.e., for $d^{(t-1)}_{u_{t+1}} \geq \cdots \geq d^{(t-1)}_{u_{2n-t+1}}$, we also have $d^{(t)}_{u_{t+1}} \geq \cdots \geq d^{(t)}_{u_{2n-t+1}}$. Note that, in iteration n, the adversary does not reorder the indices at that line since it has only one index $n+1$.
- We say an algorithm is *monotone at iteration* n if $\mathcal{R}$ does not reorder the indices at Line 14.
- For every $t \in \{n+1, \ldots, 2n-1\}$, we say an algorithm is *monotone at iteration* t if $\mathcal{R}$ does not reorder the indices at Line 23 in iteration t of the robustness phase, i.e., for $d^{(t-1)}_{u_{t-n}} \leq \cdots \leq d^{(t-1)}_{u_n}$, we also have $d^{(t)}_{u_{t-n}} \leq \cdots \leq d^{(t)}_{u_n}$. Note that, in iteration $2n$, the adversary has only $\{n\}$ at that line.

The following lemma implies that the monotonicity of $\mathcal{M}$ in the common phase can be assumed without loss of generality.

Lemma 43. *For any algorithm $\mathcal{M}$, there exists an algorithm $\overline{\mathcal{M}}$ which is monotone at every iteration $t \in \{1, \ldots, n-1\}$ while satisfying $\overline{\mathcal{M}}(\mathcal{R}) = \mathcal{M}(\mathcal{R})$ and $\overline{\mathcal{M}}(\mathcal{C}) = \mathcal{M}(\mathcal{C})$.*

Proof. We inductively prove the lemma. Let t be the first iteration where $\mathcal{M}$ reorders the indices at Line 12 of $\mathcal{R}$ (and Line 12 of $\mathcal{C}$). Let $\sigma : \{t+1, \ldots, 2n-t+1\} \to \{t+1, \ldots, 2n-t+1\}$ be the permutation such that $d^{(t)}_{u_{\sigma(t+1)}} \geq \ldots \geq d^{(t)}_{u_{\sigma(2n-t+1)}}$ right before the adversary reorders the indices. Since we have $d^{(t-1)}_{u_{t+1}} \geq \ldots \geq d^{(t-1)}_{u_{2n-t+1}}$ due to the induction hypothesis from the previous iteration, we can derive that $d^{(t)}_{u_{\sigma(i)}} \geq d^{(t-1)}_{u_i}$ for every $i \in \{t+1, \ldots, 2n-t+1\}$. Let us now consider another algorithm $\overline{\mathcal{M}}$, whose output is denoted by $\overline{x}$, such that, up to iteration $t-1$, $\overline{\mathcal{M}}$ behaves the same as $\mathcal{M}$, but in iteration t, $\overline{\mathcal{M}}$ pushes $\overline{x}_{u_i,v_t}$ towards $u_i \in N(v_t)$ for every $i \in \{t+1, \ldots, 2n-t+1\}$, where

$$\overline{x}_{u_i,v_t} := \begin{cases} d^{(t)}_{u_t} - d^{(t-1)}_{u_t}, & \text{if } i = t, \\ d^{(t)}_{u_{\sigma(i)}} - d^{(t-1)}_{u_i}, & \text{otherwise.} \end{cases}$$

[8] https://github.com/coin-or/pulp

Observe that the level $\overline{d}_{u_i}^{(t)}$ of $u_i \in N(v_t)$ at this moment in the execution of $\overline{\mathcal{M}}$ is

$$\overline{d}_{u_i}^{(t)} := \begin{cases} d_{u_t}^{(t)}, & \text{if } i = t, \\ d_{u_{\sigma(i)}}^{(t)}, & \text{otherwise.} \end{cases}$$

Therefore, $\overline{\mathcal{M}}$ satisfies $\overline{d}_{u_{t+1}}^{(t)} \geq \cdots \geq \overline{d}_{u_{2n-t+1}}^{(t)}$, implying that the adversary would not reorder the indices of $N(v_t) \setminus \{u_t\}$ against $\overline{\mathcal{M}}$. Moreover, $\mathcal{M}$ and $\overline{\mathcal{M}}$ have the same configuration of levels of $N(v_t)$ at the end of iteration t, and hence, by letting $\overline{\mathcal{M}}$ behave the same as $\mathcal{M}$ until termination, we can see that both $\overline{\mathcal{M}}$ and $\mathcal{M}$ eventually return solutions of the same size against the adversary. □

By a similar argument, we can also assume without loss of generality the monotonicity of $\mathcal{M}$ against $\mathcal{R}$ in each iteration $t \in \{n+1, \ldots, 2n-1\}$ of the robustness phase. We omit the proof of the next lemma.

Lemma 44. *For any algorithm $\mathcal{M}$, there exists an algorithm $\overline{\mathcal{M}}$ which is monotone at every iteration $t \in \{n+1, \ldots, 2n-1\}$ while satisfying $\overline{\mathcal{M}}(\mathcal{R}) = \mathcal{M}(\mathcal{R})$.*

What remains to show is generality of monotonicity at iteration n and uniformity. We first prove that it is no loss of generality to assume the uniformity of the algorithm, and then show that, given the algorithm is uniform, we can further assume without loss of generality that the algorithm is monotone at iteration n.

Due to Lemma 43, let us from now consider any algorithm $\mathcal{M}$ that is greedy and monotone in the common phase. Due to the monotonicity, let us slightly abuse the notation and write $d_i^{(t)} := d_{u_i}^{(t)}$ for any $t \in \{0, \ldots, n\}$ and $i \in \{1, \ldots, 2n\}$ for simplicity.

In what follows, we will prove the generality by inductively modifying any algorithm $\mathcal{M}$ into another algorithm $\overline{\mathcal{M}}$. To this end, let us first define some notation related to $\overline{\mathcal{M}}$. Let us denote by $\overline{x}$ the solution of $\overline{\mathcal{M}}$. For each $t \in \{0, \ldots, n\}$ and $i \in \{1, \ldots, 2n\}$, let us denote by $\overline{d}_i^{(t)}$ the level of $u_i \in L$ at the end of iteration t in the execution of $\overline{\mathcal{M}}$ against the adversaries in the common phase (i.e., the value of $d_{u_i}^{(t)}$ in Algorithms 1 and 2).

We newly define the notation only for the common phase due to the following lemma.

Lemma 45. *Given any greedy algorithm $\mathcal{M}$, suppose we construct another greedy algorithm $\overline{\mathcal{M}}$ and define the execution of $\overline{\mathcal{M}}$ in the common phase to satisfy the following conditions:*

(A) the sum of levels of $\{u_1, \ldots, u_n\}$ is the same in both $\overline{\mathcal{M}}$ and $\mathcal{M}$, i.e., $\sum_{i=1}^{n} \overline{d}_i^{(n)} = \sum_{i=1}^{n} d_i^{(n)}$; and

(B) in each iteration $t \in \{1, \ldots, n\}$, $\overline{\mathcal{M}}$ pushes as much amount of water as $\mathcal{M}$, i.e., $\sum_{u \in N(v_t)} \overline{x}_{u,v_t} \geq \sum_{u \in N(v_t)} x_{u,v_t}$.

We can then complete the execution of $\overline{\mathcal{M}}$ in the following iterations to satisfy $\overline{\mathcal{M}}(\mathcal{R}) \geq \mathcal{M}(\mathcal{R})$ and $\overline{\mathcal{M}}(\mathcal{C}) = \mathcal{M}(\mathcal{C})$.

Proof. Due to Condition (B), the total amount pushed by $\overline{\mathcal{M}}$ throughout the common phase is no less than that by $\mathcal{M}$, implying that we have $\sum_{i=1}^{2n} \overline{d}_i^{(n)} \geq \sum_{i=1}^{2n} d_i^{(n)}$. From Condition (A), we can also derive $\sum_{i=n+1}^{2n} \overline{d}_i^{(n)} \geq \sum_{i=n+1}^{2n} d_i^{(n)}$. Therefore, against $\mathcal{R}$, by letting $\overline{\mathcal{M}}$ behave the same as $\mathcal{M}$ in the robustness phase, we can obtain

$$\overline{\mathcal{M}}(\mathcal{R}) = \sum_{i=1}^{n} d_{u_i}^{(2n)} + \sum_{i=n+1}^{2n} \overline{d}_i^{(n)} \geq \sum_{i=1}^{n} d_{u_i}^{(2n)} + \sum_{i=n+1}^{2n} d_i^{(n)} = \mathcal{M}(\mathcal{R}).$$

On the other hand, since any greedy algorithm would saturate $\{u_{n+1}, \ldots, u_{2n}\}$ against $\mathcal{C}$, it is easy to define the execution of $\overline{\mathcal{M}}$ in the consistency phase to satisfy

$$\overline{\mathcal{M}}(\mathcal{C}) = \sum_{i=1}^{n} \overline{d}_i^{(n)} + n = \sum_{i=1}^{n} d_i^{(n)} + n = \mathcal{M}(\mathcal{C}).$$

□

We are now ready to prove that we can assume without loss of generality that the algorithm is uniform.

Lemma 46. *Given any algorithm $\mathcal{M}$ that is greedy and monotone in the common phase, there exists an algorithm $\overline{\mathcal{M}}$ that is uniform as well while satisfying $\overline{\mathcal{M}}(\mathcal{R}) \geq \mathcal{M}(\mathcal{R})$ and $\overline{\mathcal{M}}(\mathcal{C}) = \mathcal{M}(\mathcal{C})$.*

Proof. For $t \in \{1, \ldots, n\}$, we say that an algorithm is *uniform at iteration* t if the algorithm pushes water uniformly towards $N(v_t) \setminus \{u_t\}$. Recall that the amount pushed towards $A(v_t) = u_t$ does not affect the uniformity of the algorithm at iteration t.

We prove the lemma by inductively showing that, for some $t^\star \in \{1, \ldots, n\}$, if $\mathcal{M}$ is monotone up to iteration $t^\star - 1$, we can construct $\overline{\mathcal{M}}$ that is monotone up to iteration $t^\star$ while the execution of $\overline{\mathcal{M}}$ in the common phase satisfies the conditions of Lemma 45. In particular, instead of Condition (A), we will consider a stronger condition:

(A′) in each iteration $t \in \{1, \ldots, n\}$, the level of u_t is equal in both $\overline{\mathcal{M}}$ and $\mathcal{M}$, i.e., $\overline{d}_t^{(t)} = d_t^{(t)}$.

We assume without loss of generality that $\mathcal{M}$ is *non-uniform* at iteration $t^\star$ for the first time; otherwise, $\overline{\mathcal{M}} = \mathcal{M}$ would immediately satisfy the conditions. Observe that, under this assumption, $\mathcal{M}$ saturates $v_{t^\star}$ in this iteration.

Let us now describe the execution of $\overline{\mathcal{M}}$ in the common phase. For iterations $t \in \{1, \ldots, t^\star - 1\}$, $\overline{\mathcal{M}}$ behaves the same as $\mathcal{M}$. In iteration $t^\star$, however, since we now want $\overline{\mathcal{M}}$ to be uniform at this iteration while guaranteeing the conditions of Lemma 45, $\overline{\mathcal{M}}$ pushes the same amount $x_{u_{t^\star}, v_{t^\star}}$ as $\mathcal{M}$ towards its advice $A(v_{t^\star}) = u_{t^\star}$ while it distributes the remaining $1 - x_{u_{t^\star}, v_{t^\star}}$ units uniformly to the other neighbors $N(v_{t^\star}) \setminus \{u_{t^\star}\}$, i.e., we have, for any $i \in \{t^\star, \ldots, 2n - t^\star + 1\}$,

$$\overline{x}_{u_i, v_{t^\star}} := \begin{cases} x_{u_{t^\star}, v_{t^\star}}, & \text{if } i = t^\star, \\ \frac{1 - x_{u_{t^\star}, v_{t^\star}}}{2n - 2t^\star + 1}, & \text{otherwise.} \end{cases}$$

In the subsequent iterations $t \in \{t^\star + 1, \ldots, n\}$, $\overline{\mathcal{M}}$ iterates $i \in \{t, \ldots, 2n - t + 1\}$ in this order and pushes water through (u_i, v_t) until the level of u_i reaches $d_i^{(t)}$ or v_t gets saturated. We will later argue that, for any neighbor $u_i \in N(v_t)$ that is filled a positive amount by $\overline{\mathcal{M}}$ in this iteration, it is guaranteed to have $d_i^{(t)} \geq \overline{d}_i^{(t-1)}$, implying that $\overline{\mathcal{M}}$ is well-defined.

It is trivial to see that $\overline{\mathcal{M}}$ is still greedy and monotone in the common phase. Note also that it is now uniform up to iteration $t^\star$. It remains to prove that the execution of $\overline{\mathcal{M}}$ in the common phase is indeed well-defined while satisfying Conditions (A′) and (B) as well. It is true up to iteration $t^\star - 1$ since the executions of $\mathcal{M}$ and $\overline{\mathcal{M}}$ are identical. Since every offline vertex in $\{u_1, \ldots, u_{t^\star - 1}\} \cup \{u_{2n - t^\star + 2}, \ldots, u_{2n}\}$ is not adjacent with any online vertex arriving after iteration $t^\star - 1$ in the common phase, we have

$$\overline{d}_i^{(t^\star - 1)} = \cdots = \overline{d}_i^{(n)} = d_i^{(t^\star - 1)} = \cdots = d_i^{(n)} \tag{21}$$

for any $i \in \{1, \ldots, t^\star - 1\} \cup \{2n - t^\star + 2, \ldots, 2n\}$.

For the remaining iterations in the common phase, recall that $\mathcal{M}$ saturates $v_{t^\star}$. Suppose $\mathcal{M}$ does not saturate v_n in the last iteration n of the phase. In this case, since $\mathcal{M}$ is greedy while $N(v_{t+1}) \subsetneq N(v_t)$ for every $t \in \{1, \ldots, n\}$ in the common phase, there must exist $t_s \in \{t^\star, \ldots, n - 1\}$ such that $\mathcal{M}$ saturates v_t in every iteration $t \leq t_s$ whereas all offline vertices in $N(v_{t_s + 1})$ become saturated at iteration $t_s + 1$; for any following iterations $t \geq t_s + 2$, $\mathcal{M}$ pushes nothing, i.e., $\sum_{u \in N(v_t)} x_{u, v_t} = 0$. For the other case when $\mathcal{M}$ saturates v_n in iteration n, let us say $t_s := n$.

Consider iteration $t^\star$. It is easy to see that the execution of $\overline{\mathcal{M}}$ in this iteration is well-defined and satisfies Conditions (A′) and (B). Let $\overline{d} = \overline{d}_{t^\star + 1}^{(t^\star)} = \cdots = \overline{d}_{2n - t^\star + 1}^{(t^\star)}$ denote the uniform level of the neighbors $N(v_{t^\star}) \setminus \{u_{t^\star}\}$ other than the advice $A(v_{t^\star}) = u_{t^\star}$ in $\overline{\mathcal{M}}$. Since $d_{t^\star + 1}^{(t^\star)} > \overline{d}$ and $d_{2n - t^\star + 1}^{(t^\star)} < \overline{d}$, there always exists $p^{(t^\star)}$ and $q^{(t^\star)}$ such that

- $t^\star \le p^{(t^\star)} < q^{(t^\star)} \le 2n - t^\star$;
- for every $i \in \{t^\star, \dots, p^{(t^\star)}\}$, the levels of u_i in $\overline{\mathcal{M}}$ and $\mathcal{M}$ are the same, i.e., $\overline{d}_i^{(t^\star)} = d_i^{(t^\star)}$;
- for every $i \in \{p^{(t^\star)} + 1, \dots, q^{(t^\star)}\}$, the level of u_i in $\overline{\mathcal{M}}$ is strictly less than that in $\mathcal{M}$, i.e., $\overline{d}_i^{(t^\star)} = \overline{d} < d_i^{(t^\star)}$; and
- for every $i \in \{q^{(t^\star)} + 1, \dots, 2n - t^\star + 1\}$, the level of u_i in $\overline{\mathcal{M}}$ is no less than that in $\mathcal{M}$, i.e., $\overline{d}_i^{(t^\star)} = \overline{d} \ge d_i^{(t^\star)}$.

In fact, for each subsequent iteration $t \in \{t^\star + 1, \dots, t_s\}$ where $\mathcal{M}$ saturates v_t, we inductively show that there exists $p^{(t)}$ and $q^{(t)}$ that satisfy the following properties:

(i) $t \le p^{(t)} < q^{(t)} \le 2n - t^\star$;

(ii) for every $i \in \{t^\star, \dots, p^{(t)}\}$, the levels of u_i in $\overline{\mathcal{M}}$ and $\mathcal{M}$ are identical, i.e., $\overline{d}_i^{(t)} = d_i^{(t)}$;

(iii) for every $i \in \{p^{(t)} + 1, \dots, q^{(t)}\}$, the level of u_i in $\overline{\mathcal{M}}$ is strictly less than that in $\mathcal{M}$, i.e., $\overline{d}_i^{(t)} < d_i^{(t)}$; and

(iv) for every $i \in \{q^{(t)} + 1, \dots, 2n - t^\star + 1\}$, the level of u_i in $\overline{\mathcal{M}}$ is at least that in $\mathcal{M}$, i.e., $\overline{d}_i^{(t)} \ge d_i^{(t)}$.

Note that Properties (i) and (ii) immediately imply Condition (A′). We will see the other propositions — $\overline{\mathcal{M}}$ is well-defined and satisfies Condition (B) — while we prove the existence of $p^{(t)}$ and $q^{(t)}$.

We break the analysis into the three cases as follows.

Case 1. $p^{(t-1)} \ge 2n - t + 1$**.** Observe that we have $\overline{d}_i^{(t-1)} = d_i^{(t-1)}$ for every $i \in \{t, \dots, 2n-t+1\}$ due to Property (ii) at the beginning of iteration t. We can therefore see that $\overline{\mathcal{M}}$ effectively behaves the same as $\mathcal{M}$ in this iteration, implying that $\overline{\mathcal{M}}$ is well-defined in this iteration. Note also that $\overline{\mathcal{M}}$ saturates v_t as $\mathcal{M}$ does in this iteration, implying that Condition (B) is satisfied. Moreover, since v_t is adjacent only with $N(v_t) = \{u_t, \dots, u_{2n-t+1}\}$, for every $i \in \{1, \dots, t-1\} \cup \{2n-t+2, \dots, 2n\}$, we have $\overline{d}_i^{(t)} = \overline{d}_i^{(t-1)}$ and $d_i^{(t)} = d_i^{(t-1)}$, implying that $p^{(t)} := p^{(t-1)}$ and $q^{(t)} := q^{(t-1)}$ would satisfy the properties at the end of iteration t.

Case 2. $p^{(t-1)} < 2n - t + 1 \le q^{(t-1)}$**.** Notice that, in this case, we have

$$\begin{aligned}
\sum_{i=t}^{2n-t+1} \left(d_i^{(t)} - \overline{d}_i^{(t-1)}\right) &= \sum_{i=t}^{2n-t+1} \left(d_i^{(t)} - d_i^{(t-1)}\right) + \sum_{i=t}^{2n-t+1} \left(d_i^{(t-1)} - \overline{d}_i^{(t-1)}\right) \\
&= 1 + \sum_{i=p^{(t-1)}+1}^{2n-t+1} \left(d_i^{(t-1)} - \overline{d}_i^{(t-1)}\right) \\
&> 1,
\end{aligned}$$

where the second equality follows from the fact that $\mathcal{M}$ saturates v_t. Moreover, since $d_t^{(t)} - \overline{d}_t^{(t-1)} \le 1$, we can deduce that there exists $p \in \{t, \dots, 2n-t\}$ such that, for every $i \in \{t, \dots, 2n - t + 1\}$,

$$\overline{x}_{u_i, v_t} = \begin{cases} d_i^{(t)} - \overline{d}_i^{(t-1)}, & \text{if } i \le p, \\ z, & \text{if } i = p+1, \\ 0, & \text{if } i \ge p+2, \end{cases} \text{ and hence } \overline{d}_i^{(t)} = \begin{cases} d_i^{(t)}, & \text{if } i \le p, \\ \overline{d}_i^{(t-1)} + z, & \text{if } i = p+1, \\ \overline{d}_i^{(t-1)}, & \text{if } i \ge p+2, \end{cases}$$

where $z := 1 - \sum_{i=t}^{p} \left(d_i^{(t)} - \overline{d}_i^{(t-1)}\right)$ denotes the remaining amount of v_t to be saturated in this iteration. We can thus see that $\overline{\mathcal{M}}$ is well-defined and satisfies Condition (B) in this iteration. Observe also that $p^{(t)} := p$ and $q^{(t)} := q^{(t-1)}$ would satisfy the properties at the end of this iteration.

Case 3. $p^{(t-1)} < q^{(t-1)} < 2n - t + 1$**.** Let q denote the index such that $d_q^{(t)} > \overline{d}_q^{(t-1)}$ and $d_{q+1}^{(t)} \leq \overline{d}_{q+1}^{(t-1)}$. Observe that q is unique and $q \in \{q^{(t-1)}, \ldots, 2n - t + 1\}$ due to the properties at iteration $t-1$ and the monotonicity of $\mathcal{M}$. We first show that $\overline{\mathcal{M}}$ would iterate up to q to push water in this iteration, i.e., $\sum_{i=t}^{q}\left(d_i^{(t)} - \overline{d}_i^{(t-1)}\right) > 1$, implying that $\overline{\mathcal{M}}$ is well-defined and satisfies Condition (B). Note that it suffices to show that

$$\sum_{i=t}^{q}\left(d_i^{(t-1)} - \overline{d}_i^{(t-1)}\right) > \sum_{i=q+1}^{2n-t+1}\left(d_i^{(t)} - d_i^{(t-1)}\right) \tag{22}$$

since, if the above inequality is indeed true, we can immediately derive

$$\begin{aligned}
\sum_{i=t}^{q}\left(d_i^{(t)} - \overline{d}_i^{(t-1)}\right) &= \sum_{i=t}^{q}\left(d_i^{(t)} - d_i^{(t-1)}\right) + \sum_{i=t}^{q}\left(d_i^{(t-1)} - \overline{d}_i^{(t-1)}\right) \\
&> \sum_{i=t}^{2n-t+1}\left(d_i^{(t)} - d_i^{(t-1)}\right) \\
&= 1.
\end{aligned}$$

Since both $\overline{\mathcal{M}}$ and $\mathcal{M}$ have so far saturated $\{v_1, \ldots, v_{t-1}\}$, we have $\sum_{i=1}^{2n} d_i^{(t-1)} = \sum_{i=1}^{2n} \overline{d}_i^{(t-1)}$, implying that

$$\begin{aligned}
0 &= \sum_{i=1}^{2n}\left(d_i^{(t-1)} - \overline{d}_i^{(t-1)}\right) \\
&= \sum_{i=t}^{2n-t^\star+1}\left(d_i^{(t-1)} - \overline{d}_i^{(t-1)}\right) \\
&= \sum_{i=t}^{q^{(t-1)}}\left(d_i^{(t-1)} - \overline{d}_i^{(t-1)}\right) - \sum_{i=q^{(t-1)}+1}^{2n-t^\star+1}\left(\overline{d}_i^{(t-1)} - d_i^{(t-1)}\right),
\end{aligned}$$

where the second equality follows from Equation (21) and the fact that $p^{(t-1)} \geq t-1$ due to Property (i). We can thus obtain

$$\begin{aligned}
\sum_{i=t}^{q}\left(d_i^{(t-1)} - \overline{d}_i^{(t-1)}\right) &= \sum_{i=t}^{q^{(t-1)}}\left(d_i^{(t-1)} - \overline{d}_i^{(t-1)}\right) - \sum_{i=q^{(t-1)}+1}^{q}\left(\overline{d}_i^{(t-1)} - d_i^{(t-1)}\right) \\
&= \sum_{i=q+1}^{2n-t^\star+1}\left(\overline{d}_i^{(t-1)} - d_i^{(t-1)}\right) \\
&= \sum_{i=q+1}^{2n-t+1}\left(\overline{d}_i^{(t-1)} - d_i^{(t-1)}\right) + \sum_{i=2n-t+2}^{2n-t^\star+1}\left(\overline{d}_i^{(t-1)} - d_i^{(t-1)}\right) \\
&\geq \sum_{i=q+1}^{2n-t+1}\left(\overline{d}_i^{(t-1)} - d_i^{(t-1)}\right) \\
&> \sum_{i=q+1}^{2n-t+1}\left(d_i^{(t)} - d_i^{(t-1)}\right),
\end{aligned}$$

where the first inequality is derived from the condition that $q^{(t-1)} < 2n - t + 1$ and the last from the definition of q, showing Equation (22). Together with the fact that $d_t^{(t)} - \overline{d}_t^{(t-1)} \leq 1$, we can also infer that there exists $p \in \{t, \ldots, q-1\}$ in the execution of $\overline{\mathcal{M}}$ such that, for every $i \in \{t, \ldots, 2n-t+1\}$,

$$\overline{x}_{u_i,v_t} = \begin{cases} d_i^{(t)} - \overline{d}_i^{(t-1)}, & \text{if } i \leq p, \\ z, & \text{if } i = p+1, \\ 0, & \text{if } i \geq p+2, \end{cases} \text{ and hence } \overline{d}_i^{(t)} = \begin{cases} d_i^{(t)}, & \text{if } i \leq p, \\ \overline{d}_i^{(t-1)} + z, & \text{if } i = p+1, \\ \overline{d}_i^{(t-1)}, & \text{if } i \geq p+2, \end{cases}$$

where we again denote by $z := 1 - \sum_{i=t}^{p} \left(d_i^{(t)} - \overline{d}_i^{(t-1)}\right)$ the remaining amount of v_t to be saturated in this iteration. Observe that $p^{(t)} := p$ and $q^{(t)} := q$ would satisfy the properties at the end of this iteration.

We have shown that, for every iteration $t \in \{1, \dots, t_s\}$ where $\mathcal{M}$ saturates v_t, $\overline{\mathcal{M}}$ also saturates v_t while $\overline{d}_t^{(t_s)} = d_t^{(t_s)}$. Therefore, if $t_s = n$, this immediately completes the proof of Lemma 46.

Suppose, on the other hand, that $t_s + 1 \le n$. In iteration $t_s + 1$, $\mathcal{M}$ saturates $N(v_{t_s+1})$, i.e., $d_i^{(t_s+1)} = 1$ for every $i \in \{t, \dots, 2n - t_s\}$. We claim that

$$\sum_{i=t_s+1}^{2n-t_s} \left(1 - \overline{d}_i^{(t_s)}\right) \ge \sum_{i=t_s+1}^{2n-t_s} \left(1 - d_i^{(t_s)}\right),$$

implying that $\overline{\mathcal{M}}$ is well-defined and satisfies Condition (B) in this iteration. Indeed, if $q^{(t_s)} \ge 2n - t_s + 1$, it is easy to see that the claim is satisfied due to Properties (ii) and (iii) of iteration t_s. Otherwise, if $q^{(t_s)} < 2n - t_s + 1$, we have

$$\sum_{i=t_s+1}^{q^{(t_s)}} \left(d_i^{(t_s)} - \overline{d}_i^{(t_s)}\right) = \sum_{i=q^{(t_s)}+1}^{2n-t^\star+1} \left(\overline{d}_i^{(t_s)} - d_i^{(t_s)}\right) \ge \sum_{i=q^{(t_s)}+1}^{2n-t_s} \left(\overline{d}_i^{(t_s)} - d_i^{(t_s)}\right),$$

where the equality follows from the fact that both $\mathcal{M}$ and $\overline{\mathcal{M}}$ have saturated all the online vertices that have arrived until this iteration (i.e., $\sum_{i=1}^{2n} d_i^{(t_s)} = \sum_{i=1}^{2n} \overline{d}_i^{(t_s)}$) while the inequality is due to Property (iv). We can then prove the claim since

$$\begin{aligned}
\sum_{i=t_s+1}^{2n-t_s} \left(1 - \overline{d}_i^{(t_s)}\right) &= \sum_{i=t_s+1}^{2n-t_s} \left(1 - d_i^{(t_s)}\right) + \sum_{i=t_s+1}^{2n-t_s} \left(d_i^{(t_s)} - \overline{d}_i^{(t_s)}\right) \\
&= \sum_{i=t_s+1}^{2n-t_s} \left(1 - d_i^{(t_s)}\right) + \sum_{i=t_s+1}^{q^{(t_s)}} \left(d_i^{(t_s)} - \overline{d}_i^{(t_s)}\right) - \sum_{i=q^{(t_s)}+1}^{2n-t_s} \left(\overline{d}_i^{(t_s)} - d_i^{(t_s)}\right) \\
&\ge \sum_{i=t_s+1}^{2n-t_s} (1 - d_i^{(t_s)}).
\end{aligned}$$

Note that Condition (A′) is satisfied in this iteration due to the execution of $\overline{\mathcal{M}}$.

Finally, for any iteration $t \ge t_s + 2$, it is easy to see that $\overline{\mathcal{M}}$ is well-defined and satisfies Conditions (A′) and (B). □

Let us lastly prove that the monotonicity at iteration n is also without loss of generality, i.e., at the beginning of Line 14 in Algorithm 1, $\mathcal{M}$ would satisfy $d_1^{(n)} \le d_2^{(n)} \le \ldots \le d_n^{(n)}$.

Lemma 47. *Given any algorithm $\mathcal{M}$ that is greedy and uniform in the common phase, there exists an algorithm $\overline{\mathcal{M}}$ that is monotone at iteration n as well while satisfying $\overline{\mathcal{M}}(\mathcal{R}) \ge \mathcal{M}(\mathcal{R})$ and $\overline{\mathcal{M}}(\mathcal{C}) = \mathcal{M}(\mathcal{C})$.*

Proof. We prove this lemma by inductively showing that, if there exists $t^\star \in \{1, \dots, n-1\}$ such that $d_{t^\star}^{(n)} = d_{t^\star}^{(t^\star)} > d_{t^\star+1}^{(t^\star+1)} = d_{t^\star+1}^{(n)}$ in the execution of $\mathcal{M}$, we can construct another algorithm $\overline{\mathcal{M}}$ that satisfies the conditions of Lemma 45 with, for every $t \in \{1, \dots, n\}$,

$$\overline{d}_t^{(n)} = \overline{d}_t^{(t)} = \begin{cases} d_{t^\star+1}^{(t^\star+1)} = d_{t^\star+1}^{(n)}, & \text{if } t = t^\star, \\ d_{t^\star}^{(t^\star)} = d_{t^\star}^{(n)}, & \text{if } t = t^\star + 1, \\ d_t^{(t)} = d_t^{(n)}, & \text{otherwise.} \end{cases} \tag{23}$$

Notice that Condition (A) immediately follows from Equation (23).

Let us now describe the execution of $\overline{\mathcal{M}}$ in the common phase. In fact, as we want $\overline{\mathcal{M}}$ to be greedy and uniform, the execution of $\overline{\mathcal{M}}$ is determined by Equation (23) (as long as it is feasible). Precisely speaking, for each iteration $t \in \{1, \dots, n\}$, $\overline{\mathcal{M}}$ first pushes water through (u_t, v_t) until the level $\overline{d}_t^{(t)}$ of u_t satisfies Equation (23). The remaining amount is then distributed uniformly towards $N(v_t) \setminus \{u_t\}$.

We need to prove that the execution of $\overline{\mathcal{M}}$ is feasible and that Condition (B) is met by $\overline{\mathcal{M}}$. To this end, let t_s be the last round at which $\mathcal{M}$ fully saturates v_t in the common phase. Since $\mathcal{M}$ is greedy while we have $d_{t^\star+1}^{(t^\star+1)} < d_{t^\star}^{(t^\star)} \leq 1$, we can observe that $\mathcal{M}$ saturates $v_{t^\star+1}$ in iteration $t^\star+1$, i.e., $t_s \geq t^\star + 1$. Note also that, up to iteration $t^\star - 1$, $\overline{\mathcal{M}}$ would behave the same as $\mathcal{M}$, implying that the execution of $\overline{\mathcal{M}}$ up to this iteration is feasible, and hence, Condition (B) is also satisfied.

Let $z := d_{t^\star}^{(t^\star)} - d_{t^\star+1}^{(t^\star+1)} > 0$. In iteration $t^\star$, $\overline{\mathcal{M}}$ pushes water towards $u_{t^\star}$ only until its level $\overline{d}_{t^\star}^{(t^\star)}$ becomes $d_{t^\star+1}^{(t^\star+1)}$. Therefore, $\overline{\mathcal{M}}$ distributes $\frac{z}{2n-2t^\star+1}$ more units than $\mathcal{M}$ towards each neighbor $u \in N(v_{t^\star}) \setminus \{u_{t^\star}\}$, implying that, for any $i \in \{t^\star+1, \dots, 2n-t^\star+1\}$,

$$\begin{aligned}
\overline{d}_i^{(t^\star)} &= d_i^{(t^\star)} + \frac{z}{2n-2t^\star+1} \\
&\leq d_{t^\star+1}^{(t^\star+1)} + \frac{z}{2n-2t^\star+1} \\
&< d_{t^\star+1}^{(t^\star+1)} + z \\
&= d_{t^\star}^{(t^\star)} \\
&\leq 1,
\end{aligned} \tag{24}$$

where the first inequality follows from the fact that $d_i^{(t^\star)} = d_{t^\star+1}^{(t^\star)} \leq d_{t^\star+1}^{(t^\star+1)}$ due to the uniformity of $\mathcal{M}$, and the second from that $t^\star + 1 \leq n$. This shows that the execution of $\overline{\mathcal{M}}$ in this iteration is indeed feasible and also satisfies Condition (B).

For each subsequent iteration $t \in \{t^\star+1, \dots, t_s\}$ where $\mathcal{M}$ saturates v_t, we claim that the level of $N(v_t) \setminus \{u_t\}$ in $\overline{\mathcal{M}}$ is less than that in $\mathcal{M}$, i.e., for every $i \in \{t+1, \dots, 2n-t+1\}$,

$$\overline{d}_i^{(t)} < d_i^{(t)}. \tag{25}$$

Note that this immediately implies that the execution of $\overline{\mathcal{M}}$ is feasible and Condition (B) is also satisfied in this iteration. Let us inductively prove the claim. Indeed, in iteration $t^\star+1$, note that the amount of water pushed towards $u_{t^\star+1}$ is

$$\begin{aligned}
\overline{d}_{t^\star+1}^{(t^\star+1)} - \overline{d}_{t^\star+1}^{(t^\star)} &= d_{t^\star}^{(t^\star)} - \overline{d}_{t^\star+1}^{(t^\star)} \\
&= d_{t^\star+1}^{(t^\star+1)} + z - d_{t^\star+1}^{(t^\star)} - \frac{z}{2n-2t^\star+1} \\
&= d_{t^\star+1}^{(t^\star+1)} - d_{t^\star+1}^{(t^\star)} + \frac{2n-2t^\star}{2n-2t^\star+1} \cdot z,
\end{aligned}$$

where the second equality comes from the definition of z and Equation (24), meaning that $\overline{\mathcal{M}}$ pushes $\frac{2n-2t^\star}{2n-2t^\star+1} \cdot z$ more units towards $u_{t^\star+1}$ than $\mathcal{M}$. We can therefore deduce that every neighbor $N(v_{t^\star+1}) \setminus \{u_{t^\star+1}\}$ other than the advice $u_{t^\star+1}$ would gain $\frac{2n-2t^\star}{(2n-2t^\star-1)(2n-2t^\star+1)} \cdot z$ less units of water in $\overline{\mathcal{M}}$ than in $\mathcal{M}$ at this iteration while it has gained $\frac{z}{2n-2t^\star+1}$ more units in the previous iteration, i.e., for every $i \in \{t^\star+2, \dots, 2n-t^\star\}$,

$$\overline{d}_i^{(t^\star+1)} = d_i^{(t^\star+1)} + \frac{z}{2n-2t^\star+1} - \frac{2n-2t^\star}{(2n-2t^\star-1)(2n-2t^\star+1)} \cdot z < d_i^{(t^\star+1)}$$

as claimed in Equation (25).

For the remaining iterations $t \in \{t^\star+2, \dots, t_s\}$, let $\varepsilon := d_t^{(t-1)} - \overline{d}_t^{(t-1)}$. Observe that $\overline{\mathcal{M}}$ pushes ε more units towards u_t than $\mathcal{M}$ to have $\overline{d}_t^{(t)} = d_t^{(t)}$. This implies that $\overline{\mathcal{M}}$ pushes $\frac{2n-2t+2}{2n-2t+1} \cdot \varepsilon$ less units towards each neighbor in $N(v_t) \setminus \{u_t\}$, yielding that

$$\overline{d}_i^{(t)} = d_i^{(t)} - \frac{2n-2t+2}{2n-2t+1} \cdot \varepsilon < d_i^{(t)}$$

for every $i \in \{t+1, \ldots, 2n-t+1\}$. This completes the proof of Equation (25).

Note that, if $t_s = n$, the proof of Lemma 47 immediately follows from the claim. On the other hand, if $t_s < n$, it is easy to observe that the execution of $\overline{\mathcal{M}}$ is feasible and that Condition (B) is also satisfied since every neighbor $N(v_{t_s+1})$ of v_{t_s+1} becomes saturated in $\mathcal{M}$, and therefore, $\mathcal{M}$ pushes no water from iteration $t_s + 2$ (if any). □

6 Experiments

In this section, we present experimental results for empirical evaluation of our algorithms. We experimented on synthetic random graphs defined in Section 6.1 and real-world graphs defined in Section 6.2. For weighted instances, each offline vertex is given a uniform random weight between 0 and 1000. Section 6.3 presents the way of generating advice given a noise parameter, followed by description of benchmarked algorithms in Section 6.4. Each plot is generated by letting each algorithm solve 10 instances for 10 different noise parameter values. That is, a plot for weighted instances with 6 algorithms involved solving 600 instances while a plot for unweighted instances with 10 algorithms involved solving 1000 instances. We defer the full plots of our experimental results to Appendix B. All experiments were performed on a personal laptop (Apple Macbook 2024, M4 chip, 16GB memory).

6.1 Synthetic random graphs

Erdős-Rényi (ER) graphs. Given a number of nodes $n \in \{100, 200, 300\}$ and edge probability $p \in \{0.1, 0.2, 0.5\}$, an ER graph is generated with n offline nodes and n online nodes and each edge in the complete bipartite graph exists independently with probability p.

Upper Triangular (UT) graphs. Given a number of nodes $n \in \{100, 200, 300\}$, a UT graph is generated with n offline nodes and n online nodes where the i-th online node is connected to the last $n - i + 1$ offline nodes.

6.2 Real-world graphs

To evaluate our algorithmic performance on real-world graph structures, we considered 6 publicly available graphs from the Network Data Repository [RA15] and pre-processed them in a similar manner to [BKP20] to obtain random bipartite graphs: first, shuffle all n node indices in the real-world graph, take the first $\lfloor n/2 \rfloor$ as the offline vertices and the next $\lfloor n/2 \rfloor$ as online vertices and only keep the bipartite crossing edges. Each random shuffle of the real-world graph induces a random bipartite graph instance which we then experiment on. Note that such a pre-processing step is necessary because these real-world graphs are not bipartite to begin with.

6.3 Advice generation

For each graph $\mathcal{G}$ with n vertices and a given noise parameter $\gamma \in [0, 1]$, we generate a noisy prediction $\widehat{\mathcal{G}}_\gamma$ of $\mathcal{G}$ as follows: each online vertex v retains a random $(1 - \gamma)$ fraction of its true neighbors and gains a random γ fraction of its non-neighbors. Thus, when $\gamma = 0$, the prediction is exact ($\widehat{\mathcal{G}}_0 = \mathcal{G}$), and when $\gamma = 1$, it corresponds to the complement graph ($\widehat{\mathcal{G}}_1 = \overline{\mathcal{G}}$).

To generate the advice for the t-th arriving online vertex (for $t \in [n]$), we solve a linear program that maximizes the (weighted) matching objective. This is done subject to two components: the actual decisions made for the first $t - 1$ arrivals in the true graph $\mathcal{G}$, and a noisy prediction of the future arrivals from time $t + 1$ to n, based on $\widehat{\mathcal{G}}_\gamma$. Importantly, the current arrival at time t is not included in the noisy future but is instead the decision variable of interest. In more detail, the advice at time t is generated by perturbing the true future subgraph (i.e., the part of $\mathcal{G}$ involving vertices $t + 1$ to n) to create a noisy forecast. We then solve for the optimal decision at time t that maximizes the matching value, given the past decisions up to $t - 1$ (in $\mathcal{G}$) and the predicted future (in $\widehat{\mathcal{G}}_\gamma$). Since we use the true graph up to and including time t, this process ensures that the advice at each time step is always feasible and based on a valid optimization problem over a fully specified n-vertex instance.

6.4 Benchmarked algorithms

The two baselines are GREEDY and BALANCE. The former greedily matches the online vertex with its highest weighted available offline neighbor while the latter fractionally matches based on the penalty function $g(z) = e^{z-1}$. In the unweighted setting, BALANCE is equivalent to the classic WATERFILLING algorithm. Note that both GREEDY and BALANCE are independent of any predictions so they would achieve constant performance for any noise parameter $\gamma \in [0, 1]$. We also implemented and benchmarked our LEARNINGAUGMENTEDBALANCE (LAB; Algorithm 3) and PUSHANDWATERFILL (PAW; Algorithm 4) algorithms, where each takes as inputs λ_{LAB} and λ_{PAW} respectively. Note that the guarantees for PAW only hold for unweighted instances.

Recall from Theorems 1 and 3 that LAB and PAW have different consistency values with respect to their parameters: the consistency of LAB is $1 + \lambda_{\text{LAB}} - \exp(\lambda_{\text{LAB}} - 1)$ while the consistency of PAW is $1 - (1 - \lambda_{\text{PAW}}) \exp(\lambda_{\text{PAW}} - 1)$. To compare between them at the same consistency value, we set $\lambda_{\text{PAW}} = 1 + W(\lambda_{\text{LAB}} - \exp(\lambda_{\text{LAB}} - 1))$. Since LAB with $\lambda_{\text{LAB}} = 0$ and PAW with $\lambda_{\text{PAW}} = 0$ are already equivalent with BALANCE of consistency $1 - 1/e$, we consider consistency ratios of $\{0.7, 0.8.0.9, 1.0\}$ when running LAB and PAW. This translates to the following parameters:

- For consistency 0.7, $\lambda_{\text{LAB}} \approx 0.111113$ and $\lambda_{\text{PAW}} \approx 0.510598$.
- For consistency 0.8, $\lambda_{\text{LAB}} \approx 0.293239$ and $\lambda_{\text{PAW}} \approx 0.740829$.
- For consistency 0.9, $\lambda_{\text{LAB}} \approx 0.516817$ and $\lambda_{\text{PAW}} \approx 0.888167$.
- For consistency 1.0, $\lambda_{\text{LAB}} = \lambda_{\text{PAW}} = 1$.

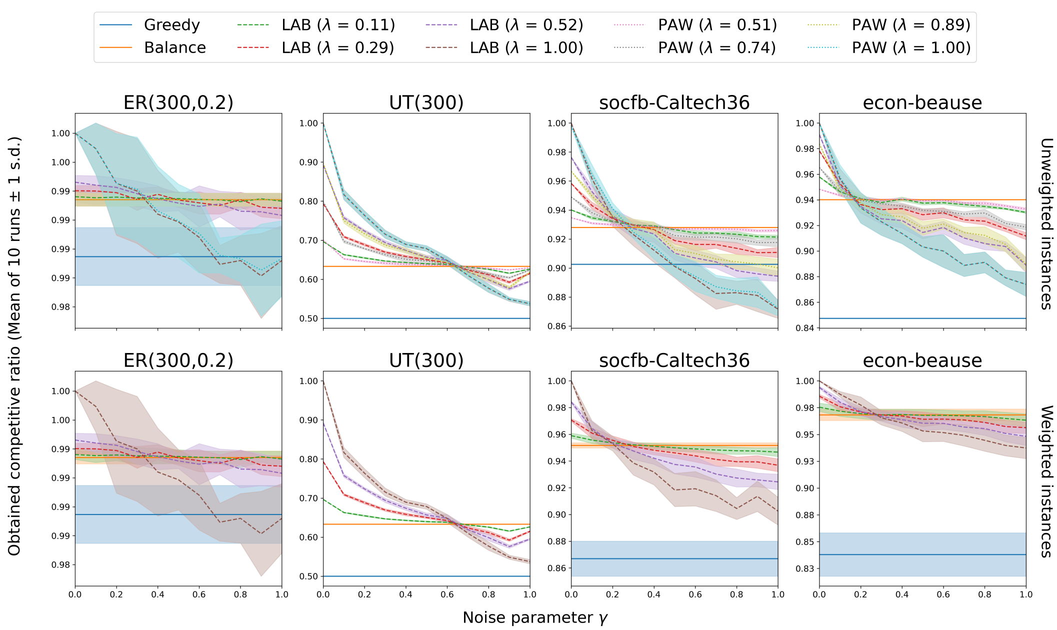


Figure 7: Subset of empirical results: ER(300, 0.2), UT(300), and 2 real-world graphs (socfb-Caltech36, econ-beause). See the Appendix B for our full set of experiments.

6.5 Qualitative takeaways

Fig. 7 illustrates a subset of our empirical results. As predicted by our analysis, the competitive ratio attained by both LAB and PAW degrades as the noise parameter γ increases. In particular, when $\gamma = 0$ (i.e., perfect advice), both LAB and PAW achieve a competitive ratio of 1 when $\lambda_{\text{LAB}} = \lambda_{\text{PAW}} = 1$. As γ grows large, the advice becomes increasingly uninformative, and it is unsurprising that the advice-free algorithm BALANCE eventually outperforms both learning-augmented algorithms, with the crossing point depending on the underlying graph instance.

Interestingly, across all our experiments — including those in the appendix — we consistently observe a phenomenon where there appears to exist a critical noise level γ^* such that the competitive ratios of all runs of LAB and PAW (across different λ values) converge and coincide with that of BALANCE. This suggests that at γ^*, the advice becomes effectively uncorrelated with the input, causing the behavior of LAB and PAW to resemble that of BALANCE regardless of the weighting parameter λ. While we do not currently have a theoretical explanation for this convergence, it is a compelling empirical observation that may point to deeper structure in the robustness-consistency tradeoff and warrants further investigation in future work.

7 Conclusion and Open Problems

We studied the robustness-consistency tradeoffs of learning-augmented algorithms for online bipartite fractional matching. We proposed and analyzed two algorithms, LAB and PAW, and established an improved hardness result.

In our current work, PAW relies on integral advice while LAB can accommodate fractional advice. While it is a natural question to ask if there can be a unified algorithm and analysis, our current analytical framework is unable to do so. The analysis of LAB is agnostic to the weights, making it unclear how to demonstrate an improved tradeoff in the unweighted case. Meanwhile, the analysis of PAW crucially relies on the integrality of the advice, and we were unable to obtain a comparable bound in the fractional case. We do not rule out the possibility of a unified analysis, and we view this as a compelling direction for future work. We do not rule out the possibility of a unified analysis and view this as an intriguing direction for future work.

Besides unifying the two variants, there are serveral other natural open and interesting research directions. Firstly, it would be interesting to develop a theoretical explanation for the crossing point phenomenon observed in our experiments; see the discussion in Section 6.5. Another direction would be to close the gap between our algorithmic results and the impossibility bounds. Progress on this front could come from an analytic proof of the impossibility result, as well as a tight analysis of LAB in the unweighted setting. Finally, it would be interesting to extend our framework to broader variants of online matching, including Display Ads, the generalized assignment problem [FKM+09, SE23], and the multi-stage setting [FN24].

Acknowledgments and Disclosure of Funding

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-PhD/2021-08-013). This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2021-II212068, Artificial Intelligence Innovation Hub). This work was partly supported by an IITP grant funded by the Korean Government (MSIT) (No. RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University)). Supported by NCN grant number 2020/39/B/ST6/01641.

References

[ACE+23] Antonios Antoniadis, Christian Coester, Marek Eliáš, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. *ACM transactions on algorithms*, 19(2):1–34, 2023.

[ACI22] Anders Aamand, Justin Chen, and Piotr Indyk. (Optimal) Online Bipartite Matching with Degree Information. *Advances in Neural Information Processing Systems*, 35:5724–5737, 2022.

[AGKK20] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. Secretary and online matching problems with machine learned advice. *Advances in Neural Information Processing Systems*, 33:7933–7944, 2020.

[AGKM11] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1253–1264. SIAM, 2011.

[ALT21] Yossi Azar, Stefano Leonardi, and Noam Touitou. Flow time scheduling with uncertain processing time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1070–1080, 2021.

[APT22] Yossi Azar, Debmalya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 35–66. SIAM, 2022.

[BC22] Guy Blanc and Moses Charikar. Multiway online correlated selection. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1277–1284. IEEE, 2022.

[BCJG25a] Arnab Bhattacharyya, Davin Choo, Philips George John, and Themis Gouleakis. Learning multivariate gaussians with imperfect advice. *International Conference on Machine Learning (ICML)*, 2025.

[BCJG25b] Arnab Bhattacharyya, Davin Choo, Philips George John, and Themis Gouleakis. Product distribution learning with imperfect advice. *Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[BCK+22] Nikhil Bansal, Christian Coester, Ravi Kumar, Manish Purohit, and Erik Vee. Learning-augmented weighted paging. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 67–89. SIAM, 2022.

[BJN07] Niv Buchbinder, Kamal Jain, and Joseph Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *European Symposium on Algorithms*, pages 253–264. Springer, 2007.

[BKP20] Allan Borodin, Christodoulos Karavasilis, and Denis Pankratov. An experimental study of algorithms for online bipartite matching. *Journal of Experimental Algorithmics (JEA)*, 25:1–37, 2020.

[BM08] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *Acm Sigact News*, 39(1):80–87, 2008.

[BMS20] Etienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. *Advances in Neural Information Processing Systems*, 33:20083–20094, 2020.

[BNW23] Niv Buchbinder, Joseph Naor, and David Wajc. Lossless online rounding for online bipartite matching (despite its impossibility). In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2030–2068. SIAM, 2023.

[BP24] Ziyad Benomar and Vianney Perchet. Non-clairvoyant scheduling with partial predictions. In *Forty-first International Conference on Machine Learning, ICML 2024*, 2024.

[CGB23] Davin Choo, Themistoklis Gouleakis, and Arnab Bhattacharyya. Active Causal Structure Learning with Advice. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 5838–5867, 2023.

[CGLB24] Davin Choo, Themis Gouleakis, Chun Kai Ling, and Arnab Bhattacharyya. Online bipartite matching with imperfect advice. *International Conference on Machine Learning*, 235:8762–8781, 2024.

[CLV03] Fan Chung, Linyuan Lu, and Van Vu. Spectra of random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 100(11):6313–6318, 2003.

[DHK+16] Nikhil R Devanur, Zhiyi Huang, Nitish Korula, Vahab S Mirrokni, and Qiqi Yan. Whole-page optimization and submodular welfare maximization with online bidders. *ACM Transactions on Economics and Computation (TEAC)*, 4(3):1–20, 2016.

[DJK13] Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 101–107. SIAM, 2013.

[DJSW19] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. *Journal of the ACM (JACM)*, 66(1):1–41, 2019.

[DSSX21] John P Dickerson, Karthik A Sankararaman, Aravind Srinivasan, and Pan Xu. Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. *ACM Transactions on Economics and Computation (TEAC)*, 9(3):1–17, 2021.

[EADL24] Alex Elenter, Spyros Angelopoulos, Christoph Dürr, and Yanni Lefki. Overcoming brittleness in pareto-optimal learning augmented algorithms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[EFFS21] Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. An economics-based analysis of RANKING for online bipartite matching. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 107–110. SIAM, 2021.

[FHTZ22] Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. Edge-weighted online bipartite matching. *Journal of the ACM*, 69(6):1–35, 2022.

[FKM+09] Jon Feldman, Nitish Korula, Vahab Mirrokni, Shanmugavelayutham Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In *International workshop on internet and network economics*, pages 374–385. Springer, 2009.

[FMMM09] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126. IEEE, 2009.

[FN24] Yiding Feng and Rad Niazadeh. Batching and optimal multistage bipartite allocations. *Management Science*, 2024.

[FNS24] Yiding Feng, Rad Niazadeh, and Amin Saberi. Two-stage stochastic matching and pricing with applications to ride hailing. *Operations Research*, 72(4):1574–1594, 2024.

[FSZ16] Moran Feldman, Ola Svensson, and Rico Zenklusen. Online contention resolution schemes. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1014–1033. SIAM, 2016.

[GHH+22] Ruiquan Gao, Zhongtian He, Zhiyi Huang, Zipei Nie, Bijun Yuan, and Yan Zhong. Improved online correlated selection. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1265–1276. IEEE, 2022.

[GKM+19] Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 26–37. IEEE, 2019.

[GLS+22] Elena Grigorescu, Young-San Lin, Sandeep Silwal, Maoyuan Song, and Samson Zhou. Learning-augmented algorithms for online linear and semidefinite programming. *Advances in Neural Information Processing Systems*, 35:38643–38654, 2022.

[GM08] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to Adwords. In *SODA*, volume 8, pages 982–991, 2008.

[Gur24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[HHIS24] Hadi Hosseini, Zhiyi Huang, Ayumi Igarashi, and Nisarg Shah. Class fairness in online matching. *Artificial Intelligence*, 335:104177, 2024.

[HKT+20] Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. Fully online matching. *Journal of the ACM (JACM)*, 67(3):1–25, 2020.

[HPT+19] Zhiyi Huang, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, and Yuhao Zhang. Tight competitive ratios of classic matching algorithms in the fully online model. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2875–2886. SIAM, 2019.

[HS21] Zhiyi Huang and Xinkai Shu. Online stochastic matching, poisson arrivals, and the natural linear program. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 682–693, 2021.

[HSY22] Zhiyi Huang, Xinkai Shu, and Shuyi Yan. The power of multiple choices in online stochastic matching. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 91–103, 2022.

[HTW24] Zhiyi Huang, Zhihao Gavin Tang, and David Wajc. Online matching: A brief survey. *ACM SIGecom Exchanges*, 22(1):135–158, 2024.

[HTWZ20] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. Fully online matching II: Beating ranking and water-filling. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1380–1391. IEEE, 2020.

[HZZ24] Zhiyi Huang, Qiankun Zhang, and Yuhao Zhang. AdWords in a panorama. *SIAM Journal on Computing*, 53(3):701–763, 2024.

[IKPP22] Sungjin Im, Ravi Kumar, Aditya Petety, and Manish Purohit. Parsimonious learning-augmented caching. In *International Conference on Machine Learning*, pages 9588–9601. PMLR, 2022.

[IKQP23] Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. *ACM Transactions on Parallel Computing*, 10(4):1–26, 2023.

[JM22] Billy Jin and Will Ma. Online bipartite matching with advice: Tight robustness-consistency tradeoffs for the two-stage model. *Advances in Neural Information Processing Systems*, 35:14555–14567, 2022.

[JPS22] Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. Online algorithms for weighted paging with predictions. *ACM Transactions on Algorithms (TALG)*, 18(4):1–27, 2022.

[JW21] Billy Jin and David P Williamson. Improved analysis of ranking for online vertex-weighted bipartite matching in the random order model. In *International Conference on Web and Internet Economics*, pages 207–225. Springer, 2021.

[KMT11] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 587–596, 2011.

[KP00] Bala Kalyanasundaram and Kirk R Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1-2):319–325, 2000.

[KP09] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 508–520. Springer, 2009.

[KPS18] Ravi Kumar, Manish Purohit, and Zoya Svitkina. Improving online algorithms via ML predictions. *Advances in Neural Information Processing Systems*, 31, 2018.

[KRTV13] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *European symposium on algorithms*, pages 589–600. Springer, 2013.

[KVV90] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

[LLMV20] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1859–1877. SIAM, 2020.

[LV21] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68(4):1–25, 2021.

[Meh13] Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013.

[MNS07] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 288–294, 2007.

[MNS12] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Online optimization with uncertain information. *ACM Transactions on Algorithms (TALG)*, 8(1):1–29, 2012.

[MSVV07] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. AdWords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22–es, 2007.

[MV22] Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with Predictions. *Communications of the ACM*, 65(7):33–35, 2022.

[NSW25] Joseph Naor, Aravind Srinivasan, and David Wajc. Online dependent rounding schemes for bipartite matchings, with applications. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3116–3154. SIAM, 2025.

[RA15] Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, pages 4292–4293, 2015.

[SA21] Yongho Shin and Hyung-Chan An. Making three out of two: Three-way online correlated selection. In *32nd International Symposium on Algorithms and Computation, ISAAC 2021*, volume 212 of *LIPIcs*, pages 49:1–49:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[SE23] Fabian Spaeh and Alina Ene. Online ad allocation with predictions. *Advances in Neural Information Processing Systems*, 36:17265–17295, 2023.

[SE24] Karim Abdel Sadek and Marek Eliás. Algorithms for caching and MTS with reduced number of predictions. In *The Twelfth International Conference on Learning Representations, ICLR 2024*, 2024.

[SLLA23] Yongho Shin, Changyeol Lee, Gukryeol Lee, and Hyung-Chan An. Improved learning-augmented algorithms for the multi-option ski rental problem via best-possible competitive analysis. In *International Conference on Machine Learning*, pages 31539–31561. PMLR, 2023.

[WLW20] Shufan Wang, Jian Li, and Shiqiang Wang. Online algorithms for multi-shop ski rental with machine learned advice. *Advances in Neural Information Processing Systems*, 33:8150–8160, 2020.

[WW15] Yajun Wang and Sam Chiu-wai Wong. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *International Colloquium on Automata, Languages, and Programming*, pages 1070–1081. Springer, 2015.

[ZTCD24] Hailiang Zhao, Xueyan Tang, Peng Chen, and Shuiguang Deng. Learning-augmented algorithms for the Bahncard problem. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024*, 2024.

A Pseudocodes of our algorithms

Algorithm 3: Learning-Augmented Balance Algorithm (LAB)

Input: Offline vertices U, tradeoff parameter $\lambda \in [0, 1]$
Data: Online vertices V, edges E, and fractional advice $a \in \mathbb{R}^E$
Output: Fractional matching $x \in \mathbb{R}^E$

foreach $u \in U$ **do**
    $X_u \leftarrow 0$ // Amount allocated by algorithm
    $A_u \leftarrow 0$ // Amount allocated by advice
foreach *arrival of* $v \in V$ *with neighbors* $N(v)$ *and advice* $\{a_{u,v}\}_{u \in N(v)}$ **do**
    foreach $u \in N(v)$ **do**
        $A_u \leftarrow A_u + a_{u,v}$ // Accumulate advice
    Find the smallest $\ell \geq 0$ such that $\sum_{u \in N(v)} x_{u,v} \leq 1$, where
     $x_{u,v} := \min\{z \in [0, 1 - X_u] \mid w_u \cdot (1 - f(A_u, X_u + z)) \leq \ell\}$ // e.g. via binary search
    foreach $u \in N(v)$ **do**
        $X_u \leftarrow X_u + x_{u,v}$ // Accumulate actual fractional matching
return x

Algorithm 4: Push-and-Waterfill Algorithm (PAW)

Input: Offline vertices U, trade-off parameter $\lambda \in [0, 1]$
Data: Online vertices V, edges E, and integral advice $A : V \to U \cup \{\perp\}$
Output: Fractional matching $x \in \mathbb{R}^E$

foreach $u \in U$ **do**
    $d_u \leftarrow 0$ // Level of u
foreach *arrival of* $v \in V$ *with neighbors* $N(v)$ *and advice* $A(v)$ **do**
    (Phase 1): Push to advised neighbor $A(v)$, up to $\tau = \max\{0, \lambda - d_{A(v)}\}$ amount
    if $A(v) \in N(v)$ **then**
        $\tau \leftarrow \max\{0, \lambda - d_{A(v)}\}$
        $x_{A(v),v} \leftarrow \tau$
        $d_{A(v)} \leftarrow d_{A(v)} + \tau$
    else
        $\tau \leftarrow 0$
    (Phase 2): Waterfill the remaining $1 - \tau$
    Find the largest ℓ such that $\sum_{u \in N(v)} \max\{0, \ell - d_u\} \leq 1 - \tau$
    $\ell \leftarrow \min\{\ell, 1\}$
    foreach $u \in N(v)$ **do**
        $x_{u,v} \leftarrow x_{u,v} + \max\{0, \ell - d_u\}$
        $d_u \leftarrow d_u + \max\{0, \ell - d_u\}$
return x

B Plots of our experimental results

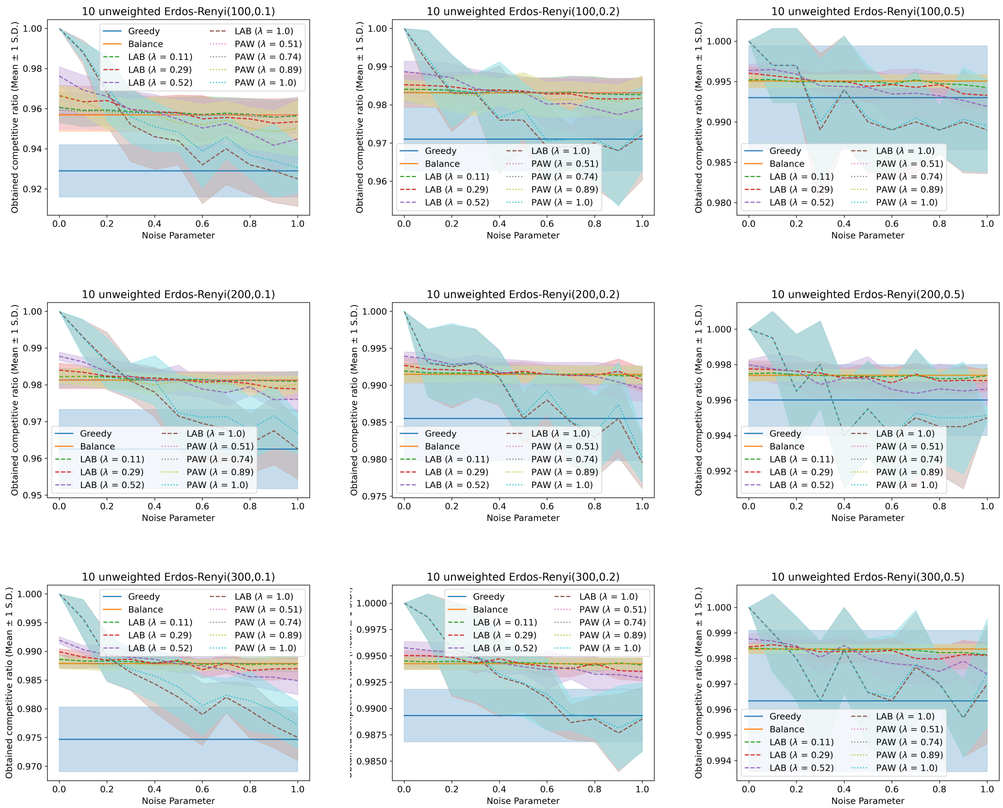


Figure 8: Empirical results for unweighted Erdős-Rényi graph instances with $n \in \{100, 200, 300\}$ and $p \in \{0.1, 0.2, 0.5\}$

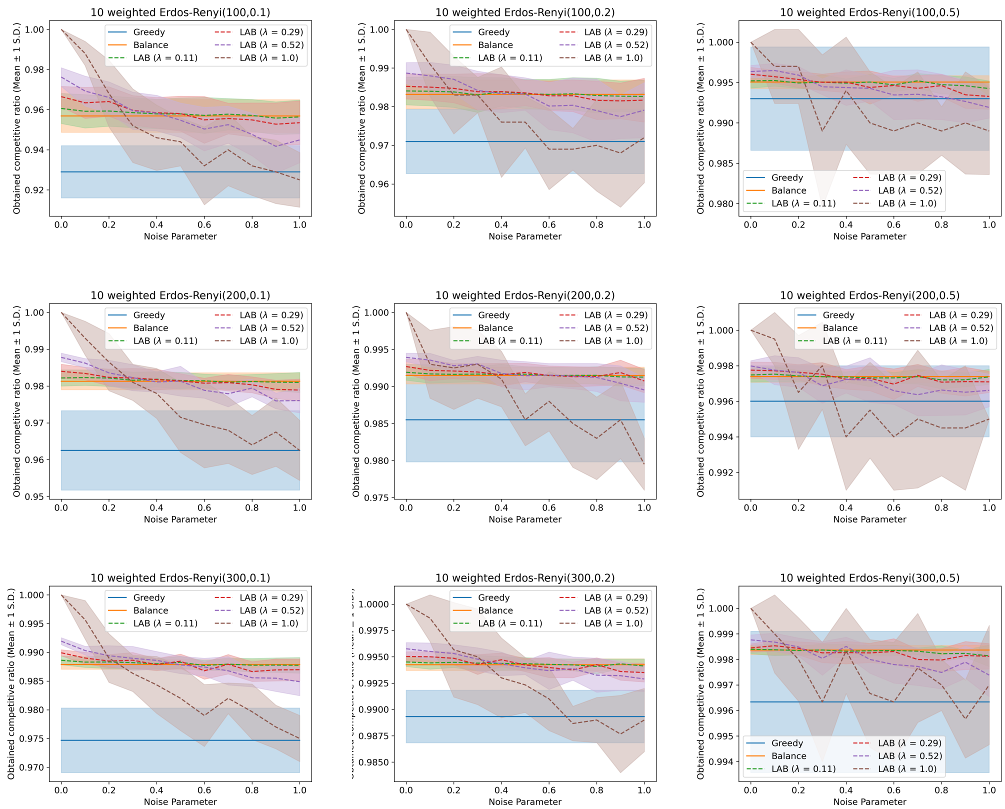


Figure 9: Empirical results for unweighted Erdős-Rényi graph instances with $n \in \{100, 200, 300\}$ and $p \in \{0.1, 0.2, 0.5\}$

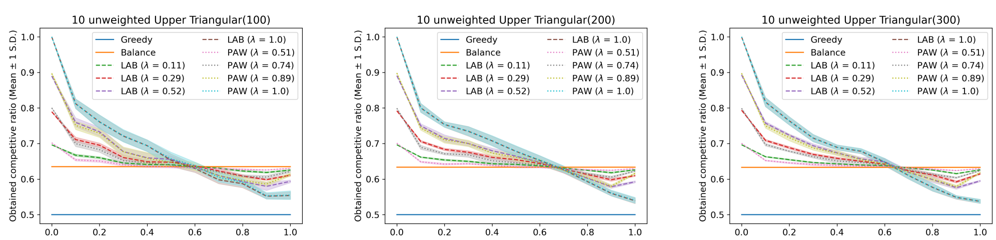


Figure 10: Empirical results for unweighted Upper Triangular graph instances with $n \in \{100, 200, 300\}$.

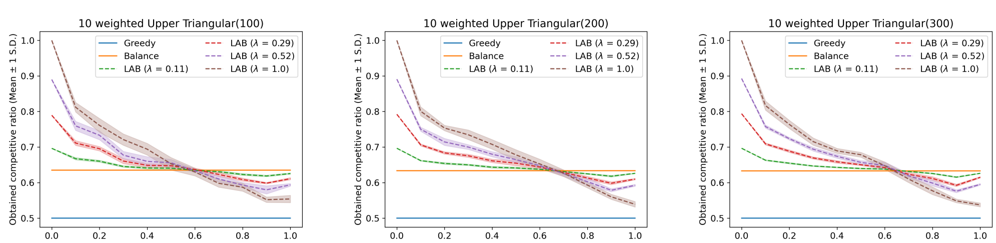


Figure 11: Empirical results for weighted Upper Triangular graph instances with $n \in \{100, 200, 300\}$.

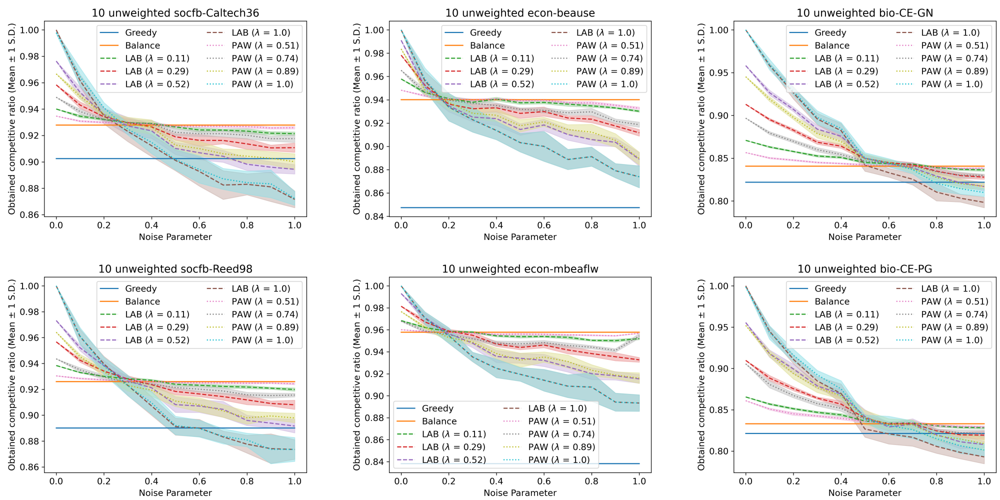


Figure 12: Empirical results for unweighted real-world graph instances.

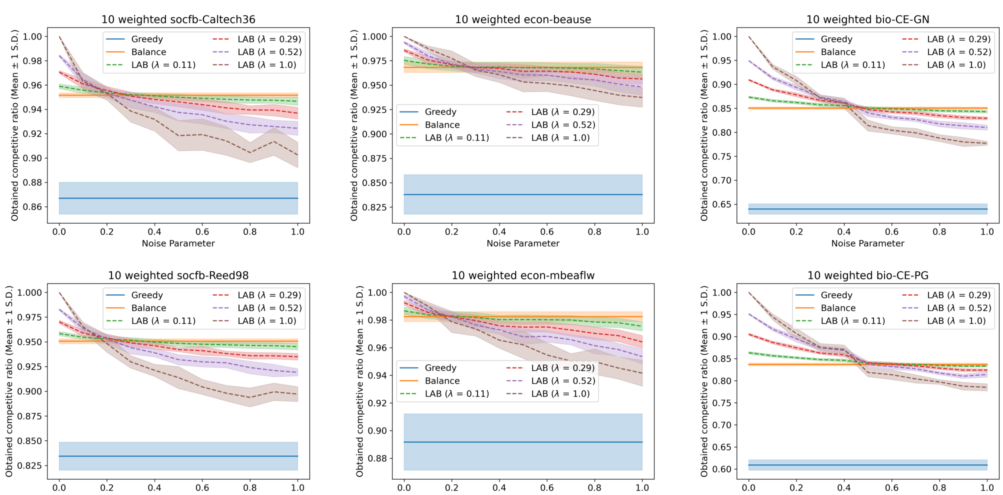


Figure 13: Empirical results for unweighted real-world graph instances.