The Online Submodular Assignment Problem

Daniel Hathcock*   Billy Jin†   Kalen Patton‡   Sherry Sarkar*   Michael Zlatin*

Abstract

Online resource allocation is a rich and varied field. One of the most well-known problems in this area is online bipartite matching, introduced in 1990 by Karp, Vazirani, and Vazirani [KVV90]. Since then, many variants have been studied, including AdWords, the generalized assignment problem (GAP), and online submodular welfare maximization.

In this paper, we introduce a generalization of GAP which we call the submodular assignment problem (SAP). This generalization captures many online assignment problems, including all classical online bipartite matching problems as well as broader online combinatorial optimization problems such as online arboricity, flow scheduling, and laminar restricted allocations. We present a fractional algorithm for online SAP that is $(1 - \frac{1}{e})$-competitive.

Additionally, we study several integral special cases of the problem. In particular, we provide a $(1 - \frac{1}{e} - \varepsilon)$-competitive integral algorithm under a small-bids assumption, and a $(1 - \frac{1}{e})$-competitive integral algorithm for online submodular welfare maximization where the utility functions are given by rank functions of matroids.

The key new ingredient for our results is the construction and structural analysis of a "water level" vector for polymatroids, which allows us to generalize the classic water-filling paradigm used in online matching problems. This construction reveals connections to submodular utility allocation markets and principal partition sequences of matroids.

*Carnegie Mellon University
†Cornell University
‡Georgia Institute of Technology

1 Introduction

Online assignment problems are fundamental in the study of online algorithms. Perhaps the most well-known online assignment problem is online bipartite matching, introduced by Karp, Vazirani, and Vazirani [KVV90]. In online bipartite matching, we are given one side of a bipartite graph (the *offline* vertices) in advance, while the vertices on the other side arrive online. When an online vertex arrives, all of its incident edges are revealed, and the algorithm selects at most one of the edges. The goal is to maximize the number of edges chosen, subject to the edges being a matching in the graph. For this problem, Karp, Vazirani, and Vazirani proposed the Ranking algorithm which achieves a tight $1 - 1/e$ competitive ratio.

Since then, online bipartite matching has received considerable attention, and more general variations of the problem have been studied. Some of the most prominent examples include:

- **Vertex and Edge Weighted Variants.** In vertex weighted online bipartite matching, each *offline* vertex has a weight, and the goal is to maximize the sum of the weights of the matched offline vertices. In the more general edge-weighted setting, individual edges have weight and the goal is to maximize the sum of the weights of the selected edges.

- **AdWords.** This was introduced by Mehta, Saberi, Vazirani, and Vazirani [MSVV07], motivated by the AdWords market in digital advertising. Each offline vertex i has a budget B_i and each edge e has a bid b_e. Selecting an edge consumes an amount of budget from the offline vertex equal to the bid of the edge. The goal is to maximize the total sum of the bids of the selected edges, subject to the budget constraints. Note that vertex-weighted bipartite matching is a special case of AdWords with $b_{ij} = B_i$ for all edges ij.

- **Generalized Assignment Problem (GAP).** Here, every offline vertex has a budget B_i, and every edge e has both a value v_e and a cost b_e. The goal is to maximize the total value of the selected edges, such that the total cost of the edges incident to any offline vertex does not exceed its budget. This is one of the broadest online matching problems that has been studied in the literature, and generalizes all of the settings above. In particular, AdWords is the special case of GAP with $b_e = v_e$ for all edges e. Edge-weighted bipartite matching is the case with all B_i and b_e equal to 1.

All of the above problems admit $(1 - 1/e)$-competitive algorithms under various assumptions. For vertex-weighted bipartite matching, Aggarwal, Goel, Karande and Mehta [AGKM11] give a generalization of the Ranking algorithm which is $(1 - 1/e)$-competitive. For AdWords, [MSVV07] show the same competitive ratio can be achieved for the fractional version of the problem, and more generally for the integral version under a small-bids assumption.[1] Edge-weighted bipartite matching and GAP are commonly studied under the "free disposal" assumption, which is necessary to outmaneuver a trivial $1/n$ hardness in these settings. Under free disposal, $(1 - 1/e)$-competitive algorithms can be obtained for fractional edge-weighted bipartite matching and GAP, and for GAP under a small-bids assumption [FKM+09].

Nevertheless, many natural online assignment problems exist which are not captured by the above settings. We illustrate these in the examples below. To our knowledge, no optimally competitive algorithms for these problems are implied by prior work.

---

[1]The small-bids assumption states that the ratio $\frac{b_e}{B_i}$ should be small, for any offline i and any edge e incident to i.

- **Laminar Restricted Matchings.** Consider the AdWords problem. Suppose that in addition to the budget constraints for each offline node, we have a laminar family $\mathcal{S}$ of subsets of offline nodes, and there is a budget constraint for each $S \in \mathcal{S}$. For instance, this can model a setting where a company has several departments each with their own individual ad budget, and the company as a whole also has an additional budget constraint for the total amount that can be spent across all its departments.

- **Matroid Coloring.** Suppose we have a matroid $\mathcal{M}$ whose elements arrive one by one online. We have Δ colors and may irrevocably assign a color to each element as it arrives, subject to the constraint that each color must be independent in $\mathcal{M}$. The objective is to color as many elements as possible. Two natural applications of this problem are:

  - *Online Arboricity.* Suppose the edges of an undirected graph $G = (V, E)$ arrive online. When each edge arrives, we irrevocably decide whether or not to select it. The goal is to maintain the largest possible sub-graph with arboricity[2] at most Δ. One way to solve this problem is by modelling it as a matroid coloring problem, where $\mathcal{M}$ is the graphic matroid associated with G. The arboricity of a graph is a well studied property which has been used to maintain dynamic edge orientations [CCH+23] and proper colorings of a sub-graph [CR22].

  - *Flow Scheduling.* Suppose we have a network N, with integer capacities on the edges, that is known up front with a single sink t. The times where the network is available for use is partitioned into Δ many time slots. Source vertices with unit demand appear one by one. When a source s_j appears, we must schedule it in one of the time slots (or not schedule it at all). The goal is to maximize the number of assignments, such that for every time slot, it is feasible to simultaneously send the flow for all sources scheduled in that slot. This is matroid coloring where $\mathcal{M}$ is a gammoid.

- **Coflows.** Say we have a computing resource which may process some tasks in parallel. For example, perhaps a single server rack is made up of different servers, each of which is equipped to handle only certain types of tasks. What a server rack can handle is modelled via a bipartite graph, with potential tasks on one side and servers of the server rack on the other. Tasks which may be processed together on a single rack form a transversal matroid; these are called *coflows*, inspired by applications to MapReduce [CS12]. Coflows governed by general matroid constraints have been studied [JKR17, IMPP19] in an offline setting. In an online formulation of this problem, we have Δ server racks and n tasks arriving online. The tasks are splittable, but have different costs and values for being completed at different servers (i.e., some servers are closer or cheaper than others). We must irrevocably split tasks among computing resources, though we my drop tasks later on; the goal is to handle as many tasks as possible.

In this paper, we define the *Online Submodular Assignment Problem*, which captures all of the problems described earlier as special cases. Via our results on this more general problem, we provide $1 - 1/e$ competitive algorithms for all the problems above.

---

[2]The arboricity of a graph is the minimum number of forests required to cover its edges.

1.1 Problem Statement

The *Online Submodular Assignment Problem* (*Online SAP* for brevity), is as follows. We have an (offline) monotone submodular[3] function f over ground set E with $f(\varnothing) = 0$ and $f(\{e\}) > 0$ for all $e \in E$.[4] Every element $e \in E$ has a value v_e and a cost b_e. The ground set, initially unknown, is partitioned into parts $Q_1, \ldots, Q_m$ that arrive online one-by-one. Upon arrival, each Q_j reveals its contained elements along with their values and costs. We have offline access to an evaluation oracle for f that may be called on any subset of elements revealed so far.

When a part Q_j arrives, we may select at most one element from Q_j. At any point, we also can choose to freely dispose of elements previously selected (known as the *free disposal* assumption).[5] The goal is to choose a set $S^* \subseteq E$ so as to maximize $\sum_{e \in S^*} v_e$ while maintaining that S^* satisfies the online assignment constraints

$$|S^* \cap Q_j| \leq 1 \quad \text{for all } j \in \{1, \ldots, m\}$$

and the offline submodular constraints

$$\sum_{e \in S} b_e \leq f(S) \quad \text{for all } S \subseteq S^*$$

We note that, since online SAP is a generalization of edge weighted online bipartite matching, the free disposal assumption is necessary to avoid a trivial $1/n$-hardness.

In the fractional variant of this problem, we instead choose a fractional allocation $(x_e)_{e \in Q_j}$ on the elements in Q_j when it arrives. In accord with the free disposal assumption, we may decrease x_e at any point. The objective is to maximize the final value of $\sum_{e \in E} v_e x_e$. As before, we must allocate no more than 1 total unit to elements in each Q_j. In other words, we have $x(Q_j) := \sum_{e \in Q_j} x_e \leq 1$. Moreover, the total cost vector $bx := (b_e x_e)_{e \in E}$ must obey submodular constraints defined by f, i.e., so that $bx(S) = \sum_{e \in S} b_e x_e \leq f(S)$ for every $S \subseteq E$. Put another way, we must maintain a point $x \in \mathcal{P}_f \cap \mathcal{Q}$, where $\mathcal{P}_f$ and $\mathcal{Q}$ are defined respectively as:

$$\mathcal{P}_f := \left\{ x \in \mathbb{R}_{\geq 0}^E : bx(S) \leq f(S) \text{ for every } S \subseteq E \right\}$$

and

$$\mathcal{Q} := \left\{ x \in \mathbb{R}_{\geq 0}^E : x(Q_j) \leq 1 \text{ for every } j = 1, \ldots, n \right\}.$$

Note that Online SAP captures all three assignment problems posed in the introduction. We show how the Laminar Restricted Matching Problem can be modeled as Online SAP in Appendix B. In Online Matroid Coloring, to color a matroid $\mathcal{M}$ online with Δ colors, we consider the product matroid $\mathcal{M}^\Delta := \mathcal{M} \times \ldots \times \mathcal{M}$ and define the submodular constraint to be the rank function of the lifted matroid $f := \text{rank}_{\mathcal{M}^\Delta}$. The assignment constraint dictates each element may map to at most 1 color, and the submodular constraint $f := \text{rank}_{\mathcal{M}^\Delta}$ enforces that each color remains an independent set in $\mathcal{M}$. The third problem is a version of weighted matroid coloring, where elements have different valuations for different colors.

---

[3]A function $f : 2^E \to \mathbb{R}_{\geq 0}$ is submodular if for all $A, B \subseteq E$, we have $f(A \cup B) \leq f(A) + f(B) - f(A \cap B)$. It is monotone if $f(A) \leq f(B)$ whenever $A \subseteq B$.

[4]This assumption is without loss of generality, since any e with $f(\{e\}) = 0$ can be removed.

[5]For our results this assumption is not used in settings where $v_e = b_e$ for all e, generalizing results for AdWords.

1.2 Our Contributions

We introduce the online submodular assignment problem, which encompasses many online assignment problems. Some of these are well-known, including vertex- and edge-weighted bipartite matching, AdWords, and GAP. Others, such as laminar restricted matching, matroid coloring, and coflow assignment, have not been solved previously. Not only do we get optimal competitive ratios for online SAP in several settings, but in doing so we develop a novel framework for handling submodular constraints in online assignment. We consider the development of this machinery to be the primary contribution of our work, as we believe it may be broadly useful for future applications to problems with similar structure.

Our first main theorem concerns the fractional version of online SAP.

Theorem 1.1. *There exists a deterministic $(1-1/e)$-competitive algorithm for the fractional Online Submodular Assignment problem.*

We note that the $(1 - 1/e)$ competitive ratio is tight, as there is a matching upper bound even for the special case of fractional online bipartite matching [Fei19].

Next, we show that our fractional algorithm can be adapted to an integral algorithm under a "small bids" assumption. This type of assumption is often made in the AdWords setting [MSVV07, FKM⁺09], where the costs (or "bids") are assumed to be small compared to the budgets of the advertisers. Specifically, the cost b_e of an offline vertex is assumed to at most a ε-fraction of the total budget of its offline vertex for some $\varepsilon > 0$. We note that it is still an open problem to determine the optimal competitive ratio for integral AdWords without the assumption of small bids; only recently have researchers developed an algorithm that achieves a competitive ratio better than $1/2$ [HZZ20].

Our next result concerns the integral version of online SAP under a small bids assumption about the marginal functions $f_T(S) := f(S \cup T) - f(T)$. This generalizes the small bids assumption for AdWords.

Assumption 1.2 (Small Bids)**.** *Assume there exists some $\varepsilon > 0$ such that for all $e \in E$ and $T \subseteq E$ with $f_T(\{e\}) > 0$, we have $b_e \leq \varepsilon f_T(\{e\})$.*

Theorem 1.3. *Under the small bids assumption (Assumption 1.2), there is a deterministic integral algorithm for online SAP which is $(1 - \varepsilon)^2 \cdot \left(1 - \frac{1}{e}\right)$-competitive.*

In addition to our result in the small bids setting, we also obtain $1 - 1/e$ competitiveness in a special case of the integral setting without the need for a small bids assumption. Suppose that the submodular function f is the rank function of a cross-product matroid $\mathcal{M} := \mathcal{M}_1 \times \ldots \times \mathcal{M}_n$, and each part Q_j contains at most one element from each $\mathcal{M}_i$. Then, assuming $b_e = v_e = 1$ for all e, this case of Online SAP is equivalent to the Online Submodular Welfare Maximization problem where the agents have matroid rank valuations.

The Online Submodular Welfare Maximization Problem (OSWM), is the problem of assigning m indivisible items, which arrive online, to n agents with utility functions $f_i : 2^{[m]} \to \mathbb{R}_{\geq 0}$. Each utility function f_i is assumed to be a monotone, submodular function on $[m]$. The goal is to find an assignment $\sigma : [m] \to [n]$ that maximizes the total welfare of the agents, i.e., the sum of the utilities $\sum_{i=1}^{n} f_i(\sigma^{-1}(i))$. In the offline setting, a $1 - 1/e$ approximation algorithm for OSWM can be achieved using an algorithm for Monotone Submodular Maximization subject to a matroid constraint [CCPV11]. Surprisingly however, Kapralov, Post, and Vondrák [KPV13] show that,

in the online setting, achieving an competitive ratio greater than $1/2$ in polynomial time ($1/2$ is achieved trivially with the Greedy algorithm) is impossible unless NP = RP. By providing an integral algorithm for Online SAP in this special case, we are able to show that this barrier can be circumvented if the class of monotone submodular utility functions is restricted to rank functions of a matroid over ground set $[m]$.

Theorem 1.4. *There exists a randomized polynomial time $(1-1/e)$-competitive algorithm for Online Submodular Welfare Maximization when the agents' utility functions are matroid rank functions.*

This setting captures (integral) online matroid coloring. Indeed, given a matroid $\mathcal{M}$ we seek to color online, each item represents an edge and each agent represents a color class. Thus, our result implies a $(1 - \frac{1}{e})$-competitive algorithm for integral online matroid coloring.

1.3 Technical Overview

We primarily examine online submodular assignment in the fractional setting, as the same techniques yield results for the integral setting with small bids. For this problem, we will employ a continuous pricing-based allocation, in which we place prices p_e on elements $e \in E$ and allocate continuously to the element maximizing utility $v_e - p_e$. This approach is similar to algorithms by [MSVV07], [FKM+09], and [KP00], which have yielded $(1 - 1/e)$-competitive algorithms for a wide range of online matching problems, including edge weighted online bipartite matching, AdWords, and GAP. Each of these can be interpreted as general forms of the classic water filling algorithm for online bipartite matching.

In standard online matching, the water-filling algorithm keeps track of a water level for each offline vertex, which is the fractional amount it has been filled so far. Upon the arrival of each online vertex, the algorithm continuously sends water to its neighbors with the current lowest water level, until either all neighbors have been filled or the arriving vertex has been depleted.

Submodular Water Levels The key challenge in obtaining a water-filling type algorithm for online SAP is adapting the concept of "water levels" from prior work on matchings, to a general submodular constrained linear program. Intuitively, given a fractional allocation $\{x_e\}_{e \in E}$, we would like to assign a water level w_e to element e which represents how "filled" it is under the allocation x. This water level is then used to determine the price of each element, with a higher water level leading to a higher price.

In online bipartite matching, the water level of an edge (i, j) (where i is offline and j is online), is the fraction of vertex i's capacity occupied by x, namely $\sum_{j' \sim i} x_{ij'}$. In the submodular assignment setting, edges become elements $e \in E$, but there is no longer a notion of unit-capacity vertices. Suppose momentarily that $b_e = 1$ for all e, for ease of presentation. Then each e is involved in many submodular constraints of the form $x(S) \leq f(S)$ for each $S \ni e$, and it is not immediately clear how these constraints should be aggregated into a single water level w_e.

Nevertheless, we show that there exist such "submodular water levels" that can be computed efficiently, and that they satisfy several key properties that make a water-filling type algorithm possible. We present three different ways of describing the submodular water levels.

1. **A combinatorial description.** Water levels can be viewed as a measure of density. We describe an iterative process in Algorithm 1 which finds a densest set under x (i.e. the set with the least multiplicative slack in its constraint), contracts this set, and repeats until all elements

are contracted. The density at which an element gets contracted is the water level of that element. This algorithm also demonstrates that the level sets of w correspond to a weighted version of the *principal partition sequence* studied in [Nar91, Fuj08] for poly-matroids.

2. **A max-min relation.** The water level of an individual element e can also be described directly via a simple max-min formula $\max_{S \ni e} \min_T \frac{x(S \setminus T)}{f_T(S)}$, where f_T denotes the *contraction* of f by T. This definition exhibits a surprising minimax theorem showing that the water level is also $\min_{T \not\ni e} \max_S \frac{x(S \setminus T)}{f_T(S)}$, and moreover, a "saddle point" pair of optimizing sets S^*, T^* for both of these expressions can be derived from the combinatorial description of water levels. Importantly, this definition immediately demonstrates that the water level of each element is a continuous piecewise linear function of x.

3. **A convex program.** The concept of water levels also arise from market equilibria. Jain and Vazirani [JV10] introduced a notion of *submodular utility allocation (SUA) markets* as a generalization of linear Fisher markets [NRTV07, Chapter 5]. In a submodular utility allocation market, we have some items A, each with weight m_a. We would like to fractionally select a set of items, with the goal of maximizing the Nash social welfare of the items; however, the set of items picked must be feasible in a poly-matroid defined by a submodular function f. It turns out the water level vector is precisely the optimal solution of an SUA market.

Once we have our notion of water levels defined, we can define an algorithm for online SAP which is inspired by prior work in matching settings. At a high level, when a part Q_j arrives, we assign a utility to each of its elements depending on its value and current water level. We make a small increase to the allocation of the highest utility element, and continue this process until we can no longer increase any element while preserving feasibility.

The second key technical challenge in performing a water-filling type analysis is understanding how the water levels change throughout the algorithm. We use a primal-dual framework, as in [DJK13], to analyze the approximation ratio of our algorithm. In this framework, the algorithm maintains a primal solution at every step, and for the sake of analysis, we continuously maintain a dual solution in parallel. This dual solution will depend on the current water levels of the primal allocation. Critically, we must prove that throughout the algorithm, the dual value is at most the primal value and the dual solution is approximately feasible. To perform such analysis, we need a strong structural understanding of the water level vector w and how it changes with x. We show that the vector w is remarkably well-behaved, and each of our three viewpoints on water levels sheds light on the properties that we will use in our analysis:

- **Monotonicity and Continuity.** For each $e \in E$, its water level w_e is monotone and continuous in the allocation vector x (Proposition 3.4).

- **Feasibility Indication.** The water level vector w indicates the feasibility of allocation vector x (Proposition 3.5). In particular, x is feasible if and only if the water levels are at most 1.

- **Locality.** Changing the allocation x_e of some element by a small amount can only affect the the water level of elements whose water level is equal to e's water level.

- **Duality.** The water levels satisfy a duality property with the Lovász extension of f (Proposition 3.6). That is $L_f(w) = \sum_{e \in E} x_e$. This property can be derived from the KKT conditions of the convex program formulation of water levels.

These properties allow us to keep track of how the dual objective of SAP changes with incremental changes of the primal and dual assignments. Specifically, careful setting of the dual ensures that approximate feasibility is maintained. Moreover, we prove a derivative formula for the Lovász extension of f in terms of the current water levels (Lemma 3.8) which allows us to upper bound the rate of change of the dual objective by that of the primal.

OSWM with Matroid Rank Valuations   Our algorithm for Online Submodular Welfare Maximization for matroidal rank valuations is a natural generalization of the classical RANKING algorithm for online bipartite matching. We interpret the arriving nodes as items which we assign to the offline nodes (the agents). At the very beginning of the algorithm, we randomly permute the agents. When an item arrives, we assign it to the "available" agent of highest priority.

In the bipartite matching case, and even for online b-matching (studied in [AS21]), whether an agent is available simply depends on how many items have been allocated to this agent so far. In our setting, we say that an item is available to an agent if the agent's marginal utility for the item is non-zero, given the items already allocated to this agent. Hence, the availability of an agent depends on the item being considered and changes throughout the course of the algorithm.

Despite this, we show that our Matroidal RANKING algorithm always yields a tight $(1 - \frac{1}{e})$-competitive allocation in expectation. Similar to the analysis in [DJK13], we construct an approximately feasible dual solution which lower bounds the welfare of our allocation. The main technical hurdle is to define a suitable threshold r_{ij}^* for each agent-item pair (i, j) so that the dual assignment (which is constructed online) is approximately feasible, and whose objective value matches the welfare of our online assignment.

Finally, we show that our algorithm can be extended in a natural way to the case where each agent has a non-negative weight, and we seek to maximize the weighted sum of their utilities.

1.4   Related Work

Online Matching   There is an extensive line of work on online matching, starting with the work of Karp, Vazirani and Vazirani [KVV90], who gave a $(1 - 1/e)$-competitive algorithm for online bipartite matching in the adversarial order setting. The same competitive ratio was extended to the vertex-weighted setting in [AGKM11], and further to the vertex-weighted b-matching setting in [AS21]. Devanur, Jain, and Kleinberg [DJK13] showed how the results in [KVV90] and [AGKM11] could be derived using the online primal-dual framework, which unified and simplified the existing analyses. While the RANKING algorithm requires $O(n \log n)$ bits of randomness, Buchbinder, Naor, and Wajc [BNW23] provide a randomized rounding scheme requiring only $(1 \pm o(1)) \log \log n$ bits of randomness.

For edge-weighted online bipartite matching, [FHTZ20] were the first to break the $\frac{1}{2}$-competitive barrier. This has been subsequently refined in [SA21, GHH+21, BC21] to a 0.5368-competitive ratio. Online bipartite matching has also been studied in the vertex-arrival Bayesian setting, also known as the *Ride Hail* problem; [PPSW21] give a better-than-$1/2$ approximation for this setting.

Online Matroid Intersection   Offline matroid intersection captures a broad class of combinatorial problems and has been well studied in both polyhedral theory and algorithms (for a survey on this topic, see [Sch03, Chapter 41]). In [GS17], Guruganesh and Singla study an online matroid intersection problem, with edges arriving in random order, beating the $1/2$-competitive ratio

achieved by Greedy. This was very recently improved by Huang and Sellier [HS23] to $2/3 + \varepsilon$. In [BGH$^+$23], the part-arrival online matroid intersection model is considered. They instead study the problem of *maintaining* a max cardinality independent set, while minimizing recourse.

Offline Submodular Welfare Maximization The offline variant of Submodular Welfare Maximization problem can be cast as a problem of maximizing a monotone submodular function subject to a (partition) matroid constraint. While a simple greedy algorithm achieves an approximation ratio of $\frac{1}{2}$, [CCPV11] gave an improved $(1 - 1/e)$-approximation using the Continuous Greedy algorithm followed by Pipage Rounding. There can be no $(1 - 1/e + \varepsilon)$-approximation unless P=NP since this problem captures max-k-cover as a special case [Fei98].

When the monotone submodular function is the rank function of a matroid, the welfare maximizing offline allocation can be computed optimally. These allocations have been well-studied in the context of designing socially optimal allocations which satisfy certain desirable fairness constraints [VZ23, DFSH23]. For example, in [BCIZ21] it is shown that an optimal allocation which is envy-free up to one item (EFX) exists and can be computed efficiently. In [BEF21] they study the case of valuations which have binary marginals, but which are not necessarily submodular [BEF21]. They also give truthfulness guarantees for private valuations, which has been further studied in [BV22].

Principal Partition The *principal partition* of a matroid is related to our definition of water levels. It is precisely the nested sets found in Algorithm 1, when x is the all ones vector. There have been several works studying principal partitions, including generalizations to arbitrary vectors $x \in \mathbb{R}^E$ and extensions from rank functions to submodular functions [Nar91, Fuj08]. The objects studied in [Nar91] and [Fuj08] are closely related to our water levels; in our work, we (1) shift perspective from the family of nested sets in [Fuj08] to the properties of a single vector w, and (2) study properties of how w changes dynamically with the weights given by x, such as monotonicity (Proposition 3.4), duality (Proposition 3.6), and locality (Proposition 3.7). These properties are clearly visible with our new perspective. We make the novel connection between principal partitions and water-filling in online bipartite matching.

The principal partition has been used for constant competitive algorithms in the matroid secretary problem under random assignment [Sot13]. Huang and Sellier [HS23] also use the principal partition to get an improved approximation ratio of $2/3 - \varepsilon$ for online matroid intersection with random order arrivals in a stream. Chandrasekaran and Wang [CW23] use the principal partition sequence for improved approximations in the submodular k-partition problem.

1.5 Organization of the Paper

First, in Section 2, we introduce some preliminaries. Next, in Section 3, we develop a theory of water levels and provide three equivalent formulations. In Section 4, we look at a special and illuminating case of online SAP: online matroid intersection with part arrivals. We give a water-filling algorithm for fractional online poly-matroid intersection and prove that it achieves a competitive ratio of $1 - 1/e$. Section 5 gives the general case analysis for fractional online SAP, as well as online SAP with a small bids assumption. In Section 6, we give a $(1 - 1/e)$-competitive algorithm for (integral) Online Submodular Welfare Maximization with matroidal utilities.

2 Preliminaries

We analyze the algorithm for fractional online SAP via the primal-dual framework. The primal linear program (LP) describing Online Submodular Assignment, as well as its dual are:

$$
\begin{array}{ll}
\max & \sum_{e \in E} v_e x_e \\
\text{s.t.} & \sum_{e \in Q_j} x_e \leq 1, \qquad \forall j \in [n] \\
& \sum_{e \in S} b_e x_e \leq f(S), \qquad \forall S \subseteq E \\
& x_e \geq 0, \qquad \forall e \in E.
\end{array}
\qquad
\begin{array}{ll}
\min & \sum_{S \subseteq E} f(S) \cdot \alpha_S + \sum_{j=1}^{n} \beta_j \\
\text{s.t.} & b_e \sum_{S \ni e} \alpha_S + \beta_{j(e)} \geq v_e, \qquad \forall e \in E \\
& \alpha_S, \beta_j \geq 0, \qquad \forall S \subseteq E, \, j \in [n].
\end{array}
$$

In the dual, $j(e)$ denotes the index j for which $e \in Q_j$. The solution to the primal LP is the (fractional) optimal *offline* solution to a given instance, and we use OPT to denote its value. By strong duality, the optimal values of the primal and dual LPs are the same.

It will be useful for our analysis to re-write the dual objective in a different form. Let us define $\gamma \in \mathbb{R}_{\geq 0}^E$ by $\gamma_e := \sum_{S \ni e} \alpha_S$. By a standard uncrossing argument, we may assume that the optimal dual α is supported on a nested family of sets. Thus, we can recover the part of the objective $\sum_{S \subseteq E} f(S) \cdot \alpha_S$ from γ as

$$
\sum_{S \subseteq E} f(S) \cdot \alpha_S = \int_0^\infty f(\{e \in E : \gamma_e \geq t\}) \, dt. \tag{1}
$$

The right-hand-side integral is exactly the *Lovász extension* L_f of a submodular function f. This is a natural continuous extension of the submodular function f which has been studied in many contexts. Although equivalent formulations exist, it will be convenient for us to use the following definition.

Definition 2.1. *Let f be a monotone submodular function on E with $f(\varnothing) = 0$. The Lovász extension $L_f : \mathbb{R}_{\geq 0}^E \to \mathbb{R}$ of f is*

$$
L_f(w) := \int_0^\infty f(\{e \in E : w_e \geq t\}) \, dt.
$$

With this definition, we can write our dual in terms of γ as follows.

$$
\begin{array}{ll}
\min & L_f(\gamma) + \sum_{j=1}^{n} \beta_j \\
\text{s.t} & b_e \gamma_e + \beta_{j(e)} \geq v_e, \qquad \text{for all } e \in E \\
& \gamma, \beta \geq 0.
\end{array}
$$

3 Water Level Machinery

Recall that, we want to assign each element $e \in E$ a water level $w_e = w_e^{(x)}$ which depends on the current allocation x, where x satisfies the submodular constraints given by f:

$$
x(S) \leq f(S) \quad \forall S \subseteq E.
$$

Notice that, for the sake of defining water levels, we are assuming unweighted costs (i.e. $b_e = 1$). This removes clutter from our definitions, as we can simply add weights later by taking $w^{(bx)}$.

To understand how we define this water level vector $w = w^{(x)} \in \mathbb{R}_{\geq 0}^E$ of allocation x, we first enumerate several properties that the water levels should satisfy in order for the water-filling algorithm to work as it does for online bipartite matching:

1. (Monotonicity) $w^{(x)}$ is coordinate-wise non-decreasing in x.

2. (Indication of Feasibility) $w^{(x)} \leq 1$ if and only if $x(S) \leq f(S)$ for all $S \subseteq E$.

3. (Locality) If $w_{e_1}^{(x)} \neq w_{e_2}^{(x)}$, then $\frac{\partial w_{e_1}^{(x)}}{\partial x_{e_2}} = 0$.

4. (Duality) $L_f(w^{(x)}) = \sum_{e \in E} x_e$.

Monotonicity and feasibility indication are natural properties which intuitively require that w_e is an indicator of how close x_e is to being part of a tight constraint. The need for locality and duality is less obvious, but they are important for the details of the primal-dual analysis of water-filling. Specifically, this is because the dual value γ_e of an element $e \in E$ will be defined as a function of the water level w_e. The locality and duality properties are needed to relate increases in the dual objective term $L_f(\gamma)$ to increases to the primal objective $\sum_{e \in E} x_e$ as the algorithm progresses.

3.1 Definition of Water Levels and Equivalent Formulations

In order to define a water level vector that satisfies our desired properties, it will be convenient (and enlightening) to provide both algorithmic and static definitions, which we will prove are equivalent. By juggling three different definitions, we are able to provide succinct proofs of the four key properties of water levels.

First, let's consider a naive construction $\widetilde{w}_e^{(x)}$. For $x \in \mathbb{R}_{\geq 0}^E$, we define $\widetilde{w}_e^{(x)} = \max_{S \ni e} \frac{x(S)}{f(S)}$, i.e. the maximum density of a poly-matroid constraint involving x_e. Such a definition clearly satisfies monotonicity and indication of feasibility. However, this definition does not capture the water levels from the classic bipartite matching setting (i.e. in a partition matroid), and we can see this already in the simple setting of $E = \{1, 2\}$ and $f(S) = |S|$. In such a setting, the poly-matroid effectively only has the constraints $0 \leq x_1, x_2 \leq 1$, so we intuitively should let $w_e = x_e$. However, notice that if $x_1 < x_2$, then we have $\widetilde{w}_1 = \max_{S \ni 1} \frac{x(S)}{f(S)} = \frac{x_1 + x_2}{2}$. We see that this construction deviates from what we expect, and indeed we also find that our desired locality and duality properties are not satisfied. This problem arises because the heavier element x_2 influences the density of the densest constraint on x_1, despite the two variables being functionally independent.

The critical insight is that we can prevent this undesirable behavior by contracting sets with larger density before assigning the values of w_e to sets with lower density. This inspires the following formulation of water levels in Definition 3.1.

Definition 3.1. *The* water level vector $w^{(x)} \in \mathbb{R}_{\geq 0}^E$ *with respect to an allocation x in $\mathbb{R}_{\geq 0}^E$ is defined as*

$$w_e^{(x)} := \max_{S \ni e} \min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \frac{x(S \setminus T)}{f_T(S)}$$

where $f_T(S)$ denotes the contracted function $f(S \cup T) - f(T)$.

Not only does this modification fix the problems with our naive definition, but it reveals a great degree of hidden structure. First, it happens to be the case that the min and max in Definition 3.1 are reversible, i.e. $w_e^{(x)} = \min_T \max_S \frac{x(S \setminus T)}{f_T(S)}$. We will prove this by showing the existence of a saddle point S^*, T^*:

$$\min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \frac{x(S^* \setminus T)}{f_T(S^*)} = \frac{x(S^* \setminus T^*)}{f_{T^*}(S^*)} = \max_{S \ni e} \frac{x(S \setminus T^*)}{f_{T^*}(S)}. \tag{2}$$

Furthermore, these optimal sets form a nested family. We can find $\varnothing = S_0 \subseteq S_1 \subseteq \cdots \subseteq S_k = E$ such that for each $e \in S_{\ell+1} \setminus S_\ell$, the pair $(S_{\ell+1}, S_\ell)$ is a saddle point for $w_e^{(x)}$ in the max-min expression of Definition 3.1. We find the nested family via an intuitive and efficiently computable combinatorial description of water levels, Algorithm 1.

Algorithm 1: A Combinatorial Presentation of Water Levels

    input : A point $x \in \mathbb{R}_{\geq 0}^E$.

1 Initialize $\ell \leftarrow 0$, $S_0 \leftarrow \varnothing$.

2 while $S_\ell \neq E$ **do**

3   Let $S_{\ell+1}$ be the unique maximal set[6] in

$$\arg \max_{\substack{S \subseteq E \\ S \setminus S_\ell \neq \varnothing}} \frac{x(S \setminus S_\ell)}{f_{S_\ell}(S)},$$

     i.e. the largest densest set over the contracted polymatroid.

4   Let $t_{\ell+1} = \frac{x(S_{\ell+1} \setminus S_\ell)}{f_{S_\ell}(S_{\ell+1})}$ be the density of $S_{\ell+1}$.

5   Set $\widehat{w}_e \leftarrow t_{\ell+1}$ for all $e \in S_{\ell+1} \setminus S_\ell$.

6 $\ell \leftarrow \ell + 1$.

7 return $\widehat{w}$.

Theorem 3.2 (Saddle Point for Water Levels). *For any monotone submodular function $f : 2^E \to \mathbb{R}_{\geq 0}$ with $f(\varnothing) = 0$ and $x \in \mathbb{R}_{\geq 0}^E$, the vector $w = w^{(x)}$ in Definition 3.1 has the property*

$$w_e^{(x)} := \max_{S \ni e} \min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \frac{x(S \setminus T)}{f_T(S)} = \min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \max_{S \ni e} \frac{x(S \setminus T)}{f_T(S)}.$$

Moreover, the output $\widehat{w}$ of Algorithm 1 is equal to w.

For readability, we delay the proof of the min-max property of Definition 3.1 and its equivalence to Algorithm 1 until Section 3.3.

We also find an unexpected connection to market equilibria. Jain and Vazirani [JV10] introduced a notion of *submodular utility allocation markets* which can be described with the following convex

---

[6]The maximal such set is unique due to the submodularity of f.

program.

$$\max_{u} \quad \sum_{e \in E} m_e \log u_e$$
$$\text{s.t.} \quad \sum_{e \in S} u_e \leq f(S), \quad \forall S \subseteq E \qquad (\alpha_S) \qquad \qquad \text{(SUA)}$$
$$u_e \geq 0.$$

It turns out the water levels of an allocation x can be computed from the optimal utilities of an SUA market where each element e has weight $m_e := x_e$. Algorithm 1 gives us optimal duals to (SUA), which we also prove in Section 3.3.

Theorem 3.3. *Consider the vector $\widehat{w}$, the nested sets $S_1 \subset \cdots \subset S_L$, and the levels $t_1, \ldots, t_L$ generated by Algorithm 1. Then $t_1 > \cdots > t_L \geq 0$. Moreover, if we define*

$$\widehat{\alpha}_{S_L} := t_L$$
$$\widehat{\alpha}_{S_\ell} := t_\ell - t_{\ell+1} \quad \ell = 1, \ldots, L-1$$

and $\widehat{\alpha}_S = 0$ for all other $S \subseteq E$, then, $\widehat{u}_e = \frac{x_e}{\widehat{w}_e}$ is an optimal primal solution to (SUA) and $\widehat{\alpha}_S$ is an optimal dual solution.

3.2 Key Properties of Water Levels

Armed with the characterizations of water levels, we show they satisfy the desired properties.

Proposition 3.4 (Monotone and Continuous). *The vector $w^{(x)}$ is coordinate-wise non-decreasing in x. Furthermore, $w^{(x)}$ is continuous with respect to x.*

Proof. Both properties follow immediately from Definition 3.1, since the $w_e^{(x)}$ is a maximum of minimums over monotone increasing linear functions on x. $\qquad \square$

Proposition 3.5 (Indication of Feasibility). *$w^{(x)} \leq 1$ if and only if $x(S) \leq f(S)$ for all $S \subseteq E$.*

Proof. This follows from Algorithm 1. If $w^{(x)} \leq 1$, then in particular $t_1 = \max_{S \subseteq E} \frac{x(S)}{f(S)} \leq 1$, which means $x(S) \leq f(S)$ for all $S \subseteq E$. Conversely, if $x(S) \leq f(S)$ for all $S \subseteq E$ then clearly $t_1 \leq 1$. Moreover, the densities t_ℓ are decreasing by Theorem 3.3, implying $t_\ell \leq 1$ for all ℓ. Thus $w^{(x)} \leq 1$. $\qquad \square$

Proposition 3.6 (Duality). *For any $x \in \mathbb{R}_{\geq 0}^E$, we have $L_f(w^{(x)}) = \sum_{e \in E} x_e$.*

Proof. For this, we refer to the convex program formulation of water levels from Theorem 3.3. Taking the optimal primal/dual pair u_e, α_S from Theorem 3.3, the complementary slackness conditions of (SUA) give $\alpha_S \sum_{e \in S} u_e = \alpha_S f(S)$ for each $S \subseteq E$. Summing over those S with $\alpha_S > 0$, we get

$$\sum_{\ell=1}^L \alpha_{S_\ell} f(S_\ell) \overset{(a)}{=} \sum_{\ell=1}^L \alpha_{S_\ell} \sum_{e \in S_\ell} u_e \overset{(b)}{=} \sum_{\ell=1}^L \alpha_{S_\ell} \sum_{e \in S_\ell} \frac{x_e}{w_e} = \sum_{e \in E} \left(\sum_{\ell: S_\ell \ni e} \alpha_{S_\ell} \right) \cdot \frac{x_e}{w_e} \overset{(c)}{=} \sum_{e \in E} x_e.$$

Here, (a) is using $\alpha_S \sum_{e \in S} u_e = \alpha_S f(S)$ for each $S \subseteq E$, (b) is because $u_e = \frac{x_e}{w_e}$ by Theorem 3.3, and (c) is because $\sum_{\ell: S_\ell \ni e} \alpha_{S_\ell} = w_e$, using the fact that the sets S_ℓ are nested and the definition of α_{S_ℓ} in Theorem 3.3. Finally, the LHS is equal to $L_f(w^{(x)})$ by (1) and using $\sum_{\ell: S_\ell \ni e} \alpha_{S_\ell} = w_e$. $\qquad \square$

Proposition 3.7 (Locality). *For $x \in \mathbb{R}_{\geq 0}^E$ and any $e_1, e_2 \in E$ with $w_{e_1}^{(x)} \neq w_{e_2}^{(x)}$, we have $\frac{\partial w_{e_1}^{(x)}}{\partial x_{e_2}} = 0$.*

Proof. This follows from the continuity of $w^{(x)}$ with respect to x (implied by Definition 3.1) and the algorithmic definition given by Algorithm 1. Let $w_{e_2}^{(x)} = t_\ell$, for some step ℓ, where $t_1, \ldots, t_L$ are the values from Algorithm 1 on input x, and let $\varepsilon := \frac{\min(t_{\ell-1} - t_\ell, t_\ell - t_{\ell+1})}{2}$. By continuity, we may choose $\delta_\varepsilon > 0$ small enough so that for any $\delta \in (-\delta_\varepsilon, \delta_\varepsilon)$ and $y := x + \delta \mathbb{1}_{e_2}$, we have

$$w_e^{(y)} \in (w_e^{(x)} - \varepsilon, w_e^{(x)} + \varepsilon)$$

for all $e \in E$. It suffices to show for any such y that $w_{e_1}^{(y)} = w_{e_1}^{(x)}$. Consider for each t the sets $E_{\geq t}^{(x)} := \{e : w_e^{(x)} \geq t\}$ and $E_{\geq t}^{(y)} := \{e : w_e^{(y)} \geq t\}$. Then by our choice of ε, we have

$$E_+ := E_{\geq t_\ell + \varepsilon}^{(x)} = E_{\geq t_\ell + \varepsilon}^{(y)}, \text{ and}$$
$$E_- := E_{\geq t_\ell - \varepsilon}^{(x)} = E_{\geq t_\ell - \varepsilon}^{(y)}.$$

Note that $e_2 \in E_- \setminus E_+$.

It is clear then that the first $\ell - 1$ steps of the Algorithm 1 on x and on y are identical, with $S_{\ell-1} = E_+$ in both cases. This follows because x and y differ only in their value at e_2, and $w_{e_2}^{(y)} < w_{e_2}^{(x)} + \varepsilon = t_\ell + \varepsilon$, so the algorithm assigns water levels to all elements $e \in E_+$ before assigning a water level to e_2. Hence, if e_1 has $w_{e_1}^{(x)} > w_{e_2}^{(x)}$, we have $w_{e_1}^{(y)} = w_{e_1}^{(x)}$ as desired.

In the other case that e_1 has $w_{e_1}^{(x)} < w_{e_2}^{(x)}$, observe that $e_1 \in E \setminus E_-$. Moreover, for both inputs x and y, Algorithm 1 will have some step $\ell' \geq \ell$ (ℓ' may differ for x and y) at which $S_{\ell'} = E_-$. Since $e_2 \in E_-$, and x and y differ only at e_2, then every future step of the algorithm is identical for the two inputs. In particular, $w_{e_1}^{(y)} = w_{e_1}^{(x)}$. $\square$

In addition, a key consequence of the duality and locality properties is the following "chain-rule" lemma about the partial derivatives of $L_f(G(w^{(x)}))$ with respect to entries of x. Such a lemma will be useful in the primal-dual analysis of online SAP.

Lemma 3.8 (Water Level Chain Rule). *If $G : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a non-decreasing differentiable function with continuous derivative $G' = g$, then for all $x \in \mathbb{R}_{\geq 0}^E$ and $e \in E$,*

$$\frac{\partial(L_f(G(w^{(x)})))}{\partial x_e} = g(w_e^{(x)}),$$

where $G(w^{(x)}) := (G(w_e^{(x)}))_{e \in E}$.

Proof. For given $x \in \mathbb{R}_{\geq 0}^E$ and $e \in E$, let $y = x + \varepsilon \cdot \mathbb{1}_e$. Then expanding L_f as an integral, applying a change of variables, and invoking the monotonicity of G, we have

$$
\begin{aligned}
L_f(G(w^{(y)})) - L_f(G(w^{(x)})) &= \int_0^\infty \left(f(\{e' : G(w_{e'}^{(y)}) \geq t\}) - f(\{e' : G(w_{e'}^{(x)}) \geq t\}) \right) dt \\
&= \int_0^\infty \left(f(\{e' : G(w_{e'}^{(y)}) \geq G(u)\}) - f(\{e' : G(w_{e'}^{(x)}) \geq G(u)\}) \right) g(u) du \\
&= \int_0^\infty \left(f(\{e' : w_{e'}^{(y)} \geq u\}) - f(\{e' : w_{e'}^{(x)} \geq u\}) \right) g(u) du.
\end{aligned}
$$

Now, we crucially use locality (Proposition 3.7) to claim that $\{e' : w_{e'}^{(y)} \geq u\} = \{e' : w_{e'}^{(x)} \geq u\}$ for any $u \notin [w_e^{(x)}, w_e^{(y)}]$. This follows because for each $e' \in E$ and every $z = x + \delta \cdot \mathbb{1}_e$ for $\delta \in [0, \varepsilon]$, we have $\frac{\partial w_{e'}^{(z)}}{\partial z_e} = 0$ unless $w_{e'}^{(z)} = w_e^{(z)}$. In particular, if $w_{e'}^{(x)}$ lies outside the range $[w_e^{(x)}, w_e^{(y)}]$, then the water level of e' never changes as we move z from x to y. Likewise, the water level of any e' whose water level lies within the range $[w_e^{(x)}, w_e^{(y)}]$ may change, but it cannot increase beyond $w_e^{(y)}$. So for each $e' \in E$, either $w_{e'}^{(y)} = w_{e'}^{(x)} \notin [w_e^{(x)}, w_e^{(y)}]$ or $w_{e'}^{(x)}, w_{e'}^{(y)} \in [w_e^{(x)}, w_e^{(y)}]$.

Using this fact, we may restrict the above integral to the range $u \in [w_e^{(x)}, w_e^{(y)}]$. Together with the intermediate value theorem, we have that there exists some $\hat{u} \in [w_e^{(x)}, w_e^{(y)}]$ such that

$$L_f(G(w^{(y)})) - L_f(G(w^{(x)})) = \int_{w_e^{(x)}}^{w_e^{(y)}} \Big(f(\{e' : w_{e'}^{(y)} \geq u\}) - f(\{e' : w_{e'}^{(x)} \geq u\}) \Big) g(u) du$$

$$= g(\hat{u}) \cdot \int_{w_e^{(x)}}^{w_e^{(y)}} \Big(f(\{e' : w_{e'}^{(y)} \geq u\}) - f(\{e' : w_{e'}^{(x)} \geq u\}) \Big) du$$

$$= g(\hat{u}) \cdot (L_f(w^{(y)}) - L_f(w^{(x)})).$$

From duality (Proposition 3.6), we know that $L_f(w^{(y)}) - L_f(w^{(x)}) = \sum_{e' \in E}(y_{e'} - x_{e'}) = \varepsilon$. Therefore, we ultimately get

$$\frac{L_f(G(w^{(y)})) - L_f(G(w^{(x)}))}{\varepsilon} = g(\hat{u}).$$

Taking the limit as $\varepsilon \to 0$, we have $\hat{u} \to w_e^{(x)}$ since $w_e^{(x)} \leq \hat{u} \leq w_e^{(y)}$. By continuity of g, we finally see that

$$\frac{\partial (L_f(G(w^{(x)})))}{\partial x_e} = \lim_{\varepsilon \to 0} \frac{L_f(G(w^{(y)})) - L_f(G(w^{(x)}))}{\varepsilon} = g(w_e^{(x)}). \qquad \square$$

3.3 Proofs of Equivalence of Water Level Definitions

In this sub-section, we prove the combinatorial decomposition in Algorithm 1 and the SUA market formulation both produce the water levels vector defined in Definition 3.1.

Theorem 3.2 (Saddle Point for Water Levels). *For any monotone submodular function $f : 2^E \to \mathbb{R}_{\geq 0}$ with $f(\varnothing) = 0$ and $x \in \mathbb{R}_{\geq 0}^E$, the vector $w = w^{(x)}$ in Definition 3.1 has the property*

$$w_e^{(x)} := \max_{S \ni e} \min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \frac{x(S \setminus T)}{f_T(S)} = \min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \max_{S \ni e} \frac{x(S \setminus T)}{f_T(S)}.$$

Moreover, the output $\widehat{w}$ of Algorithm 1 is equal to w.

Proof. To show the desired min-max property, it suffices to show that for any $e \in E$, there exist sets S^*, T^* such that $e \in S$ and $f_T(\{e\}) \neq 0$ and

$$\min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \frac{x(S^* \setminus T)}{f_T(S^*)} = \frac{x(S^* \setminus T^*)}{f_{T^*}(S^*)} = \max_{S \ni e} \frac{x(S \setminus T^*)}{f_{T^*}(S)}. \tag{3}$$

14

From this, it follows that

$$\max_{\substack{S \ni e}} \min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \frac{x(S \setminus T)}{f_T(S)} \geq \min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \frac{x(S^* \setminus T)}{f_T(S^*)}$$

$$= \frac{x(S^* \setminus T^*)}{f_{T^*}(S^*)}$$

$$= \max_{S \ni e} \frac{x(S \setminus T^*)}{f_{T^*}(S)}$$

$$\geq \min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \max_{S \ni e} \frac{x(S \setminus T)}{f_T(S)}.$$

Since we clearly have $\max_S \min_T \frac{x(S \setminus T)}{f_T(S)} \leq \min_T \max_S \frac{x(S \setminus T)}{f_T(S)}$ (omitting constraints on S, T) it follows that all inequalities above must be equalities.

To show that such sets obeying eq. (3) exist, we will show that they are exactly those given by the family $S_0, S_1, \ldots, S_L$ resulting from Algorithm 1. Specifically, we will show that for each ℓ,

$$\min_{\substack{T \subseteq E \\ f_T(S_\ell) \neq 0}} \frac{x(S_\ell \setminus T)}{f_T(S_\ell)} = \frac{x(S_\ell \setminus S_{\ell-1})}{f_{S_{\ell-1}}(S_\ell)} = \max_{S \setminus S_{\ell-1} \neq \varnothing} \frac{x(S \setminus S_{\ell-1})}{f_{S_{\ell-1}}(S)}. \tag{4}$$

From this, it follows that for each $e \in S_\ell \setminus S_{\ell-1}$, the pair of sets $(S_{\ell-1}, S_\ell)$ form an optimal pair (S^*, T^*) in eq. (3) for e. This not only completes the proof of our min-max property, but also implies $w_e = \frac{x(S_\ell \setminus S_{\ell-1})}{f_{S_{\ell-1}}(S_\ell)} = t_\ell = \widehat{w}_e$.

We proceed will show by induction that eq. (4) holds for each $\ell \geq 1$. Notice that the second equality in eq. (4) holds by choice of S_ℓ in Algorithm 1, so we only need to show the first equality.

For the case $\ell = 1$, suppose some set T has $\frac{x(S_1 \setminus T)}{f_T(S_1)} < t_1$. We may assume $T \subseteq S_1$, since otherwise we can take $T \cap S_1$ since $f_T(S_1) \leq f_{T \cap S_1}(S_1)$. Then we have

$$t_1 = \frac{x(S_1)}{f(S_1)} = \frac{x(S_1 \setminus T) + x(T)}{f_T(S_1) + f(T)}.$$

Since we know $\frac{x(S_1 \setminus T)}{f_T(S_1)} < t_1$, it must then be true that $\frac{x(T)}{f(T)} > t_1$. However, this is impossible as S_1 is chosen to have maximum density.

For the inductive step, let $\ell \geq 1$ and assume eq. (4) holds for all previous steps $\ell' < \ell$. Our reasoning will be similar to the base case, but with some extra steps. As before, take some $T \subseteq E$ with minimum possible value of $\frac{x(S_\ell \setminus T)}{f_T(S_\ell)}$, and suppose $\frac{x(S_\ell \setminus T)}{f_T(S_\ell)} < t_\ell$. We may again assume $T \subseteq S_\ell$. In addition, we assume that there are no $e' \notin T$ with $f_T(\{e'\}) = 0$, since adding such e' to T can only improve the choice of T.

We consider two cases: either $S_{\ell-1} \subseteq T$, or $S_{\ell-1} \setminus T \neq \varnothing$. In the former case we have

$$t_\ell = \frac{x(S_\ell \setminus S_{\ell-1})}{f_{S_{\ell-1}}(S_\ell)} = \frac{x(S_\ell \setminus T) + x(T \setminus S_{\ell-1})}{f_T(S_\ell) + f_{S_{\ell-1}}(T)}.$$

Since we assumed that $\frac{x(S_\ell \setminus T)}{f_T(S_\ell)} < t_\ell$, it must be the case that $\frac{x(T \setminus S_{\ell-1})}{f_{S_{\ell-1}}(T)} > t_\ell$. However, this contradicts the choice of t_ℓ in Algorithm 1 as the maximum density of a set after contracting $S_{\ell-1}$.

Now, consider the second case where $S_{\ell-1} \setminus T$ is nonempty. Pick $e' \in S_{\ell-1} \setminus T$, and let $\ell' < \ell$ be such that $\widehat{w}_{e'} = t_{\ell'}$. We then have

$$\frac{x(S_\ell \setminus T)}{f_T(S_\ell)} = \frac{x(S_\ell \setminus (S_{\ell'} \cup T)) + x(S_{\ell'} \setminus T)}{f_{S_{\ell'} \cup T}(S_\ell) + f_T(S_{\ell'})}.$$

Notice that $\frac{x(S_{\ell'} \setminus T)}{f_T(S_{\ell'})} \geq \frac{x(S_{\ell'} \setminus S_{\ell'-1})}{f_{S_{\ell'-1}}(S_{\ell'})} = t_{\ell'}$ by eq. (4) applied to ℓ', which holds by our inductive hypothesis. Since $\frac{x(S_{\ell'} \setminus T)}{f_T(S_{\ell'})} \geq t_{\ell'} > t_\ell > \frac{x(S_\ell \setminus T)}{f_T(S_\ell)}$, this means that we must have $\frac{x(S_\ell \setminus (S_{\ell'} \cup T))}{f_{S_{\ell'} \cup T}(S_\ell)} < \frac{x(S_\ell \setminus T)}{f_T(S_\ell)}$. However, this contradicts the choice of T.

$\square$

Theorem 3.3. *Consider the vector $\widehat{w}$, the nested sets $S_1 \subset \cdots \subset S_L$, and the levels $t_1, \ldots, t_L$ generated by Algorithm 1. Then $t_1 > \cdots > t_L \geq 0$. Moreover, if we define*

$$\widehat{\alpha}_{S_L} := t_L$$
$$\widehat{\alpha}_{S_\ell} := t_\ell - t_{\ell+1} \quad \ell = 1, \ldots, L-1$$

and $\widehat{\alpha}_S = 0$ for all other $S \subseteq E$, then, $\widehat{u}_e = \frac{x_e}{\widehat{w}_e}$ is an optimal primal solution to (SUA) and $\widehat{\alpha}_S$ is an optimal dual solution.

Proof. We begin with an rephrasing of Algorithm 1. Instead of contracting elements as we did in Algorithm 1, we "freeze" elements in Algorithm 2. Scaling x until some set is saturated is equivalent to measuring the multiplicative slack of sets; therefore, the densities in Line 3 of Algorithm 1 are precisely the time steps at which we freeze a new set of elements in Line 4 of Algorithm 2.

Algorithm 2: An Alternate Combinatorial Presentation of Water Levels

    input : A point $x \in \mathbb{R}_{\geq 0}^n$.

1 Initialize $t = 0$, all elements are considered "unfrozen"

2 **while** *there exists an unfrozen element* **do**

3   Raise t until the vector

$$x^{(t)} = \begin{cases} t \cdot x_e & \text{if } e \text{ is unfrozen} \\ t_{\text{frozen}}(e) \cdot x_e & \text{if } e \text{ is frozen} \end{cases}$$

      has a tight set including at least one unfrozen element.

4   Freeze all the elements in the (unique) largest such tight set S_t of $t \cdot x$.

5   We set all newly frozen elements in S to have $t_{\text{frozen}}(e) := t$.

6   Set $\widehat{w}_e = \frac{1}{t}$ for all $e \in S_t$.

    output: A vector $\widehat{w}$.

We use the KKT conditions to show $\widehat{u}_e$ and $\widehat{\alpha}_S$ are optimal. Denote the Lagrange multipliers for the constraints $\widehat{u}_e \geq 0$ by μ_e. We will set $\mu_e = 0$ if $x_e > 0$ and $\mu_e = w_e$ otherwise. The KKT conditions are as follows:

- *Primal Feasibility:* $\sum_{e \in S} \widehat{u}_e \leq f(S)$ for all $S \subseteq E$.

- *Dual Feasibility:* $\widehat{\alpha}_S \geq 0$ for all $S \subseteq E$.

- *Stationarity Conditions:* For all $e \in E$,

$$\frac{x_e}{\widehat{u}_e} = \sum_{S \ni e} \widehat{\alpha}_S - \mu_e.$$

- *Complementary Slackness:* $\widehat{\alpha}_S > 0$ implies $\sum_{e \in S} \widehat{u}_e = f(S)$ and $\mu_e \cdot \widehat{u}_e = 0$.

Primal feasibility follows from the fact that Algorithm 2 maintains feasibility of $t \cdot x$ on unfrozen elements. Dual feasibility also easily follows since t only rises. If $x_e > 0$, then $\mu_e = 0$ and

$$\frac{x_e}{\widehat{u}_e} = \widehat{w}_e = \sum_{S \ni e} \widehat{\alpha}_S$$

as desired. Otherwise, if $x_e = 0$, then since $\mu_e = \widehat{w}_e$, the stationary condition still holds. Lastly, we check complementary slackness. We have a positive $\widehat{\alpha}_S$ precisely on the sets $E_1, \ldots, E_L$, and so it suffices to check these sets are tight. Indeed, by definition of Algorithm 2, these sets are tight. Lastly, we have that if $x_e > 0$, then $\mu_e = 0$ and that finishes our complementary slackness conditions. □

4 A Warm Up: Online Matroid Intersection

With our new vector of water levels $w^{(x)}$ defined, we will see how they can be used to naturally extend the water-filling paradigm. Before proving Theorem 1.1 for general online SAP, we will see how our techniques can be applied in a simpler setting: fractional online matroid intersection with part arrival.

In *Online Matroid Intersection*, the goal is to maximize the size of a common independent set between two matroids, when the elements are initially unknown and arrive in some online fashion. We focus on the case where one of the matroids is a partition matroid. In particular, suppose $\mathcal{M}$ is an arbitrary matroid and $\mathcal{Q}$ is a partition matroid with parts $Q_1, \ldots, Q_n$, both defined over (an initially unknown) ground set E. We have access to an independence oracle for $\mathcal{M}$ restricted to the elements which have been revealed so far; in other words, $\mathcal{M}$ is known offline. Parts from the partition matroid arrive online. When a part Q_j arrives, the elements in Q_j are revealed, and we immediately and irrevocably choose at most one element from Q_j. The goal is to maximize the cardinality of the set of chosen elements, subject to the set being independent in both matroids. In the fractional version of this problem, instead of choosing one $e \in Q_j$, we select values $x_e \geq 0$ for $e \in Q_j$ so that $x(Q_j) \leq 1$ and x remains feasible in the matroid polytope. Online matroid intersection, and the corresponding fractional problem, are instances of online (fractional) SAP where $v_e = b_e = 1$ for all $e \in E$, and $f(S) := \text{rank}_{\mathcal{M}}(S)$.

Our fractional algorithm for this problem will, upon receiving part Q_j, continuously allocate infinitesimally small dx_e to x_e for some $e \in Q_j$ which has minimum water level, i.e. $e \in \arg\min_{e' \in Q_j} w_{e'}^{(x)}$. This continues until either 1 unit of water has been output by Q_j (the constraint $x(Q_j) = 1$ becomes tight) or no more water can be output because every $e \in Q_j$ has water level $w_e^{(x)} = 1$ (each such e is part of a tight constraint $x(S) = f(S)$). The algorithm then moves onto the next arriving part and repeats the process. We use a primal-dual analysis to prove that the algorithm is $(1 - 1/e)$-competitive.

4.1 Analysis

We proceed with a primal-dual analysis to prove Theorem 1.1 for fractional online matroid intersection. The dual program is:

$$\min \quad L_f(\gamma) + \sum_{j=1}^{n} \beta_j$$

$$\text{s.t} \quad \gamma_e + \beta_{j(e)} \geq 1, \qquad \text{for all } e \in E$$

$$\gamma, \beta \geq 0.$$

where L_f is the Lovász extension of f, and $j(e)$ is defined such that $e \in Q_{j(e)}$. We will construct a set of dual variables based on the primal allocation. Specifically, upon each arrival of Q_j, we start by setting $\beta_j = 0$. Then upon each infinitesimal increase of x_e by dx_e for $e \in Q_j$, we increase γ to maintain

$$\gamma_{e'} := G(w_{e'}) \quad \text{for all } e' \in E$$

and increase β_j by

$$d\beta_j := (1 - g(w_e))dx_e.$$

where $g(x) := e^{x-1}$, and $G(x) := \int_0^x g(t)\,dt = e^{x-1} - e^{-1}$. Observe that since the algorithm only increases the primal allocation x, by Proposition 3.4 the dual variables also only increase as the algorithm progresses.

To show a $1 - 1/e$ competitive ratio, we need to show $(1 - 1/e)$-approximate feasibility of the dual, and that the dual increase is at most the primal increase.

4.1.1 Approximate Feasibility

We will show that, immediately after the allocation to Q_j completes, we have $\gamma_e + \beta_j \geq 1 - 1/e$ for all $e \in Q_j$. Since dual values only increase as the algorithm progresses, this would imply inequality also holds for the final dual values.

Let $w^* = \min_{e \in Q_j} w_e^{(x)}$ be the minimum water level of an element of Q_j immediately following the allocation to Q_j. We claim that $\beta_j \geq 1 - g(w^*)$. To see this, notice that $d\beta_j \geq (1 - g(w^*))dx_e$ at each point in time during the allocation, so we clearly have $\beta_j \geq (1 - g(w^*))\sum_{e \in Q_j} x_e$. If $\sum_{e \in Q_j} x_e = 1$, we have our claim. Otherwise, every $e \in Q_j$ must be involved in a tight offline constraint, so $w^* = 1$. In this case, $1 - g(w^*) = 0 \leq \beta_j$.

Using this claim, we easily obtain for each element $e \in Q_j$,

$$\gamma_e + \beta_j \geq G(w_e) + 1 - g(w^*) \geq G(w_e) + 1 - g(w_e) = 1 - 1/e.$$

4.1.2 Primal equals Dual

To show that the primal objective equals the dual objective, we will show that the rate of change in primal and dual objectives are equal at each instant in the continuous allocation. In the allocation to Q_j, when x_e receives infinitesimal allocation dx_e, the change in the primal objective is exactly dx_e.

Meanwhile, the change in dual objective is $d(L_f(G(w^{(x)}))) + d\beta_j$. By Lemma 3.8, we know that $d(L_f(G(w^{(x)}))) = g(w_e)dx_e$, and by definition of β_j we have $d\beta_j = (1 - g(w_e))dx_e$. Hence, we have that the change in dual is also $g(w_e)dx_e + (1 - g(w_e))dx_e = dx_e$.

4.1.3   Proof of the Main Theorem

The above discussion immediately gives the proof of the main theorem:

Proof of Theorem 1.1 for online matroid intersection. Let x be the primal allocation given by water-filling, and γ, β the associated dual assignment defined above. Since we showed that at each step of the algorithm, $\Delta\text{Primal} \geq \Delta\text{Dual}$, we have that $\sum_e x_e \geq L_f(\gamma) + \sum_j \beta_j$. By approximate feasibility, we have that $\gamma' := \frac{e}{e-1} \cdot \gamma$ and $\beta' := \frac{e}{e-1} \cdot \beta$ together form a feasible dual solution. Finally, positive homogeneity[7] of the Lovász extension and duality together give

$$\sum_e x_e \geq L_f(\gamma) + \sum_j \beta_j$$

$$= \left(1 - \frac{1}{e}\right) \cdot \left(L_f(\gamma') + \sum_j \beta'_j\right)$$

$$\geq \left(1 - \frac{1}{e}\right) \cdot \text{OPT}_{\text{Dual}} = \left(1 - \frac{1}{e}\right) \cdot \text{OPT}. \qquad \square$$

5   General Online Submodular Assignment

In order to motivate our algorithm for online SAP, it will be useful to adopt an economics perspective similar to that presented in [BM08]. We think of each online arrival j as a bidder and each offline element $e \in E$ as a good. Each bidder j desires at most one unit of goods from the set Q_j, and receives utility v_e for each unit of good e received. We, as the seller, allocate a quantity x_e of good e to bidder j for each $e \in Q_j$, but we are also limited the supply constraints $bx(S) \leq f(S)$ for each $S \subseteq E$. Our goal is to maximize the welfare $\sum_{e \in E} v_e x_e$.

In this setting, it will be important to discriminate between items based on their "bang-per-buck" $\frac{v_e}{b_e}$. This is due to the fact that, with free disposal, it can be beneficial to discard an allocation with small bang-per-buck in order to make space for a higher value item. To this end, for $t \geq 0$ define the vector w^t to be

$$w^t := w^{((b_e x_e)_e\, :\, v_e/b_e \geq t)},$$

i.e. the water levels of items when only considering allocations on item e with bang-per-buck at least t.

Using this definition, we give a pricing-based algorithm for allocating x_e values. For each $e \in E$, we place an instantaneous per-unit price on e of

$$p_e := b_e \cdot \int_0^{v_e/b_e} g(w_e^t)dt.$$

Then, upon the arrival of Q_j, bidder j will continuously "purchase" an infinitesimal amount dx_e of good e, for some $e \in Q_j$ which has highest marginal utility, i.e. $e \in \arg\max_{e' \in Q_j}(v_{e'} - p_{e'})$. If e is not part of any tight set, then we may increase it freely. Otherwise, in order to accommodate the new increase to x_e, we will decrease the allocation in e_{alt} by $dx_{e_{\text{alt}}} = -\frac{b_e}{b_{e_{\text{alt}}}} \cdot dx_e$, where

$$e_{\text{alt}} \in \arg\min_{e'}\left\{\frac{v_{e'}}{b_{e'}} : e' \in \bigcap_{\substack{S \ni e \\ bx(S)=f(S)}} S\right\}.$$

---

[7] $L_f(\lambda x) = \lambda L_f(x)$ for $\lambda \geq 0$. This follows from the definition of Lovász extension.

In other words, e_{alt} is the lowest bang-per-buck element that, upon decreasing $x_{e_{\text{alt}}}$, creates space for an increase in x_e. Note that the choice of e_{alt} is always non-empty as it contains e. This continues until either 1 unit of good is allocated to Q_j (the constraint $x(Q_j) = 1$ becomes tight) or no good produces positive utility ($v_e = p_e$ for each $e \in Q_j$). The algorithm then moves onto the next arriving part and repeats the process.

5.1 Analysis

We proceed with a primal-dual analysis to prove Theorem 1.1. Recall the dual program from Section 2:

$$\min \quad L_f(\gamma) + \sum_{j=1}^{n} \beta_j$$

$$\text{s.t} \quad b_e \cdot \gamma_e + \beta_{j(e)} \geq v_e, \qquad \text{for all } e \in E$$

$$\gamma, \beta \geq 0.$$

Upon each arrival of Q_j, we start by setting $\beta_j = 0$. Then upon each infinitesimal increase of x_e by dx_e for $e \in Q_j$, we maintain

$$\gamma_{e'} := \int_0^{\infty} G(w_{e'}^t)\, dt \quad \text{for all } e' \in E$$

and increase β_j by

$$d\beta_j := (v_e - p_e)dx_e = b_e \left(\int_0^{v_e/b_e} (1 - g(w_e^t))dt \right) dx_e.$$

where $g(x) := e^{x-1}$, and $G(x) := \int_0^x g(t)\, dt = e^{x-1} - e^{-1}$.

5.1.1 Approximate Feasibility

The goal is to show that immediately after the allocation to Q_j completes, we have $b_e\gamma_e + \beta_j \geq (1 - 1/e)v_e$ for all $e \in Q_j$. Again, by monotonicity of water levels (Proposition 3.4) the dual variables also only increase as the algorithm progresses, so this implies the approximate feasibility for the final dual values.

We denote $u^* = \max_{e \in Q_j}(v_e - p_e)$ to be the maximum marginal utility of an element of Q_j immediately following the allocation to Q_j. We claim that $\beta_j \geq u^*$. At every point during the allocation, $d\beta_j \geq u^* dx_e$, so clearly $\beta_j \geq u^* \sum_{e \in Q_j} x_e$. If $\sum_{e \in Q_j} x_e = 1$ and we have our claim. Otherwise, we must have reached zero marginal utility during our allocation, so $u^* = 0$, in which case $\beta_j \geq u^*$ trivially.

Using this bound on β_j, we conclude

$$b_e\gamma_e + \beta_j \geq b_e \int_0^{\infty} G(w_e^t)\, dt + u^* \geq b_e \int_0^{v_e/b_e} \left(G(w_e^t) + 1 - g(w_e^t) \right) dt = v_e(1 - 1/e).$$

for each element $e \in Q_j$.

5.1.2   Primal exceeds Dual

We will show the primal exceeds the dual by instead dealing with a "dual surrogate" which upper bounds the dual value.

$$L_f(\gamma) = L_f\left(\int_0^\infty G(w^t)\,dt\right) \leq \int_0^\infty L_f(G(w^t))\,dt$$

The left hand term is precisely the γ term in the dual objective. We call the right hand term the γ term of the "dual surrogate". The inequality follows from Jensen's inequality and the convexity of L_f.

We will show that the rate of change in primal and dual surrogate objectives is equal at each instant in the continuous allocation. In the allocation to Q_j, when x_e receives infinitesimal allocation dx_e, note that $x_{e_{\text{alt}}}$ is infinitesimally reduced by $\frac{b_e}{b_{e_{\text{alt}}}} \cdot dx_e$. Hence, the change in the primal objective is $b_e(v_e/b_e - v_{e_{\text{alt}}}/b_{e_{\text{alt}}})dx_e$.

Meanwhile, the change in surrogate dual objective is $\int_0^\infty \left(dL_f(G(w^t))\,dt + d\beta_j\right)$. Using the chain rule lemma for water levels (Lemma 3.8) with the fact that $w^t = w^{((b_e x_e)_{e:v_e/b_e \geq t})}$, we know that

$$d(L_f(G(w^t))) = \begin{cases} g(w_e^t)b_e \cdot dx_e & v_{e_{\text{alt}}}/b_{e_{\text{alt}}} \leq t \leq v_e/b_e \\ 0 & \text{otherwise.} \end{cases}$$

To see why $d(L_f(G(w^t)))$ is only non-zero for $t \in [v_{e_{\text{alt}}}/b_{e_{\text{alt}}}, v_e/b_e]$, observe that if $t > v_e/b_e$, then clearly x_e's value does not effect w^t. On the other hand, if $t < v_{e_{\text{alt}}}/b_{e_{\text{alt}}}$, we claim the water level of x_e remains 1 while deallocating e_{alt} and allocating e. Recall e_{alt} was chosen as the minimum bang-per-buck element in $S_{\text{tight}} = \bigcap_{\text{tight } S \ni e} S$. Note that S_{tight} is itself a tight set, one that remains tight if we restrict our allocation to elements of bang-per-buck at least t. S_{tight} also remains tight while shifting mass from e_{alt} to e. Thus, the water level vector w^t remains identical.

By definition of β_j we have $d\beta_j = b_e\left(\int_0^{v_e/b_e}(1 - g(w_e^t))dt\right)dx_e$. Hence, in total, we have that the change in surrogate dual is

$$\int_0^\infty \left(dL_f(G(w^t))\right)dt + d\beta_j = b_e\left(\int_{v_{e_{\text{alt}}}/b_{e_{\text{alt}}}}^{v_e/b_e} g(w_e^t)dt\right)dx_e + b_e\left(\int_0^{v_e/b_e}(1 - g(w_e^t))dt\right)dx_e.$$

Notice that $w_e^t = 1$ when $t \leq v_{e_{\text{alt}}}/b_{e_{\text{alt}}}$, so we have $(1 - g(w_e^t)) = 0$ for such t. Using this, we can simplify the change in surrogate dual as

$$b_e\left(\int_{v_{e_{\text{alt}}}/b_{e_{\text{alt}}}}^{v_e/b_e} g(w_e^t)dt\right)dx_e + b_e\left(\int_{v_{e_{\text{alt}}}/b_{e_{\text{alt}}}}^{v_e/b_e}(1 - g(w_e^t))dt\right)dx_e = b_e\left(\frac{v_e}{b_e} - \frac{v_{e_{\text{alt}}}}{b_{e_{\text{alt}}}}\right)dx_e.$$

The above discussion applied to an argument identical to that of Section 4.1.3 immediately gives the proof of the main theorem.

5.2   Integral Algorithm under Small Bids Assumption

So far, we analyzed a fractional algorithm for online SAP and showed that it gets a $1-1/e$ competitive ratio. Now, we will show that a similar algorithm and analysis applies to integral SAP under a small bids assumption. To illustrate the key ideas, in this section we will only examine the AdWords

version of SAP (in other words, when $b_e = v_e$ for all $e \in E$). The AdWords version of SAP does not require free disposal, which makes the analysis simpler.

Recall the small bids assumption for SAP:

Assumption 1.2 (Small Bids). *Assume there exists some $\varepsilon > 0$ such that for all $e \in E$ and $T \subseteq E$ with $f_T(\{e\}) > 0$, we have $b_e \leq \varepsilon f_T(\{e\})$.*

The small bids assumption allows us to prove the following lemma, which essentially states that the water level is an ε-Lipschitz function of the allocation. Intuitively, this is useful because it means that selecting any single element can at most increase the water levels by ε, which allows for an analysis that approximates the fractional case. Indeed, when we analyze the integral algorithm under the small bids assumption, we will only use Lemma 5.1, and not use Assumption 1.2 directly.

Lemma 5.1 (Water Levels are Lipschitz). *Suppose Assumption 1.2 holds. Let $x \in \mathbb{R}_{\geq 0}^E$ and suppose $y = x + t\mathbb{1}_e$ for some $t \geq 0$ and $e \in E$. Then we have*

$$\left\| w^{(bx)} - w^{(by)} \right\|_\infty \leq \varepsilon t.$$

Proof. Note that since $y \geq x$, we have $w^{(by)} \geq w^{(bx)}$ by monotonicity (Proposition 3.4). Furthermore, by locality (Proposition 3.7), since y is obtained from x by increasing the allocation on a single coordinate e, the element whose water level increased the most is that of e itself. Thus it suffices to show that $w_e^{(by)} \leq w_e^{(bx)} + \varepsilon t$. From the definition of water levels (Definition 3.1), we have

$$w_e^{(bx)} := \max_{S \ni e} \min_{\substack{T \subseteq E \\ f_T(\{e\}) \neq 0}} \frac{bx(S \setminus T)}{f_T(S)}.$$

Let S_x and T_x be the sets that attain the above max min for $w_e^{(bx)}$. Analogously, define S_y and T_y to be the sets that attain the max min for $w_e^{(by)}$. Then

$$
\begin{aligned}
w_e^{(by)} &= \frac{by(S_y \setminus T_y)}{f_{T_y}(S_y)} \leq \frac{by(S_y \setminus T_x)}{f_{T_x}(S_y)} \\
&= \frac{bx(S_y \setminus T_x)}{f_{T_x}(S_y)} + \frac{b(y-x)(S_y \setminus T_x)}{f_{T_x}(S_y)} \\
&\leq \frac{bx(S_x \setminus T_x)}{f_{T_x}(S_x)} + \frac{b(y-x)(S_y \setminus T_x)}{f_{T_x}(S_y)} \\
&= w_e^{(bx)} + \frac{b(y-x)(S_y \setminus T_x)}{f_{T_x}(S_y)}.
\end{aligned}
$$

Since $y - x = t\mathbb{1}_e$, we have

$$\frac{b(y-x)(S_y \setminus T_x)}{f_{T_x}(S_y)} \leq \frac{tb_e}{f_{T_x}(S_y)} \leq \frac{tb_e}{f_{T_x}(\{e\})} \leq t\varepsilon.$$

Here, the second inequality is because $e \in S_y$ and f is monotone, and the last inequality is by the small-bids assumption. $\square$

With Lemma 5.1 in hand, we are now ready to analyze integral online SAP under the small-bids assumption.

Theorem 1.3. *Under the small bids assumption (Assumption 1.2), there is a deterministic integral algorithm for online SAP which is $(1 - \varepsilon)^2 \cdot \left(1 - \frac{1}{e}\right)$-competitive.*

Proof of Theorem 1.3 . Define $f'(S) := (1 - \varepsilon)f(S)$. Since f is monotone and submodular, so is f'. In this proof, all water levels will be with respect to f', unless explicitly noted otherwise.

Consider the following integral algorithm:

1. Initialize $x = 0$. (x is the integral allocation to be returned.)

2. Upon the arrival of part j, pick

$$e_j \in \arg\max \left\{ b_e \left(1 - g(w_e^{(bx)})\right) : e \in Q_j, \ w_e^{(bx)} < 1 \right\}.$$

3. Update $x \leftarrow x + \mathbb{1}_{e_j}$. (If there is no $e \in Q_j$ with $w_e^{(bx)} < 1$, leave x unchanged.)

The idea here is the same as the fractional algorithm: always select the item with the highest utility. The only difference is that we are artificially scaling down the capacity constraints by a factor of $1 - \varepsilon$. The reason for this is to ensure that the allocation returned by the integral algorithm is always feasible to the original problem. Indeed, since the integral algorithm only allocates items whose water level (with respect to f') is less than 1, Lemma 5.1 and locality (Proposition 3.7) imply that the final water levels (again, under f'), are at most $1 + \varepsilon$. Since $f' = (1 - \varepsilon)f$, this implies that the final water levels of the allocation with respect to the original function f are at most $(1 + \varepsilon)(1 - \varepsilon) < 1$, which by indication of feasibility (Proposition 3.5) means that the integral algorithm is guaranteed to produce a feasible solution.

Having seen that the solution returned by the integral algorithm is always feasible, we now proceed to bound its competitive ratio. We recall the primal and dual LPs (with capacity constraints given by f') below:

$$
\begin{aligned}
&\max \quad \sum_e b_e x_e &\qquad &\min \quad L_{f'}(\gamma) + \sum_j \beta_j \\
&\text{s.t.} \quad \sum_{e \in Q_j} x_e \leq 1 \quad \forall j &\qquad &\text{s.t.} \quad b_e \gamma_e + \beta_{j(e)} \geq b_e \quad \forall e \in E \\
&\phantom{\text{s.t.}} \quad \sum_{e \in S} b_e x_e \leq f'(S) \quad \forall S \subseteq E &\qquad &\phantom{\text{s.t.}} \quad \gamma, \beta \geq 0. \\
&\phantom{\text{s.t.}} \quad x_e \geq 0 \quad \forall e \in E
\end{aligned}
$$

Setting the dual variables. Consider the arrival of part j. Let x denote the allocation *right before* j arrives, and let x' denote the allocation *right after* j arrives. After j makes its allocation, we update the dual variables as follows:

- $\gamma_e = G(w_e^{(bx')})$ for all $e \in E$.

- $\beta_j = \max\{0, \max\{b_e(1 - g(w_e^{(bx)})) : e \in Q_j\}\}$.

Note that the values of γ_e are updated in each iteration, whereas the values of β_j are updated once in iteration j and then remain unchanged forever. Moreover, note that the dual variables are non-decreasing throughout the course of the algorithm, because the water levels are non-decreasing and g is an increasing function.

23

Approximate dual feasibility. Consider any $e \in E$, and let $j = j(e)$. Let x denote the *final allocation* of the algorithm. Then,

$$b_e \gamma_e + \beta_j = b_e G(w_e^{(bx)}) + \beta_j \overset{(a)}{\geq} b_e G(w_e^{(bx)}) + b_e(1 - g(w_e^{(bx)})) \overset{(b)}{=} \left(1 - \frac{1}{e}\right) b_e.$$

Here, (a) uses the fact that the water levels are non-decreasing throughout the course of the algorithm, and g is increasing. (b) is using the definition of $g(t) = e^{t-1}$.

Change in primal vs. change in dual. The last thing we need to show is that the value of the algorithm is not too small compared to the value of the dual solution. To do this, we compare the change in the primal to the change in the dual in each iteration.

Consider the arrival of part j. Let x denote the algorithm's allocation right before j arrives. There are two cases: Either the algorithm selects an element or it does not. If the algorithm does not select any element, then $\Delta P = 0$. Moreover, the only reason the algorithm did not select an element is if $w_e^{(bx)} \geq 1$ for all $e \in Q_j$. This implies $\beta_j = 0$. Since γ also does not change in this case, we have $\Delta D = 0$.

It remains to consider the case where the algorithm selects an element e_j in part j. Let $x' = x + \mathbb{1}_{e_j}$ be the algorithm's allocation after j's arrival. Then, on the one hand we have

$$\Delta P = b_{e_j}.$$

On the other hand, we have

$$\Delta D = \beta_j + L_{f'}(\gamma') - L_{f'}(\gamma),$$

where $\gamma = G(w^{(bx)})$ and $\gamma' = G(w^{(bx')})$. Since element e_j was selected, we know

$$\beta_j = b_{e_j}(1 - g(w_{e_j}^{(bx)})).$$

Also,

$$
\begin{aligned}
&L_{f'}(\gamma') - L_{f'}(\gamma) \\
&= L_{f'}(G(w^{(bx')})) - L_{f'}(G(w^{(bx)})) \\
&= \left\langle \nabla_z L_{f'}(G(w^{(bz)})), x' - x \right\rangle && \text{(for some } z \in [x, x'], \text{ by Mean Value Theorem)} \\
&= \left\langle bg(w^{(bz)}), x' - x \right\rangle && \text{(by Lemma 3.8)} \\
&= b_{e_j} g(w_{e_j}^{(bz)}) \\
&\leq b_{e_j} \left(g(w_{e_j}^{(bx)}) + \frac{\varepsilon}{1 - \varepsilon} \right) && \text{(by Lemma 5.1 and } f' = (1 - \varepsilon)f)
\end{aligned}
$$

Thus,

$$\Delta D \leq b_{e_j} \left(1 + \frac{\varepsilon}{1 - \varepsilon}\right) = \frac{1}{1 - \varepsilon} \Delta P.$$

Final Competitive Ratio. To conclude, we have

$$\mathsf{ALG} = P \geq (1 - \varepsilon)D \geq (1 - \varepsilon)^2 \left(1 - \frac{1}{e}\right) \mathsf{OPT}.$$

Here, the first inequality is because $\Delta P \geq (1 - \varepsilon)\Delta D$ in each time step. For the second inequality, the $1 - \frac{1}{e}$ factor comes from approximate dual feasibility, and the $1 - \varepsilon$ factor comes from the fact that we performed the primal-dual analysis on the problem with capacities scaled down by $1 - \varepsilon$, which reduces the optimal value by a factor of at most $1 - \varepsilon$. □

6   Online Submodular Welfare Maximization for Matroidal Utilities

In this section, we give a $(1 - 1/e)$-competitive algorithm for the Online Submodular Welfare Maximization problem where the utility function of each agent is the rank function of a matroid.

Formally, there are n agents, and m items. The items arrive one at a time online. Each agent has an associated utility function $f_i : 2^{[m]} \to \mathbb{Z}_{\geq 0}$ which is the rank function of a matroid $\mathcal{M}_i$ on ground set $[m]$. In each time step, we must irrevocably assign the arriving item to some agent. Suppose items $U_i \subseteq [m]$ have been assigned to agents $i \in [n]$. Then the *welfare* of this allocation is

$$\sum_{i \in [n]} f_i(U_i).$$

The goal is to assign items to maximize welfare, as compared to the optimal offline allocation. We work in the value oracle model, where we can query the value $f_i(S)$ for any $i \in [n]$ and $S \subseteq [m]$ in constant time.

Our algorithm extends to the setting where each agent has a non-negative weight a_i. In this case, the utility function of each agent is $f_i := a_i \cdot \mathrm{rank}_{\mathcal{M}_i}$.

6.1   The Matroidal Ranking Algorithm

First, some notation. For an allocation of items to agents, we will denote by U_i the set of items assigned to agent i. Given such an allocation, we say that an item j is *available* to agent i if $f_i(U_i + j) > f_i(U_i)$.

The algorithm proceeds as follows. Independently for each agent i, select r_i uniformly at random from $[0, 1]$. Let the *priority* of agent i be defined as $a_i \cdot (1 - g(r_i))$, where $g(z) := e^{z-1}$. When an item arrives, consider the set of agents to whom this item is available, and assign the item to the highest priority agent among these. See Algorithm 3 for a formal description.

Remark 6.1 (Perusal perspective)**.** *We note that the Matroidal Ranking Algorithm yields the same allocation as the following procedure. In order of decreasing priority, each agent "peruses" the full set of items in their arrival order, and greedily picks any item which increases its utility. While this perusal perspective cannot be implemented online, it yields an identical allocation as the Matroidal Ranking Algorithm, and will be useful for the analysis.*

Algorithm 3: `Matroidal Ranking Algorithm`

---

    input : An instance of Matroid-OSWM with agents $i \in [n]$, items $[m]$, and utility
            functions $f_i : 2^{[m]} \to \mathbb{Z}_{\geq 0}$.

    output: An allocation of items $U_i \subseteq [m]$ to each agent i with welfare at least $(1 - \frac{1}{e})$ times
            the welfare of the optimal offline allocation.

---

1 Select a value $r_i \in [0,1]$ uniformly and independently for each agent, and set the priority of
    agent i to be $a_i(1 - g(r_i))$.

2 When an item j arrives, assign it to the highest priority agent to whom it is available.

3 **return** the resulting allocation U_i.

6.2 Analysis

Consider the primal and dual problems below. The primal has variables x_{ij} representing to what extent item $j \in [m]$ is allocated to agent $i \in [n]$. Notice that in the primal, rather than directly optimizing for the welfare of the agents, we simply maximize the total (weighted) quantity of items assigned, while the constraints enforce that each agent receives an independent set of items with respect to their matroid. Furthermore, there are constraints for each item enforcing that each is assigned at most once. Thus, an integer binary solution to the primal corresponds to a feasible allocation with objective value equal to the welfare of the allocation.

$$
\begin{array}{ll}
\max \quad \sum_{i \in [n]} \left(a_i \cdot \sum_{j \in [m]} x_{ij} \right) \qquad & \min \quad \sum_{i \in [n]} \sum_{S \subseteq [m]} f_i(S)\, \alpha_{i,S} + \sum_{j \in [m]} \beta_j \\[2ex]
\text{s.t.} \quad x(S) \leq f_i(S), \qquad \forall i \in [n],\ S \subseteq [m] \qquad & \text{s.t.} \quad \sum_{S \ni j} \alpha_{i,S} + \beta_j \geq a_i, \qquad \forall i \in [n],\ j \in [m] \\[2ex]
\qquad\quad \sum_{i \in [n]} x_{ij} \leq 1, \qquad \forall j \in [m] \qquad & \qquad\quad \alpha, \beta \geq 0 \\[2ex]
\qquad\quad x \geq 0 &
\end{array}
$$

Consider the primal solution $\overline{x}$ induced by the allocation at the end of the Matroidal Ranking algorithm. This $\overline{x}$ depends on the random values r_i which were chosen for each agent. We will construct a dual solution $(\overline{\alpha}, \overline{\beta})$ whose objective value is the same as that of $\overline{x}$, and which is approximately feasible in expectation. In particular, we will have

$$
\mathbb{E}_{w \sim [0,1]^n} \left[\sum_{S \ni j} \overline{\alpha}_{i,S} + \overline{\beta}_j \right] \geq \left(\frac{e-1}{e} \right) \cdot a_i
$$

for each $(i,j) \in [n] \times [m]$. This implies that the scaled up solution $(\frac{e}{e-1})(\overline{\alpha}, \overline{\beta})$ is feasible in expectation, and that $\overline{x}$ is $(\frac{e-1}{e})$-approximately optimal.

Dual Assignment We now define the dual solution $(\overline{\alpha}, \overline{\beta})$. For each agent $i \in [n]$, if U_i is the set of items assigned to agent i at the end of the algorithm, let $S_i := \operatorname{span}_{\mathcal{M}_i}(U_i)$ be the *span* of U_i with respect to $\mathcal{M}_i$ (i.e. the largest set of items containing U_i whose rank is equal to $\operatorname{rank}_{\mathcal{M}_i}(U_i)$). We assign $\overline{\alpha}_{i,S_i} := a_i \cdot g(r_i)$. For each item j, if $j \in U_i$ for some $i \in [n]$, we set $\overline{\beta}_j := a_i \cdot (1 - g(r_i))$. The remaining variables are set to zero.

26

Primal = Dual The dual solution described above has objective value equal to the primal solution returned by the algorithm. To see this, note that whenever an item is assigned to an agent i, the objective value of the dual increases by exactly a_i. Furthermore, since we only assign an item to an agent if it is available to them, the primal objective value increases by a_i as well.

Expected Approximate Feasibility We now show that the dual solution is approximately feasible in expectation. Fix a particular agent-item pair $(\widehat{i}, \widehat{j})$. We will focus on the dual constraint $\sum_{S \ni \widehat{j}} \alpha_{\widehat{i}, S} + \beta_{\widehat{j}} \geq a_{\widehat{i}}$. First, condition on all random choices r_i for $i \neq \widehat{i}$. We denote these choices by $r_{-\widehat{i}}$. Note that, once $r_{-\widehat{i}}$ is fixed, the run of the algorithm is determined by the value of $r_{\widehat{i}}$. Hence, for any choice $r_A \in [0, 1]$ we will denote a run of the algorithm in which $r_{\widehat{i}} = r_A$ as A.

For an agent $i \in [n]$ and run A of the algorithm with $r_{\widehat{i}} = r_A \in [0, 1]$, let $U_i^{(A,t)} \subseteq [m]$ denote the set of items assigned to agent i at time step t of run A of the algorithm. Likewise, let $U_i^{(A)}$ denote the set of items assigned to agent i at the end of run A of the algorithm. We will also write $\mathrm{span}(U_i^{(A,t)})$ to mean $\mathrm{span}_{\mathcal{M}_i}(U_i^{(A,t)})$, the span in agent i's matroid of the set of items assigned to agent i (and similarly for $\mathrm{span}(U_i^{(A)})$).

We now define the *critical threshold* r^* to be maximum value of $r_{\widehat{i}}$ such that $\widehat{j}$ is in the span of the items assigned to $\widehat{i}$ in the final allocation. Formally,

$$r^* := \sup\left\{ r_A \in [0, 1] : \widehat{j} \in \mathrm{span}\left(U_{\widehat{i}}^{(A)} \right) \right\}.$$

We define $\sup(\varnothing) = 0$ by convention.

The following key lemma characterizes several invariants that hold throughout the Matroidal Ranking algorithm. Specifically, it describes how the $\mathrm{span}(U_i^{(A,t)})$ changes as r_A changes. This will allow us to lower bound the expected amount of dual value assigned during the procedure.

Lemma 6.2. *Fix $r_{-\widehat{i}}$, and values r_A and r_B in $[0, 1]$ with $r_A < r_B$. Consider the two separate runs of the algorithm: A with $r_{\widehat{i}} = r_A$ and B with $r_{\widehat{i}} = r_B$. Then at each iteration t, we have*

(1) $\mathrm{span}(U_{\widehat{i}}^{(A,t)}) \supseteq \mathrm{span}(U_{\widehat{i}}^{(B,t)})$,

(2) $\mathrm{span}(U_i^{(A,t)}) = \mathrm{span}(U_i^{(B,t)})$ *for all $i \in [n]$ with $r_i \leq r_A$,*

(3) If $r_B = 1$, then $\mathrm{span}(U_i^{(A,t)}) \subseteq \mathrm{span}(U_i^{(B,t)})$ for all $i \neq \widehat{i}$.

Proof. Points (1) and (2) follow from the perusal perspective of Algorithm 3. In particular, for point (1), agent $\widehat{i}$ only peruses earlier in run A than in run B, so $\widehat{i}$ has more items to choose from in run A. For the tth item j, if $j \in U_{\widehat{i}}^{(B,t)}$ and it was not already spanned by $U_{\widehat{i}}^{(A,t-1)}$, then it would be chosen by $\widehat{i}$ in step t of run A. So $U_{\widehat{i}}^{(B,t)} \subseteq \mathrm{span}(U_{\widehat{i}}^{(A,t)})$, which implies point (1).

For point (2), the perusal of all agents i with $r_i < r_A$ is identical in both runs A and B of the algorithm, so in particular, $U_i^{(A,t)} = U_i^{(B,t)}$, implying point (2).

We prove point (3) by induction. Let $r_B = 1$. Suppose for induction (3) holds at iteration t, and a new item j arrives. Consider some $i \neq \widehat{i}$. First, if $j \in \mathrm{span}(U_i^{(B,t)})$ already at time t, then we have by induction

$$\mathrm{span}(U_i^{(A,t+1)}) \subseteq \mathrm{span}\left(\mathrm{span}(U_i^{(A,t)}) \cup \{j\} \right) \subseteq \mathrm{span}(U_i^{(B,t+1)})$$

as desired.

So suppose otherwise that $j \notin \mathrm{span}(U_i^{(B,t)})$. This means that in run B, when item j arrives, it is available to agent i. For the invariant in point (3) to break, item j must be assigned to i in run A but not assigned to i in run B. If j is not assigned to i in run B, it must be assigned to some other i' with $r_{i'} < r_i$ (since when j arrives, it is available to agent i). In particular, since $r_{\widehat{i}} = 1$, we know $i' \neq i$ and we may apply induction to i'. This tells us that at time t (before j's arrival), $\mathrm{span}(U_{i'}^{(A,t)}) \subseteq \mathrm{span}(U_{i'}^{(B,t)})$, and therefore in run A, item j was also available to i'. Since $r_{i'} < r_i$, this contradicts that j is assigned to i in run A. $\qquad \square$

This yields the following pair of corollaries.

Corollary 6.3. *If $r_{\widehat{i}} < r^*$, then $\widehat{j} \in \mathrm{span}(U_{\widehat{i}})$*

Proof. This follows directly from the definition of r^*, and point (1) from Lemma 6.2. $\qquad \square$

Corollary 6.4. *If $r^* < 1$, then item $\widehat{j}$ is always assigned to an agent i with r_i at most r^*.*

Proof. Observe first that for any $\varepsilon > 0$, if $r_{\widehat{i}} = r^* + \varepsilon \leq 1$ then item $\widehat{j}$ is assigned to some agent i with $r_i < r^* + \varepsilon$. This is because $r_{\widehat{i}} > r^*$ implies both that item $\widehat{j}$ is *available* to $\widehat{i}$ when it arrives, and that it is not assigned to $\widehat{i}$. Since this holds for every $\varepsilon > 0$ yet there are only finitely many agents, there is some such $i =: i^*$ with $r_{i^*} \leq r^*$, and some $\varepsilon^* > 0$ such that $\widehat{j}$ is assigned to i^* when $r_{\widehat{i}} = r^* + \varepsilon^*$.

Now we claim that for any value of $r_{\widehat{i}}$, item $\widehat{j}$ is always available to i^* when $\widehat{j}$ arrives. This implies the claim. First, we compare instance A of the algorithm with $r_{\widehat{i}} = r_A := r^* + \varepsilon^*$ to any instance B with $r_{\widehat{i}} = r_B > r^* + \varepsilon^*$. By Lemma 6.2(2) applied to i^*, we have $\mathrm{span}(U_{i^*}^{(A,t)}) = \mathrm{span}(U_{i^*}^{(B,t)})$ at the time t when $\widehat{j}$ arrives. So, in particular, $\widehat{j}$ is available to i^* in instance B, since it's available in instance A.

Since the above holds for any $r_B > r^* + \varepsilon$, it in particular holds for $r_B = 1$. Now we apply Lemma 6.2(3) to i^* on any instance A with $r_{\widehat{i}} = r_A < 1$ and instance B with $r_{\widehat{i}} = r_B = 1$. We have $\mathrm{span}(U_{i^*}^{(A,t)}) \subseteq \mathrm{span}(U_{i^*}^{(B,t)})$ at time t when $\widehat{j}$ arrives. Therefore, again, since i^* is available to $\widehat{j}$ in instance B, it is also available in instance A.

So in all cases, $\widehat{j}$ is available to i^* (with $r_{i^*} \leq r^*$) when it arrives. So $\widehat{j}$ is always assigned to an agent i with r_i at most r^*. $\qquad \square$

This gives us all ingredients required for the final proof that the Matroidal Ranking algorithm achieves a $(1 - 1/e)$-competitive ratio.

Proof of Theorem 1.4. Let $\overline{x}$ be the primal solution given by the Matroidal Ranking algorithm, and $(\overline{\alpha}, \overline{\beta})$ the corresponding dual solution described above. We showed that the primal and dual objectives are equal: $\sum_i (a_i \cdot \sum_j \overline{x}_{ij}) = \sum_{i,S} f_i(S)\overline{\alpha}_{i,S} + \sum_j \overline{\beta}_j$. To argue that $\overline{x}$ is $(1 - 1/e)$-competitive in expectation with the offline optimal primal solution, it then suffices by duality to show that, in expectation, $\overline{\alpha}, \overline{\beta}$ are approximately feasible.

For any fixed agent-item pair $(\widehat{i}, \widehat{j})$, we condition on the values of $r_{-\widehat{i}}$, and let r^* be the critical threshold. Then Corollary 6.3 implies that

$$\mathbb{E}_{r_{\widehat{i}} \sim [0,1]}\left[\sum_{S \ni \widehat{j}} \overline{\alpha}_{\widehat{i},S} \,\middle|\, r_{-\widehat{i}} \right] \geq \int_0^{r^*} a_{\widehat{i}} \cdot g(z)\, dz = a_{\widehat{i}} \cdot \left(g(r^*) - \frac{1}{e} \right).$$

28

Similarly, Corollary 6.4 implies that

$$\mathbb{E}_{r_{\widehat{i}} \sim [0,1]}\left[\overline{\beta}_{\widehat{j}} \,\middle|\, r_{-\widehat{i}}\right] \geq \int_0^1 a_{\widehat{i}} \cdot (1 - g(r^*))\, dz = a_{\widehat{i}} \cdot (1 - g(r^*)).$$

(note if $r^* = 1$, then the RHS is 0, so the inequality still holds, despite Corollary 6.4 not applying). The sum is then $a_{\widehat{i}} \cdot (1 - 1/e)$, and since this does not depend on the conditional values of $r_{-\widehat{i}}$, we may drop the conditioning to get

$$\mathbb{E}_{w \sim [0,1]^n}\left[\sum_{S \ni \widehat{j}} \overline{\alpha}_{\widehat{i},S} + \overline{\beta}_{\widehat{j}}\right] \geq a_{\widehat{i}} \cdot \left(1 - \frac{1}{e}\right)$$

as desired. □

References

[AGKM11] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1253–1264. SIAM, 2011. 1, 7

[AS21] Susanne Albers and Sebastian Schubert. Optimal algorithms for online b-matching with variable vertex capacities. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 2:1–2:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 7

[BC21] Guy Blanc and Moses Charikar. Multiway online correlated selection. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1277–1284. IEEE, 2021. 7

[BCIZ21] Nawal Benabbou, Mithun Chakraborty, Ayumi Igarashi, and Yair Zick. Finding fair and efficient allocations for matroid rank valuations. *ACM Transactions on Economics and Computation*, 9(4):1–41, 2021. 8

[BEF21] Moshe Babaioff, Tomer Ezra, and Uriel Feige. Fair and truthful mechanisms for dichotomous valuations. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 5119–5126. AAAI Press, 2021. 8

[BGH+23] Niv Buchbinder, Anupam Gupta, Daniel Hathcock, Anna R. Karlin, and Sherry Sarkar. Maintaining matroid intersections online. *CoRR*, abs/2309.10214, 2023. 8

[BM08] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *Acm Sigact News*, 39(1):80–87, 2008. 19

[BNW23] Niv Buchbinder, Joseph (Seffi) Naor, and David Wajc. Lossless online rounding for online bipartite matching (despite its impossibility). In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 2030–2068. SIAM, 2023. 7

[BV22] Siddharth Barman and Paritosh Verma. Truthful and fair mechanisms for matroid-rank valuations. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, pages 4801–4808. AAAI Press, 2022. 8

[CCH+23] Chandra Chekuri, Aleksander Bjørn Grodt Christiansen, Jacob Holm, Ivor van der Hoog, Kent Quanrud, Eva Rotenberg, and Chris Schwiegelshohn. Adaptive out-orientations with applications. *CoRR*, abs/2310.18146, 2023. 2

[CCPV11] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011. 4, 8

[CR22] Aleksander B. G. Christiansen and Eva Rotenberg. Fully-dynamic $\alpha + 2$ arboricity decompositions and implicit colouring. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 42:1–42:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. 2

[CS12] Mosharaf Chowdhury and Ion Stoica. Coflow: a networking abstraction for cluster applications. In Srikanth Kandula, Jitendra Padhye, Emin Gün Sirer, and Ramesh Govindan, editors, *11th ACM Workshop on Hot Topics in Networks, HotNets-XI, Redmond, WA, USA - October 29 - 30, 2012*, pages 31–36. ACM, 2012. 2

[CW23] Karthekeyan Chandrasekaran and Weihang Wang. Approximating submodular k-partition via principal partition sequence. In Nicole Megow and Adam D. Smith, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2023, September 11-13, 2023, Atlanta, Georgia, USA*, volume 275 of *LIPIcs*, pages 3:1–3:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. 8

[DFSH23] Amitay Dror, Michal Feldman, and Erel Segal-Halevi. On fair division under heterogeneous matroid constraints. *Journal of Artificial Intelligence Research*, 76:567–611, 2023. 8

[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*, SODA '13, page 101–107, USA, 2013. Society for Industrial and Applied Mathematics. 6, 7

[Fei98] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998. 8

[Fei19] Uriel Feige. *Tighter Bounds for Online Bipartite Matching*, pages 235–255. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019. 4

[FHTZ20]   Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. Edge-weighted online bipartite matching. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 412–423. IEEE, 2020. 7

[FKM+09]   Jon Feldman, Nitish Korula, Vahab S. Mirrokni, S. Muthukrishnan, and Martin Pál. Online ad assignment with free disposal. In Stefano Leonardi, editor, *Internet and Network Economics, 5th International Workshop, WINE 2009, Rome, Italy, December 14-18, 2009. Proceedings*, volume 5929 of *Lecture Notes in Computer Science*, pages 374–385. Springer, 2009. 1, 4, 5

[Fuj08]   Satoru Fujishige. Theory of principal partitions revisited. In William J. Cook, László Lovász, and Jens Vygen, editors, *Research Trends in Combinatorial Optimization, Bonn Workshop on Combinatorial Optimization, November 3-7, 2008, Bonn, Germany*, pages 127–162. Springer, 2008. 6, 8

[GHH+21]   Ruiquan Gao, Zhongtian He, Zhiyi Huang, Zipei Nie, Bijun Yuan, and Yan Zhong. Improved online correlated selection. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1265–1276. IEEE, 2021. 7

[GS17]   Guru Prashanth Guruganesh and Sahil Singla. Online matroid intersection: Beating half for random arrival. In Friedrich Eisenbrand and Jochen Koenemann, editors, *Integer Programming and Combinatorial Optimization*, pages 241–253, Cham, 2017. Springer International Publishing. 7

[HS23]   Chien-Chung Huang and François Sellier. Robust sparsification for matroid intersection with applications. *CoRR*, abs/2310.16827, 2023. 8

[HZZ20]   Zhiyi Huang, Qiankun Zhang, and Yuhao Zhang. Adwords in a panorama. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1416–1426. IEEE, 2020. 4

[IMPP19]   Sungjin Im, Benjamin Moseley, Kirk Pruhs, and Manish Purohit. Matroid coflow scheduling. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 145:1–145:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. 2

[JKR17]   Hamidreza Jahanjou, Erez Kantor, and Rajmohan Rajaraman. Asymptotically optimal approximation algorithms for coflow scheduling. In Christian Scheideler and Mohammad Taghi Hajiaghayi, editors, *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 45–54. ACM, 2017. 2

[JV10]   Kamal Jain and Vijay V. Vazirani. Eisenberg-gale markets: Algorithms and game-theoretic properties. *Games Econ. Behav.*, 70(1):84–106, 2010. 6, 11

[KP00]   Bala Kalyanasundaram and Kirk Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000. 5

[KPV13]   Michael Kapralov, Ian Post, and Jan Vondrák. Online submodular welfare maximiza-
          tion: Greedy is optimal. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth
          Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans,
          Louisiana, USA, January 6-8, 2013*, pages 1216–1225. SIAM, 2013. 4

[KVV90]   Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm
          for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on
          Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 352–358.
          ACM, 1990. , 1, 7

[MSVV07] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords
          and generalized online matching. *J. ACM*, 54(5):22, 2007. 1, 4, 5

[Nar91]   H. Narayanan. The principal lattice of partitions of a submodular function. *Linear
          Algebra and its Applications*, 144:179–216, 1991. 6, 8

[NRTV07] Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorith-
          mic Game Theory*. Cambridge University Press, 2007. 6

[PPSW21] Christos H. Papadimitriou, Tristan Pollner, Amin Saberi, and David Wajc. Online
          stochastic max-weight bipartite matching: Beyond prophet inequalities. In Péter Biró,
          Shuchi Chawla, and Federico Echenique, editors, *EC '21: The 22nd ACM Conference
          on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, pages 763–764.
          ACM, 2021. 7

[SA21]   Yongho Shin and Hyung-Chan An. Making three out of two: Three-way online cor-
          related selection. In Hee-Kap Ahn and Kunihiko Sadakane, editors, *32nd Interna-
          tional Symposium on Algorithms and Computation, ISAAC 2021, December 6-8, 2021,
          Fukuoka, Japan*, volume 212 of *LIPIcs*, pages 49:1–49:17. Schloss Dagstuhl - Leibniz-
          Zentrum für Informatik, 2021. 7

[Sch03]   Alexander Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. B*,
          volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2003. Matroids,
          trees, stable sets, Chapters 39–69. 7

[Sot13]   José A. Soto. Matroid secretary problem in the random-assignment model. *SIAM J.
          Comput.*, 42(1):178–211, 2013. 8

[VZ23]   Vignesh Viswanathan and Yair Zick. A general framework for fair allocation under ma-
          troid rank valuations. In Kevin Leyton-Brown, Jason D. Hartline, and Larry Samuelson,
          editors, *Proceedings of the 24th ACM Conference on Economics and Computation, EC
          2023, London, United Kingdom, July 9-12, 2023*, pages 1129–1152. ACM, 2023. 8

A   Proof for General Online SAP with Small Bids

For Theorem 1.3 when v_e is not necessarily equal to b_e, we utilize free disposal. However, the difficulty is that now, we must also dispose of allocations in an integral manner.

To do this, we first define a fractional algorithm which will increase x in integer increments, but will be allowed to decrease x by fractional amounts. This fractional algorithm will follow a similar procedure to the algorithm in Section 5, but with the following changes.

- We replace the function f with $f' := (1-\varepsilon)f$, i.e. decreasing our budget uniformly by a $1-\varepsilon$ factor. We also consider water levels $w^{(x)}$ with respect to f', and update the definitions of p_e, γ, β_j accordingly.

- Immediately after j arrives, we pick
$$e_j \in \arg\max_{e \in Q_j}(v_e - p_e),$$
and continuously increment only x_{e_j} until 1 unit of mass has been allocated.

- At each moment in time that x_{e_j} is being incremented by a small amount dx_{e_j}, if $w_{e_j}^{(bx)} = 1$ we pick
$$e_{\text{alt}} \in \arg\min\left\{\frac{v_e}{b_e} : e \in \bigcap_{\substack{S \ni e_j \\ bx(S) = f'(S)}} S\right\},$$
and decrease $x_{e_{\text{alt}}}$ by $dx_{e_{\text{alt}}} = -\frac{b_{e_j}}{b_{e_{\text{alt}}}}x$. To prevent multiple choices of e_{alt} being possible, we perturb the v_e values so that v_e/b_e is distinct for each $e \in E$. Notice e_{alt} may change over the course of allocating to x_{e_j}, but e_j itself does not change.

This algorithm obtains a $(1 - O(\varepsilon))(1 - 1/e)$ approximation of OPT, the optimal fractional solution, by a corresponding analysis to that in Section 5.2. However, it may be the case that we still have fraction values in x, as some x_e values may have been partially decreased.

To adapt this into an integral allocation, we do this following. As we receive part j online, we run the above fractional algorithm, storing the resulting allocation in an auxiliary vector $\widehat{x}$. Meanwhile, we generate our true allocation x as follows:

1. Set $x_{e_j} = 1$ for e_j chosen by the fractional algorithm.

2. For any $\widehat{x}_{e_{\text{alt}}}$ which was decreased to 0, we set $x_{e_{\text{alt}}} = 0$.

In other words, we only integrally dispose of $x_{e_{\text{alt}}}$ if the fractional algorithm completely decreases $\widehat{x}_{e_{\text{alt}}}$ to 0, leaving all partially decreased allocations at value 1.

Clearly, this is an integral allocation $x \in \{0,1\}^E$ with value
$$\sum_{e \in E} v_e x_e \geq \sum_{e \in E} v_e \widehat{x}_e \geq (1 - O(\varepsilon))(1 - 1/e)\mathsf{OPT}.$$

We claim also that this allocation has $w_e^{(bx)} \leq \frac{1}{1-\varepsilon}$ for all $e \in E$. This would imply that x is feasible with respect to our original submodular function f, as we would have for all $S \subseteq E$,
$$x(S) \leq \frac{1}{1-\varepsilon}f'(S) = f(S).$$

To show this claim, consider the state of the algorithm immediately after allocating to Q_j. We use the following, proven later.

Lemma A.1. *The fractional allocation $\widehat{x}$ is an optimal solution to the LP*

$$\max\{\sum_e v_e z_e : z \in \mathcal{P}_{f'},\ z \le x\}.$$

Assuming this claim, let $e_1, \dots, e_k \in E$ be the elements where $\widehat{x}_{e_i} < x_{e_i}$. Moreover, label these elements so that $\frac{v_{e_1}}{b_{e_1}} > \cdots > \frac{v_{e_k}}{b_{e_k}}$. Notice that $w_{e_i}^{(b\widehat{x})} = 1$ for all i, which implies $\max_{e \in E} w_e^{(bx)} = \max_{i \in [k]} w_{e_i}^{(bx)}$ by Proposition 3.7.

Let $\varnothing = T_0 = T_1 \subseteq \cdots \subseteq T_k$ be sets such that for $i \ge 1$, T_i is maximal such that

$$T_i \in \arg\min_T \left\{ f'(T) \colon T \ni e_i,\ \frac{bx(T \setminus T_{i-1})}{f'_{T_{i-1}}(T)} = 1 \right\}.$$

In other words, all T_i are tight sets for $\widehat{x}$ in $\mathcal{P}_{f'}$, and T_i is the smallest (as in $f'(T)$) such extension of T_{i-1} that includes e_i.

Claim A.2. $e_{i'} \notin T_i$ for $i' > i$.

Proof. Assume the claim is false. Take the least i such that there exists $e_{i'} \in T_i$ for some $i' > i$. Now for some small $\delta > 0$, consider the allocation vector $x' := \widehat{x} + \delta \cdot \left(\frac{\mathbb{1}_{e_i}}{b_{e_i}} - \frac{\mathbb{1}_{e_{i'}}}{b_{e_{i'}}} \right)$ this has $x' \le x$, and $\sum_e v_e x'_e > \sum_e v_e \widehat{x}_e$.

Moreover, we claim δ can be chosen so that $x' \in \mathcal{P}_{f'}$. This would mean x' contradicts our relation between $\widehat{x}$ and x. To see this claim, notice that we only fail if there is some $S \subseteq E$ with $\frac{b\widehat{x}(S)}{f'(S)} = 1$ and $e_i \in S$, but $e_{i'} \notin S$. If this is the case, then the set $T' := T_i \cap (S \cup T_{i-1})$ has $T' \ni e_i$, $\frac{b\widehat{x}(T')}{f'(T')} = 1$ (as tight sets are closed under union and intersection), and $f'(T') < f'(T_i)$ (as $f'(T') = x(T') \le x(T_i \setminus \{x_{i'}\}) \le x(T_i) = f'(T_i)$). This contradicts the choice of T_i. $\qquad\square$

Intuitively, our plan is to show that sequentially increasing each $\widehat{x}_{e_i}$ to $x_{e_i} = 1$ does not affect the water level for any $e_{i'}$ with $i' > i$. The small bids assumption then implies each such increase does not create too high a water level in x. To this end, consider a sequence of vectors $\widehat{x} = y^{(0)}, y^{(1)}, \dots, y^{(k)} = x$ that interpolate from $\widehat{x}$ to x as follows. We set

$$y_e^{(i)} := \begin{cases} x_e & e = e_{i'},\ i' \le i, \\ \widehat{x}_e & \text{otherwise.} \end{cases}$$

Notice that $y^{(i)}$ only differs from $\widehat{x}$ on T_i, so Algorithm 1 and the construction of the T_i sets tell us that $w_e^{(bx)} = w_e^{(by^{(i)})}$ for all $e \notin T_i$. Hence, each e_i, we have

- $w_{e_i}^{(by^{(i')})} = w_{e_i}^{(bx)} = 1$ for all $i' < i$.

- $w_{e_i}^{(by^{(i)})} \le w_{e_i}^{(by^{(i-1)})} + \frac{\varepsilon}{1-\varepsilon} = \frac{1}{1-\varepsilon}$ by our small bids assumption and $f' = (1-\varepsilon)f$.

- $w_{e_i}^{(by^{(i')})} \le w_{e_{i'}}^{(by^{(i')})} \le \frac{1}{1-\varepsilon}$ for all $i' > i$ by the locality property Proposition 3.7.

Therefore, since $y^{(k)} = x$, we have $\max_{e \in E} w_e^{(bx)} = \max_i w_{e_i}^{(bx)} \le \frac{1}{1-\varepsilon}$ as desired.

Proof of Lemma A.1. We will show that if a feasible point z in the LP is optimal if the following property holds: For every $e \in E$, either

(1) $z_e = x_e$, or

(2) there is a set $S \subseteq E$ with $bz(S) = f'(S)$ and $e \in \arg\min_{e \in S}(\frac{v_e}{b_e})$.

Notice that $\widehat{x}$ has this property. Indeed, $\widehat{x}_e \neq x_e$ implies that the algorithm deallocates from $\widehat{x}_e$ at some point, which occurs when e is the element with minimum $\frac{v_e}{b_e}$ in a tight set S, which remains tight throughout the remainder of the algorithm.

To show that a feasible vector z with these properties is optimal in the LP, consider the dual LP:

$$\min \ L_{f'}(\xi) + \sum_{e \in E} y_e x_e$$

$$\text{s.t.} \ \ b_e \xi_e + y_e \geq v_e \qquad\qquad \forall e \in E$$

$$\xi, y \in \mathbb{R}^E_{\geq 0}.$$

We construct a dual solution as follows. For $t \geq 0$, let $S_t \subseteq E$ be the maximal subset of elements such that $bz^t(S_t) = f'(S_t)$, where $z^t := (z_e)_{e: \, v_e/b_e \geq t}$ is the vector z restricted to elements with bang-per-buck at least t. Notice that $S_t \supseteq S_u$ for $u \geq t$.

Then we set our dual values at

$$\xi_e = \max\{t \geq 0 : e \in S_t\},$$

$$y_e = \max\{v_e - b_e \xi_e, \, 0\}.$$

We claim that this is a feasible dual solution with objective $\sum_{e \in E} v_e z_e$, which implies the optimality of z in the primal LP.

Note that feasibility of (ξ, y) is obvious, so we just need to check the dual objective. We have

$$L_{f'}(\xi) = \int_0^\infty f'(\{e : \xi_e \geq t\})dt = \int_0^\infty f'(S_t)dt = \int_0^\infty \left(\sum_{e \in S_t} b_e z_e \right) dt = \sum_{e \in E} b_e z_e \cdot \max\{t \geq 0 : e \in S_t\}.$$

Thus, we see that $L_{f'}(\xi) = \sum_{e \in E} b_e \xi_e z_e$.

Let $T = \{e \in E : y_e > 0\}$. Our property of z tells us that $z_e = x_e$ for $e \in T$. Moreover, if $e \notin T$, then $b_e \xi_e \geq v_e$. However, in case $b_e \xi_e > v_e$, it must be the case that $z_e = 0$, as

$$f'(S_{\xi_e}) \geq bz(S_{\xi_e}) \geq bz^{\xi_e}(S_{\xi_e}) + b_e z_e = f'(S_{\xi_e}) + b_e z_e.$$

Therefore, we can say that if $e \notin T$, then $b_e \xi_e z_e = v_e z_e$.

Finally, this allows us to compute

$$L_{f'}(\xi) + \sum_{e \in E} y_e x_e = \sum_{e \in T} (b_e \xi_e z_e + y_e x_e) + \sum_{e \notin T} (b_e \xi_e z_e + y_e x_e)$$

$$= \sum_{e \in T} (b_e \xi_e z_e + (v_e - b_e \xi_e)z_e) + \sum_{e \notin T} (v_e z_e + 0)$$

$$= \sum_{e \in T} v_e z_e + \sum_{e \notin T} v_e z_e$$

$$= \sum_{e \in E} v_e z_e.$$

$\square$

B   Laminar AdWords as a Case of Online SAP

In the AdWords problem, we have bidders known offline. Impressions j arrive online; bidder i bids b_{ij} dollars for this impression. Each bidder i has budget B_i. When an impression arrives, we must irrevocably allocate it to a bidder who may afford it. Our goal is to maximize revenue. Written in terms of an linear program,

$$\max \sum_{ij \in E} b_{ij} x_{ij}$$

$$\text{s.t} \quad \sum_j b_{ij} x_{ij} \leq B_i \quad \text{for all bidders } i$$

$$\sum_i x_{ij} \leq 1 \quad \text{for all impressions } j$$

$$x \geq 0.$$

In the laminar setting, we account for a laminar family $\mathcal{L}$ of budget constraints on the bidders. Formally speaking, $\mathcal{L}$ is a laminar family over edges E, and each set $S \in \mathcal{L}$ has a budget constraint of B_S. The laminar AdWords linear program is

$$\max \sum_{ij} b_{ij} x_{ij}$$

$$\text{s.t} \quad \sum_{ij \in S} b_{ij} x_{ij} \leq B_S \quad \text{for all } S \in \mathcal{L}$$

$$\sum_i x_{ij} \leq 1 \quad \text{for all impressions } j$$

$$x \geq 0.$$

We will show a single monotone submodular function f which captures these constraints.

Theorem B.1. *Laminar AdWords may be captured as a case of online SAP.*

Proof. Let

$$f(S) := \min \left\{ \sum_{T \in \mathcal{S}} B_T : \mathcal{S} \subseteq \mathcal{L} \text{ covers } S \right\}.$$

In other words, $f(S)$ is the most restricted budget constraint on S imposed by $\mathcal{L}$. A solution satisfying the submodular assignment problem with f clearly satisfies the laminar AdWords linear program. A solution to laminar AdWords linear program will also satisfy the submodular assignment problem with f; this follows from the fact that a minimizing sub-family $\mathcal{S}$ achieving $f(S)$ will be disjoint sets.

We move on to showing f is monotone and submodular. The former property is follows from definition. So it remains to show f is submodular. Take $S, T \subseteq E$. We will show

$$f(S \cup T) \leq f(S) + f(T) - f(S \cap T).$$

Say S and T are realized by covers $\mathcal{S}$ and $\mathcal{T}$ respectively. Then, $\mathcal{S} \cup \mathcal{T}$ (here, we allow the union family to contain duplicate sets) is clearly a cover for $S \cup T$. We will show a sub-family $\mathcal{N} \subseteq \mathcal{S} \cup \mathcal{T}$ which covers $S \cap T$ and moreover, $\mathcal{S} \cup \mathcal{T} \setminus \mathcal{N}$ is still a cover for $S \cup T$. Proving this, we are left with

$$
\begin{aligned}
f(S \cup T) &\leq \text{budget of } (\mathcal{S} \cup \mathcal{T} \setminus \mathcal{N}) \\
&= f(S) + f(T) - \text{budget of } (\mathcal{N}) \\
&\leq f(S) + f(T) - f(S \cap T).
\end{aligned}
$$

So, it remains to find a set $\mathcal{N}$ satisfying (1) $\mathcal{N}$ covers $S \cap T$ and (2) $\mathcal{S} \cup \mathcal{T} \setminus \mathcal{N}$ is still a cover for $S \cup T$. Let $\mathcal{N}$ be a collection of lowest level[8] sets which covers $S \cap T$. Clearly (1) is satisfied with this definition of $\mathcal{N}$. To see (2), note that elements in $S \cap T$ must be covered twice in the family $\mathcal{S} \cup \mathcal{T}$. Therefore, for any set $A \in \mathcal{N}$, there exists a set $B \in \mathcal{S} \cup \mathcal{T}$ covering A. This implies removing $\mathcal{N}$ from $\mathcal{S} \cup \mathcal{T}$ still leaves us with a cover for $S \cup T$. $\qquad \square$

---

[8]The laminar family is partially ordered inclusion wise, where a set A is lower than B if $A \subseteq B$.