

On Multiplicative Weight Updates for Concave and Submodular Function Maximization

[Extended Abstract]

Chandra Chekuri^{*}
Dept. of Computer Science
Univ. of Illinois
Urbana, IL 61801, USA
chekuri@illinois.edu

T.S. Jayram
IBM Almaden Research
Center
San Jose, CA 95120, USA
jayram@us.ibm.com

Jan Vondrák
IBM Almaden Research
Center
San Jose, CA 95120, USA
jvondrak@us.ibm.com

ABSTRACT

We develop a continuous-time framework based on multiplicative weight updates to approximately solve continuous optimization problems. The framework allows for a simple and modular analysis for a variety of problems involving convex constraints and concave or submodular objective functions. The continuous-time framework avoids the cumbersome technical details that are typically necessary in actual algorithms. We also show that the continuous-time algorithms can be converted into implementable algorithms via a straightforward discretization process. Using our framework and additional ideas we obtain significantly faster algorithms compared to previously known algorithms to maximize the multilinear relaxation of a monotone or non-monotone submodular set function subject to linear packing constraints.

Categories and Subject Descriptors

G.1.6 [Optimization]: Constrained Optimization

Keywords

submodular function, multiplicative weight updates

1. INTRODUCTION

The “multiplicative weight updates (MWU) method” has a wide variety of applications in computer science and can be considered a meta-algorithm. The excellent survey of Arora, Hazan and Kale [2] takes this point of view and describes several applications that follow from the basic method and its analysis. One of the key applications of the MWU method is to obtain fast near-optimal algorithms for a large class of continuous optimization problems such as fractional packing

^{*}Work on this paper supported in part by an IBM Faculty Award and NSF grant CCF-1319376.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

and covering and mixed packing/covering problems. Plotkin, Shmoys and Tardos [35], and Grigoriadis and Khachiyan [22] initiated this line of work and subsequently there has been a large amount of literature on this topic; see [38, 19, 20, 17, 39, 8, 18, 27, 29, 1, 3, 26, 40].

Linear functions and constraints have been primarily the setting of interest, but recent applications have shown the usefulness of considering more general objectives as well, such as concave or submodular. Our aim in this paper is to develop a versatile framework that allows the inclusion of concave or submodular objectives under convex constraints. A concrete goal is to obtain faster algorithms for submodular objective functions. We define the relevant notions first. A set function $f : 2^N \rightarrow \mathbb{R}$ over a finite ground set N is *submodular* if for all $A, B \subseteq N$ it satisfies $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. It is *monotone* if $f(A) \leq f(B)$ for all $A \subseteq B$. In this paper we consider only non-negative submodular functions and use the term non-monotone to refer to a function that may not be monotone. There are many applications for constrained submodular function maximization. The meta-problem of interest here is $\max_{S \in \mathcal{S}} f(S)$ where $\mathcal{S} \subseteq 2^N$ is a collection of feasible sets that model the constraints (typically \mathcal{S} is a down-closed family of sets¹). Most of the problems here are NP-Hard and the main focus has been on developing approximation algorithms.

The multilinear extension F of f is a continuous function that extends f to the domain $[0, 1]^N$; for a point $\mathbf{x} \in [0, 1]^n$ where N is identified with $\{1, 2, \dots, n\}$, the function F is defined as $F(\mathbf{x}) = \sum_{S \subseteq N} f(S) \prod_{i \in S} x_i \prod_{i \notin S} (1 - x_i)$. Equivalently, $F(\mathbf{x}) = \mathbb{E}[f(R)]$ where R is a random set obtained by picking each element $i \in N$ independently with probability x_i . The multilinear relaxation for the discrete optimization problem $\max_{S \in \mathcal{S}} f(S)$ is the continuous optimization problem $\max\{F(\mathbf{x}) : \mathbf{x} \in P\}$ where P is a convex relaxation for the constraint set \mathcal{S} (that is, P contains the convex hull of the characteristic vectors of the sets in \mathcal{S}). A solution to the multilinear relaxation has to be then suitably rounded. This paradigm has led to a number of new algorithmic results for constrained submodular function maximization [10, 37, 11, 33, 30, 13, 16]. The multilinear extension F is neither convex nor concave. In fact it is NP-Hard to solve the multilinear relaxation to within a $(1 - 1/e - \epsilon)$ factor for any fixed $\epsilon > 0$ even for the simple cardinality constraint polytope $\{\mathbf{x} \in [0, 1]^n : \sum_i x_i \leq k\}$ and when f is an explicitly given monotone function f . This follows from a reduction from

¹We say \mathcal{S} is down-closed if $A \subseteq B, B \in \mathcal{S} \Rightarrow A \in \mathcal{S}$

the maximum k -coverage problem [10]. Nevertheless constant factor approximation algorithms are known via continuous versions of greedy [37, 11, 16] and local-search [33, 13]. In particular, there is a $(1 - 1/e)$ -approximation for the monotone case [37, 11] and a $1/e$ -approximation for the non-negative case [16] assuming that P is a solvable polytope². Although the methodology is powerful and has shown promise in some empirical work [21], one of the limitations is the slow running time of the algorithms. One of our motivations for considering MWU-based algorithms for the multilinear relaxation is the work of Azar and Gamzu [4]. They developed a MWU-based greedy algorithm for monotone submodular function maximization subject to linear packing constraints; we discuss more details of their work later in the paper.

An MWU algorithm can be viewed as a black-box Turing reduction of an optimization problem to a simpler problem which can be solved faster. For instance the problem we desire to solve could be the maximum s - t flow problem and the simpler problem could be the s - t shortest path problem. Typically the MWU approach reduces optimization with many constraints to optimization with one constraint. Since the reduction uses the algorithm for the simpler problem in a black-box fashion it is common to refer to it as an “oracle”. Clearly, several considerations influence this approach, the main ones being (i) the objective function (ii) the type of constraints, and (iii) the “oracle” that is available to the algorithm. A significant effort has been devoted to obtain running times that are “width independent” (independent of the numerical values of coefficients present in the constraints), either by width reduction techniques as suggested in [35], or via the idea of variable-sized (non-uniform) increments [19, 20].

1.1 Our Results

We introduce a continuous-time framework for optimization of concave and submodular functions based on MWU which is motivated by the desire to obtain a clean high-level analysis. We analyze algorithms in the framework using simple differential equations. The advantage is that we can initially sidestep the discretization issues involved in actual implementations. After the high-level analysis we are able to derive implementable algorithms by discretizing the continuous process. A key advantage is that the discretization process is systematic; we choose the (variable) step sizes to ensure that the the continuous process can be approximated by the discrete process. In particular, it allows us to cleanly integrate the issues involved in handling the constraints and the objective function.

In this version of the paper we focus on maximization problems subject to packing constraints. Our framework yields width-independent algorithms for maximizing linear, concave and submodular functions. The framework and several results also apply to minimizing convex objectives, covering constraints, and mixed packing and covering constraints; these will be discussed in detail in a future version of the paper. Several previous results including the work of Garg and Könemann on variable-sized increments [19, 20] can be explained cleanly in our framework.

After we present the generic framework, we focus on deriving fast algorithms for the problem of maximizing the

²We say that $P \subseteq [0, 1]^n$ is solvable if there is an efficient algorithm to optimize any linear function over P .

multilinear extension F of a submodular function f defined over a ground set of n elements subject to m explicitly given linear packing constraints. We will assume that f is available via a value oracle and our main metric for the efficiency of an algorithm will be the number of calls it makes to f or to an oracle that returns the partial derivatives of F . The special properties of F and the sampling issues in evaluating $F(\mathbf{x})$ make it non-trivial to adapt the MWU method to solve the multilinear relaxation.

We obtain the following results. Our first result applies to maximization of the multilinear extension of a monotone submodular function.

THEOREM 1. *Let F be the multilinear extension of a monotone submodular function $f : 2^N \rightarrow \mathbb{R}_+$. Given $\epsilon \in (0, \frac{1}{2})$, $A \in \mathbb{R}_+^{m \times n}$, there is an algorithm that computes a solution \mathbf{y} to the problem $\max\{F(\mathbf{x}) : 0 \leq \mathbf{x} \leq \mathbf{1}, A\mathbf{x} \leq \mathbf{1}\}$ such that $F(\mathbf{y}) \geq (1 - 1/e - \epsilon)\text{OPT}$ and:*

- *the algorithm makes $O(\frac{n}{\epsilon^2} \text{polylog}(m, n))$ value oracle calls to partial derivatives of F ,*
- *and in addition makes $O(\frac{(m+n)^2}{\epsilon^2} \text{polylog}(m, n))$ arithmetic steps.*

If F is accessible through a value oracle to f , the algorithm can be implemented so that the number of calls to the value oracle for f is $\tilde{O}(n^2/\epsilon^4)$.

Our second result applies to non-negative (possibly non-monotone) submodular functions. More precisely it holds for smooth submodular functions [37], i.e., twice-differentiable functions with the property that $\frac{\partial^2 F}{\partial x_i \partial x_j} \leq 0$ for all $i, j \in [n]$ and $\mathbf{x} \in [0, 1]^n$. Smooth submodular functions arise naturally as multilinear extensions of submodular functions, but they can also arise in other ways (see [21] for a maximization problem involving a smooth submodular function which is not multilinear).

THEOREM 2. *Let F be a non-negative smooth submodular function. Given $\epsilon \in (0, \frac{1}{2})$, $A \in \mathbb{R}_+^{m \times n}$, there is an algorithm that computes a solution \mathbf{y} to the problem $\max\{F(\mathbf{x}) : 0 \leq \mathbf{x} \leq \mathbf{1}, A\mathbf{x} \leq \mathbf{1}\}$, such that $F(\mathbf{y}) \geq (1/e - \epsilon)\text{OPT}$ and:*

- *$O(mn^2/\epsilon^2)$ value oracle calls to the partial derivatives of F ,*
- *and in addition makes $O(m^2n^2/\epsilon^2)$ arithmetic steps.*

If F is the multilinear extension of a non-negative submodular function f the algorithm can be implemented such that the number of calls to the value oracle to f is $\tilde{O}(mn^3/\epsilon^4)$.

We remark that the approximation ratios match the best known bounds while the running times are significantly improved. We briefly compare the run-times we obtain with those from prior work. The previous polynomial-time approximation algorithms for solving the multilinear relaxation based on continuous greedy [37, 11, 16] and local search [13] are polynomial-time but very slow. Moreover, the running times are not explicitly stated, partly due to the fact that the algorithms assume an oracle to optimize a linear function over the underlying polytope P . Faster variants of continuous greedy (with near-quadratic running times) have been developed in recent work by Badanidiyuru and Vondrák [6]

but these are for specific polytopes such as the matroid polytope or the knapsack polytope (1 linear constraint). When the polytope P is induced by m linear packing constraints we could use a MWU based algorithm for linear optimization that provides a $(1 - \epsilon)$ -approximation in $\tilde{O}(n + m)$ time [29]; even with this optimization the overall running time of the algorithms for the multilinear relaxation can be conservatively estimated to be $\Omega(n^5)$ both in terms of value oracle calls to f and arithmetic operations. The algorithms we describe in this paper yield significant improvements and point towards practical feasibility of multilinear relaxation based algorithms.

Azar and Gamzu [4] gave an MWU-based greedy algorithm for the discrete setting $\max\{f(S) : A\mathbf{x} \leq \mathbf{1}, \mathbf{x} \in \{0, 1\}^n\}$. The number of oracle calls to f in their algorithm can be $\Omega(n^2)$ and the number of arithmetic operations can be $\Omega(mn)$. Among other results they obtain a $(1 - 1/e - \epsilon)$ -approximation for the monotone case when the width³ of the system is $\Omega(\log m/\epsilon^2)$; the width is defined as $\min_{i,j} \lceil \frac{1}{A_{ij}} \rceil$. Our algorithm for the monotone case is similar in spirit to theirs although we optimize the multilinear relaxation while they directly optimize f . We derive our algorithm organically from a general framework; to obtain good running-times while dealing with the multilinear extension F we need to use several ideas, some borrowed from recent work [6]. Further, our framework extends naturally to non-monotone submodular functions, unlike the discrete greedy approach.

Applications: Several approximation algorithms for constrained submodular function maximization are based on solving the multilinear relaxation followed by rounding; for some problems this is the only known approach or gives better bounds than any other method. A general framework for rounding is via contention resolution schemes [13]. A number of interesting constraints can be modeled by linear packing systems of the form $A\mathbf{x} \leq \mathbf{1}$ including several types of useful matroids such as uniform, partition and laminar matroids and their intersections.⁴ For many of these problems the bottleneck, in terms of the run-time of the algorithm, is solving the multilinear relaxation. The contention resolution schemes are typically based on randomized rounding followed by simple pruning rules. Thus, we obtain algorithms for a large number of problems that are faster by factors of n^3 or more. Since these specific applications are not the focus of this paper we do not go into details, however, we briefly discuss two concrete examples to illustrate the advantage of using the multilinear relaxation.

First, consider the problem $\max\{f(S) : A\mathbf{x} \leq \mathbf{1}, \mathbf{x} \in \{0, 1\}^n\}$ where the matrix A is k -column sparse (the maximum number of non-zero entries in each column is at most k). Such column-sparse problems capture several important problems; for instance matchings in general graphs are captured by 2-sparse matrices. The multilinear relaxation based approach yields an $\Omega(\frac{1}{k^{1/W}})$ approximation for both monotone and non-monotone submodular functions [7, 13] where W is the width of system. The combinatorial approach of Azar and Gamzu [4], even though it is based on MWU, gives

³For a system of packing constraints $A\mathbf{x} \leq \mathbf{b}$, width is the minimum ratio b_i/A_{ij} over all i, j .

⁴The matroid polytope has an exponential number of constraints in general, while these special cases have simple descriptions with a linear (in n) number of explicit constraints.

a weaker bound of $\Omega(\frac{1}{Wk^{1/W}})$ for only the monotone case. In contrast, since we solve the multilinear relaxation, we are able to match the previous bounds with a run-time that is essentially the same as the combinatorial greedy algorithm of [4].

Second, consider the problem of routing pairs in a capacitated path or tree (called UFP on the line/tree) to maximize a submodular function of the routed pairs [13]. The constraints can be modeled via simple linear packing constraints. There is a constant factor approximation for these problems via the multilinear relaxation. Even when the objective function is a linear function, the only way to obtain a constant factor approximation is via a linear programming relaxation.

Multiple Submodular and Concave Objectives: We can show that our continuous-time framework extends to handle multiple objective functions. A standard approach is to convert the objectives into constraints with lower bounds on each constraint. This is relatively easy to do for concave objectives, however, handling multiple submodular objectives is less straightforward. Multiple monotone submodular objectives were considered in [12] with an application to an a fair-allocation problem; an algorithm based on adapting the continuous greedy was described in the same paper. The continuous-time MWU algorithm and the details of discretization and the run-time analysis are deferred to a later version of the paper.

Other Related Work: Submodular functions are playing an increasingly important role in various applications, in particular machine learning. There are several efforts to obtain faster algorithms for both maximization and minimization. Here we focus on maximization. There is a spectrum of results ranging from the more theoretical settings [14, 4, 31, 6, 9] where the goal is obtain provable guarantees on the running time, as well as in applied settings with motivations coming from machine learning applications [21, 24, 5].

Continuous-time variants of the multiplicative weight update method have been considered in the past, e.g. in the context of packing/covering LPs [34], and regret minimization in online learning [36, 32]. Khandekar [27] and the survey of Arora *et al.* [2] apply the discrete-time experts framework for online optimization and the corresponding regret bounds in designing and analyzing fast algorithms. Our continuous-time analysis and discretization is similar in spirit to some of the ideas in [32]. Our framework and results are more explicit and in particular we address optimization problems involving submodular functions which are not considered in [32].

Fast algorithms for packing, covering and mixed packing and covering problems have also been designed via the logarithmic potential function [23, 15, 25]. It will be interesting to see whether a continuous-time framework can be developed for the logarithmic potential function.

Due to space limitations in this version of the paper, we omit some proofs and the implementation details of the algorithms that yield Theorems 1 and 2.

2. MWU FOR OPTIMIZATION: A CONTINUOUS POINT OF VIEW

In this section we present a continuous-time framework of multiplicative weight updates for optimization. This point

of view eliminates some of the issues which are important in the eventual implementation - namely discretization and running time. On the positive side, the continuous presentation is cleaner and highlights the main conceptual ideas in multiplicative weight updates, without cumbersome technical details. Our main purpose here is to use the continuous framework to show how different variants of multiplicative weight updates can be tied together and unified in a simple manner.

2.1 MWU template for optimization

We consider the following generic optimization problem.

$$\begin{aligned} \max f(\mathbf{x}) : \\ g_i(\mathbf{x}) \leq 1; \quad 1 \leq i \leq m; \\ \mathbf{x} \in P \end{aligned}$$

where $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $1 \leq i \leq m$ are convex functions and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a generic continuous function. P is a convex (polyhedral) constraint (like a box constraint) which is assumed to be “simple” — we know how to optimize over it quickly. We will assume that the feasible region is bounded and f is bounded in the feasible region and that an optimum solution exists. The MWU method effectively collapses the “complicated” constraints $g_i(x) \leq 1$ into a single convex constraint that can be dealt with more efficiently. Let $\mathbf{x}^* = \arg \max\{f(\mathbf{y}) : \mathbf{y} \in P, g_i(\mathbf{y}) \leq 1 \forall i\}$ denote an optimal solution and let $\text{OPT} = f(\mathbf{x}^*)$.

The continuous framework that we present below has the following template. It uses a parameter $\eta > 0$ and involves two time-varying vectors: a weight vector $\mathbf{w}(t) \in \mathbb{R}_+^m$ and a domain point $\mathbf{v}(t) \in \mathbb{R}^n$, for $t \in [0, 1]$. These two vectors are jointly related to each other. The vector $\mathbf{v}(t)$ is related to $\mathbf{w}(t)$ via a carefully designed optimization problem depending on f . The important feature of this optimization problem is that $\mathbf{v}(t)$ is a solution satisfying $\mathbf{v}(t) \in P$ and a single convex constraint $\sum_{i=1}^m w_i(t) g_i(\mathbf{v}(t)) \leq \sum_{i=1}^m w_i(t)$. The weight vector $\mathbf{w}(t)$ is related to $\mathbf{v}(t)$ via a differential equation. We describe this system of equations below as an algorithm and can be interpreted as a limit of a process where the functions are evaluated at discrete points of t going from 0 to 1.

Algorithm 1 MWU template

```

1: procedure MWU( $f, g_1, \dots, g_m, P, \eta$ ):
2:    $\mathbf{w}(0) = 1$   $\triangleright \mathbf{w}(t) \in \mathbb{R}^m$  for all  $t$ 
3:   for  $t \in [0, 1]$  do
4:     Solve an appropriate optimization problem and
       obtain  $\mathbf{v}(t)$ 
5:      $\triangleright \mathbf{v}(t) \in P$  and  $\sum_{i=1}^m w_i(t) g_i(\mathbf{v}(t)) \leq \sum_{i=1}^m w_i(t)$ 
6:     for  $i \in [m]$  do
7:        $\frac{dw_i}{dt} = \eta w_i(t) g_i(\mathbf{v}(t))$ 
8:     end for
9:   end for
10:  return  $\mathbf{x}_{\text{out}} = \int_0^1 \mathbf{v}(t) dt$ 
11: end procedure

```

The key result of the continuous-time framework is that implies the near-feasibility of the output.

THEOREM 3. *The point \mathbf{x}_{out} returned by Algorithm 1 satisfies $\mathbf{x}_{\text{out}} \in P$ and $g_i(\mathbf{x}_{\text{out}}) \leq 1 + \frac{\ln m}{\eta}$ for $1 \leq i \leq m$. In particular if $\eta \geq \frac{\ln m}{\epsilon}$, $g_i(\mathbf{x}_{\text{out}}) \leq 1 + \epsilon$ for all i .*

Further, suppose there is a point $\mathbf{x}_0 \in P$ such that $g_i(\mathbf{x}_0) = 0 \forall i$. Then the modified solution $\mathbf{x}'_{\text{out}} = \theta \mathbf{x}_{\text{out}} + (1 - \theta) \mathbf{x}_0$, where $\theta = 1/(1 + \frac{\ln m}{\eta})$, is feasible.

Given any point \mathbf{x}_0 in the interior of the feasible set, we can assume in fact by a suitable transformation to the g_i 's, e.g., $\tilde{g}_i(\mathbf{x}) = \frac{g_i(\mathbf{x}) - g_i(\mathbf{x}_0)}{1 - g_i(\mathbf{x}_0)}$, that $g_i(\mathbf{x}_0) = 0$ while applying Algorithm 1.

PROOF. First, observe that $\mathbf{x}_{\text{out}} = \int_0^1 \mathbf{v}(t) dt$ is a convex combination of points in P , and therefore also in P . Secondly, we analyze the constraints $g_i(\mathbf{x}) \leq 1$. By the convexity of g_i , we have:

$$g_i(\mathbf{x}_{\text{out}}) = g_i\left(\int_0^1 \mathbf{v}(t) dt\right) \leq \int_0^1 g_i(\mathbf{v}(t)) dt.$$

By the update rule for $w_i(t)$, we get

$$\int_0^1 g_i(\mathbf{v}(t)) dt = \frac{1}{\eta} \int_0^1 \frac{1}{w_i(t)} \frac{dw_i}{dt} dt = \frac{1}{\eta} \ln w_i(1) \quad (1)$$

since $w_i(0) = 1$. We also have

$$\frac{d}{dt} \sum_{i=1}^m w_i(t) = \sum_{i=1}^m \frac{dw_i}{dt} = \eta \sum_{i=1}^m w_i(t) g_i(\mathbf{v}(t)) \leq \eta \sum_{i=1}^m w_i(t),$$

by the constraints on $\mathbf{v}(t)$. Solving this differential inequality, we obtain

$$\sum_{i=1}^m w_i(t) \leq e^{\eta t} \sum_{i=1}^m w_i(0) = e^{\eta t} m.$$

Thus, for each fixed i , $w_i(1) \leq e^\eta m$. By eq. (1), $g_i(\mathbf{x}_{\text{out}}) \leq \frac{1}{\eta} \ln w_i(1) \leq 1 + \frac{\ln m}{\eta}$.

Finally, consider $\mathbf{x}'_{\text{out}} = \theta \mathbf{x}_{\text{out}} + (1 - \theta) \mathbf{x}_0$ where $\theta = 1/(1 + \frac{\ln m}{\eta})$ and $\mathbf{x}_0 \in P$, $g_i(\mathbf{x}_0) = 0$. By convexity, $\mathbf{x}'_{\text{out}} \in P$, and we have $g_i(\mathbf{x}'_{\text{out}}) \leq \theta g_i(\mathbf{x}_{\text{out}}) + (1 - \theta) g_i(\mathbf{x}_0) \leq 1$. \square

REMARK 1. *The analysis did not assume non-negativity of the g_i 's. In particular, a covering constraint of the form $h(\mathbf{x}) \geq 1$ where h is a concave function can be modeled as $g(\mathbf{x}) \leq 1$ where $g(\mathbf{x}) = -h(\mathbf{x}) + 2 \leq 1$ is a convex function. The continuous-time process, unlike the discrete time experts framework, does not have an error term that depends on range of the functions g_i . Finally, the error terms can be made $(1 + \epsilon)$ by choosing $\eta = \frac{\ln m}{\epsilon}$ which does not depend on n but only on the number of constraints m .*

REMARK 2. *Let $\phi(t) = \sum_{i=1}^m w_i(t)$ be the potential function at time t ; note that $\frac{1}{\eta} \ln \frac{\phi(t)}{m} \leq t$ by the proof above. Several algorithms/analyses work with this quantity as a proxy for “time”. We believe that fixing the time evolution to be the fixed interval of length 1 allows for a cleaner separation of the analysis of optimality and feasibility, and makes it easier to integrate the analysis of the continuous greedy algorithm (for submodular maximization) with multiplicative weight updates.*

2.2 Concave and linear objective functions

We apply the framework to maximize a concave function f . Suppose $\mathbf{v}(t)$ is an optimal solution to a concave maximization problem with a single constraint:

$$\mathbf{v}(t) = \arg \max_{\mathbf{y} \in P} \left\{ f(\mathbf{y}) : \sum_{i=1}^m w_i(t) g_i(\mathbf{y}) \leq \sum_{i=1}^m w_i(t) \right\} \quad (2)$$

We analyze the objective value \mathbf{x}_{out} returned by the algorithm as follows. By concavity of f :

$$f(\mathbf{x}_{\text{out}}) = f\left(\int_0^1 \mathbf{v}(t) dt\right) \geq \int_0^1 f(\mathbf{v}(t)) dt$$

At each time t , \mathbf{x}^* is a candidate solution for the optimization problem given by eq. (2). Therefore, $f(\mathbf{v}(t)) \geq f(\mathbf{x}^*)$ for all t and we conclude that $f(\mathbf{x}_{\text{out}}) \geq \text{OPT}$.

Now suppose further that f is non-negative and consider the modified solution $\mathbf{x}'_{\text{out}} = \theta \mathbf{x}_{\text{out}} + (1 - \theta) \mathbf{x}_0$ given by Theorem 3, with $\theta = \frac{1}{1 + \epsilon}$, which we showed to be completely feasible. By concavity, $f(\mathbf{x}'_{\text{out}}) \geq \theta f(\mathbf{x}_{\text{out}}) + (1 - \theta) f(\mathbf{x}_0) \geq \theta f(\mathbf{x}_{\text{out}}) \geq \text{OPT}/(1 + \epsilon)$. We summarize these observations as follows:

THEOREM 4. *Let f be a concave function. The point \mathbf{x}_{out} returned by Algorithm 1 where $\mathbf{v}(t)$ obeys eq. (2) satisfies $f(\mathbf{x}_{\text{out}}) \geq \text{OPT}$, $\mathbf{x}_{\text{out}} \in P$ and $g_i(\mathbf{x}_{\text{out}}) \leq 1 + \epsilon$ for all i . Moreover, if f is non-negative, then the modified solution \mathbf{x}'_{out} is completely feasible with $f(\mathbf{x}'_{\text{out}}) \geq \text{OPT}/(1 + \epsilon)$.*

Linear objective with linear packing constraints: An important special case is when $f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$ is a linear function and all the constraints are linear, that is, the feasible region is defined as $A\mathbf{x} \leq \mathbf{1}$. The oracle required in this case is to solve $\max \mathbf{c}^T \mathbf{x}$ subject to a single constraint of the form $\mathbf{a}^T \mathbf{x} \leq 1$. An optimum solution to this problem can be easily computed by finding the i that maximizes the ratio c_i/a_i . Moreover, we observe that this optimum solution has a single non-zero coordinate. We note that number of variables can be much larger than m as long as the oracle can compute the best coordinate in each iteration efficiently. For instance in the maximum throughput multicommodity flow problem the variables correspond to paths between source-sink pairs and the constraints correspond to edges; the Garg-Könemann algorithm corresponds to picking a shortest path among the source-sink pairs in each iteration.

2.3 Monotone submodular functions

Next, we show how the framework can be adapted to the case of a monotone submodular objective functions. Let the objective function $F : \mathbb{R}^n \rightarrow \mathbb{R}_+$ be a monotone smooth submodular function; in particular, we require that $\frac{\partial F}{\partial x_i} \geq 0$ (via monotonicity) and $\frac{\partial^2 F}{\partial x_i \partial x_j} \leq 0$ for all $i, j \in [n]$ (via submodularity) over $[0, 1]^n$. We assume that $P \subseteq [0, 1]^n$.

For ease of notation, let $\mathcal{G}_t = \{\mathbf{y} : \sum_{i=1}^m w_i(t) g_i(\mathbf{y}) \leq \sum_{i=1}^m w_i(t)\}$ at time t denote the set defined by combining all the constraints g_i using the weight vector $\mathbf{w}(t)$.

Let $\mathbf{x}(t) \in \mathbb{R}^n$ for all $t \in [0, 1]$ such that $\mathbf{x}(0) = 0$ and for $t \in [0, 1]$:

$$\boxed{\begin{aligned} \mathbf{v}(t) &= \arg \max_{\mathbf{y} \in P} \left\{ \mathbf{y} \cdot \nabla F|_{\mathbf{x}(t)} : \mathbf{y} \in \mathcal{G}_t \right\} \\ \frac{d\mathbf{x}}{dt} &= \mathbf{v}(t) \end{aligned}} \quad (3)$$

Observe that is a linear optimization problem, with one additional convex constraint. In case the g_i 's are linear constraints, this is just a linear programming problem.

We state an abstract inequality that relies only on the continuous submodularity of F .

LEMMA 5. *Let $\mathbf{x}(t) \in [0, 1]^n$, $\mathbf{v}(t)$ and $\mathbf{z}(t)$ be such that (a) $\frac{d\mathbf{x}}{dt} = \mathbf{v}(t)$, (b) $\mathbf{z}(t) \geq 0$, and (c) $\mathbf{z}(t) \cdot \nabla F|_{\mathbf{x}(t)} \leq \mathbf{v}(t) \cdot \nabla F|_{\mathbf{x}(t)}$. Then:*

$$\frac{d}{dt} F(\mathbf{x}(t)) \geq F(\mathbf{x}(t) + \mathbf{z}(t)) - F(\mathbf{x}(t)).$$

PROOF. By submodularity, F is concave along the non-negative direction $\mathbf{z}(t)$, therefore:

$$\begin{aligned} F(\mathbf{x}(t) + \mathbf{z}(t)) - F(\mathbf{x}(t)) &\leq \mathbf{z} \cdot \nabla F|_{\mathbf{x}(t)} \\ &\leq \mathbf{v}(t) \cdot \nabla F|_{\mathbf{x}(t)} = \frac{d\mathbf{x}}{dt} \cdot \nabla F|_{\mathbf{x}(t)} = \frac{d}{dt} F(\mathbf{x}(t)). \end{aligned}$$

This concludes the proof. \square

THEOREM 6. *Let F be monotone smooth submodular function with $F(0) = 0$. The point \mathbf{x}_{out} returned by Algorithm 1 with $\eta = \frac{\ln m}{\epsilon}$ and F as the objective function where the subproblem is solved using eq. (3) satisfies $F(\mathbf{x}_{\text{out}}) \geq (1 - \frac{1}{e})\text{OPT}$, $\mathbf{x}_{\text{out}} \in P$ and $g_i(\mathbf{x}_{\text{out}}) \leq 1 + \epsilon$ for all i . Further, suppose F is non-negative, $0 \in P$ and $g_i(0) = 0$. Then the modified solution \mathbf{x}'_{out} of Theorem 3 with $x_0 = 0$ obeys all the constraints and satisfies $F(\mathbf{x}'_{\text{out}}) \geq (1 - \frac{1}{e} - \epsilon)\text{OPT}$.*

PROOF. The (near) feasibility of \mathbf{x}_{out} follows from Theorem 3.

Because $\mathbf{x}(0) = 0$, we have $\mathbf{x}(t) = \int_0^t \mathbf{v}(\tau) d\tau$. Since $\mathbf{v}(t) \in [0, 1]^n$, we have $\mathbf{x}(t) \in [0, t]^n$ for all $t \in [0, 1]$. We analyze the optimality of the returned value $\mathbf{x}_{\text{out}} = \mathbf{x}(1)$.

Define the non-negative vector $\mathbf{z}(t) = (\mathbf{x}(t) \vee \mathbf{x}^*) - \mathbf{x}(t)$, where \vee denotes coordinate-wise maximum. Both $\mathbf{x}^* \geq 0$ and $\mathbf{x}(t) \geq 0$, so $\mathbf{z}(t) \leq \mathbf{x}^*$. Because $\mathbf{x}(t) \in [0, 1]^n$, the gradient at $\mathbf{x}(t)$ is non-negative (since F is monotone), hence $\mathbf{z}(t) \cdot \nabla F|_{\mathbf{x}(t)} \leq \mathbf{x}^* \cdot \nabla F|_{\mathbf{x}(t)} \leq \mathbf{v}(t) \cdot \nabla F|_{\mathbf{x}(t)}$, since \mathbf{x}^* a candidate solution for the optimization problem given by eq. (3). Applying Lemma 5:

$$\begin{aligned} \frac{d}{dt} F(\mathbf{x}(t)) &\geq F(\mathbf{x}(t) + \mathbf{z}(t)) - F(\mathbf{x}(t)) \\ &= F(\mathbf{x}(t) \vee \mathbf{x}^*) - F(\mathbf{x}(t)) \geq \text{OPT} - F(\mathbf{x}(t)), \end{aligned}$$

by monotonicity. This has the solution $F(\mathbf{x}(t)) \geq (1 - e^{-t})\text{OPT}$, proving the theorem with $t = 1$.

Suppose F is non-negative. By Theorem 3, the modified solution \mathbf{x}'_{out} is completely feasible and by submodularity, $F(\mathbf{x}'_{\text{out}}) \geq \frac{1}{1 + \epsilon} F(\mathbf{x}_{\text{out}}) \geq \frac{1}{1 + \epsilon} (1 - \frac{1}{e})\text{OPT} \geq (1 - \frac{1}{e} - \epsilon)\text{OPT}$. \square

2.4 Non-negative submodular functions

For non-negative (non-monotone) submodular functions, we need only a minor adjustment which is based on the $1/e$ -approximation algorithm of Feldman, Naor and Schwartz [16]. Here too we will only require that F is a smooth submodular function. The added ingredient here is that we increase the coordinates less aggressively than in the monotone case, because increasing coordinates too much can possibly hurt the solution. This is expressed by the following simple lemma (paraphrasing [16])⁵.

LEMMA 7. *If $\theta \in (0, 1)$, $\mathbf{x} \in [0, \theta]^N$ and $\mathbf{x}^* \in [0, 1]^N$, then $F(\mathbf{x} \vee \mathbf{x}^*) \geq (1 - \theta)F(\mathbf{x}^*)$.*

⁵Note that since F is not necessarily monotone, $F(\mathbf{x} \vee \mathbf{x}^*)$ could be less than $F(\mathbf{x}^*)$. In [16] the lemma is shown when \mathbf{x}^* is an optimum integer solution, while we show it with respect to a fractional solution \mathbf{x}^* .

PROOF. Let $\mathbf{y} = (\mathbf{x} \vee \mathbf{x}^*) - \mathbf{x}^*$ and consider the ray $\mathbf{r}(\lambda) = \mathbf{x}^* + \lambda \mathbf{y}$, $\lambda \geq 0$. This ray lies in the non-negative orthant because $\mathbf{y} \geq 0$. Consider the point $\mathbf{z} = \mathbf{r}(\lambda')$ where $\lambda' = \frac{1}{\theta}$; note that $\lambda' \geq 1$. Fix a coordinate i . If $x_i \leq x_i^*$ then $y_i = 0$ and $z_i = x_i^* \leq 1$. Otherwise, $z_i = x_i^* + \lambda' y_i = x_i^* + \frac{1}{\theta}(x_i - x_i^*) \leq \frac{1}{\theta} x_i \leq 1$, by the assumption that $x_i \leq \theta$. Thus, $\mathbf{z} \in [0, 1]^n$, and by nonnegativity, $F(\mathbf{z}) \geq 0$. We have $\mathbf{x} \vee \mathbf{x}^* = \mathbf{r}(1) = (1 - \theta)\mathbf{r}(0) + \theta\mathbf{r}(\frac{1}{\theta}) = (1 - \theta)\mathbf{x}^* + \theta\mathbf{z}$, and so by concavity along the ray $\mathbf{r}(\lambda)$ we obtain:

$$F(\mathbf{x} \vee \mathbf{x}^*) \geq (1 - \theta)F(\mathbf{x}^*) + \theta F(\mathbf{z}) \geq (1 - \theta)F(\mathbf{x}^*).$$

This concludes the proof. \square

In other words, we can lower-bound the value of $\mathbf{x} \vee \mathbf{x}^*$ if the coordinates of \mathbf{x} are not too large. To limit the speed at which coordinates increase, we move along a direction $\mathbf{v}(t)$ that is additionally constrained by $\mathbf{v}(t) \leq \mathbf{1} - \mathbf{x}(t)$; this still allows us to find a direction of sufficient marginal gain.

We assume that $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ are *non-decreasing* convex functions and $P \subseteq [0, 1]^n$ is a down-closed convex polyhedral constraint; i.e., the full constraint set is down-closed. Let $\mathbf{x}(0) = 0$ and for $t \in [0, 1]$:

$$\boxed{\begin{aligned} \mathbf{v}(t) &= \arg \max_{\mathbf{y} \in P} \left\{ \mathbf{y} \cdot \nabla F|_{\mathbf{x}(t)} : \mathbf{y} \leq \mathbf{1} - \mathbf{x}(t), \mathbf{y} \in \mathcal{G}_t \right\} \\ \frac{d\mathbf{x}}{dt} &= \mathbf{v}(t) \end{aligned}} \quad (4)$$

where $\mathcal{G}_t = \{\mathbf{y} : \sum_{i=1}^m w_i(t)g_i(\mathbf{y}) \leq \sum_{i=1}^m w_i(t)\}$ is defined as before. The following lemma bounds each coordinate as a function of time.

LEMMA 8. *At time t , we have $x_i(t) \leq 1 - e^{-t}$.*

PROOF. From the differential equation $\frac{dx_i}{dt} = v_i(t) \leq 1 - x_i$, we obtain $\frac{d}{dt}(e^t x_i(t)) = e^t x_i(t) + e^t \frac{dx_i}{dt} \leq e^t$. Using the initial condition $x_i(0) = 0$, we get $e^t x_i(t) \leq e^t - 1$ and hence $x_i(t) \leq 1 - e^{-t}$. \square

We note that without the constraint $\mathbf{v}(t) \leq \mathbf{1} - \mathbf{x}(t)$, we would obtain $x_i(t) \leq t$ which also leads to a constant factor, but a smaller one than $1/e$. The analysis of the objective value is just a bit more involved than in the monotone case.

THEOREM 9. *Let F be a non-negative smooth submodular function with $F(0) = 0$. The point \mathbf{x}_{out} returned by Algorithm 1 with $\eta = \frac{\ln m}{\epsilon}$, F as the objective function where the subproblem is solved using eq. (4) satisfies $F(\mathbf{x}_{\text{out}}) \geq \frac{1}{e}\text{OPT}$, $\mathbf{x}_{\text{out}} \in P$ and $g_i(\mathbf{x}_{\text{out}}) \leq 1 + \epsilon$ for all i . Further, suppose F is non-negative, $0 \in P$ and $g_i(0) = 0$. Then the modified solution \mathbf{x}'_{out} of Theorem 3 with $x_0 = 0$ obeys all the constraints and satisfies $F(\mathbf{x}'_{\text{out}}) \geq \frac{1}{e}(1 - \epsilon)\text{OPT}$.*

PROOF. The direction vector $\mathbf{z}(t) = \mathbf{x}(t) \vee \mathbf{x}^* - \mathbf{x}(t)$ is feasible for the optimization problem given by eq. (4): $\mathbf{z}(t) \leq \mathbf{1} - \mathbf{x}(t)$, $\mathbf{z}(t) \leq \mathbf{x}^* \in P$ and also $g_i(\mathbf{z}(t)) \leq g_i(\mathbf{x}^*) \leq 1$ by the monotonicity of g_i . Therefore, $\mathbf{z}(t) \cdot \nabla F(\mathbf{x}(t)) \leq \mathbf{v}(t) \cdot \nabla F|_{\mathbf{x}(t)}$. Applying Lemma 5, we obtain: $\frac{d}{dt}F(\mathbf{x}(t)) \geq F(\mathbf{x}(t) + \mathbf{z}(t)) - F(\mathbf{x}(t)) = F(\mathbf{x}(t) \vee \mathbf{x}^*) - F(\mathbf{x}(t)) \geq e^{-t}\text{OPT} - F(\mathbf{x}(t))$, by Lemma 7 and Lemma 8. Rewriting, we have $\frac{d}{dt}(e^t F(\mathbf{x}(t))) = e^t(F(\mathbf{x}(t)) + \frac{dF}{dt}) \geq \text{OPT}$. We obtain $e^t F(\mathbf{x}(t)) \geq t \cdot \text{OPT}$. At time $t = 1$, we obtain $F(\mathbf{x}(1)) \geq \frac{1}{e}\text{OPT}$.

The (near) feasibility of \mathbf{x}_{out} follows from Theorem 3. The modified solution \mathbf{x}'_{out} is feasible via Theorem 3. By submodularity, $F(\mathbf{x}'_{\text{out}}) \geq \frac{1}{1+\epsilon}F(\mathbf{x}_{\text{out}}) \geq \frac{1}{e}(1 - \epsilon)\text{OPT}$. \square

3. WIDTH-INDEPENDENT DISCRETIZATION

We describe a modified version of the basic MWU template where we solve the subproblem only at certain discrete time steps. Namely, suppose we solve the subproblem at a certain time t_\circ . Now consider the differential equation $\frac{dw_i}{dt} = \eta w_i(t)g_i(\mathbf{v}(t_\circ))$, $t \geq t_\circ$, for each $i \in [m]$ using the initial value $\mathbf{w}(t_\circ)$. This is just an approximation to the original differential equation since we have avoided computing $\mathbf{v}(t)$ for $t > t_\circ$. On the other hand, we can solve for this analytically to get $w_i(t) = w_i(t_\circ) \exp((t - t_\circ)\eta g_i(\mathbf{v}(t_\circ)))$. We show that there is a good interval $[t_\circ, t_\circ + \delta]$, for some step size δ , such that the degradation in the quality of the solution is not severe compared to the continuous-time process. The important point is that the largest step size δ that can be taken at t_\circ is a function of $\mathbf{v}(t_\circ)$ and the g_i 's and is variable. Moreover, it will result in a width-independent algorithm. The width-independent analysis via the variable-sized steps is not novel. However we believe that the view point of discretizing a fixed time interval $[0, 1]$ to closely follow the differential equation based analysis is novel in the context of MWU.

REMARK 3. *We will assume that $g_i(x)$ is non-negative on P for $1 \leq i \leq m$, that is, we focus on packing problems. Covering constraints and mixed packing and covering constraints can also be handled in a similar fashion, however we need the idea of dropping covered constraints [28, 39]. We defer these details for a later version.*

Algorithm 2 MWU template with discretization

```

1: procedure MWU-DISCRETE( $f, g_1, \dots, g_m, P, \eta, \epsilon$ ):
2:    $\mathbf{w}(0) = 1$ 
3:    $\mathbf{x} = 0$ 
4:    $t = 0$ 
5:   while  $t < 1$  do
6:     Solve subproblem and obtain  $\mathbf{v}(t) \in P$ 
7:      $\triangleright \sum_{i=1}^m w_i(t)g_i(\mathbf{v}(t)) \leq \sum_{i=1}^m w_i(t)$ 
8:      $\delta = \min\{\frac{\epsilon}{\eta} \cdot \frac{1}{\max_i g_i(\mathbf{v}(t))}, 1 - t\}$ 
9:      $\triangleright \delta$  is positive since  $g_i(\mathbf{v}(t)) \geq 0 \forall i$ 
10:     $\mathbf{x} = \mathbf{x} + \delta \mathbf{v}(t)$ 
11:    for  $i \in [m]$  do
12:       $w_i(t + \delta) = w_i(t) \exp(\eta \delta g_i(\mathbf{v}(t)))$ 
13:    end for
14:     $t = t + \delta$ 
15:  end while
16:  return  $\mathbf{x}_{\text{out}} = \mathbf{x}$ 
17: end procedure

```

LEMMA 10. *Let $\epsilon \in [0, 1/2)$. Fix any iteration of the while loop, and let t_\circ be the value of t at the start of the iteration and δ be the step size for that iteration. Then, $\sum_i w_i(t_\circ + \delta) \leq \exp(\eta \delta (1 + \epsilon)) \sum_i w_i(t_\circ)$.*

PROOF. Extend $\mathbf{w}(t)$ to $t \in [t_\circ, t_\circ + \delta]$, via $w_i(t) = w_i(t_\circ) e^{(t-t_\circ)\eta g_i(\mathbf{v}(t_\circ))}$ for all i . Since $g_i(\mathbf{v}(t_\circ)) \geq 0$, it follows that $\eta g_i(\mathbf{v}(t_\circ)) \leq \epsilon/\delta$. Hence $w_i(t) \leq w_i(t_\circ) \exp((t - t_\circ)\epsilon/\delta)$.

$$\begin{aligned} \frac{d}{dt} \sum_{i=1}^m w_i(t) &= \eta \sum_{i=1}^m w_i(t)g_i(\mathbf{v}(t_\circ)) \\ &\leq \eta e^{(t-t_\circ)\epsilon/\delta} \sum_{i=1}^m w_i(t_\circ)g_i(\mathbf{v}(t_\circ)) \leq \eta e^{(t-t_\circ)\epsilon/\delta} \sum_{i=1}^m w_i(t_\circ), \end{aligned}$$

because $\mathbf{v}(t_\circ)$, obtained via the subproblem, satisfies the weighted combination of the constraints with $\mathbf{w}(t_\circ)$. Integrating both sides in the interval $[t_\circ, t_\circ + \delta]$, and rearranging, we obtain

$$\begin{aligned} \sum_{i=1}^m w_i(t_\circ + \delta) &\leq \left(1 + \frac{\eta\delta(e^\epsilon - 1)}{\epsilon}\right) \sum_{i=1}^m w_i(t_\circ) \\ &\leq \exp\left(\frac{\eta\delta(e^\epsilon - 1)}{\epsilon}\right) \sum_{i=1}^m w_i(t_\circ). \end{aligned}$$

Using the approximation, $e^\epsilon \leq 1 + \epsilon + \epsilon^2$ for $\epsilon \in [0, 1/2)$, we obtain the desired bound. \square

THEOREM 11. *Let $\epsilon \in [0, 1/2)$. If $\eta = \ln m / \epsilon$ the point \mathbf{x}_{out} returned by Algorithm 2 satisfies $\mathbf{x}_{\text{out}} \in P$ and $g_i(\mathbf{x}_{\text{out}}) \leq 1 + 2\epsilon$ for all i . The number of iterations is $O\left(\frac{m \ln m}{\epsilon^2}\right)$.*

PROOF. We set up some notation. The number of iterations of the while loop is given by T . Let δ_j be the step size chosen in the j -th iteration for $j \in \{0, 1, \dots, T-1\}$. Let $t_j = \sum_{\ell=0}^{j-1} \delta_\ell$ denote the value of t at the start of the j -th iteration. Thus $t_0 = 0$ and define $t_T = \sum_{j=0}^{T-1} \delta_j = 1$. We use $w_i(t_j)$ and $\mathbf{v}(t_j)$ to refer to the values of the appropriate quantities at time t_j . For each j :

1. $\ln w_i(t_{j+1}) = \ln w_i(t_j) + \eta\delta_j g_i(\mathbf{v}(t_j)) \geq \ln w_i(t_j) \forall i$;
2. $\ln(\sum_i w_i(t_{j+1})) \leq \ln(\sum_i w_i(t_j)) + \eta\delta_j(1 + \epsilon)$, by Lemma 10.

The output \mathbf{x}_{out} equals $\sum_{j=0}^{T-1} \delta_j \mathbf{v}(t_j)$. By convexity, $\mathbf{x}_{\text{out}} \in P$. Fix $i \in [m]$. Applying the fact that g_i is convex followed by the identity in 1) above, we have:

$$\begin{aligned} g_i(\mathbf{x}_{\text{out}}) &= g_i\left(\sum_{j=0}^{T-1} \delta_j \mathbf{v}(t_j)\right) \leq \sum_{j=0}^{T-1} \delta_j g_i(\mathbf{v}(t_j)) \\ &= \frac{1}{\eta} \sum_{j=0}^{T-1} \ln w_i(t_{j+1}) - \ln w_i(t_j) = \frac{\ln w_i(1)}{\eta}. \end{aligned}$$

To bound the right side above, we sum the inequality in 2) above over all j . Since $\sum_i w_i(0) = m$, we obtain:

$$\ln\left(\sum_i w_i(1)\right) \leq \sum_{j=0}^{T-1} \eta(1 + \epsilon)\delta_j + \ln m = \eta(1 + \epsilon) + \ln m \quad (5)$$

Thus, $g_i(\mathbf{x}_{\text{out}}) \leq 1 + \epsilon + \frac{\ln m}{\eta} = 1 + 2\epsilon$, for the choice of η .

We now bound the number of iterations. Fix an iteration j . Note that if $\delta_j = 1 - t_j$, then the algorithm will terminate at the end of that iteration. Therefore, $\eta\delta_j \max_i g_i(\mathbf{v}(t_j)) = \epsilon$ for all $j < T - 1$. If equality is achieved at $i \in [m]$, then by (i) above, $\ln w_i(t_{j+1}) = \ln w_i(t_j) + \epsilon$. Since the weights never decrease, aggregated over all $j < T - 1$, there is at least one $i \in [m]$ such that $\ln w_i(1) \geq (T - 1)\epsilon/m + \ln w_i(0) = (T - 1)\epsilon/m$. Therefore, by eq. (5) above:

$$T \leq \frac{m}{\epsilon}(\eta(1 + \epsilon) + \ln m) + 1 = O\left(\frac{m \ln m}{\epsilon^2}\right),$$

for the choice of η . \square

Discretization to handle the objective function: We have seen a discretization step that ensures that the weight updates track the differential equation. This also results in a width-independent number of iterations. So far we have ignored the objective function. For linear objectives with pure

linear packing constraints the step-size is not constrained by the objective function, and as we mentioned in Section 2.2, there is an optimum solution to the oracle with only one non-zero coordinate. For more complicated objective functions such as concave functions and submodular functions we may use an oracle based on gradient of f . In such cases the step size may be constrained to be smaller than what is necessary for the weight update in line 8 of Algorithm 2. For instance, to keep things simple, suppose the step size cannot be more than some fixed quantity α in order to preserve the objective to some desired accuracy. Then line 8 can be changed to $\delta = \min\{\alpha, \frac{\epsilon}{\eta} \cdot \frac{1}{\max_i g_i(\mathbf{v}(t))}, 1 - t\}$. Then one can prove that the number of iterations of the algorithm will be $O\left(\frac{m \ln m}{\epsilon^2} + \frac{1}{\alpha}\right)$. Of course, one could also choose variable step sizes depending on the objective function and one needs to do a corresponding analysis to bound the overall number of iterations.

3.1 Discretization for multilinear relaxation of monotone submodular functions

Here we consider the problem $\max\{F(\mathbf{x}) : \mathbf{x} \geq 0, A\mathbf{x} \leq \mathbf{1}\}$ where $F : [0, 1]^N \rightarrow \mathbb{R}_+$ is the multilinear extension of a monotone submodular function⁶, and $A\mathbf{x} \leq \mathbf{1}$ is a system of m linear packing constraints ($A \geq 0$). We assume that these constraints include in particular the constraint $\mathbf{x} \leq \mathbf{1}$, so that $\mathbf{x} \in [0, 1]^N$ for any feasible solution. We also assume for now that F and its first partial derivatives $\frac{\partial F}{\partial x_i}$ are accessible directly through a value oracle. (Which is the case in certain applications where F has an explicit form.) Later, we discuss the issue of estimating $F(\mathbf{x})$ from values oracle for f by random sampling.

For monotone submodular functions, the multilinear extension satisfies $\frac{\partial F}{\partial x_i} \geq 0$ and $\frac{\partial^2 F}{\partial x_i \partial x_j} \leq 0$ for all $i, j \in N$. Moreover, it is linear in each coordinate, i.e. $\frac{\partial^2 F}{\partial x_i^2} = 0$ for all $i \in N$. The general framework for applying multiplicative weight updates to this problem was given in Section 2.3. We use Algorithm 2 where the subproblem to be solved is implemented via an appropriate discretization of eq. (3). At time t with step size δ , we have

$$\begin{aligned} \mathbf{v}(t) &= \arg \max_{\mathbf{y} \in P} \left\{ \mathbf{y} \cdot \nabla F|_{\mathbf{x}(t)} : \sum_{i,j} w_i(t) A_{ij} y_j \leq \sum_i w_i(t) \right\} \\ \mathbf{x}(t + \delta) &= \mathbf{x}(t) + \delta \mathbf{v}(t) \end{aligned}$$

We note some simplifications due to the special case that we consider here: the polyhedron P is the non-negative orthant and the constraint functions are linear. This implies that the rule for selecting an optimal direction in each iteration becomes much simpler here. Since we have only one linear constraint per iteration here, the optimal direction is simply $\mathbf{v}(t) = \frac{\sum_i w_i(t)}{\sum_i w_i(t) A_{ij^*}} \mathbf{e}_{j^*}$ for the coordinate j^* that maximizes the benefit/cost ratio $\frac{\sum_i w_i(t)}{\sum_i w_i(t) A_{ij^*}} \frac{\partial F}{\partial x_{j^*}}$. Following the general framework, the time step is chosen as $\delta = \min_i \frac{\epsilon}{\eta A_{ij^*} v_{j^*}}$. (Here we run from time 0 to $1 - 2\epsilon$, to ensure feasibility of the final solution.) Therefore, if coordinate x_{j^*} is chosen then it is incremented by $\Delta_{j^*} = \delta v_{j^*} = \min_i \frac{\epsilon}{\eta A_{ij^*}}$. Note

⁶The continuous framework in Section 2.3 is applicable for any monotone smooth submodular function. However, one can algorithmically exploit the multilinearity of F when it is the extension of a discrete set function f .

that this quantity depends only on j^* . From here, the corresponding time increment can be written as $\delta = \Delta_{j^*}/v_{j^*} = \frac{\sum_i w_i(t) A_{ij^*}}{\sum_i w_i(t)} \Delta_{j^*}$. We obtain the following algorithm.

Algorithm 3 MWU for Monotone Submodular Functions with Linear Constraints

```

1: procedure MWU-MONOSUBMOD( $F, N, A, \eta, \epsilon$ ):
2:    $\mathbf{x} = 0$ 
3:   for  $i = 1$  to  $m$  do
4:      $w_i = 1$ 
5:   end for
6:   for  $j \in N$  do
7:      $\Delta_j = \min_i \frac{\epsilon}{\eta A_{ij}}$ 
8:   end for
9:    $t = 0$ 
10:  while  $t < 1 - 2\epsilon$  do
11:     $j^* = \arg \max_{j \in N} \left( \frac{1}{\sum_i A_{ij} w_i} \frac{\partial F}{\partial x_j} \Big|_{\mathbf{x}} \right)$ 
12:     $x_{j^*} = x_{j^*} + \Delta_{j^*}$ 
13:     $t = t + \frac{\sum_i w_i A_{ij^*}}{\sum_i w_i} \Delta_{j^*}$ 
14:    for  $i = 1$  to  $m$  do
15:       $w_i = w_i e^{\eta A_{ij^*} \Delta_{j^*}}$ 
16:    end for
17:  end while
18:  return  $\mathbf{x}$ 
19: end procedure

```

We analyze the algorithm as follows. Note that the algorithm stops at time $t = 1 - 2\epsilon$. This ensures that the output solution is feasible. This follows from the analysis in the proof of Theorem 11. Next, because F is multilinear and $\mathbf{v}(t)$ is a basis vector, $F(\mathbf{x}(t+\delta)) - F(\mathbf{x}(t)) = \delta \mathbf{v}(t) \cdot \nabla F|_{\mathbf{x}(t)}$. (Here, δ denotes the appropriate time step, which is $\delta = \frac{\sum_i w_i A_{ij^*}}{\sum_i w_i} \Delta_{j^*}$ whenever x_{j^*} is being incremented.) Similar to the proof of Theorem 6, the vector $\mathbf{z}(t) = (\mathbf{x}(t) \vee \mathbf{x}^*) - \mathbf{x}(t)$ is a candidate solution to the optimization problem, so $\mathbf{v}(t) \cdot \nabla F|_{\mathbf{x}(t)} \geq \mathbf{z}(t) \cdot \nabla F|_{\mathbf{x}(t)}$. By monotonicity and submodularity, $\mathbf{z}(t) \cdot \nabla F|_{\mathbf{x}(t)} \geq F(\mathbf{x}(t) + \mathbf{z}(t)) - F(\mathbf{x}(t)) \geq \text{OPT} - F(\mathbf{x}(t))$. Thus, $\mathbf{v}(t) \cdot \nabla F(\mathbf{x}(t)) \geq \text{OPT} - F(\mathbf{x}(t))$. Therefore, $F(\mathbf{x}(t+\delta)) - F(\mathbf{x}(t)) = \delta(\text{OPT} - F(\mathbf{x}(t)))$. Rewriting we get $\text{OPT} - F(\mathbf{x}(t+\delta)) \leq (1-\delta)(\text{OPT} - F(\mathbf{x}(t)))$.

Using the notation in the proof of Theorem 11, if \mathbf{x}_{out} is the solution and t_{out} is the time at termination of the algorithm, we obtain

$$\text{OPT} - F(\mathbf{x}_{\text{out}}) \leq (\text{OPT} - F(\mathbf{x}(0))) \cdot \prod_j (1 - \delta_j) \leq \text{OPT} \cdot e^{-t_{\text{out}}}.$$

We have $t_{\text{out}} \geq 1 - 2\epsilon$ and hence $F(\mathbf{x}_{\text{out}}) \geq (1 - e^{-1+2\epsilon}) \text{OPT} \geq (1 - 1/e - \epsilon) \text{OPT}$.

Implementation details that yield the claimed running time are deferred to a longer version of the paper.

3.2 Discretization for non-monotone submodular functions

Now let us turn to the case of non-monotone submodular functions. We only assume here that F arises as a multilinear extension from a non-negative submodular function. Again, we follow the continuous-time framework from Section 2.4. We recall that the subproblem to solve in the

continuous framework is as follows:

$$\mathbf{v}(t) = \arg \max_{0 \leq \mathbf{y} \leq \mathbf{1} - \mathbf{x}} \{ \mathbf{y} \cdot \nabla f|_{\mathbf{x}(t)} : \sum_{i,j} w_i(t) A_{ij} y_j \leq \sum_i w_i(t) \} \quad (6)$$

(see Section 2.4). Here, the optimal direction $\mathbf{v}(t)$ is not necessarily a single coordinate, which makes the analysis somewhat more involved and the running time slower. The main added issue is that the objective function is no longer linear when moving along a general direction $\mathbf{v}(t)$, and hence we have another reason to be careful about the step size. (This was already an issue in the original implementation of the continuous greedy algorithm [37, 11].) To ensure that we get sufficient gains on the objective function, we force the time steps to be bounded by $\min_i \frac{\epsilon}{\eta n \sum_j A_{ij} v_j}$ where $n = |N|$, instead of $\min_i \frac{\epsilon}{\eta \sum_j A_{ij} v_j}$. We obtain the following algorithm.

Algorithm 4 MWU for Nonmonotone Submodular Functions with Linear Constraints

```

1: procedure MWU-NONMONOSUBMOD( $F, N, A, \eta, \epsilon$ ):
2:    $\mathbf{x} = 0$ 
3:   for  $i = 1$  to  $m$  do
4:      $w_i = 1$ 
5:   end for
6:   for  $j \in N$  do
7:      $\Delta_j = \min\{ \min_i \frac{\epsilon}{\eta n A_{ij}}, \frac{\epsilon}{\eta n} \}$ 
8:   end for
9:    $t = 0$ 
10:  while  $t < 1 - 2\epsilon$  do
11:     $\mathbf{v} = \arg \max_{\mathbf{y}} \{ \mathbf{y} \cdot \nabla F|_{\mathbf{x}+\Delta} : 0 \leq \mathbf{y} \leq \mathbf{1} - \mathbf{x}, \sum_{i,j} w_i A_{ij} y_j \leq \sum_i w_i \}$ 
12:     $\delta = \min\{ \min_i \frac{\epsilon}{\eta n \sum_j A_{ij} v_j}, \frac{\epsilon}{\eta n}, 1 - t \}$ 
13:     $\mathbf{x} = \mathbf{x} + \delta \mathbf{v}$ 
14:     $t = t + \delta$ 
15:    for  $i = 1$  to  $m$  do
16:       $w_i = w_i e^{\eta \delta \sum_j A_{ij} v_j}$ 
17:    end for
18:  end while
19:  return  $\mathbf{x}$ 
20: end procedure

```

LEMMA 12. *The total number of iterations is at most $(2 + \epsilon)m n \eta / \epsilon$.*

PROOF. Suppose that the number of iterations is $T > (2 + \epsilon)m n \eta / \epsilon$. Clearly, there are at most $n \eta / \epsilon \leq m n \eta / \epsilon$ steps where the step size is $\delta = \frac{\epsilon}{\eta n}$. Therefore, there must be at least $(1 + \epsilon)m n \eta / \epsilon$ steps where the step size is $\delta = \frac{\epsilon}{\eta \sum_j A_{ij} v_j}$ for some $i \in [m]$, and by the pigeonhole principle there is a particular $i \in [m]$ for which there are at least $(1 + \epsilon)n \eta / \epsilon$ such steps. By the multiplicative weight update, w_i increases by a factor of $e^{\epsilon/n}$ in such a step. Therefore, w_i would reach a value of $e^{(1+\epsilon)\eta}$, which cannot happen by the proof of Theorem 11. \square

LEMMA 13. *For $\eta = \frac{\ln m}{\epsilon}$, at termination we have $\mathbf{x} \in P$ and $\sum_j A_{ij} x_j \leq 1 + 2\epsilon$ for each i .*

LEMMA 14. *At time t , we have $x_j \leq 1 - e^{-(1+\epsilon/\eta)t}$ for each $j \in N$.*

PROOF. Note that at each step, we have $v_j(t) \leq 1 - x_j(t)$, and we increment $x_j(t)$ to $x_j(t + \delta) = x_j(t) + \delta v_j(t)$. Therefore, $1 - x_j(t + \delta) = 1 - x_j(t) - \delta v_j(t) \geq 1 - x_j(t) - \delta(1 - x_j(t)) = (1 - \delta)(1 - x_j(t))$. Using $\delta \leq \epsilon/\eta$, we get $1 - \delta \geq e^{-(1+\epsilon/\eta)\delta}$, and $1 - x_j(t + \delta) \geq e^{-(1+\epsilon/\eta)\delta}(1 - x_j(t))$. By induction, starting from $x_j(0) = 0$, we obtain $1 - x_j(t) \geq e^{-(1+\epsilon/\eta)t}$. \square

Next, we prove a lower bound on the gain that the algorithm makes in one step. Note that instead of considering the gradient at the current point $\mathbf{x}(t)$, the algorithm considers a slightly higher point $\mathbf{x}(t) + \Delta$. First, we show that the increment Δ is chosen so that it upper-bounds any possible step that the algorithm could make at this point.

LEMMA 15. Define $\Delta \in \mathbb{R}_+^n$ as

$$\Delta_j = \min \left\{ \min_i \frac{\epsilon}{\eta n A_{ij}}, \frac{\epsilon}{\eta n} \right\}.$$

Then at any time t , if the current solution is $\mathbf{x}(t)$, then after one step the solution is $\mathbf{x}(t + \delta) \leq \mathbf{x}(t) + \Delta$.

PROOF. Given a direction vector $\mathbf{v}(t)$ found by the algorithm, the step size is chosen as

$$\delta = \min \left\{ \min_i \frac{\epsilon}{\eta n \sum_j A_{ij} v_j}, \frac{\epsilon}{\eta n}, 1 - t \right\}.$$

Therefore, the increment δv_j in coordinate x_j is upper-bounded by $\frac{\epsilon}{\eta n}$ (since $v_j \leq 1$) and also by

$$\delta v_j \leq \min_i \frac{\epsilon v_j}{\eta n \sum_{j'} A_{ij'} v_{j'}} \leq \min_i \frac{\epsilon}{\eta n A_{ij}}.$$

This completes the proof. \square

LEMMA 16. If the current solution is $\mathbf{x}(t)$, then there is a vector $0 \leq \mathbf{v} \leq \mathbf{1} - \mathbf{x}(t)$ such that $\sum_{i,j} w_i A_{ij} v_j \leq \sum_i w_i$ and

$$\mathbf{v} \cdot \nabla F \Big|_{\mathbf{x}(t) + \Delta} \geq \left(e^{-t} - \frac{2\epsilon}{\eta} \right) OPT - F(\mathbf{x}(t) + \Delta).$$

PROOF. Consider the optimum \mathbf{y}^* , $OPT = F(\mathbf{y}^*)$. The direction vector

$$\mathbf{v} = ((\mathbf{x}(t) + \Delta) \vee \mathbf{y}^*) - (\mathbf{x}(t) + \Delta)$$

satisfies $0 \leq \mathbf{v} \leq \mathbf{1} - \mathbf{x}(t)$ and $\sum_{i,j} w_i A_{ij} v_j \leq \sum_i w_i$, by the feasibility of \mathbf{y}^* .

Next, by the concavity of F along non-negative directions,

$$\begin{aligned} \mathbf{v} \cdot \nabla F \Big|_{\mathbf{x}(t) + \Delta} &\geq F(\mathbf{x}(t) + \Delta + \mathbf{v}) - F(\mathbf{x}(t) + \Delta) \\ &= F((\mathbf{x}(t) + \Delta) \vee \mathbf{y}^*) - F(\mathbf{x}(t) + \Delta). \end{aligned}$$

Since we have $x_j(t) \leq 1 - e^{-(1+\epsilon/\eta)t}$ by Lemma 14, we have $x_j(t) + \Delta_j \leq 1 - e^{-(1+\epsilon/\eta)t} + \frac{\epsilon}{\eta n}$. Therefore, by Lemma 7, $F((\mathbf{x}(t) + \Delta) \vee \mathbf{y}^*) \geq (e^{-(1+\epsilon/\eta)t} - \frac{\epsilon}{\eta n}) F(\mathbf{y}^*) = (e^{-(1+\epsilon/\eta)t} - \frac{\epsilon}{\eta n}) OPT$. We estimate $e^{-(1+\epsilon/\eta)t} \leq (1 - \epsilon t/\eta) e^{-t} \geq e^{-t} - \epsilon/\eta$ which finishes the proof. \square

We need one more bound, comparing the values of $F(\mathbf{x}')$ for $\mathbf{x} \leq \mathbf{x}' \leq \mathbf{x} + \Delta$. Since Δ was chosen to be ‘‘small’’, we prove that these values cannot differ by too much.

LEMMA 17. For any pair of points such that $0 \leq \mathbf{x}' - \mathbf{x} \leq \Delta$, we have $F(\mathbf{x}') - F(\mathbf{x}) \leq \frac{\epsilon}{\eta} OPT$.

LEMMA 18. For each time step, define $\phi : [t, t + \delta] \rightarrow \mathbb{R}$ as $\phi(t + \lambda) = F(\mathbf{x}(t) + \lambda \mathbf{v}(t))$ for $\lambda \in [0, \delta]$. If $\eta \geq 3$, then for each $\tau \in (t, t + \delta)$, we have

$$\phi'(\tau) \geq (e^{-\tau} - \epsilon) OPT - \phi(\tau).$$

LEMMA 19. At time $t = 1$, we obtain

$$F(\mathbf{x}(1)) \geq \left(\frac{1}{e} - \epsilon \right) OPT.$$

COROLLARY 20. For $\eta = \ln m/\epsilon$, the algorithm runs in $O(mn \log m/\epsilon^2)$ iterations and returns a solution of value $F(\mathbf{x}) \geq (1/e - \epsilon) OPT$.

Implementation details that yield the claimed running time are deferred to a longer version of the paper.

Acknowledgments: We thank Ken Clarkson and Neal Young for several helpful discussions.

4. REFERENCES

- [1] S. Arora, E. Hazan, and S. Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *Proc. 46th FOCS*, pages 339–348. IEEE Comp. Soc. Press, 2005.
- [2] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.
- [3] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proc. 39th STOC*, pages 227–236. ACM Press, 2007.
- [4] Y. Azar and I. Gamzu. Efficient submodular function maximization under linear packing constraints. In A. Czumaj, K. Mehlhorn, A. M. Pitts, and R. Wattenhofer, editors, *ICALP (1)*, volume 7391 of *Lecture Notes in Computer Science*, pages 38–50. Springer, 2012.
- [5] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, pages 671–680. ACM, 2014.
- [6] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514, 2014.
- [7] N. Bansal, N. Korula, V. Nagarajan, and A. Srinivasan. Solving packing integer programs via randomized rounding with alterations. *Theory of Computing*, 8(1):533–565, 2012.
- [8] D. Bienstock. *Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice*. Springer, 2002.
- [9] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957. SIAM, 2014.
- [10] G. Călinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In M. Fischetti and D. P. Williamson, editors, *IPCO*, volume 4513 of *Lecture Notes in Computer Science*, pages 182–196. Springer, 2007.

- [11] G. Călinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.
- [12] C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 575–584. IEEE, 2010.
- [13] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. In L. Fortnow and S. P. Vadhan, editors, *STOC*, pages 783–792. ACM, 2011.
- [14] F. Chierichetti, R. Kumar, and A. Tomkins. Max-cover in map-reduce. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *WWW*, pages 231–240. ACM, 2010.
- [15] F. Diedrich and K. Jansen. Faster and simpler approximation algorithms for mixed packing and covering problems. *Theoretical computer science*, 377(1):181–204, 2007.
- [16] M. Feldman, J. Naor, and R. Schwartz. A unified continuous greedy algorithm for submodular maximization. In R. Ostrovsky, editor, *FOCS*, pages 570–579. IEEE, 2011.
- [17] L. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, 2000.
- [18] N. Garg and R. Khandekar. Fractional covering with upper bounds on the variables: Solving LPs with negative entries. In *Proc. 12th Ann. Europ. Symp. on Algorithms (ESA'04)*, pages 371–382, 2004.
- [19] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proc. 39th FOCS*, pages 300–309. IEEE Comp. Soc. Press, 1998.
- [20] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37:630–652, 2007.
- [21] J. Gillenwater, A. Kulesza, and B. Taskar. Near-optimal map inference for determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 2735–2743, 2012.
- [22] M. Grigoriadis and L. Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization*, 4(1):86–107, 1994.
- [23] M. D. Grigoriadis, L. G. Khachiyan, L. Porkolab, and J. Villavicencio. Approximate max-min resource sharing for structured concave optimization. *SIAM Journal on Optimization*, 11(4):1081–1091, 2001.
- [24] R. Iyer, S. Jegelka, and J. Bilmes. Fast semidifferential-based submodular function optimization. In *Proceedings of The 30th International Conference on Machine Learning*, pages 855–863, 2013.
- [25] K. Jansen and H. Zhang. Approximation algorithms for general packing problems and their application to the multicast congestion problem. *Mathematical Programming*, 114(1):183–206, 2008.
- [26] S. Kale. *Efficient Algorithms Using The Multiplicative Weights Update Method*. PhD thesis, Princeton University, 2006. Technical Report TR-804-07.
- [27] R. Khandekar. *Lagrangian Relaxation based Algorithms for Convex Programming Problems*. PhD thesis, Indian Institute of Technology, Delhi, 2004.
- [28] J. Könemann. Fast combinatorial algorithms for packing and covering problems. Master’s thesis, Universität des Saarlandes, 1998. Available at: <http://www.math.uwaterloo.ca/jochen/docs/diplom.ps>.
- [29] C. Koufogiannakis and N. E. Young. Beating simplex for fractional packing and covering linear programs. In *FOCS*, pages 494–504. IEEE Computer Society, 2007.
- [30] A. Kulik, H. Shachnai, and T. Tamir. Approximations for monotone and nonmonotone submodular maximization with knapsack constraints. *Mathematics of Operations Research*, 38(4):729–739, 2013.
- [31] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in mapreduce and streaming. In G. E. Blelloch and B. Vöcking, editors, *SPAA*, pages 1–10. ACM, 2013.
- [32] J. Kwon and P. Mertikopoulos. A continuous-time approach to online optimization. *arXiv preprint arXiv:1401.6956*, 2014.
- [33] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM J. Discrete Math.*, 23(4):2053–2078, 2010.
- [34] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *ACM STOC*, pages 448–457, 1993.
- [35] S. A. Plotkin, D. B. Shmoys, and É. Tardos. Fast approximation algorithm for fractional packing and covering problems. *Math. Oper. Res.*, 20:257–301, 1995. Preliminary version in FOCS’91.
- [36] S. Sorin. Exponential weight algorithm in continuous time. *Mathematical Programming*, 116(1-2):513–528, 2009.
- [37] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In C. Dwork, editor, *STOC*, pages 67–74. ACM, 2008.
- [38] N. E. Young. Randomized rounding without solving the linear program. In *Proc. 6th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA’95)*, pages 170–178. ACM Press, 1995.
- [39] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings of 42nd IEEE Symposium on Foundations of Computer Science*, pages 538–546. IEEE, 2001.
- [40] N. E. Young. Nearly linear-time approximation schemes for mixed packing/covering and facility-location linear programs. *ArXiv e-prints*, July 2014.