Online Primal-Dual Algorithms with Configuration Linear Programs

Nguyễn Kim Thắng*
IBISC, University Paris-Saclay, France

Abstract

In this paper, we present online primal-dual algorithms for covering/packing problems with *non-convex* objectives. Convex objectives have been extensively studied in recent years in which analyses rely crucially on the convexity and the Fenchel duality. However, problems with non-convex objectives resist against current approaches and non-convexity represents a strong barrier in optimization in general and in the design of online algorithms in particular. In our approach, we consider configuration linear programs with the multilinear extension of the objectives. We follow the multiplicative weight update framework in which a novel point is that the primal update is defined based on the gradient of the multilinear extension. We introduce a new notion called *(local) smoothness* in order to characterize the competitiveness of our algorithms. The approach leads to competitive algorithms for covering/packing problems with non-convex objectives. In particular, we give competitive algorithms for online submodular minimization/maximization problems under general covering/packing constraints. Prior to our work, no competitive algorithm is known for online submodular optimization in these general settings.

Besides, building upon the resilient ideas from the primal-dual framework with configuration LPs, we present a simple greedy algorithm for a general cost minimization problem, which encompasses several well-studied online problems. The algorithm gives *optimal* (up to a constant factor) competitive ratios for problems of network routing, vector scheduling, energy-efficient scheduling and non-convex facility location.

*Research supported by the ANR project OATA n° ANR-15-CE40-0015-01

1 Introduction

In the paper, we consider problems of minimizing the total cost of resources used to satisfy online requests. One phenomenon, known as the *economy of scale*, is that the cost grows sub-linearly with the amount of resources used. This happens in many scenarios in which one gets a discount when buying resources in bulk. A representative setting is the extensively-studied domain of sub-modular optimization. Another phenomenon, known as the *diseconomy of scale*, is that the cost grows super-linearly on the quantity of used resources. An illustrative example for this phenomenon is the energy cost in computation where the cost grows super-linearly, typically as a convex function. The diseconomy of scale has been widely studied in the domain of convex optimization [14]. However, in many settings, the costs are the mix of both phenomena and the objective functions are indeed non-convex. Non-convex objective functions appear in various problems, ranging from scheduling, sensor energy management, to influence and revenue maximization, and facility location. For example, in scheduling of malleable jobs on parallel machines, the cost grows as a non-convex function [34] which is due to the parallelization and the synchronization. Besides, in the practical aspect of facility location, the facility costs to serve clients are rarely constant or simply a convex function of the number of clients. The costs would initially increase fast until some threshold on the number of clients, then become more stable before quickly increase again as the number of clients augments. This behaviour of cost functions widely happens in economic contexts. Such situations raise the demand of designing algorithms with performance guarantee for non-convex objective functions. In this paper, we consider problems in which the cost increases *arbitrarily* with the amount of used resources. We measure the performance of an algorithm by the *competitive ratio*. Specifically, an algorithm is r-competitive if for any instance, the ratio between the cost of the algorithm and that of an optimal solution is at most r.

1.1 Primal-Dual Approach for $0-1$ Covering Problems

$0-1$ Covering Problems. Let $\mathcal{E}$ be a set of n resources and let $f : \{0,1\}^n \to \mathbb{R}^+$ be an *arbitrary* monotone cost function. Let $x_e \in \{0,1\}$ be a variable indicating whether resource e is selected. The covering constraints $\sum_e a_{i,e}x_e \geq 1$ for every i are revealed one-by-one and at any step, one needs to maintain a feasible integer solution $\mathbf{x}$. The goal is to design an algorithm that minimizes $f(\mathbf{x})$ subject to the online covering constraints and $x_e \in \{0,1\}$ for every e.

Approach and Contribution. We consider an approach based on linear programming. The first crucial step for any LP-based approach is to derive a LP formulation with reasonable *integrality gap*, which is defined as the ratio between the optimal integer solution of the formulation and the optimal solution with the integrality conditions relaxed. As the cost functions are non-linear, it is not surprising that the natural relaxation suffers from large integrality gap. This issue has been observed and resolved by Makarychev and Sviridenko [41]. Makarychev and Sviridenko [41] considered an offline variant of the problem in which the resource cost functions are of form $f(z) = z^\alpha$ for constant α. They systematically strengthen the natural formulations by introducing an exponential number of new variables and new constraints connecting new variables to original ones. Consequently, the new formulation, in the form of a configuration LP, significantly reduces the integrality gap.

The configuration LPs have been used mostly in offline settings and the approach is to round an optimal fractional solution to an integer one and bound the approximation ratio. The first encountered difficulty of this approach is that a configuration LP has exponential size, so one has to look for a separation oracle in order to compute an optimal fractional solution. Finding

separation oracles is in general far from trivial and it represents an obstacle in using configuration LP to design performant algorithms. Besides, many rounding schemes are *intrinsically offline* (including the one in [41]) and it is not suitable in online setting where input is released in pieces.

We consider primal-dual approaches in our paper. Very recently, Azar et al. [8] have presented a general primal-dual framework when function f is convex with monotone gradient. The framework is indeed inspired by the Buchbinder-Naor framework [15] for linear objectives. This work is along the line of current results for convex objectives [23, 7, 42, 31, 33, 24, 8], which rely crucially on the convexity of cost functions and Fenchel duality.

We overcome this obstacle for non-convex functions (and also for convex functions with non-monotone gradients) by considering configuration LP corresponding to the problem and the multilinear extension of function f. Given $f : \{0,1\}^n \to \mathbb{R}^+$, its *multilinear extension* $F : [0,1]^n \to \mathbb{R}^+$ is defined as $F(\mathbf{x}) := \sum_S \prod_{e\in S} x_e \prod_{e\notin S}(1-x_e) \cdot f(\mathbf{1}_S)$ where $\mathbf{1}_S$ is the characteristic vector of S (i.e., the e^{th}-component of $\mathbf{1}_S$ equals 1 if $e \in S$ and equals 0 otherwise). An alternative way to define F is to set $F(\mathbf{x}) = \mathbb{E}\big[f(\mathbf{1}_T)\big]$ where T is a random set such that a resource e appears independently in T with probability x_e. Note that $F(\mathbf{1}_S) = f(\mathbf{1}_S)$.

Building upon the primal-dual framework in [8, 15], we present a competitive algorithm, for the fractional $0-1$ covering problem. Specifically, we introduce the notion of *locally-smooth* for minimization problems and characterize the competitive ratio using the local smoothness' parameters. Given two vectors $\mathbf{x}$ and $\mathbf{y}$ in $[0,1]^n$, let $\mathbf{x} \vee \mathbf{y}$ be the vector such that its component at coordinate e' is $\max\{x_{e'}, y_{e'}\}$.

Definition 1 *Let $\mathcal{E}$ be a set of n resources. A differentiable function $F : [0,1]^n \to \mathbb{R}^+$ is (λ,μ)-*min-locally-smooth *if for any set $S \subset \mathcal{E}$, and for any vectors $\mathbf{x}^e \in [0,1]^n$ where $e \in \mathcal{E}$, the following inequality holds.*

$$\sum_{e\in S} \nabla_e F(\mathbf{x}^e) \leq \lambda F\big(\mathbf{1}_S\big) + \mu F\big(\mathbf{x}\big) \tag{1}$$

where $\mathbf{x} := \bigvee_{e\in S} \mathbf{x}^e$, meaning that $x_{e'} = \max_e\{x^e_{e'}\}$ for every coordinate e'.

If the gradient $\nabla F(\mathbf{x})$ is non-decreasing, we only need a simpler version. We say that a differentiable function $F : [0,1]^n \to \mathbb{R}^+$ with monotone gradient is (λ,μ)-*min-locally-smooth* if for any set $S \subset \mathcal{E}$, and for any vector $\mathbf{x} \in [0,1]^n$, the following inequality holds.

$$\sum_{e\in S} \nabla_e F(\mathbf{x}) \leq \lambda F\big(\mathbf{1}_S\big) + \mu F\big(\mathbf{x}\big) \tag{2}$$

Let us explain intuitively the local smoothness by considering Inequality (2). Imagine that $\mathbf{x}$ is the current solution of an algorithm. Then, if the total increase of the objective function F at the current solution (the left-hand side) can be bounded by a combination of the current cost (the second term of the right-hand side) and the cost of an adversary (the first term of the right-hand side), then the algorithm is competitive with competitive ratio depending on λ and μ.

This definition is inspired by the smoothness framework introduced by Roughgarden [47] in the context of algorithmic game theory to characterize the price of anarchy for large classes of games. We characterize the performance of algorithms using the notion of smoothness in a similar way as the price of anarchy characterized by the smoothness argument [47]. Through this notion, we show an interesting connection between online algorithms and algorithmic game theory.

Specifically, in our approach, the smoothness notion allows us to prove the dual feasibility and also to establish the competitiveness of algorithms. The main result of the paper is a general primal-dual scheme in which the competitive ratio is determined in terms of the locally-smoothness parameters.

Theorem 1 *Let F be the multilinear extension of the objective cost f and d be the maximal row sparsity of the constraint matrix, i.e., $d = \max_i |\{a_{ie} : a_{ie} > 0\}|$. Assume that F is $\big(\lambda, \frac{\mu}{\ln(1+2d^2)}\big)$-min-locally-smooth for some parameters $\lambda > 0$ and $\mu < 1$. Then there exists a $O\big(\frac{\lambda}{1-\mu} \cdot \ln d\big)$-competitive algorithm for the fractional covering problem.*

Our algorithm, as well as the one in [8] for convex with monotone gradients and the recent algorithm for ℓ_k-norms [44], are extensions of the Buchbinder-Naor primal-dual framework [15]. A distinguishing point of our algorithm compared to the ones in [8, 44] relies on the multiplicative update, which is crucial in online primal-dual methods. The approaches in [8, 44] use the gradient $\nabla f(\mathbf{x})$ at the current primal solution $\mathbf{x}$ to define a multiplicative update for the primal. In our approach, we multiplicatively update the primal by some parameter related to the gradient of the multilinear extension $\nabla F(\mathbf{x})$. This parameter is always maintained to be at least $\nabla F(\mathbf{x})$ and in case $\nabla F(\mathbf{x})$ is non-decreasing, the parameter is indeed equal to $\nabla F(\mathbf{x})$. This multiplicative update, together with the configuration LPs and the notion of local smoothness, enable us to derive a competitive algorithm for convex objective functions whose gradients are not necessarily monotone and more generally, for non-convex objectives. Moreover, other advantage of our approach are: (i) it avoids the cumbersome technical details in the analysis as well as in the assumptions of objective functions; (ii) it reduces the analysis of bounding the competitive ratios to determining the local-smoothness parameters.

Applications. Specifically, we apply our algorithm to several widely-studied classes of functions in optimization. First, for the class of non-negative polynomials of degree k, the algorithm yields a $O\big((k \log d)^k\big)$-competitive fractional solution that matches to a result in [8]. Second, beyond convexity, we consider a natural class of non-convex cost functions which represent a typical behaviour of resources in serving demand requests. Non-convexity represents a strong barrier in optimization in general and in the design of algorithms in particular. We show that our algorithm is competitive for this class of functions. Finally, we illustrate the applicability of our algorithm to the class of submodular functions. We make a connection between the local-smooth parameters to the concept of *total curvature* κ of submodular functions. The total curvature has been widely used to determines both upper and lower bounds on the approximation ratios for many submodular and machine learning problems [21, 29, 9, 50, 37, 48]. We show that our algorithm yields a $O\big(\frac{\log d}{1-\kappa}\big)$-competitive fractional solution for the problem of minimizing a submodular function under covering constraints. To the best of our knowledge, the submodular minimization under general convering constraints has not been studied in the online computation setting.

1.2 Primal-Dual Algorithm for $0-1$ Packing Problems

$0-1$ Packing Problems. Let $\mathcal{E}$ be a set of n resources and let $f : \{0,1\}^n \to \mathbb{R}^+$ be an *arbitrary* cost function. Let $x_e \in \{0,1\}$ be a variable indicating whether resource e is selected. The packings constraints $\sum_e b_{i,e} x_e \leq 1$ for every i are given in advance and resources e are revealed online one-by-one. At any time, one needs to maintain a feasible integer solution $\mathbf{x}$. The goal is to design an algorithm that maximizes $f(\mathbf{x})$ subject to the online packing constraints and $x_e \in \{0,1\}$ for every e.

We follow the primal-dual approach to design competitive algorithms for fractional packing problems. We introduce an appropriate notion of smoothness for maximization problems. We notice that this notion is different to that for minimization problems. On one hand, it is due to different natures of minimization and maximization problems. On the other hand, in non-convex problems only weak duality holds while strong duality does not. So informally, there is no symmetry

between primal and dual. Specifically, in linear programming, the dual of the dual is the primal while this property does not hold in non-convex settings.

Definition 2 *A differentiable function $F : [0,1]^n \to \mathbb{R}^+$ is (λ, μ)-*max-locally-smooth *if for any set $S \subset \mathcal{E}$, and for any vectors $\mathbf{x}^e \in [0,1]^n$, the following inequality holds:*

$$\sum_{e \in S} \nabla_e F(\mathbf{x}^e) \geq \lambda F\big(\mathbf{1}_S\big) - \mu F\big(\mathbf{x}\big).$$

where $\mathbf{x} := \bigvee_{e \in S} \mathbf{x}^e$, meaning that $x_{e'} = \max_e \{x^e_{e'}\}$ for any coordinate e'.

Similar to the approach for covering constraints, the max-local-smoothness notion allows us to prove the dual feasibility and also to establish the competitiveness of algorithms, which is determined in terms of the max-locally-smoothness parameters.

Theorem 2 *Let F be the multilinear extension of the objective cost f. Denote the row sparsity $d := \max_i |\{b_{ie} : b_{ie} > 0\}|$ and $\rho := \max_i \max_{e,e' : b_{ie'} > 0} b_{ie}/b_{ie'}$. Assume that F is (λ, μ)-max-locally-smooth for some parameters $\lambda > 0$ and $\mu < 1$. Then there exists a $O\big(\frac{2\ln(1+d\rho)+\mu}{\lambda}\big)$-competitive algorithm for the fractional packing problem.*

Note that when f is a linear function, the smooth parameters $\lambda = 1$ and $\mu = 0$. In this case, the performance guarantee is the same (up to a constant factor) to that of maximizing a linear function under packing constraints [15] and so asymptotically optimal.

Applications. We consider applications to online submodular maximization problems. Submodular functions are interesting since they are neither convex nor concave. Besides, submodular maximization constitutes a major research agenda in optimization, machine learning and has been widely studying. However, in the context of online algorithms, not much has been known especially for submodular maximization with constraints. Designing competitive algorithms for online submodular maximization has been raised in the recent survey [39] as an important direction. Buchbinder et al. [16] have studied the online problem of maximizing the sum of weighted rank functions subject to matroid constraints. The objective here is a particular submodular function. The authors give an algorithm with competitive ratio depending logarithmically on the numbers of elements and the values of weighted rank functions. In another approach, Buchbinder et al. [17] have considered submodular optimization with preemption, where one can reject previously accepted elements, and have given constant competitive algorithms for unconstrained and knapsack-constraint problems.

We show that Algorithm 2 yields competitive fractional solutions for online submodular maximization with packing constraints. The competitive ratio is $O\big(\log(1 + d\rho)\big)$ which is independent of the submodular objective. Note that using the online contention resolution rounding schemes [26], one can obtain randomized algorithms for several specific constraint polytopes, for example, knapsack polytopes, matching polytopes and matroid polytopes.

1.3 A General Problem and Primal-Dual Approach

We consider a problem — an interesting special case in the $0-1$ covering framework — which captures several well-studied problems.

A General Problem. In the problem, there is a set of resources $\mathcal{E}$ and requests arrive online. At the arrival of request i, a set of feasible strategies (actions) $\mathcal{S}_i$ to satisfy request i is revealed. Each strategy $s_{ij} \in \mathcal{S}_i$ consists of a subset of resources in $\mathcal{E}$. Each resource e is associated to an *arbitrary* non-negative non-decreasing cost function f_e and the cost induced by resource e depends on the set of requests using e. The cost of a solution is the total cost of resources, i.e., $\sum_e f_e(A_e)$ where A_e is the set of requests using resource e. The goal is design an algorithm that upon the arrival of each request, selects a feasible strategy for the request while maintaining the cost of the overall solution as small as possible.

We follow the primal-dual approach with configuration LPs. We define a notion of *smoothness* of functions, which is similar to the notion of local-smoothness.

Definition 3 *Let $\mathcal{N}$ be a set of requests. A set function $f : 2^{\mathcal{N}} \to \mathbb{R}^+$ is* (λ, μ)-smooth *if for any set $A = \{a_1, \ldots, a_n\} \subseteq \mathcal{N}$ and any collection $B_1 \subseteq B_2 \subseteq \ldots \subseteq B_n \subseteq B \subseteq \mathcal{N}$, the following inequality holds.*

$$\sum_{i=1}^{n} \big[f\big(B_i \cup a_i\big) - f\big(B_i\big)\big] \leq \lambda f\big(A\big) + \mu f\big(B\big)$$

A set of cost functions $\{f_e : e \in \mathcal{E}\}$ is (λ, μ)-smooth *if every function f_e is* (λ, μ)-smooth.

Given a (λ, μ)-smooth function, the quantity $\frac{\lambda}{1-\mu}$ informally measures how far the function is from being linear. If a function is linear then it is $(1, 0)$-smooth. The notion of smoothness is different to the locally smoothness but they share some similarities. Intuitively, the inequality in the definition of smoothness means the following. Imagine that B_i is the current solution of an algorithm at step i. Then, if the total marginal increase by following a strategy a_i at step i (the left-hand side) can be bounded by a combination of the algorithm cost (the second term of the right-hand side) and the cost of an adversary (potentially the set of all strategies a_i's), then the algorithm is competitive (again, depending on λ and μ).

Theorem 3 *Assume that all resource cost functions are (λ, μ)-smooth for some parameters $\lambda > 0$, $\mu < 1$. Then there exists a greedy $\frac{\lambda}{1-\mu}$-competitive algorithm for the general problem.*

Applications. We show the applicability of the theorem by deriving competitive algorithms for several problems in online setting, such as Minimum Power Survival Network Routing, Vector Scheduling, Energy-Efficient Scheduling, Prize Collecting Energy-Efficient Scheduling, Non-Convex Facility Location. Among such applications, the most representative ones are the Energy-Efficient Scheduling problem and the Non-Convex Facility Location problem. Recently, the author has implicitly applied this approach to give a simple asymptotically optimal algorithm [46] for a subspace approximation problem [22, 32].

In Online Energy-Efficient Scheduling, one has to process jobs on unrelated machines without migration with the objective of minimizing the total energy. No result has been known for this problem in multiple machine environments. Among others, a difficulty is the construction of formulation with bounded integrality gap. We notice that for this problem, Gupta et al. [31] gave a primal-dual competitive algorithm for a single machine. However, their approach cannot be used for unrelated machines due to the large integrality gap of their formulation. For this problem, we present competitive algorithms with *arbitrary* cost functions beyond the convexity property. Note that the convexity of cost functions is a crucial property employed in the analyses of previous work. If the cost functions have typical form $f(x) = x^\alpha$ then the competitive ratio of our algorithm is $O(\alpha^\alpha)$ and this is optimal up to a constant factor for all the problems above. Besides, in offline setting, this ratio is close to the currently best-known approximation ratio $B_\alpha \approx \big(\frac{\alpha}{\log \alpha}\big)^\alpha$ [41].

In Online Non-Convex Facility Location, clients arrive online and have to be assigned to facilities. The cost of a facility consists of a fixed opening cost and and a serving cost, which is an arbitrary monotone function depending on the number of clients assigned to the facility. The objective is to minimize the total client-facility connection cost and the facility cost. This problem is related to the capacitated network design and energy-efficient routing problems [5, 40]. In the latter, given a graph and a set of connectivity demands, the cost of each edge (node) is *uniform* and given by $c+f^\alpha$ where c is a fixed cost for every edge and f is the total of flow passed through the edge (node). (Here uniformity means the cost functions are the same for every edge.) The objective is to minimize the total cost while satisfying all connectivity demands. Antoniadis et al. [5], Krishnaswamy et al. [40] have provided online/offline algorithms with poly-logarithmic guarantees. It is an intriguing open questions (originally raised in [3]) to design a poly-logarithmic competitive algorithm for non-uniform cost functions. The Online Non-Convex Facility Location can be seen as a step towards this goal. In fact, the former can be considered as the connectivity problem on a simple depth-2-graph and the cost functions are now non-uniform.

This problem is beyond the scope of general problem but we show that the primal-dual framework can be used to derive competitive algorithm. Specifically, we present a $O(\log n + \frac{\lambda}{1-\mu})$-competitive algorithm if the cost function is (λ, μ)-smooth. The algorithm is inspired by primal-dual algorithm [27] in classic setting and our configuration LP-based approach. In particular, for the problem with non-uniform cost functions such as $c_i + w_i f_i^\alpha$ where c_i, w_i are parameters depending on facility i and f_i is the number of clients assigned to facility i, the algorithm yields a competitive ratio of $O(\log n + \alpha^\alpha)$.

Besides, the algorithm in Theorem 3 can be used in offline setting. Restricted to the class of polynomials with non-negative coefficients, our algorithm yields the competitive ratio of $O(\alpha^\alpha)$ while the best-known approximation ratio is $B_\alpha \approx \left(\frac{\alpha}{\log \alpha}\right)^\alpha$ [41]. Our greedy algorithm is light-weight and much simpler and faster than that in [41] which involves in solving an LP of exponential size and rounding fractional solutions. Hence, our algorithm can also be used to design approximation algorithms if one looks for the tradeoff between the simplicity and the performance guarantee.

1.4 Related work

In this section we summarize related work to our approach. The related work of each specific problem is formally given in the corresponding section.

In this paper, we systematically strengthen natural LPs by the construction of the configuration LPs presented in [41]. Makarychev and Sviridenko [41] propose a scheme that consists of solving the new LPs (with exponential number of variables) and rounding the fractional solutions to integer ones using decoupling inequalities. By this method, they derive approximation algorithms for several (offline) optimization problems which can formulated by linear constraints and objective function as a power of some constant α. Specifically, the approximation ratio is proved to be the Bell number B_α for several problems.

In our approach, a crucial element to characterize the performance of an algorithm is the smoothness property of functions. The smooth argument is introduced by Roughgarden [47] in the context of algorithmic game theory and it has successfully characterized the performance of equilibria (price of anarchy) in many classes of games such as congestion games, etc [47]. This notion inspires the definition of smoothness in our paper.

Primal-dual methods have been shown to be powerful tools in online computation. Buchbinder and Naor [15] presented a primal-dual method for linear programs with packing/covering constraints. Their method unifies several previous potential-function-based analyses and give a principled approach to design and analyze algorithms for problems with linear relaxations. Con-

vex objective functions have been extensively studied in online settings in recent years, in areas such as energy-efficient scheduling [2, 45, 23, 35, 7], paging [42], network routing [31], combinatorial auctions [13, 33], matching [24]. Recently, Azar et al. [8] gave an unified framework for covering/packing problems with convex objectives whose gradients are monotone. Consequently, improved algorithms have been derived for several problems. The above approaches relies crucially on the convexity of cost functions. Specifically, the construction of dual programs is based on convex conjugates and Fenchel duality for primal convex programs. Very recently, Nagarajan and Shen [44] have considered objective functions as the of sum of ℓ_k-norms. This class of functions do not fall into the framework developped in [8] since the gradients are not necessarily monotone. Nagarajan and Shen [44] proved that the algorithm presented in [8] yields a nearly tight $O\big(\log d + \log \frac{\max a_{ij}}{\min a_{ij}}\big)$-competitive ratio where a_{ij}'s are entries in the covering matrix. In the approaches, it is not clear how to design competitive algorithms for *non-convex* functions or even for other convex functions with non-monotone gradient. A distinguishing point of our approach is that it gives a framework to study non-convex cost functions.

2 Primal-Dual Framework for $0-1$ Covering Problems

Consider the following integer optimization problem. Let $\mathcal{E}$ be a set of n resources and let $f : \{0,1\}^n \to \mathbb{R}^+$ be a monotone cost function. Let $x_e \in \{0,1\}$ be a variable indicating whether resource e is selected. The problem is to minimize $f(\mathbf{x})$ subject to covering constraints $\sum_e a_{i,e}x_e \geq 1$ for every constraint i and $x_e \in \{0,1\}$ for every e. In the online setting, the constraints are revealed one-by-one and at any step, one needs to maintain a feasible integer solution $\mathbf{x}$.

2.1 Algorithm for Fractional Covering

Recall that a differentiable function $F : [0,1]^n \to \mathbb{R}^+$ is (λ,μ)-min-locally-smooth if for any set $S \subset \mathcal{E}$, and for any vectors $\mathbf{x}^e \in [0,1]^n$ where $e \in \mathcal{E}$, the following inequality holds:

$$\sum_{e\in S} \nabla_e F(\mathbf{x}^e) \leq \lambda F\big(\mathbf{1}_S\big) + \mu F\big(\mathbf{x}\big)$$

where $\mathbf{x} := \bigvee_{e\in S} \mathbf{x}^e$, meaning that $x_{e'} = \max_e\{x^e_{e'}\}$ for any coordinate e'.

Formulation. We say that $S \subset \mathcal{E}$ is a *configuration* if $\mathbf{1}_S$ corresponds to a feasible solution. Let x_e be a variable indicating whether the resource e is used. For configuration S, let z_S be a variable such that $z_S = 1$ if and only if $x_e = 1$ for every resource $e \in S$, and $x_e = 0$ for $e \notin S$. In other words, $z_S = 1$ iff $\mathbf{1}_S$ is the selected solution of the problem. For any subset $A \subset \mathcal{E}$, define $c_{i,A} = \max\{1 - \sum_{e'\in A} a_{i,e'}; 0\}$ and $a_{i,e,A} := \min\{a_{i,e}; c_{i,A}\}$. Denote $b_{i,e,A} = \frac{a_{i,e,A}}{c_{i,A}}$ where $c_{i,A} > 0$. We consider the following formulation and the dual of its relaxation.

$$\min \sum_S f(\mathbf{1}_S) z_S$$

$$\sum_{e \notin A} b_{i,e,A} \cdot x_e \geq 1 \qquad \forall i, A \subset \mathcal{E}$$

$$\sum_{S: e \in S} z_S = x_e \qquad \forall e$$

$$\sum_S z_S = 1$$

$$x_e, z_S \in \{0,1\} \qquad \forall e, S$$

$$\max \sum_{i,A} \alpha_{i,A} + \gamma$$

$$\sum_i \sum_{A: e \notin A} b_{i,e,A} \cdot \alpha_{i,A} \leq \beta_e \qquad \forall e$$

$$\gamma + \sum_{e \in S} \beta_e \leq f(\mathbf{1}_S) \qquad \forall S$$

$$\alpha_i \geq 0 \qquad \forall i$$

In the primal, the first constraints are knapsack-constraints of form $\sum_{e \notin A} a_{i,e,A} \cdot x_e \geq c_{i,A}$ corresponding to the given polytope. Note that it is sufficient to consider only constraints with $c_{i,A} > 0$. The second constraint ensures that if a resource e is chosen then the selected solution must contain e. The third constraint says that one solution (configuration) must be selected.

Algorithm. Assume that function $F(\cdot)$ is $\left(\lambda, \frac{\mu}{4\ln(1+2d^2)}\right)$–min-locally smooth. Let d be the maximal number of positive entries in a row, i.e., $d = \max_i |\{a_{ie} : a_{ie} > 0\}|$. Denote $\nabla_e F(\mathbf{x}) = \partial F(\mathbf{x})/\partial x_e$. Consider the following Algorithm 1 which follows the scheme in [8] with some more subtle steps due to the non-monotone behavior of the gradient. In the algorithm, the current dual variable α increases at constant rate (Step 6) and the update of dual variables β's in shown in Step 8. If the gradient $\nabla_e F(\mathbf{x})$ at coordinate e is monotone then β_e is set to be $\frac{1}{\lambda}\nabla_e F(\mathbf{x})$. However, in case $\nabla_e F(\mathbf{x})$ decreases, the value of β_e is kept unchanged. The primal update rule follows a multiplicative increase where the increasing rate of x_e is inversely proportional to β_e (Step 9). Finally, using the same idea in [8], some dual variables α will be decreased in order to maintain the feasibility of our dual solution.

Dual variables. Variables $\alpha_{i,A}$ and β_e are constructed in the algorithm. Let $\mathbf{x}$ be the current solution of the algorithm. Define $\gamma = -\frac{\mu}{4\lambda \cdot \ln(1+2d^2)} F(\mathbf{x})$. Note that due to the algorithm, $\beta_e \geq \frac{1}{\lambda} \cdot \nabla_e F(\mathbf{x})$.

The following lemma gives a lower bound on x-variables. Remark that the monotonicity of the gradient is crucial in the analysis of [8], in particular to prove the bounds on x-variables. However, by our approach the gradient monotonicity is not needed.

Lemma 1 *Let e be an arbitrary resource. At any moment during the execution of the algorithm where the k^{th} request has been released , it always holds that*

$$x_e \geq \frac{1}{\max b_{i,e,A} \cdot d} \left[\exp\left(\frac{\ln(1+2d^2)}{\beta_e} \cdot \sum_{A: e \notin A} \sum_i b_{i,e,A} \cdot \alpha_{i,A} \right) - 1 \right]$$

where denote $\max b_{i,e,A} := \max\{b_{i,e,A} > 0 | \forall A : e \notin A, \forall 1 \leq i \leq k : \alpha_{i,A} > 0\}$.

Proof Fix a resource e. We prove the lemma by induction. At the beginning of the instance, while no request has been released, both sides of the lemma are 0. Assume that the lemma holds until the arrival of k^{th} request. Consider a moment τ and let A^* be the current set of resources e' such that $x_{e'} = 1$. If at time τ, $x_e = 1$ then by the algorithm, the set A^* has been updated so

Algorithm 1 Algorithm for Covering Constraints.

Initially, set $A^* \leftarrow \emptyset$. Intuitively, A^* consists of all resources e such that $x_e = 1$.
All primal and dual variables are initially set to 0.
At every step, always maintain $z_S = \prod_{e \in S} x_e \prod_{e \notin S}(1 - x_e)$.
Upon the arrival of primal constraint $\sum_e a_{k,e} x_e \geq 1$ and the new corresponding dual variable α_k.
while $\sum_{e \notin A^*} b_{k,e,A^*} x_e < 1$ **simultaneously do** # `Increase primal, dual variables`
  Increase τ at rate 1 and increase α_{k,A^*} at rate $\frac{1}{\lambda \cdot \ln(1+2d^2)}$.
  for $e \notin A^*$ such that $b_{k,e,A^*} > 0$ **simultaneously do**
    if $\beta_e < \frac{1}{\lambda} \nabla_e F(\mathbf{x})$ **then** $\beta_e \leftarrow \frac{1}{\lambda} \nabla_e F(\mathbf{x})$
    Increase x_e according to the following function

$$\frac{\partial x_e}{\partial \tau} \leftarrow \frac{b_{k,e,A^*} \cdot x_e + 1/d}{\lambda \cdot \beta_e}$$

  end for
  if $x_e = 1$ **then** update $A^* \leftarrow A^* \cup \{e\}$.
  for $e \notin A^*$ such that $\sum_{i=1}^{k} \sum_{A: e \notin A} b_{i,e,A} \cdot \alpha_{i,A} \geq \beta_e$ **do** # `Decrease dual variables`
    Let $m_e^* \leftarrow \arg\max\{b_{i,e,A} | \forall A : e \notin A, \forall 1 \leq i \leq k : \alpha_{i,A} > 0\}$.
    Increase $\alpha_{m_e^*,A}$ continuously at rate $-\frac{b_{k,e,A^*}}{b_{m_e^*,e,A}} \cdot \frac{1}{\lambda \cdot \ln(1+2d^2)}$.
  end for
end while

that $e \in A^*$. The increasing rates of both sides in the lemma inequality are 0. In the remaining, assume that $x_e < 1$. Recall that by the algorithm, $\beta_e \geq \frac{1}{\lambda} \nabla_e F(\mathbf{x})$. We consider two cases where $\beta_e > \frac{1}{\lambda} \nabla_e F(\mathbf{x})$ and $\beta_e = \frac{1}{\lambda} \nabla_e F(\mathbf{x})$.

Case 1: $\beta_e > \frac{1}{\lambda} \nabla_e F(\mathbf{x})$**.** In this case, by the algorithm, the value of β_e remains unchanged at time τ. Hence, the derivative of the right hand side of the lemma inequality according to τ is

$$\sum_i \frac{\partial \alpha_{i,A^*}}{\partial \tau} \cdot \frac{b_{i,e,A^*}}{\max b_{i,e,A} \cdot d} \cdot \frac{\ln(1+2d^2)}{\beta_e} \cdot \exp\left(\frac{\ln(1+2d^2)}{\beta_e} \cdot \sum_{A: e \notin A} \sum_i b_{i,e,A} \alpha_{i,A} \right)$$
$$\leq \frac{b_{k,e,A^*} \cdot x_e + 1/d}{\lambda \cdot \beta_e} = \frac{\partial x_e}{\partial \tau}$$

In the inequality, we use the induction hypothesis; $\frac{\partial \alpha_{k,A^*}}{\partial \tau} > 0$ and $\frac{\partial \alpha_{i,A^*}}{\partial \tau} \leq 0$ for $i \neq k$ and $\frac{\partial \beta_e}{\partial \tau} = 0$; and the increasing rate of α_{k,A^*} according to the algorithm. So the rate in the left-hand side is always larger than that in the right-hand side. Moreover, at some steps in the algorithm, α-variables might be decreased while the x-variables are maintained monotone. Hence, the lemma inequality holds.

Case 2: $\beta_e = \frac{1}{\lambda} \nabla_e F(\mathbf{x})$**.** In this case, by the algorithm, $\frac{1}{\lambda} \nabla_e F(\mathbf{x})$ is locally non-decreasing at τ (since otherwise, β_e is not maintained to be equal to $\frac{1}{\lambda} \nabla_e F(\mathbf{x})$). Therefore, $\frac{\partial \beta_e}{\partial \tau} \geq 0$ and so $\partial\left(\frac{1}{\beta_e}\right)/\partial \tau \leq 0$. Hence, the derivative of the right hand side of the lemma inequality according to τ

is upper bounded by

$$\sum_i \frac{\partial \alpha_{i,A^*}}{\partial \tau} \cdot \frac{b_{i,e,A^*}}{\max b_{i,e,A} \cdot d} \cdot \frac{\ln(1+2d^2)}{\beta_e} \cdot \exp\left(\frac{\ln(1+2d^2)}{\beta_e} \cdot \sum_{A: e \notin A} \sum_i b_{i,e,A} \alpha_{i,A}\right)$$

which is bounded by $\frac{\partial x_e}{\partial \tau}$ by the same argument as the previous case. The lemma follows. □

Lemma 2 *The dual variables defined as above are feasible.*

Proof As long as a primal covering constraint is unsatisfied, the x-variables are always increased. Therefore, at the end of an iteration, the primal constraint is satisfied. Consider the first dual constraint. The algorithm always maintains that $\sum_i \sum_{A: e \notin A} b_{i,e,A} \alpha_{i,A} \leq \beta_e$ (strict inequality happens only if $x_e = 1$). Whenever this inequality is violated then by the algorithm, some α-variables are decreased in such a way that the increasing rate of $\sum_i \sum_{A: e \notin A} b_{i,e,A} \alpha_{i,A}$ is at most 0. Hence, by the definition of β-variables, the first dual constraint holds.

Consider the second dual constraint. Let $\mathbf{x}$ be the current solution of the algorithm. By the algorithm, for each fixed resource e, $\beta_e = \frac{1}{\lambda} \nabla_e F(\mathbf{y}^e)$ for some $\mathbf{y}^e$ where $y^e_{e'} \leq x_{e'}$ for every resources e'. (Since at some moment, the algorithm increases x_e without increasing β_e for some e.) Moreover, $\mathbf{y} := \bigvee_e \mathbf{y}^e \leq \mathbf{x}$ (meaning that $y_{e'} \leq x_{e'}$ for every e'). By definitions of dual variables, the second dual constraint (after rearranging terms) reads

$$\frac{1}{\lambda} \sum_{e \in S} \nabla_e F(\mathbf{y}^e) \leq F(\mathbf{1}_S) + \frac{\mu}{4\lambda \cdot \ln(1+2d^2)} F(\mathbf{x})$$

Besides, as F is monotone, $F(\mathbf{x}) \geq F(\mathbf{y})$. To prove the above inequality, it is sufficient to prove that

$$\frac{1}{\lambda} \sum_{e \in S} \nabla_e F(\mathbf{y}^e) \leq F(\mathbf{1}_S) + \frac{\mu}{4\lambda \cdot \ln(1+2d^2)} F(\mathbf{y})$$

This inequality is exactly the $\left(\lambda, \frac{\mu}{4\ln(1+2d^2)}\right)$-min-local smoothness of F. Hence, the lemma follows. □

We are now ready to prove the main theorem.

Theorem 1 *Let F be the multilinear extension of the objective cost f and d be the maximal row sparsity of the constraint matrix, i.e., $d = \max_i |\{a_{ie} : a_{ie} > 0\}|$. Assume that F is $\left(\lambda, \frac{\mu}{\ln(1+2d^2)}\right)$-min-locally-smooth for some parameters $\lambda > 0$ and $\mu < 1$. Then there exists a $O\left(\frac{\lambda}{1-\mu} \cdot \ln d\right)$-competitive algorithm for the fractional covering problem.*

Proof We will bound the increases of the cost and the dual objective at any time τ in the execution of Algorithm 1. Let A^* be the current set of resources e such that $x_e = 1$. The derivative of the objective with respect to τ is:

$$\sum_e \nabla_e F(\mathbf{x}) \cdot \frac{\partial x_e}{\partial \tau} = \sum_{\substack{e: b_{k,e,A^*} > 0 \\ x_e < 1}} \nabla_e F(\mathbf{x}) \cdot \frac{b_{k,e,A^*} \cdot x_e + 1/d}{\lambda \cdot \beta_e} \leq \sum_{e: b_{k,e,A^*} > 0} \left(b_{k,e,A^*} \cdot x_e + \frac{1}{d}\right) \leq 2 \quad (3)$$

The first inequality follows $\nabla_e F(\mathbf{x}) \leq \lambda \cdot \beta_e$. The second inequality is due to the definition of d and the fact that $\sum_{e \notin A^*} b_{k,e,A^*} \cdot x_e \leq 1$ always holds during the algorithm.

For a time τ, let $U(\tau)$ be the set of resources e such that $\sum_i \sum_{A:e\notin A} b_{i,e,A}\alpha_{i,A} = \beta_e$ and $b_{k,e,A^*} > 0$. Note that $|U(\tau)| \le d$ by definition of d. As long as $\sum_{e\notin A^*} b_{k,e,A^*}x_e < 1$, by Lemma 1, we have for every $e \in U(\tau)$,

$$\frac{1}{b_{k,e,A^*}} > x_e \ge \frac{1}{\max b_{i,e,A}\cdot d}\left[\exp\left(\ln(1+2d^2)\right) - 1\right]$$

Therefore, $\frac{b_{k,e,A^*}}{\max_i b_{i,e,A}} \le \frac{1}{2d}$.

We are now bounding the increase of the dual at time τ. The derivative of the dual with respect to τ is:

$$\begin{aligned}
\frac{\partial D}{\partial \tau} &= \sum_i \sum_A \frac{\partial \alpha_{i,A}}{\partial \tau} + \frac{\partial \gamma}{\partial \tau} = \sum_i c_{i,A^*}\cdot\frac{\partial \alpha_{i,A^*}}{\partial \tau} + \frac{\partial \gamma}{\partial \tau} \\
&= \frac{1}{\lambda\cdot\ln(1+2d^2)}\left(1 - \sum_{e\in U(\tau)} \frac{b_{k,e,A^*}}{b_{m_e^*,e,A}}\right) - \frac{\mu}{4\lambda\cdot\ln(1+2d^2)}\sum_e \nabla_e F(\mathbf{x})\cdot\frac{\partial x_e}{\partial \tau} \\
&\ge \frac{1}{\lambda\cdot\ln(1+2d^2)}\left(1 - \sum_{e\in U(\tau)}\frac{1}{2d}\right) - \frac{\mu}{2\lambda\cdot\ln(1+2d^2)} \\
&\ge \frac{1-\mu}{2\lambda\cdot\ln(1+2d^2)}
\end{aligned}$$

The third equality holds since α_{k,A^*} is increased and other α-variables in $U(\tau)$ are decreased. The first inequality follows the fact that $\frac{b_{k,e,A^*}}{\max_i b_{i,e,A^*}} \le \frac{1}{2d}$ and Inequality (3). The last inequality holds since $|U(\tau)| \le d$. Hence, the competitive ratio is $O\big(\frac{\lambda}{1-\mu}\cdot\ln d\big)$. □

2.2 Applications

In this section, we consider the applications of Theorem 1 for classes of cost functions which have been extensively studied in optimization such as polynomials with non-negative coefficients, ℓ_k-norms and submodular functions. We are interested in deriving fractional solutions[1] with performance guarantee. We show that Algorithm 1 with multilinear extension yields competitive fractional solutions for the classes of functions mentioned above and also for some natural classes of non-convex functions.

We first take a closer look into the definition of min-local smoothness. Let F be a multilinear extension of a set function f. By definition of multilinear extension, $F(\mathbf{x}) = \mathbb{E}\big[f(\mathbf{1}_T)\big]$ where T is a random set such that a resource e appears in T with probability x_e. Moreover, since F is linear in x_i, we have

$$\begin{aligned}
\frac{\partial F}{\partial x_e}(\mathbf{x}) &= F(x_1,\ldots,x_{e-1},1,x_{e+1},\ldots,x_n) - F(x_1,\ldots,x_{e-1},0,x_{e+1},\ldots,x_n) \\
&= \mathbb{E}\Big[f\big(\mathbf{1}_{R\cup\{e\}}\big) - f\big(\mathbf{1}_R\big)\Big]
\end{aligned}$$

[1]Rounding schemes (in order to obtain integral solution) for concrete problems are problem-specific and are not considered in this section. Several rounding techniques have been shown for different problems, for example in [8] for polynomials with non-negative coefficients, or using online contention resolution schemes for submodular functions [26].

where R is a random subset of resources $N \setminus \{e\}$ such that e' is included with probability $x_{e'}$. Therefore, in order to prove that F is (λ, μ)-min-locally-smooth, it is equivalent to show that, for any set $S \subset \mathcal{E}$ and for any vectors $\mathbf{x}^e \in [0,1]^n$ for $e \in \mathcal{E}$,

$$\sum_{e \in S} \mathbb{E}\Big[f\big(\mathbf{1}_{R^e \cup \{e\}}\big) - f\big(\mathbf{1}_{R^e}\big)\Big] \leq \lambda f\big(\mathbf{1}_S\big) + \mu \mathbb{E}\Big[f\big(\mathbf{1}_R\big)\Big] \tag{4}$$

where R^e is a random subset of resources $N \setminus \{e\}$ such that e' is included with probability $x^e_{e'}$ and R is a random subset of resources $N \setminus \{e\}$ such that e' is included with probability $\max_{e \in S} x^e_{e'}$. Note that if ∇F is monotone (in all coordinates) then the following inequality implies Inequality (4).

$$\sum_{e \in S} \mathbb{E}\Big[f\big(\mathbf{1}_{R \cup \{e\}}\big) - f\big(\mathbf{1}_R\big)\Big] \leq \lambda f\big(\mathbf{1}_S\big) + \mu \mathbb{E}\Big[f\big(\mathbf{1}_R\big)\Big] \tag{5}$$

Polynomials with non-negative coefficients. Let $g : \mathbb{R} \to \mathbb{R}$ be a polynomial with non-negative coefficients and the cost function $f : \{0,1\}^n \to \mathbb{R}^+$ defined as $f(\mathbf{1}_S) = g\big(\sum_{e \in S} a_e\big)$ where $a_e \geq 0$ for every e. The following proposition shows that our algorithm yields the same competitive ratio as the one derived in [8] for this class of cost functions. This bound indeed is *tight* [8] (up to a constant factor). Note that, Azar et al. [8] gave randomized algorithms for several problems by rounding their fractional solutions. As one can approach a multilinear extension of any function up to a high precision [49], applying the same rounding schemes in [8] for the corresponding problems based on our fractional solutions, one can obtain randomized algorithms with similar bounds as in [8].

Proposition 1 ([8]) *For any convex polynomial function g of degree k, there exists an $O\big((k \ln d)^k\big)$-competitive algorithm for the fractional covering problem.*

Proof We prove that Algorithm 1 is $O\big((k \ln d)^k\big)$-competitive for this class of cost functions. By Theorem 1, it is sufficient to verify that F is $((k \ln k)^{k-1}, \frac{k-1}{k \ln d})$-min-locally smooth. Note that ∇F is monotone. We indeed prove a stronger inequality than (5), that is for any set $R \subset \mathcal{E}$,

$$\sum_{e \in S} \Big[f\big(\mathbf{1}_{R \cup \{e\}}\big) - f\big(\mathbf{1}_R\big)\Big] \leq O\big((k \ln k)^{k-1}\big) \cdot f(\mathbf{1}_S) + \frac{k-1}{k \ln k} \cdot f\big(\mathbf{1}_R\big)$$

or equivalently, for any set $R \subset \mathcal{E}$,

$$\sum_{e \in S} \Big[g\Big(a_e + \sum_{e' \in R} a_{e'}\Big) - g\Big(\sum_{e' \in R} a_{e'}\Big)\Big] \leq O\big((k \ln k)^{k-1}\big) \cdot g\Big(\sum_{e \in S} a_e\Big) + \frac{k-1}{k \ln k} \cdot g\Big(\sum_{e' \in R} a_{e'}\Big)$$

This inequality holds by Lemma 6 (in the appendix). Hence, the proposition follows. □

Beyond convex functions. Consider the following natural cost functions which represent more practical costs when serving clients as mentioned in the introduction (the cost initially increases fast then becomes more stable before growing quickly again). Let $g : \mathbb{R} \to \mathbb{R}$ be a non-convex function defined as $g(y) = y^k$ if $y \leq M_1$ or $y \geq M_2$ and $g(y) = g(M_1)$ if $M_1 \leq y \leq M_2$ where $M_1 < M_2$ are some constants. The cost function $f : \{0,1\}^n \to \mathbb{R}^+$ defined as $f(\mathbf{1}_S) = g\big(\sum_{e \in S} a_e\big)$ where $a_e \geq 0$ for every e. In fact, the corresponding multilinear extension F is $((k \ln k)^{k-1}, \frac{k-1}{k \ln d})$-min-locally smooth. Again, it sufficient to verify Inequality (5) and the proof is similar to the one in Proposition 1 (or more specifically, Lemma 6 in the appendix) and note that the derivative of g for $M_1 < y < M_2$ equals 0.

Proposition 2 *The algorithm is $O\big((k \ln d)^k\big)$-competitive for minimizing the non-convex objective function defined above under covering constraints.*

Submodular functions. Consider the class of submodular functions f, that is $f(\mathbf{1}_{S\cup\{e\}}) - f(\mathbf{1}_S) \geq f(\mathbf{1}_{T\cup\{e\}}) - f(\mathbf{1}_T)$ for every e and $S \subset T$ and $f(\mathbf{1}_\emptyset) = 0$. Submodular optimization has been extensively studying in optimization and machine learning. In the context of online algorithms, Buchbinder et al. [17] have considered submodular optimization with preemption, where one can reject previously accepted elements, and have given constant competitive algorithms for unconstrained and knapsack-constraint problems. To the best of our knowledge, the problem of online submodular minimization under covering constraints have not been considered.

An important concept in studying submodular functions is the *curvature*. Given a submodular function f, the *total curvature* κ_f [21] of f is defined as

$$\kappa_f = 1 - \min_e \frac{f(\mathbf{1}_\mathcal{E}) - f(\mathbf{1}_{\mathcal{E}\setminus\{e\}})}{f(\mathbf{1}_{\{e\}})}$$

Intuitively, the total curvature mesures how far away f is from being *modular*. The concept of curvature has been used to determines both upper and lower bounds on the approximation ratios for many submodular and learning problems [21, 29, 9, 50, 37, 48].

In the following, we present a competitive algorithm for minimizing a monotone submodular function under covering constraints where the competitive ratio is characterized by the curvature of the function (and also the sparsity d of the covering matrix). We first look at an useful property of the total curvature.

Lemma 3 *For any set S, it always holds that*

$$f(\mathbf{1}_S) \geq (1 - \kappa_f) \sum_{e\in S} f(\mathbf{1}_{\{e\}}).$$

Proof Let $S = \{e_1, \ldots, e_m\}$ be an arbitrary subset of $\mathcal{E}$. Let $S_i = \{e_1, \ldots, e_i\}$ for $1 \leq i \leq m$ and $S_0 = \emptyset$. We have

$$\begin{aligned} f(\mathbf{1}_S) \geq f(\mathbf{1}_\mathcal{E}) - f(\mathbf{1}_{\mathcal{E}\setminus S}) &= \sum_{i=0}^{m-1} f(\mathbf{1}_{\mathcal{E}\setminus S_i}) - f(\mathbf{1}_{\mathcal{E}\setminus S_{i+1}}) \geq \sum_{i=1}^{m} f(\mathbf{1}_\mathcal{E}) - f(\mathbf{1}_{\mathcal{E}\setminus\{e_i\}}) \\ &\geq (1 - \kappa_f) \sum_{i=1}^{m} f(\mathbf{1}_{e_i}) \end{aligned}$$

where the first two inequalities are due to submodularity of f and the last inequality follows by the definition of the curvature. □

Proposition 3 *The algorithm is $O\big(\frac{\log d}{1-\kappa_f}\big)$-competitive for minimizing monotone submodular function under covering constraints.*

Proof It is sufficient to verify that F is $\big(\frac{1}{1-\kappa_f}, 0\big)$-min-locally smooth. Indeed, the $\big(\frac{1}{1-\kappa_f}, 0\big)$-min-local smoothness holds due to the submodularity and Lemma 3: for any subsets R^e,

$$\sum_{e\in S} \big[f\big(\mathbf{1}_{R^e\cup\{e\}}\big) - f\big(\mathbf{1}_{R^e}\big)\big] \leq \sum_{e\in S} \big[f\big(\mathbf{1}_{\{e\}}\big)\big] \leq \frac{1}{1-\kappa_f} \cdot f(\mathbf{1}_S)$$

Therefore, the proposition follows. □

3 Primal-Dual Framework for Packing Problems

Consider the following integer optimization problem. Let $\mathcal{E}$ be a set of n resources and let $f : \{0,1\}^n \to \mathbb{R}^+$ be an *abitrary* cost function. Let $x_e \in \{0,1\}$ be a variable indicating whether resource e is selected. The packings constraints $\sum_e b_{i,e} x_e \leq 1$ for every i are given in advance and resources e are revealed online one-by-one. At any time, one needs to maintain a feasible integer solution $\mathbf{x}$. The goal is to design an algorithm that maximizes $f(\mathbf{x})$ subject to the online packing constraints and $x_e \in \{0,1\}$ for every e.

3.1 Algorithm for Fractional Packing

Recall that a differentiable function $F : [0,1]^n \to \mathbb{R}^+$ is (λ, μ)-max-locally-smooth if for any set $S \subset \mathcal{E}$, and for any vectors $\mathbf{x}^e \in [0,1]^n$, the following inequality holds:

$$\sum_{e \in S} \nabla_e F(\mathbf{x}^e) \geq \lambda F\big(\mathbf{1}_S\big) - \mu F\big(\mathbf{x}\big).$$

where $\mathbf{x} := \bigvee_{e \in S} \mathbf{x}^e$, meaning that $x_{e'} = \max_e \{x^e_{e'}\}$ for any coordinate e'.

Formulation. We say that $S \subset \mathcal{E}$ is a *configuration* if $\mathbf{1}_S$ corresponds to a feasible solution. Let x_e be a variable indicating whether the resource e is used. For configuration S, let z_S be a variable such that $z_S = 1$ if and only if $x_e = 1$ for every resource $e \in S$, and $x_e = 0$ for $e \notin S$. In other words, $z_S = 1$ iff $\mathbf{1}_S$ is the selected solution of the problem. We consider the following formulation and the dual of its relaxation.

$$\begin{aligned}
&\max \sum_S f(\mathbf{1}_S) z_S && && \min \sum_i \alpha_i + \gamma && \\
&\sum_e b_{i,e} \cdot x_e \leq 1 && \forall i && \sum_i b_{i,e} \cdot \alpha_i \geq \beta_e && \forall e \\
&\sum_{S : e \in S} z_S = x_e && \forall e && \gamma + \sum_{e \in S} \beta_e \geq f(\mathbf{1}_S) && \forall S \\
&\sum_S z_S = 1 && && \alpha_i \geq 0 && \forall i \\
&x_e, z_S \in \{0,1\} && \forall e, S && &&
\end{aligned}$$

In the primal, the first constraints represent the given polytope. Note that the box constraint $x_e \leq 1$ is included among these constraints. The second constraint ensures that if a resource e is chosen then the selected solution must contain e. The third constraint says that one solution (configuration) must be selected.

Algorithm. Assume that function $F(\cdot)$ is (λ, μ)-max-locally smooth. Let d be the maximal number of positive entries in a row, i.e., $d = \max_i |\{b_{ie} : b_{ie} > 0\}|$. Define $\rho = \max_i \max_{e,e' : b_{ie'} > 0} b_{ie}/b_{ie'}$. Denote $\nabla_e F(\mathbf{x}) = \partial F(\mathbf{x})/\partial x_e$. In the algorithm, at the arrival of a new resource e, while $\nabla_e F(\mathbf{x}) > 0$ (i.e., one can still improve the cost by increasing x_e) and $\sum_i b_{i,e} \alpha_e \leq \frac{1}{\lambda} \nabla_e F(\mathbf{x})$, the primal variable x_e and dual variables α_i's are increased by appropriate rates. We will argue in the analysis that the primal/dual solutions returned by the algorithm are feasible. Recall that by definition of multilinear extension, $\nabla_e F(\mathbf{x}) = \mathbb{E}\big[f\big(\mathbf{1}_{R \cup \{e\}}\big) - f\big(\mathbf{1}_R\big)\big]$ where R is a random subset of resources

$N \setminus \{e\}$ such that e' is included with probability $x_{e'}$. Therefore, during the iteration of the while loop with respect to resource e, only x_e is modified and $x_{e'}$ remain fixed for $e' \neq e$, so $\nabla_e F(\mathbf{x})$ is constant during the iteration.

Algorithm 2 Algorithm for Packing Constraints.

1: All primal and dual variables are initially set to 0.
2: At every step, always maintain $z_S = \prod_{e \in S} x_e \prod_{e \notin S}(1 - x_e)$.
3: Upon the arrival of new resource e.
4: **while** $\sum_i b_{i,e}\alpha_i \leq \frac{1}{\lambda}\nabla_e F(\mathbf{x})$ and $\nabla_e F(\mathbf{x}) > 0$ **do**
5: Increase τ at rate 1 and increase x_e at rate $\frac{1}{\nabla_e F(\mathbf{x}) \cdot \ln(1+d\rho)}$.
6: **for** i such that $b_{i,e} > 0$ **simultaneously do**
7: Increase α_i according to the following function $\frac{\partial \alpha_i}{\partial \tau} \leftarrow \frac{b_{i,e} \cdot \alpha_i}{\nabla_e F(\mathbf{x})} + \frac{1}{d\lambda}$
8: **end for**
9: **end while**

Dual variables. Variables α_i's are constructed in the algorithm. Let $\mathbf{x}$ be the current solution of the algorithm and let $\mathbf{x}^e$ be the solution after the while loop with respect to resource e. Define $\gamma = \frac{\mu}{\lambda} F(\mathbf{x})$ where $\mathbf{x}$ and $\beta_e = \frac{1}{\lambda} \cdot \nabla_e F(\mathbf{x}^e)$. Note that by the observation above, during the while loop with respect to resource e, $\beta_e = \frac{1}{\lambda} \cdot \nabla_e F(\mathbf{x})$.

The following lemma gives a lower bound on α-variables.

Lemma 4 *At any moment during the execution of the algorithm, it always holds that for every i*

$$\alpha_i \geq \frac{\nabla_e F(\mathbf{x})}{\max_{e'} b_{i,e'} \cdot d\lambda} \left[\exp\left(\ln\big(1 + d\rho\big) \cdot \sum_{e'} b_{i,e'} \cdot x_{e'} \right) - 1 \right]$$

Proof We prove the lemma by induction. At the beginning of the instance, while no resource has been released, both sides of the lemma are 0. Assume that the lemma holds until the arrival of a resource e. Consider a moment τ during the loop corresponding to resource e. Note that as F is a linear extension, $\partial \nabla_e F(\mathbf{x}) / \partial \tau = 0$. The derivative of the right hand side of the lemma inequality according to τ is

$$\begin{aligned}
&\frac{\nabla_e F(\mathbf{x})}{\max_{e'} b_{i,e'} \cdot d\lambda} \cdot \ln\big(1 + d\rho\big) \cdot b_{i,e} \cdot \frac{\partial x_e}{\partial \tau} \cdot \exp\left(\ln\big(1 + d\rho\big) \cdot \sum_{e'} b_{i,e'} \cdot x_{e'} \right) \\
&\leq \frac{\nabla_e F(\mathbf{x})}{\max_{e'} b_{i,e'} \cdot d\lambda} \cdot \ln\big(1 + d\rho\big) \cdot b_{i,e} \cdot \frac{1}{\nabla_e F(\mathbf{x}) \cdot \ln(1 + d\rho)} \cdot \left(\frac{\max_{e'} b_{i,e'} \cdot d\lambda \cdot \alpha_i}{\nabla_e F(\mathbf{x})} + 1 \right) \\
&\leq \frac{b_{i,e} \cdot \alpha_i}{\nabla_e F(\mathbf{x})} + \frac{1}{d\lambda} = \frac{\partial \alpha_i}{\partial \tau}
\end{aligned}$$

where in the first inequality, we use the induction hypothesis. So the rate in the left-hand side is always larger than that in the right-hand side. Hence, the lemma follows. □

Lemma 5 *The dual variables defined as above are feasible.*

Proof We first prove the primal feasibility. During the execution of the algorithm, if $\sum_i b_{i,e'} x_{e'} > 1$ for some constraint i then by Lemma 1,

$$\alpha_i > \frac{\nabla_e F(\mathbf{x})}{\max_{e'} b_{i,e'} \cdot d\lambda} \left[\exp\left(\ln\big(1 + d\rho\big) \right) - 1 \right] = \frac{\rho \cdot \nabla_e F(\mathbf{x})}{\lambda \max_{e'} b_{i,e'}} \geq \frac{\nabla_e F(\mathbf{x})}{\lambda b_{i,e}}$$

Therefore, $\sum_i b_{i,e}\alpha_i > \frac{1}{\lambda}\nabla_e F(\mathbf{x})$ and so the algorithm would have stopped increasing x_e at some earlier point. Hence, the constraint $\sum_i b_{i,e'}x_{e'} \leq 1$ is alway maintained.

The first dual constraint is satisfied by the algorithm. The second dual constraint reads

$$\frac{1}{\lambda}\sum_{e\in S}\nabla_e F(\mathbf{x}^e) + \frac{\mu}{\lambda}F(\mathbf{x}) \geq F(\mathbf{1}_S)$$

which is, by arranging terms, exactly the (λ,μ)-max-local smoothness of F. Hence, the lemma follows. □

We are now ready to prove the main theorem.

Theorem 2 *Let F be the multilinear extension of the objective cost f. Denote the row sparsity $d := \max_i |\{b_{ie} : b_{ie} > 0\}|$ and $\rho := \max_i \max_{e,e':b_{ie'}>0} b_{ie}/b_{ie'}$. Assume that F is (λ,μ)-max-locally-smooth for some parameters $\lambda > 0$ and $\mu < 1$. Then there exists a $O\big(\frac{2\ln(1+d\rho)+\mu}{\lambda}\big)$-competitive algorithm for the fractional packing problem.*

Proof We will bound the increases of the cost and the dual objective at any time τ in the execution of Algorithm 2. The derivative of the primal with respect to τ is:

$$\nabla_e F(\mathbf{x}) \cdot \frac{\partial x_e}{\partial \tau} = \nabla_e F(\mathbf{x}) \cdot \frac{1}{\nabla_e F(\mathbf{x}) \cdot \ln(1+d\rho)} = \frac{1}{\ln(1+d\rho)}$$

We are now bounding the increase of the dual at time τ. The derivative of the dual with respect to τ is:

$$\begin{aligned}
\frac{\partial D}{\partial \tau} &= \sum_i \frac{\partial \alpha_i}{\partial \tau} + \frac{\partial \gamma}{\partial \tau} = \sum_{i:b_{i,e}>0}\left(\frac{b_{i,e}\cdot\alpha_i}{\nabla_e F(\mathbf{x})} + \frac{1}{d\lambda}\right) + \frac{\mu}{\lambda}\frac{\partial F(\mathbf{x})}{\partial \tau} \\
&= \sum_{i:b_{i,e}>0}\frac{b_{i,e}\cdot\alpha_i}{\nabla_e F(\mathbf{x})} + \sum_{i:b_{i,e}>0}\frac{1}{d\lambda} + \frac{\mu}{\lambda}\cdot\frac{1}{\ln(1+d\rho)} \\
&\leq \frac{2}{\lambda} + \frac{\mu}{\lambda\cdot\ln(1+d\rho)} = \frac{2\ln(1+d\rho)+\mu}{\lambda\cdot\ln(1+d\rho)}
\end{aligned}$$

where the inequality holds since during the algorithm $\sum_i b_{i,e}\cdot\alpha_i \leq \beta_e = \frac{1}{\lambda}\nabla_e F(\mathbf{x})$. Hence, the competitive ratio is $O\big(\frac{2\ln(1+d\rho)+\mu}{\lambda}\big)$. □

Note that the competitive ratio is the same up to a constant factor as the performance guarantee for maximizing a linear function under packing constraints. Specifically, if function f is linear then the smooth parameters $\lambda = \mu = 1$.

3.2 Applications to online submodular maximization

Consider a online submodular maximization subject to packing constraints. We incorporate additional constraints $x_e \leq 2/3$ (instead of box constraint $x_e \leq 1$) for every e. The advantage of these stronger constraints, as shown below, is that we can bound the smooth parameters while loosing only a constant factor in the competitive ratio. We are now determining smooth parameters of the multilinear extension F.

Lemma 6 *Let f be an arbitrary submodular function. Then, the multilinear extension F is (1,1)-smooth if f is monotone and is $(1/3,1)$-smooth if f is non-monotone.*

Proof For arbitrary submodular function f, it holds that [25, Lemma III.5] for any vector $\mathbf{y}$ and any subset S, $F\big(\mathbf{1}_S \vee \mathbf{y}\big) \geq (1 - \max_e y_e)F(\mathbf{1}_S)$. Moreover, if f is monotone, $F\big(\mathbf{1}_S \vee \mathbf{y}\big) \geq F(\mathbf{1}_S)$.

Consider arbitrary vectors $\mathbf{x}^e$ and let $\mathbf{x} = \bigvee_e \mathbf{x}^e$. As F is the linear extension of a submodular function, $\nabla_e F(\mathbf{x}^e) \geq \nabla_e F(\mathbf{x}) = \mathbb{E}\big[f\big(\mathbf{1}_{R\cup\{e\}}\big) - f\big(\mathbf{1}_R\big)\big]$ where R is a random subset of resources $N \setminus \{e\}$ such that e' is included with probability $x_{e'}$. Therefore, for any subset S,

$$\begin{aligned}
F(\mathbf{x}) + \sum_{e\in S} \nabla_e F(\mathbf{x}^e) &\geq F(\mathbf{x}) + \sum_{e\in S} \mathbb{E}\big[f\big(\mathbf{1}_{R\cup\{e\}}\big) - f\big(\mathbf{1}_R\big)\big] \\
&= \mathbb{E}\left[f(\mathbf{1}_R) + \sum_{e\in S}\big[f\big(\mathbf{1}_{R\cup\{e\}}\big) - f\big(\mathbf{1}_R\big)\big]\right] \geq \mathbb{E}\big[f(\mathbf{1}_{R\cup S})\big] = F\big(\mathbf{1}_S \vee \mathbf{x}\big) \\
&\geq \begin{cases} F(\mathbf{1}_S) & \text{if } f \text{ monotone,} \\ (1 - \max_e x_e)F(\mathbf{1}_S) & \text{otherwise} \end{cases} \geq \begin{cases} F(\mathbf{1}_S) & \text{if } f \text{ monotone,} \\ 1/3 \cdot F(\mathbf{1}_S) & \text{otherwise} \end{cases}
\end{aligned}$$

where the second inequality is due to the submodularity of f, and the last inequality holds since $x_e \leq 2/3$ for every e. The lemma follows. □

The previous lemma and Theorem 2 lead to the following result.

Proposition 4 *Algorithm 2 yields a $O\big(\ln(1 + d\rho)\big)$-competitive fractional solution for maximizing (arbitrary) submodular functions under packing constraints.*

One can derive online randomized algorithms for specific problems by rounding the fractional solutions. For example, using the online contention resolution rounding schemes [26], one can obtain randomized algorithms for several specific constraint polytopes, for example, knapsack polytopes, matching polytopes and matroid polytopes.

4 Primal-Dual Algorithm for the General Problem

We first present a primal-dual algorithm for the general problem described in Section 1.3. Despite the simplicity of the algorithm, it has indeed various direct and indirect applications (shown in subsequent sections) with optimal bounds in typical settings.

4.1 A Greedy Algorithm

Recall that in the problem, there is a set of resources $\mathcal{E}$ and requests arrive online. At the arrival of request i, a set of feasible strategies (actions) $\mathcal{S}_i$ to satisfy request i is revealed. Each strategy $s_{ij} \in \mathcal{S}_i$ consists of a subset of resources in $\mathcal{E}$. Each resource e is associated to a non-negative non-decreasing *arbitrary* cost function f_e and the cost induced by resource e depending on the set of requests using e. The cost of a solution is the total cost of resources, i.e., $\sum_e f_e(A_e)$ where A_e is the set of requests using resource e. The goal is design an algorithm that upon the arrival of each request, selects a feasible strategy while maintaining the cost of the overall solution as small as possible.

Formulation. We consider the formulation for the resource cost minimization problem following the configuration LP construction in [41]. We say that A is a *configuration* associated to resource e if A is a subset of requests using e. Let x_{ij} be a variable indicating whether request i selects strategy (action) $s_{ij} \in \mathcal{S}_i$. For configuration A and resource e, let z_{eA} be a variable such that $z_{eA} = 1$ if and only if for every request $i \in A$, $x_{ij} = 1$ for some strategy $s_{ij} \in \mathcal{S}_i$ such that $e \in s_{ij}$.

In other words, $z_{eA} = 1$ iff the set of requests using e is exactly A. We consider the following formulation and the dual of its relaxation.

$$\begin{aligned}
&\min \sum_{e,A} f_e(A) z_{e,A} && && \max \sum_i \alpha_i + \sum_e \gamma_e && \\
&\sum_{j:s_{ij}\in\mathcal{S}_i} x_{ij} = 1 && \forall i && \alpha_i \le \sum_{e:e\in s_{ij}} \beta_{ie} && \forall i,j \\
&\sum_{A:i\in A} z_{eA} = \sum_{j:e\in s_{ij}} x_{ij} && \forall i,e && \gamma_e + \sum_{i\in A} \beta_{ie} \le f_e(A) && \forall e,A \\
&\sum_A z_{eA} = 1 && \forall e && && \\
&x_{ij}, z_{eA} \in \{0,1\} && \forall i,j,e,A && &&
\end{aligned}$$

In the primal, the first constraint guarantees that request i selects some strategy $s_{ij} \in \mathcal{S}_i$. The second constraint ensures that if request i selects strategy s_{ij} that contains resource e then in the solution, the set of requests using e must contain i. The third constraint says that in the solution, there is always a configuration associated to resource e.

Algorithm. We first interpret intuitively the dual variables, dual constraints and derive useful observations for a competitive algorithm. Variable α_i represents the increase of the total cost due to the arrival of request i. Variable $\beta_{i,e}$ stands for the marginal cost on resource e if request i uses e. By this interpretation, the first dual constraint clearly indicates the behaviour of an algorithm. That is, if a new request i is released, select a strategy $s_{ij} \in \mathcal{S}_i$ that minimizes the marginal increase of the total cost. Therefore, we deduce the following greedy algorithm.

Let A_e^* be the set of current requests using resource e. Initially, $A_e^* \leftarrow \emptyset$ for every e. At the arrival of request i, select strategy s_{ij}^* that is an optimal solution of

$$\min \sum_{e\in s_{ij}} \left[f_e(A_e^* \cup i) - f_e(A_e^*) \right] \qquad \text{s.t.} \qquad s_{ij} \in \mathcal{S}_i. \tag{6}$$

Although computational complexity is not a main issue for online problems, we notice that in many applications, the optimal solution for this mathematical program can be efficiently computed (for example when f_e's are convex and $\mathcal{S}_i$ can be represented succinctly in form of a polynomial-size polytope).

Dual variables. Assume that all resource cost f_e are (λ, μ)-smooth for some fixed parameters $\lambda > 0$ and $\mu < 1$. We are now constructing a dual feasible solution. Define α_i as $1/\lambda$ times the optimal value of the mathematical program (6). Informally, α_i is proportional the increase of the total cost due to the arrival of request i. For each resource e and request i, define

$$\beta_{i,e} := \frac{1}{\lambda} \left[f_e(A^*_{e,\prec i} \cup i) - f_e(A^*_{e,\prec i}) \right]$$

where $A^*_{e,\prec i}$ is the set of requests using resource e (due to the algorithm) prior to the arrival of i. In other words, β_{ij} equals $1/\lambda$ times the marginal cost of resource e if i uses e. Finally, for every resource e define dual variable $\gamma_e := -\frac{\mu}{\lambda} f_e(A_e^*)$ where A_e^* is the set of all requests using e (at the end of the instance).

Lemma 7 *The dual variables defined as above are feasible.*

Proof The first dual constraint follows immediately the definitions of $\alpha_i, \beta_{i,e}$ and the decision of the algorithm. Specifically, the right-hand side of the constraint represents $1/\lambda$ times the increase cost if the request selects a strategy s_{ij}. This is larger than $1/\lambda$ times the minimum increase cost optimized over all strategies in $\mathcal{S}_i$, which is α_i.

We now show that the second constraint holds. Fix a resource e and a configuration A. The corresponding constraint reads

$$\begin{aligned} & -\frac{\mu}{\lambda} f_e(A^*_e) + \frac{1}{\lambda} \sum_{i \in A} \Big[f_e(A^*_{e,\prec i} \cup i) - f_e(A^*_{e,\prec i}) \Big] \leq f_e(A) \\ \Leftrightarrow \quad & \sum_{i \in A} \Big[f_e(A^*_{e,\prec i} \cup i) - f_e(A^*_{e,\prec i}) \Big] \leq \lambda f_e(A) + \mu f_e(A^*_e) \end{aligned}$$

This inequality is due to the definition of (λ, μ)-smoothness for resource e. Hence, the second dual constraint follows. □

Theorem 3 *Assume that all resource cost functions are (λ, μ)-smooth for some parameters $\lambda > 0$, $\mu < 1$. Then there exists a greedy $\frac{\lambda}{1-\mu}$-competitive algorithm for the general problem.*

Proof By the definitions of dual variables, the dual objective is

$$\sum_i \alpha_i + \sum_e \gamma_e = \sum_e \frac{1}{\lambda} f_e(A^*_e) - \sum_e \frac{\mu}{\lambda} f_e(A^*_e) = \frac{1-\mu}{\lambda} \sum_e f_e(A^*_e)$$

Besides, the cost of the solution due to the algorithm is $\sum_e f_e(A^*_e)$. Hence, the competitive ratio is at most $\lambda/(1-\mu)$. □

Applications. Theorem 3 yields simple algorithm with *optimal* competitive ratios for several problems as shown in the following sections. Among others, we give optimal algorithms for energy efficient scheduling problems (in unrelated machine environment) and the facility location with client-dependent cost problem. Prior to our work, no competitive algorithm has been known for the problems. The proofs are now reduced to computing smooth parameters λ, μ that subsequently imply the competitive ratios. We mainly use the smooth inequalities in Lemma 6, developed in [20], to derive the explicit competitive bounds in case of non-negative polynomial cost functions.

4.2 Minimum Power Survival Network Routing

Problem. In the problem, we are given a graph $G(V, E)$ and requests arrive online. The demand of a request i is specified by a source $s_i \in V$, a sink $t_i \in V$, the load vector $p_{i,e}$ for every edge (link) $e \in E$ and an integer number k_i. At the arrival of request i, one needs to choose k_i edge-disjoint paths connecting s_i to t_i. Request i increases the load $p_{i,e}$ for each edge e used to satisfy its demand. The load ℓ_e of an edge e is defined as the total load incurred by the requests using e. The *power* cost of edge e with load ℓ_e is $f_e(\ell_e)$. The objective is to minimize the total power $\sum_e f_e(\ell_e)$. Typically $f_e(\ell_e) = c_e \ell_e^{\alpha_e}$ where c_e and α_e are parameters depending on e.

This problems generalizes the Minimum Power Routing problem — a variant in which $k_i = 1$ and $p_{i,e} = 1\ \forall i, e$ — and the Load Balancing problem — a variant in which $k_i = 1$, all the sources (sinks) are the same $s_i = s_{i'}\ \forall i, i'$ ($t_i = t_{i'}\ \forall i, i'$) and every $s_i - t_i$ path has length 2.

For the Minimum Power Routing in offline setting, Andrews et al. [4] gave a polynomial-time poly-log-approximation algorithm. The result has been improved by Makarychev and Sviridenko [41] who gave an B_α-approximation algorithm. In online setting, Gupta et al. [31] presented an α^α-competitive online algorithm. For the Load Balancing problem, the currently best-known approximation is B_α due to [41] via their rounding technique based on decoupling inequality. In online setting, it has been shown that the optimal competitive ratio for the Load Balancing problem is $\Theta(\alpha^\alpha)$ [18].

Contribution. In the problem, the set of strategy $\mathcal{S}_i$ for each request i is a solution consists of k_i edge-disjoint paths connecting s_i and t_i. Applying the general framework, we deduce the following greedy algorithm.

Let ℓ_e be the load of edge e. Initially, set $\ell_e \leftarrow 0$ for every edge e. At the arrival of request i, compute a strategy consisting of k_i edge-disjoint paths from s_i and t_i such that the increase of the total cost is minimum. Select this strategy for request i and update ℓ_e.

We notice that computing the strategy for request i can be done efficiently. Given the current loads ℓ_e on every edge e, create a graph H consisting of the same vertices and edges as graph G. For each edge e in graph H, define the capacity to be 1 and the cost on e to be $f_e(p_{i,e}+\ell_e)-f_e(\ell_e)$. Then the computing k_i edge-disjoint paths from s_i and t_i with the minimal marginal cost in G is equivalent to solving a transportation problem in graph H.

Proposition 5 *If the congestion costs of all edges are (λ,μ)-smooth then the algorithm is $\lambda/(1-\mu)$-competitive. In particular, if $f_e(z) = z^{\alpha_e}$ then the algorithm is $O(\alpha^\alpha)$-competitive where $\alpha = \max_e \alpha_e$.*

Proof The proposition follows directly from Theorem 3 and the particular case is derived additionally by Lemma 6. □

Note that one can generalizes the problem to capture more general or different types of connectivity demands and the congestion costs are incurred from vertices instead of edges. The same results hold.

4.3 Online Vector Scheduling

Problem. In the problem, there are m unrelated machines and jobs arrive online. The load of a job j in machine i is specified by a vector $p_{ij} = \langle p_{ij}(k) : 1 \le k \le d\rangle$ where $p_{ij}(k) \ge 0$ and d, a fixed parameter, is the dimension of the vector. At the arrival of a job j, vectors p_{ij} for all i are revealed and job j must be assigned immediately to a machine. Given a job-machine assignment σ, the load in dimension k of machine i is defined as $\ell_{i,\sigma}(k) := \sum_{j:\sigma(j)=i} p_{ij}(k)$ for $1 \le k \le d$. The L_α-norm for $\alpha \ge 1$ in dimension k is $\|\Lambda_\sigma(k)\|_\alpha := \left(\sum_{i=1}^m \ell_{i,\sigma}(k)^\alpha\right)^{1/\alpha}$; and the L_∞-norm (makespan norm) in dimension k is $\|\Lambda_\sigma(k)\|_\infty := \max_{i=1}^m \ell_{i,\sigma}(k)$. In the L_α-norm, the objective is to find an online assignment σ minimizing $\max_k \|\Lambda_\sigma(k)\|_\alpha$. In the L_∞-norm, the objective is to find an online assignment σ minimizing $\max_k \|\Lambda_\sigma(k)\|_\infty$. An algorithm is r-competitive for the L_α-norm if it outputs an assignment σ such that for any assignment σ^*, it holds that $\max_k \|\Lambda_\sigma(k)\|_\alpha \le r \cdot \max_k \|\Lambda_{\sigma^*}(k)\|_\alpha$. Similarly for the L_∞-norm objective.

The online vector scheduling is introduced by Chekuri and Khanna [19]. Recently, Im et al. [36] showed an optimal competitive algorithm for this problem. Their analysis is based on a carefully constructed potential function. In the following, we can also derive an optimal algorithm for this problem based on our general framework. The analysis is much simpler and follows directly Theorem 3.

Contribution. In the problem, the set of strategy $\mathcal{S}_i$ for each job j is a machine i. Applying the general framework, we deduce the following greedy algorithm.

In the L_α-norm objective for $1 \le \alpha \ne \infty$, consider function $C(\sigma) := \sum_{k=1}^{d} \big(\sum_{i=1}^{m} \ell_{i,\sigma}(k)^\alpha\big)^{\frac{\alpha+\log d}{\alpha}}$ where σ is a job-machine assignment of all jobs released so far. In the L_∞-norm objective, consider function $C(\sigma) := \sum_{k=1}^{d} \sum_{i=1}^{m} \ell_i(k)^{\alpha+\log d}$ for $\alpha = \log(m)$.

Initially, σ is an empty assignment. At the arrival of j, assign j to machine i^* that minimizes the increase of $C(\sigma)$. Again, the assignment of a job j can be efficiently computed.

Proposition 6 ([36]) *For the L_α-norm objective, the algorithm is $O(\max\{\alpha, \log d\})$-competitive. For the L_∞-norm objective, the algorithm is $O(\log d + \log m)$-competitive.*

Proof Let σ^* is an optimal assignment for the L_α-norm objective. We have

$$\begin{aligned}
\bigg(\sum_{i=1}^{m} \ell_{i,\sigma}(k)^\alpha\bigg)^{\frac{\alpha+\log d}{\alpha}} &\le \sum_{k=1}^{d} \bigg(\sum_{i=1}^{m} \ell_{i,\sigma}(k)^\alpha\bigg)^{\frac{\alpha+\log d}{\alpha}} \\
&\le O\big(\alpha + \log d\big)^{\alpha+\log d} \cdot \sum_{k=1}^{d} \bigg(\sum_{i=1}^{m} \ell_{i,\sigma^*}(k)^\alpha\bigg)^{\frac{\alpha+\log d}{\alpha}} \\
&\le O\big(\alpha + \log d\big)^{\alpha+\log d} \cdot d \cdot \max_{k=1}^{d} \bigg(\sum_{i=1}^{m} \ell_{i,\sigma^*}(k)^\alpha\bigg)^{\frac{\alpha+\log d}{\alpha}}
\end{aligned}$$

In these inequalities, we apply Theorem 3 and Lemma 6 (note that $C(\sigma)$ is a polynomial of degree $(\alpha + \log d)$). Taking the $(\alpha + \log d)^{\text{th}}$ root, the result for L_α-norm objective follows.

For the L_∞-norm, with $\alpha = \log m$ similarly we have

$$\begin{aligned}
\max_{k=1}^{d} \max_{i=1}^{m} \ell_{i,\sigma}(k)^{\alpha+\log d} &\le \sum_{k=1}^{d} \sum_{i=1}^{m} \ell_{i,\sigma}(k)^{\alpha+\log d} \\
&\le \big(\alpha + \log d\big)^{\alpha+\log d} \cdot \sum_{k=1}^{d} \sum_{i=1}^{m} \ell_{i,\sigma^*}(k)^{\alpha+\log d} \\
&\le \big(\alpha + \log d\big)^{\alpha+\log d} \cdot d \cdot m \cdot \max_{k=1}^{d} \max_{i=1}^{m} \ell_{i,\sigma^*}(k)^{\alpha+\log d}
\end{aligned}$$

Again, taking the $(\alpha + \log d)^{\text{th}}$ root for $\alpha = \log m$, the proposition follows. □

4.4 Online Energy-Efficient Scheduling

Problem. Energy-efficient algorithms have received considerable attention and has been widely studied in scheduling. One main direction is to design algorithms toward a more realistic setting — online setting with multiple machine environment [1]. We consider an energy minimization problem in the online multiple machine setting. In the problem, we are given m unrelated machines and a set of jobs. Each job j is specified by its released date r_j, deadline d_j and processing volumes p_{ij} if job j is processed in machine i. We consider *non-migration* schedules; that is, every job j has to be assigned to exactly one machine and is fully processed in that machine during time interval $[r_j, d_j]$. However, jobs can be executed *preemptively*, meaning that a job can be interrupted during its execution and can be resumed later on. An algorithm can choose appropriate speed $s_i(t)$ for every machine i at any time t in order to complete all jobs. Every machine i has a non-decreasing energy

power function $P_i(s_i(t))$ depending on the speed $s_i(t)$. Typically, $P_i(z)$ has form z^{α_i} for constant $\alpha_i \geq 1$ or in a more general context, $P_i(z)$ is assumed to be convex. In the problem, we consider general non-decreasing continuous functions P_i *without* convexity assumption. The objective is to minimize the total energy consumption while completing all jobs. In the *online* setting, jobs arrive over time and the assignment and scheduling have to be done irrevocably.

In offline setting, for typical energy function $P(z) = z^\alpha$, the best known algorithms [30, 10] have competitive ratio $O(B_\alpha)$ where B_α is the Bell number. Prior to our work, the only known result for this online problem is in the single machine setting and the energy power function $P(z) = z^\alpha$. Specifically, Bansal et al. [11] gave a $2\big(\frac{\alpha}{\alpha-1}\big)^\alpha e^\alpha$-competitive algorithm. In terms of lower bounds, Bansal et al. [12] showed that no deterministic algorithm has competitive ratio less than $e^{\alpha-1}/\alpha$ for single machine. For unrelated machines, the lower bound $\Omega(\alpha^\alpha)$ follows the construction of Caragiannis [18] for Load Balacing (with L_α-norm) (by considering all jobs have the same span $[r_j, d_j] = [0, 1]$). Kling and Pietrzyk [38] gave a $O(\alpha^\alpha)$-competitive algorithm in the multi-identical-processor setting in which job migration is allowed. Surprisingly, no competitive algorithm is known in the non-migratory multiple-machine environment, that is in contrast to the similar online problem with objective as the total energy plus flow-time [2]. The main difference here is that for the latter, one can make a tradeoff between energy and flow-time and derive a competitive algorithm whereas for the former, one has to deal directly with a non-linear objective and no LP with relatively small integrality gap was known. We notice that Gupta et al. [31] gave also a primal-dual competitive algorithm for the single machine environment. However, their approach cannot be used for unrelated machines due to the large integrality gap of the formulation.

Contribution. In the problem, speed of a job can be an arbitrary (non-negative) real number. However, in order to employ tools from linear programming, we consider discrete setting with a small loss in the competitive ratio. Fix an arbitrary constant $\epsilon > 0$ and $\delta > 0$. Define the set of speeds $\mathcal{V} = \{\ell \cdot \epsilon : 0 \leq \ell \leq L\}$ for some sufficiently large L. During a time interval $[t, t+\delta]$, a job can be executed at a speed in $\mathcal{V}$. As the energy cost functions are continuous, this assumption on the setting worsens the energy cost by at most a factor $(1+\tilde{\epsilon})$ for arbitrarily small $\tilde{\epsilon}$. Given a job j, a set of feasible strategy $\mathcal{S}_j$ of j is a feasible non-migratory execution of a job j on some machine. Specifically, a strategy of job j can be described as the union over all machines i of solutions determined by the following program: $\sum_{t=r_j}^{d_j} \delta \cdot v_{ijt} \geq p_{ij}$ s.t. $v_{ijt} \in \mathcal{V}$, where in the sum we increment each time t by δ. Applying the general framework, we derive the following algorithm.

Algorithm. Assume that the energy functions are (λ, μ)-smooth. Let u_{it} be the speed of machine i at time t. Initially, set $u_{it} \leftarrow 0$ for every machine i and time t. At the arrival of a job j, compute the minimum energy increase if job j is assigned to machine i and define β_{ij} equal $1/\lambda$ times that minimum energy. It is indeed an optimization problem

$$\min \sum_{r_j}^{d_j} \delta \cdot \Big[P_i\big(u_{it} + v_{ijt}\big) - P_i\big(u_{it}\big) \Big] \quad \text{s.t} \quad \sum_{r_j}^{d_j} \delta \cdot v_{ijt} \geq p_{ij}, \quad v_{ijt} \in \mathcal{V} \tag{7}$$

Observe that if P_i is a convex function then it is a convex program and can be solved efficiently. In this case, using the KKT conditions, the optimal solution can be constructed as follows. We initiate a variable v_{ijt} as 0. While $\sum_{r_j}^{d_j} \delta \cdot v_{ijt} < p_{ij}$, i.e., the total volume of job j has not been completed, continue increasing v_{ijt} at $\arg\min_{r_j \leq t \leq d_j} u_{it} + v_{ijt}$. Note that this is exactly algorithm OA in [11] for a single machine. Let $v^*_{i^*jt}$ be an optimal solution of the mathematical program (7). Then, assign job j to machine $i^* \in \arg\min_i \beta_{ij}$ and execute j at time t by speed $v^*_{i^*jt}$.

Proposition 7 *If the energy cost functions are (λ, μ)-smooth then the algorithm is $(1+\epsilon)\lambda/(1-\mu)$-competitive for arbitrarily small ϵ. In particular, if $P_i(z) = z^{\alpha_i}$ then the algorithm is $(1+\epsilon)O(\alpha^\alpha)$-competitive where $\alpha = \max_i \alpha_i$.*

Proof The proposition follows directly from Theorem 3. In the particular case $P_i(z) = z^{\alpha_i}$, the functions are (λ, μ)-smooth with $\mu = (\alpha - 1)/\alpha$ and $\lambda = O(\alpha^{\alpha-1})$ by Lemma 6. The competitive ratio of this case follows. □

4.5 Online Prize Collecting Energy-Efficient Scheduling

Problem. We consider the same setting as in the Energy Minimization problem. Additionally, each job j has a penalty π_j. There is no penalty from job j if it is completely executed during $[r_j, d_j]$ in some machine i. Otherwise, if job j is not completed (even most volume of job j have been executed) then the algorithm has to pay a penalty π_j. The objective is to minimize the total penalty of uncompleted jobs plus the energy cost.

Contribution. The result does not follow immediately from Theorem 3 but the approach is exactly the one in the general framework.

By the same formulation as the previous section, assume that the set of speeds is finite and discrete. The set of feasible strategy $\mathcal{S}_j$ of a job j is a feasible non-migratory execution of a job j on some machine, defined in the previous section. The sets $\mathcal{S}_j$'s are also finite and discrete. We say that A is a *configuration* of machine i if it is a schedule of a subset of jobs in i. Specifically, a configuration A consists of tuples (i, j, k) specifying that a job j is assigned to machine i and is executed according to strategy $s_{jk} \in \mathcal{S}_j$.

We are now formulating a configuration LP for the problem. Let x_{ijk} be a variable indicating whether job j is processed in machine i according to strategy $s_{jk} \in \mathcal{S}_j$. For configuration A and machine i, let z_{iA} be a variable such that $z_{iA} = 1$ if and only if $x_{ijk} = 1$ for every $(i, j, k) \in A$. In other words, $z_{iA} = 1$ iff A is the solution of the problem restricted on machine i. Let $c_{i,A}$ be the energy cost of configuration A in machine i. We consider the following formulation.

$$
\begin{aligned}
\min \sum_{i,A} c_{i,A} z_{iA} + \sum_j \left(1 - \sum_{i,k} x_{ijk}\right)\pi_j & \\
\sum_{i,k} x_{ijk} \leq 1 & \qquad \forall j \\
\sum_{A:(i,j,k)\in A} z_{iA} = x_{ijk} & \qquad \forall i, j, k \\
\sum_A z_{iA} = 1 & \qquad \forall i \\
x_{ijk}, z_{iA} \in \{0, 1\} & \qquad \forall i, j, A
\end{aligned}
$$

The first constraint guarantees that a job j can be assigned to at most one machine i and is executed according to at most one feasible strategy. The second constraint ensures that if job j is assigned to machine i and is executed according to strategy $s_{jk} \in \mathcal{S}_j$ then the configuration corresponding to the solution restricted on machine i must contain (i, j, k). The third constraint

says that there is always a configuration associated to machine i for every i. The dual of the relaxation reads

$$\begin{aligned}
\max \sum_j (\pi_j - \alpha_j) + \sum_i \gamma_i & \\
\alpha_j + \beta_{ijk} &\geq \pi_j && \forall i, j, k \\
\gamma_i + \sum_{(i,j,k)\in A} \beta_{ijk} &\leq c_{iA} && \forall i, A \\
\alpha_j &\geq 0 && \forall j
\end{aligned}$$

Greedy Algorithm. Assume that all energy power functions are (λ, μ)-smooth. Fix λ and μ. At the arrival of job j, compute the minimum energy increase if j is assigned to some machine i. If the minimum energy increase is larger than $\lambda \cdot \pi_j$ then reject the job. Otherwise, assign and execute j such that the energy increase is minimum.

Proposition 8 *Assume that all energy power functions are (λ, μ)-smooth. Then the algorithm is $\lambda/(1-\mu)$-competitive.*

Proof We define the dual variables similarly as in the general framework. Let $A^*_{i,\prec j}$ be configuration of machine i (due to the algorithm) before the arrival of job j. (Initially, $A^*_{i,\prec 1} \leftarrow \emptyset$ for every machine i.) For each machine i and a strategy $s_{jk} \in \mathcal{S}_j$ such that s_{jk} is a schedule of j in machine i, define

$$\beta_{ijk} = \frac{1}{\lambda}\left[c_i(A^*_{i,\prec j} \cup s_{jk}) - c_i(A^*_{i,\prec j})\right].$$

If s_{jk} is not a schedule of j in machine i then define $\beta_{ijk} = \infty$. Moreover, define

$$\alpha_j = \max\{\pi_j - \min_{i,k} \beta_{ijk}, 0\} \qquad \text{and} \qquad \gamma_i = \frac{\mu}{\lambda} c_i(A^*_i)$$

where A^*_i is the configuration of machine i at the end of the instance (when all jobs have been released).

The variables constitute a dual feasible solution. The first dual constraint follows the definition of α_j. The second dual constraint follows the definition of (λ, μ)-smoothness. Note that that for any configuration A of a machine i (a feasible schedule in machine i), if $(i, j, k) \in A$ then by definition of dual variables, $\beta_{ijk} \neq \infty$.

We are now bounding the dual. The algorithm has the property immediate-reject. It means that if the algorithm accepts a job then the job will be completed; and otherwise, the job is rejected at its arrival. By the algorithm, $\alpha_j = 0$ for every rejected job j. Besides, if job j is accepted then $\pi_j - \alpha_j = \beta_{ijk}$ where i is the machine to which job j is assigned and job j is executed according to strategy s_{jk}. Therefore, by the definition of dual variables, $\sum_j (\pi_j - \alpha_j)$, where the sum is taken over accepted jobs j, equals $1/\lambda$ times the total energy consumption. Recall that the total energy consumption of the algorithm is $\sum_i c_i(A^*_i)$. The dual objective is

$$\sum_j (\pi_j - \alpha_j) + \sum_i \gamma_i = \sum_{j:j \text{ rejected}} \pi_j + \frac{1}{\lambda}\sum_i c_i(A^*_i) - \frac{\mu}{\lambda}\sum_i c_i(A^*_i)$$

Moreover, the primal is equal to the total penalty of rejected jobs plus $\sum_i c_i(A^*_i)$. Therefore, the ratio between primal and dual is at most $\lambda/(1-\mu)$. □

4.6 Facility Location with Client-Dependent Facility Cost

Non-Convex Facility Location. In the problem, we are given a metric space (M, d) is given and clients arrive online. Let N and n be the set and the number of clients, respectively. A location $i \in M$ is characterized by a fixed opening cost a_i and an arbitrary non-decreasing serving cost function $f_i : 2^N \to \mathbb{R}^+$. If a subset S of clients is served by a facility at location i then the facility cost at this location is $a_i + f_i(S)$. At the arrival of a client, an algorithm need to assign the client to some facility. The goal is to minimize the total cost, which is the total distance from clients to their facilities plus the total facility cost.

Facility Location is one of the most widely studied problems. In the classic version, the facility cost consists only of the opening cost. There is a large literature in the offline setting. In online setting, Meyerson [43] gave a randomized $O(\frac{\log n}{\log \log n})$-competitive algorithm. This competitive ratio matches to the randomized lower bound due to Fotakis [28]. For deterministic algorithms, Fotakis [27] first presented a primal-dual $O(\log n)$-competitive algorithm and subsequently improved to the optimal $O(\frac{\log n}{\log \log n})$-competitive algorithm [28]. The online capacitated facility location in which function $f_i(S) = 0$ if $|S| \leq u_i$ for some capacity u_i and $f_i(S) = \infty$ has been studied in [6]. Using a primal-dual framework for mixed packing and covering constraints, the authors derived a $O(\log m \log mn)$-competitive algorithm.

Contribution. We derive a competitive algorithm by combining the primal-dual algorithm due to Fotakis [27] for the online (classic) facility location and our primal-dual framework for non-convex functions.

Let x_{ij} and y_i be variables indicating whether client j is assigned to facility i and whether facility i is open, respectively. For subset $S \subset N$, let $z_{i,S}$ be a variable such that $z_{i,S} = 1$ if and only if $x_{ij} = 1$ for every client $j \in S$, and $x_e = 0$ for $j \notin S$. We consider the following formulation and the dual of its relaxation.

$$\begin{aligned}
\min \sum_i a_i y_i + \sum_{i,j} d_{ij} x_{ij} + \sum_{i,S} f_i(S) z_{i,S} & \\
\sum_i x_{ij} \geq 1 \qquad & \forall j \\
y_i \geq x_{ij} \qquad & \forall i, j \\
\sum_{S: j \in S} z_{i,S} = x_{ij} \qquad & \forall i, j \\
\sum_S z_{i,S} = 1 \qquad & \forall i \\
x_{ij}, z_{i,S} \in \{0, 1\} \qquad & \forall i, j, S
\end{aligned}$$

$$\begin{aligned}
\max \sum_j \alpha_j + \sum_i \theta_i & \\
\alpha_j \geq d_{ij} + \beta_{ij} + \gamma_{ij} \qquad & \forall i, j \\
\sum_j \beta_{ij} \leq a_i \qquad & \forall i \\
\theta_i + \sum_{j \in S} \gamma_{ij} \leq f_i(S) \qquad & \forall i, S \\
\alpha_j, \beta_{ij} \geq 0 \qquad & \forall i, j
\end{aligned}$$

Algorithm. Assume that all serving cost f_i are (λ, μ)-smooth. Intuitively, β_{ij} and γ_{ij} can be interpreted as the contributions of client j to the opening cost and the serving cost at location i. At the arrival of client j, continuously increase α_j. For any facility such that $\alpha_j = d_{ij}$, start increasing β_{ij}. If $\sum_{j'} \beta_{ij'} = a_i$ then (stop increasing β_{ij}) start increasing γ_{ij} until $\frac{\mu}{\lambda}\big[f_i(S \cup j) - f_i(S)\big]$ where S is the current set of clients assigned to i. Assign j to the first facility i such that $\gamma_{ij} = \frac{\mu}{\lambda}\big[f_i(S \cup j) - f_i(S)\big]$ (open i if it has not been opened).

Proposition 9 *Assume that all serving cost f_i are (λ, μ)-smooth. Then the algorithm is $O\big(\log n + \frac{\lambda}{1-\mu}\big)$-competitive.*

Proof We define dual variables similarly as in Theorem 3. The α-variables, β-variables and γ-variables are defined in the algorithm. Define θ_i equal $-1/\lambda$ times the (final) serving cost at facility i. Let $\pi(j)$ is the facility to which j is assigned and $\pi(N)$ the set of all open facilities by the algorithm.

The dual variables constitute a feasible solution. The first and second dual constraints are due to the algorithm. Note that by the definition of γ-variables, it always holds that $\gamma_{ij} \leq \frac{\mu}{\lambda}\big[f_i(S \cup j) - f_i(S)\big]$ where S is the set of clients assigned to i before the arrival of j. The last constraint follows the (λ, μ)-smoothness of serving costs. We are now bounding the primal and the dual. We have

$$\sum_{i \in \pi(N)} a_i + \sum_j d_{\pi(j),j} \leq O(\log n) \sum_j \Big(\alpha_j - \gamma_{\pi(j),j}\Big)$$

$$\sum_i f_i\big(\pi^{-1}(i)\big) \leq \frac{\lambda}{1-\mu}\Big(\sum_j \gamma_{\pi(j),j} + \sum_i \theta_i\Big)$$

where the first inequality is due to Fotakis [27] and the second one follows the definition of dual variables. The proposition follows. □

5 Conclusion

In this paper, we have presented primal-dual approaches based on configuration linear programs to design competitive algorithms for covering problems with non-convex objectives. Non-convexity until now is considered as a barrier in optimization. We hope that our approach would contribute some elements toward the study of non-linear/non-convex problems. Our work gives raise to several directions. Motivating by the results for the DPC and the ICF problems, the most interesting question is whether one can give improvement for different problems using appropriate continuous extensions (with small local smoothness parameters). Besides, the local-smoothness has provided tight bounds for classes of polynomials with non-negative coefficients. So a question is to determine tight bounds for different classes of functions. Moreover, is there connection between local-smoothness and total curvature in submodular optimization? Intuitively, both concepts measure how far way a function is from being modular.

Appendix

A Technical Lemmas

In this section, we show technical lemmas in order to determine smoothness parameters for polynomials with non-negative coefficients. The following lemma has been proved in [20].

Lemma 5 ([20]) *Let k be a positive integer. Let $0 < a(k) \leq 1$ be a function on k. Then, for any $x, y > 0$, it holds that*

$$y(x+y)^k \leq \frac{k}{k+1}a(k)x^{k+1} + b(k)y^{k+1}$$

where α is some constant and

$$b(k) = \begin{cases} \Theta\left(\alpha^k \cdot \left(\frac{k}{\log ka(k)}\right)^{k-1}\right) & \text{if } \lim_{k\to\infty}(k-1)a(k) = \infty, \quad (8a) \\ \Theta\left(\alpha^k \cdot k^{k-1}\right) & \text{if } (k-1)a(k) \text{ are bounded } \forall k, \quad (8b) \\ \Theta\left(\alpha^k \cdot \frac{1}{ka(k)^k}\right) & \text{if } \lim_{k\to\infty}(k-1)a(k) = 0. \quad (8c) \end{cases}$$

Lemma 6 *For any sequences of non-negative real numbers $\{a_1, a_2, \ldots, a_n\}$ and $\{b_1, b_2, \ldots, b_n\}$ and for any polynomial g of degree k with non-negative coefficients, it holds that*

$$\sum_{i=1}^{n}\left[g\left(b_i + \sum_{j=1}^{i} a_j\right) - g\left(\sum_{j=1}^{i} a_j\right)\right] \leq \lambda(k)\cdot g\left(\sum_{i=1}^{n} b_i\right) + \mu(k)\cdot g\left(\sum_{i=1}^{n} a_i\right)$$

where $\mu(k) = \frac{k-1}{k}$ and $\lambda(k) = \Theta\left(k^{k-1}\right)$. The same inequality holds for $\mu(k) = \frac{k-1}{k\ln k}$ and $\lambda(k) = \Theta\left((k\ln k)^{k-1}\right)$.

Proof We first prove for $\mu(k) = \frac{k-1}{k}$ and $\lambda(k) = \Theta\left(k^{k-1}\right)$. Let $g(z) = g_0 z^k + g_1 z^{k-1} + \cdot + g_k$ with $g_t \geq 0\ \forall t$. The lemma holds since it holds for every z^t for $0 \leq t \leq k$. Specifically,

$$\begin{aligned} \sum_{i=1}^{n}\left[g\left(b_i + \sum_{j=1}^{i} a_j\right) - g\left(\sum_{j=1}^{i} a_j\right)\right] &= \sum_{t=1}^{k} g_{k-t}\cdot\sum_{i=1}^{n}\left[\left(b_i + \sum_{j=1}^{i} a_j\right)^t - \left(\sum_{j=1}^{i} a_j\right)^t\right] \\ &\leq \sum_{t=1}^{k} g_{k-t}\cdot\left[t\cdot b_i\cdot\left(b_i + \sum_{j=1}^{i} a_j\right)^{t-1}\right] \leq \sum_{t=1}^{k} g_{k-t}\cdot\left[\lambda(t)\left(\sum_{i=1}^{n} b_i\right)^t + \mu(t)\left(\sum_{i=1}^{n} a_i\right)^t\right] \\ &\leq \lambda(k)\cdot g\left(\sum_{i=1}^{n} b_i\right) + \mu(k)\cdot g\left(\sum_{i=1}^{n} a_i\right) \quad (9) \end{aligned}$$

The first inequality follows the convex inequality $(x+y)^{k+1} - x^{k+1} \leq (k+1)y(x+y)^k$. The second inequality follows Lemma 5 (Case 2 and $a(k) = 1/(k+1)$). The last inequality holds since $\mu(t) \leq \mu(k)$ and $\lambda(t) \leq \lambda(k)$ for $t \leq k$.

The case $\mu(k) = \frac{k-1}{k\ln k}$ and $\lambda(k) = \Theta\left((k\ln k)^{k-1}\right)$ is proved similarly. The only different step is in the second inequality of (9). In fact, applying Lemma 5 (Case 3 and $a(k) = \frac{1}{(k+1)\ln k}$), one gets the lemma inequality for $\mu(k) = \frac{k-1}{k\ln k}$ and $\lambda(k) = \Theta\left((k\ln k)^{k-1}\right)$. □

References

[1] Susanne Albers. Energy-efficient algorithms. *Commun. ACM*, 53(5):86–96, 2010.

[2] S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *Proc. 23rd ACM-SIAM Symposium on Discrete Algorithms*, pages 1228–1241, 2012.

[3] Matthew Andrews, Spyridon Antonakopoulos, and Lisa Zhang. Minimum-cost network design with (dis)economies of scale. In *Proc. 51th Symposium on Foundations of Computer Science (FOCS)*, pages 585–592, 2010.

[4] Matthew Andrews, Antonio Fernández Anta, Lisa Zhang, and Wenbo Zhao. Routing for power minimization in the speed scaling model. *IEEE/ACM Trans. Netw.*, 20(1):285–294, 2012.

[5] Antonios Antoniadis, Sungjin Im, Ravishankar Krishnaswamy, Benjamin Moseley, Viswanath Nagarajan, Kirk Pruhs, and Cliff Stein. Hallucination helps: Energy efficient virtual circuit routing. In *Proc. 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1141–1153, 2014.

[6] Yossi Azar, Umang Bhaskar, Lisa Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 85–100, 2013.

[7] Yossi Azar, Nikhil R. Devanur, Zhiyi Huang, and Debmalya Panigrahi. Speed scaling in the non-clairvoyant model. In *Proc. 27th ACM on Symposium on Parallelism in Algorithms and Architectures*, pages 133–142, 2015.

[8] Yossi Azar, Niv Buchbinder, TH Hubert Chan, Shahar Chen, Ilan Reuven Cohen, Anupam Gupta, Zhiyi Huang, Ning Kang, Viswanath Nagarajan, Joseph Seffi Naor, and Debmalya Panigrahi. Online algorithms for covering and packing problems with convex objectives. In *Proc. Symposium on Foundations of Computer Science (FOCS)*, 2016.

[9] Maria-Florina Balcan and Nicholas J. A. Harvey. Learning submodular functions. In *Proc. Machine Learning and Knowledge Discovery in Databases*, pages 846–849, 2012.

[10] Evripidis Bampis, Alexander V. Kononov, Dimitrios Letsios, Giorgio Lucarelli, and Maxim Sviridenko. Energy efficient scheduling and routing via randomized rounding. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 449–460, 2013.

[11] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1), 2007.

[12] Nikhil Bansal, Ho-Leung Chan, Dmitriy Katz, and Kirk Pruhs. Improved bounds for speed scaling in devices obeying the cube-root rule. *Theory of Computing*, 8(1):209–229, 2012.

[13] Avrim Blum, Anupam Gupta, Yishay Mansour, and Ankit Sharma. Welfare and profit maximization with production costs. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 77–86. IEEE, 2011.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[15] Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2009.

[16] Niv Buchbinder, Joseph Seffi Naor, R Ravi, and Mohit Singh. Approximation algorithms for online weighted rank function maximization under matroid constraints. In *International Colloquium on Automata, Languages, and Programming*, pages 145–156, 2012.

[17] Niv Buchbinder, Moran Feldman, and Roy Schwartz. Online submodular maximization with preemption. In *Proc. 26th Symposium on Discrete Algorithms*, pages 1202–1216, 2015.

[18] Ioannis Caragiannis. Better bounds for online load balancing on unrelated machines. In *Proc. 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 972–981, 2008.

[19] Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM journal on computing*, 33(4):837–851, 2004.

[20] Johanne Cohen, Christoph Dürr, and Nguyen Kim Thang. Smooth inequalities and equilibrium inefficiency in scheduling games. In *International Workshop on Internet and Network Economics*, pages 350–363, 2012.

[21] Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete applied mathematics*, 7(3):251–274, 1984.

[22] Amit Deshpande, Madhur Tulsiani, and Nisheeth K Vishnoi. Algorithms and hardness for subspace approximation. In *Proc. 22nd Symposium on Discrete Algorithms*, pages 482–496, 2011.

[23] Nikhil R. Devanur and Zhiyi Huang. Primal dual gives almost optimal energy efficient online algorithms. In *Proc. 25th ACM-SIAM Symposium on Discrete Algorithms*, 2014.

[24] Nikhil R. Devanur and Kamal Jain. Online matching with concave returns. In *Proc. 44th ACM Symposium on Theory of Computing*, pages 137–144, 2012.

[25] Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *Symposium on Foundations of Computer Science (FOCS)*, pages 570–579, 2011.

[26] Moran Feldman, Ola Svensson, and Rico Zenklusen. Online contention resolution schemes. In *Proc. 27th Symposium on Discrete Algorithms*, pages 1014–1033, 2016.

[27] Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *Journal of Discrete Algorithms*, 5(1):141–148, 2007.

[28] Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1): 1–57, 2008.

[29] Michel X Goemans, Nicholas JA Harvey, Satoru Iwata, and Vahab Mirrokni. Approximating submodular functions everywhere. In *Proc. 20th Symposium on Discrete algorithms*, pages 535–544, 2009.

[30] Gero Greiner, Tim Nonner, and Alexander Souza. The bell is ringing in speed-scaled multiprocessor scheduling. *Theory of Computing Systems*, 54(1):24–44, 2014.

[31] Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *Proc. 10th Workshop on Approximation and Online Algorithms*, pages 173–186, 2012.

[32] Venkatesan Guruswami, Prasad Raghavendra, Rishi Saket, and Yi Wu. Bypassing UGC from some optimal geometric inapproximability results. *ACM Transactions on Algorithms*, 12(1):6, 2016.

[33] Zhiyi Huang and Anthony Kim. Welfare maximization with production costs: a primal dual approach. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 59–72. SIAM, 2015.

[34] Sascha Hunold. One step toward bridging the gap between theory and practice in moldable task scheduling with precedence constraints. *Concurrency and Computation: Practice and Experience*, 27(4):1010–1026, 2015.

[35] Sungjin Im, Janardhan Kulkarni, Kamesh Munagala, and Kirk Pruhs. Selfishmigrate: A scalable algorithm for non-clairvoyantly scheduling heterogeneous processors. In *Proc. 55th IEEE Symposium on Foundations of Computer Science*, 2014.

[36] Sungjin Im, Nathaniel Kell, Janardhan Kulkarni, and Debmalya Panigrahi. Tight bounds for online vector scheduling. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 525–544. IEEE, 2015.

[37] Rishabh K Iyer, Stefanie Jegelka, and Jeff A Bilmes. Curvature and optimal algorithms for learning and minimizing submodular functions. In *Advances in Neural Information Processing Systems*, pages 2742–2750, 2013.

[38] Peter Kling and Peter Pietrzyk. Profitable scheduling on multiple speed-scalable processors. *ACM Transactions on Parallel Computing*, 2(3):19, 2015.

[39] Andreas Krause and Daniel Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press, 2014.

[40] Ravishankar Krishnaswamy, Viswanath Nagarajan, Kirk Pruhs, and Cliff Stein. Cluster before you hallucinate: approximating node-capacitated network design and energy efficient routing. In *Proc. 46th Symposium on Theory of computing*, pages 734–743, 2014.

[41] Konstantin Makarychev and Maxim Sviridenko. Solving optimization problems with diseconomies of scale via decoupling. In *Proc. 55th Symposium on Foundations of Computer Science*, pages 571–580, 2014.

[42] Ishai Menache and Mohit Singh. Online caching with convex costs. In *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*, pages 46–54. ACM, 2015.

[43] Adam Meyerson. Online facility location. In *Proceedings. 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001.

[44] Viswanath Nagarajan and Xiangkun Shen. Online covering with sum of ℓ_q-norm objectives. In *Proc. 44th Colloquium on Automata, Languages and Programming (ICALP)*, 2017.

[45] Kim Thang Nguyen. Lagrangian duality in online scheduling with resource augmentation and speed scaling. In *Proc. 21st European Symposium on Algorithms*, pages 755–766, 2013.

[46] Kim Thang Nguyen. A greedy algorithm for subspace approximation. In *16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, 2018 (to appear).

[47] Tim Roughgarden. Intrinsic robustness of the price of anarchy. *Journal of the ACM*, 62(5): 32, 2015.

[48] Maxim Sviridenko, Jan Vondrák, and Justin Ward. Optimal approximation for submodular and supermodular optimization with bounded curvature. *Mathematics of Operations Research*, 2017.

[49] Jan Vondrák. Polyhedral techniques in combinatorial optimization. Lecture notes, Nov 18 2010.

[50] Jan Vondrák. Submodularity and curvature: The optimal algorithm. *RIMS Kokyuroku Bessatsu*, 2010.