A water-filling primal-dual algorithm for approximating non-linear covering problems

Fielbaum, Andrés; Morales, Ignacio; Verschae, José

DOI
10.4230/LIPIcs.ICALP.2020.46

Publication date
2020

Document Version
Final published version

Published in
Proceedings 47th International Colloquium on Automata, Languages, and Programming, ICALP 2020

A Water-Filling Primal-Dual Algorithm for Approximating Non-Linear Covering Problems

Andrés Fielbaum
Department of Cognitive Robotics, Faculty of Mechanical, Maritime and Materials Engineering, TU Delft, The Netherlands
a.s.fielbaumschnitzler@tudelft.nl

Ignacio Morales
Departamento de Ingeniería Industrial, Escuela de Ingeniería, Pontificia Universidad Católica, Santiago, Chile
inmorales@uc.cl

José Verschae
Instituto de Ingeniería Matemática y Computacional, Facultad de Matemáticas y Escuela de Ingeniería, Pontificia Universidad Católica, Santiago, Chile
jverschae@uc.cl

Abstract

Obtaining strong linear relaxations of capacitated covering problems constitute a significant technical challenge even for simple settings. For one of the most basic cases, the Knapsack-Cover (Min-Knapsack) problem, the relaxation based on *knapsack-cover inequalities* has an integrality gap of 2. These inequalities are exploited in more general problems, many of which admit primal-dual approximation algorithms.

Inspired by problems from power and transport systems, we introduce a general setting in which items can be taken fractionally to cover a given demand. The cost incurred by an item is given by an arbitrary non-decreasing function of the chosen fraction. We generalize the knapsack-cover inequalities to this setting an use them to obtain a $(2+\varepsilon)$-approximate primal-dual algorithm. Our procedure has a natural interpretation as a bucket-filling algorithm which effectively overcomes the difficulties implied by having different slopes in the cost functions. More precisely, when some superior segment of an item presents a low slope, it helps to increase the priority of inferior segments. We also present a rounding algorithm with an approximation guarantee of 2.

We generalize our algorithm to the Unsplittable Flow-Cover problem on a line, also for the setting of fractional items with non-linear costs. For this problem we obtain a $(4+\varepsilon)$-approximation algorithm in polynomial time, almost matching the 4-approximation algorithm known for the classical setting.

2012 ACM Subject Classification Theory of computation → Packing and covering problems; Mathematics of computing → Discrete optimization; Mathematics of computing → Linear programming

Keywords and phrases Knapsack-Cover Inequalities, Non-Linear Knapsack-Cover, Primal-Dual, Water-Filling Algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.46

Category Track A: Algorithms, Complexity and Games

Related Version A full version of this manuscript can be found at `https://arxiv.org/abs/1912.12151`.

Funding This article was partially funded by Fondecyt Project Nr. 1181527.

1 Introduction

Covering problems have been heavily studied by the combinatorial optimization community. Understanding their polyhedral descriptions, and how to approximate them, is a challenging and important task even for simple variants. One of the main tools for obtaining strong linear relaxations of covering problems are the *knapsack-cover inequalities*, introduced by Carr et al. [9], and have been used extensively [4, 18, 3, 3, 8, 10, 19, 1, 20, 2, 22]. Although the inequalities are exponentially many, they can be approximately separated up to a factor of $1+\varepsilon$ in polynomial time. In many cases they are well adjusted for primal-dual algorithms, which avoid having to solve the relaxation and yield faster combinatorial algorithms [4, 8, 10, 20].

In the classical Knapsack-Cover problem we are given a set of n items N and a demand $D \in \mathbb{N}$. Each item i has an associated covering capacity u_i and cost c_i. The objective is to choose a subset of items covering D at a minimum cost. We introduce a new natural generalization of this problem motivated by applications on the operation of power systems and the design of public transport systems. In this version we can choose items partially at a given cost, which might be non-linear. More precisely, each item $i \in N$ have an associated non-decreasing cost function $f_i : \mathbb{N} \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$. We must choose a number $x_i \in \mathbb{N}$ for each i such that $\sum_{i \in N} x_i \geq D$. The cost of such solution is $\sum_{i \in N} f_i(x_i)$. We can reduce the Knapsack-Cover problem to this setting by considering $f_i(0) = 0$, $f_i(1) = \ldots = f_i(u_i) = c_i$ and $f_i(x) = \infty$ for $x > u_i$. We say that we are in the *list model* if the input contains the numbers $f_i(0), f_i(1), \ldots, f_i(D)$ explicitly as a list. In this case the reduction above is pseudo-polynomial. On the other hand, if each f_i is given by an oracle that outputs $f_i(x)$ for any x, we say that we are in the *oracle model*. In this case the reduction above can be made polynomial. We call our newly introduced problem the *Non-Linear Knapsack-Cover problem*. Our setting also generalizes the *Single-Demand Facility Location* problem studied by Carnes and Shmoys [8]. In this setting each item (facility) has an activation cost b_i and then the cost grows linearly at a rate of a_i, that is, $f_i(0) = 0$, $f_i(x) = b_i + c_i x$ for $x \in \{1, \ldots, u_i\}$, and $f_i(x) = \infty$ otherwise.

More generally, we study the Non-Linear variant of the Unsplittable Flow-Cover problem on a path (UFP-cover), that extends the Non-Linear Knapsack-Cover problem. In the original UFP-cover problem, first considered by Bar-Noy et al. [4], we have a discrete interval $I = \{1, \ldots, k\}$ and a set N of n items, each one characterized by a capacity or height u_i, a cost c_i, and a sub-interval $I_i \subseteq \{1, \ldots, k\}$. We also have a demand D_t for each $t \in I$. The problem consists on selecting the cheapest set of items such that the total height at any point in I is at least the demand, that is, we must pick a set S minimizing $\sum_{i \in S} c_i$ such that $\sum_{i \in S : I_i \ni t} u_i \geq D_t$ for all $t \in I$. In this paper we generalize this problem to the case where items can be taken partially. As before we are giving a non-decreasing function $f_i = \mathbb{N} \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$ for each item. We can choose to set the height of any item to a value $x_i \in \mathbb{N}$ by paying a cost $f_i(x_i)$. We mush choose heights in order to cover the demand at each point $t \in I$ at a minimum total cost $\sum_i f_i(x_i)$. Notice that this setting generalizes Non-Linear Knapsack-Cover.

In this article we provide a generalization of the knapsack-cover inequalities to the Non-Linear Knapsack-Cover problem, which we also apply to Non-Linear UFP-Cover. The obtained relaxations yield primal-dual algorithms matching the classical settings. Namely, for Non-Linear Knapsack-Cover we show a 2-approximation algorithm, and for Non-Linear UFP-Cover a 4-approximation algorithm, both running in polynomial time in the list model. For the oracle model, they can be adapted to yield a $(2 + \varepsilon)$- and $(4 + \varepsilon)$-approximation, respectively, in polynomial time. Additionally, we show a rounding technique for the Non-Linear Knapsack-Cover case also achieving a 2-approximation for the list model, together with a polynomial time separation algorithm.

Applications

One of our main motivations for considering non-linear cost functions comes from the Unit Commitment Problem (UCP), a prominent problem in the operation of power systems. In its most basic version, a central planner, called Independent System Operator (ISO), must schedule the production of energy generated from a given set of power plants, in order to satisfy a given demand. For the case of one time period, the problem corresponds to Non-Linear Knapsack-Cover. More precisely, items correspond to power plants, and $f_i(x)$ is the cost of producing x units of power by power plant i. A common issue in this setting is that plants incur fixed costs for starting production, and after the resource is available, a minimum amount of energy must be produced. It is worth noticing that after paying the fixed cost the behavior of the cost functions might be non-linear and are often modelled by convex quadratic functions [24].

On the other hand, Non-Linear UFP-Cover appears in the optimization of transport systems. Consider an avenue with several bus stops $\{1, \dots, k\}$. We interpret the avenue as a path network where bus stops correspond to edges. Passengers need to move (in a single direction) within bus stops, and hence, we associate to each passenger a set of consecutive bus stops (i.e., a path) which they need to traverse. The set of paths defines a flow that needs to traverse the network. Hence, we obtain a demand D_t at each bus stop t, representing the total number of passengers (amount of flow) that must traverse it. On the supply side, there are potential bus transit lines, each covering some sub-path of the avenue, and hence covering the demand on some subset of consecutive bus stops $I_i = \{f_i, f_i + 1, \dots, l_i\} \subseteq \{1, \dots, k\}$. As the planner of this system, we must choose which lines to operate. Additionally, for each chosen line, we must choose its operation frequency and the type of vehicle to use. Such combinations of frequency and vehicle define the amount of passengers (demand) the line can transport. Assume that for line i and each demand x, we can optimally choose (that is, we have an oracle) the frequency and vehicle combination to minimize the operating cost of the line, which we call $f_i(x)$. In other words, for each line i, we must choose a demand $x_i \geq 0$ to cover, such that the total operation cost $\sum_i f_i(x_i)$ is minimized. It is not hard to see that covering the demand at each bus stop guarantees that all passengers can be transported. Hence, it suffices that $\sum_{i:t \in I_i} x_i \geq D_t$ holds for each bus stop $t \in \{1, \dots, k\}$. Hence, we obtain an instance of Non-Linear UFP-Cover.

We might wonder what type of cost functions f_i one can get in this setting. There is a vast literature concerning economies of scale in public transport lines: for instance, Mohring [21] states that there are economies of scale in public transport, Fielbaum et al. [14] show that they get exhausted. Coulombel and Monchambert [12] propose that the system could face diseconomies of scale when the demand exceeds certain thresholds. Hence, techniques to manage non-linear functions (that can have convex and concave regions) are needed.

Related Work

The use of the primal-dual method to derive approximation algorithms is introduced by Bar-Yehuda and Even [5] and Chvátal [11], becoming one of the major tools for designing approximation algorithms [23]. Bar-Noy et al. [4] are the firsts to consider the primal-dual framework based on knapsack-cover inequalities. However, they pose their algorithm in the equivalent local-ratio framework [6], even before the knapsack-cover inequalities were introduced and without stating the underlying LP-relaxation. Their techniques yield a 4-approximation algorithm and their analysis is tight [10]. Additionally, this problem admits a quasi-polynomial time approximation scheme (QPTAS) [16]. On the other hand,

Carnes and Shmoys [8] give an explicit description of the primal-dual method, obtaining a 2-approximation algorithm for Knapsack-Cover, the Single-Demand Facility Location problem, and the more general Single-item Lot-Sizing problem with Linear Holding Costs. Cheung et al. [10] consider the Generalized Min-Sum Scheduling problem on a single machine without release dates; they obtain a $(4+\varepsilon)$-approximation algorithm based on the primal-dual framework on an LP with knapsack-cover inequalities. Finally, McCormick et al. [20] consider covering problems with precedence constraints, where they give a primal-dual algorithm with approximation ratio equal to the width of the precedence relations. We remark that Non-Linear Knapsack-Cover can be modeled within this framework, but applying this result to this case yields an unbounded approximation guarantee.

Outside the primal-dual framework, the literature is rich on the use of the knapsack-cover inequalities and its generalizations together with rounding techniques. The problems considered include the Min-Sum General Scheduling problem on a single [3] and multiple [22, 2] machines, the Uniform [18] and Non-Uniform [19] Capacitated Multi-item Lot-sizing problem, and Capacitated Facility Location [1]. On the other hand, it is not known if there exists a compact set of constraints matching the strength of the knapsack-cover inequalities. Recently, Bazzi et al. [7] gave a formulation with an integrality gap of $2 + \varepsilon$ for the Knapsack-Cover problem with a quasipolinomial number of inequalities.

It is also worth mentioning that the most common technique for dealing with non-linear cost functions in capacitated covering problems is a doubling technique: split the cost function in segments where the function doubles. Then, each segment can be considered independently as a single item. This technique removes the precedence dependence between different segments, at factor 4 loss in the approximation ratio; see for example [3]. Similarly, with a randomized shifting strategy an e-approximation is achievable [17, 15]. Our approach strengthen the knapsack-cover inequalities and allows to avoid the extra loss.

Our Contribution

Let z_{ij} be a binary variable that represents whether $x_i \geq j$ in the solution, i.e., if the j-th unitary *segment* of item i is *taken*. Defining $g_{ij} = f_i(j) - f_i(j-1)$, then the cost of any solution is $\sum_{ij} g_{ij} z_{ij}$, and for any solution to be feasible it must hold for all i, j that if $z_{ij} = 1$ then $z_{i,j-1} = 1$. In a greedy algorithm, one might be tempted to take segments with low g_{ij}. This fails as such segments might be preceded by another segment with $g_{ik} \gg g_{ij}$ for $k < j$. This poses two fundamental questions when assessing the value of a segment: (i) how to take into consideration (mandatory) preceding segments of high cost? (ii) how to take into account low costs segments to the right, specially considering segments that finally might not be part of the final solution (since the demand can be completely covered by previous segments)?

We introduce a natural variant of the knapsack-cover inequalities for non-linear cost functions. These generalize the basic version of the inequalities, as well the generalization of Carnes and Shmoys [8] for the Single-Demand Facility Location Problem. Our inequalities are then used to derive a primal-dual algorithm that helps to handle the fundamental questions stated above. Our algorithm can be interpreted as a water-filling algorithm. Each segment j of an item i has a corresponding *bucket* B_{ij} of capacity g_{ij}, representing an inequality in the dual linear program. All buckets for a given item i are placed on a stairway, where bucket B_{ij} is on the j-th step of the stairs. A segment is taken, i.e. we set $z_{ij} = 1$, if its corresponding bucket and all previous ones (which are in lower steps of the stairs) are full. Water reaches buckets through two mechanisms. Water from an external source is poured directly into each bucket at a rate of either 1 or 0 (units of water per time unit). The

first time a bucket B_{ij} becomes full, then the water arriving to this bucket spills to bucket $B_{i,j-1}$, which now fills at a rate of 2 (as long as B_{ij} is still receiving water from the external source). If $B_{i,j-1}$ also becomes full and $j > 2$, then the water pouring into B_{ij} and $B_{i,j-1}$ spills to $B_{i,j-2}$ which now fills at a rate of 3, etc. For a bucket to receive water from the external source it must satisfy two properties: (i) its corresponding segment is not yet in the primal solution, and (ii) all previous segments of the item are not enough to cover the remaining demand. Our primal-dual algorithm helps to take care of the tensions implied by the questions above by making buckets filling faster due to water spilled from higher buckets, and prevents spilling water from a bucket if they are so high that they are useless to help covering the remaining demand.

For the case of Non-Linear UFP-Cover, our algorithm works similarly. However, the primal solution constructed with the algorithm might contain redundant segments due to sub-intervals of I that might be covered in subsequent steps of the algorithm. For this reason we need to perform a *reverse-delete* (or pruning) strategy to remove unnecessary segments, in the reverse order in which they were introduced in the primal solution.

We notice that, for both algorithms, our analysis is tight as they achieve the same performance guarantee as their classic variants [8, 10]. Additionally, the integrality gap of our formulation for the Non-Linear Knapsack-Cover problem is also 2, as the same lower bound of the the classical setting holds [9].

Finally, we show a rounding technique for the LP relaxation of the Non-Linear Knapsack-Cover problem and a polynomial time separation algorithm for the generalized knapsack-cover inequalities in the list model. These results, together with some of the proofs and extra details, can be found in the full version of this manuscript [13].

2 A Generalization of the Knapsack-Cover Inequalities for Non-Linear Knapsack-Cover

We first study the Non-Linear Knapsack-Cover problem. Recall that in this setting we consider a demand $D \in \mathbb{N}$ and a set N of n items, each with a non-decreasing function f_i. We assume that all f_i are defined over a common domain $\{0, 1, \dots, m\}$, for some $m \leq D$, and that $f_i(0) = 0$. Hence, each item $i \in N$ has m *segments* of unit length, indexed by a common set $M = \{1, \dots, m\}$, each having a unit cost $g_{ij} = f_i(j) - f_i(j-1) \geq 0$. In what follows we assume that our instance admits a feasible solution. We start by considering the list model.

It is worth mentioning that the problem described can be solved in polynomial time (respectively pseudo-polynomial) in the list (respectively oracle) model by a straightforward adaptation of the classical dynamic program for Knapsack. In the oracle model the problem is (weakly) NP-hard as it contains Knapsack-Cover as a special case. For this model the dynamic program can be turned into an FPTAS also by adapting well known rounding techniques [23]. However, these techniques alone cannot handle Non-Linear UFP-Cover.

2.1 Knapsack-Cover Inequalities for Non-Linear Costs

To write a linear relaxation of this problem, consider $a \in \{0, \dots, m\}^N$, where a_i represents that all segments $j \in \{1, \dots, a_i\}$ have been taken already for item $i \in N$ (and $a_i = 0$ represents that no segment of i is taken yet). We face the residual problem, where we must decide about segments not taken yet, and we must cover the residual demand $D(a) := \max\{D - \sum_{i \in N} a_i, 0\}$. Recall that z_{ij} is a variable that indicates whether the j-th segment of item i is taken. Since we must only cover the residual demand $D(a)$, there is an optimal solution where $z_{ij} = 0$ for $j > a_i + D(a)$; therefore, we conclude that for item i we can only take up to segment $m_i(a) := \min\{m, a_i + D(a)\}$.

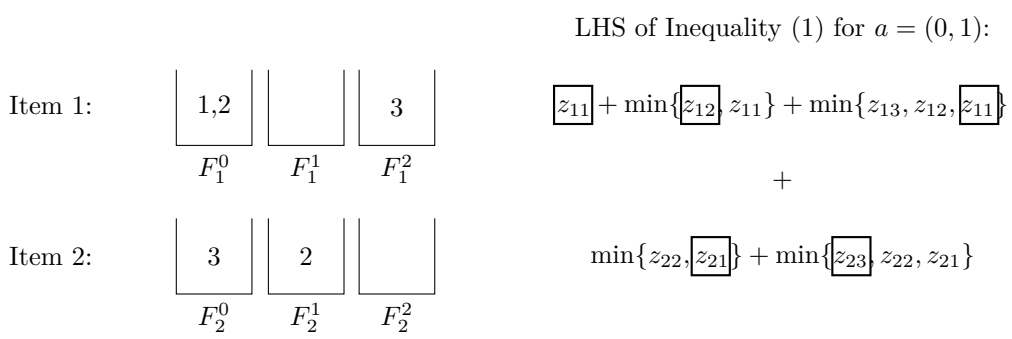

Figure 1 Example of an inequality in (1) for a given pair (a, F), with $n = 2$ items, $m = 3$ segments and demand $D = 4$. We consider a vector a with $a_1 = 0, a_2 = 1$, and hence $m_1(a) = m_2(a) = 3$. The left side of the figure shows a specific index F. On the right is shown the left-hand-side (LHS) of the convex inequality in (1), where we cover a term with a square if the corresponding term is chosen for the inequality in (2) indexed by (a, F). This yields the inequality $z_{11} + z_{12} + z_{11} + z_{21} + z_{23} \geq D(a)$, where $D(a) = 3$. That is, $2z_{11} + z_{12} + z_{21} + z_{23} \geq D(a)$, implying that $\tau(1, 1, a, F) = 2, \tau(1, 2, a, F) = \tau(2, 1, a, F) = \tau(2, 3, a, F) = 1$, and all the other $\tau(i, j, a, F)$ parameters equal 0.

To obtain a linear program, we relax the condition that $z_{ij} = 1$ implies $z_{ik} = 1$ for $k < j$. To do so, note that in a feasible solution variable z_{ij} should never be larger than $\min\{z_{ij}, z_{i,j-1}, \ldots, z_{i1}\}$, and hence we can replace in our formulation the appearance of z_{ij} by this minimum. We conclude that the following is a relaxation of the Non-Linear Knapsack-Cover problem, which we call [GKC]:

$$\begin{aligned}
&\min \sum_{i \in N, j \in M} g_{ij} z_{ij} \\
&\sum_{i \in N} \sum_{j=a_i+1}^{m_i(a)} \min\{z_{ij}, z_{i,j-1}, \ldots, z_{i1}\} \geq D(a) && \text{for all } a \in \{0, \ldots, m\}^N, \qquad (1) \\
&z_{ij} \geq 0 && \text{for all } i \in N, j \in M.
\end{aligned}$$

We call the set of inequalities (1) the *knapsack-cover inequalities for non-linear costs*. This relaxation can be easily linearized. Indeed, if a program has a constraint of the form $\min\{x_1, x_2\} \geq b$, then we can replace it with $x_1 \geq b$ and $x_2 \geq b$. More generally, if the constraint is $\min\{x_1, x_2\} + \min\{x_3, x_4\} \geq b$, then we must consider all constraints $x_i + x_j \geq b$ for all $i \in \{1, 2\}$ and $j \in \{3, 4\}$. More generally, convex inequality in (1) can be replaced with exponentially many linear ones. The linear inequalities are constructed by replacing each summand $\min\{z_{ij}, z_{i,j-1}, \ldots, z_{i1}\}$ in (1) by one of its terms $z_{ij}, z_{i,j-1}, \ldots, z_{i1}$.

Each of the new linear inequalities will be indexed by a pair (a, F). The chosen notation will prove useful when describing and analyzing the water-filling algorithm below, as they will indicate which buckets are spilling water and which are receiving it. Consider a fixed vector a as above. The index F indicates, for each item $i \in N$ and segment $j \in M$, which term z_{ip} is selected from the set $\{z_{ij}, \ldots, z_{i1}\}$ for each i, j. A given F can be thought as an array of containers $(F_i^k)_{i \in N, k \in \{0, \ldots, m-1\}}$. Each item $i \in N$ corresponds to a row of this array, with containers (sets) $F_i^0, \ldots, F_i^{m-1}$. We distribute the set $\{a_i + 1, \ldots, m\}$ within these containers, and thus $\bigcup_{k=0}^{m-1} F_i^k = \{a_i + 1, \ldots, m\}$. Assigning an index $j \in \{a_i + 1, \ldots, m\}$ to F_i^k represents that in that inequality we select $z_{i,j-k}$ from $\{z_{ij}, \ldots, z_{i1}\}$. See Figure 1 for a concrete example of this construction. The obtained set of inequalities is given by

$$\sum_{i\in N}\sum_{k=0}^{m-1}\sum_{j\in F_i^k: j\le m_i(a)} z_{i,j-k} \ge D(a) \qquad \text{for all } (a,F)\in\mathcal{F}, \tag{2}$$

where $\mathcal{F}$ is the set of all ordered pairs (a,F) with $a\in\{0,\dots,m\}^n$, and F satisfies (i) $\bigcup_{k=0}^{m-1} F_i^k = \{a_i+1,\dots,m\}$, and (ii) if $j\in F_i^k$ then $j-k\ge 1$ for all k,j. Let us consider now a given pair (a,F), an item i, and $j>a_i$. The term z_{ij} might appear several times in the respective constraint (2), depending on how many "minimums" are replaced by z_{ij}. If $k\in F_i^{k-j}$, then $\min\{z_{ik}, z_{i,k-1},\dots,z_{i1}\}$ is replaced by z_{ij}. Moreover, recall that the residual demand $D(a)$ will never be covered by a segment z_{ij} for $j>m_i(a)$, and hence the number of times that z_{ij} appears in the left-hand-side of the inequality is

$$\tau(i,j,a,F) = |\{k\ge j : k\in F_i^{k-j}, k\le m_i(a)\}|. \tag{3}$$

For a concrete example consider Figure 1.

With this definition, we obtain a linear relaxation that is equivalent to [GKC].

▶ **Lemma 1.** *The convex program [GKC] is equivalent to*

$$\begin{aligned} \textit{[P-GKC]:}\ \min & \sum_{i\in N}\sum_{j\in M} z_{ij}g_{ij} \\ s.t. & \sum_{i\in N}\sum_{j=1}^{m}\tau(i,j,a,F)\cdot z_{ij} \ge D(a) \qquad \textit{for all } (a,F)\in\mathcal{F}, \\ & z\ge 0. \end{aligned} \tag{4}$$

A routinary computation yields that the dual of this linear program, which we call [D-GKC], is the problem of maximizing $\sum_{(a,F)\in\mathcal{F}} D(a)v_{aF}$ subject to $v\ge 0$ and

$$\sum_{(a,F)\in\mathcal{F}} \tau(i,j,a,F)\cdot v_{aF} \le g_{ij} \quad \text{for all } i\in N, j\in M. \tag{5}$$

2.2 A 2-approximate Primal-Dual Algorithm

We provide a primal-dual 2-approximation algorithm based on the LP-relaxation [P-GKC] and its dual [D-GKC]. It is worth having in mind the bucket representation of the algorithm given above in the introduction.

Algorithm description

The water-filling algorithm described in the introduction is an intuitive representation of a greedy algorithm for the dual [D-GKC]. Each bucket corresponds to a dual inequality: Each of the inequalities in the dual [D-GKC] represents a bucket, the left hand size corresponds to the amount of water in the bucket, while the right hand side is its capacity. The greedy dual algorithm raises dual variables one by one starting from a dual solution $v\equiv 0$. In each iteration of the main loop, we raise a variable v_{aF}. The index a is chosen such that a_i represents the largest value ℓ for which all buckets $B_{i1},\dots,B_{i\ell}$ are full (or equivalently, $z_{i1}=\dots=z_{i\ell}=1$), for each $i\in N$. To choose F, a segment j will belong to F_i^k if and only if the water from the external source falling into bucket B_{ij} (if any) spills down to bucket $B_{i,j-k}$. Number k is chosen such that it is the smallest number for which $B_{i,j-k}$ is not full, representing the idea that the water of full buckets falls down to the previous buckets on the stairs. Also, buckets receiving water from the external source are the buckets B_{ij} with

$a_i + 1 \leq j \leq m_i(a)$. This way, $\tau(i, j, a, F)$ corresponds to the filling rates of bucket B_{ij} in the current iteration, which considers the water directly from the external source and the water spilled from higher buckets. We stop raising variable v_{aF} as soon as one dual inequality becomes tight, i.e., some bucket B_{ij} becomes full. After, we update the value of z by setting $z_{ik} = 1$ if $B_{i\ell}$ is full for all $\ell \leq k$. Notice as we do not require that $k \leq m_i(a)$, and hence the returned primal solution might not satisfy the total demand exactly. Finally, we update a and F as described above and repeat the main loop until the residual demand $D(a)$ equals 0. The pseudo-code is given in Algorithm 1. The algorithm calls a sub-routine (Line 14) that explains how to update F when a bucket gets full but cannot be taken, which is given in Algorithm 2.

Algorithm 1 Primal-Dual Water-Filling Algorithm for Non-Linear Knapsack-Cover.

1: $z, v \leftarrow 0$; *% primal and dual solutions.*
2: $a \leftarrow 0$; *% a_i represents the largest value for which buckets $B_{i1}, \ldots, B_{i,a_i}$ are full.*
3: $F_i^k \leftarrow \emptyset \ \ \forall i \in N, k \in M$;
4: $F_i^0 \leftarrow M$; *% $j \in F_i^k$ iff water from bucket B_{ij} falls to bucket $B_{i,j-k}$.*
5: **while** $D(a) > 0$ **do**
6: Increase v_{aF} until a dual constraint indexed by (i, j), for some item $i \in N$ and segment $j \in M$, becomes tight. *% Bucket B_{ij} becomes full.*
% Update z_{ij}:
7: **if** $j = a_i + 1$ **then**
8: Let $q > a_i$ be the maximum number such that $B_{i,a_i+1}, \ldots, B_{iq}$ are full.
9: **for** $\ell = j, \ldots, q$ **do**
10: $z_{i\ell} \leftarrow 1$; *% Take available segments.*
11: **end for**
12: $a_i \leftarrow q$;
13: **else** *% If we cannot raise variable z_{ij}:*
14: $F \leftarrow \text{Update}(F, i, j, a)$ *% Call Algorithm 2 to update the sets F*
15: **end if**
16: **end while**
17: **return** v, z.

Algorithm 2 Updating Buckets Subroutine

input a, i, j, F *% a, F represent the current state of the buckets, i, j represent which is the bucket that just got full. We require $j > a_i + 1$.*
Let $p < j$ be the maximum number so that B_{ip} is not full.
Let $q > j$ be the minimum number so that B_{iq} is not full (and $q = m + 1$ if $B_{ij}, \ldots, B_{im}$ are all full).
for $\ell = j, j + 1, \ldots, q - 1$ **do**
$F_i^{\ell-j} \leftarrow F_i^{\ell-j} \setminus \{\ell\}$ and $F_i^{\ell-p} \leftarrow F_i^{\ell-p} \cup \{\ell\}$ *% The water from the external source falling to $B_{i\ell}$, which was previously spilling to bucket B_{ij}, now spills to bucket B_{ip}.*
end for
return F

Analysis

The algorithm terminates, as each iteration of the main loop (Line 5) corresponds to some bucket that becomes full, so we enter the while loop at most nm times. The main challenge is to show that the algorithm is 2-approximate.

It follows directly that the dual solution constructed is feasible through the execution of the algorithm. The primal solution is feasible as we kept iterating the main loop until the residual demand $D(a)$ is zero. As in most approximate primal-dual algorithms, the crux of the analysis is to show that an approximate form of the complementary slackness conditions are satisfied. This is summarized in the next key lemma.

▶ **Lemma 2.** *Let $\bar{v}, \bar{z}$ be the primal and dual solutions computed by the algorithm. Then,for all $(a, F) \in \mathcal{F}$ such that $\bar{v}_{aF} > 0$, it holds that*

$$\sum_{i \in N} \sum_{j=1}^{m} \tau(i, j, a, F) \bar{z}_{ij} \leq 2D(a).$$

Proof. Let $(\bar{\imath}, \bar{\jmath})$ be the indices of the dual inequality that became tight in the last iteration of the algorithm, and let $\bar{z}, \bar{v}$ be the output primal and dual solutions.

Let us fix a variable $v_{aF} > 0$ and consider the iteration of the main loop of the algorithm where we were raising that variable. For a given item $i \in N$, the expression $\sum_{j \geq 1} \tau(i, j, a, F) \bar{z}_{ij}$ represents the total number of buckets that are receiving water from the external source and whose water is spilling to some bucket that ends up in the final solution. We analyze the last item separately from the other items. Regarding item $\bar{\imath}$, notice that $\sum_{j \geq 1} \tau(\bar{\imath}, j, a, F) \bar{z}_{\bar{\imath}j} \leq D(a)$, just because the buckets obtaining water from the external source are in the interval $a_{\bar{\imath}} + 1, \ldots, m_{\bar{\imath}}(a)$, which are at most $D(a)$ many.

Consider now $i \neq \bar{\imath}$ and a bucket B_{ij} that is "part"of $\tau(i, j', a, F)$ for some segment j' included in the final solution, that is, either $j' = j$ or B_{ij} is pouring into $B_{ij'}$, case in which all the buckets between j and j' are full in this iteration of the algorithm. Then, as $\bar{z}_{ij'} = 1$, by construction B_{ij} will be taken as well; that is, $\bar{z}_{ij} = 1$. Additionally, no water (either directly or indirectly) reaches a bucket B_{ik} with $k \leq a_i$, and hence $\tau(i, k, a, F) = 0$. So the quantity $\sum_{i \in N \setminus \{\bar{\imath}\}} \sum_{j \geq 1} \tau(i, j, a, F) \bar{z}_{ij}$ is upper bounded by the total number of buckets in the final solution, of items other than $\bar{\imath}$, that are above a. This number cannot be higher than $D(a)$, otherwise the algorithm would have finished before filling the last bucket $B_{\bar{\imath},\bar{\jmath}}$. ◀

The proof of the next theorem follows from the previous lemma and standard techniques from primal-dual analysis. The details are deferred to the full version.

▶ **Theorem 3.** *Algorithm 1 is a polynomial time 2-approximation algorithm for Non-Linear Knapsack-Cover in the list model.*

The extension of this theorem to the oracle model is given in Section 4.

3 Unsplittable Flow-Cover on the Line

We now show that extending the ideas of Section 2 we can also achieve a 4-approximation for the Non-Linear UFP-Cover problem. Recall that an instance of this problem is given by an interval $I = \{1, \ldots, k\}$, a set N of n items, where each item is characterized by a capacity or height u_i, a cost c_i, and a sub-interval $I_i \subseteq \{1, \ldots, k\}$. We also have a demand D_t for each $t \in I$.

In the non-linear case, we can choose the height of each item from within a set $M = \{1, \ldots, m\}$. In other words, each item consists of a list of (vertical) segments, and one must choose a prefix of them. Again, we first focus on the list model where the costs of segments $g_{i1}, g_{i2}, \ldots, g_{im}$ are given in a list. As before, we use variables $z_{ij} \in \{0, 1\}$ to represent if segment j of item i is in the solution. With this the cost is $\sum_{i \in N} \sum_{j \in M} z_{ij} g_{ij}$ and the constraint that each demand must be covered is given by

$$\sum_{i \in N : t \in I_i} \sum_{j \in M} z_{ij} \geq D_t \text{ for all } t \in I. \tag{6}$$

Finally, we require that $z_{i,j+1} = 1 \Rightarrow z_{ij} = 1$ for all $i \in N, j \in \{1, \ldots, m-1\}$.

We now present the problem using the generalized knapsack-cover inequalities, and the relaxation explained in Section 2, applied to each inequality in (6) separately. For this, define $D_t(a) = \max\left(D_t - \sum_{i : t \in I_i} a_i, 0\right)$ and $\tau(i, j, a, F, t) = \{k \geq j : k \in F_i^{k-j}, k \leq m_i(a)\}$, where $m_i(a) = \min\{m, a_i + D_t(a)\}$. The relaxed primal problem, which we call [P-UFP], asks to minimize $\sum_{i,j} z_{ij} g_{ij}$ for $z \geq 0$ subject to

$$\sum_{i : t \in I_i} \sum_{j \geq 1} \tau(i, j, a, F, t) \cdot z_{ij} \geq D_t(a) \quad \text{for all } (t, a, F) \in \mathcal{H}$$

where $\mathcal{H}$ is the set of triplets (t, a, F) where $t \in I$ and $(a, F) \in \mathcal{F}$, as defined in Section 2.2. This yields a dual relaxation [D-UFP], whose objective is $\max \sum_{(a,t,F) \in \mathcal{H}} v_{atF} \cdot D_t(a)$, and we must optimize over all $v \geq 0$ satisfying

$$\sum_{(a,t,F) \in \mathcal{H} : t \in I_i} \tau(i, j, a, F, t) \cdot v_{atF} \leq g_{ij} \quad \text{for all } i \in N, j \in M. \tag{7}$$

Algorithm Description

Our primal-dual algorithm is given in Algorithm 3. In this case our approach has two phases. During the *growing* phase (Lines 5–15), we construct a dual solution, which then directly implies a feasible primal solution. In the *pruning* phase (Lines 16–20) we remove unnecessary segments from the primal solution. As before, for each item $i \in N$ we have a stair of buckets, where each bucket B_{ij} corresponds to a given inequality in the dual, indexed by $j \in M$ and $i \in N$. In each iteration of the growing phase buckets receive water (that might fall to inferior buckets) from an external source at a rate of 1 or 0. Once we define the rates, the water dynamics work in exactly the same way as in Section 2.2: water reaching a given bucket that is full is spilled to the next bucket to the left until it reaches a bucket that is not full. The only difference is that only some of the items receive water. More precisely, in a given iteration of the growing phase, we select $t \in I$ with largest unsatisfied demand $D_t(a)$ (break ties arbitrarily). This is a greedy criterion to increase the dual objective function as fast as possible. Only buckets for items $i \in I$ such that $t \in I_i$ receive water from the external source. For such an item i, the subset of buckets receiving water from the external source are again buckets B_{ij} with $j \in \{a_i + 1, \ldots, m_i(a)\}$. The water dynamics can be emulated by raising a single dual variable at a time. Notice that the only difference to the dual in Section 2 is that when raising a given variable v_{atF}, only inequalities for items $i \in N$ where $t \in I_i$ are affected, corresponding to the fact that only buckets corresponding to such items receive water from the external source.

When one or more buckets become full, we pick one of these buckets. As before, a full bucket means that the corresponding segment (i, j) is available. We take a given segment, that is, we define a primal variable z_{ij} to 1, as soon as all preceding buckets of item i are available.

In other words, if B_{ij} becomes full for $j = a_i + 1$, then we take set $z_{ij} = \ldots = z_{iq} = 1$ where $B_{ij}, \ldots, B_{iq}$ are full but $B_{i,q+1}$ is not. This is the case even if $q > m_i(a)$. All taken buckets (or segments) are considered to be a "block", denoted by b_{ij} (where j denotes the first segment of the block). After this we update t and continue with a new iteration of the main loop of the growing phase.

Although this first phase gives a feasible primal solution, some blocks might have become redundant, that is, the solution would remain feasible without them. In the second phase we remove redundant blocks when we can. To do this, we check for each block b_{ij}, in reverse order in which they were added, whether removing the block from the primal solution makes the given primal unfeasible. If b_{ij} is redundant and it is the superior block (i.e., the block containing the highest segment that is still in the solution) of its item, we remove it; if b_{ij} is redundant but there are blocks over it in the solution when it is checked, we cannot remove it (the solution would become unfeasible). Note that doing so, all the superior blocks that are in the final solution are not redundant.

Algorithm 3 Primal-Dual Algorithm for Non-Linear UFP-Cover.

$z, v \leftarrow 0$ *% primal and dual solutions.*
$a \leftarrow 0$
$F_i^k \leftarrow \emptyset$ for all $i, k \geq 1$; $F_i^0 \leftarrow M$
$\mathcal{B}_i \leftarrow \emptyset$ for all i *% Set containing the blocks of item i.*
while $D_t(a) > 0$ for some t **do**
  Select t that maximizes $D_t(a)$ (break ties arbitrarily).
  Increase v_{atF} until a dual constraint indexed by (i, j), for some item i and segment j, becomes tight. Break ties in favor of a bucket with smallest index j.
  if $j > a_i + 1$ **then**
    $F \leftarrow \text{Update}(F, i, j, a)$ *% Water pouring into B_{ij} pours into a lower bucket.*
  else
    Let $q > a_i$ be the maximum number such that $j' \notin F_i^0$ for all $j' = j + 1, \ldots, q$ *% we take all full buckets that poured into B_{i,a_i+1}, even the ones that are now truncated.*
    Set $a_i \leftarrow q$
    Set $\mathcal{B}_i \leftarrow \mathcal{B}_i \cup b_{ij}$, with $b_{ij} = \{j, \ldots, q\}$ *% b_{ij} is a block that enters the primal solution.*
  end if
end while
for all b_{ij} in reversed order in which they are defined in the growing phase **do**
  if b_{ij} can be removed from the primal solution without leaving any demand unsatisfied **and** $j \geq j'$ for all j' such that $b_{ij'} \in \mathcal{B}_i$ **then**
    % We eliminate redundant blocks, unless they have a superior block over it
    $\mathcal{B}_i \leftarrow \mathcal{B}_i \setminus b_{i,j}$
  end if
end for
return z, where $z_{ij} = 1$ for $j \leq \sum_{b_{ik} \in \mathcal{B}_i} |b_{ik}|$.

The proof of correctness is analogous as in Section 2.2. To show the approximation factor we need the following key lemma.

▶ **Lemma 4.** *Consider the output (z, v) of the algorithm. Let (a, t, F) such that $v_{atF} > 0$. Then*

$$\sum_{i:t \in I_i} \sum_{j \geq 1} \tau(i, j, a, F, t) \cdot z_{ij} \leq 4 D_t(a).$$

Proof. Let us fix a variable $v_{atF} > 0$ raised in the main loop of the growing phase. Denote by $\mathcal{B}_i$ the set of blocks for each i at the end of the pruning phase. Out of those, consider the ones that are above a, and that contribute to fulfill the demand in t, that is, $S_{ta} = \{b_{ij} \in \cup_{i \in N} \mathcal{B}_i : t \in I_i, j \geq a_i + 1\}$. Let us denote by $\overline{b}_i$ the superior block of each item (i.e. $\overline{b}_i = b_{ij}$ with $b_{ij} \in S_{ta}$ and $j \geq j'$ for all j' such that $b_{ij'} \in \mathcal{B}_i$). For each of these superior blocks, as they were not removed, it must exist some $t_i \in I$ such that its demand would become unsatisfied when removing $\overline{b}_i$, which is of course also true if we look only at the blocks in S_{ta}, i.e.,

$$D_{t_i}(a) > \left(\sum_{b_{kj} \in S_{ta}} |b_{kj}| \right) - |\overline{b}_i|. \tag{8}$$

Inequality (8) is true because we removed the blocks in a reversed order, and the blocks that conform a were introduced before $\overline{b}_i$ in the growing phase. Let us classify the blocks in S_{ta} into two subsets, $S^L_{ta} = \{b_{i\ell} \in S_{ta} : t_i \leq t\}$ and $S^R_{ta} = \{b_{i\ell} \in S_{ta} : t_i > t\}$.

We divide the proof of Lemma 4 into two analogous inequalities. Let us show that

$$\sum_{i : t \in I_i} \sum_{j : b_{ij} \in S^R_{ta}} z_{ij} \tau(i, a, t, F) \leq 2 D_t(a). \tag{9}$$

To do this, define $t^R = \min\{t_i : \overline{b}_i \in S^R_{ta}\}$. Note that t^R is covered by every interval I_i with $\overline{b}_i$ in S^R_{ta}, as they cover t (which is at most t^R) and their t_i (which is larger or equal than t^R). Define (i_1, j_1) such that $t_{i_1} = t^R$ and $\overline{b}_{i_1} = b_{i_1, j_1}$. On the one hand, by definition of τ, we have that $z_{i_1,j_1} \tau(i_1, j_1, a, t, F) \leq \tau(i_1, j_1, a, t, F) \leq D_t(a)$. On the other hand we study

$$\sum_{i : t \in I_i} \sum_{\substack{j : B_{ij} \in S^R_{ta}, \\ (i,j) \neq (i_1, j_1)}} z_{ij} \tau(i, j, a, F, t).$$

Consider an item $i \neq i_1$, and the iteration while increasing variable v_{atF}. The summands are the number of buckets that were spilling over each of the segments above a_i that are in the final solution (because we only sum when $z_{ij} = 1$, and buckets B_{ik} for $k \leq a_i$ do not receive water). This quantity cannot be higher than the sum of the cardinality of all the blocks above a_i in the final solution (recall that when blocks are taken in Line 11, they include truncated buckets that have poured onto the taken segments). For i_1, the same argument holds, but the superior block $\overline{b}_{i_1}$ does not need to be considered because it never poured onto the inferior blocks (otherwise they would have been the same block). Thus it holds that

$$\sum_{i : t \in I_i} \sum_{\substack{j : b_{ij} \in S^R_{ta}, \\ (i,j) \neq (i_1, j_1)}} z_{ij} \tau(i, j, a, F, t) \leq \sum_{\substack{b_{ij} \in S^R_{ta}, \\ (i,j) \neq (i_1, j_1)}} |b_{ij}| \leq D_{t^R}(a) \leq D_t(a). \qquad \blacktriangleleft$$

With this lemma we can show the following main results. The first proof follows from standard LP techniques which are completely analogous to the proof of Theorem 3. The extension to the oracle model is given in Section 4.

▶ **Theorem 5.** *Algorithm 3 is a polynomial 4-approximation algorithm for Non-Linear UFP-Cover in the list model.*

4 Arbitrary non-decreasing functions

We now show how to adapt our algorithms in Sections 2 and 3 for the more general oracle model. Here we assume that each function $f_i : \{0, \dots, m\} \to \mathbb{Q}_{\geq 0} \cup \{\infty\}$, where $m \leq D$ is not necessarily polynomially bounded. We assume that each function f_i is given by an oracle, such that a polynomial number of bits is enough to describe all values $f_i(x)$. Using standard techniques, we first approximate each function f_i by a piece-wise constant function with a polynomial number of steps. After, we discuss how to emulate Algorithm 1 and 3 in polynomial time for such functions.

First of all, by scaling we can assume, without loss of generality, that $f_i(x) \in \mathbb{N} \cup \{\infty\}$.

▶ **Lemma 6.** *Consider $\varepsilon > 0$ and let $f : \{1, \dots, m\} \to \mathbb{N} \cup \{\infty\}$ be a non-decreasing function. Let $f_{\max}$ be the maximum finite value of $f(x)$. There exists a non-decreasing piece-wise constant function $\tilde{f}$ such that*

$$f(x) \leq \tilde{f}(x) \leq (1+\varepsilon) f(x) \qquad \text{for all } x \in \{1, \dots, m\},$$

where $(\tilde{f}(x))_{x \in \mathbb{N}}$ takes at most $\lceil \log_{1+\varepsilon} f_{\max} \rceil + 1$ many different values. The function $\tilde{f}$ can be computed in polynomial time.

Proof. To prove the lemma we can assume that $f(x) > 0$, as the values $f(x) = 0$ just corresponds to separate piece in $\tilde{f}$. For all other $x \in \{1, \dots, m\}$, we can simply set $\tilde{f}(x) = (1+\varepsilon)^{\lceil \log_{1+\varepsilon} f(x) \rceil}$. The constant-wise pieces (intervals) of $\tilde{f}$ can be easily computed in polynomial time with a binary search approach. ◀

We now sketch how to adapt the algorithms of Sections 2 and 3 for this scenario. Let $\tilde{f}_i$ be the obtained function after applying the last lemma to f_i. We partition the set $\{0, 1, \dots, m\}$ in intervals $J_{i1}, J_{i2}, \dots, J_{im_i}$ correspondent to the piece-wise constant pieces of $\tilde{f}_i$, that is $\tilde{f}_i(x) = \tilde{f}_i(x')$ if $x, x' \in J_{ik}$. We denote by $u_{ik} \in \mathbb{N}$ the cardinality of interval $J_{ik} \subseteq \mathbb{N}$. To adapt the algorithms, note that as they originally deal with unitary segments, a piecewise constant function can be replaced (preserving the same costs for any solution) by a piecewise constant function with a pseudopolynomial number of unitary segments. More precisely, if $J_{ik} = \{\ell, \ell+1, \dots, u\}$, then $g_{i\ell} = \tilde{f}_i(\ell) - \tilde{f}_i(\ell-1)$, and $g_{ir} = 0$ for all $r \in \{\ell+1, \dots, u\}$. Applying our algorithms to this instance would imply a pseudopolynomial running time. However, as all but m_i many buckets for item i has zero capacity g_{ij}, we can handle all of them simultaneously to make our algorithms run in polynomial time.

To do this, we can process all segments in J_{ik} in a single step: when the algorithm begins, all their respective buckets but the first one get instantaneously full, so the first one will receive water at a rate equal to the length of the constant interval (equivalently, the interval J_{ij} is represented by a bucket of height g_{ij} that gets filled at a rate u_{ij}). Any time a bucket pours onto some inferior bucket, its rate also increases by the length of the interval corresponding to the pouring bucket. Truncations, given by the fact that in a given iteration only buckets B_{ij} for $j \in \{a_i + 1, \dots, m_i(a)\}$ get water from the external source for each item i, make these rates diminish accordingly. With these rules, the algorithms can be easily adapted to run in polynomial time implying the following theorems.

▶ **Theorem 7.** *There exists a polynomial time $(2+\varepsilon)$-approximation for the Knapsack-Cover Problem with Non-Linear Costs and arbitrary non-decreasing functions.*

▶ **Theorem 8.** *There exists a $(4+\varepsilon)$-approximation for the Unsplittable Flow-Cover on the Line Problem with Non-Linear Costs and arbitrary non-decreasing functions.*

References

1 H.-C. An, M. Singh, and O. Svensson. LP-based algorithms for capacitated facility location. *SIAM Journal on Computing*, 46(1):272–306, 2017.

2 N. Bansal and J. Batra. Geometry of scheduling on multiple machines. *arXiv:1907.05473 [cs]*, 2019. arXiv:1907.05473.

3 N. Bansal and K. Pruhs. The geometry of scheduling. *SIAM Journal on Computing*, 43(5):1684–1698, 2014.

4 A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.

5 R. Bar-Yehuda and S. Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.

6 R. Bar-Yehuda and D. Rawitz. On the equivalence between the primal-dual schema and the local ratio technique. *SIAM Journal on Discrete Mathematics*, 19(3):762–797, 2005.

7 A. Bazzi, S. Fiorini, S. Huang, and O. Svensson. Small extended formulation for knapsack cover inequalities from monotone circuits. *Theory of Computing*, 14(1):1–29, 2018.

8 Tim Carnes and David B. Shmoys. Primal-dual schema for capacitated covering problems. *Mathematical Programming*, 153(2):289–308, 2015.

9 R. D. Carr, L. K. Fleischer, V. J. Leung, and C. A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the 11th annual ACM-SIAM symposium on Discrete algorithms (SODA 2000)*, pages 106–115, 2000.

10 M. Cheung, J. Mestre, D. B. Shmoys, and J. Verschae. A primal-dual approximation algorithm for min-sum single-machine scheduling problems. *SIAM Journal on Computing*, 31(2):825–838, 2017.

11 V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

12 N. Coulombel and N. Monchambert. Diseconomies of scale and subsidies in urban public transportation. *HAL:02373768*, 2019.

13 A. Fielbaum, I. Morales, and J. Verschae. A water-filling primal-dual algorithm for approximating non-linear covering problems. *arXiv:1912.12151 [cs.DS]*, 2019. arXiv:1912.12151.

14 Andrés Fielbaum, Sergio Jara-Diaz, and Antonio Gschwender. Beyond the Mohring effect: Scale economies induced by transit lines structures design. *Economics of Transportation*, 22:100–163, 2020.

15 M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82(1):111–124, 1998.

16 W. Höhn, J. Mestre, and A. Wiese. How unsplittable-flow-covering helps scheduling with job-dependent cost functions. In *Automata, Languages, and Programming (ICALP 2014)*. Springer, 2014.

17 M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–79, 1996.

18 Retsef Levi, Andrea Lodi, and Maxim Sviridenko. Approximation algorithms for the capacitated multi-item lot-sizing problem via flow-cover inequalities. *Mathematics of Operations Research*, 33(2):461–474, 2008.

19 S. Li. Constant approximation algorithm for non-uniform capacitated multi-item lot-sizing via strong covering inequalities. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2311–2325, 2017.

20 S. Thomas McCormick, Britta Peis, J. Verschae, and A. Wierz. Primal-dual algorithms for precedence constrained covering problems. *Algorithmica*, 78(3):771–787, 2017.

21 Herbert Mohring. Optimisation and scale economies in urban bus transport. *American Economic Review*, 62(4):591–604, 1972.

22 B. Moseley. Scheduling to approximate minimization objectives on identical machines. In *Automata, Languages, and Programming (ICALP 2019)*, pages 86:1–86:14, 2019.

23 D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 1 edition, 2011.

24 J. Zhu. *Optimization of power system operation*, volume 47. John Wiley & Sons, 2015.