On Improved Interval Cover Mechanisms for Crowdsourcing Markets

Evangelos Markakis, Georgios Papasotiropoulos, Artem Tsikiridis

Athens University of Economics and Business

Abstract. We study a covering problem motivated by spatial models in crowdsourcing markets, where tasks are ordered according to some geographic or temporal criterion. Assuming that each participating bidder can provide a certain level of contribution for a subset of consecutive tasks, and that each task has a demand requirement, the goal is to find a set of bidders of minimum cost, who can meet all the demand constraints. Our focus is on truthful mechanisms with approximation guarantees against the optimal cost. To this end, we obtain two main results. The first one, is a truthful mechanism that achieves a bounded approximation guarantee. This mechanism improves the state of the art, which is a mechanism with an arbitrarily large factor in worst case. The second one, concerns a class of instances that generalizes the minimum knapsack problem. Namely, we consider inputs with a constant number of tasks, for which we provide a truthful FPTAS. Finally, we also highlight connections of our problem with other well-studied optimization problems, out of which, we infer further conclusions on its (in)approximability.

1 Introduction

We consider a mechanism design problem, that emerges under certain spatial models for crowdsourcing and labour matching markets. It is instructive to start with an example, so as to introduce the main aspects of the model. Imagine a set of cities, located geographically in a consecutive order, and consider a company that has opened a store in each of these cities. In Figure 1, we see an instance with 5 cities, named A to E. The company needs to meet a demand constraint, i.e., a lower bound on the volume of goods that need to be transported to each store, based on consumption and future planning. To achieve this goal, it chooses to hire other firms, or single individuals, that can make deliveries, via a reverse (procurement) auction (for an exposition on auctions for transportation routes, see [15]). Suppose that every participating entity, referred to as a bidder or a worker, can only cover a certain interval of contiguous cities, at a cost that she specifies, and furthermore, she can only accommodate a certain volume of goods, assumed to be the same for each city in her declared interval (among others, dependent on the transportation means that she owns). The problem boils down to selecting a set of winning bidders who can jointly cover all the demand constraints at minimum cost, and in a way that prevents the workers from misreporting their true preferences.

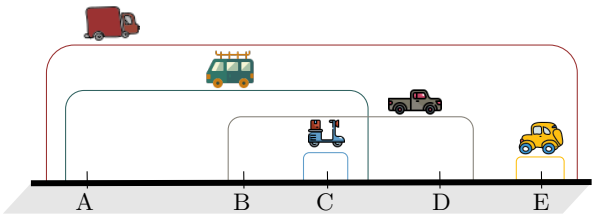

Fig. 1: An example with 5 tasks (A to E) and 5 workers.

In the crowdsourcing jargon, we can view the store in each city as a task with a demand requirement. For instance, in labeling/classification tasks, the demand could correspond to the number of people who should execute the task in order to acquire a higher confidence on the outcome. In other cases, it could be interpreted as a (not necessarily integral) volume coverage requirement.

vehicle	contribution	cost
truck	8	10
van	6	5
pickup	5	4
motorbike	1	2
car	3	4

Table 1: Contribution and costs of the workers of Figure 1.

Coming back to the example of Figure 1, say that the demand of cities A to E are given by the vector $(6, 2, 9, 1, 3)$. The contribution per task, as well as the cost of the workers are given by Table 1. The interval covered by each worker is visible in Figure 1. Obviously, hiring all the workers is a feasible solution. Selecting the workers with the van, the pickup and the car also forms a feasible solution since the demands of all cities are indeed satisfied. Notice that the optimal solution is to hire the workers with the truck and the motorbike at a total cost of 12.

From an algorithmic viewpoint, there has been already significant progress on the problem. As it can be easily seen to be NP-hard, the main results on this front include constant factor approximation algorithms, with further improvements for special cases. However, in the context of mechanism design, one needs to consider strategic aspects as well. Bidders may attempt to report higher costs in order to achieve higher payments, or they can lie about the subset of tasks they can actually fulfil. It would be therefore desirable to have mechanisms that, on the one hand, achieve competitive approximation guarantees, and on the other, deter bidders from misreporting. To our knowledge, the currently best result on this direction is by [18], where however the approximation ratio of their truthful mechanism is unbounded in terms of the number of tasks and workers (and depends on numerical parameters of each instance). It has remained open since then, if truthfulness can be compatible with bounded approximation guarantees.

Contribution

We focus on truthful mechanisms and their approximation guarantees against the optimal cost. To this end, we provide two main results. The first one, in Section 3, is a truthful Δ-approximation mechanism, where Δ is the maximum number of workers that are willing to work on the same task (the maximum

being taken over all tasks). This mechanism, improves significantly the state of the art, coming to a guarantee that is polynomially bounded in terms of the input size. Apart from the improvement, we note also that our result is based on the local-ratio technique from approximation algorithms [4], a technique that has not been used very often for building truthful mechanisms (for an exception, see [20]). Moving on, our second result, in Section 4, concerns the class of instances with a constant number of tasks, which generalizes MIN-KNAPSACK, an NP-hard variant of KNAPSACK that corresponds to the single-task case of our problem. For this class we provide a truthful FPTAS, by mainly exploiting and adapting the framework of [8]. In doing so, we also identify a flaw in a previous attempt for designing a truthful FPTAS for MIN-KNAPSACK. Finally, in Section 5, we discuss some further implications and extensions. Namely, we bring to light some interesting connections with well-studied optimization problems, related to unsplittable flow problems and caching. By exploiting known results for these problems, we rule out a FPTAS for the general case, and also identify a special case where a PTAS is likely to exist.

Related Work

As already discussed, the work most related to ours is [18], which introduced the model in the context of crowdsourcing. They provide a truthful optimal mechanism when the workers have an identical contribution, whereas for the general case, they present a truthful approximate mechanism, the ratio of which is dependent on the contribution parameters of the workers. Furthermore, [41] studies the case of unit-demand tasks and unit-contribution workers and identifies a truthful optimal mechanism. For the same setting, they also propose a mechanism when workers can submit multiple bids, for which they attain a logarithmic factor. Additionally, in [42] the authors studied the prominent special case of a single task (MIN-KNAPSACK) and provided a randomized, truthful-in-expectation mechanism that achieves an approximation factor of 2.

Regarding the purely algorithmic problem, without the constraint of truthfulness, it has appeared under the name of (0-1)RESOURCE ALLOCATION, and a 4-approximation was presented in [10]. The currently best known algorithm achieves a factor of 3, in [33]. For the MIN-KNAPSACK problem, a PTAS is implied by [17] and a FPTAS is given in [29].

There are quite a few problems that can be viewed as generalizations of what we study here, such as *general scheduling problem* [2], *multidimensional min-knapsack*, *column restricted covering integer programs* [11]. Moreover, several problems in discrete optimization can be seen as related variants, but are neither extensions nor special cases of ours. Indicatively: *bandwidth allocation* [12], *multiset multicover*[7,38], *geometric knapsack* [23], *capacitated network design* [9], *admission control* [36].

Finally, for general spatio-temporal models appearing in the crowdsourcing literature, we refer to two recent surveys [25,39], which cover to a big extent the relevant results.

2 Preliminaries

In this section, we first define formally the problem that we study, together with some additional necessary notation. In the sequel, we discuss the relevant definitions for the design of truthful mechanisms. We note that all the missing proofs from the following sections can be found in the full version of our work.

Problem Statement

We are interested in the optimization problem defined below. For motivating applications, we refer the reader to Section 1.2 in [18].

Cost Minimization Interval Cover (CMIC): Consider a set of tasks, say $\{1, \ldots, m\}$, that are ordered in a line, and a set of available bidders $\mathcal{N} = \{1, \ldots, n\}$. We will interchangeably use the term bidder or worker in the sequel. An instance of CMIC is determined by a tuple $(\mathbf{b}, \mathbf{q}, \mathbf{d})$, where:

- The vector $\mathbf{b} = (b_1, \ldots, b_n)$ is the bidding profile. For each bidder $j \in \mathcal{N}$, $b_j = (c_j, [s_j, f_j])$, where $[s_j, f_j] = I_j$ is the interval of contiguous tasks $\{s_j, s_j + 1, \ldots, f_j\} \subseteq [m]$, that j is able to contribute to, and $c_j \in \mathbb{R}_{\geq 0}$ is the cost incurred to her, if she is selected to contribute. We can assume positive costs since workers of zero cost are trivially included in the solution. We often denote $\mathbf{b}$ as $(\mathbf{c}, \mathbf{I})$ where $\mathbf{c}$ is the cost vector and $\mathbf{I}$ is the vector of the intervals.
- The vector $\mathbf{q} = (q_1, \ldots, q_n)$ is the contribution vector, so that $q_j \in \mathbb{R}_{>0}$ denotes the contribution that bidder j can make to the tasks that belong to $[s_j, f_j]$.
- The vector $\mathbf{d} = (d_1, \ldots, d_m)$ specifies the demand $d_j \in \mathbb{R}_{>0}$ of a task j.

The goal is to select a set of bidders $S \subseteq \mathcal{N}$ of minimum cost, that satisfies

$$\sum_{i \in \mathcal{N}_j(\mathbf{I}) \cap S} q_i \geq d_j, \ \forall \ j \in [m], \tag{1}$$

where for $j \in [m]$, $\mathcal{N}_j(\mathbf{I}) := \{i \in \mathcal{N} \mid j \in [s_i, f_i]\}$, i.e. is the set of bidders who can contribute to task j.

The problem belongs to the broad family of problems described by covering integer programs. It may also seem reminiscent of multicover variants of the SET COVER problem on an interval universe. We note however that in CMIC, each worker is allowed to be picked at most once, and moreover, the coverage requirement is not necessarily integral, which is a substantial difference.

Given a feasible solution $S \subseteq \mathcal{N}$, we refer to its total cost, $\sum_{i \in S} c_i$, as the derived *social cost*. Throughout all our work, we focus on deterministic allocation algorithms that use a consistent deterministic tie-breaking rule. Given an instance $P = (\mathbf{b}, \mathbf{q}, \mathbf{d})$, and an allocation algorithm A, we denote by $\mathcal{W}_A(\mathbf{b})$ the set of bidders selected by A, when $\mathbf{q}, \mathbf{d}$ are clear from the context[1]. In the same spirit, we let also $C(A, \mathbf{b}) := \sum_{i \in \mathcal{W}_A(\mathbf{b})} c_i$ be the *social cost* derived by algorithm A on input P. Finally, we use $OPT(\mathbf{b})$ to denote the cost of an optimal solution.

[1] It is convenient to highlight the dependence on $\mathbf{b}$, especially when arguing about truthful mechanisms in the remaining sections.

Truthful Mechanisms

We move now to the strategic scenario, where bids can be private information. A mechanism for a reverse auction, like the ones we study here, is a tuple $M = (A, \boldsymbol{\pi})$, consisting of an *allocation algorithm* A and a *payment rule* $\boldsymbol{\pi}$. Initially, each bidder $j \in \mathcal{N}$ is asked to submit a bid $b_j := (c_j, [s_j, f_j])$, which may differ from her actual cost and interval. Then, given a bidding profile $\mathbf{b} = (\mathbf{c}, \mathbf{I})$, the allocation algorithm $A(\mathbf{b})$ selects the set of winning bidders, i.e., this is a binary setting where the allocation decision for each bidder is whether she is included in the solution or not. Finally, the mechanism computes a vector of payments $\boldsymbol{\pi}(\mathbf{b})$, so that $\pi_i(\mathbf{b})$ is the amount to be paid to bidder i by the auctioneer. Naturally, we will consider that non-winning bidders do not receive any payment.

Note that we consider the contribution q_j, for $j \in \mathcal{N}$, to be public information, available to the auctioneer. The reason is that such a parameter could be estimated by past statistical information on the performance or capacity of the worker. As an example, we refer to [18], where for labeling tasks, it is explained how q_j can be computed as a function of a worker's quality (i.e., the probability for a worker to label correctly), that can be available in a crowdsourcing platform with rating scores or reviews for the workers.

Our setting corresponds to what is usually referred to as a pseudo-2-parameter environment (or almost-single-parameter [5]), since each bidder has two types of private information, the monetary cost and the interval. And in particular, our model can be seen as a special case of *single-minded* bidders, who are interested in a single subset each, but for reverse auctions. As in [18], when the true type of a worker i is $(c_i, [s_i, t_i])$, and she declares her true interval or any non-empty subset of it, then she enjoys a utility of $\pi_i - c_i$, if she is selected as a winner by the mechanism (with π_i being her payment). If she declares any other interval that contains any task $j \notin [s_i, t_i]$, and she is selected as a winner, then the bidder has a utility of $-\infty$, or equivalently has an infinite cost. This simply models the fact that the worker may not be capable of executing or does not desire to be assigned to any task outside her true interval (and therefore would have no incentive for such deviations).

The previous discussion allows us to exploit the sufficient conditions proposed by [31] (for forward auctions), to obtain truthful mechanisms, as an extension of the seminal result by [34]. For reverse auctions, the same framework is also applicable, implying that as long as an allocation algorithm is exact (each selected bidder is assigned her declared interval), the crucial property we need to enforce is *monotonicity*. Monotonicity of an algorithm means that if a winning bidder declares a more competitive bid, she should still remain a winner. To be more precise, we define first the following partial order on possible bids.

Definition 1. *Let $b_i = (c_i, [s_i, f_i])$ and $b'_i = (c'_i, [s'_i, f'_i])$ be two bids of bidder $i \in \mathcal{N}$. We say that $b'_i \succeq b_i$, if $c_i \geq c'_i$ and $[s_i, f_i] \subseteq [s'_i, f'_i]$.*

Definition 2. *An allocation algorithm A is monotone if for every bidding profile $\mathbf{b}$, for any bidder $i \in \mathcal{W}_A(\mathbf{b})$ and any bid $b'_i \succeq b_i$, it holds that $i \in \mathcal{W}_A(b'_i, \mathbf{b}_{-i})$.*

Theorem 1 (cf. [31], Theorem 9.6). *In settings with single-minded bidders, given a monotone and exact algorithm A, there exists an efficiently computable payment rule $\boldsymbol{\pi}$, such that $\mathcal{M} = (A, \boldsymbol{\pi})$ is a truthful mechanism.*

Finally, we stress that all our proposed algorithms are exact by construction, which means that bidders will only be assigned to the set of tasks which they asked for, as in [8], and hence we only need to care about monotonicity.

3 An Improved Truthful Approximation Mechanism

As already stated in Section 1, the currently best algorithm for CMIC has an approximation ratio of 3 [33], based on refining the 4-approximation algorithm by [10]. However, as the next proposition shows, these algorithms are not monotone[2]. More generally, it has been noted in [18] that primal-dual algorithms with a "delete phase" at the end, are typically non-monotone, without, however, providing an example. For the sake of completeness, in the full version of our work, we provide a concrete example that proves the following statement:

Proposition 1. *The current state of the art constant factor approximation algorithms for* CMIC *by [10,33] are not monotone.*

So far, the only truthful mechanism that has been identified [18], achieves an approximation ratio of $2 \cdot \max_{i \in \mathcal{N}}\{q_i\}$. Note that this approximation ratio is dependent on the contribution parameters of the workers, which can become arbitrarily large, and not bounded by any function of n and m. The main result of this section is the following theorem, established via a greedy, local-ratio algorithm, which reduces the gap between truthful and non-truthful mechanisms. In particular, this gap is related to the maximum number of workers that contribute to any given task, which for an instance $((\mathbf{c}, \mathbf{I}), \mathbf{q}, \mathbf{d})$ of CMIC, is $\Delta(\mathbf{I}) := \max_{j=1,\ldots,m} |\mathcal{N}_j(\mathbf{I})|$. We denote it simply by Δ when $\mathbf{I}$ is clear from context. Obviously, Δ is always upper bounded by the number of workers, n.

Theorem 2. *There exists a truthful, polynomial-time mechanism, that achieves a Δ-approximation for the* CMIC *problem.*

The rest of the section is devoted to the proof of Theorem 2. The main component of the proof is an approximation-preserving reduction to a particular job scheduling problem for a single machine [3], defined as follows:

Loss Minimization Interval Scheduling (LMIS): We are given a limited resource whose amount may vary over a time period, which WLOG, is defined by the integral time instants $\{1, \ldots, m\}$. We are also given a set of activities $\mathcal{J} = \{1, \ldots, n\}$, each of which requires the utilization of the resource, for an interval of time instants. An instance of LMIS is determined by a tuple $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$, where:

[2] For a similar reason, the 40-approximation for CMIC by [11], which uses as a subroutine a primal-dual algorithm involving a "delete phase", is non-monotone as well.

- The vector $\mathbf{p} = (p_1, \ldots, p_n)$ specifies a penalty $p_j \in \mathbb{R}_{>0}$, for each activity $j \in \mathcal{J}$, reflecting the cost that is incurred by not scheduling the activity.
- For each activity $j \in \mathcal{J}$, we are given an interval[3] $T_j = [s_j, f_j]$, such that $s_j, f_j \in \{1, \ldots, m\}$ are the start and finish times of j respectively. Let $\mathbf{T} = (T_1, \ldots, T_n)$, be the vector of all activity intervals.
- The vector $\mathbf{r}$ contains, for each activity $j \in \mathcal{J}$, the width $r_j \in \mathbb{R}_{>0}$, reflecting how much resource the activity requires, i.e., this means that activity j requires r_j units of resource at every integral time instant of its interval T_j.
- The vector $\mathbf{D} = (D_1, \ldots, D_m)$ specifies the amount of available resource $D_i \in \mathbb{R}_{>0}$, at each integral time instant $i \in [m]$.

Let $\mathcal{J}_i(\mathbf{T}) := \{j \in \mathcal{J} \mid i \in T_j\}$. The goal in LMIS is to select a set of activities $S \subseteq \mathcal{J}$ to schedule, that meet the resource constraint

$$\sum_{j \in S \cap \mathcal{J}_i(\mathbf{T})} r_j \leq D_i, \quad i = 1, \ldots, m, \tag{2}$$

and such that $\sum_{j \in \mathcal{J} \setminus S} p_j$ is minimized, i.e., we want to minimize the sum of the penalties for the non-scheduled activities.

Our work highlights an interesting connection between LMIS and CMIC. This can be seen via the reduction provided by algorithm $\hat{A}$ below, where for an instance $((\mathbf{c}, \mathbf{I}), \mathbf{q}, \mathbf{d})$, we let $\mathbf{Q}(\mathbf{I}) = (Q_1(\mathbf{I}), \ldots, Q_m(\mathbf{I}))$ and $Q_j(\mathbf{I}) := \sum_{i \in \mathcal{N}_j(\mathbf{I})} q_i, \forall j \in [m]$.

Algorithm 1: $\hat{A}(\mathbf{b})$

▷ **Input:** A bidding profile $\mathbf{b} = (\mathbf{c}, \mathbf{I})$ of a CMIC instance $((\mathbf{c}, \mathbf{I}), \mathbf{q}, \mathbf{d})$

1 Construct the LMIS instance $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D}) = (\mathbf{c}, \mathbf{I}, \mathbf{q}, \mathbf{Q}(\mathbf{I}) - \mathbf{d})$, with $\mathcal{J} = \mathcal{N}$.
2 Run an approximation algorithm for the LMIS instance, and let S be the set of scheduled activities.
3 **return** $\mathcal{N} \setminus S$

Theorem 3. *Algorithm $\hat{A}$ converts any α-approximation algorithm for* LMIS *to an α-approximation algorithm for* CMIC.

Theorem 3 is based on the lemma below, which shows the connection between the feasible solutions of the two problems.

Lemma 1. *Consider a* CMIC *instance $P = ((\mathbf{c}, \mathbf{I}), \mathbf{q}, \mathbf{d})$. Let also P' be the* LMIS *instance defined by $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D}) = (\mathbf{c}, \mathbf{I}, \mathbf{q}, \mathbf{Q}(\mathbf{I}) - \mathbf{d})$, with $\mathcal{J} = \mathcal{N}$. Then, for every feasible solution S of P, it holds that $\mathcal{J} \setminus S$ is a feasible solution for P' with the same cost, and vice versa.*

[3] Originally, the problem was defined using a semi-closed interval for each activity, but it is easy to see that defining it using a closed one instead, is equivalent and more convenient for our purposes.

Approximation Guarantee and Monotonicity of $\hat{A}$

If we only cared about the approximation ratio of $\hat{A}$, it would suffice to use as a black box any algorithm for LMIS. And in fact, the best known algorithm for LMIS achieves a 4-approximation, and was obtained in [3], using the local-ratio framework. Plugging in this algorithm however, does not ensure that we will end up with a truthful mechanism for CMIC. Instead, we will consider an appropriate modification of the algorithm by [3], which enforces monotonicity of $\hat{A}$, but at the price of a higher approximation ratio. This is presented as Algorithm 2 below.

We introduce first a notion that will be useful both for the statement of Algorithm 2 and for our analysis. Consider an instance $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$ of LMIS. Given a set of jobs $S \subseteq \mathcal{J}$, and a time instant $i = 1, \ldots, m$, we define

$$R_i(S, \mathbf{T}, \mathbf{D}) := \sum_{\ell \in S \cap \mathcal{J}_i(\mathbf{T})} r_\ell - D_i.$$

The quantity $R_i(S, \mathbf{T}, \mathbf{D})$ measures how much (if at all), the resource constraint of Equation (2), for the i-th time instant is violated when scheduling all the activities in S. Accordingly, define $R^*(S, \mathbf{T}, \mathbf{D}) := \max_{i=1,\ldots,m} R_i(S, \mathbf{T}, \mathbf{D})$. Note that a schedule S is feasible if and only if $R^*(S, \mathbf{T}, \mathbf{D}) \leq 0$.

Algorithm 2 constructs a feasible schedule S as follows: Initially, it checks if the entire set of activities $S = \mathcal{J}$ constitutes a feasible schedule. If not, the algorithm iteratively removes one activity per iteration from S, in a greedy fashion, until S becomes feasible. The algorithm determines the time instant t^* with the most violated feasibility constraint, by computing $R^*(S, \mathbf{T}, \mathbf{D})$, and considers all activities from S whose interval contains t^*. Then, it removes from S one of these activities that minimizes a certain ratio, dependent on the current penalties and resource requirements, while it simultaneously decreases the penalty of all other activities that contain t^*.

Algorithm 2 LMIS-LR$(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$

▷ **Input:** An instance $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$ of LMIS

Initialize $S = \mathcal{J}, k = 0,$ and $\mathbf{p}_k = (p_{i,k})_{i \in [n]} = \mathbf{p}$.
while $R^*(S, \mathbf{T}, \mathbf{D}) > 0$ **do**
  Let $t^* \in [m]$ be a maximizer of $R^*(S, \mathbf{T}, \mathbf{D})$.
  $S_k = S \cap \mathcal{J}_{t^*}(\mathbf{T})$
  $\varepsilon_k = \min_{i \in S_k} \dfrac{p_{i,k}}{\min\{R^*(S, \mathbf{T}, \mathbf{D}), r_i\}}$
  For $i = 1, \ldots, m$ let

$$p_{i,k+1} = \begin{cases} p_{i,k} - \varepsilon_k \min\{R^*(S, \mathbf{T}, \mathbf{D}), r_i\}, & \text{if } i \in S_k, \\ p_{i,k}, & \text{o/w.} \end{cases}$$

  Let $j^* \in S_k$ be a minimizer of ε_k.
  Set $S = S \setminus \{j^*\}$, and $k = k + 1$.
return S

Remark 1. A variation of Algorithm 2 for LMIS is stated in [3]. The main difference is that the algorithm of [3] has an additional step to ensure that the solution returned is maximal. The extra step helps in improving the approximation ratio, but it destroys the hope for monotonicity of $\hat{A}$. This can be demonstrated using the same example that we used for the primal-dual algorithms of Proposition 1. Furthermore, we note that the algorithm of [3] is presented using the local-ratio jargon. We have chosen to present Algorithm 2 in a self-contained way for ease of exposition but for its analysis, we do make use of the local-ratio framework.

Theorem 4. *Algorithm 2 achieves a Δ-approximation for the* LMIS *problem, where $\Delta = \max_{j=1,\ldots,m} |\mathcal{N}_j(\mathbf{I})|$, and the analysis of its approximation is tight.*

It remains to be shown that $\hat{A}$, with Algorithm 2 as a subroutine, becomes a monotone allocation algorithm for CMIC. For establishing the monotonicity, according to Definition 2, we have to examine the ways in which a winning worker i can deviate from the truth with a bid $b'_i \succeq b_i$, where b_i is her initial bid. By Definition 1, this means that under b'_i, a lower or equal cost and a larger or the same interval are declared, compared to b_i. But to argue about $\hat{A}$, we first have to understand how such deviations from the truth affect the outcome of Algorithm 2, when it is called by $\hat{A}$. Lowering the cost at a CMIC instance corresponds to lowering the penalty of an activity at the LMIS instance that $\hat{A}$ constructs. The lemma below examines precisely what happens when we lower the penalty of a non-scheduled activity in a LMIS instance.

Lemma 2. *For an instance $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$ of* LMIS*, let S be the schedule returned by Algorithm 2, and $j \in \mathcal{J} \setminus S$. Then, for any $p'_j \leq p_j$, it holds that $j \in \mathcal{J} \setminus S'$, where S' is the schedule returned by Algorithm 2 for $((p'_j, \mathbf{p}_{-j}), \mathbf{T}, \mathbf{r}, \mathbf{D})$.*

The next lemma examines enlarging the interval of an activity. This needs a different argument from the previous lemma because the deviation causes an activity to participate in more time instants.

Lemma 3. *For an instance $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$ of* LMIS*, let S be a schedule returned by Algorithm 2 and $j \in \mathcal{J} \setminus S$. For any interval $T'_j \supseteq T_j$, consider the instance $P' = (\mathbf{p}, (T'_j, \mathbf{T}_{-j}), \mathbf{r}, \mathbf{D}')$, where $\mathbf{D}' = (D'_1, \ldots, D'_m)$ such that:*

$$D'_\ell = \begin{cases} D_\ell + r_j, & \text{if } \ell \in T'_j \setminus T_j, \\ D_\ell, & \text{o/w.} \end{cases}$$

Then, $j \in \mathcal{J} \setminus S'$, where S' is the schedule returned by Algorithm 2 for P'.

Combining Lemma 2 and Lemma 3 we get the following:

Theorem 5. *Algorithm $\hat{A}$ is monotone, when using Algorithm 2 as a black box for solving* LMIS*.*

Proof. Fix a bidding profile $\mathbf{b}$ and a winning worker $i \in \mathcal{W}_{\hat{A}}(\mathbf{b})$. Let $b_i = (c_i, [s_i, f_i])$, and consider an arbitrary deviation of i, say $b''_i = (c'_i, [s'_i, f'_i])$, such

that $b_i'' \succeq b_i$. We need to show that $i \in \mathcal{W}_{\hat{A}}(b_i'', \mathbf{b}_{-i})$ and we will do this in two steps. First, we consider the deviation $b_i' = (c_i', [s_i, f_i])$. Since b_i' differs from b_i only with respect to the declared cost and the bids of the remaining workers remain the same, we can directly use Lemma 2 to conclude that $i \in \mathcal{W}_{\hat{A}}(b_i', \mathbf{b}_{-i})$. Having established that i is still a winner under $(b_i', \mathbf{b}_{-i})$, consider now the deviation from b_i' to b_i''. Note that $(b_i'', \mathbf{b}_{-i})$ differs from $(b_i', \mathbf{b}_{-i})$ only with respect to the declared interval of bidder i. Recall also that Algorithm $\hat{A}$ calls Algorithm 2 with input the tuple $(\mathbf{c}, \mathbf{I}, \mathbf{q}, \mathbf{Q}(\mathbf{I}) - \mathbf{d})$. This means that under b_i'', the vector $\mathbf{Q}(\mathbf{I}) - \mathbf{d}$ in the constructed LMIS instance changes only for time instants that belong to $[s_i', f_i'] \setminus [s_i, f_i]$ (where we simply add q_i). But then, this can be handled by Lemma 3, and obtain that $i \in \mathcal{W}_{\hat{A}}(b_i'', \mathbf{b}_{-i})$.

To conclude, it is trivial that Algorithm $\hat{A}$ runs in polynomial time, when using Algorithm 2 for solving LMIS, and hence, the monotonicity of $\hat{A}$, together with Theorems 3 and 4 complete the proof of Theorem 2.

4 A Truthful FPTAS for a Small Number of Tasks

At what follows, we investigate whether we can have truthful mechanisms with a better approximation ratio for special cases of restricted problem size. An instance of CMIC with a constant number of workers can be optimally solved in polynomial time by a brute force algorithm, which, together with the VCG payment scheme, results in a truthful mechanism. On the other hand, the story is different when we have a small number of tasks, since CMIC is NP-hard even for one task [18]. Building upon this negative result, we provide a truthful mechanism that achieves the best possible approximation factor, for the case of a constant number of tasks, and our main result of this section is the following:

Theorem 6. *There exists a truthful FPTAS for* CMIC*, when the number of tasks is constant.*

We would like first to pay attention to the special case of a single task, which corresponds to the MIN-KNAPSACK problem, the minimization version of KNAPSACK, where items have costs (instead of values) and there is a covering requirement (instead of a capacity constraint) for the selected items. The work of [8], which proposes a truthful FPTAS for the classic maximization version of KNAPSACK, claims (without providing a proof) that the analogous result holds for MIN-KNAPSACK too. To our knowledge, the only published work that explicitly attempts to extend [8] and describe a truthful FPTAS for MIN-KNAPSACK is [13]. However, we found that the analysis of truthfulness there is flawed and we refer to our full version for a counterexample, establishing the following claim.

Proposition 2. *The FPTAS for* MIN-KNAPSACK*, proposed by [13], is not monotone.*

Therefore, our Theorem 6 helps to resolve any potential ambiguities for MIN-KNAPSACK. Finally, for two or more tasks, we are not aware of any truthful mechanism attaining any bound better than the one provided by Theorem 2.

Remark 2. As in other approaches for constructing a truthful FPTAS, e.g. [8], we also make the assumption from here onwards, that $c_i \geq 1$ for every worker i, i.e., the workers will not be allowed to declare a cost lower than 1. In the full version of our work, we prove that we can adjust the assumption to $c_i \geq \delta$ for any arbitrarily small δ, but for convenience, here we stick to $\delta = 1$.

Furthermore, we bring in some additional notation that we use in this section. Given a bidding profile $\mathbf{b}$, let $c_{sum}(\mathbf{b}) := \sum_{i \in \mathcal{N}} c_i$. Similarly let $c_{min}(\mathbf{b})$ (resp. $c_{max}(\mathbf{b})$), be the minimum (resp. maximum) cost by the bidders.

4.1 A Pseudopolynomial Dynamic Programming Algorithm

The first step towards designing the FPTAS, is a pseudopolynomial dynamic programming algorithm that returns the optimal solution for the case of a constant number of tasks. For simplicity, we focus on describing the algorithm for the case of two tasks. The generalization is rather obvious (and discussed briefly at the end of this subsection).

Given an instance with two tasks, let d_1, d_2 be the demand requirements of the tasks. Note that $\mathcal{N}$ can be partitioned into three sets, W_0, W_1, W_2, since we can have at most three types of workers: W_0 is the set of workers who can contribute to both tasks, and for $\ell \in \{1, 2\}$, W_ℓ is the set of workers who are capable of contributing only to task ℓ.

We define a 3-dimensional matrix $Q[\ell, i, c]$, where for $\ell = 0, 1, 2$, for $i = 0, 1, \ldots n$, and for $c = 0, 1, \ldots, c_{sum}(\mathbf{b})$, $Q[\ell, i, c]$ denotes the maximum possible contribution that can be jointly achieved by any set of workers in $W_\ell \cap \{1, \ldots, i\}$ with a total cost of exactly c. For our purposes, we assume[4] that for $i \in [n]$, each c_i is an integer, so that c also takes only integral values. Our algorithm is based on computing the values of the cells of Q and we claim that this can be done by exploiting the following recursive relation:

$$Q[\ell, i, c] = \begin{cases} 0, & \text{if } i = 0 \\ Q[\ell, i-1, c], & \text{if } i > 0 \text{ and either } i \notin W_\ell \text{ or } c_i > c \\ \max\{Q[\ell, i-1, c], Q[\ell, i-1, c-c_i] + q_i]\}, & \text{o/w} \end{cases} \tag{3}$$

Observe that for a feasible solution S, the workers who contribute to the demand of task 1 (resp. 2) are those from $S \cap W_0$ and $S \cap W_1$ (resp. $S \cap W_0$ and $S \cap W_2$). Hence, for $\ell \in \{1, 2\}$, it should hold that $\sum_{j \in S \cap W_0} q_j + \sum_{j \in S \cap W_\ell} q_j \geq d_\ell$. Our algorithm then can work as follows: After computing the values of Q, according to Equation (3), return the set of workers that minimize $c^{(0)} + c^{(1)} + c^{(2)}$, subject to $Q[0, n, c^{(0)}] + Q[\ell, n, c^{(\ell)}] \geq d_\ell$, for $\ell \in \{1, 2\}$. This can be done by enumerating all possible options, for breaking down the final cost as a sum of 3 values, $c^{(0)}$, $c^{(1)}$ and $c^{(2)}$. The formal statement can be found below.

[4] It becomes clear in the next subsection, that the dynamic programming procedure is only needed for integral cost values.

Algorithm 3: DP($\mathbf{b}$) (presented for two tasks)

▷ **Input:** A bidding profile $\mathbf{b} = (\mathbf{c}, \mathbf{I})$ of a CMIC instance $(\mathbf{b}, \mathbf{q}, \mathbf{d})$ with $m = 2$

```
for ℓ ∈ {0, 1, 2} do
    for i ∈ {0, 1, ..., n} do
        for c ∈ {0, 1, ..., c_sum(b)} do
            Compute Q[ℓ, i, c] using Equation (3)
return the set of workers that minimize c^(0) + c^(1) + c^(2), s.t.
  Q[0, n, c^(0)] + Q[ℓ, n, c^(ℓ)] ≥ d_ℓ, ∀ℓ ∈ {1, 2} (or +∞, if (b, q, d) has no solution)
```

The optimality of the DP algorithm is straightforward from the preceding discussion. Furthermore, its running time is pseudopolynomial, since the size of the table Q is $3 \cdot (|\mathcal{N}| + 1) \cdot (c_{sum}(\mathbf{b}) + 1)$ and to find the optimal solution we need to check at most $\binom{c_{sum}(\mathbf{b})}{3}$ different combinations for the decomposition of the total cost in three terms, as described earlier. It is easy to extend these ideas, for more tasks given the interval structure of the problem, i.e., the first dimension of Q will have a range of $O(m^2)$ and the enumeration part of the algorithm will require an order of $\binom{c_{sum}}{m^2}$ steps. Finally, we note that since this is an optimal mechanism and we use a deterministic, consistent tie-breaking rule, it will trivially be monotone.

Henceforth, we will be referring to this pseudopolynomial dynamic programming algorithm for any constant number of tasks, as the DP algorithm.

Theorem 7. *Given an instance of* CMIC *on a profile* $\mathbf{b}$*, with a constant number of tasks and integer costs, Algorithm DP*($\mathbf{b}$) *is optimal, monotone, and runs in pseudopolynomial time, i.e. polynomial in the input size and in* c_{sum}*.*

4.2 The FPTAS

In order to convert the DP algorithm to a truthful FPTAS, we adapt the framework of [8]. To that end, we define, for every integer k, an algorithm $A_k(\mathbf{b}, \epsilon)$, that uses the DP algorithm as a subroutine, on a subset of the initial set of bidders, with rounded costs, as follows:

Algorithm 4: $A_k(\mathbf{b}, \epsilon)$

▷ **Input:** A bidding profile $\mathbf{b} = (\mathbf{c}, \mathbf{I})$ of a CMIC instance $(\mathbf{b}, \mathbf{q}, \mathbf{d})$, $\epsilon \in (0, 1)$

Let $\mathcal{L}_k(\mathbf{c}) = \{i \in \mathcal{N} : c_i \leq 2^{k+1}\}$

$a_k = \frac{n}{\epsilon 2^k}$.

for $i \in \mathcal{L}_k(\mathbf{c})$ **do**

  $\bar{c}_i = \lceil a_k \cdot c_i \rceil$

$\bar{\mathbf{b}} = (\bar{c}_i, [s_i, f_i])_{i \in \mathcal{L}_k(\mathbf{c})}$

return DP($\bar{\mathbf{b}}$)

Lemma 4. *Let* $0 < \epsilon < 1$*. For a bidding profile* $\mathbf{b}$ *and* $k \geq 0$*, the algorithm* $A_k(\mathbf{b}, \epsilon)$ *runs in time polynomial in the input size and in* $\frac{1}{\epsilon}$*, and if* $2^k \leq OPT(\mathbf{b}) < 2^{k+1}$*, it computes a solution of cost at most* $(1 + \epsilon)OPT(\mathbf{b})$*.*

Using Lemma 4, we can achieve the desired approximation by checking all possible values for k. However, to provide a polynomial-time mechanism, we can only test polynomially many such values, and hope that we compute the same outcome as if we were able to test all such algorithms. We will show that Algorithm 5, that tests all values up to a certain threshold, is what we need.

Algorithm 5: $A_{\text{FPTAS}}(\mathbf{b}, \epsilon)$

▷ **Input:** A bidding profile $\mathbf{b} = (\mathbf{c}, \mathbf{I})$ of a CMIC instance $((\mathbf{c}, \mathbf{I}), \mathbf{q}, \mathbf{d})$, $\epsilon \in (0, 1)$

for $k = 0, \dots, \lceil \log\left(\frac{nc_{max}(\mathbf{b})}{\epsilon}\right) \rceil$ **do**
  Run $A_k(\mathbf{b}, \epsilon)$ and store the winning set and its cost.

return the set of workers that achieve the minimum cost among the above, breaking ties in favor of the algorithm with the lowest index

Let $k^*(\mathbf{b}) := \lceil \log\left(\frac{nc_{max}(\mathbf{b})}{\epsilon}\right) \rceil$ (or simply k^* when the bidding profile is clear from the context). To see that the algorithm is well-defined, recall that $c_{max}(\mathbf{b}) \geq 1$ and since also $\frac{n}{\epsilon} \geq 1$, we have that $k^* \geq 0$. To establish that this is indeed a FPTAS, we prove in the following lemma that one cannot find a better solution by running an algorithm A_k for a value of k higher than k^*. In combination with Lemma 4, this directly establishes that A_{FPTAS} is a FPTAS for CMIC.

Lemma 5. *Given an instance of* CMIC *and an* $\epsilon \in (0, 1)$*, it holds that* $\mathcal{W}_{A_k(\mathbf{b})} = \mathcal{W}_{A_{k^*}(\mathbf{b})}$*, for every* $k > k^*(\mathbf{b})$*.*

Monotonicity of A_{FPTAS}

To establish monotonicity, we will make use of the following operator:

Definition 3. *Let* $\mathcal{A} = \{A_0, A_1, \dots\}$ *be the set of all allocation algorithms* A_k*. For a profile* $\mathbf{b}$ *and a finite collection of algorithms* $S \subseteq \mathcal{A}$*, let* $MIN(S, \mathbf{b}) := \arg\min_{A \in S} C(A, \mathbf{b})$*, with ties broken in favor of the lowest index.*

Given a bidding profile $\mathbf{b}$, the algorithm A_{FPTAS} can be expressed as $\text{MIN}\{A_0, \dots, A_{k^*(\mathbf{b})}\}$. Hence, the next step is to determine when is the MIN operator monotone. The framework of [8] defines a set of sufficient conditions, for maximization objectives. We adapt these properties below, and we note that they are sufficient conditions for minimization problems as well.

Definition 4. *A monotone allocation algorithm A is bitonic w.r.t. the social cost function C if for any bidding profile* $\mathbf{b}$ *and any worker i, the following hold:*

1. $i \in \mathcal{W}_A(\mathbf{b}) \Rightarrow C(A, \mathbf{b}) \geq C(A, (b'_i, \mathbf{b}_{-i})) \quad \forall b'_i \succeq b_i$
2. $i \notin \mathcal{W}_A(\mathbf{b}) \Rightarrow C(A, \mathbf{b}) \geq C(A, (b'_i, \mathbf{b}_{-i})) \quad \forall b'_i \preceq b_i$

Lemma 6. *For* $k \geq 0$*, Algorithm 4 is monotone and bitonic w.r.t. the social cost function* C*.*

Lemma 6 will be used in conjunction with the following Lemma.

Lemma 7. *(implied by [8]) For a bidding profile* $\mathbf{b}$*,* $MIN(\{A_0, \ldots, A_\ell\}, \mathbf{b})$ *is monotone if* $A_0, \ldots, A_\ell$ *are monotone algorithms that are additionally bitonic w.r.t. the social cost function* C*.*

Before we proceed, we would like to comment on a subtle point regarding the MIN operator. We stress that the set of algorithms A_k that are called by A_{FPTAS}, depends on the input profile $\mathbf{b}$. There is no a priori fixed set of algorithms that are run in every profile, but instead, this is determined by the quantity $k^*(\mathbf{b})$. As a result, Lemma 7 does not suffice on its own. For the monotonicity of A_{FPTAS}, we also need to consider the case that a winning worker declares a lower cost that changes $c_{max}(\mathbf{b})$, and decreases the number of algorithms that A_{FPTAS} runs. This is where Lemma 5 comes to rescue, as explained in the proof of Theorem 8, and this is where the flaw in [13] is located (i.e., the algorithm of [13] does not perform enough iterations).

Given the above discussion, by performing a suitable case analysis and by using Lemmas 6 and 7, we can prove the following theorem, which also completes the proof of Theorem 6 and concludes this section.

Theorem 8. *For the domain of bidding profiles* $\mathbf{b}$*, such that* $c_{min}(\mathbf{b}) \geq 1$*, the algorithm* A_{FPTAS} *is monotone.*

5 Further Implications and Extensions

Apart from truthful mechanisms, it would be interesting to also consider CMIC from the purely algorithmic angle. In particular, the existence of a FPTAS (or other types of approximation schemes) could be possible beyond the case studied in Section 4. Furthermore, it is natural to examine whether the mechanisms presented in the previous sections could be used in more general crowdsourcing scenarios. The following subsections shed more light towards these two questions.

5.1 Relation to Unsplittable Flow Problems and (In)approximability

At what follows, we first establish a connection between CMIC and other relevant algorithmic problems, namely UFP, weighted UFP-COVER, and FAULT-CACHING. These problems concern (a) the selection of subpaths of a path-graph so as to satisfy a demand (resp. capacity) constraint on every edge, at a minimum (resp. maximum) total cost and (b) the minimization of the total cache misses in a fault model with a cache of fixed size and requests for non-uniform size pages. We refer to the full version of the work for the definitions of the problems. Such connections are interesting in their own right, and are summarized in the following theorem.

Theorem 9. CMIC *is equivalent to weighted* UFP-COVER. *Furthermore* UFP *and* FAULT-CACHING *are special cases of* CMIC.

We can now exploit some known results about these problems and obtain analogous implications for CMIC. The following table lists the most important results that concern the polynomial solvability and the approximability of (weighted) UFP-COVER, UFP and FAULT-CACHING that also apply for CMIC, due to Theorem 9. As an example, we can now rule out a FPTAS for the general case of CMIC (and in fact for its simpler cases, as stated in the first two lines of Table 2), unless P=NP. We note that before our work, there were no results about the hardness of CMIC, other than the obvious NP-hardness result in [18]. We comment further on Table 2 in Section 6.

Restriction	**Result for CMIC**	**Reduction**
$d_i = 1 \quad \forall i \in [m]$ $q_i \in \{1,2,3\} \quad \forall i \in \mathcal{N}$ $c_i \in \mathbb{Z}$	strongly NP-hard $\overset{[40]}{\Longrightarrow} \nexists$ FPTAS*	UFP [6]
$d_i = 1 \quad \forall i \in [m]$ $c_i = 1 \quad \forall i \in \mathcal{N}$	strongly NP-hard $\overset{[40]}{\Longrightarrow} \nexists$ FPTAS*	FAULT-CACHING [14]
$c_i = 1 \quad \forall i \in \mathcal{N}$	W[1]-hard $\overset{[21]}{\Longrightarrow} \nexists$ EPTAS$^{\odot}$	UFP-COVER [16]
q_i : quasi-poly $\quad \forall i \in \mathcal{N}$	QPTAS $\overset{[1]}{\Longrightarrow} \nexists$ APX-hardness$^{\diamond}$	weighted UFP-COVER [26]

Table 2: Results concerning the (in)approximability of CMIC implied by Theorem 9 together with existing works on UFP-COVER, UFP and FAULT-CACHING. We have used the symbols $\star, \odot, \diamond$ to denote the complexity assumptions P $\neq$ NP, W[1] $\neq$ FPT and QP $\neq$ NP respectively. For the W[1]-hardness result, it is assumed that UFP-COVER is parameterized by the cardinality of the optimal solution.

5.2 Generalization to Non-Interval Structures

The focus of our work has been largely on highlighting the difference on the approximation ratio between truthful and non-truthful algorithms. We conclude our work by showing that one can get tighter results, when moving to more general scenarios. A direct generalization of CMIC is to drop the linear arrangement of the tasks, and allow each worker $j \in \mathcal{N}$ to declare an arbitrary subset of tasks $I_j \subseteq [m]$. The rest of the input remains the same (costs, contributions and demands), and we refer to this problem as Cost Minimization Demand Cover (CMDC). Notice that CMDC with $q_i = 1$ for every worker i, and $d_j = 1$ for every task j, is nothing but the famous SET COVER problem.

Further extensions of CMDC have been studied under various names in a series of works regarding covering IPs (e.g., [28,38]) and approximation algorithms that match the factor of Algorithm 1 exist (e.g. [9,22,30,37]). However, the focus of these works was not about monotonicity and it is unclear if any of these algorithms are monotone (in fact some of them are certainly non-monotone).

Towards obtaining a monotone algorithm, a careful inspection of the proofs of Section 3, suffices to deduce that Algorithm 1 can be used for this more general setting as well (after defining first the appropriate generalization of LMIS) and it continues to yield the Δ factor for CMDC. Furthermore, under this setting, Algorithm 1 yields essentially a tight result, according to the following Proposition. The proof of Proposition 3 is straightforward due to the hardness results

for k-UNIFORM HYPERGRAPH VERTEX COVER [27,19], which is a special case of SET COVER, and therefore a special case of CMDC.

Proposition 3. *For the class of instances where Δ is constant,* CMDC *is $\Delta-1-\epsilon$ inapproximable, unless $P = NP$, and $\Delta-\epsilon$ inapproximable assuming the Unique Games Conjecture.*

Finally, we note that the algorithms of Section 4 can in principle also be applied for CMDC with no loss in the approximation factor, but at the expense of a much higher running time (doubly exponential in m, which still remains polynomial, as long as the number of tasks is a constant).

6 Discussion and Open Problems

From a mechanism design viewpoint, the most important question for future research is to design truthful mechanisms with better approximation guarantees for CMIC, as there is still a large gap between the non-truthful 3-approximation and our truthful Δ-approximation of Section 3. Moreover, exploring the approximability of well-motivated special cases of CMIC, other than the restriction on a constant number of tasks that we examined in Section 4, is also an intriguing topic (Table 2 in Section 5.1 suggests such possible restrictions). In the context of crowdsourcing, some of these special cases are also meaningful to study in a 2-dimensional model, where workers can cover circular areas of a given radius, or other geometric shapes. Apart from positive results, we believe it is very interesting to investigate the existence of lower bounds on the worst case performance of polynomial time truthful mechanisms. After all, it is conceivable that there may be a strict separation on the approximability by monotone and non-monotone algorithms, as, e.g., for combinatorial public project problems [35].

From a purely algorithmic viewpoint, we would like to point out some interesting open-problems for CMIC, that emerge from the results of Table 2. In particular, it has been known by [18], that we have polynomial solvability when all the workers have the same contribution parameter. On the contrary, by the first row of Table 2, we have a hardness result when the workers can be partitioned in three distinct groups by their contribution. It still remains open to determine what happens when we have two distinct groups for the contribution of the workers. Interestingly the last row of Table 2 implies that the existence of a PTAS is likely (at least for the case of quasi-polynomially bounded contributions) and, yet, such a PTAS still remains to be found. Finally, even though the problem is W[1]-hard (parameterized by the number of workers in the optimal solution) for unit costs, it belongs to FPT for unit costs and integral numerical values [32,24], and hence an EPTAS may well exist for this case.

Acknowledgements. This research was supported by the Hellenic Foundation for Research and Innovation. The first two authors were supported by the "1st Call for HFRI Research Projects to support faculty members and researchers and the procurement of high-cost research equipment" (Project Num. HFRI-FM17-3512) and the third author by the HFRI PhD Fellowship grant (Fellowship Num. 289).

References

1. Bansal, N., Chakrabarti, A., Epstein, A., Schieber, B.: A quasi-PTAS for unsplittable flow on line graphs. In: Proceedings of the 38th ACM Symposium on Theory of Computing. pp. 721–729 (2006)
2. Bansal, N., Pruhs, K.: The geometry of scheduling. SIAM Journal on Computing **43**(5), 1684–1698 (2014)
3. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. Journal of the ACM **48**(5), 1069–1090 (2001)
4. Bar-Yehuda, R., Bendel, K., Freund, A., Rawitz, D.: The local ratio technique and its application to scheduling and resource allocation problems. In: Graph Theory, Combinatorics and Algorithms. Springer (2005)
5. Blumrosen, L., Nisan, N.: Algorithmic Game Theory. Introduction to mechanism design, Cambridge University Press (2007)
6. Bonsma, P., Schulz, J., Wiese, A.: A constant-factor approximation algorithm for unsplittable flow on paths. SIAM Journal on Computing **43**(2), 767–799 (2014)
7. Bredereck, R., Faliszewski, P., Niedermeier, R., Skowron, P., Talmon, N.: Mixed integer programming with convex/concave constraints: Fixed-parameter tractability and applications to multicovering and voting. Theoretical Computer Science (2020)
8. Briest, P., Krysta, P., Vöcking, B.: Approximation techniques for utilitarian mechanism design. SIAM Journal on Computing **40**(6), 1587–1622 (2011)
9. Carr, R.D., Fleischer, L.K., Leung, V.J., Phillips, C.A.: Strengthening integrality gaps for capacitated network design and covering problems. In: Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms. p. 106–115 (2000)
10. Chakaravarthy, V.T., Kumar, A., Roy, S., Sabharwal, Y.: Resource allocation for covering time varying demands. In: European Symposium on Algorithms (2011)
11. Chakrabarty, D., Grant, E., Könemann, J.: On column-restricted and priority covering integer programs. In: International Conference on Integer Programming and Combinatorial Optimization. pp. 355–368. Springer (2010)
12. Chen, B., Hassin, R., Tzur, M.: Allocation of bandwidth and storage. IIE Transactions **34**(5), 501–507 (2002)
13. Chen, J., Ye, D., Ji, S., He, Q., Xiang, Y., Liu, Z.: A truthful FPTAS mechanism for emergency demand response in colocation data centers. In: Proceedings of the IEEE International Conference on Computer Communications-INFOCOM (2019)
14. Chrobak, M., Woeginger, G.J., Makino, K., Xu, H.: Caching is hard—even in the fault model. Algorithmica **63**(4), 781–794 (2012)
15. Cramton, P., Shoham, Y., Steinberg, R., et al.: Combinatorial auctions. Tech. rep., University of Maryland (2006)
16. Cristi, A., Mari, M., Wiese, A.: Fixed-parameter algorithms for unsplittable flow cover. Theory of Computing Systems (2021)
17. Csirik, J.: Heuristics for the 0-1 min-knapsack problem. Acta Cybernetica (1991)
18. Dayama, P., Narayanaswamy, B., Garg, D., Narahari, Y.: Truthful interval cover mechanisms for crowdsourcing applications. In: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (2015)
19. Dinur, I., Guruswami, V., Khot, S., Regev, O.: A new multilayered PCP and the hardness of hypergraph vertex cover. SIAM Journal on Computing **34**(5), 1129–1146 (2005)

20. Elkind, E., Goldberg, L.A., Goldberg, P.W.: Frugality ratios and improved truthful mechanisms for vertex cover. In: Proceedings of the 8th ACM Conference on Electronic Commerce (2007)
21. Feldmann, A., Karthik, C., Lee, E., Manurangsi, P.: A survey on approximation in parameterized complexity: Hardness and algorithms. Algorithms **13**(6) (2020)
22. Fujito, T., Yabuta, T.: Submodular integer cover and its application to production planning. In: International Workshop on Approximation and Online Algorithms. pp. 154–166. Springer (2004)
23. Gálvez, W., Grandoni, F., Ingala, S., Heydrich, S., Khan, A., Wiese, A.: Approximating geometric knapsack via L-packings. ACM Transactions on Algorithms **17**(4), 1–67 (2021)
24. Gavenciak, T., Knop, D., Koutecký, M.: Integer programming in parameterized complexity: three miniatures. In: 13th International Symposium on Parameterized and Exact Computation. pp. 21:1–21:16 (2019)
25. Gummidi, S.R.B., Xie, X., Pedersen, T.B.: A survey of spatial crowdsourcing. ACM Transactions on Database Systems **44**(2), 1–46 (2019)
26. Höhn, W., Mestre, J., Wiese, A.: How unsplittable-flow-covering helps scheduling with job-dependent cost functions. Algorithmica (2018)
27. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2-\varepsilon$. Journal of Computer and System Sciences **74**(3), 335–349 (2008)
28. Kolliopoulos, S., Young, N.: Approximation algorithms for covering/packing integer programs. Journal of Computer and System Sciences **71**(4), 495–505 (2005)
29. Kothari, A., Parkes, D., Suri, S.: Approximately-strategyproof and tractable multiunit auctions. Decision Support Systems **39**(1), 105–121 (2005)
30. Koufogiannakis, C., Young, N.: Greedy δ-approximation algorithm for covering with arbitrary constraints and submodular cost. Algorithmica (2013)
31. Lehmann, D., O'Callaghan, L.I., Shoham, Y.: Truth revelation in approximately efficient combinatorial auctions. Journal of the ACM **49**(5), 577–602 (2002)
32. Lenstra, H.: Integer programming with a fixed number of variables. Mathematics of Operations Research (1983)
33. Mondal, S.: Improved algorithm for resource allocation problems. Asia-Pacific Journal of Operational Research **35**(01), 1–23 (2018)
34. Myerson, R.: Optimal auction design. Mathematics of Operations Research **6**(1), 58–73 (1981)
35. Papadimitriou, C., Schapira, M., Singer, Y.: On the hardness of being truthful. In: Proceedings of the 49th Foundations of Computer Science. pp. 250–259 (2008)
36. Phillips, C.A., Uma, R., Wein, J.: Off-line admission control for general scheduling problems. Journal of Scheduling **3**(6), 365–381 (2000)
37. Pritchard, D., Chakrabarty, D.: Approximability of sparse integer programs. Algorithmica **61**(1), 75–93 (2011)
38. Rajagopalan, S., Vazirani, V.: Primal-dual RNC approximation algorithms for (multi)-set (multi)-cover and covering integer programs. In: Proceedings of the 34th Foundations of Computer Science (1993)
39. Tong, Y., Zhou, Z., Zeng, Y., Chen, L., Shahabi, C.: Spatial crowdsourcing: a survey. The VLDB Journal **29**(1), 217–250 (2020)
40. Vazirani, V.: Approximation algorithms. Springer (2001)
41. Xu, J., Xiang, J., Yang, D.: Incentive mechanisms for time window dependent tasks in mobile crowdsensing. IEEE Transactions on Wireless Communications (2015)
42. Zhang, L., Ren, S., Wu, C., Li, Z.: A truthful incentive mechanism for emergency demand response in colocation data centers. In: Proceedings of the IEEE International Conference on Computer Communications-INFOCOM (2015)