Partial Multicovering and the $d$-consecutive Ones Property

Dror Rawitz* Shimon (Moni) Shahar*
rawitz@eng.tau.ac.il moni@eng.tau.ac.il

May 4, 2011

Abstract

A $d$-interval is the union of $d$ disjoint intervals on the real line. In the $d$-interval stabbing problem ($d$-is) we are given a set of $d$-intervals and a set of points, each $d$-interval $I$ has a stabbing requirement $r(I)$ and each point has a weight, and the goal is to find a minimum weight multiset of points that stab each $d$-interval $I$ at least $r(I)$ times. In practice there is a trade-off between fulfilling requirements and cost, and therefore it is interesting to study problems in which one is required to fulfill only a subset of the requirements. In this paper we study variants of $d$-is in which a feasible solution is a multiset of points that may satisfy only a subset of the stabbing requirements. In partial $d$-is we are given an integer $t$, and the sum of requirements satisfied by the computed solution must be at least $t$. In prize collecting $d$-is each $d$-interval has a penalty that must be paid for every unit of unsatisfied requirement. We also consider a maximization version of prize collecting $d$-is in which each $d$-interval has a prize that is gained for every time, up to $r(I)$, it is stabbed. Our study is motivated by several resource allocation and geometric facility location problems. We present a $\left(\frac{d^2 + d - 1}{\rho}\right)$-approximation algorithm for prize collecting $d$-is, where $\rho = \min_I r(I)$, and an $O(d)$-approximation algorithm for partial $d$-is. We obtain the latter result by designing a general framework for approximating partial multicovering problems that extends the framework for approximating partial covering problems from [21]. We also show that maximum prize collecting $d$-is is at least as hard to approximate as maximum independent set, even for $d = 2$, and present a $d$-approximation algorithm for maximum prize collecting $d$-dimensional rectangle stabbing.

Keywords: Approximation algorithms, $d$-interval stabbing, prize collecting multicovering, partial multicovering.
1 Introduction

The multi-cover problem is a natural extension of set cover. We are given a set of elements $U$ and a collection of subsets $S$ such that $S \subseteq U$ for every $S \in S$. In addition each element $u$ has a covering requirement $r(u) \in \mathbb{N}$. We define $\rho \triangleq \min u r(I)$. A multi-cover is a multiset $C$ of subsets such that every element $u \in U$ is covered by at least $r(u)$ subsets from $C$. Note that since $C$ is a multiset, an element $u$ may be covered by several copies of the same subset. Given a weight function $w$ on the subsets, the weight of a multi-cover $C$ is the total weight of the subsets in $C$, where multiplicities are counted. The goal in multi-cover is to find a multi-cover of minimum weight. Note that when $r(u) = 1$ for every $u$, the problem reduces to set cover. In this paper we focus on variants of multicovering in which a feasible solution is a multiset $C$ that may satisfy only a subset of the covering requirements. In prize collecting multicovering each element has a penalty that must be paid for every unit of unsatisfied requirement. In partial multicovering we are given an integer $t$, and the goal is to find a minimum weight multiset such that the total requirements satisfied is at least $t$.

We consider multi-covering problems that can be formulated using linear integer programs in which each row of the constraint matrix contains at most $d$ blocks of consecutive ones. Formally, we consider problems that can be described as $\{ x \in \mathbb{N}^n : Ax \geq b \}$, where $A \in \{0,1\}^{m \times n}$, $b \in \mathbb{N}^n$ and each row of $A$ contains at most $d$ blocks of consecutive ones. We refer to such a matrix as satisfying the $d$-consecutive ones property. Observe that the $d$-set cover problem, in which each element appears in at most $d$ subsets, is the special case where $b_j = 1$ for every $j$, and each block of consecutive ones is of size one.

Multi-cover with $d$-consecutive ones can be represented geometrically as a $d$-interval stabbing problem ($d$-is). Each element is a $d$-interval consisting of up to $d$ disjoint intervals on the real line. The $j$th segment of a $d$-interval $I$ is denoted by $I^j$. (Henceforth, we assume without loss of generality that each $d$-interval $I$ has exactly $d$ segments.) In this representation the sets are points. We denote the set of $d$-intervals by $\mathcal{I}$ and the set of points by $P$. Our goal is to find a minimum weight multiset of points that stabs each $d$-interval $I$ at least $r(I)$ times. When $d = 1$ the $d$-intervals are simply intervals, and we get the interval stabbing problem (is). Unweighted is with unit requirements is the well known clique cover problem in interval graphs [13]. Is is solvable in polynomial time, since it satisfies the consecutive ones property.

The rectangle stabbing problem (rs) is a two dimensional generalization of interval stabbing, where the input is a set $\mathcal{R}$ of axis parallel rectangles and a set $L$ of horizontal and vertical lines. Each rectangle $R \in \mathcal{R}$ has a requirement $r(R)$, and each line $\ell \in L$ has a weight. The goal is to find a minimum weight multiset of lines that stabs every rectangle $R$ at least $r(R)$ times. Observe that the problem of stabbing arbitrary objects on the plane with axis parallel lines can easily reduced to rs. Rectangle stabbing is a special case of 2-interval stabbing, because each axis can be represented by a separate region of the real line. Formally, an rs instance can be described as a 2-is instance in which, for every $I, J \in \mathcal{I}$, the left segment of $I$ and the right segment of $J$ are disjoint. The $d$-dimensional rectangle stabbing problem ($d$-rs) is a $d$-dimensional version of rectangle stabbing. In this case, the set $\mathcal{R}$ consists of axis parallel $d$-dimensional rectangles and the set $L$ consists of hyper-planes that are orthogonal to one of the $d$ axes. $d$-rs is the special case of $d$-is in which the segments of the $d$-intervals are labeled in such a way that for any two $d$-intervals $I, J \in \mathcal{I}$ if $i \neq j$ then $I^i \cap J^j = \emptyset$.

Hassin and Megiddo [15] showed that rs is NP-hard even in the special case of unit weights and
requirements and where each rectangle is intersected by only one horizontal line and two vertical lines. Kovaleva and Spieksma [23] showed that this special case of rs is APX-hard. Hence, d-is is also APX-hard for $d \geq 2$. Dinur et al. [7] proved that it is NP-hard to approximate $d$-SET COVER within a factor of $d - 1 - \varepsilon$, for any $\varepsilon > 0$ (assuming $d > 2$ is a constant). The same lower bound applies to d-is. Recently, Dom et al. [8] showed that both rs and 2-is (unit weights and requirements) are $W[1]$-complete.

In this paper we consider several variants of d-is in which a feasible solution is a partial cover. That is, a feasible solution is a multiset $H$ that stabs the $d$-intervals, but not necessarily all of them. In partial $d$-interval stabbing (PARTIAL-d-is) we are given a parameter $t$, and the goal is to find a minimum weight multiset $H$ that satisfies at least $t$ of the total requirement. In the prize collecting version of d-is a feasible solution is a multiset $H$ that stabs the $d$-intervals, but not necessarily all of them. Formally, in the PRIZE COLLECTING $d$-INTERVAL STABBING problem (PC-d-is) each $d$-interval $I$ has a penalty, and we are required to pay this penalty for every unit of requirement that was not satisfied. We also consider the MAXIMUM PRIZE COLLECTING $d$-INTERVAL STABBING problem (MAX-PC-d-is), which is a maximization problem in which each $d$-interval has a prize that is gained for each unit of requirement that is satisfied. The weight of a solution is the sum of prizes minus the cost of the multiset.

1.1 Related Work and Applications

Applications related to numeric computation and image processing motivated the investigation of the rectangle stabbing problem by Gaur et al. [11] (see also [12]). They obtained a $d$-approximation algorithm for $d$-rs with unit requirements by reducing $d$ dimensions to $d$ instances of is using linear programming. Rs and $d$-rs have many other applications such as radiotherapy, planning an air raid on a multitarget site [15] and embedded sensor networks [6]. Kovaleva and Spieksma [22] considered a special case of rs called the segment stabbing, in which each rectangle is intersected by only one horizontal line. They obtained an $(\frac{e}{e-1})$-approximation algorithm for segment stabbing with unit requirements. They also provided a $(\frac{\rho+1}{\rho})$-approximation algorithm for rs (with requirements) by extending the algorithm for rs from [11].

Hassin and Segev [16] also extended the algorithm from [11] by presenting a $d$-approximation algorithm for d-is. They applied their algorithm to MULTI-RADIUS COVER, where one is given a graph with non-negative edge requirements. Each vertex represents a transmission station and is associated with a list of pairs of transmission radius and cost. MULTI-RADIUS COVER asks to assign a transmission radius to each station, such that the sum of radii seen by each edge is not less than its requirement, and such that the total cost is minimized. Hassin and Segev obtained a 2-approximation algorithm by showing that multi-radius cover is a special case of 2-is. Mecke et al. [24] also provided a $d$-approximation algorithm for d-is (with unit requirements). Their work was motivated by the planning of station locations in a transportation network.

A $d$-approximation algorithm for PRIZE-COLLECTING $d$-SET COVER was presented in [3]. pc-is can be solved in polynomial time, since its constraint matrix is totally unimodular (see Section 2 for more details). Hassin and Tamir [17] studied a facility location problem on the real line that extends pc-is. Their results imply that is and pc-is with unit requirements can be solved in $O(n)$ time and $O(n \log n)$, respectively. Furthermore, pc-is with unit requirements can be solved in $O(n)$ time, if all intervals have the same length. Kovaleva and Spieksma [23] presented a $(2 + \varepsilon)$-approximation algorithm for PRIZE COLLECTING SEGMENT STABBING. They also presented 2-approximation algorithms for the case of unit weights and no penalties, and for the case of unit
requirements. Könemann et al. [21] obtained a \( d \)-approximation algorithm for PC-\( d \)-IS with unit requirements.

The greedy algorithm is an \( H_m \)-approximation algorithm for PARTIAL SET COVER, where \( H_m \) is the \( m \)th harmonic number [20, 27]. Bshouty and Burroughs [4] obtained a 2-approximation algorithm for PARTIAL VERTEX COVER. \( d \)-approximation algorithms for PARTIAL \( d \)-SET COVER are given in [1, 9, 10]. Könemann et al. [21] presented a reduction from partial covering to prize collecting covering. Specifically, they showed how to use a Lagrangian multiplier preserving algorithm with factor \( \alpha \) for a prize collecting covering problem to design a \((\frac{4}{3} + \varepsilon)\alpha\)-approximation algorithm, for any \( \varepsilon > 0 \), for the partial version of this problem (see Section 3). Using this result they were able to obtain a \((\frac{4d}{3} + \varepsilon)\)-approximation algorithm for PARTIAL-\( d \)-IS with unit requirements. Recently, Mestre [25] has shown that in general no algorithm that uses a Lagrangian multiplier preserving \( \alpha \)-approximation algorithm as a black box can yield an approximation factor better than \( \frac{4}{3} \alpha \).

The maximum prize collecting set cover problem was defined by Charikar et al. [5], who studied the problem of online advertising in the Internet. In one of their models, users are elements and advertisements are sets, and an element belongs to a set if the corresponding advertisement is relevant for the corresponding user. The cost of a set is the cost of sending the advertisement. Furthermore, there is a revenue (or prize) associated with each user that is obtained if it receives a relevant advertisement. They presented a greedy algorithm that chooses in each step the set that maximizes the ratio between the change in revenue and the change in cost. They bounded the approximation ratio in terms of \( y = \frac{r^*}{c^*} \), where \( r^* \) and \( c^* \) are the revenue and weight of an optimal solution. They proved that the approximation ratio of the greedy algorithm is at most \((1 - \frac{\ln y}{y - 1})^{-1}\), and that the integrality gap of the natural linear program is at least \((1 - \frac{\ln y}{y - 1})^{-1}\). They also noted that this bound tends to \( \infty \) when \( y \) goes to 1, and therefore this is not a uniform bound.

The multi-cover problem in which each column of the constraint matrix contains at most \( d \) blocks of consecutive ones was studied by Hochbaum and Levin [19]. (This problem can be viewed as covering points using \( d \)-intervals.) They presented a \( d \)-approximation algorithm for this problem that is based on a reduction to the problem of covering points by intervals which is solvable in polynomial time. Another problem whose constraint matrix \( A \) contains columns with at most \( d \) blocks of consecutive ones is MAXIMUM INDEPENDENT SET in \( t \)-interval graphs. A \( d \)-interval graph is the intersection graph of \( d \)-intervals. Bar-Yehuda et al. [2] presented a \( 2d \)-approximation algorithm for MAXIMUM INDEPENDENT SET in \( d \)-interval graphs.

### 1.2 Our Results

Section 2 contains two approximation algorithms for PC-\( d \)-IS: a \((\frac{\nu + d - 1}{\nu})\)-approximation algorithm for PC-\( d \)-IS that extends the \((\frac{\nu + 1}{\nu})\)-approximation algorithms for 2-RS from [22], and a Lagrangian multiplier preserving \( d \)-approximation algorithm. In Section 3 we design a general framework for approximating partial multicovering problems that is based on the framework for approximating partial covering problems from [21]. Specifically, we show how to design a \((\frac{4}{3} + \varepsilon)\alpha\)-approximation algorithm, for any \( \varepsilon > 0 \), for a partial multicovering problem using a Lagrangian multiplier preserving algorithm with factor \( \alpha \) for the prize collecting version of the problem. We obtain a \((\frac{4d}{3} + \varepsilon)\)-approximation algorithm for PARTIAL-\( d \)-IS using the above mentioned Lagrangian multiplier preserving \( d \)-approximation algorithm for PC-\( d \)-IS. Finally, in Section 4 we show that MAX-PC-\( d \)-IS is as hard to approximate as MAXIMUM INDEPENDENT SET, even for \( d = 2 \), and we obtain a \( d \)-approximation algorithm for MAX-PC-\( d \)-RS.
2 Prize Collecting $d$-interval Stabbing

In this section we present a $(\frac{\rho+d-1}{\rho})$-approximation algorithm for PC-$d$-IS that is based on an optimal solution of an LP relaxation of the problem. (Recall that $\rho = \min_I r(I)$.) We also provide a $d$-approximation algorithm for PC-$d$-IS that possesses a property that will become useful in designing an approximation algorithm for PARTIAL-$d$-IS in Section 3.

2.1 LP Formulation

PC-$d$-IS can be formalized using the following linear integer program.

\[
\begin{align*}
\min & \sum_{p \in P} w(p)x(p) + \sum_{I \in \mathcal{I}} w(I)z(I) \\
\text{s.t.} & \quad \sum_{p \in I} x(p) + z(I) \geq r(I) \quad \forall I \in \mathcal{I} \\
& \quad x(p), z(I) \in \mathbb{N} \quad \forall p \in P, I \in \mathcal{I}
\end{align*}
\]

(IP-PC)

where $x(p)$ is the number of copies of $p$ in the solution, and $z(I)$ is the unsatisfied requirement of $I$. The LP-relaxation of IP-PC is obtained by replacing the integrality constraints by: $x(p) \geq 0$, for every $p$, and $z(I) \geq 0$, for every $I$. We denote the LP relaxation by LP-PC.

Observe that when $d = 1$ IP-PC can be written as $[M, I]$, where $M$ is a matrix that has the consecutive ones property. As mentioned by Kőnemann et al. [21], such matrices form a well known class of totally unimodular matrices (see, e.g., [26, page 540]). It follows that when $d = 1$ the integrality gap of IP-PC is 1, and that it can be solved in polynomial time.

2.2 $(\frac{\rho+d-1}{\rho})$-approximation Algorithm

We present a $(\frac{\rho+d-1}{\rho})$-approximation algorithm for PC-$d$-IS that is based on a fractional optimal solution of LP-PC and an algorithm for solving PC-IS.

Let ALG be an algorithm for PC-IS that computes integral optimal solutions of LP-PC. Our algorithm starts by computing an optimal solution $(x^*, z^*)$ of LP-PC whose value is OPT*. Next, it constructs a PC-IS instance $(\mathcal{J}, r', w')$ as follows. The set of points is the same as in the original PC-$d$-IS instance and their weights remain as they were. The set of intervals $\mathcal{J}$ contains all segments of all $d$-intervals in $\mathcal{I}$, i.e.,

$$\mathcal{J} = \{J_{ij} : J_{ij} = I_i^j, I_i \in \mathcal{I}, j \in \{1, \ldots, d\}\}.$$

The requirement of $J_{ij}$ is

$$r'(J_{ij}) = \left[ \frac{\rho + d - 1}{\rho} \cdot \beta(I_i^j) \right] r(I_i),$$

where

$$\beta(I_i^j) = \frac{\sum_{p \in I_i^j} x(p)}{\sum_{p \in I_i} x(p)},$$

and its weight is $w'(J_{ij}) = w(I_i)$. Our algorithm invokes ALG on the new PC-IS instance and computes an optimal solution $(x', z')$. Finally, it outputs the solution $(x, z)$, where $x = x'$ and $z(I_i) = \sum_j z'(J_{ij})$, for every $d$-interval $I_i$. 

4
Lemma 1. \((x, z)\) is a feasible solution.

Proof. We prove that \((x, z)\) is feasible by showing that the constraints of \(\text{IP-PC}\) are satisfied. Since \((x', z')\) is feasible with respect to the \(\text{PC-IS}\) instance, it follows that for any \(d\)-interval \(I_i\)

\[
\sum_{p \in I_i} x(p) + z(I_i) = \sum_{j=1}^{d} \left( \sum_{p \in I_{ij}} x'(p) + z'(J_{ij}) \right)
\]

\[
\geq \sum_{j=1}^{d} r'(J_{ij})
\]

\[
= \sum_{j=1}^{d} \left[ \frac{\rho + d - 1}{\rho} \cdot \beta(I_i^j) r(I_i) \right].
\]

Furthermore, since \(\sum_{j=1}^{d} [\lambda_j] \geq \left[ \sum_{j=1}^{d} \lambda_j \right] - (d - 1)\), for every \(\lambda_1, \ldots, \lambda_d \in \mathbb{R}_+\), we have

\[
\sum_{p \in I_i} x(p) + z(I_i) \geq \left[ \sum_{j} \left( \frac{\rho + d - 1}{\rho} \cdot \beta(I_i^j) r(I_i) \right) \right] - (d - 1)
\]

\[
= \left[ \frac{\rho + d - 1}{\rho} \cdot r(I_i) \right] - (d - 1)
\]

\[
\geq \left[ \rho(r(I_i) + d - 1) \right] - (d - 1)
\]

\[
= r(I_i),
\]

where the second inequality follows from \(r(I) \geq \rho\).

We now show that the algorithm computes \((\frac{\rho+d-1}{\rho})\)-approximate solutions.

Lemma 2. \(\sum_p w(p)x(p) + \sum_I w(I)z(I) \leq \frac{\rho+d-1}{\rho} \cdot \text{OPT}\).

Proof. We define a fractional solution \((\bar{x}, \bar{z})\) for the \(\text{PC-IS}\) instance as follows: \(\bar{x}(p) = \frac{\rho+d-1}{\rho} \cdot x^*(p)\), for every \(p\), and \(\bar{z}(J_{ij}) = \frac{\rho+d-1}{\rho} \cdot \beta(I_i^j) \cdot z^*(I_i)\), for every \(I_i\) and \(j\). We show that this solution is feasible with respect to the \(\text{PC-IS}\) instance. The constraint for \(J_{ij}\) is satisfied, since

\[
\sum_{p \in J_{ij}} \bar{x}(p) + \bar{z}(J_{ij}) = \sum_{p \in J_{ij}} \frac{\rho + d - 1}{\rho} \cdot x^*(p) + \frac{\rho + d - 1}{\rho} \cdot \beta(I_i^j) z^*(I_i)
\]

\[
= \frac{\rho + d - 1}{\rho} \cdot \left( \beta(I_i^j) \sum_{p \in I_i} x^*(p) + \beta(I_i^j) z^*(I_i) \right)
\]

\[
\geq \frac{\rho + d - 1}{\rho} \cdot \beta(I_i^j) r(I_i)
\]

\[
\geq r'(J_{ij}).
\]
Since the integrality gap of the PC-IS instance is 1, it follows that
\[
\sum_p w(p)x(p) + \sum_{I_i} w(I_i)z(I_i) = \sum_p w'(p)x'(p) + \sum_{J_{ij}} w'(J_{ij})z'(J_{ij}) \\
\leq \sum_p w'(p)\bar{x}(p) + \sum_{J_{ij}} w'(J_{ij})\bar{z}(J_{ij}) \\
= \frac{\rho + d - 1}{\rho} \left( \sum_p w(p)x^*(p) + \sum_{I_i} w(I_i)z^*(I_i) \right) \\
= \frac{\rho + d - 1}{\rho} \cdot \text{OPT}^* ,
\]
and we are done. \hfill \Box

From Lemmas 1 and 2 we get the following result.

**Theorem 1.** There exists a polynomial time \((\frac{\rho + d - 1}{\rho})\)-approximation algorithm for PC-d-IS.

We show that the analysis is tight using the following instance. Consider a d-interval \(I\) with requirement \(\rho\), where the \(j\)th segment is stabbed by a point of weight 1 for every \(j\). An optimal fractional solution is \(x^*(p) = \rho/d\), for every \(p\). In this case the requirements on each interval in the PC-IS instance is \([(d + \rho - 1)/d]\). It follows that the weight of the computed solution is \(d \cdot [(d + \rho - 1)/d]\), while the weight of \(x^*\) is \(\rho\).

### 2.3 d-approximation Algorithm

We now present a \(d\)-approximation algorithm for PC-d-IS. This algorithm is presented here, despite the fact that its approximation ratio is not as good as the ratio that is obtained by the previous algorithm. In the next section it will serve as a building block for the algorithm for PARTIAL-d-IS.

Our algorithm starts by computing an optimal solution \((x^*, z^*)\) of \(LP-PC\) whose value is \(\text{OPT}^*\). Next, it constructs an PC-IS instance as follows. The set of points is the same as in the original PC-d-IS instance and their weights remain as they were. We define the set of intervals as follows. Let \(J_i\) be the most covered segment of \(I_i\), namely \(J_i = I_i^k\), where \(k = \arg \max_j \sum_{p \in I_i} x_p\). The requirement of \(J_i\) is \(r(I_i)\) and its weight is \(w'(J_i) = d \cdot w(I_i)\). Our algorithm invokes ALG on the new PC-IS instance, and computes an optimal solution \((x, z)\), where ALG is an algorithm for PC-IS that computes integral optimal solutions of LP-PC.

\((x, z)\) is a feasible solution, since \(I_i\) is stabbed at least as many times as \(J_i\). In the next lemma we prove that the algorithm computes \(d\)-approximations.

**Lemma 3.** \(\sum_p w(p)x(p) + d \cdot \sum_{I_i} w(I_i)z(I) \leq d \cdot \text{OPT}^*\).

**Proof.** We define a fractional solution \((\bar{x}, \bar{z})\) as follows: \(\bar{x}(p) = d \cdot x^*(p)\), for every \(p\), and \(\bar{z}(J_i) = z^*(I_i)\), for every \(i\). We show that this solution is feasible with respect to the PC-IS instance. The constraint for \(J_i\) is satisfied, since
\[
\sum_{p \in J_i} \bar{x}(p) + \bar{z}(J_i) = \sum_{p \in J_i} d \cdot x^*(p) + z^*(I_i) \geq \sum_{p \in I_i} x^*(p) + z^*(I_i) \geq r(I_i) .
\]
Furthermore, since the integrality gap of the PC-IS instance is 1, it follows that

\[
\sum_P w(p)x(p) + \sum_{I_i} d \cdot w(I_i)z(I_i) = \sum_P w'(p)x(p) + \sum_{J_i} w'(J_i)z(J_i) \\
\leq \sum_P w(p)x(p) + \sum_{J_i} w'(J_i)z(J_i) \\
= \sum_P d \cdot w(p)x^*(p) + \sum_I d \cdot w(I)z^*(I) \\
= d \cdot \text{OPT}^* ,
\]

and we are done.

\[\square\]

3 Partial $d$-interval Stabbing

In this section we present a \((\frac{4d}{3} + \varepsilon)\)-approximation algorithm for PARTIAL-$d$-IS. First, we present a general framework for approximating partial multicovering problems. We show how to design a \(\alpha\)-approximation algorithm, for any \(\varepsilon > 0\), for a partial multicovering problem using a Lagrangian multiplier preserving algorithm with factor \(\alpha\) for the prize collecting version of the problem (see Definition 1). Our framework is based on and extends the framework for partial covering problems by Köhann et al. [21].\(^1\) The \((\frac{4d}{3} + \varepsilon)\)-approximation algorithm for PARTIAL-$d$-IS is obtained using the Lagrangian multiplier preserving $d$-approximation algorithm for PC-$d$-IS from Section 2.3.

3.1 Partial Multicovering

In this section we extend the generic algorithm for partial covering problems from [21] to partial multicovering.

Given a set of elements \(U\) and their requirement function \(r : U \rightarrow \mathbb{N}\) and a collection of subsets \(S\), the partial multicovering problem can be formalized using the following linear integer program:

\[
\begin{align*}
\min & \quad \sum_{S \in S} w(S)x(S) \\
\text{s.t.} & \quad \sum_{S : u \in S} x(S) + z(u) \geq r(u) \quad \forall u \in U \\
& \quad \sum_{u \in U} z(u) \leq r(U) - t \\
& \quad x(S), z(u) \in \mathbb{N} \quad \forall S \in S, \ u \in U 
\end{align*}
\]

where \(r(U) \triangleq \sum_u r(u)\). The optimal solution of IP-PART is denoted by OPT. Observe that we may assume, without loss of generality, that \(w(S) > 0\) for all \(S \in S\).

Intuitively, the following reduction is at the heart of our algorithm for partial multicovering. Given a partial multicovering instance \((U, S, r, w, t)\) define the following partial covering instance

\[^1\]Köhann et al. [21] considered the case where each element has a profit and the goal is to cover elements so as to gain at least \(t\) profit. In this paper we consider the unit profit case to avoid confusion, but our results extend to the more general case where we gain such a profit for every unit of satisfied requirement.
\((U', S', w', t)\). The new set of elements \(U'\) is obtained from \(U\) by replacing every element \(u \in U\) with \(r(u)\) copies of \(u\), i.e., \(U' = \bigcup_{u \in U} U'_i\) where \(U'_i = \{u_{i,k} : 1 \leq k \leq r(u_i)\}\). In addition, each subset \(S \in \mathcal{S}\) has an \textit{incarnation} for every possible combination of copies of its elements. More formally, for every subset \(S = \{u_{i_1}, \ldots, u_{i_l}\} \in \mathcal{S}\) the collection \(S'\) contains all possible subsets \(S' \subseteq U'_1 \times \cdots \times U'_l\). The cost of every such subset \(S'\) is \(w'(S') = w(S)\). The above reduction cannot be carried out in polynomial time, since the requirements may be too large. However, in what follows we show that there is no need to actually perform the reduction explicitly in order to extend the framework from [21] to partial multicovering. Intuitively, our arguments are base on the fact that all elements in \(U'_i\) are interchangeable from the viewpoint of the original subsets.

### 3.1.1 Preprocessing

We start with a preprocessing phase. Let \(k = \lceil 1/\varepsilon \rceil\). We guess the \(k\) most expensive copies of subsets from \(\mathcal{S}\) that take part in an optimal solution. More specifically, we enumerate all \(\sum_{j=0}^{k} (S+j-1) = O(k(|S| + k - 1)^k)\) possible vectors \(x_0 \in \{0, \ldots, k\}^{|S|}\) such that \(\sum_{S} x_0(S) \leq k\). For each such guess \(x_0\), we check whether \((x_0, z_0)\) it is feasible with respect to \textsc{ip-part}, where \(z_0(u) = r(u) - \sum_{S : u \in S} x_0(S)\). If it is we move to the next guess. Otherwise, we update \(r\) and the covering requirement \(t\), according to \(x_0\), namely \(r(u)\) is decreased by \(\sum_{S : u \in S} x_0(S)\), and \(t\) is decreased by \(\sum_{S} \sum_{u \in S} x_0(S)\). Also, any subset whose weight is greater than \(\min_{S} : x_0(S) > 0\ w(S)\) is eliminated. It is not hard to verify that the weight of each remaining subset is at most \(\varepsilon \cdot \text{OPT}\). In this case we continue to the next step of the algorithm. After testing all possible guesses, we return the best solution we find.

### 3.1.2 Lagrangian Relaxation

Next, we lift the partial requirement constraint to the objective function multiplied by \(\lambda > 0\). The resulting Lagrangian relaxation is:

\[
\min_{S \in \mathcal{S}} \sum_{S : u \in U} w(S)x(S) + \lambda \left( \sum_{u \in U} z(u) - (r(U) - t) \right)
\]

\text{(IP-LR(\(\lambda\)))}

s.t.

\[
\sum_{S : u \in S} x(S) + z(u) \geq r(u) \quad \forall u \in U
\]

\[
x(S), z(u) \in \mathbb{N} \quad \forall S \in \mathcal{S}, u \in U
\]

Observe that we get a prize collecting multicovering instance \(I_{\lambda}\) with uniform penalties and an additional additive term in the objective function. Denote the optimal value of \textsc{ip-lr}(\(\lambda\)) by \(\text{LR}(I_{\lambda})\) and let \(\text{OPT}(I_{\lambda}) = \text{LR}(I_{\lambda}) + \lambda (r(U) - t)\).

It is not hard to show that the optimum of \textsc{ip-lr}(\(\lambda\)) is not worse that the optimum of \textsc{ip-part}:

**Observation 4.** \(\text{LR}(I_{\lambda}) = \text{OPT}(I_{\lambda}) - \lambda (r(U) - t) \leq \text{OPT}, \text{ for any } \lambda \geq 0.\)

**Proof.** First, observe that any solution \((x, z)\) of \textsc{ip-part} is also a feasible solution of \textsc{ip-lr}(\(\lambda\)). Moreover, since \(\sum_{u \in U} z(u) \leq r(U) - t\), the cost of \((x, z)\) with respect to \textsc{ip-lr}(\(\lambda\)) is smaller than its cost with respect to \textsc{ip-part} by an additive factor of \(\lambda \left( \sum_{u \in U} z(u) - (r(U) - t) \right) \leq 0.\)

We solve \(I_{\lambda}\) for various values of \(\lambda\) using the following type of algorithm:

8
**Definition 1.** Let \( \Pi \) be a multicovering problem. A polynomial time algorithm for the prize collecting version of \( \Pi \) is called Lagrangian multiplier preserving with factor \( \alpha \) (or, \( \alpha \)-LMP) if for every instance \((U, S, r, w)\) it computes a solution \((x, z)\) that satisfies

\[
\sum_{S \in \mathcal{S}} w(S)x(S) + \alpha \cdot \sum_{u \in U} w(u)z(u) \leq \alpha \cdot \text{OPT}(\Pi) .
\]  

We provide bounds on the interesting values of \( \lambda \).

**Lemma 5.** Let \( \text{alg} \) be an \( \alpha \)-LMP algorithm, and let \((x^\lambda, z^\lambda)\) be the solution computed by \( \text{alg} \) on \( \mathcal{I}_\lambda \). If \( \lambda > r(U)w(S) \), then \((x^\lambda, z^\lambda)\) satisfies all requirements, while \((x^0, z^0)\) does not satisfy any requirement.

**Proof.** Suppose that \( \lambda > r(U)w(S) \) and there is an element \( u \) for which \( z(u) \geq 1 \). Then,

\[
\sum_{S \in \mathcal{S}} w(S)x^\lambda(S) + \alpha \cdot \sum_{u \in U} \lambda z^\lambda(u) \geq \alpha \lambda > \alpha r(U)w(S) \geq \alpha \text{OPT}(\mathcal{I}_\lambda)
\]

in contradiction to (1). On the other hand, \( \text{OPT}(\mathcal{I}_0) = 0 \), and thus we must have \( x^0 = 0 \). \( \square \)

Now, using an \( \alpha \)-LMP algorithm \( \text{alg} \) for the prize collecting version of the multicovering problem, we search for \( \lambda_1 \) and \( \lambda_2 \) that satisfy:

1. \( 0 \leq \lambda_1 - \lambda_2 \leq \varepsilon w_{\min}/r(U) \), where \( w_{\min} = \min_u w(u) \).
2. A total requirement of \( t_1 \geq t \) is satisfied by \((x^{\lambda_1}, z^{\lambda_1})\) and a total requirement of \( t_2 \leq t \) is satisfied by \((x^{\lambda_2}, z^{\lambda_2})\).

Observe that this can be done in polynomial time, since \( \lambda_1 \in [0, r(U)w(S)] \).

Now if \( t_1 = t \) or \( t_2 = t \), an \( \alpha \)-approximate solution can be obtained.

**Lemma 6.** If \( t_1 = t \), then \((x^{\lambda_1}, z^{\lambda_1})\) is \( \alpha \)-approximate with respect to IP-PART. Also, if \( t_2 = t \), then \((x^{\lambda_2}, z^{\lambda_2})\) is \( \alpha \)-approximate with respect to IP-PART.

**Proof.** Since \( \text{alg} \) is \( \alpha \)-LMP we have that

\[
\sum_{S} w(S)x^{\lambda_1}(S) + \alpha \sum_{u} w(u)z^{\lambda_1}(u) \leq \alpha \text{OPT}(- \mathcal{I}_\lambda) .
\]

It follows that

\[
\sum_{S} w(S)x^{\lambda_1}(S) \leq \alpha \left( \text{OPT}(\mathcal{I}_{\lambda_1}) - \sum_{u} w(u)z^{\lambda_1}(u) \right)
\]

\[
= \alpha \left( \text{OPT}(\mathcal{I}_{\lambda_1}) - \lambda_1 (r(U) - t_1) \right)
\]

\[
= \alpha \left( \lambda_1 (t_1 - t - (r(U) - t_1)) \right)
\]

\[
\leq \alpha \left( \text{OPT} + \lambda_1 (t_1 - t) \right) .
\]

And if \( t_1 = t \), we get that \( x^{\lambda_1} \) is \( \alpha \)-approximate. A similar argument can be used to show that

\[
\sum_{S} w(S)x^{\lambda_2}(S) \leq \alpha \left( \text{OPT} + \lambda_2 (t_2 - t) \right) ,
\]

and therefore \( x^{\lambda_2} \) is \( \alpha \)-approximate, if \( t_2 = t \). \( \square \)
If $t_2 < t < t_1$, then a linear combination of our two solutions constitute is $(1 + \varepsilon)\alpha$-approximate.

**Lemma 7.** If $t_2 < t < t_1$, then

$$\beta \sum_S w(S)x^{\lambda_1}(S) + (1 - \beta) \sum_S w(S)x^{\lambda_2}(S) \leq \alpha(1 + \varepsilon)\text{OPT},$$

where $\beta = \frac{t_1 - t}{t_1 - t_2} \in (0, 1)$.

**Proof.** Using Equations (2) and (3) we get that

$$\beta \sum_S w(S)x^{\lambda_1}(S) + (1 - \beta) \sum_S w(S)x^{\lambda_2}(S)$$

$$\leq \beta \alpha(\text{OPT} + \lambda_1(t_1 - t)) + (1 - \beta) \alpha(\text{OPT} + \lambda_2(t_2 - t))$$

$$\leq \alpha \left( \text{OPT} + \beta(\lambda_2 + \frac{\varepsilon w_{\text{min}}}{r(U)})(t_1 - t) + (1 - \beta)(t_2 - t) \right)$$

$$= \alpha \left( \text{OPT} + \lambda_2(t_1 - t) + (1 - \beta)(t_2 - t) + \beta \frac{\varepsilon w_{\text{min}}(t_1 - t)}{r(U)} \right)$$

$$\leq \alpha(\text{OPT} + \varepsilon w_{\text{min}})$$

$$= \alpha(1 + \varepsilon)\text{OPT},$$

where the last inequality follows from $\beta(t_1 - t) + (1 - \beta)(t_2 - t) = 0$ and $t_1 - t \leq r(U)$. \hfill \Box

### 3.1.3 Augmentation

If $t_2 < t < t_1$, we compute a feasible solution $(x', z')$ by augmenting $(x^{\lambda_2}, z^{\lambda_2})$ using subsets $S$ that satisfy $x^{\lambda_1}(S) - x^{\lambda_2}(S) > 0$.

In [21] a partial cover $x'$ is constructed by augmenting $x^{\lambda_2}$ as follows. First, elements that are uncovered by the solution $x^{\lambda_2}$ are assigned to subsets $S$ such that $x^{\lambda_2}(S) = 0$ and $x^{\lambda_1}(S) = 1$. Then, subsets are added to $x'$ according a non-decreasing order of $w(S)/\varphi(S)$, where $\varphi(S)$ denotes the number of elements assigned to $S$. We extend this approach to multicovering in the following manner. For each element $u$ let $\varphi_u : S \rightarrow \mathbb{N}$ be a function that satisfies:

1. $0 \leq \varphi_u(S) \leq \max \{x^{\lambda_1}(S) - x^{\lambda_2}(S), 0\}$, for every $S$.

2. $\sum_S \varphi_u(S) \leq z^{\lambda_2}(u)$.

Roughly speaking, $z^{\lambda_2}(u)$ can be viewed as the number of uncovered “copies” of $u$ with respect to $x^{\lambda_2}$, and $\varphi_u(S)$ can be seen an assignment of such copies to subset incarnations that are used by $x^{\lambda_1}$, but are not used by $x^{\lambda_2}$.

**Lemma 8.** There exists a set of polynomial-time computable functions $\{\varphi_u\}_{u \in U}$ that satisfy the above conditions and also satisfy $\sum_u \sum_S \varphi_u(S) > t - t_2$.

**Proof.** A function $\varphi_u$ that satisfy the above conditions can be computed as follows. We go over the subsets from $S_1$ to $S_n$, and assign $\varphi_u(S_i) \leftarrow \max \{x^{\lambda_1}(S_i) - x^{\lambda_2}(S_i), 0\}$, as long as the inequality $\sum_{j=1}^i \varphi_u(S_j) \leq z^{\lambda_2}(u)$ continues to hold. Otherwise, we assign $\varphi_u(S_i) \leftarrow z^{\lambda_2}(u) - \sum_{j=1}^{i-1} \varphi_u(S_j)$ and $\varphi_u(S_j) \leftarrow 0$ for all $j > i$. The running time to compute $\varphi_u$ is $O(|S|)$. 

---

"Roughly speaking, \( z^{\lambda_2}(u) \) can be viewed as the number of uncovered "copies" of \( u \) with respect to \( x^{\lambda_2} \), and \( \varphi_u(S) \) can be seen an assignment of such copies to subset incarnations that are used by \( x^{\lambda_1} \), but are not used by \( x^{\lambda_2} \)."
To prove that $\sum_u \sum_S \varphi_u(S) > t - t_2$ observe that, for every $u$, we have that either $\sum_S \varphi_u(S) = z^{\lambda_2}(u)$ or

$$\sum_S \varphi_u(S) = \sum_S \max \left\{ x^{\lambda_1}(S) - x^{\lambda_2}(S), 0 \right\} \geq \sum_S x^{\lambda_1}(S) - \sum_S x^{\lambda_2}(S) \geq [r(u) - z^{\lambda_1}(u)] - [r(u) - z^{\lambda_2}(u)] = z^{\lambda_2}(u) - z^{\lambda_1}(u),$$

where the second inequality is because $\sum_S x^{\lambda_2}(S) + z^{\lambda_2}(u) = r(u)$, if $z^{\lambda_2}(u) > 0$.

It follows that

$$\sum_{u \in U} \sum_{S \in S} \varphi_u(S) \geq \sum_{u \in U} z^{\lambda_2}(u) - \sum_{u \in U} z^{\lambda_1}(u) = (r(U) - t_2) - (r(U) - t_1) = t_1 - t_2 > t - t_2,$$

and the lemma follows. \qed

Let $\Delta_S(i)$ be the number of elements in $U$ that have at least $i$ copies assigned to $S$, namely we define $\Delta_S(i) = |\{ u : \varphi_u(S) \geq i \}|$. $\Delta_S(i)$ stands for the additional coverage we obtain by increasing $x'(S)$ from $i - 1$ to $i$.

**Observation 9.** Let $S \in S$. $\Delta_S$ satisfies the following properties:

1. $\Delta_S(i) \in \{0, \ldots, |S|\}$.
2. $\Delta_S$ is a non-decreasing function, i.e., $\Delta_S(i + 1) \leq \Delta_S(i)$, for every $i$.
3. We can compute in polynomial time the values of $i$ for which $\Delta_S(i + 1) > \Delta_S(i)$.

Next, we define a full order $< \! \! <$ on pairs $(S, i)$ as follows: $(S_j, i) \prec (S_{j'}, i')$ if

1. $w(S_j)/\Delta_{S_j}(i) < w(S_{j'})/\Delta_{S_{j'}}(i')$.
2. $w(S_j)/\Delta_{S_j}(i) = w(S_{j'})/\Delta_{S_{j'}}(i')$ and $j < j'$.
3. $w(S_j)/\Delta_{S_j}(i) = w(S_{j'})/\Delta_{S_{j'}}(i')$ and $j = j'$ and $i < i'$.

**Observation 10.** The full order $< \! \! <$ consists of at most $|S|(|U| + 1)$ blocks of the form $(S_j, i), (S_j, i + 1), \ldots, (S_j, i')$, for $i \leq i'$.

Now, starting with $x' = x^{\lambda_2}$, we augment $x'$ according to the above order. If we ignore the running time, we may pick in each step the next pair $(S, i)$ and increase $x'(S)$ by 1. Notice that in this case $x'(S) = i - 1$, since otherwise $(S, i - 1)$ could have been chosen. We stop increasing $x'(S)$, when $x'$ reaches the covering requirement $t$. This process terminates, since $\sum_u \sum_S \varphi_u(S) > t - t_2$.

Let $(S_0, q)$ denote the last pair that increased $x'$. To obtain polynomial running time, we notice that Observation 10 implies that we may increase $x'$ using at most $|S||U|$ non-unit advancement steps. In each such an advancement step we actually have $|S|$ options, and each option corresponds to a block of the form $(S_j, i), (S_j, i + 1), \ldots, (S_j, i')$, for $i \leq i'$. Hence, we add all the pairs in the block, or we add as many as we need to reach $t$.\[\]
Lemma 11. Let $\beta = \frac{t_1 - t_2}{t_1 - t_2}$. Then,

$$
\sum_S w(S)x'(S) \leq \sum_S w(S)x^{\lambda_2}(S) + \beta \sum_S w(S) \max \left\{ x^{\lambda_1}(S) - x^{\lambda_2}(S), 0 \right\} + \varepsilon_{\text{OPT}}.
$$

Proof. First, recall that due to the preprocessing phase we know that $w(S) \leq \varepsilon_{\text{OPT}}$ for every $S$. Therefore, it is sufficient to prove that

$$
\sum_S w(S)x''(S) - w(S_0) \leq \beta \sum_S w(S) \max \left\{ x^{\lambda_1}(S) - x^{\lambda_2}(S), 0 \right\},
$$

where $x'' = x' - x^{\lambda_2}$. (Recall that $S_0$ is the last set to enter $x'$.)

Consider a random variable $K$, where

$$
\Pr[K = (S, i)] = \frac{\Delta_S(i)}{\sum_{(S', i')} \Delta_{S'}(i')}.
$$

Also, define $R = w(S)/\Delta_S(i)$ if $K = (S, i)$. Since the pairs are ordered in nondecreasing order of $w(S)/\Delta_S(i)$ we have that

$$
\mathbb{E}[R | K \prec (S_0, q)] \leq \mathbb{E}[R].
$$

Now

$$
\mathbb{E}[R] = \sum_{(S, i)} \frac{w(S)}{\Delta_S(i)} \cdot \frac{\Delta_S(i)}{\sum_{(S', i')} \Delta_{S'}(i')}
$$

$$
= \frac{\sum_{(S, i)} w(S)}{\sum_{(S', i')} \Delta_{S'}(i')}
$$

$$
= \frac{\sum_{(S, i)} w(S)}{\sum_u \sum_S \varphi_u(S)}
$$

$$
\leq \frac{1}{t_1 - t_2} \sum_S w(S) \max \left\{ x^{\lambda_1}(S) - x^{\lambda_2}(S), 0 \right\},
$$

and

$$
\mathbb{E}[R | K \prec (S_0, q)] = \sum_{(S, i) \prec (S_0, q)} \frac{w(S)}{\Delta_S(i)} \cdot \frac{\Delta_S(i)}{\sum_{(S', i') \prec (S_0, q)} \Delta_{S'}(i')}
$$

$$
= \frac{\sum_{(S, i) \prec (S_0, q)} w(S)}{\sum_{(S', i') \prec (S_0, q)} \Delta_{S'}(i')}
$$

$$
\geq \frac{1}{t_1 - t_2} \left( \sum_S w(S)x''(S) - w(S_0) \right).
$$

Hence, we obtain that

$$
\sum_S w(S)x''(S) - w(S_0) \leq \beta \sum_S w(S) \max \left\{ x^{\lambda_1}(S) - x^{\lambda_2}(S), 0 \right\},
$$

as required. \(\square\)
3.1.4 The Computed Solution

Finally, we show that either \( x^{\lambda_1} \) or \( x' \) is the desired approximate solution.

\[
\text{Lemma 12. } \min \left\{ \sum_s w(S)x^{\lambda_1}(S), \sum_s w(S)x'(S) \right\} \leq \left( \frac{4}{3} + O(\sqrt{\varepsilon}) \right) \alpha \text{OPT}.
\]

Proof. By Lemma 7 we have

\[
\sum_s w(S)x^{\lambda_1}(S) \leq \frac{1}{\beta} \left( \alpha(1 + \varepsilon) \text{OPT} - (1 - \beta) \sum_s w(S)x^{\lambda_2}(S) \right) = \frac{\alpha(1 + \varepsilon) - (1 - \beta)\rho}{\beta} \text{OPT},
\]

where \( \rho \triangleq \frac{\sum_s w(S)x^{\lambda_2}(S)}{\text{OPT}} \in [0, \alpha] \). On the other hand, Lemma 11 implies that

\[
\sum_s w(S)x'(S) \leq \sum_s w(S)x^{\lambda_2}(S) + \beta \sum_s w(S) \max \left\{ x^{\lambda_1}(S) - x^{\lambda_2}(S), 0 \right\} + \varepsilon \text{OPT}
\]
\[
\leq \sum_s w(S)x^{\lambda_2}(S) + \beta \sum_s w(S)x^{\lambda_1}(S) + +\varepsilon \text{OPT}
\]
\[
\leq \beta(1 - \varepsilon) \text{OPT} + \beta \sum_s w(S)x^{\lambda_2}(S) + \varepsilon \text{OPT}
\]
\[
= (\beta(1 - \varepsilon) + \beta \rho + \varepsilon) \text{OPT},
\]

where the third inequality is due to Lemma 7 and the definition of \( \rho \).

The lemma follows, since

\[
\max_{\beta \in (0,1]} \min_{\rho \in [0,\alpha]} \left\{ \frac{\alpha(1 + \varepsilon) - (1 - \beta)\rho}{\beta}, \beta(1 - \varepsilon) + \beta \rho + \varepsilon \right\} \leq \left( \frac{4}{3} + O(\sqrt{\varepsilon}) \right) \alpha
\]

due to Lemma 6 in [21].

This leads to the main result of the section:

\[
\text{Theorem 2. Let } \Pi \text{ be a multicovering problem. If there exists an } \alpha\text{-LMP algorithm for the prize collecting version of } \Pi, \text{ then there exists a polynomial time } \left( \frac{4}{3} + \varepsilon \right) \alpha\text{-approximation algorithm for the partial version of } \Pi, \text{ for any } \varepsilon > 0.
\]

3.2 Partial \( d \)-interval Stabbing

Observe that in Section 2.3 we presented a \( d \)-LMP algorithm for PC-\( d \)-IS. Hence, by Theorem 2 we get the following result:

\[
\text{Theorem 3. There exists a polynomial time } \left( \frac{4d}{3} + \varepsilon \right)\text{-approximation algorithm for PARTIAL-} d\text{-IS, for any } \varepsilon > 0.
\]
4 Maximum Prize Collecting $d$-interval Stabbing

In this section we consider the maximization version of prize collecting $d$-is. We show that MAX-PC-$d$-IS is as hard to approximate as MAXIMUM INDEPENDENT SET even in the special case of $d = 2$. Note that MAXIMUM INDEPENDENT SET cannot be approximated within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ in polynomial time, unless NP=ZPP [18]. On the positive side, we present a $d$-approximation algorithm for maximum prize collecting $d$-rs. Our algorithm basically computes an optimal solution for each axis and chooses the solution with maximum weight.

4.1 LP Formulation

MAX-PC-$d$-IS (and MAX-PC-$d$-RS) can be formalized using the following linear integer program.

\[
\begin{align*}
\text{max} & \quad \sum_{I \in \mathcal{I}} w(I)(r(I) - z(I)) - \sum_{p \in P} w(p)x(p) \\
\text{s.t.} & \quad \sum_{p \in I} x(p) + z(I) \geq r(I) \quad \forall I \in \mathcal{I} \\
& \quad x(p), z(I) \in \mathbb{N} \quad \forall p \in P, I \in \mathcal{I}
\end{align*}
\]

(IP-MAXPC)

where $x(p)$ stands for the number of copies of $p$ in the solution, and $z(I)$ is the unsatisfied requirement of $I$. The LP-relaxation of IP-MAXPC is obtained by replacing the integrality constraints by: $x(p) \geq 0$, for every $p \in P$, and $z(I) \geq 0$, for every $I \in \mathcal{I}$. We denote the LP relaxation by LP-MAXPC.

Observe that apart from the objective function IP-MAXPC and IP-PC are identical. Furthermore, the sums of the objective functions is $\sum_I w(I)r(I)$. Hence, a polynomial time algorithm for IP-PC implies a polynomial algorithm for IP-MAXPC. It follows that MAX-PC-IS (i.e., MAX-PC-$d$-IS with $d = 1$) can be solved in polynomial time.

4.2 Hardness of Approximation

We present a polynomial time approximation preserving reduction from MAXIMUM INDEPENDENT SET to MAX-PC-$2$-IS. It follows that MAX-PC-$2$-IS is as hard to approximate as MAXIMUM INDEPENDENT SET.

Let $G = (V, E)$ be a simple graph, where $V = \{v_1, \ldots, v_n\}$. Without loss of generality we assume that $\deg(v_i) \geq 1$ for every $i$. (We can ignore isolated vertices, since they are contained in any optimal solution.) We construct a MAX-PC-$2$-IS instance $(P, \mathcal{I}, r, w)$ as follows. First, for every vertex $v_i$ we add the point $i$ to $P$, namely $P = \{i : v_i \in V\}$. Next, for every edge $e \in E$ we define a 2-interval of the form $I_e = \{[i, i] : v_i \in e\}$. Note that each such 2-interval actually consists of two points. The covering requirement of all interval is 1, namely $r(I_e) = 1$ for all $e$. We also define $w(i) = \deg(v_i) - 1$, for every $i$, and $w(I_e) = 1$, for every $e \in E$.

Given a set if points $H \subseteq P$ we denote the objective function of IP-MAXPC for the case of unit requirements by $\bar{w}(H)$, namely

\[
\bar{w}(H) = \sum_{I : \cap H \neq \emptyset} w(I) - \sum_{p \in H} w(p)
\]

We are now ready to prove the correctness of our reduction.
Lemma 13. Let \( U \subseteq V \) be an independent set in \( G \). Then there exists a solution \( H_U \) with respect to the \( \text{max-pc-2-is} \) instance such that \( \bar{w}(H_U) = |U| \).

Proof. Define \( H_U = \{ i : v_i \in U \} \). Then, \( \bar{w}(H_U) = |U| \). Since \( U \) is an independent set, each a 2-interval is stabbed by \( H_U \) at most once. Hence,

\[
\bar{w}(H_U) = |\{ I : I \cap H_U \neq \emptyset \}| - \sum_{u \in U} (\deg(u) - 1) = \sum_{u \in U} \deg(u) - \sum_{u \in U} (\deg(u) - 1) = |U|,
\]
as required.

Lemma 14. Let \( H \) be a feasible solution with respect to the \( \text{max-pc-2-is} \) instance. Then there exists an independent set \( U_H \) in \( G \) such that \( |U_H| \geq \bar{w}(H) \).

Proof. First, assume that \( H \) induces an independent set, namely that if \( i, j \in H \) then \( \{ i, j \} \notin \mathcal{I} \), or \( (v_i, v_j) \notin E \). In this case, \( U_H = \{ v_i : i \in H \} \) is an independent set and \( |U_H| = \bar{w}(H) \).

If \( H \) does not induce an independent set, we show how to construct a solution \( H' \) such that \( \bar{w}(H') \geq \bar{w}(H) \) and \( H' \subseteq H \). Assume that there are \( i, j \in H \) such that \( I_e = \{ i, j \} \in \mathcal{I} \). Let \( H' = H \setminus \{ j \} \). We show that \( \bar{w}(H') \geq \bar{w}(H) \):

\[
\bar{w}(H') = |\{ I : I \cap H' \neq \emptyset \}| - \sum_{i \in H'} w(i)
\geq |\{ I : I \cap H \neq \emptyset \}| - (\deg(v_j) - 1) + w(j) - \sum_{i \in H} w(i)
= \bar{w}(H).
\]

We can iteratively remove points from \( H \) until it induces an independent set. The lemma follows.

Lemmas 13 and 14 lead to the following hardness result:

Theorem 4. \( \text{max-pc-2-is} \) at least as hard to approximate as \( \text{maximum independent set} \).

Proof. By Lemmas 13 and 14 the size of the maximum independent set in \( G \) is equal to the optimum of the \( \text{max-pc-2-is} \) instance. Suppose we have an \( r \)-approximation algorithm for \( \text{max-pc-2-is} \) denoted by \( \text{alg} \). We design an algorithm for \( \text{maximum independent set} \) as follows. We construct a \( \text{max-pc-2-is} \) instance using the reduction, and compute an \( r \)-approximate solution \( H \) using \( \text{alg} \). Then, using Lemma 14 we obtain an \( r \)-approximate solution \( H' \) that corresponds to an independent set \( U \). It follows that \( U \) is an \( r \)-approximate independent set.

We note that if the \( \text{max-pc-2-is} \) instance \( (P, \mathcal{I}, r, w) \) created by our reduction is an \( \text{RS} \) instance, then the given graph is bipartite. It follows that our reduction can not be used in the case of \( \text{max-pc-d-rs} \). Indeed, in the next section we provide a \( d \)-approximation algorithm for \( \text{max-pc-d-rs} \).

4.3 \( d \)-dimensional Rectangle Stabbing

In this section we present a \( d \)-approximation algorithm for \( \text{max-pc-d-rs} \). The algorithm computes an optimal solution of the one dimensional instance induced by each axis, and chooses a solution.
of maximum weight. The analysis of the approximation ratio relies on an optimal solution of the LP relaxation of the problem, and the integrality of $\text{max}-\text{pc-is}$.

Given an $\text{max-}d\text{-rs}$ instance, we define $P^i$ to be the set of hyper-planes orthogonal to the $i$th axis. Also, let $w_I$ and $w_P$ denote the weights of the rectangles and lines, respectively.

Let $(x^*, z^*)$ be an optimal solution of $\text{LP-maxpc}$. We define $d$ solutions $(x^i, z^i)$, for $i \in \{1, \ldots, d\}$, where $(x^i, z^i)$ is defined as follows: $x^i(p) = x^*(p)$ if $p \in P^i$, and $x^i(p) = 0$ otherwise, and $z^i(I) = \max\{r(I) - \sum_{p \in P^i} x^i(p), 0\}$. That is, $x^i$ is the projection of $x^*$ on the $i$th axis, and $z^i$ is the unsatisfied stabbing requirement using the $i$th axis. Observe that $(x^i, z^i)$ is a feasible solution for every $i$.

**Lemma 15.** There exists an axis $i$ for which $w_I \cdot (r - z^i) - w_P \cdot x^i \geq \frac{1}{d}(w_I \cdot (r - z^*) - w_P \cdot x^*)$.

**Proof.** $\sum_{i=1}^d x^i = x^*$ by definition. Also, $\sum_{i=1}^d (r - z^i) \geq (r - z^*)$, since $\sum_{p \in I} x^*(p)$ may be greater than 1. It follows that

$$\sum_{i} w_I \cdot (r - z^i) - \sum_{i} w_P \cdot x^i \geq w_I \cdot (r - z^*) - w_P \cdot x^*,$$

and the lemma follows. □

**Theorem 5.** There exists a $d$-approximation algorithm for $\text{max-}d\text{-rs}$.

**Proof.** We use Lemma 15 to design an approximation algorithm for $\text{max-}d\text{-rs}$. We simply solve the one dimensional instance induced by $P^i$, for every $i$, and output the solution of maximum weight. We denote this solution by $(x, z)$. □

We show that the analysis is tight using an instance that contains one line $p_i$ for each dimension, and $d$ rectangles denoted by $I_1, \ldots, I_d$, where $I_i$ is stabbed only by $p_i$. The weight of all the lines is 1, and the prize of all the rectangles is 2. The optimal solution contains all the lines and its weight is $d$, while the algorithm picks only one line and its weight is 1.

**Acknowledgments**

We thank Arie Tamir for helpful discussions. We also thank the anonymous referees for their helpful comments and suggestions.

**References**


17