Capacitated Arc Stabbing

Dror Rawitz*  
rawitz@eng.tau.ac.il

Shimon (Moni) Shahar*  
monishahar@gmail.com

September 24, 2012

Abstract

In the Capacitated Arc Stabbing problem (CAS) we are given a set of arcs and a set of points on a circle. We say that a point \( p \) covers, or stabs, an arc \( A \) if \( p \) is contained in \( A \). Each point has a weight and a capacity that determines the number of arcs it may cover. The goal is to find a minimum weight set of points that stabs all the arcs. CAS models a periodic multi-item lot sizing problem in which we are given a set of production requests each with its own periodic release time and deadline. Production takes place in batches of bounded capacity: each time unit \( t \) is associated with a capacity \( c(t) \) and weight \( w(t) \), where \( c(t) \) bounds the number of requests that can be manufactured at time \( t \), and \( w(t) \) is a fixed cost for manufacturing any positive number of requests up to \( c(t) \) at time \( t \). The goal is to find a minimum weight periodic schedule. We present a polynomial time algorithm for CAS that is based on a non-trivial reduction to Capacitated Interval Stabbing. Our approach applies to both hard and soft capacities. We also consider two variants of CAS in which some arcs may remain uncovered: in the partial variant there is a covering requirement \( g \), and the goal is to find a minimum weight set of points that covers at least \( g \) arcs; and in the prize collecting variant each arc has a penalty that must be paid if this arc is not covered.

Keywords: arc stabbing, capacitated covering, interval stabbing, lot sizing, partial covering, prize collecting covering.

1 Introduction

In the (uncapacitated) Arc Stabbing problem we are given a set \( A \) of \( n \) arcs and a set \( P \) of \( m \) points on a circle. We say that a point \( p \) covers or stabs an arc \( A \) if \( p \in A \). Each point \( p \in P \) has a weight \( w(p) \), and our goal is to find a minimum weight set of points \( H \subseteq P \) that covers all arcs in \( A \). Interval Stabbing is the special case of Arc Stabbing, where the circle is not fully covered by arcs, and therefore arcs correspond to intervals on the real line. In other words, in Interval Stabbing we are given a set of intervals and a set of weighted points on the line, and the goal is to find a minimum set of points that covers all the intervals.

In this paper we introduce an extension of Arc Stabbing called Capacitated Arc Stabbing (abbreviated CAS). In CAS each point \( p \in P \) has a capacity \( c(p) \in \mathbb{N} \) that determines the number of arcs it may cover. In the hard capacities case one may take only one copy of each point, while in the soft capacities case multiple copies of a point may be used to cover additional arcs, provided

*School of Electrical Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel.
that the weight of \( p \) is counted for each copy. A feasible solution is an assignment of every arc to a point. (Formal definitions are given in Section 2.) **Arc Stabbing** is the special case where \( c(p) = \infty \) for every point \( p \).

CAS models the following periodic multi-item lot sizing problem. We are given a repeating period consisting of \( T \) discrete time units indexed \( t = 0, \ldots, T - 1 \), and a set of unit size production **requests** denoted by \( R_1, \ldots, R_n \), where each request has periodic release time \( r_i \) and deadline \( d_i \). Production takes place in mixed batches of bounded capacity. Each time unit \( t \) is associated with a capacity \( c(t) \) and weight \( w(t) \), where the number of requests that can be manufactured at time unit \( t \) is at most \( c(t) \), and \( w(t) \) is a fixed manufacturing cost at time \( t \) for any positive number of requests up to \( c(t) \). A feasible solution is a periodic production schedule in which \( R_i \) is produced exactly once during every period within the arc \([r_i, d_i]\). Moreover, \( R_i \) must be produced at the same time within each period. The goal is to find a minimum weight periodic schedule.

Another possible application of CAS is event coverage by sensors located on a closed fence surrounding a convex area. Events take place outside the perimeter, and each event can be covered by any sensor that has a clear view of the event. Hence, sensors that can cover an event are located on some contiguous segment of the fence. Each sensor has a cap on the number of events that it can cover and an operation cost, and the goal is to cover all events at minimum cost.

We also study two extensions of CAS, in which arcs may remain uncovered. In Partial CAS we are given a covering requirement \( g \), and the goal is to find a minimum weight assignment of at least \( g \) arcs to points. (If \( g = |A| \), we return to CAS.) In the periodic lot sizing terminology this means that we would like to find a periodic schedule that satisfies at least \( r \) requests. In Prize Collecting CAS we are not obligated to cover all arcs, but must pay penalties for uncovered arcs. More formally, both points and arcs have non-negative weights, and the weight of a feasible solution is the total weight of its points plus the total weight of uncovered arcs. (If all penalties are large enough, say \( w(A) \geq \max_{p \in A} w(p) \), the problem reduces to CAS.) Alternatively, we seek a periodic schedule that satisfies a subset of the requests, and the cost of the schedule consists of production cost and penalties for unsatisfied requests.

**Our results.** We present a polynomial time algorithm for CAS that is based on a reduction to **Capacitated Interval Stabbing** (**CIS**) that can be solved in polynomial time [11]. Our reduction is somewhat complicated, and it is based on a characterization of an optimal solution. More specifically, we show that there exists an optimal solution that adheres to a limited and circular version of the *earliest due date* rule. The reduction applied to both hard and soft capacities.

We note that naive attempts to “break the circle” fail. For instance, “guessing” a point \( p \) that participates in an optimal solution works well for Arc Stabbing, but not for CAS. The problem is that, after breaking the circle at \( p \), one has to decide for every arc that contains \( p \) whether it should be covered by \( p \), by points on the left side of the resulting line or by points on the right side of this line. (This is relatively easy if no arc is contained in another, namely in proper circular-arc graphs.) Furthermore, in the soft capacities case, preferring cost-effective points (i.e., points with low cost per covered interval) leads to a 2-approximation algorithm even in the case of CIS [12].

We also present polynomial time algorithms for Partial CAS. The presentation is composed of two parts. First, we provide a polynomial time algorithm that solves Partial CIS that uses dynamic programming and is based on the algorithm for CIS from [11]. Second, we show that the above reduction can be extended to the partial setting. A similar approach is used to design a polynomial time algorithm for Prize Collecting CAS.
Related work. INTERVAL STABBLING is the well known CLIQUE COVER problem in interval graphs that can be solved in polynomial time [17, page 174]. The prize collecting version of INTERVAL STABBLING is solvable in $O(n \log n)$ time [21]. ARC STABBLING is a variant of CLIQUE COVER in circular-arc graphs in which one is only allowed to use point-induced cliques, and it can be solved in polynomial time using the above mentioned reduction to INTERVAL STABBLING (i.e., “breaking the circle”). Hochbaum and Levin [22] gave an $O(mn + n^2 \log n)$ time algorithm for solving unweighted ARC STABBLING. Hsu and Tsai [23] showed that CLIQUE COVER in circular-arc graphs can be solved in linear time.

In the CIRCLE COVER problem the input is a set of $n$ arcs on a circle, and the goal is to find a minimum number of arcs whose union covers the circle. (i.e., points are covered by arcs). CIRCLE COVER is motivated by a surveillance problem in which one is given a convex area whose outer boundary needs to be watched, and there are a number of sensors each covering a certain range of the boundary faces. An $O(n \log n)$ time algorithm was given in [25], and parallel algorithms where presented in [7, 2].

Wolsey [30] presented a greedy algorithm for weighted SET COVER with hard capacities that achieves a logarithmic approximation ratio. More recently, several approximation algorithms for CAPACITATED VERTEX COVER were presented: the soft capacities case was studied in [19, 4, 15], and the hard capacities case was considered in [10, 14, 18]. Berman et al. [6] gave a 1.5-approximation algorithm for the problem of covering points by capacitated arcs. In their model the points may have different demands, but the capacities are uniform and soft.

Even et al. [11] presented a polynomial time dynamic programming algorithm for CIS both with hard and soft capacities. They extended the algorithm to solve a multi-item lot sizing problem in which there are costs for maintaining inventory, or holding costs, that abide by the following constraint: the requests are indexed by increasing order of importance, such that shortening the holding time of the more important requests at the expense of extending the holding time of the less important requests never increases the overall holding costs. Note that if arbitrary request sizes are allowed this problem becomes NP-hard (there is a reduction from knapsack [13]).

Anily, Tzur and Wolsey [1] proposed a polynomial size linear program that solves a closely related model, in which the requests are not necessarily of unit size, the capacities are uniform, and the holding costs are time-dependent. As opposed to this model, in [11] there are only unit size (or polynomially bounded size) requests, but a more general holding cost structure and non-uniform capacities. Recently Levi, Lodi and Sviridenko [26] have shown that without the relative importance property of the requests, the model from [1] is strongly NP-hard even for unit size requests. The same applies to the model from [11].

Even et al. [11] also considered the CAPACITATED $d$-DIMENSIONAL RECTANGLE STABBING problem which is the $d$-dimensional generalization of CIS. In this problem the intervals are replaced by axis parallel $d$-dimensional rectangles and the points become hyperplanes that are orthogonal to one of the $d$ axes. They presented $O(d)$-approximation algorithms that rely on LP-rounding. (In the hard capacities case they obtained a bi-criteria algorithm that computes solutions that are $O(d)$-approximate and use at most two copies of each hyperplane.) We note that the techniques from [11] can also be used to design $O(d)$-approximation algorithms for the version of CAS in which arcs are replaced by $d$-arcs, where a $d$-arc is a union of $d$ disjoint arcs. (CAPACITATED $d$-DIMENSIONAL RECTANGLE STABBING is a special case of this $d$-dimensional variant of CAS.)

PARTIAL SET COVER was first studied by Kearns [24], who proved that the performance guarantee of the greedy algorithm is at most $2H_m + 3$, where $m$ is the size of the ground set and $H_m$
is the \( m \)th harmonic number. Slavík [28] showed that it is actually bounded by \( H_g \), where \( g \) is the covering requirement. Partial Vertex Cover was studied in [9, 3, 16, 29, 20]. Partial Capacitated Vertex Cover was studied in [27, 5].

**Paper organization.** Section 2 contains a formal definition of the problems, notation and terminology, and a short discussion about the earliest due date rule. Section 3 contains our reduction from CAS to CIS. We consider Partial CAS in Section 4 and Prize Collecting CAS in Section 5.

## 2 Preliminaries

**The problems.** Given a set \( A \) of \( n \) arcs and a set \( P \) of \( m \) points on a circle, a capacitated arc stabbing is formally defined as follows. An assignment is a function \( S : P \rightarrow 2^A \), where \( S(p) \subseteq A \) is a set containing the arcs that are assigned to \( p \). For every point \( p \), we require that \( S(p) \subseteq \mathcal{A}(p) \), where \( \mathcal{A}(p) \) denotes the set of arcs that contain \( p \). An arc \( A \) is said to be covered by \( S \) (or simply covered) if there exists a point \( p \) such that \( A \in S(p) \). Since there is no reason to cover an arc more than once, we henceforth assume, without loss of generality, that \( S(p) \cap S(q) = \emptyset \) for every \( p \neq q \). An assignment \( S \) is called a cover of \( A \) if every arc in \( A \) is covered by \( S \), i.e., if \( \mathcal{A}(S) = A \), where \( \mathcal{A}(S) = \bigcup_p S(p) \). The support of an assignment \( S \) is the set of points \( \sigma(S) = \{ p : S(p) \neq \emptyset \} \).

In the hard capacities case a feasible assignment \( S \) must satisfy \( |S(p)| \leq c(p) \) and the weight of \( S \) is \( \sum_{p \in \sigma(S)} w(p) \). We note that in this case there may not exist a feasible solution. However, one may check whether \( P \) can cover all arcs using a maximum flow algorithm. In the soft capacities case, the multiplicity (or number of copies) of a point \( p \) with respect to an assignment \( S \) is the smallest integer \( \alpha(p) \) for which \( |S(p)| \leq \alpha(p)c(p) \). The weight of a cover \( S \) is \( w(S) = \sum_p \alpha(p)w(p) \). In CAS our goal is to find a minimum weight assignment.

In Partial CAS we are given a covering requirement \( g \leq n \), and the goal is to find a minimum weight assignment \( S \) that covers at least \( g \) arcs, namely such that \( |\mathcal{A}(S)| \geq g \). In Prize Collecting CAS we are not obligated to cover all arcs, but must pay penalties for uncovered arcs. More formally, both points and arcs have non-negative weights, any assignment is feasible, and the goal is to find an assignment \( S \) that minimizes the expression \( w(S) + \sum_{A \in \mathcal{A}(S)} w(A) \).

Observe that CAS with soft capacities can be reduced to CAS with hard capacities. This can be done by making \( \lceil \mathcal{A}(p)/c(p) \rceil \) copies of every point \( p \). A similar reduction applies to Partial CAS and Prize Collecting CAS. Hence in the sequel we focus on hard capacities.

**Notation and terminology.** Throughout the paper we describe circular objects in a clock-wise manner. We use right instead of clock-wise, and left instead of counter clock-wise. Given two point \( p \) and \( q \) on the circle, the arc \( [p, q] \) is the one starting from \( p \) and ending at \( q \) when going from \( p \) to \( q \) in a clock-wise manner. An example is given in Figure 1. Therefore, each arc \( A \in A \) has a left endpoint \( l(A) \) and a right endpoint \( r(A) \). The left (right) endpoint is the endpoint one encounters when going counter clock-wise (clock-wise) from the center of the arc. Given a point \( p \), let \( p-1 \) be the first point to the left of \( p \), and similarly \( p+1 \) denotes the first point to the right of \( p \). We sometimes write \( [p, q] \) or \( (p, q) \) instead of \( [p, q-1] \) or \( [p+1, q] \).

Without loss of generality, we assume that there is no more than one arc endpoint at any point on the circle. Furthermore, we assume that all arc endpoints are contained in \( P \), since we can always add a point with zero capacity to \( P \). Also, \( A \) may contain complete circles. If \( A \in A \)
Figure 1: An arc $[p, q]$.

is a complete circle, then we can add a point $p_A$ with zero capacity and replace $A$ with the arc $[p_A + 1, p_A - 1]$.

The EDD rule. The earliest due date rule (abbreviated EDD) is the following simple rule: process job with earliest due date first. The EDD rule was widely used in the design of scheduling algorithms (see, e.g., [8, Chapter 4]). Let $(P, I)$ be a CIS instance and let $S : P \rightarrow 2^I$ be an assignment. We define a new assignment $S_{\text{EDD}}$ using the EDD rule. We consider the points in $P$ from left to right, and for each such point $p$ we assign (up to) $|S(p)|$ uncovered intervals that contain $p$ with leftmost right endpoints (i.e., with earliest due date). We use the standard method of interchanging pairs to prove that $S_{\text{EDD}}$ is feasible.

Lemma 1. $S_{\text{EDD}}$ is a feasible assignment.

Proof. We prove the lemma by induction on the number of intervals in $I(S)$. In the base case, $S$ covers one interval and we are done. For the induction step let $p_0$ be the left most point in $\sigma(S)$, and let $I_0$ be the interval with leftmost right endpoint that contains $p_0$, i.e., $I_0$ should be assigned to $p_0$ according to the EDD rule. Also, let $p$ be the point that covers $I_0$ in $S$. We define a new assignment $S'$ for $I(S) \setminus \{I_0\}$ from $S$ by choosing an arbitrary interval $I \in S(p_0)$, and replacing $I_0$ with $I$, namely $S'(p_0) = S(p_0) \setminus \{I\}$ and $S'(p) = S(p) \setminus \{I_0\} \cup \{I\}$. $S'$ is feasible, since $p \in [p_0, r(I_0)] \subseteq I$. Now we can use the inductive assumption on $S'$ and $I \setminus \{I_0\}$ to obtain an EDD assignment $S'_{\text{EDD}}$ for $I \setminus \{I_0\}$. Finally, $S_{\text{EDD}}$ is obtained by assigning $I_0$ to $p_0$. \hfill \Box

Observe that $w(S_{\text{EDD}}) = w(S)$, since $|S_{\text{EDD}}(p)| = |S(p)|$, for every $p \in P$. It follows that given an assignment $S$ of intervals to points, one may obtain an EDD assignment for $I(S)$ without changing the load on the points or the weight of the assignment.

3 From Arcs to Intervals

In this section we present a polynomial time reduction from CAS to CIS. Put together with the polynomial time algorithm for CIS from [11] we obtain an algorithm for CAS.

Let $p$ be a point on the circle. We write $p_1 \prec_p p_2$ if one reaches $p_1$ before reaching $p_2$ when going clock-wise starting from $p$, namely if the arc $[p, p_1]$ is shorter than the arc $[p, p_2]$.

Definition 1 (Proper). Let $S$ be an assignment and let $A$ and $B$ be arcs such that $A \in S(p)$ and $B \in S(q)$. The arcs $A$ and $B$ are called proper with respect to $S$ if one of the following conditions is satisfied:

1. $[p, q] \subseteq A \cap B$ and $r(A) \prec_p r(B)$. That is, when going from $p$ in a clock-wise manner the only sequence allowed is $p, q, r(A), r(B)$. (See examples in Figures 2a–2c.)
Figure 2: Depiction of proper arcs $A$ and $B$ with respect to an assignment $S$, where $A \in S(p)$ and $B \in S(q)$. In the first two figures the intersections consist of one arc, while in the other figures the intersections consist of two arcs.

2. $p, q \in A \cap B$, $[p, q] \not\subseteq A \cap B$ and $r(A) \prec_p r(B)$. That is, when going from $p$ in a clock-wise manner the only sequence allowed is $p, r(A), q, r(B)$. (See example in Figure 2d.)

If $A$ and $B$ are proper with respect to $S$, then $B$ and $A$ are also called proper with respect to $S$.

Notice that two arcs $A$ and $B$ are proper with respect to $S$ if both arcs are covered by the same point, i.e. if $A, B \in S(p)$ for some $p \in P$. Also, $A$ and $B$ are proper with respect to $S$ if either $p \not\in B$ or $q \not\in A$. (For example, if $A \cap B = \emptyset$.)

In the next definition we focus on a specific arc $A$ and require that $A$ and any other arc $B$ are proper with respect to some assignment $S$.

**Definition 2** $(A\text{-edd})$. Let $A \in A$ be an arc. An assignment $S$ is called $A$-edd if $A \in A(S)$ and $A$ and $B$ are proper with respect to $S$, for every $B \in A(S)$.

We show that there exists an $A$-edd optimal assignment for some arc $A$.

**Lemma 2.** Let $A \in A$ be any arc such that $A \not\subseteq B$, for every $B \in A \setminus \{A\}$. Then, there exists an optimal assignment $S$ that is $A$-edd.

**Proof.** Let $S_0$ be an optimal assignment. We construct a new $A$-edd assignment $S$ by reassigning the arcs in $A$ such that $|S(q)| = |S_0(q)|$, for every $q \in P$.

First, observe that if $B \in S_0(q)$ and $q \not\in A$, then $A$ and $B$ are proper with respect to $S_0$. Hence, there is no need to reassign $B$. For this reason we define $S(q) = S_0(q)$ for every $q \not\in A$. It remains to deal with points within $A$. In what follows we describe a reassignment process of arcs that are assigned by $S_0$ to points in $A$. We show that either we obtain an $A$-edd assignment, or we increase the size of a certain set of arcs. Since there is a finite number of arcs, we eventually reach an $A$-edd assignment.

Let $B \subseteq A$ denote the set of arcs that are covered by $S_0$ using points in $A$, namely

$$B = \{ B \; : \; B \in S_0(q) \text{ and } q \in A \} .$$

Clearly, $A \cap B \neq \emptyset$ for every $B \in B$. Moreover, observe that $A \in B$. Our goal is to reassign the arcs in $B$ such that $A$ and $B$ are proper for every $B \in B \setminus \{A\}$.

Since $A$ is covered by $S_0$, but is not contained by other arcs covered by $S_0$, there are four types of intersections between $A$ and an arc $B \in B$:
Figure 3: Depiction of the possible intersection types between $A$ and $B \in B$. $A$ is represented by a solid line, while $B$ is represented by a dashed line.

Figure 4: Figures (a) and (b) feature arcs from $T_\ell$ and $T_r$. $A$ is represented by a solid line, while $B$ is represented by a dashed line. The thick part of $B$ represents $B'$. An example of the problematic scenario is given in Figure (c).

We partition $B \setminus \{A\}$ into four sets according to the intersection types with $A$:

1. $L$ contains arcs having a left intersection with $A$.
2. $R$ contains arcs having right intersection with $A$.
3. $C$ denotes the set of arcs that are contained in $A$.
4. $T$ is the set of arcs that have a two sided intersection with $A$.

We further partition $T$ into two sets $T_\ell$ and $T_r$ according to the location of the point that covers the arc $B$. Formally,

$$T_\ell = \{ B \in T : B \in S_0(q) \text{ and } q \in [\ell(A), r(B)] \},$$

$$T_r = \{ B \in T : B \in S_0(q) \text{ and } q \in [\ell(B), r(A)] \}.$$

Examples are given in Figures 4a and 4b.

In what follows we try to obtain an $A$-EDD assignment. If we fail, we will obtain a new optimal assignment that will augment $T_\ell$ on the expense of $T_r$. 

7
We define a new arc \( B' \subseteq A \cap B \), for every arc \( B \in B \), as follows:

\[
B' = \begin{cases} 
B \cap A & B \in B \setminus T, \\
[\ell(A), r(B)] & B \in T_L, \\
[\ell(B), r(A)] & B \in T_R.
\end{cases}
\]

If \( B \in T \), then the intersection \( A \cap B \) contains two arcs, and \( B' \) is the arc in which \( B \) is covered by \( S_0 \) (see Figure 4). Let \( B' = \{ B' : B \in B \} \). Notice that a cover of \( B' \) induces a cover of \( B \), since \( B' \subseteq B \), for every \( B \in B \). Furthermore, \( B' \) can be covered by \( S_0' \), where \( S_0' \) is obtained from \( S_0 \) by replacing \( B \) with \( B' \), for every \( B \in B \).

We obtain a new cover \( S_1' \) of \( B' \) using the EDD rule. More specifically, we consider the points in \( P \cap A \) from left to right starting with \( \ell(A) \). For each such point \( q \) we assign \( |S_0'(q)| = |S_0(q)| \) uncovered arcs with leftmost right endpoints (i.e., earliest due date). The arc with the leftmost right endpoint is chosen arbitrarily in case of a tie. Notice that \( A \) is chosen after any arc \( B' \), such that \( B \in L \cup T_L \) and before any arc \( B' \) such that \( B \in R \cup T_R \), \( S_1' \) is feasible due to Lemma 1.

We complete the assignment \( S \) by the assignment of \( B \) to the points in \( A \) that is induced by the assignment \( S_1' \). (Recall that \( S(q) = S_0(q) \) for every \( q \notin A \).) Let \( p \) be the point that covers \( A \). Due to the EDD assignment of \( B' \) we have that

- If \( B \in L \cup T_L \) and \( B \in S(q) \), then \( q \prec_{\ell(A)} p \).
- If \( B \in R \cup T_R \) and \( B \in S(q) \), then \( p \prec_{\ell(A)} q \).
- If \( B \in C \) and \( B \in S(q) \), then either \( p \in [\ell(A), \ell(B)] \) or \( p \in [q, r(A)] \).

Hence, if \( B \notin T_R \), then \( A \) and \( B \) are proper with respect to \( S \). If \( p \in (r(B), r(A)] \), for every \( B \in T_R \), then \( A \) and \( B \) are proper with respect to \( S \), for every \( B \in T_R \). In this case, \( S \) and \( A \) satisfy the requirements of the lemma. However, there may exist an arc \( B \in T_R \) such that \( p \in [\ell(A), r(B)] \), which means that \( A \) and \( B \) are not proper with respect to \( S \) (see Figure 4c). In this case we define a new assignment \( S_1 \) by taking \( S \) and switching the covering points of \( A \) and \( B \).

We repeat the construction above using \( S_1 \) instead of \( S_0 \). Notice that the new set \( T_L \) (with respect to \( S_1 \)) contains all arcs from the old \( T_L \) (with respect to \( S_0 \)), since every arc \( C \) in the old \( T_L \) is covered by a point within \( C' = [\ell(A), r(C)] \) by \( S_1 \). Furthermore, the new set contains at least one new arc, i.e. \( B \). Hence, an A-EDD assignment is obtained after repeating this process at most \( n \) times. \( \square \)

Next, using Lemma 2 we show CAS can be reduced to CIS.

**Theorem 1.** CAS can be solved in polynomial time.

**Proof.** Let \( A \in A \) be any arc such that \( A \not\subset B \), for every \( B \in A \setminus \{ A \} \), and let \( S \) be an A-EDD optimal assignment. We know that there exists such an assignment due to Lemma 2. Now, suppose that it is known that \( A \in S(p) \), for some point \( p \). In this case, we break the circle at \( p \). For this we need to consider all arcs that contain \( p \). Let \( B \) be an arc such that \( p \in B \). Since \( S \) is A-EDD, we have that if \( B \in S(q) \), then

- If \( A \) and \( B \) are left intersecting or \( B \subset A \), then \( q \notin (p, r(B)) \).
- If \( A \) and \( B \) are right intersecting, then \( q \notin (\ell(B), p) \).
- If \( A \) and \( B \) have a two sided intersection and \( p \in [\ell(A), r(B)] \), then \( q \notin (p, r(B)) \).
- If \( A \) and \( B \) have a two sided intersection and \( p \in [\ell(B), r(A)] \), then \( q \notin (\ell(B), p) \).

8
In all four cases the arc $B$ cannot be covered either in $[\ell(B), p)$ or in $(p, r(B)]$. It follows that all arcs that go through $p$ can be cut off at $p$, namely either $[\ell(B), p)$ or $(p, r(B)]$ can be removed. After cutting the arcs, we break $p$ into two points $p^+$ and $p^-$. If $r(B) = p$, then it is replaced with $r(B) = p^-$, and if $\ell(B) = p$, then it is replaced with $\ell(B) = p^+$. The weight of both $p^-$ and $p^+$ is zero. Since we do not know the identity of the arcs that are covered by $p$, we guess how many arcs that end at $p^-$ were originally covered by $S$ using $p$. Since $A \in S(p)$ there are $c(p)$ such possibilities, one of which is satisfied by $S$. For a guess $c(p^-) = k$ we define $c(p^+) = c(p) - k - 1$ and obtain a CIS instance.

All that remains is to go over CIS instances that are induced by all possible choices of a point $p \in A$, and a number $k$, and to compute an optimal solution for each such instance using the polynomial time algorithm for CIS from [11]. The best solution induces an optimal assignment for the CAS instance. Since there are $n$ arcs, $m$ points and $n$ choices for $k$, we obtain an optimal assignment in polynomial time.

4 Partial Capacitated Arc Stabbing

In this section we present a polynomial time algorithm that solves Partial CAS. Recall that in Partial CAS we are given a covering requirement $g \leq n$, and our goal is to find a minimum weight set of points that stabs at least $g$ arcs. The algorithm consists of two steps: the first is a reduction from Partial CAS to Partial CIS and the second is an algorithm for Partial CIS. The latter is based on a dynamic programming algorithm for CIS from [11].

4.1 Reduction

Our reduction from Partial CAS to Partial CIS is a variant of the reduction from Section 3. First, observe that in the partial setting a feasible assignment may not cover an arc. Hence, we cannot simply pick any arc $A$ that is not contained in other arcs. However, given an optimal assignment $S$, there exists some arc $A \in A(S)$ such that $A \not\subseteq B$ for every $B \in A(S)$. Hence we obtain the following variant of Lemma 2.

**Lemma 3.** There exists an optimal assignment $S$ that is $A$-edd for some arc $A \in A(S)$. Moreover $A$ is not properly contained in any arc in $A(S)$.

Proving Lemma 3 can be done similarly to proving Lemma 2. The main difference is that we ignore the arcs in $A \setminus A(S)$.

Lemma 3 implies the following result.

**Theorem 2.** Partial CAS can be solved in polynomial time, given a polynomial time algorithm for Partial CIS.

**Proof.** First, suppose that it is known that there exists an $A$-edd optimal assignment $S$, for some arc $A \in A$ such that $A \in S(p)$. In this case, we may break the circle at $p$ as was done in the proof of Theorem 1.

All that remains is to go over instances that are induced by all possible choices of an arc $A \in A$, a point $p \in A$, and a number $k$, and to compute an optimal solution for each such instance (after removing all arcs that contain $A$). The best solution induces an optimal assignment for the Partial CAS instance. Since there are $n$ arcs, $m$ points and $n$ choices for $k$, we obtain an optimal assignment in polynomial time, if Partial CIS can be solved in polynomial time. □
4.2 Dynamic Programming Algorithm

In this section we provide a dynamic programming algorithm for PARTIAL CIS. Recall that in PARTIAL CIS we are given a set \( I \) of \( n \) intervals and a set \( P \) of \( m \) points on the real line. Each point \( p \in P \) has a capacity \( c(p) \) and a weight \( w(p) \). We are also given a covering requirement \( g \leq n \).

Let \( I_1, I_2, \ldots, I_n \) be an ordering of the intervals by their left endpoint, i.e. \( \ell(I_i) < \ell(I_{i+1}) \) for every \( i \). Without loss of generality, we look for an optimal solution \( S \) that has the following leftmost interval first property: For any point \( p \in \sigma(S) \), if \( p \) covers \( I \), then all points \( p' \in P \) with \( \ell(I) \leq p' \leq p \) that are also in the solution are used to their full capacity and for all intervals \( I' \in I \) covered by these points either \( \ell(I') < \ell(I) \) or \( r(I') < p \). That is, \( I \) cannot be covered by any point that is to the left of \( p \), and cannot be swapped with any interval \( I' \) with \( \ell(I') > \ell(I) \) that is covered by such a point. Observe that if time is reversed, the leftmost interval first rule is similar to, but not the same as, the EDD rule. (The rules may imply different assignments when \(|I(S)| < \sum_{p \in \sigma(S)} c(p)\).)

In [11] it is shown that any assignment of intervals to points can be transformed into an assignment that has the leftmost interval first property by reassignments. Given an assignment \( S \) that does not satisfy this property, one can perform a series of corrections until the property holds. Namely, as long as the assignment does not satisfy the property, each interval \( I \) is moved to the leftmost vacant point that can cover it, and any two intervals \( I \) and \( I' \) covered by points \( p \) and \( p' \) respectively, are swapped if \( \ell(I) < \ell(I') \leq p' < p \leq r(I') \).

Satisfying the leftmost interval first property implies the following:

**Observation 1 ([11]).** Let \( S \) be an assignment that satisfies the leftmost interval first property. For any range \([p_1, p_2]\), let \( I \) be the interval with the minimum \( \ell(I) \) among the intervals covered by points in \([p_1, p_2]\). If \( I \) is covered by \( p \), then the right endpoint of all intervals covered by points in \([p_1, p]\) are to the left of \( p \).

To define the dynamic programming table we need the following notation. For the interval \( I_i \) and two points \( p_1 \leq p_2 \), such that \( r(I_i) \in [p_1, p_2] \), let

\[
\mathcal{I}(i, p_1, p_2) = \{ I_j \mid j \geq i \text{ and } r(I_j) \in [p_1, p_2]\},
\]

i.e. \( \mathcal{I}(i, p_1, p_2) \) is the set of all intervals whose left endpoint is at or to the right of \( \ell(I_i) \) and whose right endpoint falls within the range \([p_1, p_2]\).

The dynamic programming table \( \Pi \) is constructed as follows. The entry \( \Pi(h, i, p_1, p_2, k) \), where \( h \) is a covering requirement, \( i \in \{1, \ldots, n\} \) is an interval index, \( p_1 \) and \( p_2 \) are points, and \( r(I_i) \in [p_1, p_2] \), contains the minimum weight of a cover of at least \( h \) intervals in \( \mathcal{I}(i, p_1, p_2) \) by points in the range \([p_1, p_2]\) with the additional constraint that the point \( p_1 \) covers no more than \( k \leq c(p_1) \) intervals, and if \( k < c(p_1) \) then the weight of \( p_1 \) is assumed to be zero. Observe that the size of the table is \( O(n^3 m^2) \). Also, the optimum is given by \( \Pi(g, 1, p_L, p_R, c(p_L)) \), where \( p_L \) and \( p_R \) are the leftmost and rightmost points.

In the base case \( \Pi(h, i, p_1, p_1, k) \) only a single point \( p_1 \) is considered. Specifically, if \(|\mathcal{I}(i, p_1, p_1)| < h \), then \( \Pi(h, i, p_1, p_1, k) = \infty \), because \( \mathcal{I}(i, p_1, p_1) \) contains too few intervals. Also, if \( h > k \) then \( \Pi(h, i, x_1, x_1, k) = \infty \), since in this case \( p_1 \)'s capacity is not enough to cover \( h \) intervals. Otherwise,

\[
\Pi(h, i, p_1, p_1, k) = \begin{cases} 
  w(p_1) & k = c(p_1), \\
  0 & k < c(p_1).
\end{cases}
\]
Also, $\Pi(h, i, p_1, p_2, 0) = \Pi(h, i, p_1 + 1, p_2, c(p_1 + 1))$ if $I(i, p_1 + 1, p_2) \geq h$, and $\infty$ otherwise, since then the subproblem is infeasible.

Below, we show how to compute an entry $\Pi(h, i, p_1, p_2, k)$, for $p_1 < p_2$ and $k > 0$ in polynomial time given all entries $\Pi(h', i', p_1', p_2', k')$ with $i' > i$ and $h' \leq h$. Since the size of the table is polynomial this implies a polynomial time algorithm. The computation is based on Observation 1. To compute $\Pi(h, i, p_1, p_2, k)$ we enumerate over all possible points that can cover the interval $I_i$.

We also take into account the possibility that $I_i$ is not covered. If $p$ covers $I_i$, then it partitions the problem into two subproblems: a left instance that contains all intervals in $I(i, p_1, p_2)$ whose right endpoint is to the left of $p$ which must be covered by points to the left of $p$, and a right instance that contains the rest of the intervals (excluding $I_i$). By Observation 1 these intervals are covered by either $p$ or points to its right. This implies the following recursive equation:

$$
\Pi'(h, i, p_1, p_2, k) = \min_{p \in [p_1, \min\{r(I_i), p_2\}]} \left\{ \begin{array}{ll}
\Pi(h', i', p_1, p - 1, k) + \\
\Pi(h - h' - 1, i'', p_2, k'') + w'(p)
\end{array} \right\}
$$

where:

- $i' = \min\{j : j > i \text{ and } r(I_j) \in [p_1, p]\}$
- $i'' = \min\{j : j > i \text{ and } r(I_j) \in [p, p_2]\}$
- $\bar{i} = \min\{j : j > i \text{ and } r(I_j) \in [p_1, p_2]\}$
- $k'' = \begin{cases}
c(p) - 1 & \text{if } p > p_1, \\
1 & \text{if } p = p_1
\end{cases}$
- $w'(p) = \begin{cases}
0 & \text{if } p = p_1 \text{ and } k < c(p_1) \\
w(p) & \text{otherwise}.
\end{cases}$

The idea in Eq. (1) is that if $I_i$ is not covered, then it is ignored. Otherwise, if $I_i$ is covered by $p$ in the interval $[p_1, p_2]$, then the subproblem $I(i, p_1, p_2)$ is partitioned into a left instance and a right instance. The left instance is $I(i', p_1, p - 1)$, where $i'$ is the interval with left-most left endpoint among the intervals whose right endpoint is before $p$. We assume that $\Pi(h', i', p_1, p - 1, k)$ is the optimum of the left instance for all possible values of $h' \in \{0, \ldots, h - 1\}$. The right instance is $I(i'', p_2)$, where $i''$ is the interval with left-most left endpoint among the intervals whose right endpoint is not to the left of $p$, and $k''$ is the residual capacity of $p$. We assume that $\Pi(h - h' - 1, i'', p_2, k'')$ is the optimum of the right instance for all possible values of $h'$. The cost of covering $I_i$ by $p$ is denoted by $w'(p)$.

As for the running time, computing an entry takes $O(nm)$ time, and thus the overall time complexity is $O(n^4m^3)$. Also note that the computation of $\Pi(h, i, p_1, p_2, k)$ can be modified to compute a corresponding optimal solution.

5 Prize Collecting Capacitated Arc Stabbing

In this section show that our techniques can be used to design a polynomial time algorithm for solving Prize Collecting CAS. That is, we show a reduction from Prize Collecting CAS to Prize Collecting CIS, and an algorithm for Prize Collecting CIS.
First, we claim that a variant of the dynamic programming algorithm from Section 4 can be used to solve Prize Collecting CIS. The dynamic programming table \( \Pi \) is defined as follows. The entry \( \Pi(i, p_1, p_2, k) \), where \( i \in \{1, \ldots, n\} \) is an interval index, \( p_1 \) and \( p_2 \) are points, and \( r(I_i) \in [p_1, p_2] \), contains the minimum weight of a cover of \( I(i, p_1, p_2) \) by points in the range \([p_1, p_2] \) with the additional constraint that the point \( p_1 \) covers no more than \( k \leq c(p_1) \) intervals, and if \( k < c(p_1) \) then the weight of \( p_1 \) is assumed to be zero. The base case \( \Pi(i, p_1, p_1, k) \) is the optimal cover of \( I(i, p_1, p_1) \) using \( p_1 \). There are two possibilities:

1. Covering the \( k \) most expensive arcs by \( p_1 \) and paying the penalties for the rest. We also pay for \( p_1 \) if \( k = c(p_1) \).
2. Paying the penalties for all arcs in \( I(i, p_1, p_1) \). This option is relevant only when \( k = c(p_1) \).

Furthermore, Eq. (1) should be replaced with:

\[
\Pi'(i, p_1, p_2, k) = \min_{p \in [p_1, \min \{r(I_i), p_2\}]} \left\{ \Pi'(i', p_1, p - 1, k) + \Pi(i'', p, p_2, k'') + w'(p) \right\}
\]

\[
\Pi(i, p_1, p_2, k) = \min \left\{ \Pi'(i, p_1, p_2, k), \Pi(\bar{i}, p_1, p_2, k) + w(I_i) \right\}
\]

using the notation from Section 4. The computation of an entry takes \( O(n + m) \) time, and therefore the overall running time is \( O(n^2m^2(n + m)) \).

Next, recall that in Prize Collecting CAS any assignment is feasible, hence the assignment \( S_{\emptyset}(p) = \emptyset \), for every \( p \), is also feasible. If \( S_{\emptyset} \) is not optimal, then a variation of the reduction from Section 3 can be applied, since there exists an arc that is covered by some point, and this implies that Lemma 3 holds. Hence, we have two candidate solutions, \( S_{\emptyset} \) and the solution computed by applying the reduction and the algorithm for Prize Collecting CIS. The minimum weight assignment is an optimal solution.

References


