Approximation Algorithms for Partial Capacitated Covering Problems

Guy Flysher
Approximation Algorithms for Partial Capacitated Covering Problems

Research Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Guy Flysher

Submitted to the Senate of the Technion - Israel Institute of Technology

Adar 5767 Haifa March 2007
The Research Thesis was done Under the Supervision of Assoc. Prof. Reuven Bar-Yehuda in the Faculty of Computer Science.

I wish to express my thanks Assoc. Prof. Reuven Bar-Yehuda for his guidance, support, help, and for urging me on when I needed it.

I wish to deeply thank Dr. Dror Rawitz for his help and valuable advice.

The generous financial help of the Technion is gratefully acknowledged.

This research is dedicated to my beloved dog Didi who passed away shortly prior to its completion.
Contents

Abstract 1

Abbreviations and Notations 2

1 Introduction and Overview 3
   1.1 The problems 3
      1.1.1 Capacitated Vertex Cover 3
      1.1.2 Partial Vertex Cover 4
      1.1.3 Partial Capacitated Vertex Cover 4
      1.1.4 Prize Collecting Vertex Cover 4
   1.2 Motivation 5
   1.3 Related Work and Previous Results 5
   1.4 Our Results 6

2 Definitions and Preliminaries 8
   2.1 Notation and terminology 8
   2.2 Small, medium, and large edges 8
   2.3 Local ratio 9

3 Partial Capacitated Covering with Separable Demands 10
   3.1 Algorithm PCVC 10
   3.2 Feasibility of the solution 11
   3.3 Approximation Ratio 12
      3.3.1 Analysis of Algorithm PCVC 13
         3.3.1.1 The recursion ends with the empty assignment 13
         3.3.1.2 The recursion ends with a medium or large edge 17
         3.3.1.3 Proof of Theorem 2 19
   3.4 Algorithm PCVC without Line 2 20

4 Partial Capacitated Covering with Inseparable Demands 21

5 Partial Capacitated Vertex Cover with Unit Demands 23
   5.1 Algorithm PCVC2 23
   5.2 Feasibility of the solution 23
5.3 Approximation Ratio ............................................ 24
  5.3.1 Analysis of Algorithm PCVC2 ............................. 25

6 Partial Capacitated Vertex Cover in Hyper-Graphs ............ 28
  6.1 Notation and Terminology .................................... 28
  6.2 Non-Unit Demands ............................................ 28
  6.3 Unit Demands .................................................. 30

7 Prize Collecting Capacitated Vertex Cover ..................... 31
  7.1 Simple Reduction to Capacitated Vertex Cover ................ 31
  7.2 Capacitated Vertex Cover with Separable Demands ............. 31
  7.3 A More Careful Analysis ..................................... 32

8 Conclusions and Open Problems .................................. 34

Bibliography .......................................................... 36
List of Figures

1.1 An example showing the difference between separable demands and inseparable demands ................................................................. 4
3.1 Possible transitions between the conditions of lemma 2 ...................... 14
3.2 Possible transitions between the conditions of lemma 4 ...................... 18
3.3 An example showing how algorithm PCVC fails to achieve a 3-approximate solution if line 2 is omitted .............................................. 20
Abstract

We study the partial capacitated vertex cover problem (PCVC) in which the input consists of a graph $G$ and a covering requirement $L$. Each edge $e$ in $G$ is associated with a demand (or load) $\ell(e)$, and each vertex $v$ is associated with a (soft) capacity $c(v)$ and a weight $w(v)$. A feasible solution is an assignment of edges to vertices such that the total demand of assigned edges is at least $L$. The weight of a solution is $\sum_v \alpha(v)w(v)$, where $\alpha(v)$ is the number of copies of $v$ required to cover the demand of the edges that are assigned to $v$. The goal is to find a solution of minimum weight. We consider three variants of PCVC. In PCVC with separable demands the only requirement is that total demand of edges assigned to $v$ is at most $\alpha(v)c(v)$. In PCVC with inseparable demands there is an additional requirement that if an edge is assigned to $v$ then it must be assigned to one of its copies. The third variant is the unit demands version. We present 3-approximation algorithms for both PCVC with separable demands and PCVC with inseparable demands. We also present a 2-approximation algorithm for PCVC with unit demands. We show that similar results can be obtained for the prize collecting version of capacitated vertex cover. Our algorithms are based on a unified approach for designing and analyzing approximation algorithms for capacitated covering problems. This approach yields simple algorithms whose analyses rely on the local ratio technique and sophisticated charging schemes.
Abbreviations and Notations

$E(v)$  The group of edges incident on $v$.

$N(v)$  The set of vertices that share an edge with $v$.
        Formally, $N(v) = \{u : \exists e \in E, u, v \in e\}$

$\ell$  A load function associating each edge with a natural number.
        Formally, $\ell : E \rightarrow \mathbb{N}$.

$\ell(E')$  The total load of all the edges in $E'$. Formally, $\ell(E') = \sum_{e \in E'} \ell(e)$.

$L$  The covering requirement. This value is the minimum total load that a feasible solution must cover.

$c$  A capacity function associating each vertex with a natural number.
        Formally, $c : V \rightarrow \mathbb{N}$.

$\deg(v)$  The "weighted" degree of vertex $v$. Formally, $\deg(v) = \sum_{e \in E(v)} \ell(e)$

$A$  An assignment of edges to vertices. Formally, $A : V \rightarrow 2^E$.

$V(A)$  The support of assignment $A$. Formally, $V(A) = \{u : A(u) \neq \emptyset\}$

$\check{c}(v)$  The minimum between a vertex’s capacity and its degree.
        Formally, $\check{c}(v) = \min \{c(v), \deg(v)\}$.

$b(v)$  The minimum between a vertex’s capacity, it’s degree and the covering requirement. Formally, $b(v) = \min \{c(v), \deg(v), L\} = \min \{\check{c}(v), L\}$

$\Delta$  The maximum degree of a hyper-edge in a given hyper-graph.

$\bar{\ell}(u, v)$  The total load of the edges incident on both $u$ and $v$.
        Formally, $\bar{\ell}(u, v) = \sum_{e : u, v \in e} \ell(e)$
Chapter 1
Introduction and Overview

1.1 The problems

Given a graph $G = (V, E)$, a vertex cover is a subset $U \subseteq V$ such that each edge in $G$ has at least one endpoint in $U$. In the vertex cover problem, we are given a graph $G$ and a weight function $w$ on the vertices, and our goal is to find a minimum weight vertex cover. Vertex cover is NP-hard [22], and cannot be approximated within a factor of $10\sqrt{5} - 21 \approx 1.36$, unless P=NP [12]. On the positive side, there are several 2-approximation algorithms for vertex cover (see chapter 3 of [20], [6] and references therein).

1.1.1 Capacitated Vertex Cover

The capacitated vertex cover problem (cvc, for short) is an extension of vertex cover in which each vertex $u \in V$ has a capacity $c(u) \in \mathbb{N}$ that determines the number of edges it may cover. That is, $u$ may cover up to $c(u)$ incident edges. Multiple copies of $u$ may be used to cover additional edges, provided that the weight of $u$ is counted for each copy (soft capacities). A feasible solution is an assignment of every edge to one of its endpoints.

A capacitated vertex cover is formally defined as follows. An assignment is a function $A : V \rightarrow 2^E$. That is, for every vertex $u$, $A(u) \subseteq E(u)$, where $E(u)$ denotes the set of edges incident on $u$. An edge $e$ is said to be covered by $A$ (or simply covered) if there exists a vertex $u$ such that $e \in A(u)$. Henceforth, we assume, without loss of generality, that an edge is covered by no more than one vertex. An assignment $A$ is a cover if every edge is covered by $A$, i.e., if $\bigcup_{u \in V} A(u) = E$. The multiplicity (or number of copies) of a vertex $u$ with respect to an assignment $A$ is the smallest integer $\alpha(u)$ for which $|A(u)| \leq \alpha(u)c(u)$. The weight of a cover $A$ is $w(A) = \sum_u \alpha(u)w(u)$. Note that the presence of zero-capacity vertices may render the problem infeasible, but detecting this is easy: the problem is infeasible if and only if there is an edge whose two endpoints have zero capacity. Also note that vertex cover is the special case where $c(u) = \deg(u)$ for every vertex $u$.

In a more general version of cvc we are given a demand or load function $\ell : E \rightarrow \mathbb{N}$. In the separable edge demands case $\alpha(u)$ is the number of copies of $u$ required to cover the total demand of the edges in $A(u)$. That is, $\alpha(u)$ is the smallest integer for which
Figure 1.1: A cvc instance: $c(v_0) = 3$ and $\ell(v_0, v_i) = 2$ for $i \in \{1, 2, 3\}$. In the case of separable demands two copies of $v_0$ are sufficient to cover the three edges, but in the case of inseparable demands three copies of $v_0$ are needed to cover the edges.

$\sum_{e \in A(u)} \ell(e) \leq \alpha(u)c(u)$. In the case of inseparable edge demands there is an additional requirement that if an edge is assigned to $u$ then it must be assigned to one of its copies. (See example in Figure 1.1.) It follows that if the demand of an edge is larger than the capacity of both its endpoints, it cannot be covered. Clearly, given an assignment $A$, this additional requirement may only increase $\alpha(u)$. (Each vertex faces its own bin packing problem.) Hence, if all edges are coverable in the inseparable demands sense, then the optimum value for cvc with separable demands is not larger than the optimum value for cvc with inseparable demands.

### 1.1.2 Partial Vertex Cover

Another extension of vertex cover is the partial vertex cover problem (pvc). In pvc the input consists of a graph $G$ and an integer $L$ and the objective is to find a minimum weight subset $U \subseteq V$ that covers at least $L$ edges. pvc extends vertex cover, since in vertex cover $L = |E|$. In a more general version of pvc we are given edge demands, and the goal is to find a minimum weight subset $U$ that covers edges whose combined demand is at least $L$.

### 1.1.3 Partial Capacitated Vertex Cover

In this research we study the partial capacitated vertex cover problem (PCVC) that extends both pvc and cvc. In this problem we are asked to find a minimum weight capacitated vertex cover that covers edges whose total demand is at least $L$. We consider three variants of PCVC: PCVC with separable demands, PCVC with inseparable demands, and PCVC with unit demands.

### 1.1.4 Prize Collecting Vertex Cover

A third extension of vertex cover is the prize collecting vertex cover (also called generalized vertex cover). In this problem we are not obligated to cover any edges, but must pay a penalty for uncovered edges. More specifically, both vertices and edges have non-negative weights; every set of vertices is a feasible solution; and the weight of a feasible solution is the total weight of its vertices plus the total weight of uncovered edges. In this paper we consider the prize collecting version of capacitated vertex cover (PC-CVC). In PC-CVC the goal is to find an assignment $A$ that minimizes the expression $w(A) + \sum_{e \notin \bigcup_u A(u)} w(e)$. 

---

Technion - Computer Science Department - M.Sc. Thesis MSC-2008-02 - 2008
1.2 Motivation

CVC was proposed by Guha et al. [16], who were motivated by a problem from the area of bioinformatics. They described a chip-based technology, called GMID (Glycomolecule ID), that is used to discover the connectivity of building blocks of glycoproteins. Given a set of building blocks (vertices) and their possible connections (edges), the goal is to identify which connections exist between the building blocks in order to discover the variant of the glycoprotein in question. Given a building block \( B \) and a set \( S \) that consists of \( k \) of \( B \)'s neighbors, GMID can reveal which of the building blocks from \( S \) are connected to \( B \). The problem of minimizing the number of GMID operations needed to cover the required information graph is exactly CVC with uniform capacities. Since not all possible combinations of ties between building blocks may appear, it is sometimes sufficient to probe only a fraction of the edges in the information graph. In this case the problem of minimizing the number of GMID operations is pCVC with uniform capacities. For details about the structure of glycoproteins we refer the reader to [24].

CVC can be seen as a capacitated facility location problem, where the edges are the clients and the vertices are the capacitated facilities. For instance, CVC in hyper-graphs can be used to model a cellular network planning problem, in which we are given a set of potential base stations that are supposed to serve clients that are located in receiving areas. Each such area corresponds to a specific subset of the potential base stations. We are supposed to assign areas (edges) to potential base stations (vertices), and to locate transmitters with limited bandwidth capacity in each potential base station that will serve the demand of receiving areas that are assigned to it. Our goal is to assign areas to base stations so as to minimize transmitter costs. In terms of the above cellular network planning problem, pCVC is the case where we are required to cover only a given percentage of the demands. For details about cellular planning we refer the reader to [1] and references therein.

1.3 Related Work and Previous Results

Capacitated covering problems date back to Wolsey [28] (see also [3, 11]). Wolsey presented a greedy algorithm for weighted set cover with hard capacities that achieves a logarithmic approximation ratio. Guha et al. [16] presented a primal-dual 2-approximation algorithm for CVC. (A local ratio version of this algorithm was given in [6].) They also gave a 3-approximation algorithm for CVC with separable edge demands. Both algorithms can be extended to hyper-graphs in which the maximum size of an edge is \( \Delta \). The resulting approximation ratios are \( \Delta \) and \( \Delta + 1 \), respectively. Guha et al. [16] also studied CVC in trees. They obtained polynomial time algorithms for the unit demands case and for the unweighted case, and proved that CVC with separable demands in trees is weakly NP-hard.

Gandhi et al. [14] presented a 2-approximation algorithm for CVC using an LP-rounding method called dependent rounding. Chuzhoy and Naor [11] studied the version of CVC in which one may use only a bounded number of copies of every vertex. In this case the capacities are referred to as hard. Chuzhoy and Naor presented a 3-approximation algorithm for unweighted CVC with hard capacities, and proved that the weighted version
of this problem is as hard to approximate as set cover. Gandhi et al. [13] improved the approximation ratio for unweighted CVC with hard capacities to 2.

The partial set cover problem was first studied by Kearns (Chapter 5 of [23]), who proved that the performance ratio of the greedy algorithm is at most $2H_m + 3$, where $H_m$ is the $m$th harmonic number. Slavík [26] showed that it is actually bounded by $H_m$. Bshouty and Burroughs [10] obtained the first polynomial time 2-approximation algorithm for PVC. Bar-Yehuda [4] studied PVC with demands and presented a local ratio 2-approximation algorithm for this problem. His algorithm extends to a $\Delta$-approximation algorithm in hyper-graphs in which the maximum size of an edge is $\Delta$. Gandhi et al. [15] presented a different algorithm with the same approximation ratio for PVC with unit demands in hyper-graphs that is based on a primal-dual analysis. Building on [15], Srinivasan [27] obtained a polynomial time $(2 - \Theta(\frac{1}{d}))$-approximation algorithm for PVC, where $d$ is the maximum degree of a vertex in $G$. Halperin and Srinivasan [18] developed a $(2 - \Theta(\frac{\ln \ln d}{\ln d}))$-approximation algorithm for PVC.

Recently, Mestre [25] studied PCVC with unit demands. He presented a primal-dual 2-approximation algorithm that is based on the 2-approximation algorithm for CVC by Guha et al. [16] and on the 2-approximation algorithm for PVC by Gandhi et al. [15].

We note that the parameterized complexity of PVC and CVC was investigated by Guo et al. [17], who showed that CVC is fixed-parameter tractable, while PVC is $W[1]$-hard. It follows that PCVC is also $W[1]$-hard.

The prize collecting version of vertex cover was studied by Hochbaum [21], who presented a 2-approximation algorithm based on LP-rounding and maximum flow. Hassin and Levin [19] presented a 2-approximation algorithm for a more general version where the costs differ if an edge is cover by one, both or neither of its end points. Bar-Yehuda and Rawitz [9] presented a linear time local ratio $\Delta$-approximation algorithm for prize collecting vertex cover in hypergraphs. This algorithm extends the approximation algorithm for vertex cover in hypergraphs [7].

### 1.4 Our Results

We present constant factor approximation algorithms to several variants of PCVC. Our algorithms are based on a unified approach for designing and analyzing approximation algorithms for capacitated covering problems. This approach yields simple algorithms whose analyses rely on the local ratio technique and sophisticated charging schemes. Our method is inspired by the local ratio interpretations of the approximation algorithms for CVC from [16] and the local ratio 2-approximation algorithm for PVC from [4].

We present a 3-approximation algorithm for PCVC with separable demands. Note that the analysis of this algorithm is one of the most sophisticated local ratio analyses in the literature. We present a 3-approximation algorithm for PCVC with inseparable demands. As far as we know, this algorithm is the first 3-approximation algorithm for CVC with inseparable demands. We also present a 2-approximation algorithm for PCVC with unit demands. This algorithm is much simpler and more intuitive than Mestre’s 2-approximation algorithm [25]. It is important to note that our algorithm is not a local ratio manifestation.
of Mestre's algorithm. While his algorithm relies on the algorithm for PVC of Gandhi et al. [15], our algorithm extends the algorithm for PVC from [4], and these two algorithms are different. We show that our algorithms can be extended to PCVC in hyper-graphs, where the maximum size of an edge is \( \Delta \). The approximation ratios for separable demands, inseparable demands, and unit demands are \( \Delta + 1 \), \( \Delta + 1 \), and \( \Delta \), respectively. Finally, we show that the same approximation ratios can be obtained for PC-CVC in hyper-graphs.
Chapter 2
Definitions and Preliminaries

2.1 Notation and terminology

Given an undirected graph \( G = (V, E) \), let \( E(u) \) be the set of edges incident on \( u \), and let \( N(u) \) be the set of \( u \)'s neighbors. Given an edge set \( F \subseteq E \) and a demand (or load) function \( \ell \) on the edges of \( G \), we denote the total demand of the edges in \( F \) by \( \ell(F) \). That is, \( \ell(F) = \sum_{e \in F} \ell(e) \). We define \( \text{deg}(u) = \ell(E(u)) \). Hence, in the unit demands case \( \text{deg}(u) \) is the degree of \( u \), i.e., \( \text{deg}(u) = |E(u)| = |N(u)| \).

We define \( \tilde{c}(u) = \min \{c(u), \text{deg}(u)\} \). This definition may seem odd, since we may assume that \( c(u) \leq \text{deg}(u) \) for every vertex \( u \) in the input graph. However, our algorithms repeatedly remove vertices from the given graph, and therefore we may encounter a graph in which there exists a vertex where \( \text{deg}(u) < c(u) \). We also define \( b(u) = \min \{c(u), \text{deg}(u), L\} = \min \{\tilde{c}(u), L\} \). \( b(u) \) can be viewed as the covering potential of \( u \) in the current graph. A single copy of \( u \) can cover \( c(u) \) of the demand if \( \text{deg}(u) \geq c(u) \), but if \( \text{deg}(u) < c(u) \), we cannot cover more than a total of \( \text{deg}(u) \) of the demand. Moreover, if \( L \) is smaller than \( \tilde{c}(u) \), we have nothing to gain from covering more than \( L \).

Let \( A \) be an assignment. The \textit{support} of an assignment \( A \) is the set of vertices \( V(A) = \{u : A(u) \neq \emptyset\} \). We denote by \( E(A) \) the set of edges covered by \( A \). That is, \( E(A) = \cup_{u} A(u) \). We define \( |A| = |E(A)| \) and \( \ell(A) = \ell(E(A)) \). Note that in the unit demand case \( \ell(A) = |A| \).

2.2 Small, medium, and large edges

Given a \textsc{PVCV} instance, we refer to an edge \( e = (u, v) \) as \textit{large} if \( \ell(e) > \max \{c(u), c(v)\} \). If \( \ell(e) \leq \min \{c(u), c(v)\} \) it is called \textit{small}. Otherwise, \( e \) is called \textit{medium}.

Consider the case of \textsc{PVCV} with inseparable demands. Since a large edge cannot be assigned to a single copy of one of its endpoints, it follows that no feasible solution contains large edges. Therefore, we may ignore large edges in the case of \textsc{PVCV} with inseparable demands. Notice that the instance may contain a medium edge \( e = (u, v) \) such that \( c(u) < \ell(e) \leq c(v) \). If there exist such an edge \( e \), then we may add a new vertex \( u_{e} \) to the graph, where \( w(u_{e}) = \infty \) and \( c(u_{e}) = \ell(e) \), and connect \( e \) to \( u_{e} \) instead of to \( u \). We
refer to this operation as an edge detachment. Since $e \notin A(u_e)$ in any solution $A$ of finite weight, it follows that we may assume, without loss of generality, that all edges are small. In the sequel, when we discuss PCVC with inseparable demand, we assume that all edges are small.

A problem might occur in the separable demands case when there exists a vertex $u$ such that $c(u) = 0$. In the sequel we assume without loss of generality that there are no vertices with zero capacity. If there exists such a vertex $u$ we simply define $c(u) = 1$ and $w(u) = \infty$.

2.3 Local ratio

The local ratio technique [8, 2, 5] is based on the Local Ratio Theorem, which applies to optimization problems of the following type. The input is a non-negative weight vector $w \in \mathbb{R}^n$ and a set of feasibility constraints $\mathcal{F}$. The problem is to find a vector $x \in \mathbb{R}^n$ that minimizes (or maximizes) the inner product $w \cdot x$ subject to the set of constraints $\mathcal{F}$.

**Theorem 1 (Local Ratio [5])** Let $\mathcal{F}$ be a set of constraints and let $w, w_1,$ and $w_2$ be weight vectors such that $w = w_1 + w_2$. Then, if $x$ is $r$-approximate with respect to $(\mathcal{F}, w_1)$ and with respect to $(\mathcal{F}, w_2)$, for some $r$, then $x$ is also an $r$-approximate solution with respect to $(\mathcal{F}, w)$.
Chapter 3

Partial Capacitated Covering with Separable Demands

We present a 3-approximation algorithm for PCVC with separable demands. At the heart of our scheme is Algorithm PCVC, which is inspired by the local ratio interpretation of the approximation algorithms for cvc from [16] and the local ratio approximation algorithm for PVC from [4].

3.1 Algorithm PCVC

In the description of Algorithm PCVC, we use a function called Next-Uncovered that, given a vertex $u$, returns an uncovered edge $e$ incident on $u$ with maximum demand. That is, it returns an edge $e \in E(u) \setminus A(u)$ such that $\ell(e) \geq \ell(e')$ for every $e' \in E(u) \setminus A(u)$. (If $A(u) = E(u)$ it returns NIL.)

The algorithm consists of a five-way if condition:

(i) If $L = 0$ return the empty assignment.

(ii) If there exists an edge $e_x$ incident on $x \in V$ such that $\ell(e_x) > \max \{L, c(x)\}$ then return the assignment $A(x) \leftarrow \{e_x\}$ and $A(v) \leftarrow \emptyset$, for all $v \neq x$.

(iii) If there exists a vertex with zero degree remove this vertex and solve the problem recursively on the resulting instance. item

(iv) If there exists a zero weight vertex $u$, remove $u$ and $E(u)$ and solve the problem recursively. Now, there are two options: either the solution returned is the empty assignment or not. In the first case $u$ collects uncovered edges in a non-increasing order of demands until the total demand of assigned edges is at least $L$. Then, if the total demand of assigned edges is less than $c(u)$, $u$ continues to collect as many edges as possible as long as the total demand is less than $c(u)$. Intuitively, if a single copy of $u$ is used, then it would be wise to assign as many edges as possible to this copy. In the second option the solution returned is not the empty assignment. In this case,
if the total demand of covered edges is less than $L$, all edges in $E(u)$ are assigned to $u$. Otherwise, no edges are assigned to $u$.

(v) If there are no zero weight vertices in $G$, then construct a weight function $w_1$ which is proportional to $b$ and subtracts $w_1$ from $w$. (Recall that $b(u) = \min \{\deg(u), c(u), L\}$.) Then, recursively compute a solution with respect to the new weights and return this solution. Observe that $w_1$ is constructed in a way that ensures that there exists a zero weight vertex with respect to the weight function $w_2 = w - w_1$.

Algorithm 1: PCVC$(V, E, w, L)$

1: if $L = 0$ then return $A(v) \leftarrow \emptyset$ for all $v \in V$
2: if there exists $x \in V$ and $e_x \in E(x)$ such that $\ell(e_x) > \max \{L, c(x)\}$ then return $A(x) \leftarrow \{e_x\}$ and $A(v) \leftarrow \emptyset$ for all $v \neq x$
3: if there exists $u \in V$ such that $\deg(u) = 0$ then return $\text{PCVC}(V \setminus \{u\}, E, w, L)$
4: if there exists $u \in V$ such that $w(u) = 0$ then
5: $A \leftarrow \text{PCVC}(V \setminus \{u\}, E \setminus \{u\}, w, \max \{L - \deg(u), 0\})$
6: if $V(A) = \emptyset$
7: while $\ell(A(u)) < L$ do $A(u) \leftarrow A(u) \cup \{\text{Next-Uncovered}(u)\}$
8: while $(A(u) \neq E(u))$ and $(\ell(A(u)) + \ell(\text{Next-Uncovered}(u)) < c(u))$ do $A(u) \leftarrow A(u) \cup \{\text{Next-Uncovered}(u)\}$
9: else
10: if $\ell(A) < L$ then $A(u) \leftarrow E(u)$
11: return $A$
12: Let $\varepsilon = \min_{u \in V} \{w(u)/b(u)\}$
13: Define the weight functions $w_1(v) = \varepsilon \cdot b(v)$, for every $v \in V$, and $w_2 = w - w_1$
14: return $\text{PCVC}(V, E, w_2, L)$

3.2 Feasibility of the solution

First, observe that there are $O(|V|)$ recursive calls. Hence, the running time of the algorithm is polynomial.

Consider the recursive call made in Line 5. In order to distinguish between the assignment obtained by this recursive call and the assignment returned in Line 11, we denote the former by $A'$ and the latter by $A$. Assignment $A'$ is for graph $G' = (V', E')$, and we denote the corresponding parameters $\alpha', \deg', \tilde{c}', \tilde{b}'$, and $L'$. Similarly, we use $\alpha$, $\deg$, $\tilde{c}$, $b$, and $L$, for the parameters of $A$ and $G = (V, E)$.

Lemma 1 Algorithm PCVC computes a partial capacitated vertex cover.

Proof First, notice that we assign only uncovered edges. We prove that $\ell(A) \geq L$ by induction on the depth of the recursion. Consider the base case. If the recursion ends in
Line 1 then $\ell(A) = 0 = L$. Otherwise the recursion ends in Line 2 and $\ell(A) = \ell(e_x) > L$. In both cases the solution is feasible. For the inductive step, if the recursive call was made in Line 3 or in Line 14, then $\ell(A) \geq L$ by the inductive hypothesis. If the recursive call was made in Line 5, then $\ell(A') \geq \max\{L - \deg(u), 0\}$ by the inductive hypothesis. If $V(A) = \emptyset$, then $\deg(u) \geq L$, and therefore $\ell(A(u)) \geq L$. Otherwise, if $V(A) \neq \emptyset$ then there are two options. If $\ell(A') \geq L$, then $A = A'$ and we are done. If $\ell(A') < L$, the edges in $E(u)$ are assigned to $u$, and since their combined demand is $\deg(u)$ it follows that $\ell(A) = \ell(A') + \deg(u) \geq L' + \deg(u) = L$.

3.3 Approximation Ratio

The assignment returned by Algorithm PCVC is not 3-approximate in general. For example, if there exists a very large edge whose covering is enough to attain feasibility then Line 2 will choose to cover the edge no matter how expensive its endpoints may be. Nevertheless, we can still offer the following slightly weaker guarantee.

Theorem 2 If the recursion of Algorithm PCVC ends in Line 1 then the assignment returned is 3-approximate. Otherwise, if the recursion ends in Line 2, assigning edge $e_x$ to vertex $x$, then the solution returned is 3-approximate compared to the cheapest solution that assigns $e_x$ to $x$.

The proof of the theorem is given in section 3.3.1.3.

Observe that if all edges are small then Line 2 is never executed. Hence, by Theorem 2, Algorithm PCVC computes 3-approximations when the instance contains only small edges. In section 3.4 it is shown that, in the absence of Line 2, the algorithm may fail to provide a 3-approximation if the problem instance contains medium or large edges.

Based on Theorem 2 it is straightforward to design a 3-approximation algorithm using Algorithm PCVC. First PCVC is run on the input instance. Suppose the recursion ends in Line 2 with edge $e_x$ assigned to vertex $x$. The assignment found is considered to be a candidate solution and set aside. Then the instance is modified by detaching edge $e_x$ from $x$. These steps are repeated until an execution of Algorithm PCVC ends its recursion in Line 1 with the empty assignment. At the end, a candidate solution with minimum cost is returned.

Theorem 3 There exists a 3-approximation algorithm for PCVC with separable demands.

Proof First, notice that in the algorithm just described once an edge is detached from an endpoint it cannot be detached from the same end later on. Hence, Algorithm PCVC is executed at most $O(|E|)$ times and the overall running time is polynomial.

We argue by induction on the number of calls to PCVC that at least one of the candidate solutions produced is 3-approximate. For the base case, the first call to PCVC ends with the empty assignment and by Theorem 2 the assignment is 3-approximate. For the inductive step, the first run ends with edge $e_x$ assigned to vertex $x$. If there exists an
optimal solution that assigns $e_x$ to $x$ then by Theorem 2 the assignment is 3-approximate. Otherwise the problem of finding a cover in the new instance, where $e_x$ is detached from $x$, is equivalent to the problem on the input instance. By inductive hypothesis, one of the later calls is guaranteed to produce a 3-approximate solution. We ultimately return a candidate solution with minimum cost, thus the theorem follows. ■

3.3.1 Analysis of Algorithm PCVC

In the next two sections we pave the road for the proof of Theorem 2. Our approach involves constructing a subtle charging scheme by which at each point we have at our disposal a number of coins that we distribute between the vertices. The charging scheme is later used in conjunction with the Local Ratio Theorem to relate the cost of the solution produced by the algorithm to the cost of an optimal solution.

We describe two charging schemes depending on how the recursion ends. We use the following observations in our analysis.

**Observation 1** Suppose a recursive call is made in Line 5. Then $\alpha(v) = \alpha'(v)$ for every $v \in V \setminus \{u\}$.

**Observation 2** $\alpha(v)\tilde{c}(v) \leq 2\deg(v)$ for every $v \in V$. Moreover, if $\alpha(v) > 1$ then $\alpha(v)\tilde{c}(v) \leq 2\ell(A(v))$

**Proof** Since $\tilde{c}(v) \leq \deg(v)$, the first claim is trivial if $\alpha(v) \leq 2$. If $\alpha(v) > 1$ then $\deg(v) \geq \ell(A(v)) > c(v)$, and therefore $\alpha(v)\tilde{c}(v) = \alpha(v)c(v) < \ell(A(v)) + c(v) \leq 2\ell(A(v)) \leq 2\deg(v)$. ■

3.3.1.1 The recursion ends with the empty assignment

In this section we study what happens when the recursion of Algorithm PCVC ends in Line 1. Let $(v)$ be the number of coins that are given to $v$. A charging scheme $(v)$ is called valid if $\sum_v (v) \leq 3L$. We aim to show that if the assignment $A$ was computed by Algorithm PCVC then there is a way to distribute $3L$ coins so that $(v) \geq \alpha(v)b(v)$ for every $v \in V$. The proof of the next lemma contains a recursive definition of our charging scheme. We denote by $v_0, v_1, \ldots$ the vertices of $V(A)$ in the order they join the set.

**Lemma 2** Let $A$ be an assignment that was computed by Algorithm PCVC for a graph $G = (V, E)$ and a covering demand $L$. Furthermore, assume the recursion ended in Line 1. Then, one the following conditions must hold:

1. $V(A) = \emptyset$. Also, the charging scheme $(v) = 0$ for every $v \in V$ is valid.

2. $V(A) = \{v_0\}$, $\alpha(v_0) = 1$, and $\ell(A(v_0)) \geq \frac{1}{2}c(v_0)$. Also, the charging scheme $(v_0) = 3L$ and $(v) = 0$ for every $v \neq v_0$ is valid.
There are four possible options:

1. $V(A) = \{v_0\}$, $\alpha(v_0) = 1$, and $\ell(A(v_0)) = \ell(e_0) < L$, for any vertex. Thus, if it satisfies one of the conditions it continues to do the corresponding charging scheme by assigning $(\alpha, e)$.

2. $V(A) = \{v_0, v_1\}$, $\alpha(v_0) = 1$, $\alpha(v_1) = 2$, and $\ell(A(v_0)) = \ell(e_0) = \frac{1}{2}c(v_0)$. Also, there exists a valid charging scheme $\mathcal{S}$ such that $\mathcal{S}(v_0) = L$, $\mathcal{S}(v_0) = 2\ell(A(v_0)) - (\ell(A) - L)$, $\mathcal{S}(v_1) = \alpha(v_1)c(v_1)$, and $\mathcal{S}(v) = 0$ for every $v \neq v_0, v_1$.

3. $V(A) = \{v_0\}, \alpha(v_0) = 2$, and $\ell(A(v_0)) - \ell(e_0) < L$, where $e_0$ is the last edge that was assigned to $v_0$. Also, the charging scheme $\mathcal{S}(v_0) = 3L$ and $\mathcal{S}(v) = 0$ for every $v \neq v_0$ is valid.

4. $V(A) = \{v_0, v_1\}$, $\alpha(v_0) = 1$, $\alpha(v_1) = 2$, and $\ell(A(v_0)) = \frac{1}{2}c(v_0)$. Also, there exists a valid charging scheme $\mathcal{S}$ such that $\mathcal{S}(v_0) = L$, $\mathcal{S}(v_0) = 2\ell(A(v_0)) - (\ell(A) - L)$, $\mathcal{S}(v_1) = \alpha(v_1)c(v_1)$, and $\mathcal{S}(v) = 0$ for every $v \neq v_0, v_1$.

5. $\alpha(v_0) = 1$, $\ell(A(v_0)) = \frac{1}{2}c(v_0)$, $\alpha(v) = 2$ for every $v \in V(A) \setminus \{v_0\}$, and $L > \deg(v_0) - \ell(A(v_0))$. Also, there exists a valid charging scheme $\mathcal{S}$ such that $\mathcal{S}(v_0) = 2L + \ell(A(v_0)) - (\ell(A) - L)$, $\mathcal{S}(v) = \alpha(v)c(v)$ for every $v \in A(v) \setminus \{v_0\}$, and $\mathcal{S}(v) = 0$ for every $v \not\in V(A)$.

6. $V(A) \neq \emptyset$. Also, there exists a valid charging scheme $\mathcal{S}$ such that $\mathcal{S}(v) \geq \alpha(v)c(v)$ for every $v \in V$.

**Proof** We prove the lemma by induction on the length of the creation series of the solution. (Recall that there are at most $O(|V|)$ recursive calls.) Specifically, we assume that one of the conditions holds and prove that the augmented solution always satisfies one of the conditions. The possible transitions from condition to condition are given in Figure 3.1. First, at the base of the induction the computed solution is the empty assignment and therefore Condition 1 holds. For the inductive step, there are three possible types of recursive calls corresponding to Line 3, Line 5, and Line 14 of Algorithm PCVC. The calls in Line 3 and Line 14 do not change the assignment that is returned by the recursive call, nor does it change $b$ for any vertex. Thus, if it satisfies one of the conditions it continues to satisfy them. The only correction is needed in the case of Line 3, where we need to extend the corresponding charging scheme by assigning $\mathcal{S}(u) = 0$.

For the rest of the proof we concentrate on recursive calls that are made in Line 5. We consider a solution $A'$ that was computed by the recursive call, and denote by $\mathcal{S}'$ the charging scheme that corresponds to $A'$.

Consider a recursive call in which $A'$ satisfies Condition 1. In this case, $V(A) = \{u\}$. There are four possible options:
Consider a recursive call in which \( A \) satisfies Condition 2. That is, \( V(A') = \{v_0\} \), \( \alpha'(v_0) = 1 \), and \( \ell(A'(v_0)) \geq c(v_0)/2 \). There are three possible options:

(2 \( \rightarrow \) 6) If \( \alpha(u) \geq 2 \) then we show that Condition 6 holds. Consider the charging scheme \( S(v) = S'(v) + 2 \deg(u), S(u) = \deg(u) \), and \( S(v) = S'(v) \) for every \( v \neq v_0, u \). Observe that \( \deg(u) \leq c(u) \). Hence, \( S(u) = \deg(u) = \alpha(u)c(u) \). Also, \( \ell(A(v_0)) < L \) since \( \alpha(u) > 0 \). Therefore, \( S(v_0) = 3L' + 2 \deg(u) \geq 2L \geq 2\ell(A(v_0)) \geq c(v_0) \). Hence, \( S(v) \geq \alpha(v)c(u) \) for every \( v \) and Condition 6 holds.

(2 \( \rightarrow \) 4) If \( \alpha(u) \geq 2 \) then we show that Condition 4 holds. Consider the charging scheme \( S(v_0) = S'(v_0) + \deg(u), S(u) = 2 \deg(u), \) and \( S(v) = S'(v) \) for every \( v \neq v_0, u \). Since \( \ell(A(u)) = \deg(u) > c(u) \) it follows that \( S(u) = 2 \deg(u) \geq \alpha(u)c(u) \). Moreover, \( S(v_0) \geq L \) since \( S'(v_0) \geq L' \). Hence, it remains to show that \( S(v_0) \geq 2\ell(A(v_0)) - (\ell(A) - L) \). Since \( \alpha(u) > 0 \) we know that \( \deg(u) > \ell(A') - L' \). Also, observe that \( \ell(A') - L' = \ell(A) - L \). Hence, \( S(v_0) \geq 2L' + \deg(u) = 2\ell(A') - 2(\ell(A') - L') + \deg(u) > 2\ell(A(v_0)) - (\ell(A) - L) \).

Consider a recursive call in which \( A' \) satisfies Condition 3. That is, \( V(A) = \{v_0\} \), \( \alpha(v_0) = 2 \), and \( \ell(A'(v_0)) - \ell(e_0) < L' \). There are two possible options:

(3 \( \rightarrow \) 3) If \( \alpha(u) = 0 \) then \( A \) satisfies Condition 3, since \( \ell(A(v_0)) - \ell(e_0) < L' < L \).

(3 \( \rightarrow \) 6) If \( \alpha(u) > 0 \) then we show that Condition 6 holds. Consider the charging scheme \( S(v_0) = S'(v_0) + \deg(u), S(u) = 2 \deg(u), \) and \( S(v) = S'(v) \) for every \( v \neq v_0, u \). First, since \( \ell(A(u)) = \deg(u) > c(u) \) it follows that \( S(u) = 2 \deg(u) \geq \alpha(u)c(u) \). As for \( v_0 \), first observe that since \( \alpha(v_0) = 2 \) and \( \alpha(u) > 0 \) it follows that \( c(v_0) < \ell(A(v_0)) < L \). We claim that \( A(v_0) \) contains at least two edges, otherwise \( A' \) could only have been
constructed in Line 2 and we assume the recursion ends in Line 1. Since we add edges to \( A(v_0) \) in a non-decreasing order of demands it follows that \( \ell(v_0) \leq \ell(A(v_0))/2 \). This implies that \( L' \geq \ell(A(v_0))/2 > c(v_0)/2 \) because \( \ell(A'(v_0)) - \ell(v_0) < L' \). It follows that \( \$(v) = 2L' + L \geq c(v_0) + c(v_0) = \alpha(v_0)c(v_0) \).

Consider a recursive call in which \( A' \) satisfies Condition 4. That is, \( V(A') = \{v_0, v_1\}, \alpha'(v_0) = 1, \alpha'(v_1) \geq 2, \) and \( \ell(A'(v_0)) \geq \frac{1}{2}c(v_0) \). There are two possible options:

\((4 \rightarrow 4)\) If \( \alpha(u) = 0 \) then we show that Condition 4 holds. Since \( A = A' \) the assignment properties hold. Consider the charging scheme \( $v_0 = $'v_0 + \deg(u), $u = 0, \) and \( $v = $'v \) for every \( v \neq v_0, u \). First, \( $v_0 = $'v_0 + \deg(u) \geq L' + \deg(u) = L \). Also, \( $v_0 = $'v_0 + \deg(u) \geq 2\ell(A'(v_0)) - (\ell(A') - L') + \deg(u) = 2\ell(A(v_0)) - (\ell(A) - L) \).

\((4 \rightarrow 6)\) If \( \alpha(u) > 0 \) then we show that Condition 6 holds. Consider the charging scheme \( $v_0 = $'v_0 + \deg(u), $u = 2\deg(u), \) and \( $v = $'v \) for every \( v \neq v_0, u \). First, since \( \alpha(u) > 0 \) we know that \( \ell(A') - L' < \deg(u) \). Hence, \( $v_0 = $'v_0 + \deg(u) \geq 2\ell(A'(v_0)) - (\ell(A') - L') + \deg(u) \geq 2\ell(A(v_0)) \geq c(v_0) \). It follows that \( $v_0 \geq \alpha(v_0)c(v_0) \). \( v_1 \) is funded since \( $'v_1 \geq \alpha(v_1)c(v_1) \). As for \( u \), \( $u = 2\deg(u) \geq \alpha(u)c(u) \).

Consider a recursive call in which \( A' \) satisfies Condition 5. That is, \( \alpha(v_0) = 1, \ell(A(v_0)) < \frac{1}{2}c(v_0), \alpha(v) \geq 2 \) for every \( v \in V(A) \setminus \{v_0\} \), and \( \ell(A(v_0)) > \deg(v_0) - L \). There are two possible options:

\((5 \rightarrow 5)\) If \( \alpha(u) \neq 1 \), then we show that Condition 5 continues to hold. We first show that \( \deg(v_0) - \ell(A(v_0)) < L \). We know that \( \deg(v_0) - \ell(A(v_0)) < L' \) since \( A' \) satisfies Condition 5. Since \( \ell(A(v_0)) = \ell(A'(v_0)) \), \( \deg(v_0) \leq \deg(v_0) + \deg(u) \), and \( L = L' + \deg(u) \), if follows that \( \deg(v_0) - \ell(A(v_0)) < L \).

If \( \alpha(u) = 0 \) we define \( $v_0 = $'v_0 + 2\deg(u), $u = 0 \) and \( $v = $'v \) for every \( v \neq v_0, u \). In this case, \( $v_0 = $'v_0 + 2\deg(u) \geq 2L' + \ell(A'(v_0)) - \ell(A') + 2\deg(u) = 2L + \ell(A(v_0)) - \ell(A) \).

If \( \alpha(u) \geq 2 \) we define \( $v_0 = $'v_0 + \deg(u), $u = 2\deg(u) \) and \( $v = $'v \) for every \( v \neq v_0, u \). In this case, \( $v_0 = $'v_0 + \deg(u) \geq 2L' + \ell(A'(v_0)) - \ell(A') + \deg(u) = 2L + \ell(A(v_0)) - \ell(A) \). Also, \( $u = 2\deg(u) \geq \alpha(u)c(u) \).

In both cases if \( v \in V(A) \setminus \{v_0, u\} \), then \( $v = $'v = \alpha'(v)c(v) = \alpha(v)c(v) \).

\((5 \rightarrow 6)\) If \( \alpha(u) = 1 \), we show that Condition 6 holds. We define a charging scheme \$ as follows. \( $v_0 = $'v_0 + 2\deg(u), $u = \deg(u), \) and \( $v = $'v \) for every \( v \neq v_0, u \). First, \( $v \geq \alpha'(v)c(v) = \alpha(v)c(v) \), for every \( v \in V(A) \setminus \{v_0, u\} \). Also, \( $u = \deg(u) \geq \alpha(u)c(u) \) since \( \alpha(u) = 1 \) and \( \ell(A(u)) = \deg(u) \). It remains to take care of \( v_0 \). Observe that, \( $v_0 = $'v_0 + 2\deg(u) \geq 2L' + \ell(A'(v_0)) - \ell(A') + 2\deg(u) > L + \ell(A(v_0)) \) because \( \ell(A') - L' < \deg(u) \). Since \( \deg(v_0) - \ell(A'(v_0)) \) it follows that \( \deg(v_0) - \ell(A(v_0)) < L \). Therefore, \( $v_0 > \deg(v_0) \geq \alpha(v_0)c(v_0) \).
Consider a recursive call in which $A'$ satisfies Condition 6. In this case we only have one option:

(6 → 6) We define a charging scheme $\$ as follows: $\$(u) = 2 \deg(u)$, $\$(v) = \$'(v) + \ell(u,v)$ if $v \in V(A)$ and $(u,v) \in E$, and $\$(v) = \$'(v)$ otherwise. First, notice that $\$(u) = 2 \deg(u) \geq \alpha(u)\tilde{c}(u)$ by Observation 2. Next, we show that $\$(v) \geq \alpha(v)\tilde{c}(v)$ for every $v \neq u$. First, if $\alpha(v) = 0$ or $\tilde{c}(v) = \tilde{c}'(v)$ then this is clearly true. Let $v$ be a vertex for which $\alpha(v) > 0$ and $\tilde{c}(v) \neq \tilde{c}'(v)$. It follows that $\alpha(v) = \alpha'(v) = 1$. Also, since $\$(v) = \$'(v) + \ell(v,u)$ it follows that $\$(v) \geq \tilde{c}'(v) + \ell(v,u) \geq \tilde{c}(v)$.

The lemma follows.

\begin{lemma}
Let $A$ be an assignment that was computed by Algorithm \textsc{PCVC} for a graph $G = (V, E)$ and a covering demand $L$. Furthermore, assume the recursion ended in Line 1. Then, there exists a charging scheme (with respect to $G$ and assignment $A$) in which every vertex $v$ is given at least $\alpha(v)b(v)$ coins and $\sum_v \alpha(v)b(v) \leq 3L$.
\end{lemma}

\begin{proof}
The lemma follows from Lemma 2 and the definition of $b$.
\end{proof}

### 3.3.1.2 The recursion ends with a medium or large edge

In this section we study what happens when Algorithm \textsc{PCVC} ends its recursion in Line 2 assigning edge $e_x$ to vertex $x$. The approach is similar to that used in the previous section, the main difference being that since we want to compare our solution to one that assigns $e_x$ to $x$ we have more coins to distribute. More specifically, we say a charging scheme $\$ is \emph{valid} if $\sum_v \$(v) \leq 3 \max \{L, \alpha(x)c(x)\}$. We show that there is a way to distribute these coins so that $\$(v) \geq \alpha(v)b(v)$. The proof of the next lemma contains a recursive definition of our charging scheme.

\begin{lemma}
Let $A$ be an assignment that was computed by Algorithm \textsc{PCVC} for a graph $G = (V, E)$ and a covering demand $L$. Furthermore, assume the recursion ended in Line 2 assigning edge $e_x$ to $x$. Then, one the following conditions must hold:

1. $V(A) = \{x\}$. Also, the charging scheme $\$(x) = \alpha(x)c(x)$ and $\$(v) = 0$ for every $v \neq x$ is valid.

2. $V(A) = \{x, v_1\}$, $\alpha(v_1) > 1$. Also, the charging scheme $\$(x) = \alpha(x)c(x)$, $\$(v_1) = 2\ell(A(v_1))$ and $\$(v) = 0$ for every $v \neq x, v_1$ is valid.

3. $V(A) \neq \emptyset$. Also, there exists a valid charging scheme $\$ such that $\$(v) \geq \alpha(v)\tilde{c}(v)$ for every $v \in V$ and $\sum_v \$(v) \leq 3L$.

\end{lemma}
There are three possible options:

1) The recursive call, and denote by $A$ the charging scheme that corresponds to $A$'s recursive call; $\ell' = 2 \ell(A(u))$. Notice that $\ell(A(v)) + \ell(e_x) < L$ since $\alpha(u) > 0$ and $\ell(A(u)) + \ell(e_x) < L$ (Line 2 in $v_1$'s recursive call) and $\ell(A(u)) + \ell(A(v_1)) < L$ (Line 1 in $x$'s recursive call). Furthermore, $\ell(e_x) > 0$, $\ell(v_1) = 2 \ell(A(v_1))$ and $\ell(u) = 2 \ell(A(u))$. Adding up these inequalities we get $\ell(e_x) + \ell(v_1) + \ell(u) \leq 3L$, thus Condition 3 holds.

2) If $\alpha(u) > 1$ then $A' = A$ and Condition 2 holds.

3) If $\alpha(u) = 0$ then $A' = A$ and Condition 1 holds. (This may happen, only when $\ell(e_x) = L$.)

Proof We prove the lemma by induction on the length of the creation series of $A$. We assume that one of the conditions holds and prove that the augmented solution always satisfies one of the conditions. The possible transitions from condition to condition are given in Figure 3.2. At the base of the induction the computed solution assigns edge $e_x$ to $x$ and Condition 1 holds. For the inductive step the same argument used in the proof of Lemma 2 handles for recursive calls that occurs in Line 3 and 14. Thus, we focus on recursive calls that are made in Line 5. We consider a solution $A'$ that was computes by the recursive call, and denote by $\mathcal{S}'$ the charging scheme that corresponds to $A'$.

Consider a recursive call in which $A'$ satisfies Condition 1. In this case, $V'(A) = \{x\}$. There are three possible options:

1) $\alpha(u) = 0$, then $A' = A$ and Condition 1 holds.

2) $\alpha(u) > 1$ we show that Condition 2 holds. Let $\mathcal{S}(u) = 2 \deg(u)$ and $\mathcal{S}(v) = \mathcal{S}'(v)$ for any $v \neq u$. Note that $\deg(u) < L$ and $\deg(u) = c(u)$ and $\deg(x) = 2$. Also, because of Line 2, we know that $\ell(e_x) \leq L$. If $L > \alpha(x)c(x)$ then $\mathcal{S}(x) + \mathcal{S}(u) \leq 3L$. If $L \leq \alpha(x)c(x)$ then $\mathcal{S}(x) + \mathcal{S}(u) \leq \alpha(x)c(x) + 2L \leq 3\alpha(x)c(x)$.

3) $\alpha(u) = 1$ then $\ell(A(u)) = \deg(u) < L$ and $c(u) = \deg(u)$. Let $\mathcal{S}(u) = \deg(u)$ and $\mathcal{S}'(v) = \mathcal{S}(v)$ for all $v \neq u$. Thus, $\mathcal{S}(u) = \alpha(u)c(u)$. Because of Line 2 we know that $\ell(e_x) \leq L$. Therefore $\mathcal{S}(x) + \mathcal{S}(u) \leq 2\ell(e_x) + \deg(u) \leq 3L$ and Condition 3 holds.

Consider a recursive call in which $A'$ satisfies Condition 2. There are two possible options:

1) $\alpha(u) = 0$, then $A' = A$ and Condition 1 holds.

2) $\alpha(u) > 1$ then $A' = A$ and Condition 2 holds.

3) $\alpha(u) = 1$ then $A' = A$ and Condition 1 holds. (This may happen, only when $\ell(e_x) = L$.)

Figure 3.2: Possible transitions between the conditions.
Lemma 5  Let $A$ be an assignment that was computed by Algorithm PCVC for a graph $G = (V, E)$ and a covering demand $L$. Furthermore, assume the recursion ended in Line 2 assigning edge $e_x$ to $x$. Then, there exists a charging scheme (with respect to graph $G$ and assignment $A$) in which every vertex $v$ is given at least $\alpha(v)b(v)$ coins and $\sum_v \alpha(v)b(v) \leq 3 \max \{L, \alpha(x)c(x)\}$.

Proof  The lemma follows from Lemma 4 and the definition of $b$. 

3.3.1.3  Proof of Theorem 2

For the sake of brevity when we say that a given assignment is 3-approximate we mean compared to an optimal solution if the recursion of Algorithm PCVC ended in Line 1, and compared to the cheapest solution that assigns $e_x$ to $x$ if the recursion ended in Line 2.

Our goal is to prove that the assignment produced by Algorithm PCVC is 3-approximate. The proof is by induction on the recursion. In the base case the algorithm returns an empty assignment (if the recursion ends in Line 1) or assigns $e_x$ to $x$ (if the recursion ends in Line 2), in both cases the assignment is optimal. For the inductive step there are three cases.

First, if the recursive call is made in Line 3, then by the inductive hypothesis the assignment $A'$ is 3-approximate with respect to $(V \setminus \{u\}, E)$. $A'$ is clearly 3-approximate with respect to $(V, E)$ since $\deg(u) = 0$.

Second, if the recursive invocation is made in Line 5, then by the inductive hypothesis the assignment $A'$ is 3-approximate with respect to $(V \setminus \{v\}, E \setminus E(u))$, $w$, and $\max \{L - \deg(u), 0\}$. Since $w(u) = 0$, the optimum with respect to $(V, E)$, $w$, and $L$ is equal to the optimum with respect to $(V \setminus \{v\}, E \setminus E(u))$, $w$, and $\max \{L - \deg(u), 0\}$. Moreover, since $\alpha(v) = \alpha'(v)$ for every $v \in V \setminus \{u\}$ due to Observation 1, it follows that $w(A) = w(A')$. Thus $A$ is 3-approximate with respect to $(V, E)$, $w$, and $L$.

Third, if the recursive call is made in Line 14, then by the inductive hypothesis the assignment $A'$ is 3-approximate with respect to $(V, E)$, $w_2$, and $L$. If the recursion ended in Line 1 then by Lemma 3 $w_1(A) \leq \varepsilon \cdot 3L$. We show that $w_1(A) \geq \varepsilon \cdot L$ for every feasible assignment $\tilde{A}$. First, if there exists $v \in V(\tilde{A})$ such that $b(v) = L$ then $w_1(\tilde{A}) \geq L \cdot \varepsilon$. Otherwise, $b(v) = \tilde{c}(v) < L$ for every $v \in V(\tilde{A})$. It follows that $w_1(\tilde{A}) = \varepsilon \cdot \sum_v \tilde{c}(v) \geq \varepsilon \cdot \sum_v \tilde{c}(v) \geq \varepsilon \cdot \sum_v \tilde{c}(v) \geq \varepsilon \cdot L$. Hence, if the recursion ended in Line 1, $A$ is 3-approximate with respect to $w_1$ too, and by the Local Ratio Theorem it is 3-approximation with respect to $w$ as well. On the other hand, if the recursion ended in Line 2 then by Lemma 5 $w_1(A) \leq \varepsilon \cdot 3 \max \{L, \alpha(x)c(x)\}$. We need to show that $w_1(A) \geq \varepsilon \cdot \max \{L, \alpha(x)c(x)\}$ for any feasible cover $\tilde{A}$ that assigns $e_x$ to $x$. By the previous argument we know that $w_1(\tilde{A}) \geq \varepsilon \cdot L$. In addition, because $e \in A(x)$, we have $w_1(\tilde{A}) \geq \alpha(x)w_1(x) \geq \alpha(x)\alpha(x)w_1(x) = \varepsilon \cdot \alpha(x)b(x)$. Since the if condition in Line 2 was not met in this call we have $\ell(e_x) \leq L$ and so $b(x) = c(x)$. Thus, if the recursion ended in Line 2, $A$ is 3-approximate with respect to $w_1$ too, and by the Local Ratio Theorem it is 3-approximate with respect to $w$ as well. 

19
3.4 Algorithm PCVC without Line 2

In this section we show that, in the absence of Line 2, Algorithm PCVC may fail to provide a 3-approximation if the problem instance contains medium or large edges.

Consider a graph that contains three edges whose endpoints are disjoint. (See Figure 3.3.) We define $c(u_1) = c(u_2) = L - 2$, $c(v_1) = c(v_2) = L - 1$, and $c(u_3) = c(v_3) = L$. Also let $\ell(u_1, v_1) = \ell(u_2, v_2) = L - 1$, and $\ell(u_3, v_3) = L$. The weight of the vertices is as follows: $w(u_1) = w(u_2) = L - 2$, $w(v_1) = w(v_2) = L - 1 + \delta$, and $w(u_3) = w(v_3) = L + \delta$, where $\delta > 0$ is a very small constant. We note that the edges in this instance are either small or medium. Given this instance, Algorithm PCVC (without Line 2) will compute the solution $A$ where $A(u_1) = \{(u_1, v_1)\}$, $A(u_2) = \{(u_2, v_2)\}$, and $A(v) = \emptyset$ for every $v \neq u_1, u_2$. This is because $w(u_1) = b(u_1)$ and $w(u_2) = b(u_2)$ while $w(v) > b(v)$ for every $v \neq u_1, u_2$. Observe that while $w(A) = 4(L - 2)$, there exists a solution that contains a single copy of $u_3$, and the weight of this solution is $L + \delta$.

![Figure 3.3: A pcvc instance with medium edges.](image)


Chapter 4

Partial Capacitated Covering with Inseparable Demands

In this chapter we show that Algorithm PCVC computes 3-approximate solutions for partial capacitated vertex cover with inseparable demands. Notice that when the instance has only small edges the algorithm always ends its recursion in Line 1. Therefore, by Theorem 2 the assignment found is 3-approximate for separable demands. We basically show that the charging scheme that was defined in Lemma 2 distributes enough coins in order to pay for the additional copies needed due to the inseparable demands.

Given an assignment \( A \), let \( \beta(v) \) be the number of copies of \( v \) needed when the edges in \( A(u) \) are assigned to copies of \( v \) using FIRST-FIT in a non-increasing order of demands.

**Observation 3** Let \( A \) be an assignment that was computed by Algorithm PCVC. If \( \alpha(v) \geq 2 \) then \( \beta(v)c(v) \leq 2\ell(A(v)) \) for every \( v \in V \).

**Proof** Since \( \beta \) is the number of copies needed using FIRST-FIT, it follows that all copies of \( v \), except maybe one, must be more than half full. The total length of edges assigned to the two most vacant copies of \( v \) is more than \( c(v) \). Hence, \( \beta(v) \leq 2 + \frac{\ell(A(v)) - c(v)}{\alpha(v)/2} = \frac{2\ell(A(v))}{\alpha(v)} \). \( \blacksquare \)

**Lemma 6** Let \( A \) be an assignment that was computed by Algorithm PCVC for a graph \( G = (V, E) \) and a covering demand \( L \). Then, there exists a charging scheme (with respect to graph \( G \) and assignment \( A \)) in which every vertex \( v \) is given at least \( \beta(v)b(v) \) coins and \( \sum_v \beta(v)b(v) \leq 3L \).

**Proof** Let \( \$ \) be the charging scheme that is defined in the proof of Lemma 2. We show that \( \$ \) satisfies \( \$(v) \geq \beta(v)b(v) \) for every \( v \). First, if \( \alpha(v) \leq 1 \) then \( \beta(v) = \alpha(v) \) and by Lemma 3 we are done. Let \( v \) be a vertex such that \( \alpha(v) \geq 2 \), and consider the recursive call in which \( v \) joined \( V(A) \). Since \( \alpha(v) \geq 2 \), it follows that \( \ell(A(v)) > c(v) \). There are two options: either \( v \) was given edges by Line 7 or by Line 10. Line 8 is not involved since \( \ell(A(v)) > c(v) \).
If \( v \) was given edges by Line 10, then \( A(v) = E(v) \) and by the definition of \( $ \), \( v \) gets 
\( 2\deg(v) = 2\ell(A(v)) \) coins. This can be verified in the transitions \((2 \rightarrow 4)\), \((3 \rightarrow 6)\), 
\((4 \rightarrow 6)\), \((5 \rightarrow 5)\), and \((6 \rightarrow 6)\) of Lemma 2. Hence, by Observation 3 it follows that 
\( \$ (v) \geq \beta(v)c(v) \geq \beta(v)b(v) \).

If edges were assigned to \( v \) in Line 7, then \( A \) satisfies Condition 3 (\( \alpha(v) = 2 \)) or Condition 6 (\( \alpha(v) \geq 3 \)). If \( A \) satisfies Condition 6, then \( \$ (v) = 3L \). Let \( e_v \) be the last edge that 
was assigned to \( v \). Clearly, \( \ell(A(v)) - \ell(e_v) < L \). Also, since \( \alpha(v) \geq 3 \) and all edges are small, it follows that \( \ell(e_v) \leq \ell(A(v))/3 \). Hence, \( \$ (v) = 3L > 3(\ell(A(v)) - \ell(e_v)) \geq 2\ell(A(v)) \) 
and Observation 3 it follows that \( \$ (v) \geq \beta(v)c(v) \geq \beta(v)b(v) \).

If \( A \) satisfies Condition 3, then \( \ell(A(v)) - \ell(e_0) < L \leq \ell(A(v)) \), where \( e_0 \) is the last 
edge that was assigned to \( v \). First, consider the case where \( \ell(A(v)) - \ell(e_0) \leq c(v) \). In 
this case \( \beta(v) = \alpha(v) = 2 \) and by Lemma 3 we get that \( \$ (v) \geq \beta(v)b(v) \). Next, consider the case where \( \ell(A(v)) - \ell(e_0) > c(v) \). Assume that we use FIRST-FIT to assign the 
edges from \( A(v) \setminus \{e_0\} \) to copies of \( v \), and denote by \( \beta'(v) \) the resulting number of 
copies. Due to Observation 3, it follows that \( \beta'(v) \leq 2\ell(A(v) \setminus \{e_0\})/c(v) \leq 2L/c(v) \). Since 
\( L > \ell(A(v) \setminus \{e_0\}) > c(v) \) and \( \beta(v) \leq \beta'(v) + 1 \), it follows that 
\( \beta(v)b(v) = \beta(v)c(v) \leq \beta'(v)c(v) + c(v) \leq 3L = \$ (v) \).

By using Lemma 6 instead of Lemma 2 in the proof of Theorem 2 we obtain the following:

**Theorem 4** Algorithm PCVC computes 3-approximations for PCVC with inseparable demands.
Chapter 5

Partial Capacitated Vertex Cover with Unit Demands

5.1 Algorithm PCVC2

Algorithm PCVC2 is a recursive local ratio algorithm that computes 2-approximate solutions for PCVC with unit demands. As in previous sections we define \( b(u) = \min \{ \deg(u), c(u), L \} \), and \( \bar{c}(u) = \min \{ \deg(u), c(u) \} \). Also, given an assignment \( A \), a vertex \( v \) is called vulnerable if \( 0 < |A(v)| < \bar{c}(v) \) and there exists an uncovered edge \( e \) that is incident on it.

Algorithm PCVC2 consists of a four-way if condition similar to the one that is found in Algorithm PCVC. Specifically, Algorithm PCVC2 differs from Algorithm PCVC only in the fourth entry of the if condition. In Algorithm PCVC2, if there exists a zero weight vertex \( u \), \( u \) and \( E(u) \) are removed from the graph and the problem is solved recursively. Then, as long as there are less than \( L \) covered edges, uncovered edges are assigned to vertices while giving precedence to certain vulnerable vertices. That is, as long as there exists a vulnerable vertex of a certain kind an uncovered edge is assigned to it. If there are no more such vulnerable vertices, then edges are assigned to \( u \).

5.2 Feasibility of the solution

Observe that there are \( O(|V|) \) recursive calls. Hence, the running time of the algorithm is polynomial.

Lemma 7 Algorithm PCVC2 computes a partial capacitated vertex cover.

Proof First, notice that we assign only uncovered edges. We prove by induction on the recursion that \( |A| = L \). In the base case \( |A| = 0 = L \) and we are done. For the inductive step, if the recursive call was made in Line 2 or in Line 12, then \( |A| = L \) by the inductive hypothesis. If the recursive call was made in Line 4, then \( |A'| = \max \{ L - \deg(u), 0 \} \) by the inductive hypothesis. Since the edges incident on \( u \) are not covered by \( A' \), the while loop is able to increase the number of covered edges by \( \deg(u) \), if necessary. Hence, \( |A| = L \).
Algorithm 2: \text{PCVC2}(V, E, w, L)

1: if $L = 0$ then return $A(v) = \emptyset$ for all $v \in V$
2: if there exists $u \in V$ such that $\deg(u) = 0$ then return $\text{PCVC2}(V \setminus \{u\}, E, w, L)$
3: if there exists $u \in V$ such that $w(u) = 0$ then
4: $A \leftarrow \text{PCVC2}(V \setminus \{u\}, E \setminus E(u), w, \max\{L - \deg(u), 0\})$
5: while $|A| < L$ do
6: if $V(A) = \{v_0\}$ and $v_0$ is vulnerable then
7: $A(v_0) \leftarrow A(v_0) \cup \{e\}$, where $e \in E(v_0)$ is uncovered
8: elseif there exists a vulnerable vertex $v \in N(u)$ then
9: $A(v) \leftarrow A(v) \cup \{(u, v)\}$
10: else
11: $A(u) \leftarrow A(u) \cup \{e\}$, where $e \in E(u)$ is uncovered
12: return $A$

13: Let $\epsilon = \min_{u \in V} \{w(u)/b(u)\}$
14: Define the weight functions $w_1(v) = \epsilon \cdot b(v)$, for every $v \in V$, and $w_2 = w - w_1$
15: return $\text{PCVC2}(V, E, w_2, L)$

after the while loop.

5.3 Approximation Ratio

We use the following observations in our analysis.

Observation 4 If there exists $v \in V(A)$ such that $b(v) = L$ then $V(A) = \{v\}$.

Proof The proof is by induction on the recursion. In the base case $A$ is the empty assignment, and we are done. For the inductive step there are two cases. If $|V(A)| = 1$, then the claim is satisfied. Otherwise, $|V(A)| > 1$. By the inductive hypothesis it follows that either $b'(v) < L'$ for every $v \in V(A')$ or $V(A') = \{v\}$ and $b'(v) = L'$. First, in both cases, $\deg(u) < L$, since otherwise $A'(v) = \emptyset$, for every $v$, a contradiction. Hence, $b(u) < L$. If $b'(v) < L'$ for every $v \in V(A')$, then $b(v) < L$ for every $v \in V(A')$. On the other hand, assume that $V(A') = \{v\}$ and $b'(v) = L'$. If $b(v) = L$ then $v$ is vulnerable throughout the while loop (Lines 5–8), which means that $V(A) = \{v\}$, in contradiction to $|V(A)| > 1$.

Observation 5 Suppose a recursive call is made in Line 4. Let $v \in N(u)$. Then,

1. If $v$ is vulnerable with respect to $G$ and $A'$, then $\alpha(v) = \alpha'(v) = 1$ and $\tilde{c}(v) \leq \tilde{c}'(v) + 1$. Thus $\alpha(v)\tilde{c}(v) \leq \alpha'(v)\tilde{c}'(v) + 1$.

2. If $v$ is not vulnerable with respect to $G$ and $A'$, then $A(v) = A'(v)$ and therefore $\alpha(v) = \alpha'(v)$. Moreover, there are two (not mutually exclusive) cases: $\alpha(v) = \alpha'(v) = 0$ or $\deg(v) \geq c(v) + 1$. In the latter case the degree of $v$ in $G'$ is at least $c(v)$, and therefore $\tilde{c}(v) = \tilde{c}'(v) = c(v)$. Thus in either case $\alpha(v)\tilde{c}(v) = \alpha'(v)\tilde{c}'(v)$.
Observation 6 Suppose a recursive call is made in Line 4. Then, $\alpha(v) = \alpha'(v)$ for every $v \in V \setminus \{u\}$.

Proof Clearly, $\alpha(v) = \alpha'(v)$ for every $v$ that is not vulnerable, with respect to $G$ and $A'$. Let $v$ be a vulnerable vertex. If $v \in N(u)$ this follows from Observation 5. Otherwise, it follows from the fact that edges are assigned to $v$ only if it is vulnerable.

Observation 7 $\alpha(u)b(u) \leq |A(u)| + b(u)$.

Proof $|A(u)| + b(u) = \left(\frac{|A(u)|}{b(u)} + 1\right) \cdot b(u) \geq \left(\frac{|A(u)|}{c(u)} + 1\right) \cdot b(u) > \alpha(u)b(u)$.

5.3.1 Analysis of Algorithm PCVC2

To analyze the algorithm, we use a charging scheme by which at each point we have at our disposal $2L$ coins that we may distribute between the vertices. We denote by $\$\(v\)$ the number of coins that are given to $v$. We aim to show that there is a way to distribute the coins so that $\$\(v\) \geq \alpha(v)b(v)$. We distribute the coins as follows. First, as long as there is only one vertex in $V(A)$ this vertex gets all the coins. Furthermore, whenever an edge is assigned to a vulnerable vertex $v$, then one coin is given to $v$ and the second to $u$, otherwise both of them are given to $u$. Since $|A| = L$, by Lemma 7, it follows that the number of coins that were distributed is exactly $2L$.

Lemma 8 Let $A$ be an assignment that was computed by Algorithm PCVC2 for a graph $G = (V, E)$ and a covering demand $L$. Then, there exists a charging scheme (with respect to graph $G$ and assignment $A$) in which every vertex $v$ is given at least $\alpha(v)b(v)$ coins and $\sum_v \alpha(v)b(v) \leq 2L$.

Proof The proof is by induction on the recursion. In the base case $A(v) = \emptyset$, for every $v \in V$. Hence, every vertex $v$ gets $\alpha(v)b(v) = 0$ coins, and $\sum_v \alpha(v)b(v) = 0 = 2L$.

For the inductive step, there are three possible types of recursive calls corresponding to Line 2, Line 4, and Line 12 of Algorithm PCVC2. The calls in Line 2 and Line 12 do not change the assignment that is returned by the recursive call. Thus, the claim follows from the inductive hypothesis. The only correction is needed in the case of Line 2, where we need to extend the corresponding charging scheme by assigning $\$\(u\) = 0$. For the rest of the proof we concentrate on recursive calls that are made in Line 4.

If the recursive call is made in Line 4 then by the inductive hypothesis there exists a charging scheme that allots at least $\alpha'(v)b'(v)$ coins to every vertex in $G' = (V', E')$ and uses $2L'$ coins. Observe that the transition from $G'$ to $G$ consists of adding a single vertex $u$ and the edges incident on it. Furthermore, there are three possible transformations from $A'$ to $A$. In the first case, $|A'| = 0$ and hence $L$ edges are assigned to $u$. In the second case only $u$ and a vertex $v_0$ such that $V(A') = \{v_0\}$ participate in the change. $v_0$ is not necessarily vulnerable. The third possible transformation involves $u$ and its neighbors. We
extend the charging scheme of $G'$ and $A'$ by distributing the remaining $2(L - L')$ coins in the three cases.

In the first case, $u$ receives $2|A(u)| = 2L$ coins. It follows that $\$ (u) = |A(u)| + L$. Since $|A(u)| + L \geq |A(u)| + b(u)$, $u$ is satisfied due to Observation 7.

In the second case, $\alpha (v) = \alpha' (v) = 0$ for every $v \neq u, v_0$, and therefore any vertex $v \neq u, v_0$ is satisfied. We distribute $2(L - L') = 2 \deg (u)$ coins as follows. If $A(u) = \emptyset$, then the coins are given to $v_0$. This means that $v_0$ posses $2|A(v_0)|$ coins, and using the same argument as in the first case we are done. We now assume that $A(u) \neq \emptyset$. In this case, for each edge that was assigned to $v_0$, both $u$ and $v_0$ receive a single coin, and for each edge that was assigned to $u$, we give $u$ two coins and none to $v_0$. That is, $u$ receives $\deg (u) + |A(u)|$ coins, while $v_0$ gets the rest of the coins. By Observation 7, $u$ receives at least $\alpha (u)b(\nu)$ coins. As for $v_0$, it now has $2|A'(v_0)| = 2L'$ coins from the charging scheme of $G'$ and $A'$, and also $|A(v_0)| - |A'(v_0)|$ coins from this recursive call. If $v_0$ was not vulnerable in the beginning of the while loop, then $A(v_0) = A'(v_0)$ and $b(v_0) \leq \tilde{c}(v_0) \leq A(v_0)$. Hence, $\$ (v_0) = 2|A(v_0)| \geq |A(v_0)| + b(v_0)$, and therefore by Observation 7 $v_0$ has enough coins. Next, we assume that $v_0$ was vulnerable in the beginning of the while loop. Since $v_0$ is not vulnerable at the end of the while loop and $A(u) \neq \emptyset$, it follows that $|A(v_0)| = \tilde{c}(v_0)$. Hence, $\alpha (v_0) = 1$ and $\$ (v_0) \geq |A(v_0)| = \alpha(v_0)b(v_0)$.

In the third case, we only need to take care of $u$ and its neighbors that participate in the cover. We note that due to Observation 4 we know that $b'(v) < L'$, for every $v \in V(A')$. Hence, $b'(v) = \tilde{c}'(v)$, for every $v \in V(A')$. Hence, the inductive hypothesis implies that $\$ (v) \geq \alpha' (v)b'(\nu) = \alpha'(v)\tilde{c}'(v)$. We extend the charging scheme of $G'$ and $A'$ in the following way. For each $v \in N(u)$, if $(u, v)$ is assigned to $v$, both $u$ and $v$ receive a coin, otherwise we give $u$ both coins. Consider $v \in N(u)$ such that $A(v) \neq \emptyset$. If $v$ is vulnerable then the edge $(u, v)$ is assigned to $v$, and therefore it receives a coin, which by Observation 5 is enough to satisfy $v$. If $v$ is not vulnerable, then due to Observation 5 we are done. Finally, as mentioned in Lemma 7, $|A'| = L'$, and therefore exactly $\deg (u)$ edges are assigned by the while loop. Therefore, all the neighbors of $u$ will be satisfied after the while loop. As for $u$ itself, it is given $|A(u)| + \deg (u)$ coins and it is satisfied due to Observation 7.

**Theorem 5** Algorithm **PCVC2** computes a 2-approximate partial capacitated vertex cover.

**Proof** The proof is by induction on the recursion. In the base case, the algorithm returns an empty assignment, which is optimal. For the inductive step there are three cases:

If the recursive invocation is made in Line 2, then by the inductive hypothesis the assignment is 2-approximate with respect to $(V \setminus \{u\}, E)$. It follows that it is also a 2-approximate with respect to $(V, E)$ and we are done.

If the recursive invocation is made in Line 4, then by the inductive hypothesis the assignment $A'$ is 2-approximate with respect to $G'$ and $\max \{L - \deg (u), 0\}$. Clearly, the optimum value for $G$ can only be greater than or equal to the optimum value for $G'$, and because $w(u) = 0$ and $\alpha (v) = \alpha'(v)$ for all $v \in V \setminus \{u\}$ by Observation 6, it follows that $w(A) = w(A')$. Thus $A$ is 2-approximate with respect to $G$. 

26
If the recursive call is made in Line 12, then by the inductive hypothesis the assignment is 2-approximate with respect to $w_2$. By Lemma 8 $w_1(A) \leq \varepsilon \cdot 2L$. Furthermore, we show that $w_1(\bar{A}) \geq \varepsilon \cdot L$ for every feasible assignment $\bar{A}$. If there exists $v \in V(\bar{A})$ such that $b(v) = L$ then $w_1(\bar{A}) \geq L \cdot \varepsilon$. Otherwise, $b(v) = \tilde{c}(v) < L$ for every $v \in V(\bar{A})$. It follows that $w_1(\bar{A}) = \varepsilon \cdot \sum_v \bar{\alpha}(v) \tilde{c}(v) \geq \varepsilon \cdot \sum_v |\bar{A}(v)| \geq \varepsilon \cdot L$. Thus, $A$ is 2-approximate with respect to $w_1$ as well, and by the Local Ratio Theorem it is 2-approximate with respect to $w$ as well. ■
Chapter 6

Partial Capacitated Vertex Cover in Hyper-Graphs

In this chapter we consider the partial capacitated vertex cover problem in hyper-graphs. We show that our algorithms easily extend to hyper-graphs. The resulting approximation ratios are $\Delta + 1$ for PCVC with separable demands or with inseparable demands, and $\Delta$ for PCVC with unit demands, where $\Delta \geq 2$ is the maximum size of an edge.

6.1 Notation and Terminology

Given a hyper-graph $H = (V, E)$, let $E(u)$ be the set of edges incident on $u$, and let $N(u)$ be the set of of vertices that share an edge with $u$, i.e., $N(u) = \{v : \exists e, u, v \in e\}$. As before we define $\text{deg}(u) = \ell(E(u))$. We also denote by $\bar{\ell}(u, v)$ the total load of the edges incident on both $u$ and $v$. Formally, $\bar{\ell}(u, v) = \sum_{e : u, v \in e} \ell(e)$. If $H$ is a graph, then $\bar{\ell}(u, v) = \ell(u, v)$ if $(u, v) \in E$, and $\bar{\ell}(u, v) = 0$ otherwise.

Given a PCVC instance, we refer to an edge $e$ as large if $\ell(e) > c(u)$ for every $u \in e$. If $\ell(e) \leq c(u)$ for every $u \in e$ it is called small. Otherwise, $e$ is called medium. As in graphs, we may assume without loss of generality that in the case of PCVC with inseparable demands all edges are small. We may also assume without loss of generality that there are no vertices with zero capacity. If there exists such a vertex $u$ we simply remove $u$ from the hyper-graph.

6.2 Non-Unit Demands

In this section we show that our results for PCVC with separable demands and inseparable demands extend to hyper-graphs.

First, we show that the algorithm for PCVC with separable demands (Chapter 3) computes a $(\Delta + 1)$-approximate solution on hyper-graphs. We do so by extending Theorem 2 to hyper-graphs. We modify the proof of Theorem 2 by fixing Lemmas 2 and 4.

Consider the case where the recursion ends with the empty assignment. A charging scheme $\$ is called valid if $\sum_v \$ (v) \leq (\Delta + 1)L$. We show that if the assignment $A$ was
computed by Algorithm PCVC then there is a way to distribute \((\Delta + 1)L\) coins so that 
\(\$ (v) \geq \alpha (v)b(v)\). We do so by fixing the proof of Lemma 2. Since \(\Delta \geq 2\) it follows that the only transition that should be modified is the transition \((6 \rightarrow 6)\). In fact, this transition is the cause for the increase in the approximation ratio. Consider a recursive call in which \(A’\) satisfies Condition 6. In this case we only have one option:

\((6 \rightarrow 6)\) We define a charging scheme \(\$\) as follows: \(\$ (u) = 2 \deg(u)\) and \(\$ (v) = \$'(v) + \bar{\ell}(u,v)\) for every \(v \in V\). First, notice that \(\$ (u) = 2 \deg(u) \geq \alpha (u)\tilde{c}(u)\) by Observation 2. Next, we show that \(\$ (v) \geq \alpha (v)\tilde{c}(v)\) for every \(v \neq u\). First, if \(\alpha (v) = 0\) or \(\tilde{c}(v) = \tilde{c}'(v)\) then this is clearly true. Let \(v\) be a vertex for which \(\alpha (v) > 0\) and \(\tilde{c}(v) \neq \tilde{c}'(v)\). In this case \(\deg'(v) < c(v)\) while \(\deg(v) > \deg'(v)\). It follows that \(\alpha (v) = \alpha'(v) = 1\). Since \(\$ (v) = \$'(v) + \bar{\ell}(u,v)\), it follows that \(\$ (v) \geq \tilde{c}'(v) + \bar{\ell}(u,v) = \deg'(v) + \bar{\ell}(u,v) = \deg(u) \geq \alpha (u)\tilde{c}(u)\).

All that is left to show is that the charging scheme is valid. This is easily proved by noticing that the coin distribution was performed only on edges incident on \(u\) and that for each such edge no more than \(\Delta + 1\) coins were distributed. Hence, not more than \((\Delta + 1)\deg(u)\) coins were distributed. Combined with fact that up until this stage we spent up to \((\Delta + 1)L'\) coins, the total amount of coins used is at most \((\Delta + 1)L' + (\Delta + 1)\deg(u) = (\Delta + 1)L\) and we are done.

Next, consider the case where the recursion ends with a medium or large edge. A charging scheme \(\$\) is called valid if \(\sum_v \$ (v) \leq (\Delta + 1) \max \{L, \alpha (x)c(x)\}\). We show that there is a way to distribute these coins such that \(\$ (v) \geq \alpha (v)\tilde{c}(v)\) for every vertex \(v \in V\). We do so by fixing Lemma 4 and its proof. First, Condition 3 of Lemma 4 is changed to:

3. \(V(A) \neq \emptyset\). Also, there exists a valid charging scheme \(\$\) such that \(\$ (v) \geq \alpha (v)\tilde{c}(v)\) for every \(v \in V\) and \(\sum_v \$ (v) \leq (\Delta + 1)L\).

As for the proof of the lemma, since \(\Delta \geq 2\) it follows that the only transition that should be modified is the transition \((3 \rightarrow 3)\). The proof of this transition is identical to the transition \((6 \rightarrow 6)\) in the hyper-graph version of Lemma 2.

The proof of Theorem 3 is fixed by replacing 3 by \(\Delta + 1\).

**Theorem 6** There exists a \((\Delta + 1)\)-approximation algorithm for PCVC with separable demands in hyper-graphs.

Given the extended version of Lemma 2, it is straightforward to extend Lemma 6 and Theorem 4 to hyper-graphs.

**Theorem 7** Algorithm PCVC is a \((\Delta + 1)\)-approximation algorithm for PCVC with inseparable demands in hyper-graphs.
6.3 Unit Demands

In this section we show that Algorithm PCVC2 computes $\Delta$-approximate solutions for PCVC with unit demands in hyper-graphs.

First, we present an extended version of Observation 5 for the case of hyper-graphs.

Observation 8 Suppose a recursive call is made in Line 4. Let $v \in N(u)$. Then,

1. If $v$ is vulnerable with respect to $G$ and $A'$, then $\alpha(v) = \alpha'(v) = 1$ and $\hat{c}(v) \leq \hat{c}'(v) + \bar{\ell}(u, v)$. Thus, $\alpha(v)\hat{c}(v) \leq \alpha'(v)\hat{c}'(v) + (\hat{c}(v) - \hat{c}'(v)) \leq \alpha'(v)\hat{c}'(v) + \ell(u, v)$.

2. If $v$ is not vulnerable with respect to $G$ and $A'$, then $A(v) = A'(v)$ and therefore $\alpha(v) = \alpha'(v)$. Moreover, there are two (not mutually exclusive) cases: $\alpha(v) = \alpha'(v) = 0$ or $\deg(v) \geq c(v) + \bar{\ell}(u, v)$. In the latter case the degree of $v$ in $G'$ is at least $c(v)$, and therefore $\hat{c}(v) = c'(v) = c(v)$. Thus in either case $\alpha(v)\hat{c}(v) = \alpha'(v)\hat{c}'(v)$.

To analyze the algorithm for the case of hyper-graphs, we use a charging scheme by which at each point we have at our disposal $\Delta L$ coins that we may distribute between the vertices.

Lemma 9 Let $A$ be an assignment that was computed by Algorithm PCVC2 for a graph $G = (V, E)$ and a covering demand $L$. Then, there exists a charging scheme (with respect to graph $G$ and assignment $A$) in which every vertex $u$ is given at least $\alpha(v)b(v)$ coins and $\sum_v \alpha(v)b(v) \leq \Delta L$.

Proof The proof is almost identical to the proof of Lemma 8. The only difference is in recursive calls that where made in Line 4. More specifically, in a possible transformation that involves $u$ and its neighbors. In this case, we need only take care of $u$ and its neighbors that participate in the cover. We note that due to Observation 4 we know that $b'(v) < L'$, for every $v \in V(A')$. Hence, $b'(v) = \hat{c}'(v)$, for every $v \in V(A')$. Hence, the inductive hypothesis implies that $\delta(v) \geq \alpha'(v)b'(v) = \alpha'(v)\hat{c}'(v)$. We extend the charging scheme of $G'$ and $A'$ in the following way. For each $e \in E(u)$, if $e$ is assigned to $u$, then $u$ receives two coins, otherwise we give one coin to each vertex $v \in e$ (including $u$). Observe that we distribute at most $\Delta \deg(u) = \Delta(L - L')$ coins.

Consider $v \in N(u)$ such that $A(v) = 0$. Since $|A'| = L'$, exactly $\deg(u)$ edges are assigned by the while loop. Hence, if $v$ is vulnerable at the beginning of the while loop it receives either at least $c(v) - \hat{c}'(v)$ coins or exactly $\bar{\ell}(u, v)$ coins. By Observation 8 this is enough to satisfy $v$. If $v$ is not vulnerable, then $\delta(v) \geq \delta'(v)$, and therefore due to Observation 8 we are done. Finally, $u$ is given $|A(u)| + \deg(u)$ coins and it is satisfied due to Observation 7.

The proof of the next theorem is identical to that of Theorem 5 where we now use lemma 9 instead of lemma 8.

Theorem 8 Algorithm PCVC2 computes a $\Delta$-approximate partial capacitated vertex cover in hyper-graphs.
Chapter 7

Prize Collecting Capacitated Vertex Cover

In this chapter we extend our results to PC-CVC. Specifically, we show that our algorithms may be used to obtain approximate solutions for PC-CVC with separable, inseparable, and unit demands. The approximation ratios are $\Delta + 1$, $\Delta + 1$, and $\Delta$, respectively.

7.1 Simple Reduction to Capacitated Vertex Cover

We first observe that a PC-CVC instance containing a hyper-graph with maximum edge size $\Delta$ can be viewed as a CVC instance that contains a hyper-graphs with maximum size edge $\Delta + 1$. Given a PC-CVC instance $(H = (V, E), c, \ell, w)$ we construct a CVC instance $(H' = (V', E'), c', \ell', w')$ as follows. We add a new vertex $z_e$ for every edge $e \in E$, and add $z_e$ to $e$. $z_e$ is called the anchor of $e$. That is, $V' = V \cup \{z_e : e \in E\}$, and $E' = \{e \cup \{z_e\} : e \in E\}$. Furthermore, we define $c'(v) = c(v)$ and $w'(v) = w(v)$ for every $v \in V$, and $c'(z_e) = \ell(e)$ and $w'(z_e) = w(e)$ for every $e \in E$.

This simple reduction implies that PC-CVC can be approximated using algorithms for CVC. That is, to solve PC-CVC we may use the $\Delta$-approximation algorithm for CVC with unit demands and the $(\Delta + 1)$-approximation algorithm for CVC with separable demands by Guha et al. [16], and our $(\Delta + 1)$-approximation algorithm from Chapter 4 in the case of CVC with inseparable demands. The resulting approximation ratios are $\Delta + 1$, $\Delta + 2$, and $\Delta + 2$, respectively.

In the rest of this section we assume that a PC-CVC instance is given to us in a form of such a CVC instance, and we show how to improve these ratios to $\Delta$, $\Delta + 1$, and $\Delta + 1$.

7.2 Capacitated Vertex Cover with Separable Demands

We consider CVC with separable demands. We show that in this case Algorithm PCVC becomes much simpler. In fact, it converges to a local ratio interpretation of the 3-approximation algorithm for CVC with separable demands from [16].
Examine Algorithm PCVC in the cvc case. In this case \( L = \ell(E) \), and therefore the while loop of Line 7 always terminates when \( A(u) = E(u) \). It follows that the algorithm always skips the while loop in Line 8, which means that Lines 6–10 can be replaced by \( A(u) \leftarrow E(u) \). Furthermore, Line 2 can be remove since the condition of the if statement is never met, that is, \( \ell(e) \leq \ell(E) = L \). Also, notice that \( b(v) = \tilde{c}(v) \) for every \( v \). This brings us to Algorithm CVC.

Algorithm 3: CVC \((V, E, w)\)

1: if \( E = \emptyset \) then return \( A(v) = \emptyset \) for all \( v \in V \)
2: if there exists \( u \in V \) such that \( \deg(u) = 0 \) then return CVC\((V \setminus \{u\}, E, w)\)
3: if there exists \( u \in V \) such that \( w(u) = 0 \) then
4: \( A \leftarrow \text{CVC}(V \setminus \{u\}, E \setminus E(u), w) \)
5: \( A(u) \leftarrow E(u) \)
6: return \( A \)
7: Let \( \varepsilon = \min_{u \in V} \{w(u)/\tilde{c}(u)\} \)
8: Define the weight functions \( w_1(v) = \varepsilon \cdot \tilde{c}(v) \), for every \( v \in V \), and \( w_2 = w - w_1 \)
9: return CVC\((V, E, w_2)\)

In this case, Lemma 3 can be replaced by:

Lemma 10 Let \( A \) be an assignment that was computed by Algorithm CVC for a graph \( G = (V, E) \). Then, there exists a charging scheme (with respect to graph \( G \) and assignment \( A \)) in which every vertex \( v \) is given at least \( \alpha(v)\tilde{c}(v) \) coins and \( \sum_v \alpha(v)\tilde{c}(v) \leq (\Delta + 1)\ell(E) \).

Proof We show that either Condition 1 holds or Condition 6 holds, where we refer to the conditions of Lemma 2. For this purpose we need only prove that the only possible transition from Condition 1 is to Condition 6. Consider a recursive call in which \( A' \) satisfies Condition 1. In this case, \( V(A') = \{u\} \). Since \( L = \ell(E) \) it follows that \( \ell(A(u)) = \deg(u) \), and therefore \( \$\!(u) = (\Delta + 1)L = (\Delta + 1)\deg(u) \geq \alpha(u)\tilde{c}(u) \) by Observation 2.

7.3 A More Careful Analysis

We show that Algorithm CVC actually computes \((\Delta + 1)\)-approximate solutions on instances of PC-cvc with separable demands, where the maximum size of an edge is \( \Delta \). We do so by providing a more careful analysis for this special case of cvc with maximum edge size \( \Delta + 1 \). Specifically, we show that if an assignment \( A \) was computed by Algorithm CVC, then there is a way to distribute \((\Delta + 1)L\) coins so that \( \$\!(v) \geq \alpha(v)\tilde{c}(v) \).

Lemma 11 Let \( A \) be an assignment that was computed by Algorithm CVC given a PC-cvc instance with maximum edge size \( \Delta \). Then, there exists a charging scheme (with respect to graph \( G \) and assignment \( A \)) in which every vertex \( v \) is given at least \( \alpha(v)\tilde{c}(v) \) coins and \( \sum_v \alpha(v)\tilde{c}(v) \leq (\Delta + 1)\ell(E) \).

32
The proof is almost the same as the proof of Lemma 10. The difference is that in this case we distribute $(\Delta + 1)\ell(E)$ coins and not $(\Delta' + 1)\ell(E) = (\Delta + 2)\ell(E)$ coins. Hence, we need to show how to save $\ell(E)$ coins. (Recall that a PC-CVC instance with maximum edge size $\Delta$ can be viewed as a PCVC instance with maximum edge size $\Delta' = \Delta + 1$.) The key observation is that there can be two cases: either $e$ is assigned to $z_e$ or to another vertex $v \in e$. If $e$ is assigned to $z_e$ then it is enough to allot $\deg(z_e) = \ell(e)$ coins to $z_e$ and to every other vertex $v \in e$. (According to the proof of Lemma 2 $z_e$ should get $2\ell(e)$ coins.) $z_e$ is satisfied since $\alpha(z_e) = 1$ and therefore $\delta(z_e) = \deg(z_e) = \alpha(z_e)\bar{c}(z_e)$. On the other hand, if $e$ is assigned to a vertex $v \neq z_e$, then $A(z_e) = \emptyset$ and therefore $z_e$ does not receive any coins. It follows that $2\ell(e)$ coins are given to $v$ and at most $(\Delta - 1)\ell(e)$ coins are distributed between the other vertices of the edge $e$.)

The proof of the next theorem is similar to the proof of Theorem 2.

**Theorem 9** Algorithm CVC computes $(\Delta + 1)$-approximate solutions in the case of PC-CVC with separable demands.

Our result can be extended to PC-CVC with inseparable demands as shown in Chapter 4.

**Theorem 10** Algorithm CVC computes $(\Delta + 1)$-approximate solutions in the case of PC-CVC with inseparable demands.

The arguments of Lemma 11 can be applied to PC-CVC with unit demands.

**Theorem 11** Algorithm PCVC2 computes $\Delta$-approximate solutions in the case of PC-CVC with unit demands.
Chapter 8

Conclusions and Open Problems

Main Results

Here we summarize our main results:

1. **PCVC with separable demands**: We showed a 3-approximation algorithm for PCVC with separable demands. This algorithm is the basis of all the other algorithms shown in this thesis. The analysis done to prove its approximation ratio relied on a complex charging scheme in conjunction with the local ratio theorem.

2. **PCVC with inseparable demands**: We showed a 3-approximation algorithm for PCVC with inseparable demands. The main idea was based on taking the solution returned by our algorithm for PCVC with separable demands and transforming it to a feasible solution for PCVC with inseparable demands without increasing the approximation ratio.

3. **PCVC with unit demands**: We showed that if all the edges have a demand of one, a better approximation ratio can be achieved. We presented a 2-approximation algorithms for PCVC with unit demands. This improved algorithm exploited the fact that in the case of unit demands we can fill used copies completely without increasing the number of copies (if enough edges exist).

4. **Extensions to Hyper-Graphs**: We showed how our analysis of the previous three algorithms can be extended to hyper-graphs. We showed that in PCVC on hyper graphs the approximation ratios are $\Delta + 1$, $\Delta + 1$ and $\Delta$ for PCVC with separable demands, PCVC with inseparable demands and PCVC with unit demands respectively.

5. **PC-CVC**: We presented a $(\Delta + 1)$-approximation algorithm for the prize collecting version of CVC with separable and inseparable demands and a $\Delta$-approximation algorithm for the unit demands case. The algorithm was based on a reduction to regular CVC in hyper graphs and a more careful analysis of the charging scheme.
Open Problems

Another variation of PCVC which we did not discuss in this thesis is the case of separable splittable demands. In this version we are allowed to cover an edge by splitting its load between its endpoints. Notice that we do not allow partial coverage of an edge i.e. if we cover part of the load of an edge we must cover all of its load.

Our algorithm for PCVC with separable demands also computes 3-approximate solutions for PCVC with separable splittable demands. It is interesting to check if eliminating the requirement that an edge is covered wholly by one of its end points will allow for a better approximation algorithm.
Bibliography


בעיית הכיסוי בצמתים

נתון לנו גרף לא מכוון \( G = (V, E) \). נשתמש במקבל \( V \subseteq V \) \( w(v) \).

לכל צומת \( v \) מוטפח העומס \( \deg v \).

פיזיבילי פתרון ננער את סתת \( U \subseteq V \) \( e(u, v) \) \( u \in U \) \( v \in U \).

מטרתנו היא למצוא פתרון פיזיבילי בעלות מינימאלית.

בעיית הכיסוי בצמתים היא אחת הבעיות המפורסמות ביותר במדעי המחשב.

ידוע כי בעיית הכיסוי בצמתים היא \( \mathbb{NP} \)-רissentית ולא ניתן לה崂 mediante אלגוריתם הנочный.

והו הרחבה של בעיית הכיסוי בצמתים

לכל צומת \( v \) מוטף קיבול \( c(v) \) משווה את מספר הקשתות של צומת \( v \)üp כאן הקבוצה \( A \in E \).

我々ון \( u \) \( e \in A(u) \) \( v \in \alpha(u,v) \).

 thumbnails של החשיפות \( \alpha \).

עומס אופייני \( c(v) \) נספון ב-

לכל צומת \( v \) \( c(v) \) \( |A(v)| \leq c(v) \).

 intéressant \( \alpha \).

הנושאים של הפשטות \( A \).

הנושאים \( c(v) \).

נושאstdin

בעיית הכיסוי בצמתים עם קיבולים

בעיית הכיסוי בצמתים עם קיבולים \( (CVC) \).

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי לצמצום עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בר הפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיית הכיסוי בצמתים עם קיבולים ועומס בפרדה

בעיि
the maximum cut must be equal to the cost of the solution for a given input when the load is not in the cut, therefore, the solution when the load is in the cut.

The solution for the load in the cut is another example of the load covering problem, which is in turn a special case of the load covering problem. (PVC).

In this problem, the input consists of a directed graph \( G = (V, E) \) where \( V \) is the set of vertices and \( E \) is the set of edges. The goal is to find a subset \( U \subseteq V \) of minimum weight such that at least \( L \) of the edges are covered in the cut.

The cost of the solution is defined as
\[
\sum_{v \in V} \alpha(v) w(v),
\]
where \( \alpha(v) \) is the number of times each vertex \( v \) is covered and \( w(v) \) is the weight of vertex \( v \).

In this problem, the goal is to find a load covering solution that has the minimum cost.

### Load Covering Problem with Capacity

In the load covering problem with capacity (PCVC), there is an additional constraint that if an edge is assigned to a vertex, then all of its weight must be assigned to it.

### Load Covering Problem with Prize Collection

In the load covering problem with prize collection (PC), we do not necessarily have to cover all the edges. Instead, we pay a penalty for each uncovered edge. The goal is to find a cover of minimum cost subject to the constraint that the total weight of the covered edges is at least \( L \).
The goal of our work is to find a solution that minimizes the number of edges used in the graph. In this work, we are interested in the case where there are capacities on the vertices and each vertex can be assigned a weight. The objective is to find the minimum number of edges such that the sum of the weights of the edges incident to each vertex is less than or equal to the capacity of the vertex.

We present algorithms for the different cases. For the case of coverage with capacities, we present an algorithm with a $(1+\Delta)$ approximation, where $\Delta$ is the maximum degree in the hypergraph.

We show that our algorithms work well for the prize-collecting problem, and we also provide a $(1+\Delta)$ approximation for the case where there are capacities on the hyperedges.

The algorithms we present are based on a general technique for designing and analyzing approximation algorithms. This technique results in simple algorithms that can be used for a variety of problems, including coverage with capacities and prize-collecting problems.

Technion - Computer Science Department - M.Sc. Thesis MSC-2008-02 - 2008
רשימת איורים

<table>
<thead>
<tr>
<th>מספר</th>
<th>ת목</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>דוגמה הממחישה את ההבדל בין עומסים נתונים ובketerangan נתונים לקפרס</td>
</tr>
<tr>
<td>3.1</td>
<td>התוכנה האップפרים בין התנאים של למונה</td>
</tr>
<tr>
<td>3.2</td>
<td>התוכנה האップפרים בין התנאים של למונה</td>
</tr>
<tr>
<td>3.3</td>
<td>דוגמה המראה מספר אגלוגית PCVC מספר בשתי דגש בחלקה 3-כיוון סוף לשורש 2 מספרים</td>
</tr>
<tr>
<td>4</td>
<td>מספר ממשלות</td>
</tr>
</tbody>
</table>
אלגוריתמי קירוב לבעיות כיסוי חלקיوت
עם קיבולים

גיא פליישר
אלגוריתמי קירוב לבעיות כיסוי חלקי
עם קיבולים

היבחר על מחקר
לשם מילוי חלקים של הדירישות לקבבה החוזרת המגיעה
למדעי המחשב

גיא פליישר

הוזה לסנט הטכניון – מכון טכנולוגי לישראל

חרף

מרץ 2007

אדר תשס"ז
המחק נועש בהנחיית פרופ' ראובן בר יהודה בפקולטה למדעי המחשב בטכניון.

ברצוני להודות להנחייתו עморה/of Prof. ראובן בר יהודה על הנחייתו, תמיכתו, עזרתיו וצלחתו.
אנוי קדימה בעץ חזרה.

ברצוני להודות לד"ר דוד רבין על עזרתיו הרביה על עדותי המועילות.

אני מודה לטכניון על התמיכה הקספית והנדסת ההשכלה שלי.

אני מודה לטלני על התמודדות הכספית והנדסת ההשכלה והשלמותתי.

מחק זה מוקדש לכלב האהוב דידי אשר פסק Trọng לקפץ לפינת השלמת בחורה זו.

מבחנים בעשו בחנויות פוריפ' ראובן בר יהודה בפקולטה למדעי המחשב בטכניון.