Approximation of Partial Capacitated Vertex Cover

Reuven Bar-Yehuda* Guy Flysher* Julian Mestre† Dror Rawitz‡

January 21, 2007

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 demand version. We present 3-approximation algorithms for both PCVC with inseparable demands and PCVC with separable demands. We also present a 2-approximation for PCVC with unit demands. Our analyses rely on the local ratio technique and sophisticated charging schemes.

1 Introduction

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 [17], and cannot be approximated within a factor of $10\sqrt{5} - 21 \approx 1.36$, unless P=NP [9]. On the positive side, there are several 2-approximation algorithms for vertex cover (for more details see [16, 5] and references therein).

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 $A : V \to 2^E$. 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 $|A(u)| \le \alpha(u)c(u)$. 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.

*Department of Computer Science, Technion, Haifa 32000, Israel. E-mail: {reuven, guyfl}@cs.technion.ac.il.
†Department of Computer Science, University of Maryland, College Park, MD 20742, USA. E-mail: jmestre@cs.umd.edu. Supported by the University of Maryland Dean's Dissertation Fellowship.
‡Caesarea Rothschild Institute, University of Haifa, Haifa 31905, Israel. E-mail: rawitz@cri.haifa.ac.il.

In a more general version of CVC we are given a *demand* or *load* function $\ell : E \to \mathbb{N}$. 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 $\sum_{e \in A(u)} \ell(e) \le \alpha(u)c(u)$. 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 Appendix A.) 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.

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.

In this paper 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.

Motivation. CVC was proposed by Guha et al. [13] that 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 A. 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 [19].

CVC can also 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 the problem of cellular network planning. In this problem 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.

Related work. Capacitated covering problems date back to Wolsey [23] (see also [2, 8]). Wolsey presented a greedy algorithm for weighted set cover with hard capacities that achieves a logarithmic approximation ratio.

Guha et al. [13] presented a primal-dual 2-approximation algorithm for CVC. (A local ratio version of this algorithm was given in [5].) 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 Δ. The resulting approximation ratios are Δ and $\Delta + 1$, respectively. Guha et al. [13] also studied CVC in trees. They obtained a polynomial time algorithm for this problem and proved that CVC with separable edge demands in trees is NP-hard.

Gandhi et al. [11] presented a 2-approximation algorithm for CVC using an LP-rounding method called *dependent rounding*. Chuzhoy and Naor [8] 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 this version of CVC, which is based on randomized rounding with alterations. They also proved that the weighted version of this problem is as hard to approximate as set cover. Gandhi et al. [10] improved the approximation ratio for unweighted vertex cover with hard capacities to 2.

The *partial set cover problem* was first studied by Kearns [18], who proved that the performance ratio of the greedy algorithm is at most $2H_m+3$, where H_m is the mth harmonic number. Slavík [21] showed that it is actually bounded by H_m. Bshouty and Burroughs [7] obtained the first polynomial time 2-approximation algorithm for PVC. Bar-Yehuda [3] studied PVC with demands and presented a local ratio 2-approximation algorithm for this problem. His algorithm extends to a Δ-approximation algorithm in hyper-graphs in which the maximum size of an edge is Δ. Gandhi et al. [12] 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 [12], Srinivasan [22] 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 [15] developed a $(2 - \Theta(\frac{\ln \ln d}{\ln d}))$-approximation algorithm for PVC.

Recently, Mestre [20] 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. [13] and on the 2-approximation algorithm for PVC by Gandhi et al. [12].

We note that the parameterized complexity of PVC and CVC was investigated by Guo et al. [14]. They showed that CVC is fixed-parameter tractable, while PVC is $W[1]$-hard. It follows that PCVC is also $W[1]$-hard.

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 [13] and the local ratio 2-approximation algorithm for PVC from [3].

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 [20]. 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. [12], our algorithm extends the algorithm for PVC from [3], and these two algorithms are different.

In the full version of the paper we show that our algorithms can be extended to PCVC in hyper-graphs, where the maximum size of an edge is Δ. The approximation ratios for separable demands, inseparable demands, and unit demands are $\Delta + 1$, $\Delta + 1$, and Δ, respectively.

2 Preliminaries

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 ℓ 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 $\deg(u) = \ell(E(u))$. Hence, in the unit demands case $\deg(u)$ is the degree of u, i.e., $\deg(u) = |E(u)| = |N(u)|$.

We define $\tilde{c}(u) = \min\{c(u), \deg(u)\}$. This definition may seem odd, since we may assume that $c(u) \leq \deg(u)$ 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 $\deg(u) < c(u)$. We also define $b(u) = \min\{c(u), \deg(u), L\} = \min\{\tilde{c}(u), L\}$. $b(u)$ can be seen as the covering potential of u in the current graph. A single copy of u can cover $c(u)$ of the demand if $\deg(u) \geq c(u)$, but if $\deg(u) < c(u)$, we cannot cover more than a total of $\deg(u)$ 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 *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 $\ell(A) = |A|$.

Small, medium, and large edges. Given a PCVC instance, we refer to an edge $e = (u, v)$ as *large* if $\ell(e) > c(u), c(v)$. If $\ell(e) \leq c(u), c(v)$ it is called *small*. Otherwise, e is called *medium*.

Consider the case of PCVC with inseparable demands. Since a large edge cannot be assigned to a single copy of one of its endpoints, it follows that we may ignore large edges in the case of PCVC with inseparable demands without losing any feasible solution. 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 $e \in A(v)$ in every 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 similar problem may happen 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 $w(u) = \infty$.

Local ratio. The *local ratio technique* [6, 1, 4] 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 [4]). *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 $(\mathcal{F}, w)$.*

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 [13] and the local ratio approximation algorithm for PVC from [3].

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 $e' \in E(A) \setminus A(u)$. (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. (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 $w(x) < \infty$ and $\ell(e_x) > \max\{L, c(x)\}$ 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 PCVC$(V \setminus \{u\}, E, w, L)$
4: if there exists $u \in V$ such that $w(u) = 0$ then
5:     $A \leftarrow$ PCVC$(V \setminus \{u\}, E \setminus E(u), w, \max\{L - \deg(u), 0\})$
6:     if $V(A) = \emptyset$
7:         while $\ell(A(u)) < L$ do
               $A(u) \leftarrow A(u) \cup \{$Next-Uncovered$(u)\}$
8:         while $(A(u) \neq E(u))$ and $(\ell(A(u)) + \ell($Next-Uncovered$(u)) < c(u))$ do
               $A(u) \leftarrow A(u) \cup \{$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 PCVC$(V, E, w_2, L)$
```

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 α', $\deg'$, $\tilde{c}'$, b', and L'. Similarly, we use α, 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 by induction on the recursion that $\ell(A) \geq L$. Consider the base case. If the recursion ends in Line 1 then $\ell(A) = 0 = L$. Otherwise the recursion ends in Line 2 and $\ell(A) = \ell(e) > L$. 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 $\ell(A) = \ell(A') + \deg(u) \geq L' + \deg(u) = L$. ■

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 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 4.

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 Appendix B 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.

Using 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 for* PCVC *with inseparable 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. ■

4 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 changing 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. $\alpha(v)\tilde{c}(v) \leq 2\deg(v)$. *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 $\deg(v) > c(v)$, and therefore $\alpha(v)\tilde{c}(v) = \alpha(v)c(v) < \ell(A(v)) + c(v) \leq 2\ell(A(v)) \leq 2\deg(v)$. ■

4.1 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 $\$$ 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

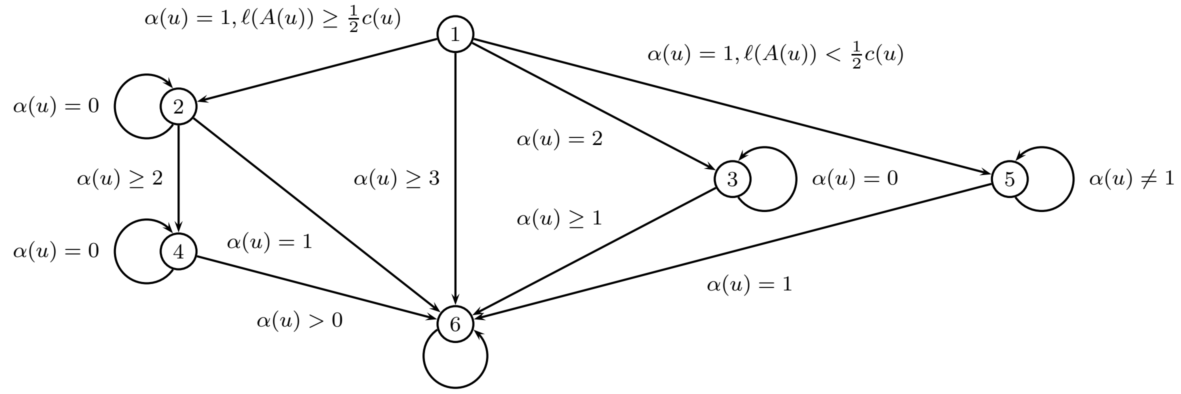

Figure 1: Possible transitions between the conditions.

to distribute $3L$ coins so that $\$(v) \geq \alpha(v)b(v)$. 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.*
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 $\$(v_0) = 3L$ and $\$(v) = 0$ for every $v \neq v_0$ is valid.*
4. *$V(A) = \{v_0, v_1\}$, $\alpha(v_0) = 1$, $\alpha(v_1) \geq 2$, and $\ell(A(v_0)) \geq \frac{1}{2}c(v_0)$. Also, there exists a valid charging scheme $\$$ such that $\$(v_0) \geq L$, $\$(v_0) \geq 2\ell(A(v_0)) - (\ell(A) - L)$, $\$(v_1) \geq \alpha(v_1)c(v_1)$, and $\$(v) = 0$ for every $v \neq v_0, v_1$.*
5. *$\alpha(v_0) = 1$, $\ell(A(v_0)) < \frac{1}{2}c(v_0)$, $\alpha(v) \geq 2$ for every $v \in V(A) \backslash \{v_0\}$, and $L > \deg(v_0) - \ell(A(v_0))$. Also, there exists a valid charging scheme $\$$ such that $\$(v_0) \geq 2L + \ell(A(v_0)) - \ell(A)$, $\$(v) \geq \alpha(v)c(v)$ for every $v \in A(v) \setminus \{v_0\}$, and $\$(v) = 0$ for every $v \notin V(A)$.*
6. *$V(A) \neq \emptyset$. Also, there exists a valid charging scheme $\$$ such that $\$(v) \geq \alpha(v)\tilde{c}(v)$ for every $v \in V$.*

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 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 $\$(u) = 0$.

For the rest of the proof we concentrate on recursive calls that are made in Line 5. We consider a solution A' that was computes by the recursive call, and denote by $\$'$ 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:

($1 \to 5$) If $\alpha(u) = 1$ and $\ell(A(u)) < c(u)/2$, then we claim that $\ell(A(u)) = \deg(u)$. Observe that edges are added to $A(u)$ in a non-increasing order of demands in Lines 7 and 8. Hence, $\ell(A(u)) < c(u)/2$ implies that $A(u) = E(u)$. It follows that $\deg(u) - \ell(A(u)) = 0 < L$. Consider the charging scheme $\$(u) = 3L$ and $\$(v) = 0$ for every $v \neq u$. $\$(u) = 3L \geq 2L + \ell(A(v_0)) - \ell(A)$ since $\ell(A) = \ell(A(v_0))$. Hence, Condition 5 holds.

($1 \to 2$) If $\alpha(u) = 1$ and $\ell(A(u)) \geq c(u)/2$, then Condition 2 holds.

($1 \to 3$) If $\alpha(u) = 2$, then by the construction of $A(u)$ (Line 7), we know that $\ell(A(u)) - \ell(e_u) < L$, where e_u is the last edge that was added to $A(u)$. Hence, Condition 3 holds.

($1 \to 6$) Finally, consider the case $\alpha(u) \geq 3$. Let e_u be the last edge assigned to u. Observe that $\ell(e_u) \leq L$, since Line 2 was not executed, and $\ell(A(u)) < L + \ell(e_u)$ due to Line 7. Hence, $\ell(A(u)) < 2L$. Furthermore, note that $\tilde{c}(u) = c(u)$, since $\deg(u) > c(u)$. Let the charging scheme be $\$(u) = 3L$ and $\$(v) = 0$ for every $v \neq u$. Condition 6 holds, since $\alpha(u)\tilde{c}(u) = \alpha(u)c(u) < \ell(A(u)) + c(u) < \frac{3}{2}\ell(A(u)) < \frac{3}{2}2L = 3L$.

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:

($2 \to 2$) If $\alpha(u) = 0$ then $A = A'$ and therefore A satisfies Condition 2.

($2 \to 6$) If $\alpha(u) = 1$ then we show that Condition 6 holds. Consider the charging scheme $\$(v_0) = \$'(v_0) + 2\deg(u)$, $\$(u) = \deg(u)$, and $\$(v) = \$'(v)$ for every $v \neq v_0, u$. Observe that $\deg(u) \leq c(u)$. Hence, $\$(u) = \deg(u) = \alpha(u)\tilde{c}(u)$. Also, $\ell(A(v_0)) < L$ since $\alpha(u) > 0$. Therefore, $\$(v_0) = 3L' + 2\deg(u) \geq 2L \geq 2\ell(A(v_0)) \geq c(v_0)$. Hence, $\$(v) \geq \alpha(v)\tilde{c}(v)$ for every v and Condition 6 holds.

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

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:

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

($3 \to 6$) 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 $\ell(A(u)) = \deg(u) \geq \tilde{c}(u)$ it follows that $\$(u) = 2\deg(u) \geq \alpha(u)\tilde{c}(u)$. 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 $\ell(e_0) \leq \ell(A(v_0))/2$. This implies that $L' \geq \ell(A(v_0))/2 > c(v_0)/2$ because $\ell(A'(v_0)) - \ell(e_0) < L'$. It follows that $\$(v_0) = 2L' + L \geq c(v_0) + c(v_0) = \alpha(v_0)\tilde{c}(v_0)$.

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:

($4 \to 4$) 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, $\$(v_1) = \$'(v_1) \geq \alpha(v_1)c(v_1)$. Furthermore, $\$(v_0) = \$'(v_0) + \deg(u) \geq L' + \deg(u) = L$. 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 $\to$ 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 $\$(v_0) \geq \alpha(v_0)\tilde{c}(v_0)$. v_1 is funded since $\$'(v_1) \geq \alpha(v_1)c(v_1)$. As for u, $\$(u) = 2\deg(u) \geq \alpha(u)\tilde{c}(u)$.

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 $\to$ 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 $v \in A(v) \setminus \{v_0, u\}$, then $\$(v) = \$'(v) = \alpha'(v)c(v) = \alpha(v)c(v)$.

(5 $\to$ 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, $\$(u) = \deg(u) \geq \alpha(u)\tilde{c}(u)$ 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 $\deg'(v_0) - \ell(A'(v_0)) < L'$ it follows that $\deg(v_0) - \ell(A(v_0)) < L$. Therefore, $\$(v_0) > \deg(v_0) \geq \alpha(v_0)\tilde{c}(v_0)$.

Consider a recursive call in which A' satisfies Condition 6. In this case we only have one option:

(6 $\to$ 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 $\tilde{c}(v) \neq \tilde{c}'(v)$. In this case $\deg'(v) < c(v)$ while $\deg(v) > \deg'(v)$. 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. ■

Lemma 3. *There exists a charging scheme (with respect to G and assignment A) in which every vertex u is given at least $\alpha(v)b(v)$ coins and $\sum_v \alpha(v)b(v) \leq 3L$.*

Proof. The lemma follows from Lemma 2 and the definition of b. ■

4.2 The recursion ends with a medium or large edge

In this section we study what happens when Algorithm **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 *valid* if $\sum_v \$(v) \leq 3\max\{L, \alpha(x)c(x)\}$. Our goal is to 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.

Lemma 4. *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 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 $\sum_v \$(v) \leq 3L$.*

For lack of space the proof of the lemma is given in Appendix C.

Lemma 5. *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. ■

4.3 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 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 to x (if the recursion ends in Line 2), in both case 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 $w_1(\bar{A}) \geq \varepsilon \cdot L$ for every feasible assignment $\bar{A}$. First, 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 $w_1(\bar{A}) = \varepsilon \cdot \sum_v \bar{\alpha}(v)\tilde{c}(v) \geq \varepsilon \cdot \sum_v \ell(\bar{A}(v)) \geq \varepsilon \cdot L$. 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 $w_1(\bar{A}) \geq \varepsilon \cdot \max\{L, \alpha(x)c(x)\}$ for any feasible cover $\bar{A}$ that assigns e_x to x. By the previous argument we know that $w_1(\bar{A}) \geq \varepsilon \cdot L$. In addition, because $e \in \bar{A}(x)$, we have $w_1(\bar{A}) \geq \bar{\alpha}(x)w_1(x) \geq \alpha(x)w_1(x) = \varepsilon \cdot \alpha(x)b(x)$. Since the if condition in Line 2 was not met in this call we have $\ell(e_x) < L$ 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.

Acknowledgments. We thank Einat Or for helpful discussions. We also thank Roee Engelberg for his suggestions.

This research was supported by REMON - Israel 4G Mobile Consortium, sponsored by Magnet Program of the Chief Scientist Office in the Ministry of Industry and Trade.

References

[1] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[2] J. Bar-Ilan, G. Kortsarz, and D. Peleg. Generalized submodular cover problems and applications. *Theoretical Computer Science*, 250:179–200, 2001.

[3] Bar-Yehuda. Using homogeneous weights for approximating the partial cover problem. *Journal of Algorithms*, 39:137–144, 2001.

[4] R. Bar-Yehuda. One for the price of two: A unified approach for approximating covering problems. *Algorithmica*, 27(2):131–144, 2000.

[5] R. Bar-Yehuda, K. Bendel, A. Freund, and D. Rawitz. Local ratio: a unified framework for approximation algorithms. *ACM Computing Surveys*, 36(4):422–463, 2004.

[6] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.

[7] N. H. Bshouty and L. Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *15th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1373 of *LNCS*, pages 298–308. Springer, 1998.

[8] J. Chuzhoy and J. Naor. Covering problems with hard capacities. In *43nd IEEE Symposium on Foundations of Computer Science*, pages 481–489, 2002.

[9] I. Dinur and S. Safra. The importance of being biased. In *34th ACM Symposium on the Theory of Computing*, pages 33–42, 2002.

[10] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, and A. Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. In *30th Annual International Colloquium on Automata, Languages and Programming*, volume 2719 of *LNCS*, pages 164–175, 2003.

[11] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding in bipartite graphs. In *43nd IEEE Symposium on Foundations of Computer Science*, pages 323–332, 2002.

[12] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1):55–84, 2004.

[13] S. Guha, R. Hassin, S. Khuller, and E. Or. Capacitated vertex covering. *Journal of Algorithms*, 48(1):257–270, 2003.

[14] J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of generalized vertex cover problems. In *9th International Workshop on Algorithms and Data Structures*, volume 3608 of *LNCS*, pages 36–48, 2005.

[15] E. Halperin and A. Srinivasan. Improved approximation algorithms for the partial vertex cover problem. In *5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, volume 2462 of *LNCS*, pages 161–174, 2002.

[16] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problem.* PWS Publishing Company, 1997.

[17] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[18] M. Kearns. *The Computational Complexity of Machine Learning.* M.I.T. Press, 1990.

[19] J. W. Lustbader, D. Puett, and R. W. Ruddon, editors. *Symposium on Glycoprotein Hormones: Structure, Function, and Clinical Implications*, 1993.

[20] J. Mestre. A primal-dual approximation algorithm for partial vertex cover: Making educated guesses. In *8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, volume 3624 of *LNCS*, pages 182–191, 2005.

[21] P. Slavík. Improved performance of the greedy algorithm for partial cover. *Information Processing Letters*, 64(5):251–254, 1997.

[22] A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 588–597, 2001.

[23] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982.

APPENDIX

A Separable Demands vs. Inseparable Demands

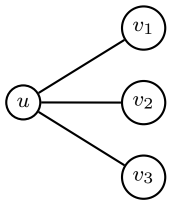

Figure 2: A CVC instance: $c(u) = 3$ and $\ell(u, v_i) = 2$ for $i \in \{1, 2, 3\}$. In the case of separable demands two copies of u are sufficient to cover the three edges, but in the case of inseparable demands three copies of u are needed to cover the edges.

B 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.) 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. When 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 $w(u_1) = b(u_1)$ 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 $L + \delta$.

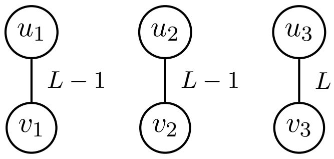

Figure 3: A PCVC instance with medium edges.

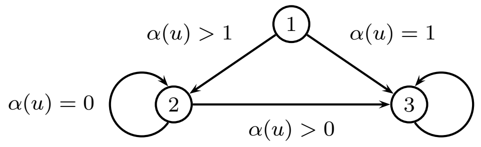

Figure 4: Possible transitions between the conditions.

C Proof of Lemma 4

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 4. 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 $\$'$ the charging scheme that corresponds to A'.

Consider a recursive call in which A' satisfies Condition 1. In this case, $V'(A) = \{x\}$. Notice that $\alpha(u) = 0$, is not possible, since in this case $\ell(A') \leq L$, and this means that Line 2 would have been executed earlier. Hence, there are two possible options:

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

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

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

(2 $\rightarrow$ 2) If $\alpha(u) = 0$, then $A' = A$ and Condition 2 holds.

(2 $\rightarrow$ 3) If $\alpha(u) > 0$ then consider the charging scheme $\$(u) = 2\ell(A(u))$ and $\$(v) = \$'(v)$ for all $v \neq u$. Notice that $\ell(A(v_1)) + \ell(e) < L$ (Line 10 in the v_1's recursive call) and $\ell(A(u)) + \ell(e) < L$ (Line 2 in u's recursive call) and $\ell(A(u)) + \ell(A(v_1)) < L$ (Line 1 in x's recursive call). Furthermore, $\$(x) \leq 2\ell(e)$, $\$(v_1) = 2\ell(A(v_1))$ and $\$(u) = 2\ell(A(u))$. Adding up these inequalities we get $\$(x) + \$(v_1) + \$(u) \leq 3L$, thus Condition 3 holds.

Consider a recursive call in which A' satisfies Condition 3. In this case we only have one option: to come back to Condition 3. The proof is identical to case (6 $\rightarrow$ 6) in Lemma 3.

The lemma follows.

D Partial Capacitated Covering with Inseparable Demands

In this section 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 fund 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 $\beta(v)c(v) \leq 2\ell(A(v))$.*

Proof. Since β 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. If all copies of v are more than half full, then $\beta(v) \leq \left\lfloor \frac{\ell(A(v))}{c(v)/2} \right\rfloor \leq \frac{2\ell(A(v))}{c(v)}$. Otherwise, the length of edges assigned to the most vacant copy of v plus the length of edges assigned to another copy of v is more than $c(v)$. Hence, $\beta(v) \leq 2 + \left\lfloor \frac{\ell(A(v))-c(v)}{c(v)/2} \right\rfloor \leq \frac{2\ell(A(v))}{c(v)}$. ■

Lemma 6. *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. Thus, $\sum_v \beta(v)b(v) \leq 3L$.*

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.*

E Partial Capacitated Vertex Cover with Unit Demands

Algorithm **PCVC2** is a recursive local ratio algorithm that computes 2-approximation solutions for PCVC with unit demands. As we did in the previous sections we define $b(u) = \min\{\deg(u), c(u), L\}$, and $\tilde{c}(u) = \min\{\deg(u), c(u)\}$. Given an assignment A, a vertex v is called *vulnerable* if $0 < |A(v)| < \tilde{c}(v)$ 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 third 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.

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

Algorithm 2 : 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 PCVC2$(V \setminus \{u\}, E, w, L)$
3: if there exists $u \in V$ such that $w(u) = 0$ then
4:     $A \leftarrow$ 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
               $A(v_0) \leftarrow A(v_0) \cup \{e\}$, where $e \in E(v_0)$ is uncovered
7:         elseif there exists a vulnerable vertex $v \in N(u)$ then
               $A(v) \leftarrow A(v) \cup \{(u, v)\}$
8:         else
               $A(u) \leftarrow A(u) \cup \{e\}$, where $e \in E(u)$ is uncovered
9:     return $A$
10: Let $\varepsilon = \min_{u \in V} \{w(u)/b(u)\}$
11: Define the weight functions $w_1(v) = \varepsilon \cdot b(v)$, for every $v \in V$, and $w_2 = w - w_1$
12: return PCVC2$(V, E, w_2, L)$
```

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, $|A| = L$ after the while loop. ■

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 $\tilde{c}(v) \le \tilde{c}'(v) + 1$. Thus $\alpha(v)\tilde{c}(v) \le \alpha'(v)\tilde{c}'(v) + 1$.*

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

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 was distributed is exactly $2L$.

Lemma 8. *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. Thus, $\sum_v \alpha(v)b(v) \leq 2L$.*

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 $\alpha(u)b(u)$ 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 it 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 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, $b'(v) = \tilde{c}'(v)$, for every $v \in V(A')$. Hence, the inductive hypothesis implies that $\$(v) \geq \alpha'(v)b'(v) = \alpha'(v)\tilde{c}'(v)$. 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 seen 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 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.

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 too, and by the Local Ratio Theorem it is 2-approximate with respect to w as well. ■