A Primal-Dual Approximation Algorithm for Partial Vertex Cover: Making Educated Guesses [*]

Julián Mestre
Department of Computer Science
University of Maryland, College Park, MD 20742
jmestre@cs.umd.edu

Abstract

We study the PARTIAL VERTEX COVER problem. Given a graph $G = (V, E)$, a weight function $w : V \to R^+$, and an integer s, our goal is to cover all but s edges, by picking a set of vertices with minimum weight. The problem is clearly NP-hard as it generalizes the well-known VERTEX COVER problem. We provide a primal-dual 2-approximation algorithm which runs in $O(n \log n + m)$ time. This represents an improvement in running time from the previously known fastest algorithm.

Our technique can also be used to get a 2-approximation for a more general version of the problem. In the PARTIAL CAPACITATED VERTEX COVER problem each vertex u comes with a capacity k_u. A solution consists of a function $x : V \to \mathbb{N}_0$ and an orientation of all but s edges, such that the number of edges oriented toward vertex u is at most $x_u k_u$. Our objective is to find a cover that minimizes $\sum_{v \in V} x_v w_v$. This is the first 2-approximation for the problem and also runs in $O(n \log n + m)$ time.

[*]Research supported by NSF Awards CCR 0113192 and CCF 0430650, and the University of Maryland Dean's Dissertation Fellowship

1 Introduction

The VERTEX COVER problem has been widely studied [14, 18]. In the simplest version, we are given a graph $G = (V, E)$ and are required to pick a minimum size subset of vertices to cover all the edges of the graph. An edge is said to be covered if at least one of its incident vertices is picked. The problem is NP-hard and was one of the first problems shown to be NP-hard in Karp's seminal paper [17]. However, several different approximation algorithms have been developed for it [14]. These algorithms develop solutions within twice the optimal in polynomial time.

Recently several generalizations of the vertex cover problem have been considered. Bshouty and Burroughs [4] were the first to study the PARTIAL VERTEX COVER problem. In this generalization we are not required to cover all edges of the graph, any s edges may be left uncovered. By allowing some number of edges to be left uncovered we hope to obtain a cover with substantially lower cost. It can be regarded as a trade off between cost and coverage.

The paper by Bshouty and Burroughs [4] was the first to give a factor 2 approximation for partial vertex cover using LP-rounding. Subsequently, combinatorial algorithms with the same approximation guarantee were proposed. Let the input be a graph on n vertices and m edges. Hochbaum [13] presented an $O(nm \log \frac{n^2}{m} \log n)$ time algorithm based on Lagrangian Relaxation; Bar-Yehuda [1] developed an $O(n^2)$ time algorithm using the local-ratio technique; finally, a primal-dual algorithm which runs in $O(n(n \log n + m))$ time was given by Gandhi et al. [7]. Our approach builds on the work of Gandhi et al. [7].

The main result of this paper is a primal-dual 2-approximation algorithm for partial vertex cover which runs in $O(n \log n + m)$ time. This represents an improvement in running time over the previously known algorithms for the problem. One additional benefit of our method is that it can be used to solve a more general version of the problem.

Motivated by a problem in computational biology, the CAPACITATED VERTEX COVER problem was proposed by Guha et al. [9]. For each vertex $v \in V$ we are given a capacity k_v and a weight w_v. A *capacitated vertex cover* is a function $x : V \to \mathbb{N}_0$ such that there exists an orientation of all edges in which the number of edges directed into $v \in V$ is at most $k_v x_v$. When e is oriented toward v we say e is *covered by* or *assigned to* v. The weight of the cover is $\sum_{v \in V} w_v x_v$. We want to find a cover with minimum weight. In their paper, Guha et al. [9] give a primal-dual 2-approximation for the problem which runs in $O(m + n \log n)$ time. An LP rounding algorithm achieving the same approximation factor was later developed by Gandhi et al. [8].

Allowing any s edges to be left unassigned we get a new problem: the PARTIAL CAPACITATED VERTEX COVER problem. Thus we bring together two generalization of the original vertex cover problem that have been studied separately. Using the primal-dual algorithm of [9] for capacitated vertex cover coupled with our method we show a 2-approximation for the partial version.

For vertex cover and partial vertex cover, the best known approximation factor for general graphs is $2 - o(1)$ [3, 10, 11, 16]. However, the best constant factor remains 2. Some of the algorithms [6, 2, 12] that achieve a ratio of 2 rank among the first approximation algorithms and can be interpreted using the primal-dual method where a maximal dual solution is found and the cost of the cover is charged to the dual variables. It is interesting to see how far the same primal-dual method can be taken to tackle a more general version of the original problem.

Finally we mention that the techniques used in this paper are not restricted to vertex-cover-like problems. For instance, the primal-dual algorithm of Jain and Vazirani [15] for the FACILITY LOCATION problem can be used in conjunction with our method to design a faster approximation algorithm for the partial version of the problem, which was studied by Charikar et al. [5].

2 LP Formulation

Let us describe an integer linear program formulation for the PARTIAL VERTEX COVER problem. Each vertex $v \in V$ has associated a variable $x_v \in \{0,1\}$ which indicates if v is picked for the cover. For every edge $e \in E$ one of its endpoints is picked or the edge is left uncovered. A variable $p_e \in \{0,1\}$ indicates the latter, i.e., $p_e = 1$ means e is left uncovered. The number of uncovered edges cannot exceed s. The LP relaxation of the IP just described is given below.

$$\begin{array}{rl} \min\ \sum_{v\in V} w_v x_v & \\ p_e + x_u + x_v \geq 1 & \forall\, e = \{u,v\} \in E \\ \sum_{e\in E} p_e \leq s & \\ x_v, p_e \geq 0 & \forall\, e \in E, v \in V \end{array} \qquad (\text{LP}_{\text{PVC}})$$

As it stands, LP_{PVC} exhibits an unbounded integrality gap. Consider the following example: a star where the center node has degree d, all leaves have unit weight while the center node has weight 10. Suppose $s = d - 2$, that is, we must cover two edges. The optimal integral solution picks two leaves, thus having a total cost of 2. On the other hand, a fractional solution can pick $2/d$ of the center node. Every edge is covered by this amount which adds up to 2. The cost of the fractional solution is $20/d$. Choose d big enough to make the cost as small as desired.

Fortunately, the integrality gap can be narrowed provided we know what is the most expensive vertex in the optimal solution. Let OPT be the optimal solution; we will also use OPT to denote the cost of the optimal solution. Suppose we knew that $h \in V$ is the most expensive vertex in OPT. We modify the LP formulation as follows: our solution must pick h and is not allowed to use vertices more expensive than h, the latter can be achieved by setting their weight to ∞. While this change does not increase the cost of the integral optimal solution, it does narrow the integrality gap. For example, in the above star example setting the weight of the center to ∞ effectively closes the integrality gap. Gandhi et al. [7] showed how to use the above modification to develop a primal-dual 2-approximation algorithm. Since we do not know the most expensive vertex in OPT, we must guess. Their procedure, which takes $O(n \log n + m)$ time, is run on every choice of $h \in V$, and the cheapest cover produced is returned. Therefore, their algorithm runs in $O(n(n \log n + m))$ time.

Our approach is along these lines, but instead of doing exhaustive guessing we run our algorithm just once and make different guesses along the way. One may say that the algorithm *makes educated guesses*. This allows us to bring down the running time.

Theorem 1. *There is a 2-approximation algorithm for the* PARTIAL VERTEX COVER *problem which runs in* $O(n \log n + m)$ *time.*

In the next section we describe our algorithm and prove the above theorem.

3 Primal-Dual algorithm

The dual program for $\mathrm{LP}_{\mathrm{PVC}}$ is given below. By $\delta(v)$ we refer to the edges incident to vertex v.

$$\begin{aligned} &\max\ \textstyle\sum_{e\in E} y_e - sz \\ &\textstyle\sum_{e\in\delta(v)} y_e \le w_v && \forall\, v\in V \\ &y_e \le z && \forall\, e \in E \\ &y_e, z \ge 0 && \forall\, e\in E \end{aligned} \tag{$\mathrm{DL}_{\mathrm{PVC}}$}$$

Initially all dual variables are set to 0, and all edges are unassigned. The algorithm works in iterations, each consisting of a *pruning step* followed by a *dual update step.* As the algorithm progresses we build a set $C \subseteq V$, which is initially empty. Along the way we may disallow some vertices; we keep track of these with the set R, which also starts off empty.

In the pruning step we check, for every vertex $v \in V \setminus \{C \cup R\}$, if $C + v$ is a feasible cover. If that is the case, we guess $C + v$ as a candidate cover, and *disallow* vertex v: set $w_v \leftarrow \infty$ and add v to R. Notice that multiple vertices may be disallowed in a single pruning step.

Let $E(R)$ be the edges with both endpoints in R. If $|E(R)| > s$ we stop as there is no hope of finding a cover anymore since $V \setminus R$ is not feasible. At this point we return the candidate cover with minimum weight.

In the dual update step we uniformly raise z and the y_e variables of unassigned edges until some vertex u becomes tight, i.e., its dual constraint is met with equality. Notice that once disallowed, a vertex can never become tight since $w_v = \infty$ for all $v \in R$. Hence, $u \in V \setminus R$. If multiple vertices become tight at the same time then arbitrarily pick one to process. Vertex u is added to C, unassigned edges in $\delta(u)$ are assigned to u and their dual variables are frozen. After processing u we proceed to the next iteration. In each dual update step only one vertex is processed, even if multiple tight vertices are available. This ensures that adding u to C does not make C feasible, otherwise u would have been disallowed in the previous pruning step.

The algorithm terminates when $|E(R)| > s$, at which point we know OPT must use at least one vertex in R. Let h be the first vertex in OPT to be disallowed, and C the set at the moment h was pruned. By definition $C + h$ is a feasible solution; suppose its cost, $w(C + h)$, is at most twice OPT. Since the algorithm returns a candidate cover with minimum weight, we achieve an approximation ratio of 2. It all boils down to proving the following lemma.

Lemma 1. *Let h be the first vertex in* OPT *to be disallowed. Also, let R and C be the sets constructed by the algorithm right before h was disallowed. Then $w(C + h) \le 2\,\mathrm{OPT}$.*

Proof. We start off by strengthening the LP formulation. Let $w'_u = w_u$ for $u \notin R$ and $w'_u = +\infty$ for $u \in R$. Since $h \in \mathrm{OPT}$ and $R \cap \mathrm{OPT} = \emptyset$, the following is a valid LP relaxation for our problem.

$$\begin{aligned} &\min\ \textstyle\sum_{v\in V} w'_v x_v \\ &p_e + x_u + x_v \ge 1 && \forall\, e=\{u,v\}\in E \\ &\textstyle\sum_{e\in E} p_e \le s \\ &x_h \ge 1 \\ &x_u, p_e \ge 0 && \forall\, e\in E, v \in V \end{aligned} \tag{$\mathrm{LP}'_{\mathrm{PVC}}$}$$

The dual of $\mathrm{LP}'_{\mathrm{PVC}}$ is given below.

$$\begin{array}{rll} \max & \sum_{e\in E} y_e - sz + \gamma & \\ & \sum_{e\in\delta(v)} y_e \le w'_v & \forall\, v \in V - h \\ & \sum_{e\in\delta(h)} y_e + \gamma \le w'_h & \\ & y_e \le z & \forall\, e \in E \\ & y_e, z, \gamma \ge 0 & \forall\, e \in E \end{array} \qquad (\mathrm{DL}'_{\mathrm{PVC}})$$

Let (y, z) be the value of the dual variables just before h was disallowed. Let $\gamma = w'_h - \sum_{e\in\delta(h)} y_e$. The solution (y, z, γ) is feasible for $\mathrm{DL}'_{\mathrm{PVC}}$. Thus, by weak duality, $\mathrm{OPT} \le \sum_{e\in E} y_e - sz + \gamma$.

Let r be the vertex added to C just before h was disallowed. (If there is no such vertex, i.e., $C = \emptyset$, then the cover $\{h\}$ is optimal.) Let f be the number of unassigned edges just before r became tight. Among these f edges, suppose t_r were incident on r and t_h were incident on h.

Every vertex $v \in C$ is tight ($w_v = \sum_{e\in\delta v} y_e$), thus

$$\begin{aligned} w(C+h) &= \left(\sum_{v\in C} \sum_{e\in\delta(v)} y_e \right) + \sum_{e\in\delta(h)} y_e + \gamma, \\ &\le 2 \left(\sum_{e\in E} y_e - fz \right) + (t_r + t_h) z + \gamma. \end{aligned}$$

The second line follows because edges assigned before r became tight are considered at most twice in the first line. The remaining edges have $y_e = z$ and there are $t_r + t_h$ of them.

Now consider the pruning step right before r became tight. Vertex r was not disallowed then because $f - t_r > s$. Similarly $f - t_h > s$. Thus $t_r + t_h \le 2\,(f - s)$. It follows that

$$w(C+h) \le 2 \left(\sum_{e\in E} y_e - sz \right) + \gamma \le 2\,\mathrm{OPT}.$$

□

For the implementation we keep two priority queues for the vertices. We refer to the value of z as the current time in the execution of the algorithm. The first queue uses the key $k_1(u)$, which denotes the time at which vertex u will become tight; initially $k_1(u) = \frac{w_u}{|\delta(u)|}$. The second queue uses the key $k_2(u)$, which denotes the number of unassigned edges incident to u; initially $k_2(u) = |\delta(u)|$.

Let g be the number of unassigned edges; initially $g = m$. In the pruning step we fetch a vertex v with maximum k_2 value. If $g - k_2(v) > s$ proceed to the dual-update step. Otherwise, disallow v by removing v from both queues and adding v to R. Repeat until $|E(R)| > s$, or $g - k_2(v) > s$.

In the dual update step we fetch a vertex v with minimum k_1 value. Set $g \leftarrow g - k_2(v)$, add v to C and remove v from both queues. For every neighbor $u \notin C \cup R$, if $k_1(u) = 1$ remove u from both queues. Otherwise increase $k_1(u)$ to $z + (k_1(u) - z)\frac{k_2(u)}{k_2(u)-1}$ and decrease $k_2(u)$ to $k_2(u) - 1$.

In order to deal with the guessing we store C as a sorted list, adding vertices at the end. Also we keep two variables h and b. When a vertex v is disallowed we check if the candidate cover $C + v$ is cheaper than the current best candidate cover. If that is the case, set $h \leftarrow v$ and $b \leftarrow |C|$. At the end of the algorithm the cover returned is made up of h and the first b elements of C.

The running time is dominated by the priority queue operations. At most $2n$ remove and $2m$ modify key operations are performed, which using Fibonacci heaps takes $O(n \log n + m)$ time. This finishes the proof of Theorem 1.

4 Generalization

Our technique can be used to solve a more general version of the vertex cover problem. In the PARTIAL CAPACITATED VERTEX COVER problem every vertex $u \in V$ has a capacity k_u and a weight w_u. A single copy of u can cover at most k_u edges incident to it, the number of copies picked is given by $x_u \in \mathbb{N}_0$. Every edge $e = \{u, v\} \in E$ is either left uncovered ($p_e = 1$) or is assigned to one of its endpoints, variables y_{eu} and y_{ev} indicate the latter. The number of edges assigned to a vertex u cannot exceed $k_u x_u$, while the ones left unassigned cannot be more than s. The LP relaxation of the program just described is given below. The additional constraint $x_v \geq y_{ev}$ is needed to fix the integrality gap (c.f. [9]).

$$\begin{array}{rll} & \min \sum_{v \in V} w_v x_v & \\ p_e + y_{eu} + y_{ev} \geq 1 & \forall\, e = \{u, v\} \in E & \\ \sum_{e \in \delta(v)} y_{ev} \leq k_v x_v & \forall\, v \in V & \qquad (\text{LP}_{\text{PCVC}}) \\ x_v \geq y_{ev} & \forall\, v \in e \in E & \\ \sum_{e \in E} p_e \leq s & & \\ p_e, y_{ev}, x_v \geq 0 & \forall\, v \in e \in E & \end{array}$$

For the sake of completeness we describe how to, given an integral feasible cover x, set the y and p variables. This an assignment problem that can be solved using max flow. Consider a network with two layers, a source, and a sink. In the first layer there is a node for every $e \in E$; in the second layer there is a node for every $v \in V$ plus a dummy node that represents the option of leaving an edge uncovered. Every e is connected with unit-capacity edges to the source, its two endpoints and the dummy node. Every node v is connected with an edge with capacity $k_v x_v$ to the sink. Finally, the dummy node is connected to the sink using an edge with capacity s. An integral flow shipping m units of flow from the source to the sink corresponds to a valid edge assignment for x and vice versa.

Our algorithm, however, does not use this approach, rather, $m - s$ edges are assigned to one of their endpoints; then as many copies of each vertex as necessary are chosen to cover the edges assigned to it. The algorithm works in iterations, each having a pruning step followed by a dual update step. For the latter we follow the algorithm in [9] which is described here for completeness. The dual program of LP_{PCVC} is given below.

$$\begin{array}{rll} & \max \sum_{e \in E} \alpha_e - sz & \\ k_v q_v + \sum_{e \in \delta(v)} l_{ev} \leq w_v & \forall\, v \in V & \\ \alpha_e \leq q_v + l_{ev} & \forall\, v \in e \in E & \qquad (\text{DL}_{\text{PCVC}}) \\ \alpha_e \leq z & \forall\, e \in E & \\ q_v, l_{ev}, \alpha_e, z \geq 0 & \forall\, v \in e \in E & \end{array}$$

Initially all variables are set to 0 and all edges are unassigned. At the beginning, vertex u is said to be high degree if more than k_u unassigned edges are incident to u, otherwise u is low degree. As we will see a vertex u may start off as high degree and later become low degree as adjacent edges get assigned to its neighbors. For such u we define L_u to be the set of unassigned edges incident to u just after u becomes low degree. If u is low degree from the beginning, i.e., $\deg(u) \leq k_u$, we define $L_u = \delta(u)$.

In the pruning step when processing a vertex $v \in V \setminus \{C \cup R\}$, although in principle we are allowed to pick multiple copies of it, we check if adding just one copy of v to the current solution makes it feasible. If that is the case, guess the current solution plus v and then disallow v: add it to R and set $w_v \leftarrow \infty$.

In the dual update step we uniformly raise z and α_e for all unassigned edges. Because of the constraint $\alpha_e \leq q_v + l_{ev}$ one of the terms on the right hand side must be raised by the same amount. Which variable is increased depends on the nature of v: If v is high degree we raise q_v, otherwise we raise l_{ev}. When some vertex u becomes tight we add it to C and *open* it: If u is high degree we assign to it all the unassigned edges in $\delta(u)$, otherwise u is low degree and all edges in L_u are assigned to it. Notice that in the later case some of the edges in L_u might have already been assigned. This means that some of the edges originally assigned to a high degree vertex when it was opened may later be taken away by the opening of low degree neighbors. Therefore the number of copies of a vertex that is needed can decrease over time after it is opened. In what follows, when we talk about the current solution we mean the current assignment of edges.

Suppose that in the dual update step the algorithm opens vertex u. If u is low degree, its opening cannot make the solution feasible, otherwise it would have been disallowed in the previous pruning step. On the other hand, if u is high degree, opening u may make the solution feasible, since we can use multiple copies of u. In the latter case we say the algorithm *ends prematurely*: we assign to u just enough edges to make the solution feasible (i.e., exactly s edges are left unassigned) and we return the best solution among the candidate covers and the current solution. If on the other hand this does not happen and R grows to the point where $|E(R)| > s$ we stop and return the best candidate cover. The pseudo-code for the algorithm is given in Appendix A.

Let us first consider the case where the algorithm ends prematurely, we will show that we can construct a solution with cost at most $2\left(\sum_{e\in E}\alpha_e - sz\right)$. Let φ be the edge assignment when the algorithm ended prematurely, where $\varphi(v)$ denotes the set of edges assigned to vertex v. Constructing the solution is simple: we pick enough copies of every vertex to cover the edges assigned to it, that is, for every $u \in V$ we chose $\left\lceil \frac{|\varphi(u)|}{k_u} \right\rceil$ copies of u.

Let u be one of the vertices opened. Furthermore suppose u was low degree when it was opened, this means only one copy is needed. If u started off as low degree then $w_u = \sum_{e\in L_u} l_{eu} = \sum_{e\in L_u}\alpha_e$ as $q_u = 0$. On the other hand if u became low degree afterwards then $|L_u| = k_u$ and only edges in L_u have nonzero l_{eu}. Thus,

$$w_u = q_u k_u + \sum_{e\in L_u} l_{eu} = \sum_{e\in L_u}(q_u + l_{eu}) = \sum_{e\in L_u}\alpha_e.$$

In both cases we can pay for u by charging once the edges in L_u.

Now let us consider the case when u is high degree. By definition more than k_u edges were assigned to u when it became tight. Note that some of these edges may later be taken away by low degree neighbors. There are two cases to consider. If $|\varphi(u)| < k_u$ we only need one copy of

u, charging once the edges originally assigned to u (which are more than k_u and have $\alpha_e = q_u$) is enough to pay for $w_u = k_u q_u$. Otherwise $\left\lceil \frac{|\varphi(u)|}{k_u} \right\rceil$ copies of u are needed. Unfortunately our budget of $\sum_{e\in\varphi(u)} \alpha_e = |\varphi(u)|q_u = \frac{|\varphi(u)|}{k_u} w_u$ is only enough to pay for the first $\left\lfloor \frac{|\varphi(u)|}{k_u} \right\rfloor$ copies. The key observation is that edges in $\varphi(u)$ will not be charged from the other side, therefore charging any k_u edges in $\varphi(u)$ one more time is enough to pay for the extra copy needed.

How many times can a single edge be charged? At most twice, either from a single (high degree) endpoint or once from each endpoint. We are leaving s edges uncharged with $\alpha_e = z$, therefore the solution can be paid with $2(\sum_{e\in E} \alpha_e - sz)$.

Before ending prematurely the algorithm may have disallowed some vertices. If none of them is used by OPT, strengthening the LP by setting their weight to ∞ does not increase the cost of the integral optimal solution. Therefore the cost of the dual solution is a lower bound on OPT. It follows from the above analysis that the cover found has cost at most 2 OPT. Otherwise OPT uses at least one vertex in R. Notice that if the algorithm terminates because $|E(R)| > s$ we also get that $R \cap \text{OPT} \neq \emptyset$. In either case, let $h \in R \cap \text{OPT}$ be such that vertices disallowed by the algorithm before h do not belong to OPT.

Let R and C be the sets constructed by the algorithm just before h was disallowed. We proceed as before by strengthening the LP relaxation. Let $w'_u = w_u$ for $u \notin R$ and $w'_u = +\infty$ for $u \in R$. Since $h \in \text{OPT}$ and $R \cap \text{OPT} = \emptyset$, the following is a valid LP relaxation for our problem.

$$\begin{array}{rll} & \min \sum_{v\in V} w'_v x_v & \\ p_e + y_{eu} + y_{ev} \geq 1 & & \forall\, e = \{u,v\} \in E \\ \sum_{e\in\delta(v)} y_{ev} \leq k_v x_v & & \forall\, v \in V \\ x_v \geq y_{ev} & & \forall\, v \in e \in E \\ \sum_{e\in E} p_e \leq s & & \\ x_h \geq 1 & & \\ p_e, y_{ev}, x_v \geq 0 & & \forall\, v \in e \in E \end{array} \qquad (\text{LP}'_{\text{PCVC}})$$

The dual of LP'_{PCVC} is given below.

$$\begin{array}{rll} & \max \sum_{e\in E} \alpha_e - sz + \gamma & \\ k_v q_v + \sum_{e\in\delta(v)} l_{ev} \leq w'_v & & \forall\, v \in V - h \\ k_h q_h + \sum_{e\in\delta(h)} l_{eh} + \gamma \leq w'_h & & \\ \alpha_e \leq q_v + l_{ev} & & \forall\, v \in e \in E \\ \alpha_e \leq z & & \forall\, e \in E \\ q_v, l_{ev}, \alpha_e, z \geq 0 & & \forall\, v \in e \in E \end{array} \qquad (\text{DL}'_{\text{PCVC}})$$

Let (α, q, l, z) be the dual solution constructed by the algorithm just before h was disallowed. Let $\gamma = w'_h - \left(k_h q_h + \sum_{e\in\delta(h)} l_{eh}\right)$. The solution $(\alpha, q, l, z, \gamma)$ is a feasible solution for DL'_{PCVC}. Hence, its cost offers a lower bound on OPT.

Now consider the edge assignment when h was pruned. Before constructing the cover we need to modify the assignment. If h is low degree, we assign to h the edges in L_h, otherwise we assign to it any k_h unassigned edges in $\delta(h)$. This may cause strictly less than s edges to be left unassigned. To fix this we take away from r, the last vertex to become tight, enough edges as to leave exactly s edges unassigned. (If there is no such vertex, i.e., $C = \emptyset$, then the cover $\{h\}$ is optimal.) This can always be done because h was not disallowed before r was opened. Let φ be the edge assignment after this modification. Now we build the cover by picking as many copies of each vertex as necessary to cover the edges assigned to it.

Suppose that just before r became tight there were f unassigned edges. Among these f edges, suppose t_r were incident on r and t_h were incident on h. There are two cases to consider depending on the number of copies of r needed.

Suppose a single copy of r is needed. This can happen because r was low degree, or because r was high degree but later some edges were taken away from r in order to leave exactly s edges uncovered. In either case, using the charging scheme described above, we get:

$$\begin{aligned}\sum_{u\in V}\left\lceil\frac{|\varphi(u)|}{k_u}\right\rceil w_u &\le 2\left(\sum_{e\in E} y_e - fz\right) + \Big(t_r + \min\{t_h, k_h\}\Big)z + \gamma,\\ &\le 2\left(\sum_{e\in E} y_e - sz\right) + \gamma.\end{aligned}$$

The second inequality follows because $f - t_r > s$ and $f - \min\{t_h, k_h\} > s$, which in turn hold because the algorithm did not end prematurely when r was opened and h was not disallowed in the pruning step before r became tight.

Now suppose r was high degree and multiple copies of r were chosen. Then

$$\begin{aligned}\sum_{u\in V}\left\lceil\frac{|\varphi(u)|}{k_u}\right\rceil w_u &\le 2\left(\sum_{e\in E} y_e - fz\right) + \Big(2|\varphi(r)| + \min\{t_h, k_h\}\Big)z + \gamma,\\ &\le 2\left(\sum_{e\in E} y_e - sz\right) + \gamma.\end{aligned}$$

The second inequality follows because after the reassignment $f - s = |\varphi(r)| + \min\{t_h, k_h\}$.

In both cases we can pay for the cover with twice the cost of the dual solution, thus, there is a candidate cover that is 2 approximate. Since we chose a candidate cover with minimum weight the algorithm is a 2-approximation.

The implementation is similar to that of the algorithm for partial vertex cover described in Section 3. We keep two priority queues for the vertices. The first queue is used during the dual update to fetch the next vertex to become tight. The second priority queue is used during the pruning step to fetch a vertex with most unassigned edges incident on.

When a certain vertex v is disallowed we check if the cost of the candidate cover induced by v and the current edge assignment is cheaper than the current best candidate cover, if that is the case we remember v. In order to do this efficiently we need to keep track of the cost of the current edge assignment. Note that when a vertex u is opened the cost increases because of the additional copies of u that are needed. However, if u is low degree it is possible that the number of copies of a high degree neighbor in L_u may decrease. This is easy to check by scanning the edges in L_u.

Suppose h was the vertex that induced the best candidate cover when it was disallowed. To recover the actual solution the algorithm is run a second time. When the algorithm tries to disallow h we stop and return the cover built at this point.

Theorem 2. *There is a 2-approximation algorithm for the* PARTIAL CAPACITATED VERTEX COVER *problem which runs in $O(n \log n + m)$ time.*

5 Conclusion

We have developed algorithms for the partial version of vertex cover and capacitated vertex cover. Our pruning/dual update technique is quite general and may be applied to other covering problems where a primal-dual already exist for the full version of the problem, i.e., $s = 0$.

Acknowledgment: Thanks to Samir Khuller for suggesting developing an algorithm for partial vertex cover that does not use exhaustive guessing and for many useful comments on earlier versions of the paper. Also thanks to the anonymous referees whose comments helped improve the exposition of the paper.

References

[1] R. Bar-Yehuda. Using homogeneous weights for approximating the partial cover problem. *Journal of Algorithms*, 39(2):137–144, 2001.

[2] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for approximating the weighted vertex cover. *Journal of Algorithms*, 2:198–203, 1981.

[3] 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.

[4] N. Bshouty and L. Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *Proceedings of the 15th Annual Symposium on the Theoretical Aspects of Computer Science (STACS'98)*, pages 298–308, 1998.

[5] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 642–651, 2001.

[6] K. L. Clarkson. A modification of the greedy algorithm for vertex cover. *Information Processing Letters*, 16(1):23–25, 1983.

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

[8] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM*, 53(3):324–360, 2006.

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

[10] E. Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, 2002.

[11] E. Halperin and A. Srinivasan. Improved approximation algorithms for the partial vertex cover problem. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'02)*, pages 161–174, 2002.

[12] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

[13] D. S. Hochbaum. The t-vertex cover problem: Extending the half integrality framework with budget constraints. In *Proceedings of the 1st International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'98)*, pages 111–122, 1998.

[14] D. S. Hochbaum, editor. *Approximation Algorithms for NP–hard Problems.* PWS Publishing Company, 1997.

[15] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48 (2):274–296, 2001.

[16] G. Karakostas. A better approximation ratio for the vertex cover problem. In *Proceedings of the 15th International Colloquium on Automata, Languages, and Programming (ICALP'05)*, pages 1043–1050, 2005.

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

[18] V. V. Vazirani. *Approximation Algorithms.* Springer-Verlag, 2001.

A Approximation Algorithm for Partial Capacitated Vertex Cover

```
PARTIAL-CAPACITATED-VERTEX-COVER($G, w, s$)

$R, C \leftarrow \emptyset$
$y_e, z \leftarrow 0$
**for** $u \in V[G]$
    $\delta'(u) = \delta(u)$                [unassigned edges incident to $u$]
    $\varphi(u) = \emptyset$                  [edges assigned to $u$]
    **if** $|\delta(u)| \leq k_u$            [$u$ is low degree]
        $L_u \leftarrow \delta(u)$
**do**
    PRUNING-STEP
    **if** $|E(R)| > s$
        **return** best candidate cover
    DUAL-UPDATE-STEP
    **if** dual update ended prematurely
        **if** $\sum_u \left\lceil \frac{|\varphi(u)|}{k_u} \right\rceil w_u <$ cost of best candidate cover
            **return** $\varphi$
        **else**
            **return** best candidate cover
```

```
PRUNING-STEP

let $g$ be the number of edges that remain unassigned
let $v$ be a vertex maximizing $\min\{k_v, |\delta'(v)|\}$
**while** $g - \min\{k_v, |\delta'(v)|\} \leq s$
    $\varphi_v \leftarrow \varphi$
    **if** $|\delta'(v)| > k_v$
        $\varphi_v(v) \leftarrow$ any $k_v$ edges in $\delta'(v)$
    **else**
        **for** $(v, x) \in L_v$
            add $(v, x)$ to $\varphi_v(v)$
            **if** $(v, x) \in \varphi_v(x)$
                remove $(v, x)$ from $\varphi_v(x)$
    **if** $C \neq \emptyset$
        let $r$ be the last vertex that became tight
        remove edges from $\varphi_v(r)$ until exactly $s$ edges remain unassigned
    let $\varphi_v$ be a candidate cover
    $w_v \leftarrow \infty$
    add $v$ to $R$
    let $v$ be a vertex maximizing $\min\{k_v, |\delta'(v)|\}$
```

```
DUAL-UPDATE-STEP
let $v$ be a vertex minimizing $\epsilon_v = \begin{cases} \frac{w_v - zk_v}{k_v} & \text{if } v \text{ is high degree} \\ \frac{w_v - \sum_{e \in L_v} y_e}{|\delta'(v))|} & \text{if } v \text{ is low degree} \end{cases}$
$z \leftarrow z + \epsilon_v$
for all unassigned edges $e$
    $y_e \leftarrow y_e + \epsilon_v$
add $v$ to $C$
if $|\delta'(v)| > k_v$                          [$v$ is high degree]
    for all $e = (v, u) \in \delta'(v)$
        add $e$ to $\varphi(v)$
        remove $e$ from $\delta'(u)$ and $\delta'(v)$
        if $|\delta'(u)| = k_u$                  [$u$ became low degree]
            $L_u \leftarrow \delta'(u)$
        if exactly $s$ edges remain unassigned
            end prematurely
else                                             [$v$ is low degree]
    for all $e = (v, u) \in L_v$
        add $e$ to $\varphi(v)$.
        if $e \in \varphi(u)$
            remove $e$ from $\varphi(u)$         [may reduce the number of copies of $u$ needed]
        if $e \in \delta'(u)$
            remove $e$ from $\delta'(u)$ and $\delta'(v)$
            if $|\delta'(u)| = k_u$              [$u$ became low degree]
                $L_u \leftarrow \delta'(u)$
```