Available at WWW.MATHEMATICSWEB.ORG POWERED BY SCIENCE DIRECT

Journal of Algorithms 48 (2003) 257–270

Journal of Algorithms

www.elsevier.com/locate/jalgor

Capacitated vertex covering ☆

Sudipto Guha,[a,1] Refael Hassin,[b] Samir Khuller,[c,*,2] and Einat Or[b]

[a] *Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA*
[b] *Department of Statistics and Operations Research, Tel-Aviv University, Tel-Aviv 69978, Israel*
[c] *Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA*

Received 3 March 2002

Abstract

In this paper we study the capacitated vertex cover problem, a generalization of the well-known vertex cover problem. Given a graph $G = (V, E)$ with weights on the vertices, the goal is to cover all the edges by picking a cover of minimum weight from the vertices. When we pick a copy of a vertex, we pay the weight of the vertex and cover up to a pre-specified number of edges incident on this vertex (its capacity). The problem is NP-hard. We give a primal–dual based approximation algorithm with an approximation guarantee of 2, and study several generalizations, as well as the problem restricted to trees.

Keywords: Approximation algorithm; Vertex cover; Capacitated network design

1. Introduction

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{1, \ldots, n\}$ and edge set E. Suppose that w_v denotes the weight of vertex v and k_v denotes the capacity of vertex v (we assume that k_v is an integer). A *capacitated vertex cover* is a function $x : V \to \mathbb{N}_0$ such that there exists an orientation of the edges of G in which the number of edges directed

☆ An extended abstract of this work appeared in ACM–SIAM Symposium on Discrete Algorithms (SODA) 2002.

* Corresponding author.

E-mail addresses: sudipto@cis.upenn.edu (S. Guha), hassin@post.tau.ac.il (R. Hassin), samir@cs.umd.edu (S. Khuller), eior@post.tau.ac.il (E. Or).

[1] The work was done while the author was at AT&T Research, Florham Park, NJ 07932.

[2] Research supported by NSF Awards CCR-9820965 and CCR-0113192.

into vertex $v \in V$ is at most $k_v x_v$. (These edges are said to be *covered* by, or *assigned* to v.) The *weight* of the cover is $\sum_{v \in V} x_v w_v$. The MINIMUM CAPACITATED VERTEX COVER problem is that of computing a minimum weight capacitated cover. The problem generalizes the MINIMUM WEIGHT VERTEX COVER problem which can be obtained by setting $k_v = |V| - 1$ for every $v \in V$. The main difference is that in vertex cover, by picking a node v in the cover we can cover all edges incident to v, in this problem we can only cover a subset of at most k_v edges incident to node v.

The problem originated in research at Glycodata,[3] a biotechnology company specializing in the areas of glycobiology and bioinformatics. One of its projects related to rational re-design of known drugs involves Glycoproteins. A glycoprotein is a protein that has U attachment points, in which it binds to a glycan. The group, H, of glycans that may appear in each attachment point is known and hence a glycoprotein has $|H|^U$ variants. The goal of this project is to determine which are the building blocks of the glycans comprising the variants of the glycoprotein found in a given (liquid) solution. Methods that identify the building blocks found in a solution exist. However, identifying the building blocks is not sufficient to determine the structure of any given variant found in the solution. It is therefore crucial to determine which of the building blocks are found in each variant, i.e., find a detailed description of the connectivity of the building blocks.

GMID (Glycomolecule ID) is a chip-based technology that is used to generate fingerprints which uniquely identify glycomolecules. It is able to answer in a single application a question of the form: For a given building block A, and for each member B in a set S of building blocks, does the solution contain a molecule which contains both building blocks A and B? The size of the set S is restricted, because of the specific technology. When planning an experiment that would use the GMID method to obtain information about a given solution, the required information may be presented as a graph where the building blocks are its vertices, and an edge exists between two vertices if the question regarding their connectivity is required. The device is able to answer $|S| = K$ questions at once if they share a common vertex. The problem of minimizing the number of experiments (i.e., GMID uses) needed to cover the required information graph, is *precisely* a capacitated vertex cover problem, with uniform capacities.

We denote by $\delta(v)$ the edges in E which are incident to v. We also denote by $d(v) = |\delta(v)|$ the degree of $v \in V$. For $S \subseteq V$ we denote by $G(S) = (V, E(S))$ the subgraph induced by S. We denote an edge with end-vertices i, j as a set $\{i, j\}$.

Since this problem generalizes vertex cover, one of the most studied problems in the area of approximation algorithms [1,9], it raises several very interesting directions for future research. There are many interesting results known about vertex cover—for example, the bipartite case can be solved in polynomial time, fixed parameter tractability, structural results, special properties about the fractional LP solutions etc [9]. It would be of interest to investigate all these properties in the context of capacitated covering.

Our problem is also related to work on the capacitated facility location problem, for the model where multiple facilities can be opened at the same location, and each facility

[3] URL: http://www.glycodata.com.

can only handle at most a specified demand. See Jain and Vazirani [11], and Chudak and Williamson [2] for recent work.

In fact, since the publication of an earlier draft of our work, several follow-up papers have appeared. First of all, using a method called *dependent rounding* Gandhi et al. [5] show how to get an alternate 2 approximation for the problem considered here, by using LP-rounding. If we add the constraint that only one copy of each vertex can be chosen, then for the unweighted case, Chuzhoy and Naor [3] have shown how to obtain a 3 approximation using LP rounding. This bound has recently been improved to 2 by Gandhi et al. [6].

1.1. Summary of results

The main results that we show are as follows. We give a primal–dual algorithm [8] that yields a factor 2 approximation for the basic problem. We also consider a generalization where each edge has a "demand" of d_e which has to be assigned to an adjacent vertex. For this generalization we show a factor 3 approximation. These results extend to r-hypergraphs (each edge in the hypergraph is a subset of at most r vertices and the edge must be assigned to one of these r vertices) with approximation factors of r and $r+1$, respectively.

Finally, when the graph is a tree we show that the problem can be solved in polynomial time, but for the more general version with edge demands the problem is NP-hard.

One can view Clarkson's greedy algorithm [4] for approximating vertex cover as a primal–dual algorithm. In this (and other algorithms) some vertices are chosen in the final solution. The cost for these vertices is charged to the dual variables corresponding to the adjacent edges. Some edges are charged once and some edges are charged twice. For vertex-cover, the fact that some edges are charged only once does not (apparently) help in improving the approximation bound. For our proof, this savings is crucial and helps us improve the bound from 3 to 2. While the actual algorithm and proof are more complex, at a high level this is the key insight for the improved approximation factor.

2. Integer programming formulation and a simple LP rounding scheme

A linear integer program (IP) of the problem can be written as follows. In this formulation, $y_{ev} = 1$ denotes that the edge $e \in E$ is covered by vertex v. Clearly, the values of x in a feasible solution correspond to a capacitated cover. While we do not really need the constraint $x_v \geqslant y_{ev}$ $v \in e \in E$ for the IP formulation, this constraint will play an important role in the relaxation. (In fact, without this constraint there is a large integrality gap between the best fractional and integral solutions. For example, consider a complete bipartite graph between two sets of vertices, A and B. A has two vertices, with each with weight W and capacity p^2. B has p vertices, each with weight 0 and capacity 1. Since there are $2p$ edges, the optimal solution must have cost W since by picking all the vertices in B we can only cover p edges. The fractional solution has cost at most W/p, by setting the x variables for the vertices in A as $1/(2p)$, and the x variables for the vertices in B as 1.)

$$\text{Minimize} \sum_v w_v x_v$$

$$
\begin{aligned}
&y_{eu} + y_{ev} \geqslant 1, \quad e = \{u, v\} \in E,\\
&k_v x_v - \sum_{e\in\delta(v)} y_{ev} \geqslant 0, \quad v \in V,\\
&x_v \geqslant y_{ev}, \quad v \in e \in E,\\
&y_{ev} \in \{0, 1\}, \quad v \in e \in E,\\
&x_v \in \mathbb{N}_0, \quad v \in V.
\end{aligned} \tag{1}
$$

We suggest the following algorithm: Solve (1) by relaxing the requirement that the variables take integral values. We require that $y_{ev} \geqslant 0$ and $x_v \geqslant 0$. If $y_{ev} \geqslant 1/2$ then we define the rounded value $y^*_{ev} = 1$ otherwise we define it as 0. For each edge $e = \{u, v\}$ either y_{eu} or y_{ev} is at least $1/2$, hence the edge can be assigned. (If both the rounded values, y^*_{eu} and y^*_{ev} are 1 then the edge can be assigned to either end.) Clearly, $y^*_{eu} \leqslant 2y_{eu}$ for the rounded value y^*.

We can now define x^*_v for $v \in V$, as $\lceil \frac{1}{k_v} \sum_{e\in\delta(v)} y^*_{ev} \rceil$. We claim that this rounding gives a 4-approximation:

Let $\sum_{e\in\delta(v)} y^*_{ev} = ak_v + k'_v$ where $0 \leqslant k'_v < k_v$ and $a \geqslant 0$.

$$\sum_{e\in\delta(v)} y^*_{ev} \leqslant \sum_{e\in\delta(v)} 2y_{ev} \leqslant 2k_v x_v \quad \Rightarrow \quad ak_v + k'_v \leqslant 2k_v x_v,$$

implying that $x_v \geqslant a/2 + (k'_v/2k_v)$.

Clearly, $x^*_v \leqslant (a + 1)$. We will prove that $x^*_v \leqslant 4x_v$. It is sufficient to prove that $a + 1 \leqslant 4[a/2 + (k'_v/2k_v)]$. We need to show that $a + 1 \leqslant 2a + 2k'_v/k_v$. If $a \geqslant 1$ we are done. If $a = 0$, then $x^*_v = 1$ while $x_v \geqslant 1/2$, since $y_{ev} \geqslant 1/2$. The last case is when $a = 0$ and $k'_v = 0$ in which case $x^*_v = 0$ and the claim holds.

3. Primal–dual algorithm

We develop a primal–dual algorithm that gives a 2-approximation. While the algorithm is quite simple, the proof is somewhat subtle.

The dual problem of the relaxation of (1) is given in (2):

$$
\begin{aligned}
&\text{Maximize} \sum_{e\in E} \alpha_e\\
&k_v q_v + \sum_{e\in\delta(v)} l_{ev} \leqslant w_v, \quad v \in V,\\
&q_v + l_{ev} \geqslant \alpha_e, \quad v \in e \in E,\\
&q_v \geqslant 0, \quad v \in V,\\
&l_{ev} \geqslant 0, \quad v \in e \in E,\\
&\alpha_e \geqslant 0, \quad e \in E,\ v \in V.
\end{aligned} \tag{2}
$$

High level description of the algorithm

Initially, no edges are assigned and all vertices are closed. As the algorithm runs, it declares certain vertices as open. When a vertex v is marked open, certain edges are assigned to it. In fact, when v is marked open, *all* unassigned edges that are incident to v are assigned to it. However, later on, if another vertex u that is adjacent to v is opened, an edge between u and v that was previously assigned to v *may* get re-assigned to u. In the end, the algorithm chooses the value of x_v^* to be $\lceil y_v/k_v \rceil$ where y_v is the number of edges assigned to v. The formal description of the algorithm is given in Fig 1.

```
Min_Capacitated_Cover
  input
    1. A graph $G = (V, E)$.
    2. A capacity function $k: V \to \mathbb{N}_0$.
    3. A cost function $w: V \to \mathbb{N}_0$.
  output
    A capacitated cover.
  begin
    $E' := E$ [$E'$ is the set of unassigned edges].
    $V' := V$ [$V'$ is the set of closed vertices].
    for every $v \in V$
      $\delta'(v) :=$ edges in $E'$ incident with $v$. $d'_v := |\delta'(v)|$.
      If $d'_v > k_v$ then $D_v := \emptyset$; otherwise $D_v := \delta'(v)$.
      $w'_v := w_v$.
    end for
    while $E' \neq \emptyset$
      $r_v := w'_v / \min(k_v, d'_v)$, $v \in V'$.
      $u := \arg\min(r_v\colon v \in V')$. (break ties arbitrarily)
      $V' := V' \setminus \{u\}$.
      $w'_v := w'_v - r_u \min(k_v, d'_v)$, $v \in V'$.
      if $d'_u > k_u$
        then
          Assign the edges in $\delta'(u)$ to $u$.
        else [$d'_u \leqslant k_u$]
          Assign the edges in $D_u$ to $u$.
          [For $D_u \setminus E'$ this is a re-assignment.]
      end if
      for every $\{u, v\} \in \delta'(u)$
        $E' := E' \setminus \{\{u, v\}\}$. $\delta'(v) := \delta'(v) \setminus \{\{u, v\}\}$.
        $d'_v := d'_v - 1$.
      if $d'_v = k_v$
        then $D_v := \delta'(v)$.
      end if
      end for
    end while
    return
    $x_v^* := \lceil |\text{edges assigned to } v| / k_v \rceil$.
  end Min_Capacitated_Cover
```

Fig. 1. Algorithm Min_Capacitated_Cover.

Initially, all the dual variables α_e are 0. This is a dual feasible solution (with all $q_v = 0$ and $l_{ev} = 0$). We use E' to denote the set of unassigned edges. We use $\delta'(v)$ to denote the unassigned edges currently incident on vertex v. We use $d'(v)$ to denote the number of unassigned edges currently incident on vertex v. We now explain how vertices are marked open. This is done by increasing all the dual variables α_e for the *unassigned edges* $e \in E'$ simultaneously. In the dual program, there are two kinds of constraints—vertex constraints and edge constraints.

To maintain dual feasibility of the edge constraints $q_v + l_{ev} \geqslant \alpha_e$, as we increase α_e, we have to increase q_v or l_{ev}. If the vertex has a large number of unassigned edges incident to it, then we increase q_v, otherwise we increase l_{ev}. Formally, if $d'(v) > k_v$ then we increase q_v, otherwise we increase l_{ev}.

For each vertex constraint, $k_v q_v + \sum_{e \in \delta(v)} l_{ev} \leqslant w_v$, initially the left-hand side is 0 and the right-hand side is the weight of the vertex. While increasing the dual variables for the unassigned edges, we stop as soon as a vertex constraint is met with equality. (In Fig. 1 this is vertex u in the main loop.) We declare this vertex as open. We assign to this vertex, u, all unassigned edges incident to it and have to stop increasing their dual variables. In addition to assigning edges in $\delta'(u)$ to u, we may also re-assign some of the previously assigned edges from $\delta(u)$ to u. We now elaborate on this point further.

For any vertex v, as soon as $d'(v) \leqslant k_v$, we *define* D_v to be the set of unassigned edges incident to vertex v. If a vertex has its initial degree at most k_v then this condition is true at the start of the algorithm. For other vertices, the initial degree may exceed k_v, but as edges are assigned, it may happen that at some stage $d'(v) = k_v$. If this event happens we define D_v to be the set of k_v edges from $\delta'(v)$ that are currently unassigned. Later on, $\delta'(v)$ may change but not D_v.

When a vertex v is declared open, if $d'(v) > k_v$ then we assign all unassigned edges in $\delta'(v)$ to v. If $d'(v) \leqslant k_v$ then we assign all edges in D_v to v. Note that some of these edges may have earlier been assigned to other vertices. (Only the edges in $\delta'(v)$ are previously unassigned.)

The pseudo-code description of the algorithm is given in Fig. 1.

Theorem 3.1. *Min_Capacitated_Cover returns a* 2*-approximation for* MINIMUM CAPACITATED COVER.

Proof. The algorithm opens a multi-set of vertices S as centers. The total cost of the solution can be represented by $w(S)$, the total weight of the subset S, counting multiple copies. Any vertex v that is declared open has the property that $w_v = k_v q_v + \sum_{e \in \delta(v)} l_{ev}$. We will charge the weight for v (all copies of v) to edges in $\delta(v)$. We will show (Lemma 3.2) that each edge gets a charge of at most $2\alpha_e$. Since the dual solution has value $\sum_e \alpha_e$, and is a lower bound on the optimal solution, we get the required bound. In other words $w(S) \leqslant 2\sum_e \alpha_e \leqslant 2w(OPT)$ where OPT is an optimal cover. $\square$

Lemma 3.2. *We can charge the weight of each open vertex v (all copies) to edges in $\delta(v)$ such that each edge e gets a charge of at most $2\alpha_e$.*

Proof. Define a vertex to be a low degree vertex if *when it is declared open* $d'(v) \leqslant k_v$, otherwise it is defined to be a high degree vertex. We will discuss the charging mechanism for both low degree and high degree vertices.

Consider a low degree vertex v. We pick only one copy of the vertex. We will charge the weight of this vertex to *all* edges in the set D_v. All these edges are assigned to v by the algorithm when this vertex is declared open, regardless of having been assigned earlier. In fact, some of the edges in D_v may be assigned to other vertices later on and are thus charged again.

If $d(v) \leqslant k_v$ then $D_v = \delta(v)$ and $q_v = 0$, and since the vertex constraint is tight then $w_v = \sum_{e\in\delta(v)} l_{ev} = \sum_{e\in\delta(v)} \alpha_e$. We thus charge the cost of vertex v to all incident edges by charging α_e to each $e \in \delta(v)$. If $d(v) > k_v$ then at some point of time $d'(v) = k_v$. At this point we fix the value of q_v and subsequently increase the l_{ev} variables. Note that $|D_v| = k_v$. When this vertex is declared open, we have that $w_v = k_v q_v + \sum_{e\in\delta(v)} l_{ev}$. For the edges not in D_v, note that $l_{ev} = 0$. Hence $w_v = k_v q_v + \sum_{e\in D_v} l_{ev}$. Since there are exactly k_v edges in D_v, we have $w_v = \sum_{e\in D_v}(q_v + l_{ev}) = \sum_{e\in D_v} \alpha_e$. Thus, the cost for vertex v is charged to all edges in D_v.

Now consider a high degree vertex v. Suppose $\delta'(v)$ is the set of unassigned edges incident to v when v is declared open. In our charging scheme these will be the only edges that will be charged by v, and previously assigned edges incident on v will not be charged. Some of these edges, a subset $R_v \subseteq \delta'(v)$, will be re-assigned to other vertices.

For a high degree vertex v, we have $l_{ev} = 0$ and $\alpha_e = q_v$ for each unassigned edge when the vertex is declared open. Thus $w_v = k_v q_v$. Since $\alpha_e = q_v$ the cost for a single copy of v needs to be charged to k_v edges. Let $\delta'(v) = p_v k_v + k'_v$ where $0 \leqslant k'_v < k_v$. If $|R_v| \geqslant k'_v$ then the number of edges assigned to v is at most $p_v k_v$. In this case the cost for p_v copies can be charged to any $p_v k_v$ edges in $\delta'(v)$. Each edge is charged at most $w_v/k_v = q_v = \alpha_e$. If $|R_v| < k'_v$ then we need $p_v + 1$ copies of v. The cost for these copies is charged to $(p_v + 1)k_v$ edges. We can charge all the edges in $\delta'(v)$ once. We still need to charge $k_v - k'_v$ edges. Since there are at least $p_v k_v$ edges that are not re-assigned, their other ends are not charged. (If the other end is ever opened, if it is a high degree vertex then it does not re-assign these edges and does not charge them. If it is a low degree vertex then the edge is re-assigned and belongs to R_v. The edges in R_v are charged at most once by v.) Since $p_v \geqslant 1$, we have at least k_v edges that we can charge a second time.

Finally, for any edge $e = \{u, v\}$ if only one end is open then the edge is charged at most twice. If both ends are open, and both are high degree, then only the end that the edge is assigned to can charge it. If both ends are low degree then it is charged at most once from each end. If one end is low degree and one end is high degree, then the edge is assigned to the low degree end and charged once from each end. This completes the proof of the lemma. □

3.1. r-hypergraphs

The above primal–dual algorithm yields a factor r approximation algorithm for r-hypergraphs. The only difference is that one hyperedge contains at most r vertices. To prove that the approximation factor is r we need to prove a lemma analogous to Lemma 3.2. Of course, in this case we would prove that a hyperedge is charged exactly $r\alpha_e$.

The proof of such a lemma is straightforward and we indicate the adjustments required for the proof to go through. The definitions of "low" and "high" degree vertices remain the same. The critical observation needed is that a hyperedge cannot be assigned to two low degree vertices.

Once again, if the hyperedge is attached to only one open vertex, the hyperedge is charged at most twice. If the hyperedge is adjacent only to open vertices with high degree, then only the vertex to which the hyperedge is assigned to charges the hyperedge. The charge in this case is also at most $2\alpha_e$. Otherwise the hyperedge is adjacent to at least one low degree vertex and therefore assigned to some low degree vertex. In this case each open vertex the hyperedge contains charges the hyperedge at most once.[4] Thus the total charge is at most $r\alpha_e$. This yields an r-approximation.

4. Approximation with d_e

Consider the case that each edge has demand d_e, and each vertex v has the property that if r copies of it are open, then summing over the edges assigned to v, $\sum_e d_e \leqslant rk_v$. In this case we have a 3-approximation. The primal and dual linear programs in this case are:

$$\begin{aligned}
&\text{Minimize } \sum_v w_v x_v\\
&y_{eu} + y_{ev} \geqslant 1, \quad e = \{u, v\} \in E,\\
&k_v x_v - \sum_{e\in\delta(v)} y_{ev} d_e \geqslant 0, \quad v \in V,\\
&x_v \geqslant y_{ev}, \quad v \in e \in E,\\
&y_{ev} \in \{0, 1\}, \quad v \in e \in E,\\
&x_v \in \mathbb{N}_0, \quad v \in V.
\end{aligned} \tag{3}$$

$$\begin{aligned}
&\text{Maximize } \sum_{e\in E} \alpha_e\\
&k_v q_v + \sum_{e\in\delta(v)} l_{ev} \leqslant w_v, \quad v \in V,\\
&q_v d_e + l_{ev} \geqslant \alpha_e, \quad v \in e \in E,\\
&q_v \geqslant 0, \quad v \in V,\\
&l_{ev} \geqslant 0, \quad v \in e \in E,\\
&\alpha_e \geqslant 0, \quad e \in E,\ v \in V.
\end{aligned} \tag{4}$$

We grow α_e in proportion to the d_e values. If $\sum_{e\in\delta'(v)} d_e > k_v$ raise q_v, otherwise raise l_{ev} appropriately. Once again, as soon as a vertex is open assign all unassigned edges

[4] Actually two high degree vertices cannot both charge a hyperedge. This is implicit in the case analysis of Lemma 3.2 Therefore the worst case is when the edge is attached to r low degree open vertices or $r - 1$ low degree open vertices along with a high degree open vertex.

adjacent to it by sufficiently many copies. We do not perform any re-assignments. Let r be the minimum integer such that for all adjacent unassigned edges e, $\sum_{e\in\delta'(v)} d_e \leqslant rk_v$.

It is easy to observe that if $r > 1$ we have $rk_v \geqslant \sum_{e\in\delta'(v)} d_e \geqslant \frac{1}{2}rk_v$ for assigned edges e. In this case $\alpha_e = d_e q_v$ for all assigned edges. We will charge all edges $2\alpha_e$, and since

$$2\sum_{e\in\delta'(v)} \alpha_e = \sum_{e\in\delta'(v)} (2d_e q_v) \geqslant rk_v q_v = rw_v$$

we can pay for all copies.

If $r = 1$ we open one copy and each adjacent edge pays α_e; in this case as before we may charge an edge already assigned elsewhere. Over all these edges by dual feasibility and the growing process, $\sum_{e\in\delta'(v)} \alpha_e \geqslant w_v$. Thus each edge gets charged at most $3\alpha_e$.

Therefore we can claim the following,

Theorem 4.1. *For the capacitated vertex cover problem with arbitrary demands on edges and arbitrary capacity and costs of vertices we have a factor* 3 *approximation.*

It appears that a re-assignment is feasible and that should reduce the cost to $2\alpha_e$ per edge e. However no simple re-assignment exists since the low degree vertex (under-saturated, filled to less than capacity) may get over-saturated and a high degree vertex (over-saturated) becomes under-saturated, thus disallowing any reassignment. This is illustrated in the example below.

4.1. Gap example

We observe that in the above primal dual method as long as we grow α_e in the intuitive fashion as described above and only select the open vertices in the final solution we can only hope for a factor 3 approximation.

Consider the chain of length three defined by vertices $ABCD$. Edges AB and CD have demand 1. BC has demand $k > 2$. Capacity of A, and D are 1. Capacity of B, and C are k. Weight of C is c. Weight of A, and D are $c/k + c\epsilon'$, and of B is $c(1+\epsilon)$ with $1/k > \epsilon' > \epsilon$. If both ϵ, ϵ' are small, the optimal solution is AB being assigned to A, BC to C, and CD to D.

It is easy to verify that in our process we will only declare B and C to be open. Since C is cheaper, we can at best have a cost of $3c + c\epsilon$. Thus the best ratio we can hope is $3 - (2 + 2k\epsilon' - k\epsilon)/(k + 2 + 2k\epsilon') > 3 - 4/(k + 4)$.

4.2. The r-hypergraph case

In this case the 3-approximation algorithm for graphs yields an $r + 1$ approximation for r hypergraphs. The critical observation is that a hyperedge cannot be charged by two or more "high" degree vertices. A low degree vertex charges the edge at most α_e. Thus the worst case would be when the edge is adjacent to $r - 1$ low degree open vertices and one high degree open vertex. Thus the total charge is at most $(r + 1)\alpha_e$.

5. Capacitated covers on trees

In this section we consider the MINIMUM CAPACITATED VERTEX COVER ON A TREE (CVCT). In the most general version we assume that the input consists of vertex weights w_v, integer capacities $k_v \geqslant 0$, $v \in V$, and integer edge demands $d_e \geqslant 1$ $e \in E$. We consider two variations with respect to whether or not the demand of an edge $e = (u, v)$ can be split so that $d_e = d_e^v + d_e^u$, where d_e^v is assigned to v and d_e^u is assigned to u. For both formulations CVCT is NP-hard.

Theorem 5.1. *CVCT is NP-hard even when $k_v = k$ $\forall v \in V$. This claim holds in both cases when edge demands can and cannot be split.*

Proof. To prove NP-hardness we do a reduction from the KNAPSACK PROBLEM, which is known to be NP-hard [7]. Consider an instance $\max \sum_{v\in S} c_v \mid S \subset \{1, \ldots, n\}$, $\sum_{v\in S} a_v \leqslant B$ of the knapsack problem. We create the following CVCT: The graph is a star with root 0 and leaves $\{1, \ldots, n+1\}$. $k_v = B + 1$ for $v = \{0, \ldots, n+1\}$. $w_v = c_v$ for $v = 1, \ldots, n$, $w_0 > \sum_{v=1}^n w_v$, and $w_{n+1} > w_0$. $d_{0,v} = a_v$ for $v = 1, \ldots, n$ and $d_{0,n+1} = 1$.

Since $w_{n+1} > w_0$, it is optimal to assign $d_{0,n+1}$ to the root 0. There is an unused capacity of size B at node 0. We wish to minimize the other costs by computing a subset S of $\{1, \ldots, n\}$ such that $\sum_{v\in S} w_v$ is maximized subject to the constraint $\sum_{v\in S} d_v \leqslant B$. The optimal solution for this CVCT gives an optimal solution for the knapsack problem. Note that even if splitting the demand is allowed, it would be of no use because if a demand of an edge is split we pay for both ends and therefore increase the cost. □

We now describe three special cases that can be solved in polynomial time. In each case we assume that the input graph is a tree $T = (V, E)$. We root T at an arbitrary vertex and renumber the vertices so that a child of a vertex has a smaller index than its parent. We define T_v as the subtree rooted at v.

The first special case assumes unit edge demands. It can be solved by algorithm Min_k-Cover unit demand (Fig. 2). The algorithm computes for every $v \in V$ two values defined as follows: let e_v be the edge connecting v and its parent ($e_n = \emptyset$). W_v^{out} is the cost of a minimum capacitated cover of T_v, and W_v^{in} is the cost of a minimum capacitated cover of $T_v \cup \{e_v\}$ under the restriction that e_v is assigned to v.

Theorem 5.2. *Algorithm Min_ k-Cover unit demand* (*Fig.* 2) *computes the minimum cost of CVCT when $d_e = 1$ for every $e \in E$.*

Proof. The proof is by induction on the index u. We omit the straightforward details. □

Given W_v^{out} and W_v^{in} for $v = 1, \ldots, n$, one can recursively obtain the assignment of the edges of T.

Our next special case assumes uniform weights. We assume that $w_v = 1$ for every $v \in V$. A restricted version of this case in which the capacities are uniform ($k_v = k$ for every

```
Min_ k-Cover unit demand
  inputs
    1. $T = (V, E)$.
    2. A capacity function $k_v$, $v \in V$.
    3. A weight function $w_v$, $v \in V$.
  output
    The cost of a capacitated cover.
  begin
    for $v = 1$ to $n$
      $W_v^{\text{out}} := 0$, $W_v^{\text{in}} := w_v$.
    end for
    for $u = 1$ to $n$
      $A_u :=$ set of children of $u$.
      for every $v \in A_u$.
        $C_v = W_v^{\text{in}} - W_v^{\text{out}}$.
      end for
      Sort $v \in A_u$ in non-increasing order of $C_v$.
      while $A_u \neq \emptyset$
        $A :=$ the first $\min\{k_u, |A_u|\}$ vertices in $A_u$.
        if $\sum_{v \in A} C_v \geqslant w_u$
          then Assign all edges $\{v, u\}$, $v \in A$ to $u$.
            $A_u := A_u \setminus A$.
          else
            Assign edges $\{v, u\}$, $v \in A_u$ to $v$.
            $A_u := \emptyset$.
        end if
      end while
      $S := \{v\colon \{v, u\}$ is assigned to $v\}$.
      $F := \{v\colon \{v, u\}$ is assigned to $u\}$.
      $S_k := \{\min(k_u - 1, |S|)$ highest C-value vertices in $S\}$.
      $W_u^{\text{out}} := \sum_{v \in S} W_v^{\text{in}} + \sum_{v \in F} W_v^{\text{out}} + w_u \lceil |F|/k_u \rceil$.
      if $|F| = 0 \pmod{k_u}$
        then $W_u^{\text{in}} := W_u^{\text{out}} + \min(w_u - \sum_{v \in S_k} C_v, \min_{v \in F} C_v)$.
        else $W_u^{\text{in}} := W_u^{\text{out}}$.
      end if
    end for
    return $W_n^{\text{out}}$.
  end Min_ k-Cover unit demand
```

Fig. 2. Algorithm Min_k-Cover unit demand.

$v \in V$) was solved in Jaeger and Goldberg [10] as a special case of the capacitated facility location problem on trees. We first show how to solve this case assuming that the demand can be split.

Theorem 5.3. *If $w_v = 1$ for every $v \in V$ and the demand can be split then algorithm Min_ k_ d_cover_splitable_demand* (*Fig.* 3) *returns a minimum cost solution.*

Again, the proof is straightforward and we omit the details.

We now turn to the case where the demand cannot be split.

```
Min_ k_ d-cover splitable demand
  input
    1. $T = (V, E)$, with all $w_v = 1$.
    2. A capacity function $k_v$ $v \in V$.
    3. A demand function $d_e$ $e \in E$.
  output
    A capacitated cover.
  begin
    for every $v \in V$, $D_v := 0$
      [$D_v$ is the total demand assigned to $v \in V$].
    end for
    for $v = 1$ to $n - 1$
      $u :=$ the parent of $v$.
      $c_v := \lceil D_v / k_v \rceil k_v - D_v$.
      [$c_v$ is the spare capacity of $v$].
      $e := (v, u)$.
      $m := \min\{d_e, c_v\}$.
      $D_v := D_v + m$.
      $d'_e := d_e - m$.
      if $k_u \geqslant d'_e$ or $k_u \geqslant k_v$
        then $D_u := D_u + d'_e$.
        else ($k_u < \min\{d'_e, k_v\}$)
          $a_e := \lfloor d'_e / k_v \rfloor k_v$.
          $D_v := D_v + a_e$.
          $d'_e := d'_e - a_e$.
          if $k_u < D'_e$
            then $D_v := D_v + d'_e$.
            else ($k_u \geqslant d'_e$)
              $D_u := D_u + d'_e$.
          end if
      end if
    end for
    return $x_u := \lceil D_u / k_u \rceil$ $u = 1, \ldots, n$.
  end Min_ k_ d-cover splitable demand
```

Fig. 3. Algorithm Min_k_d-cover splitable demand.

Theorem 5.4. *If $w_v = 1$ for every $v \in V$ and the demand cannot be split then algorithm Min_k_ d_cover_unsplitable_demand (Fig. 4) returns a minimum cost solution.*

Proof. The proof is by induction on the index u. The induction's assumption is that the algorithm returns an optimal solution for every subtree in the forest induced by the edges connecting the vertices $1, \ldots, u$ and their parents. Moreover, this solution has maximal spare capacity at the root of the subtree, among all optimal solutions to this subtree. For $n = 1$ these properties trivially hold.

Assume the claim holds for $v < u$. It is important to observe that by definition, $c_u < k_u$ so that using the spare capacity at a vertex may save at most one center.

Consider the assignment of the demand d_e where $e = (u, v)$. The assignment made by the algorithm clearly preserves the induction's hypothesis when $\lceil d_e / k_u \rceil \neq N_v$. A lower cost is attained by constructing centers at the vertex that requires less centers. It may be that the number of centers required at u to serve d_e is greater than N_v by 1 and that by using

```
Min_ k_ d-cover unsplitable demand
  inputs
    1. $T = (V, E)$, with all $w_v = 1$.
    2. A capacity function $k_v$, $v \in V$.
    3. A demand function $d_e$, $e \in E$.
  output
    A capacitated cover.
  begin
    for every $v \in V$
      $D_v := 0$.
      [$D_v$ is the total demand assigned to $v$.]
    end for
    for $v = 1$ to $n - 1$
      $u :=$ the parent of $v$.
      $c_v := \lceil D_v/k_v \rceil k_v - D_v$.
      [$c_v$ is the spare capacity of $v$].
      $e := (v, u)$.
      $N_v := \max\{\lceil (d_e - c_v)/k_v \rceil, 0\}$.
      [$N_v$ is the number of centers one must place in $v$ to cover $d_e$].
      if $\lceil d_e/k_u \rceil \leqslant N_v$.
        then $D_u := D_u + d_e$.
        else $(\lceil d_e/k_u \rceil > N_v)$
          $D_v := D_v + d_e$
      end if
    end for
    return $x_u := \lceil D_u/k_u \rceil$ $u = 1, \ldots, n$.
  end Min_k_ d-cover unsplitable demand
```

Fig. 4. Algorithm Min_k_d-cover unsplitable demand

the spare capacity at u a center can be saved. However, in this case both solutions have the same cost whereas the one that assigns the demand to v come with a greater remaining spare capacity at u.

Finally, if $\lceil d_e/k_u \rceil = N_v$ then the two options may come with the same cost but assigning the demand to u either lowers the cost by using spare capacity there or increases the spare capacity. □

If the edge demand is uniform the question of whether splitting is allowed determines the hardness of the problem. If splitting is forbidden the problem is easy and algorithm Min_Capacitated_Cover_for_Tree holds. In the case where splitting is allowed the problem is NP-hard as we show in Theorem 5.5.

Theorem 5.5. *The CVCT with splitable demand is NP-hard even when the edge demands are uniform.*

Proof. To prove NP-hardness we apply a reduction from the KNAPSACK PROBLEM. Consider an instance $\max \sum_{v \in S} c_v \mid S \subset \{1, \ldots, n\}, \sum_{v \in S} a_v \leqslant B$ of the knapsack problem. We solve the following CVCT: The graph is a two level tree with root 0 and $n + 1$ children denoted $1, \ldots, n$ and $2n + 1$. Each of the children $v = 1, \ldots, n$ has a child denoted by $u = n + v$. Child $2n + 1$ of the root vertex 0 has no children. All edges have demand

d where $\max\{a_i\colon i = 1, \ldots, n\} \leqslant d < B$. The capacities are $k_0 = B + d$, $k_v = 2d - a_v$ for $v = 1, \ldots, n$ and $k_v = 1$, $n < v \leqslant 2n$. The weights are $w_v c_v$ for every $v = 1, \ldots, n$, $w_0 \geqslant \sum_{v=1}^{n} w_v$ and $w_i \gg w_0$ for $i > n$.

It is optimal to assign the demand of $(0, 2n + 1)$ to 0 and the demand of $e = (v, n + v)$ $v = 1, \ldots, n$ to v. After the assignment, vertex v, $v = 1, \ldots, n$, has a spare capacity of $d - a_v$ that we have already paid for. It is optimal to assign $d - a_v$ of the demand of edge $e = (0, v)$ to v since it is the only edge that may use the spare capacity. After this assignment, the unassigned demand of edge $(0, v)$ is a_v for $v = 1, \ldots, n$. The remaining problem is identical to the one in the reduction introduced in Theorem 5.1. $\square$

References

[1] R. Bar-Yehuda, S. Even, A linear time approximation algorithm for the weighted vertex cover problem, J. Algorithms 2 (1981) 198–203.

[2] F. Chudak, D. Williamson, Improved approximation algorithms for capacitated facility location problems, in: Proc. of 1999 Integer Programming and Combinatorial Optimization Conference, in: Lecture Notes in Comput. Sci., Vol. 1610, 1999, pp. 99–113.

[3] J. Chuzhoy, J. Naor, Covering problems with hard capacities, in: Proc. of the 43rd IEEE Symposium on Foundations of Computer Science, 2002, pp. 481–489.

[4] K. Clarkson, A modification to the greedy algorithm for the vertex cover problem, Inform. Process. Lett. 16 (1983) 23–25.

[5] R. Gandhi, S. Khuller, S. Parthasarthy, A. Srinivasan, Dependent rounding on bipartite graphs, in: Proc. of the 43rd IEEE Symposium on Foundations of Computer Science, 2002, pp. 323–332.

[6] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, A. Srinivasan, An improved approximation algorithm for vertex cover with hard capacities, to appear in, in: Proc. of International Colloquium on Automata Languages and Programming, 2003.

[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1978.

[8] M.X. Goemans, D.P. Williamson, A general approximation technique for constrained forest problems, SIAM J. Comput. 24 (1995) 296–317.

[9] D.S. Hochbaum (Ed.), Approximation Algorithms for NP-hard problems, PWS, 1996.

[10] M. Jaeger, J. Goldberg, A polynomial algorithm for the equal capacity p-center problem on trees, Transportation Sci. 28 (1994) 167–175.

[11] K. Jain, V.V. Vazirani, Primal–dual approximation algorithms for metric facility location and k-median problems, in: Proc. of the 40th IEEE Symposium on Foundations of Computer Science, 1999, pp. 2–13.