Set Cover Revisited: Hypergraph Cover with Hard Capacities[⋆]

Barna Saha[1] and Samir Khuller[2]

[1] AT&T Shannon Research Laboratory
[2] University of Maryland College Park
barna@research.att.com, samir@cs.umd.edu

Abstract. In this paper, we consider generalizations of classical covering problems to handle hard capacities. In the hard capacitated set cover problem, additionally each set has a covering capacity which we are not allowed to exceed. In other words, after picking a set, we may cover at most a specified number of elements. Based on the classical results by Wolsey, an $O(\log n)$ approximation follows for this problem.
Chuzhoy and Naor [FOCS 2002], first studied the special case of unweighted vertex cover with hard capacities and developed an elegant 3 approximation for it based on rounding a natural LP relaxation. This was subsequently improved to a 2 approximation by Gandhi et al. [ICALP 2003]. These results are surprising in light of the fact that for weighted vertex cover with hard capacities, the problem is at least as hard as set cover to approximate. Hence this separates the unweighted problem from the weighted version.
The set cover hardness precludes the possibility of a constant factor approximation for the hard-capacitated vertex cover problem on weighted graphs. However, it was not known whether a better than logarithmic approximation is possible on unweighted *multigraphs*, i.e., graphs that may contain parallel edges. Neither the approach of Chuzhoy and Naor, nor the follow-up work of Gandhi et al. can handle the case of multigraphs. In fact, achieving a constant factor approximation for hard-capacitated vertex cover problem on unweighted multigraphs was posed as an open question in Chuzhoy and Naor's work. In this paper, we resolve this question by providing the first constant factor approximation algorithm for the vertex cover problem with hard capacities on unweighted multigraphs. Previous works cannot handle hypergraphs which is analogous to consider set systems where elements belong to at most f sets. In this paper, we give an $O(f)$ approximation algorithm for this problem. Further, we extend these works to consider partial covers.

1 Introduction

Covering problems have been widely studied in computer science and operations research, starting from the early work on set-cover [11, 15, 18]. In addition, the vertex

[⋆] Research supported by NSF CCF-0728839, NSF CCF-0937865 and a Google Research Award.

cover problem has been extremely well studied as well – this is a special case of set cover, where each element belongs to exactly two sets [2, 10]. Both these problems have played a central role in the development of many important ideas in algorithms – greedy algorithms, LP rounding, randomized algorithms, primal-dual methods, and have been the vehicle to convey many central ideas in combinatorial optimization.

In this paper, we consider covering problems with hard capacity constraints. In other words, if a set is chosen, it cannot cover all its elements, but there is an upper bound on the number of elements that the set can cover. More formally, consider a ground set of elements $\mathcal{U} = \{a_1, a_2, \ldots, a_n\}$ and a collection of subsets of $\mathcal{U}$, $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$. Each set $S \in \mathcal{S}$ has a positive integral capacity $k(S) \in \mathbb{N}$ and has an upper bound (denoted by $m(S)$) on the number of copies. In addition, each set can have arbitrary non-negative weight $\tilde{w} : \mathcal{S} \rightarrow \mathbb{R}^+$. A solution for capacitated covering problem contains each set $S \in \mathcal{S}$, $x(S)$ times where $x(S) = \{0, 1, 2, \ldots, m(S)\}$ such that there is an assignment of at most $x(S)k(S)$ elements to set S and all the elements are covered by the assignment. The goal is to minimize $\sum_{S \in \mathcal{S}} \tilde{w}(S)x(S)$. Using Wolsey's greedy algorithm [18], we can easily derive a $O(\log n)$ approximation for the capacitated set cover problem with hard capacities.

Approximation algorithms for vertex cover with (soft) capacities were developed by Guha et al [9]. In the soft capacitated covering problem there is no bound on the number of copies of each set (vertex) that can be chosen. In [9], a primal dual algorithm was developed to give a 2 approximation. This algorithm can be extended easily to handle vertex cover with (soft) capacities in hypergraphs. In other words, if we have a hyper graph with hyper edges of size at most f (set cover problem where each element belongs to at most f sets), then we can easily get an f approximation [9]. On the other hand, the case of hard capacities is quite difficult. In a surprising result, Chuzhoy and Naor [4] showed that the weighted vertex cover problem with hard capacities is set-cover hard and showed that for *unweighted* graphs a randomized rounding algorithm can give a 3 approximation. This was subsequently improved to a 2 approximation [7]. Vertex cover is a special case of set cover problem where $f = 2$. This naturally raises the question whether it is possible to obtain an f approximation for the unweighted set cover problem with hard capacities, where each element belongs to at most f sets. The approaches of [4, 7] do not extend to case when $f > 2$. Moreover, the results of [4, 7] only hold for *simple* graphs. *Obtaining a constant factor approximation algorithm for the hard-capacitated vertex cover problem for unweighted multigraphs was posed as an open question in [4]. In this paper, we resolve that question, and extending our approach we also obtain an $O(f)$-approximation for the unweighted set cover problem with hard capacities.* Further, we also provide an $O(f)$ approximation algorithm for partial cover problem with hard capacities. Partial cover is a natural generalization of covering problems where only a desired number of elements need to be covered [8]. While the works of [3, 17] extended the vertex cover with soft capacities to consider partial cover, nothing prior to our work was known in the case of hard capacities.

The notion of capacities is also natural in the context of facility location problems, as well as clustering problems and has been widely studied. Capacitated facility location and k-median problems have been an active area of research [1, 5, 16] and frequently appear in applications involving placement of warehouses, web caches and as a subroutine in several network design protocols. Non-metric capacitated facility location problem is a generalization of hard-capacitated set cover problem for which Bar-Ilan et al. [1] gave an $O(\log n + \log m)$-approximation. In this problem, there are m facilities and n clients; there is a cost associated for opening each facility and each client connects to one of the open facility paying a connection cost while the number of clients that can be assigned to an open facility remains bounded by its capacity. When, the connection costs are either 0 or ∞, we get the set cover problem with hard capacities.

In several set cover applications, an element only belongs to a few sets. This is especially true in the context of scheduling. One such example is the work of Khuller, Li and Saha [12] where they study a scheduling algorithm to allocate jobs to machines in data centers such that the minimum number of machines are activated. The goal is to minimize the energy to run machines while maintaining the makespan (maximum sum of processing times on any machine). In data centers, each data is replicated a *small* number of times (typically 3 copies). Thus a job needed to access specific data can be run on one of a small number of machines. In [12], a $(\ln n+1)$ approximation algorithm is provided that violates the makespan by a factor of 2. However, it does not consider the fact that each job can be scheduled only on f (here $f \approx 3$) machines. Incorporating this, and in addition, considering that jobs have some fixed processing time, we obtain the hard-capacitated set cover problem with elements belonging to at most f sets. The scheduling model of [6] can also be seen as a hard-capacitated set covering instance with *multiple* capacity constraints.

Our algorithms for the hard-capacitated versions of both vertex cover and set cover are based on rounding linear programming (LP) relaxations. In the following subsection, we outline the main reasons why the previous approaches fail and provide a sketch of our algorithms.

1.1 Our Approach and Contributions

The works of [4, 7] cannot handle the hard-capacitated vertex cover problem on multigraphs, neither do their approaches extend to hypergraphs or set systems with elements belonging to at most f sets. The algorithms in both of these works are based on LP rounding and involve three major steps. First, they pick all vertices with fractional values above a desired threshold. Next, a randomized rounding step is performed to choose some additional vertices. If even after step two, there are edges with unsatisfied fractional coverage, an alteration step is performed, in which vertices are chosen as long as all the edges are not fractionally fully covered maintaining the capacity constraints. Finally, the fractional edge assignment variables are rounded through a flow computation. While, the expected cost of selecting vertices in the first two steps can be easily bounded within a small factor of the optimal LP cost, the main crux of the argument relies in showing that with high probability the alteration cost can also be charged within a small factor of the cost incurred in the first two steps. When the graph does not contain any parallel edge, the random variables required to prove such a statement are all independent and thus strong concentration inequalities can be employed for the analysis.

However, the presence of parallel edges (or having hypergraphs) make these random variables *positively correlated*. This hinders the application of required concentration inequalities and the analysis breaks down.

We utilize the LP-structure to decompose the problem into two simpler instances. Instead of consolidating the variables corresponding to sets (vertices), we modify the variables associated with assignment of elements (edges) to sets (vertices). Viewing the LP solution as a bipartite graph between elements and sets, the graph is decomposed into a forest (H_1) and an additional subgraph (H_2) such that elements entirely covered by either one of these can be rounded without much loss in the approximation. There may be elements that are partially covered (fractionally) by sets in both H_1 and H_2. We further modify the remaining fractional solution to recast the capacitated covering problem on these unsatisfied elements as a multiset multicover (MM) problem *without* any capacity constraints.

We show that the partially rounded solution is feasible for the natural linear programming relaxation for MM. However the natural LP relaxation for MM has an unbounded integrality gap. Using a stronger LP relaxation, it is possible to give $\log n$-approximation algorithm for MM [14], but our fractional solution may not be feasible for such stronger relaxations. Moreover, a $\log n$ approximation for MM is not sufficient for our purpose. Instead, we show that it is possible to charge the cost of the obtained solution to a constant factor of LP cost for MM and the number of elements in the set system, and this suffices to ensure a constant approximation. Our algorithm for MM follows the paradigm of grouping and scaling used for *column restricted* (each set has same multiplicity for all elements) packing and covering problems [13]. However, our set system is not column restricted. We still can group the elements into *small* and *big* based on the extent of coverage these elements get from sets with relatively lower or higher multiplicities compared to their demands. By scaling the fractional variables and doing randomized rounding, we can satisfy the requirements of small elements, but big elements may still have residual demands left. Satisfying the requirements of big elements need a further step of careful rounding. Details are described in Section 2.2.

Our main contributions are as follows.

– We obtain an $O(1)$ approximation algorithm for the vertex cover problem with hard capacities on unweighted multigraphs for the unit multiplicity case, i.e., when all $m(v) = 1$.
– We show an $O(f)$-approximation algorithm for the unweighted set cover problem with hard capacities where each element belongs to at most f sets.
  As a corollary, we obtain an $O(1)$ approximation for the hard-capacitated vertex cover problem on unweighted multigraphs for arbitrary multiplicities.
– We consider partial covering problem with hard capacities. We give $O(1)$ approximation for partial vertex cover with hard capacities and $O(f)$ approximation for partial set cover problem with hard capacities.

In the following section, we describe a constant factor approximation algorithm for the hard-capacitated vertex cover problem on multigraphs with unit multiplicity ($m(v) =$

$1,\ \forall v \in \mathsf{V}(\mathsf{G})$). The algorithm and the analysis contain the main technical ingredients which are later used to obtain $O(f)$ approximation algorithms for the set cover and partial cover problems with hard capacities and arbitrary multiplicities. For lack of space, the latter two results appear in the full version of the paper.

2 Vertex Cover on Multigraphs with Hard Capacities

We start with the following linear programming relaxation for hard-capacitated vertex cover with unit multiplicities.

$$
\begin{aligned}
&\text{minimize } \sum_{v\in V} x(v) && && (\mathsf{LP}_{\mathsf{VC}})\\
&\text{subject to} && && \\
&\quad y(e,u)+y(e,v)=1 && \forall\, e=(u,v)\in E, && (1)\\
&\quad y(e,v)\le x(v),\ y(e,u)\le x(u) && \forall e=(u,v)\in E, && (2)\\
&\quad \sum_{e=(u,v)} y(e,v)\le k(v)x(v) && \forall v\in V, && (3)\\
&\quad 0\le x(v), y(e,v), y(e,u)\le 1 && \forall\, v\in V, \forall e=(u,v)\in E. && (4)
\end{aligned}
$$

Here $x(v)$ is an indicator variable, which is 1 if vertex v is chosen and 0 otherwise. Variables $y(e,u)$ and $y(e,v)$ are associated with edge $e=(u,v)$. $y(e,u)=1$ ($y(e,v)=1$) indicates edge e is assigned to vertex u (v). Constraints (1) ensure each edge is covered by at least one of its end-vertices. Constraints (2) imply an edge cannot be covered by a vertex v, if v is not chosen in the solution. The total number of edges covered by a vertex v is at most $k(v)$ if v is chosen and 0 otherwise (constraints (3)). We relax the variables $x(v), y(e,v)$ to take value in $[0,1]$ in order to obtain the desired LP-relaxation. The optimal solution of $\mathsf{LP}_{\mathsf{VC}}$ denoted by $\mathsf{LP}_{\mathsf{VC}}(\mathsf{OPT})$ clearly is a lower bound on the actual optimal cost OPT.

2.1 Rounding Algorithm

Let (x^*, y^*) denote an optimal fractional solution of $\mathsf{LP}_{\mathsf{VC}}$. We create a bipartite graph $\mathsf{H}=(\mathsf{A},\mathsf{B},\mathsf{E}(\mathsf{H}))$, where A represents the vertices of G, B represents the *edges* of G [3] and the links $\mathsf{E}(\mathsf{H})$ correspond to the (e,v) variables $e\in\mathsf{B}, v\in\mathsf{A}$ with non-zero y^* value [4]. Each $v\in\mathsf{A}(\mathsf{H})$ is assigned a weight of $x^*(v)$. Each link (e,v) is assigned a weight of $y^*(e,v)$. We now modify the link weights in a suitable manner to decompose the link sets of H into two graphs H_1 and H_2. Special structures of H_1 and H_2 make rounding relatively simpler on them.

– H_1 *is a forest*. For each node $v\in\mathsf{A}(\mathsf{H}_1)$ and link $(e,v)\in\mathsf{E}(\mathsf{H}_1)$, $y^*(e,v)<x^*(v)$.

[3] We often refer a vertex in $\mathsf{B}(\mathsf{H})$ by edge-vertex to indicate it belongs to $\mathsf{E}(\mathsf{G})$.

[4] in order to avoid confusion between edges of G with edges of H, we refer to edges of H by links

– *In* H_2*, if* $(e,v) \in \mathsf{E}(\mathsf{H}_2)$*, then weight of link* (e,v) *is equal to the weight of* v. Thus, for each node $v \in \mathsf{A}(\mathsf{H}_2)$ and link $(e,v) \in \mathsf{E}(\mathsf{H}_2)$, $y^*(e,v) = x^*(v)$.

  A moment's reflection shows the usefulness of such a property, essentially, in H_2, we can ignore the hard capacity constraints altogether.

The decomposition procedure is based on iteratively breaking cycles. We now explain the rounding algorithms on each of H_1 and H_2.

Rounding on H_2.

We discard all isolated vertices from H_2. Let $\eta \geq 2$ be the desired approximation factor. We select all vertices in $\mathsf{A}(\mathsf{H}_2)$ with value of x^* at least $\frac{1}{\eta}$. Let us denote the chosen vertices by $\mathcal{D}$. Then,

$$\mathcal{D} = \{v \mid v \in \mathsf{A}(\mathsf{H}_2), x^*(v) \geq \frac{1}{\eta}\}.$$

For every edge-vertex $e = (u,v) \in \mathsf{B}(\mathsf{H}_2)$, if v (or u) is in D, and $(e,v) \in \mathsf{E}(\mathsf{H}_2)$ (or $(e,u) \in \mathsf{E}(\mathsf{H}_2)$), then we set $y^*(e,v) = 1$ (or $y^*(e,u) = 1$). That is, we assign e to v, if the link (e,v) is in $\mathsf{E}(\mathsf{H}_2)$ and v is in $\mathcal{D}$, else if $u \in \mathcal{D}$ and $(e,u) \in \mathsf{E}(\mathsf{H}_2)$, the edge e is assigned to u.

Observation 1 *From constraints (3),* $\sum_{e=(u,v)} y(e,v) \leq x(v)k(v)$*. Therefore,* $\sum_{e=(u,v)} \frac{y(e,v)}{x(v)} \leq k(v)$*, and hence in* H_2*, after the assignment of edges to vertices in* $\mathcal{D}$*, all vertices maintain their capacity.*

In fact, in H_2, capacity constraints become irrelevant. *Whenever, we decide to pick a vertex in* $\mathsf{A}(\mathsf{H}_2)$*, we can immediately cover all the links in* $\mathsf{E}(\mathsf{H}_2)$ *incident on it.*

All edges with both links in $\mathsf{E}(\mathsf{H}_2)$ get covered at this stage. In addition, if $e \in \mathsf{B}(\mathsf{H}_2)$ has only one link $(e,v) \in \mathsf{E}(\mathsf{H}_2)$, but $x^*(v) = y^*(e,v) \geq \frac{1}{\eta}$, then since $v \in \mathcal{D}$, e gets covered. Therefore, the uncovered edges after this step either have no link in $\mathsf{E}(\mathsf{H}_2)$ or are fractionally covered to an extent less than $\frac{1}{\eta}$ in H_2.

Rounding on H_1.

H_1 is a forest; edge-vertices in H_1 either have both or one link in $\mathsf{E}(\mathsf{H}_1)$. While the vertices of H_1 and H_2 may overlap, the link sets are disjoint. Edge-vertices in $\mathsf{B}(\mathsf{H}_1)$ with only one link in H_1 are called *dangling* edges. We root H_1 arbitrarily to some node of $\mathsf{A}(\mathsf{H}_1)$. This naturally defines a parent-child relationship. Figure (1a) depicts the structure of H_1. Dangling edges are shown by dashed lines.

Rounding edges with both links in H_1.

Algorithm (1) describes the procedure to assign edge-vertices that have both links in $\mathsf{E}(\mathsf{H}_1)$.

We first select a collection of $\mathcal{D}'$ vertices from $\mathsf{A}(\mathsf{H}_1) \setminus \mathcal{D}$ with x^* value at least $\frac{1}{\eta}$. Any edge-vertex in $\mathsf{B}(\mathsf{H}_1)$ that has a child vertex chosen in $\mathcal{D}'$ gets assigned to its child. For

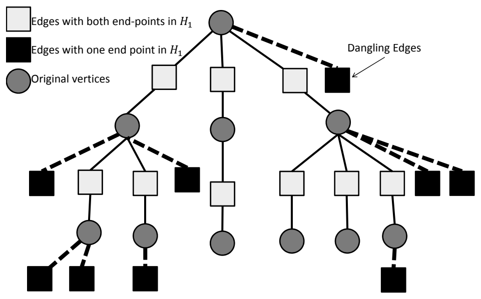
Edges with both end-points in H_1
Edges with one end point in H_1
Original vertices
Dangling Edges

Fig 1a. Structure of H_1, dangling edges are colored black and connected by dashed lines, edges with both end-points in H_1 are colored white and connected by solid lines.

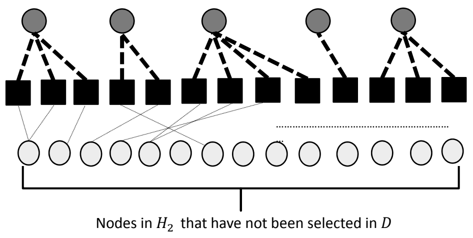

Fig 1b. Structure of H_1 after the edges with two end points in H_1 have been assigned.

Algorithm 1 Assigning edges with two links in H_1

```
1: let $\mathcal{D}' = \{v \in \mathsf{A}(\mathsf{H}_1) \mid x^*(v) \geq \frac{1}{\eta}\}$, select all the vertices in $\mathcal{D}'$.
2: for each edge-vertex $e$ with two links in $\mathsf{H}_1$ do
3:     if the child vertex of $e$ is selected in $\mathcal{D}'$ then
4:         assign $e$ to the selected child vertex.
5:     end if
6: end for
7: let $\mathsf{T}(v)$ denote the set of unassigned children edge-vertices incident on $v \in \mathsf{A}(\mathsf{H}_1)$ with
   both links in $\mathsf{H}_1$.
8: select any $t(v) = \lceil \sum_{e=(u,v)\in\mathsf{T}(v)} y^*(e,u) \rceil$ vertices from the children of the edge-vertices
   in $\mathsf{T}(v)$, and assign the corresponding $t(v)$ edge-vertices in $\mathsf{T}(v)$ to these selected children
   vertices. If $v'$ is a newly selected vertex in this step and there are edges that have links incident
   on $v'$ in $\mathsf{E}(\mathsf{H}_2)$, then assign those edges to $v'$ as well.
9: assign the remaining edge-vertices from $\mathsf{T}(v)$ to $v$.
```

each vertex $v \in \mathsf{A}(\mathsf{H}_1)$, we use $\mathsf{T}(v)$ to denote the set of children edge-vertices that are not assigned in step (4). We select $t(v) = \lceil \sum_{e=(u,v)\in\mathsf{T}(v)} y^*(e,u) \rceil$ vertices from the children of the edge-vertices in $\mathsf{T}(v)$. We assign the corresponding $t(v)$ edge-vertices in $\mathsf{T}(v)$ to these newly selected children vertices. Rest of the edges in $\mathsf{T}(v)$ are assigned to v.

Rounding dangling edges, i.e., with one link in H_1.

After Algorithm 1 finishes, let $\mathsf{L}(v)$ denote the set of unassigned dangling edge-vertices connected to v, and let $l(v) = \sum_{e=(u,v),e\in\mathsf{L}(v)} y^*(e,u)$. $\mathsf{L}(v)$ are the leaf edge-vertices of H_1. We first prove a lemma that shows after the edge-assignment in Algorithm 1, we still can safely assign at least $|\mathsf{L}(v)| - \lceil l(v) \rceil$ edges from $\mathsf{L}(v)$ to v without violating its capacity. We show the residual capacity of v after assigning edges from $E(\mathsf{H}_2)$ is at least as high as $1 + |\mathsf{T}(v)| - \lceil t(v) \rceil + |\mathsf{L}(v)| - \lceil l(v) \rceil$. The number of edges assigned to v from Algorithm 1 is at most $1 + |\mathsf{T}(v)| - \lceil t(v) \rceil$ and hence the following lemma is established.

Lemma 1. *Each vertex $v \in \mathsf{A}(\mathsf{H}_1)$ can be assigned $|\mathsf{L}(v)| - \lceil l(v) \rceil$ leaf edges-vertices without violating its capacity.*

The edge-vertices in $\mathsf{L}(v)$ are leaves of H_1, they are connected to v and have their other link in $\mathsf{E}(\mathsf{H}_2)$. We first pick *one vertex* from $\mathsf{A}(\mathsf{H}_2)$ such that it covers at least one edge from $\mathsf{L}(v)$. Let us denote this vertex by $h2(v)$ and let it cover $p2(v) \geq 1$ parallel edges $(v, h2(v))$. If $l(v) \leq p2(v)$, then following Lemma 1, the rest of the edge-vertices of $\mathsf{L}(v)$ can be assigned to v, and we do so.

If $l(v) > p2(v)$. Let $\mathsf{R}(v)$ denote the vertices of $\mathsf{A}(\mathsf{H}_2) \setminus h2(v)$ that are end-points of edges in $\mathsf{L}(v)$. If we pick enough vertices from $\mathsf{R}(v)$ such that they cover at least $l'(v) = l(v) - p2(v) + 1$ leaf-edges, then again from Lemma 1, rest of the edges from $\mathsf{L}(v)$ can be assigned to v.

We scale up all the x^* variables of $\bigcup_{v \in \mathsf{A}(\mathsf{H}_1)} \mathsf{R}(v)$ by a factor of $\frac{1}{1-\frac{1}{\eta}}$. We also scale up the corresponding y^* link variables by a factor of $\frac{1}{1-\frac{1}{\eta}}$. Let $(\bar{x}, \bar{y})$ denote the scaled up variables. Then, $\sum_{\substack{e=(u,v)\in \\ \mathsf{L}(v)\setminus(v,h2(v))}} \bar{y}(e,u) = \frac{(l(v)-p2(v)x^*(h2(v)))}{\left(1-\frac{1}{\eta}\right)} \geq \frac{\left(l(v)-\frac{p2(v)}{\eta}\right)}{\left(1-\frac{1}{\eta}\right)} > l(v) - p2(v) + 1 = l'(v)$, where the last inequality follows from the fact that $l(v) > p2(v) \geq 1$. We let $l'(v) = 0$, if $l(v) \leq p2(v)$. We now have the following multi-set multi-cover problem (MM).

For each $v \in \mathsf{A}(\mathsf{H}_1)$ with $l'(v) > 0$, we create an element $a(v)$. For each vertex $u \in \bigcup_{v \in \mathsf{A}(\mathsf{H}_1)} \mathsf{R}(v)$, we create a multi-set $S(u)$. If there are $d(v,u)$ leaf edge-vertices in $\mathsf{L}(v) \setminus (v, h2(v))$ incident upon u, then we include $a(v)$ in $S(u)$, $d(v,u)$ times . Each element $a(v)$ has a requirement of $r(a(v)) = \lfloor l'(v) \rfloor$. The goal is to pick minimum number of sets such that each element $a(v)$ is covered $\lfloor l'(v) \rfloor$ times counting multiplicities.

Note that, since the original graph is a multigraph, $d(v,u)$ can be greater than 1.

Lemma 2. *If we set $z(S(u)) = \bar{x_u}, \forall u \in \bigcup_{v \in \mathsf{A}(\mathsf{H}_1)} \mathsf{R}(v)$, then $\mathbf{z}$ is a feasible fractional solution for the above stated multi-set multi-cover problem.*

As described in Section 1.1, existing approaches are not sufficient to obtain an integral solution for the above MM problem that will ensure a constant approximation. We instead, obtain an algorithm where the total number of sets picked is close to $s + \sum_{u \in \bigcup_{v \in \mathsf{A}(\mathsf{H}_1)} \mathsf{R}(v)} \bar{x}_u$, where s is the number of vertices in $\mathsf{A}(\mathsf{H}_1)$ with $l'(v) > 0$. In Section 2.2, we prove the following theorem.

Theorem 3. *Given any feasible fractional solution $\bar{x}$ with cost F for multi-set multi-cover problem with N elements, there is a polynomial time randomized rounding algorithm that rounds the fractional solution to a feasible integral solution with expected cost at most $21N + 32F$.*

The algorithm for assigning the leaf edge-vertices in $\mathsf{L}(v)$ is given in Algorithm (2).

Since, each vertex $v \in \mathsf{A}(\mathsf{H}_1)$ covers at most $|\mathsf{L}(v)| - \lceil l(v) \rceil$ leaf edge-vertices, by Lemma 1 the capacity of all the vertices in H_1 are maintained. We now proceed to analyze the cost.

Algorithm 2 Assigning edges with only one link in H_1

```
1:  for each vertex $v \in \mathsf{A}(\mathsf{H}_1)$ with $|\mathsf{L}(v)| \geq 1$ do
2:      select the vertex $h2(v)$ that covers at least one edge-vertex from $\mathsf{L}(v)$ and assign the corresponding edge-vertices to $h2(v)$.
3:  end for
4:  for each vertex $v \in \mathsf{A}(\mathsf{H}_1)$ with $l(v) \leq p2(v)$ do
5:      assign all the remaining edge-vertices (at most $|\mathsf{L}(v)| - \lceil l(v) \rceil$) to $v$
6:  end for
7:  for each vertex $v \in \mathsf{A}(\mathsf{H}_1)$ with $l'(v) > 1$ do
8:      scale up the $\mathbf{x}^*$ variables in $\bigcup_{v \in \mathsf{A}(\mathsf{H}_1)} \mathsf{R}(v)$ by a factor of $\frac{1}{1-\frac{1}{\eta}}$ and denote it by $\bar{\mathbf{x}}$.
9:  end for
10: create the MM instance $(\{(a(v), d(v))\}, \{S(u)\})$, and round the fractional solution $\bar{\mathbf{x}}$ to obtain an integral solution.
11: for each $u$ such that $S(u)$ is chosen by MM algorithm do
12:     select $u$ and assign all the leaf-edges incident on $u$ to it.
13: end for
14: for each $v \in \mathsf{A}(\mathsf{H}_1)$ with $l'(v) > 1$ do
15:     assign all the remaining leaf edge-vertices of $\mathsf{L}(v)$ (at most $|\mathsf{L}(v)| - \lceil l(v) \rceil$) to it.
16: end for
```

Theorem 2. *There exists a polynomial time algorithm achieving an approximation factor of* 34 *for the hard-capacitated vertex cover problem with unit multiplicity on unweighted multigraphs.*

2.2 Proof of Theorem 3

In the multi-set multi-cover problem (MM), we are given a ground set of N elements U and a collection of multi-sets $\mathcal{S}$ of U, $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$. Each multi-set $S \in \mathcal{S}$ contains $M(S, e)$ copies of element $a \in U$. Each element a has a demand of $r(a)$ and needs to be covered $r(a)$ times. The objective is to minimize the number of chosen sets that satisfy the demands of all the elements. Here we propose a new algorithm that proves Theorem 3.

The following is a linear program relaxation for MM.

$$\min \sum_{S \in \mathcal{S}} x(S)$$

$$\sum_{a \in S} M(a, S) x(S) \geq r(a) \qquad \forall\, a \in U$$

$$0 \leq x(S) \leq 1 \qquad \forall\, S \in \mathcal{S}$$

2.3 Rounding Algorithm for MM

Let $\mathbf{x}^*$ denote the LP optimal solution. The rounding algorithm has several steps.

Step 1. Selecting sets with high fractional value. First, we pick all sets $S \in \mathcal{S}$ such that $x^*(S) \geq \alpha > 0$, where $\frac{1}{\alpha}$ is the desired approximation factor. Denote the chosen sets by $\mathcal{H}$. Each element a now has a residual requirement of $r(a) - \sum_{a \in S, S \in \mathcal{H}} M(S,a)$. Clearly the fractional solution x^* projected on the sets $\mathcal{S} \setminus \mathcal{H}$ is a feasible solution for the residual problem. For each element $a \in U$, let $\bar{r}(a) = r(a) - \sum_{a \in S, S \in \mathcal{H}} M(S,a)$ be the residual requirement. For some $\beta > 0$ (to be set later), let $y(S) = \beta x^*(S)$, for each $S \in \mathcal{S} \setminus \mathcal{H}$. We have for all elements $a \in U$, $\sum_{a \in S, S \in \mathcal{S} \setminus \mathcal{H}} M(S,a) y(S) \geq \beta \bar{r}(a)$.

Note that after this step, we have a fractional solution with cost

$$|H| + \sum_{S \in \mathcal{S} \setminus \mathcal{H}} y(S) \leq \frac{1}{\alpha} \sum_{S \in \mathcal{H}} x^*(S) + \beta \sum_{S \in \mathcal{S} \setminus \mathcal{H}} x^*(S).$$

For notational simplicity, we denote $\mathcal{C} = \mathcal{S} \setminus \mathcal{H}$. Next, we proceed to round the variables $y(S)$ for $S \in \mathcal{C}$.

Step 2. Rounding into powers of 2. For each multiplicity $M(S,a)$, $\forall S \in \mathcal{C}, a \in U$, we round it to the highest power of 2 lesser than or equal to $M(S,a)$ and denote it by $M^1(S,a)$. For each requirement $\bar{r}(a)$, $\forall a \in U$, consider the lowest power of 2 greater than or equal to $\bar{r}(a)$ and denote it by $\bar{r}^1(a)$. Clearly, if $\sum_{a \in S, S \in \mathcal{C}} M(S,a) y(S) \geq \beta \bar{r}(a)$, then $\sum_{a \in S, S \in \mathcal{C}} M^1(S,a) 4y(S) \geq \beta \bar{r}^1(a)$. We denote $\mathbf{y}^1 = 4\mathbf{y}$.

Step 3. Division into small and big elements. First, for each element if there is a set that completely satisfies its requirement, we pick the set. We continue the process as long as no more element can be covered entirely by a single set. Thus after this procedure, for all elements a, and for all sets S, $M^1(S,a) < \bar{r}^1(a)$ and hence $M^1(S,a) \leq \frac{\bar{r}^1(a)}{2}$. Now for each element a, we divide the sets in $\mathcal{C}$ containing a into *big* sets ($Big(a)$) and *small* sets ($Small(a)$). A set $S \in \mathcal{C}$ is said to be a big set for a, if $M^1(S,a) \geq \frac{1}{18 \ln n} \bar{r}^1(a)$, otherwise it is called a small set, i.e.,

$$Big(a) = \{S \in \mathcal{C} \,| M^1(S,a) \geq \frac{1}{18 \ln n} \bar{r}^1(a)\}$$

$$Small(a) = \{S \in \mathcal{C} \,| M^1(S,a) < \frac{1}{18 \ln n} \bar{r}^1(a)\}$$

Now, we decompose elements into *big* and *small*. An element is *small* if it is covered to an extent of $\bar{r}^1(a)$ by the sets in $Small(a)$. Else, the element is covered at least to an extent of $(\beta - 1)\bar{r}^1(a)$ by the sets in $Big(a)$ and we call it a *big* element. This follows from the inequality

$$\sum_{a \in S, S \in \mathcal{C} \cap Big(a)} M^1(S,a) y^1(S) \qquad + \sum_{a \in S, S \in \mathcal{C} \cap Small(a)} M^1(S,a) y^1(S) \geq \beta \bar{r}^1(a).$$

Therefore, either the sets in $Small(a)$ cover a to an extent of $\bar{r}^1(a)$, or the sets in $Big(a)$ cover a to an extent of $(\beta - 1)\bar{r}^1(a)$. Let $\beta_1 = \beta - 1$. In the first case, we refer a as a small element, otherwise it is a big element.

Step 4. Covering small elements. We employ simple independent randomized rounding for covering small elements. *We pick each set $S \in \mathcal{C}$ with probability γy^1_S, for some $\gamma \geq 2$.*

Lemma 3. *All small elements are covered in Step 4 with probability at least $\left(1 - \frac{1}{n^{1/3}}\right)$.*

Step 5. Covering big elements. This is the most crucial ingredient in the algorithm. For each big element, we consider only the big sets containing it. For each such big element and big set we have $\frac{1}{18 \ln n} r^1_a < M^1(S,a) \leq \frac{r^1_a}{2}$. Since, multiplicities are powers of 2, there are at most $l = \ln \ln n + 3$ different values of multiplicities of the sets for each element a.

Let $T^a_1, T^a_2, \ldots T^a_l$ denote the collection of these sets with multiplicities $\frac{\bar{r}^1(a)}{2}, \frac{\bar{r}^1(a)}{2^2}, \ldots, \frac{\bar{r}^1(a)}{2^l}$ respectively. That is, $T^a_i = \{S \in Big(a) \mid M(S,a) = \frac{\bar{r}^1(a)}{2^i}\}$. Set $\beta_1 \geq 3$.

For each $i = 1, 2, \ldots, l$, if $\sum_{S \in T^a_i} y^1(S) > i$ and the number of sets that have been picked from T^a_i in Step 4 is less than $\frac{\sum_{S \in T^a_i} y^1(S)}{(\beta_1 - 2)}$, pick new sets from T^a_i such that the total number of chosen sets from T^a_i is $\left\lceil \frac{\sum_{S \in T^a_i} y^1(S)}{(\beta_1 - 2)} \right\rceil$.

We now show that each big element gets covered the required number of times and the total cost is bounded by a constant factor of the optimal cost.

Lemma 4. *Each big element a is covered $r(a)$ times by the chosen sets.*

Lemma 5. *The expected number of sets selected in Step 4 is at most 21n', where n' are the number of big elements that are not covered after Step 5.*

Theorem 3. *The algorithm returns a solution with expected cost at most $21N + 32F$, where $F = \sum_S x^*(S)$, and covers all the elements with probability at least $1 - \frac{1}{n^{1/3}}$.*

This completes the description of the $O(1)$ approximation algorithm for hard-capacitated vertex cover problem on multigraphs with unit multiplicities. We have not tried to optimize the constants of our approach, but reducing the approximation ratio to 2 or 3 may require significant new ideas. Theorem 3 is also crucially used to obtain an $O(f)$-approximation algorithm for the set cover and partial cover problem with arbitrary multiplicities. The results for set cover and partial cover problem appear in the full version of the paper.

References

1. Judit Bar-Ilan, Guy Kortsarz, and David Peleg. Generalized submodular cover problems and applications. *Theor. Comput. Sci.*, 250:179–200, January 2001.
2. R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–45, 1985.
3. Reuven Bar-Yehuda, Guy Flysher, Julián Mestre, and Dror Rawitz. Approximation of partial capacitated vertex cover. In *ESA*, pages 335–346, 2007.
4. Julia Chuzhoy and Joseph (Seffi) Naor. Covering problems with hard capacities. *SIAM J. Comput.*, 36(2):498–515, 2006.
5. Julia Chuzhoy and Yuval Rabani. Approximating k-median with non-uniform capacities. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '05, pages 952–958, 2005.
6. Erik D. Demaine and Morteza Zadimoghaddam. Scheduling to minimize power consumption using submodular functions. In *Proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures*, SPAA '10, pages 21–29, 2010.
7. Rajiv Gandhi, Eran Halperin, Samir Khuller, Guy Kortsarz, and Aravind Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. *J. Comput. Syst. Sci.*, 72:16–33, February 2006.
8. Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.
9. Sudipto Guha, Refael Hassin, Samir Khuller, and Einat Or. Capacitated vertex covering. *Journal of Algorithms*, 48(1):257 – 270, 2003.
10. Dorit S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *Siam Journal on Computing*, 11:555–556, 1982.
11. David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9:256–278, 1974.
12. Samir Khuller, Jian Li, and Barna Saha. Energy efficient scheduling via partial shutdown. In *SODA*, pages 1360–1372, 2010.
13. Stavros G. Kolliopoulos. Approximating covering integer programs with multiplicity constraints. *Discrete Appl. Math.*, 129:461–473, 2003.
14. Stavros G. Kolliopoulos and Neal E. Young. Tight approximation results for general covering integer programs. In *IEEE Symposium on Foundations of Computer Science*, pages 522–528, 2001.
15. László Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13(4):383 – 390, 1975.
16. Mohammad Mahdian and Martin Pal. Universal facility location. In *in Proc. of European Symposium of Algorithms 03*, pages 409–421, 2003.
17. Julián Mestre. A primal-dual approximation algorithm for partial vertex cover: Making educated guesses. In *APPROX-RANDOM*, pages 182–191, 2005.
18. Laurence A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982.