A $(\Delta_d + 2)$-Approximation for Unweighted Capacitated Domination with Hard Capacities (Extended Abstract)

Mong-Jen Kao[1], Hai-Lun Tu[2], and D.T. Lee[1,2]

[1]Institute for Information Science, Academia Sinica, Taipei, Taiwan.
mong@iis.sinica.edu.tw, dtlee@ieee.org
[2]Department of Computer Science and Information Engineering, National Taiwan University, Taiwan. d95019@csie.ntu.edu.tw

Abstract

In this paper we consider the capacitated domination problem with hard capacity constraints (CD-HC), a generic model to covering problems, for *unweighted* versions .

We obtain a $(\Delta_d + 2)$-approximation for general graphs, where Δ_d is the maximum number of demanding vertices any covering vertex has in its closed neighborhood (the demanding degree).

1 Introduction

In this paper, we consider the capacitated domination problem with hard capacities (CD-HC), a generic model to covering problems with hard capacities, and provide approximation results as well as trade-offs. In this problem we are given a graph $G = (V, E)$ with four parameters, namely *weight* $w(v)$, *demand* $d(v)$, *capacity* $c(v)$, and *available multiplicities* $m(v)$, defined for each vertex $v \in V$. The demand of a vertex is the amount of service it requires. The capacity of a vertex is the amount of service each multiplicity of that vertex can provide to its closed neighbors. The objective is to compute a dominating multi-set of minimum weight such that the multiplicity of each vertex in the multi-set does not exceed its available multiplicities and the demand of each vertex is served by the capacities provided by its closed neighbors.

1.1 Previous Work

Depending on whether the available multiplicities of each object is limited, the work on capacitated covering problems falls mainly in two categories: (1) *soft capacities*, where we can use as many multiplicities as necessary, and (2) *hard capacities*, where the available multiplicities of each vertex is limited.

The capacitated vertex cover with soft capacities was first introduced by Guha et al. [7]. They gave a 2-approximation based on primal-dual schema. Subsequently, Gandhi et al. [6] proposed another 2-approximation via dependent rounding. Kao et al. [8, 9] considered capacitated domination problem and proposed a Δ^*-approximation algorithm, where Δ^* is the maximum closed degree of the graph. Special cases and variations of this problem were also considered independently [3, 4, 10].

For hard capacities, Chuzhoy and Naor [2] considered capacitated covering with hard capacities and unit demands. For unweighted capacitated vertex cover with hard capacities (VCHC), they gave a 3-approximation using randomized rounding with a specific patching procedure. They showed that the weighted version of VCHC is at least as hard as the set cover problem. Due to this reason, subsequent work on VCHC has focused on the unweighted version. For weighted capacitated set cover with unit demand, they presented a $(\ln \Delta_d + 1)$-approximation, where Δ_d is the maximum size of the sets. This approach further extends to a $(\ln \max_S f(S) + 1)$-approximation for submodular set cover, which was proved by Wolsey [12]. Gandhi et al. [5] proposed a refined approach to [2] for unweighted VCHC with unit edge demand and obtained a 2-approximation.

Saha and Khuller [11] considered unweighted

VCHC with general edge demands. They provided a 34-approximation for the case of unit multiplicity and a 38-approximation for general multiplicities. They also considered Δ_c-hypergraphs, i.e., hypergraphs with largest edge size Δ_c, and proposed a $\max\{6 \cdot \Delta_c, 65\}$-approximation. Recently, these results were improved by Cheung et al. [1], who provided a $(1 + 2/\sqrt{3})$-approximation for general graphs and a $2 \cdot \Delta_c$-approximation for Δ_c-hypergraphs.

1.2 Our Results and Contribution

In this paper, we consider the problem of capacitated domination with hard capacities and present algorithmic results for unweighted versions of this problem. The approach we use is a delicate primal-dual schema, extended from [9], combined with a flow-based procedure and a local charging argument.

For the unweighted CD-HC, we present a $(\Delta_d + 2)$-approximation algorithm for general graphs, where Δ_d is the maximum demanding degree of any covering vertices, i.e., the maximum number of demanding vertices any covering vertex has in its closed neighborhood.

The rest of this paper is organized as follows. In §2 we formally introduce the problem we consider and the primal-/dual- linear programs. In §3 we consider unweighted- CD-HC, and in §4 we conclude with an overview and a discussion on future directions.

2 Preliminaries

Let $G = (V, E)$ be a graph with vertex set V and edge set E. We denote the set of neighbors of a vertex $v \in V$ by $N_G(v)$. Formally, $N_G(v) = \{u : (u, v) \in E\}$. The *closed neighborhood* of any vertex $v \in V$, i.e., $N_G(v) \cup \{v\}$, is denoted by $N_G[v]$. For any $A \subseteq V$, we use $N_G[A]$ to denote the closed neighbors of the vertices in A, i.e., $N_G[A] = \bigcup_{u \in A} N_G[u]$. For the rest of this paper, the subscript G will be omitted when there is no confusion in the context.

2.1 Capacitated Domination with Hard Capacities (CD-HC)

Below we formally define the problem we consider. In this problem we are given a graph $G = (V, E)$ with four non-negative integral parameters defined for each $v \in V$: (1) weight $w(v)$, (2) demand $d(v)$, (3) capacity $c(v)$, and (4) available multiplicities $m(v)$.

By a demand assignment function f we mean a function that maps pairs of vertices to non-negative real numbers, i.e., $f\colon V \times V \to \mathbb{R}^+ \cup \{0\}$. In other words, $f(u, v)$ denotes the amount of demand of u that is assigned to v.

For any demand assignment function f, the corresponding multiplicity function, denoted x_f, is defined to be $x_f(v) = \left\lceil \sum_{u \in N_G[v]} f(u, v)/c(v) \right\rceil$. Literally, x_f gives the dominating multi-set to which f corresponds in that it specifies the number of times a vertex has to be included in the multi-set.

A demand assignment function f is said to be *feasible* if we have $\sum_{v \in N[u]} f(u, v) \geq d(u)$ and $x_f(u) \leq m(u)$ for each $u \in V$. In other words, a demand assignment is feasible if the demand of each vertex is fully-assigned to some of its closed neighbors (fully-served) and the multiplicity of each vertex does not exceed its available multiplicities.

Lemma 1. Let $G = (V, E)$ be an instance of CD-HC and $\mathcal{D}$ be a valid dominating multi-set for G, i.e., there exists a feasible demand assignment whose corresponding dominating multi-set is $\mathcal{D}$. Then there exists a feasible demand assignment $f\colon V \times V \to \mathbb{N} \cup \{0\}$, computable in polynomial time, such that we have $x_f(v) \leq x_{\mathcal{D}}(v)$ for every $v \in V$, where $x_{\mathcal{D}}$ is the multiplicity function for $\mathcal{D}$.

The cost of a demand assignment function f, denoted $w(f)$, is defined to be $w(f) = \sum_{u \in V} w(u) \cdot x_f(u)$. Below we formally define the problem we consider.

Definition 1 (Capacitated Domination with Hard Capacities)**.** Given a graph $G = (V, E)$ with weight $w(v)$, demand $d(v)$, capacity $c(v)$, and available multiplicities $m(v)$ defined for each $v \in V$, the problem of capacitated domination with hard capacities is to compute a feasible demand assignment function f such that $w(f)$ is minimized.

Throughout this paper, for each $v \in V$, we use $\delta_d(v)$ and $\delta_c(v)$ to denote the number of demanding vertices and the number of covering vertices in $N[v]$, respectively. Formally, $\delta_d(v) = |\{u\colon u \in N[v], d(u) > 0\}|$ and $\delta_c(v) = |\{u\colon u \in N[v], m(u) \cdot c(u) > 0\}|$. We also refer $\delta_d(v)$ and $\delta_c(v)$ to as the demanding degree

and the covering degree of v, respectively. We use Δ_d to denote the maximum demanding degree of vertices with nonzero available capacity in the input graph, i.e., we have $\Delta_d = \max_{v\in V, m(v)\cdot c(v)>0} \delta_d(v)$. Similarly, we use Δ_c to denote the maximum covering degree of demanding vertices, i.e., $\Delta_c = \max_{v\in V, d(v)>0} \delta_c(v)$.

Moreover, for the rest of this paper, we implicitly assume the feasibility of the input graph, i.e., a feasible demand assignment for the input always exists. Note that, this condition can be checked easily by Lemma 1 via a max-flow computation.

2.2 A ILP Formulation and the Dual of its Relaxation

An integer linear program for the CD-HC is given below in (1). There are two sets of variables $f(u, v)$ and $x(u)$, which correspond to the demand assignment function and the multiplicities of the vertices.

$$\begin{aligned}
\text{Minimize} \quad & \sum_{u\in V} w(u)\cdot x(u) \\
& \sum_{v\in N[u]} f(u,v) \geq d(u), && u \in V \\
& c(u)x(u) - \sum_{v\in N[u]} f(v,u) \geq 0, && u \in V \\
& d(v)x(u) - f(v,u) \geq 0, && v \in N[u],\ u \in V \\
& x(u) \leq m(u), && u \in V \\
& f(u,v), x(u) \in \mathbb{Z}^+ \cup \{0\}, && u, v \in V \quad (1)
\end{aligned}$$

The first inequality states that the demand of each vertex has to be fully-served. The second inequality connects the multiplicity function and the demand assignment function. The third inequality, which states that the multiplicity of a vertex cannot be zero if some demand is assigned to that vertex, is introduced to bound the integrality gap of the relaxation. The fourth inequality constraints the multiplicity of each vertex.

The dual linear program of the relaxation of (1) is given in (2). There are four sets of variables y_u, z_u, $g_{u,v}$, and η_u, which can be interpreted as a packing program as follows. We would like to pack more values into the variable y_u for all $u \in V$, whose values are constrained by z_v and $g_{v,u}$ that are further constrained by $w(v)$ for each $v \in N[u]$.

$$\begin{aligned}
\text{Maximize} \quad & \sum_{u\in V} d(u)y_u - \sum_{u\in V} m(u)\eta_u \\
\text{subject to} \quad & \\
& c(u)z_u + \sum_{v\in N[u]} d(v)g_{u,v} - \eta_u \leq w(u), && u \in V \\
& y_u \leq z_v + g_{v,u}, && v \in N[u],\ u \in V \\
& y_u \geq 0,\ z_u \geq 0,\ g_{v,u} \geq 0,\ \eta_u \geq 0, && v \in N[u],\ u \in V \quad (2)
\end{aligned}$$

However, the fourth set of variables, η_u, complicates the structure of the packing program in that it allows us to pack even more values into y_u in the cost of a deduction in the objective value. This provides a certain degree of flexibility, and handling this flexibility would be one of the major challenges for this problem.

3 A (Δ_d+2)-Approximation for Unweighted CD-HC

In this section we consider the unweighted case, i.e., $w(v) = 1$ for all $v \in V$, and present a (Δ_d+2)-approximation for the unweighted CD-HC. First, we describe an approach to obtaining a feasible solution for the dual program (2). In order to cope with the effect of the non-positive contribution of the η_u variables in the objective value, we use a flow-based procedure to help deal with the pending decisions. During this procedure, a feasible solution for the primal ILP (1) is also obtained. This primal-dual scheme is presented in §3.1

3.1 Primal-Dual Scheme

We describe a process for computing a feasible dual solution $\Psi = (y_u, z_u, g_{u,v}, \eta_u)$ for the dual program (2). The process starts with a trivial solution and eventually reaches a local optimal point. During the process, the values of some dual variables will be raised and some inequalities will meet with equality.

We first define some terms and notations to help present our algorithm. We say a vertex u is *saturated* if $c(u)z_u + \sum_{v\in N[u]} d(v)g_{u,v} - \eta_u = w(u)$ and *opened* if some demand from its closed neighborhood is assigned to it. During the algorithm we will maintain three vertex subsets U, V^ϕ, and S, which consist of the following: U contains the set of vertices whose demand is not yet served, V^ϕ

contains the set of vertices that have not yet been saturated, and S contains the set of vertices that are currently saturated but not yet opened. For each $u \in V$, we use $d^{\phi}(u) = \sum_{v \in N[u] \cap U} d(v)$ to denote the amount of unassigned demand from the closed neighbors of u.

Below we describe our algorithm in more detail. Initially, $U \equiv \{u : u \in V, d(u) > 0\}$ and all the dual variables are set to be zero. We increase the dual variable y_u simultaneously for each $u \in U$. To maintain the dual feasibility, as we increase y_u, we have to raise either z_v or $g_{v,u}$ for each $v \in N[u]$. If $d^{\phi}(v) \leq c(v)$, then we raise $g_{v,u}$. Otherwise, we raise z_v. In addition, if $v \in S$, i.e., v is currently saturated but not yet opened, then we also raise η_v to the extent such that it remains saturated. As soon as one of the vertices in V^{ϕ}, say, u, becomes saturated, we perform the following operations.

We use a recursive procedure Update($S \cup \{u\}, u$), described in the next paragraph, to compute a pair (S', f'), where S' is a maximal subset of $S \cup \{u\}$ that can fully-serve all the demand from $N[S'] \cap U$ and f' is the corresponding demand assignment function. If $S' = \emptyset$, then we add u to S. Otherwise, we assign the demand from $N[S'] \cap U$ to S' according to f'. Then we remove S' from S and each vertex $v \in N[S'] \cap U$ from U. This process is repeated until $U = \emptyset$.

The procedure Update(A, u)**.** Below we describe the recursive procedure Update(A, u). For any vertex subset $A \subseteq V$, we define a directed flow-graph $\mathcal{G}(A)$ with a source s^+ and a sink s^- as follows. Excluding s^+ and s^-, $\mathcal{G}(A)$ is a bipartite graph induced by $N[A] \cap U$ and A. In particular, for each $v \in N[A] \cap U$, we have a vertex v^+ and an edge (s^+, v^+) in $\mathcal{G}$. Similarly, we have a vertex v^- and an edge (v^-, s^-) for each $v \in A$. Finally, for each $v_1 \in A$ and each $v_2 \in N[v_1] \cap U$, we have an edge (v_2^+, v_1^-) in $\mathcal{G}$.

The capacity of each edge is defined as follows. For each $v \in N[A] \cap U$, $c(s^+, v^+)$ is set to be $d(v)$. For each $v \in A$, $c(v^-, s^-)$ is set to be $m(v) \cdot c(v)$. The capacities of the remaining edges are unlimited.

If $u \in A$, then we compute the max-flow of $\mathcal{G}(A)$ under the additional constraint that the flow between u^- and s^- is as small as possible, i.e., whenever there are multiple choices, the flow tends to go through v^- for any $v \in A \backslash \{u\}$ instead of u^-. Note that this can be done by augmenting the procedure of computing augmenting paths. If $u \notin A$, then we do not have this constraint and we simply compute the max-flow of $\mathcal{G}(A)$. Let $\tilde{f}$ denote the resulting flow function and $\tilde{f}(v_1, v_2)$ denotes the flow from vertex v_1 to v_2 in $\mathcal{G}$. Let

$$S' = \big\{v_1 \colon v_1 \in A, \tilde{f}(s^+, v_2^+) = d(v_2) \text{ for all } v_2 \in N[v_1] \cap U\big\}$$

be the subset of A that can fully-serve the demand from $N[S'] \cap U$. If $S' = A$ or $S' = \emptyset$, then we return $(S', \tilde{f}')$, where $\tilde{f}'$ is the demand assignment function from $N[S'] \cap U$ to S' induced by $\tilde{f}$. Otherwise we return Update(S', u).

3.2 Feasibility of Our Algorithm

In the following we present prilimary analysis for the approach. First, we show that our Primal-Dual Scheme computes a feasible demand assignment if the input graph G admits one. It is not difficult to see that, the assignment the algorithm performs in each iteration, i.e., the function $\tilde{f}'$ computed by procedure Update($S \cup \{u\}, u$) which corresponds to a demand assignment from $N[S'] \cap U$ to S', is always valid. Hence, it suffices to argue that the set U becomes empty after some iterations. To this end, we prove the following property for the procedure Update($S \cup \{u\}, u$).

Lemma 2. If there exists any $B \subseteq S \cup \{u\}$ such that B can fully-serve the demand in $N[B] \cap U$, then we have $B \subseteq S'$, where S' is the set returned by Update($S \cup \{u\}, u$).

We prove this by arguing that $B \subseteq S_i$ implies that $B \subseteq S_{i+1}$ for all $1 \leq i < k$, where S_i denotes the input of the procedure Update($\cdot$) in ith recursion.

Lemma 2 ensures that, at the end of each iteration, $\nexists B \subseteq S$ such that B can fully-serve $N[B] \cap U$. Hence, in all iterations, we have $u \in S'$ whenever $S' \neq \emptyset$, where u is the vertex to saturate in that iteration and S' is the set returned by Update($S \cup \{u\}, u$). Furthermore, the procedure Update($\cdot$) guarantees that, after a vertex is opened, all the demand from its closed neighborhood will be served.

Hence, if $S \neq \emptyset$, then the demand in $N[S] \cap U$ is not yet served and we know that none of the vertices in $N[N[S] \cap U]$ is opened, meaning that no vertex in $N[N[S] \cap U] \backslash S$ has been saturated. If none of the vertices in $N[N[S] \cap U] \backslash S$ can further saturate in later iterations, i.e., $(N[N[S] \cap U] \backslash S) \cap V^{\phi} = \emptyset$, then we have found a proof that the input graph is infeasible. In other

words, $(N[N[S] \cap U] \backslash S) \cap V^{\phi} \neq \emptyset$ as long as the input graph is feasible. Similar argument holds for the set U.

Since the cardinality of V^{ϕ} strictly decreases in each iteration, we know that both S and U will eventually become empty if the input graph is feasible. This proves the feasibility of our algorithm.

3.3 Approxmation Guarantee

Below we bound the cost of the solution computed by Algorithm Unweighted-CD-HC. The idea is to distribute the cost we spend for each multiplicity to the value of the dual solution we obtained and the cost of an arbitrary optimal demand assignment. For vertices whose cost can be paid by the dual solution, we charge the demands served by the multiplicities. (The demands pay for the services they use.) For those whose cost cannot be paid properly by the dual solution, we bound their cost using a local argument and the assumption that vertices are unweighted. Together this gives a factor of $\Delta_d + 2$.

Let $\hat{f}$ denote the demand assignment computed by our algorithm and $x_{\hat{f}}$ be the corresponding multiplicity function. For each $u \in V$, we use $d^*(u) = \sum_{v \in N[u]} \hat{f}(v, u)$ to denote the amount of demand that is assigned to u in $\hat{f}$.

For each $u \in V$ with $x_{\hat{f}}(u) > 0$, we distinguish the following three cases:

(i) $u \in S$ during some iteration,
(ii) $u \notin S$ for all iterations and $d^*(u) \geq c(u)$, and
(iii) $u \notin S$ for all iterations but $d^*(u) < c(u)$.

Let V_1, V_2, and V_3 denote the set of vertices that fall into the above three cases, respectively. Clearly, we have $w(\hat{f}) = \sum_{1 \leq i \leq 3} \sum_{u \in V_i} x_{\hat{f}}(u)$. Below we consider the three cases separately.

Lemma 3. For any $u \in V_1$, we have $d^*(u) = m(u) \cdot c(u)$. Furthermore, $w(u) \cdot m(u) = d^*(u) \cdot y_v - m(u) \cdot \eta_u$ for each $v \in N[u]$ such that $\hat{f}(v, u) > 0$.

We prove the first half of this lemma by contradiction.

Consider the specific iteration for which the vertex u was removed from S and let u_0 be the vertex that becomes saturated in that iteration. We argue that it exits an alternating path to which we can reroute the flow from u_0 to u if $d^*(u) < m(u) \cdot c(u)$.

And since $d^*(u) = m(u) \cdot c(u)$, we know that $d^{\phi}(u) \geq c(u)$ before u gets saturated. So only z_u will be raised, and $y_v = z_u$ for all $v \in N[u]$ such that $\hat{f}(v, u) > 0$. It yields the second half of this lemma.

Lemma 4. For any $u \in V_2$ and $v \in N[u]$ such that $\hat{f}(v, u) > 0$, we have $w(u) \cdot x_{\hat{f}}(u) \leq 2 \cdot d^*(u) \cdot y_v$.

Proof of Lemma 4. Since $u \notin S$ in all iterations, we have $\eta_u = 0$. Since $d^*(u) \geq c(u)$, we know that $d^{\phi}(u) \geq c(u)$ until saturated. Hence we have $w(u) = c(u) \cdot z_u$ and $y_v = z_u$ for all $v \in N[u]$ by our scheme. Therefore $w(u) \cdot x_{\hat{f}}(u) = c(u) \cdot z_u \cdot \left\lceil \frac{d^*(u)}{c(u)} \right\rceil \leq 2 \cdot d^*(u) \cdot y_v$. □

Let f_{OPT} denote an optimal demand assignment for the input graph and $x_{f_{\mathrm{OPT}}}$ denote the corresponding multiplicity function. By charging the cost incurred by vertices in V_3 to the cost incurred by f_{OPT}, we have the following lemma.

Lemma 5. We have $\sum_{u \in V_3} x_{\hat{f}}(u) \leq \Delta_d \cdot w(f_{\mathrm{OPT}})$.

For any $v \in N[u]$ such that $\hat{f}(v, u) > 0$, we know that $f_{\mathrm{OPT}}(v, v') > 0$ for some $v' \in N[v]$. Since $x_{\hat{f}}(u) = 1$, we charge the cost incurred by u in our solution to the cost incurred by v' in the optimal solution.

According to the design of our algorithm, we know that the demand of v is assigned in one particular iteration, and hence it can only be assigned to at most one vertex in V_3 since at most one vertex could be classified into V_3 in any iteration. Therefore v' can be charged by at most $\delta_d(v') \leq \Delta_d$ times.

By combining the upper-bounds we obtained for the three sets V_1, V_2, and V_3, and the discussion in §3.2, we have the following theorem.

Theorem 6. Algorithm Unweighted-CD-HC computes a $(\Delta_d + 2)$-approximation for the unweighted capacitated domination problem with hard capacities in polynomial time.

4 Conclusion

We conclude with an overview and future directions. In this paper we consider the problem of capacitated domination with hard capacities and provide algorithmic results for unweighted versions of this problem. We present a $(\Delta_d + 2)$-approximation for unweighted CD-HC. Our main ingredient is a delicate primal-dual schema combined with network flow and local arguments.

Below we discuss some future directions. Although as shown in [2], weighted VC-HC is already set-cover-hard.It is very interesting to explore the problem complexity of the weighted hard capacitated covering. On the other hand, the $(\Delta_d + 2)$ approximation factor we obtain for unweighted CD-HC seems to have room for further improvements. It is also interesting to explore for the possibility to obtain approximations with a lower-order factor.

References

[1] Wang-Chi Cheung, Michel X. Goemans, and Sam Chiu-Wai Wong. Improved algorithms for vertex cover with hard capacities on multigraphs and hypergraphs. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'14, pages 1714–1726, 2014.

[2] Julia Chuzhoy and Joseph (Seffi) Naor. Covering problems with hard capacities. *SIAM Journal on Computing*, 36(2):498–515, August 2006.

[3] Marek Cygan, Marcin Pilipczuk, and Jakub Onufry Wojtaszczyk. Capacitated domination faster than o(2n). In *Proceedings of the 12th Scandinavian Conference on Algorithm Theory*, SWAT'10, pages 74–80, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated domination and covering: A parameterized perspective. In *Proceedings of the 3rd International Conference on Parameterized and Exact Computation*, IWPEC'08, pages 78–90, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] Rajiv Gandhi, Eran Halperin, Samir Khuller, Guy Kortsarz, and Aravind Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. In *Proceedings of the 30th International Conference on Automata, Languages and Programming*, ICALP'03, pages 164–175, Berlin, Heidelberg, 2003. Springer-Verlag.

[6] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1):55–84, October 2004.

[7] Sudipto Guha, Refael Hassin, Samir Khuller, and Einat Or. Capacitated vertex covering. *Journal of Algorithms*, 48(1):257–270, August 2003.

[8] Mong-Jen Kao, Han-Lin Chen, and D.T. Lee. Capacitated domination: Problem complexity and approximation algorithms. *Algorithmica*, November 2013.

[9] Mong-Jen Kao, Chung-Shou Liao, and D. T. Lee. Capacitated domination problem. *Algorithmica*, 60(2):274–300, June 2011.

[10] Mathieu Liedloff, Ioan Todinca, and Yngve Villanger. Solving capacitated dominating set by using covering by subsets and maximum matching. In *Proceedings of the 36th International Conference on Graph-theoretic Concepts in Computer Science*, WG'10, pages 88–99, Berlin, Heidelberg, 2010. Springer-Verlag.

[11] Barna Saha and Samir Khuller. Set cover revisited: Hypergraph cover with hard capacities. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'12, pages 762–773, Berlin, Heidelberg, 2012. Springer-Verlag.

[12] Laurence A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.