Submodular Integer Cover and Its Application to Production Planning

Toshihiro Fujito[*] and Takatoshi Yabuta[**]

Graduate School of Information Science, Nagoya University,
Furo, Chikusa, Nagoya 464-8603 Japan
fujito@nuee.nagoya-u.ac.jp

Abstract. Suppose there are a set of suppliers i and a set of consumers j with demands b_j, and the amount of products that can be shipped from i to j is at most c_{ij}. The amount of products that a supplier i can produce is an integer multiple of its capacity κ_i, and every production of κ_i products incurs the cost of w_i. The *capacitated supply-demand (CSD)* problem is to minimize the production cost of $\sum_i w_i x_i$ such that all the demands (or the total demand requirement specified separately) at consumers are satisfied by shipping products from the suppliers to them.

To capture the core structural properties of CSD in a general framework, we introduce the *submodular integer cover (SIC)* problem, which extends the *submodular set cover (SSC)* problem by generalizing submodular constraints on subsets to those on integer vectors. Whereas it can be shown that CSD is approximable within a factor of $O(\log(\max_i \kappa_i))$ by extending the greedy approach for SSC to CSD, we first generalize the primal-dual approach for SSC to SIC and evaluate its performance. One of the approximation ratios obtained for CSD from such an approach is the maximum number of suppliers that can ship to a single consumer; therefore, the approximability of CSD can be ensured to depend only on the network (incidence) structure and not on any numerical values of input capacities κ_i, b_j, c_{ij}.

The CSD problem also serves as a unifying framework for various types of covering problems, and any approximation bound for CSD holds for set cover generalized *simultaneously* into various directions. It will be seen, nevertheless, that our bound matches (or nearly matches) the best result for each generalization *individually*. Meanwhile, this bound being nearly tight for standard set cover, any further improvement, even if possible, is doomed to be a marginal one.

1 Introduction

1.1 Capacitated Supply-Demand Problem

Suppose there are a set A of factories, that all produce the same product, and a set B of customers that use the product. Each factory $i \in A$ has capability

[*] Supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science, Sports and Culture of Japan.

[**] currently at Toyota Motor Corporation

of producing products in the units of κ_i tons, incurring the cost of w_i dollars per unit. Customer $j \in B$ requests b_j tons of the product every month. ¿From factory i to customer j at most c_{ij} tons of the product can be transported every month. Moreover, the total demand b_{total}, with $0 \leq b_{\text{total}} \leq \sum_{j \in B} b_j$, specifies, out of the total requested amount $\sum_{j \in B} b_j$, what amount of products need to be supplied in total. A monthly production plan specifies the number of units to be produced a month at each factory. The problem is: what is the most economical monthly production plan to fulfill at least b_{total} tons of all the needs requested by the customers ? We call this problem the *capacitated supply-demand (CSD)* problem.

Using a supply-demand model network, this problem can be defined equivalently as follows. Let $\mathcal{N} = (V, E)$ be a network, where $V = A \cup B \cup \{s, t, t'\}$, with source s, subsink t, sink t', a set A of supply nodes, and a set B of demand nodes. Each supply node $i \in A$ has an incoming arc from source s, each demand node has an outgoing arc to subsink t, one arc goes from t to sink t', and all the other arcs go from A to B (thus, $E = (\{s\} \times A) \cup E' \cup (B \times \{t\}) \cup \{(t, t')\}$ where $E' \subseteq (A \times B)$). All the arcs other than those in $\{s\} \times A$ are a priori associated with integral capacities; each capacity b_j on arc (j, t) specifies the "demand" requested by node $j \in B$, c_{ij} on arc $(i, j) \in A \times B$ limits the amount of supply that can be shipped from $i \in A$ to $j \in B$, and b_{total} on (t, t') is the total amount of demands to be supplied. The capacities on the remaining arcs $(s, i) \in \{s\} \times A$ are not given initially; rather, they need to be "purchased" as follows. For each $i \in A$ the unit capacity of κ_i is available for the unit cost of w_i, and it costs $w_i x_i$ to install the capacity of $a_i = \kappa_i x_i$ on (s, i), where a nonnegative integer x_i specifies the number of unit capacities to be used on (s, i). With all the capacities fully specified, the network is denoted by $\mathcal{N} = (A \cup B \cup \{s, t, t'\}, E, a, b, c, b_{\text{total}})$, where $a_i = \kappa_i x_i$ for each $i \in A$. The problem is then to install capacities $a \in \mathbb{Z}_+^A$ of minimum total cost on arcs in $\{s\} \times A$ by purchasing the available unit capacities κ, so that the total demand of b_{total} can be shipped in $\mathcal{N}$, or in other words, the max flow value reaches b_{total} in $\mathcal{N}$.

It should be clear by now that, denoting by f_{ij} the amount of a flow going from $i \in A$ to $j \in B$ in $\mathcal{N}$, CSD can be succinctly formulated by the following integer program:

$$\begin{array}{lrl}
 & \min \ \sum_{i \in A} w_i x_i & \\
\text{subject to:} & \sum_{j \in B} f_{ij} \leq \kappa_i x_i & i \in A \\
 & \sum_{i \in A} f_{ij} \leq b_j & j \in B \\
 & \sum_{j \in B} \sum_{i \in A} f_{ij} \geq b_{\text{total}} & \\
 & x_i \in \mathbb{Z}_+ & i \in A \\
 & 0 \leq f_{ij} \leq c_{ij} & i \in A,\ j \in B
\end{array}$$

where $w \in \mathbb{Q}_+^A, C = [c_{ij}] \in \mathbb{Z}_+^{A \times B}, \kappa \in \mathbb{Z}_+^A, b \in \mathbb{Z}_+^B$. To capture the core structural properties of CSD in a general framework, however, we next introduce the *submodular integer cover* problem, and will later derive an IP formulation for it.

1.2 Submodular Integer Cover

Definition 1. *A function $g : \mathbb{R}^n \to \mathbb{R}$ is said to be*

1. nondecreasing *if $x \le y$ implies $g(x) \le g(y)$, and*
2. submodular *if $g(x) + g(y) \ge g(x \vee y) + g(x \wedge y)$ for all $x, y \in \mathbb{R}^n$,*

where $x \vee y$ and $x \wedge y$ are, respectively, the vectors of componentwise maxima and minima of x and y: $(x \vee y)_i = \max\{x_i, y_i\}, (x \wedge y)_i = \min\{x_i, y_i\}$.

Let $G = (V, E)$ be a network with a nonnegative capacity $c(a)$ for each arc $a \in E$. Two arcs are said to be *parallel* if every simple cycle containing both of them orients them in the opposite direction, and a set of arcs is *parallel* if it consists of pairwise parallel arcs. Let P be a parallel arc set and denote the vector of capacities on arcs in P by c_P.

Proposition 2 (Gale-Politof [12]) *The maximum flow value F of G as a function in c_P is submodular.*

Recall the network $\mathcal{N}$ constructed in Sect. 1.1. The arc set $\{s\} \times A$ in $\mathcal{N}$ is then parallel, and hence, the max flow value F of $\mathcal{N}$ is submodular in $c_{\{s\}\times A}$. Define a function $\rho : \mathbb{Z}_+^A \to \mathbb{R}_+$ s.t.

$$\rho(x) = (\text{max flow value of } \mathcal{N} \text{ with } a_i = \kappa_i x_i, \forall i)$$

for a CSD instance of $(\mathcal{N}, \kappa, w)$. Then, $x \in \mathbb{Z}_+^A$ is a feasible CSD solution for $(\mathcal{N}, \kappa, w)$ iff $\rho(x) = b_{\text{total}} = \rho(\vec{\infty})$. Using ρ, CSD can be thus formulated by $\min\{\sum_{i \in A} w_i x_i \mid \rho(x) = \rho(\vec{\infty})\}$.

One can easily observe that ρ is nondecreasing as well as submodular by Proposition 2 since $\rho(x) = F((\cdots, \kappa_i x_i, \cdots))$, and CSD thus fits into the following general framework:

Definition 3. *Given a finite set A, a weight function $w : A \to \mathbb{R}$, and a nondecreasing submodular function $g : \mathbb{Z}_+^A \to \mathbb{R}$, the problem of computing $\min\{\sum_{i \in A} w_i x_i \mid g(x) = g(\vec{\infty})\}$ is called the* Submodular Integer Cover (SIC) *problem. Any $x \in \mathbb{Z}_+^A$ satisfying $g(x) = g(\vec{\infty})$ is a* solution *for the instance (A, w, g) of SIC.*

NOTE: In case when g is a *set* function (i.e., a function defined on 0-1 vectors), the problem is known as *submodular set cover* [29].

1.3 Related Problems and Previous Work

When all the unit capacities κ_i equal to one, CSD can be seen reducible to the minimum-cost flow problem. Another basic problem arising as a special case is the *set cover (SC)* or *vertex cover (VC)* problems, classic *NP*-hard problems of which polynomial time approximability has been intensively studied in the literature. In fact CSD serves as a unifying framework for various types of covering problems as will be seen below. In SC, given a family $\mathcal{F}$ of subsets of some base

Table 1. Covering Problems in the framework of CSD

	κ_i	c_{ij}	b_j	b_{total}
Set Cover	$\lvert\delta^+(i)\rvert$	1	1	$\lvert B\rvert = \sum_{j \in B} b_j$
Multicover	$\lvert\delta^+(i)\rvert$	1		$\sum_{j \in B} b_j$
Capacitated Set Cover		1	1	$\lvert B\rvert = \sum_{j \in B} b_j$
(with Demands)		1		$\sum_{j \in B} b_j$
Partial Set Cover	$\lvert\delta^+(i)\rvert$	1	1	

set U with associated nonnegative costs, it is required to compute a minimum cost subfamily $\mathcal{C}$ such that every element of U is "covered by" (i.e., contained in) some subset in $\mathcal{C}$. Defined analogously on graphs G, VC is to compute a minimum cost vertex subset C in G such that every edge of G is incident to some vertex in C. In the *multicover (MC)* problem, each element j of U in an SC instance is associated with "demand" b_j, and each j now needs to be covered b_j times (by different subsets). One of the most general problems previously considered along this line is the *multiset multicover (MMC)* problem, that can be defined by the following integer program: $\min\{w^T x \mid x^T C \geq b^T, x \leq u, x \in \mathbb{Z}_+^n\}$. It also has a version without explicit upper bounds u on x. In *capacitated SC*, each subset $S \in \mathcal{F}$ in an SC instance is associated with capacity κ_S and cost w_S. A single copy of S can cover only κ_S elements among those contained in S, and by paying w_S per copy, more copies of S can be used to cover more elements of S. In case when each element e is associated with demand b_e in capacitated SC, e has to be covered b_e times. In yet another direction of generalizations of SC, it is required to cover only b_{total} elements or more (instead of all) in *partial SC* when an additional integer b_{total} is given to SC.

For a node $v \in V$ in network $\mathcal{N}$, let $\delta^+(v)$ ($\delta^-(v)$, resp.) denote the set of arcs leaving v (entering v, resp.). For a finite set J, $J' \subseteq J$, and a vector $z \in \mathbb{Z}_+^J$ in general, $z(J')$ will be used as an abbreviation for $\sum_{j \in J'} z_j$. Those covering problems listed above can be realized in CSD by fixing some of problem parameters appropriately (see Table 1). In MMC $\min\{w^T x \mid x^T C \geq b^T, x \leq u, x \in \mathbb{Z}_+^n\}$, explicit upper bounds $x \leq u$ are called *multiplicity constraints*, and if it is non-existent, a trivial upper bound on x_i is $\max_j \lceil b_j / c_{ij} \rceil$. When cast in CSD, each $i \in A$ is replaced (not explicitly) by u_i copies, $i_1, \ldots, i_{u_i}$, each $c_{i_l j}$ is set to $\min\{c_{ij}, \max\{0, b_j - \sum_{k=1}^{l-1} c_{i_k j}\}\}$, and the unit capacity κ_{i_l} to $c(\delta^+(i_l))$. (We remark that possibly non-polynomial expansion of problem instances in this reduction causes no trouble in our algorithm.)

It was (or can be) shown in all the cases that a greedy heuristic yields a factor $H(\max_{i \in A} \kappa_i) = O(\log(\max_{i \in A} \kappa_i))$ approximation, where $H(k) = \sum_{i=1}^k (1/i)$ is the kth harmonic number; see [22, 24, 6] for SC, [8] for MMC, [26] for partial SC, and [5] for capacitated MMC. Other approximation bounds known for these problems include:

- $\max_{j \in B} |\delta^-(j)|$ [20, 2] and $\max_{j \in B} |\delta^-(j)| - (1 - o(1)) \frac{(\max_{j \in B} |\delta^-(j)| - 1) \ln \ln n}{\ln n}$ [18] for SC.

– $\max_{j \in B} |\delta^-(j)|$ for MC [17], and $\max_{j \in B} |\delta^-(j)| - b_{\min} + 1$ for unweighted MC [25], where $b_{\min} = \min_{j \in B} b_j$.
– $O(\log |B|)$ [23] and $\max_{j \in B} |\delta^-(j)| =$ "max # of nonzero entries in a row of C" [4] for MMC.
– $\max_{j \in B} |\delta^-(j)|$ for capacitated SC, and 3 for capacitated VC with "unsplittable" demands [16].
– $\max_{j \in B} |\delta^-(j)|$ for partial SC [1, 11, 13], 2 for partial VC [10, 3, 21], and $2 - \Theta(\frac{\ln \ln |\delta^+(i)|}{\ln |\delta^+(i)|})$ for unweighted partial VC on graphs of maximum degree $|\delta^+(i)|$ [19].

On the other hand, the following lower bounds are known for approximability of SC: $(1 - \epsilon) \ln |B|$ for any $\epsilon > 0$ [9] (assuming $NP \not\subset \mathrm{DTIME}\left(n^{O(\log \log n)}\right)$), and $|\delta^-(j)| - 1 - \epsilon$ [7].

CSD can be seen also related to the *capacitated facility location* problem, the *network loading* problem, and the *single-sink buy-at-bulk* problem among others. In fact, if shipping a unit product from $i \in A$ to $j \in B$ incurs a certain cost and the objective is to minimize total cost of production and shipping, CSD corresponds to the capacitated facility location problem having "flow constraints" in it. To the best of our knowledge, however, no covering-type problem with covering capacities *and* flow constraints explicitly given as in CSD, has been previously considered.

1.4 Summary of Results

In designing approximation algorithms for CSD or SIC, it is natural to consider extending known approximation algorithms for SSC. There are two such algorithms, one greedy [29] and the other primal-dual [11]. It is rather straightforward to extend the former to CSD resulting in the approximation ratio of $H(\max_{i \in A} \kappa_i) = O(\log(\max_{i \in A} \kappa_i))$. It will be seen, on the other hand, that the primal-dual approach for SIC yields an approximation algorithm for CSD running in time $O(nM(n, m))$, where $M(n, m)$ denotes time complexity of computing an $s - t$ max flow in a network with n nodes and m arcs. It requires much more intricate analysis based on reasoning on the relationship between flow values and capacity settings, however, to estimate its performance ratio, and to describe it, let $\delta^{D-}(j) = \delta^-(j) - D, b_j^D = b_j - c(D)$, and $c_{ij}^D = \min\{c_{ij}, b_j^D\}$ for any $D \subset \delta^-(j)$. It will be shown that the approximation factor guaranteed is

$$\max\left\{2, \max_{j \in B,\, D \subset \delta^-(j)} \left\{\frac{c^D(\delta^{D-}(j))}{b_j^D}\right\}\right\} \tag{1}$$

in its general form. Various consequences can be drawn immediately from (1), e.g.,

1. By assuming w.l.o.g. that $c_{ij} \leq b_j, \forall i, j$, (1) reduces to $\max_{j \in B} |\delta^-(j)|$; hence, the approximability of CSD can be ensured to dependent only on the network (incidence) structure of a given instance, and *not* on any numerical values of input capacities κ_i, b_j, c_{ij}.

The primal-dual algorithm is thus more effective than greedy when the number of suppliers that can ship products to each consumer is relatively small (such as in “vertex cover type” problems).

To measure the effectiveness of these bounds, it is instructive to compare them with existing results for the covering problems previously considered. The bound in 1. matches the best ones for capacitated SC [16], partial SC [1, 11, 13], weighted MC [17], and MMC [4] (see 3. below), respectively. Meanwhile, this bound nearly matching the best one for SC [18] as well, any further improvement would be necessarily a marginal one, even if possible, due to the strong lower bound of $\max_{j\in B}|\delta^-(j)|-1-\epsilon$ for standard SC [7]. Further consequences implied by (1) (or the bound in 1.) include:

2. Capacitated VC with splittable demands is approximable within a factor of 2, in contrast with the 3-approximation of [16] for unsplittable demands.
3. In MMC, with or without multiplicity constraints, c_{ij}^D's ($=\min\{c_{i_l j}, b_j^D\}$'s) in (1) are evaluated s.t., when summed up over all i_l's for some i, they never exceed b_j^D (more details will be given in the full version). And then, the obtained bound (assuming $\max_j |\delta^-(j)| \geq 2$) is at most

$$\max\left\{2, \max_{j,D}\left\{\frac{|\delta^{D-}(j)|b_j^D}{b_j^D}\right\}\right\} = \max\{2, \max_{j,D}|\delta^-(j)-D|\} = \max_{j\in B}|\delta^-(j)|,$$

which coincides with the approximation factor, “max # of nonzero entries in a row of C”, given in [4].
4. Depending on actual values of c_{ij} and b_j, the estimation could be further lowered. If $c_{ij} = c_j$, $\forall i, j$, let $b_j = s_j c_j + t_j$ s.t. $0 \leq s_j$, $0 < t_j \leq c_j$. Then, the value of (1) can be seen reducible to $\max_{j\in B}\left\{\max\left\{|\delta^-(j)| - s_j, (|\delta^-(j)| - s_j + 1)\frac{c_j}{c_j+t_j}\right\}\right\}$. Thus, when $c_{ij} = 1, \forall i, j$, for instance, CSD is approximable within a factor of $\max_{j\in B}(|\delta^-(j)| - b_j) + 1$. This CSD bound with such restrictions on c_{ij}'s alone already improves e.g. $\max_{j\in B}|\delta^-(j)| - b_{\min} + 1$ for unweighted MC observed in [25].

In the *multi-capacitated* version of CSD, multiple types of unit capacities are available at different prices for each $i \in A$. Such a generalization enables CSD to reflect e.g. an “economy of scale” (or “volume discount”). Our algorithm still works with this version providing the same approximation guarantee of (1) (details given in the full version).

2 Approximating Submodular Integer Cover

It is rather straightforward to obtain the greedy bound of $H(\max_{i\in A}\kappa_i)$ for CSD by extending the greedy algorithm for SSC and its performance analysis given by Wolsey [29]. Therefore, we focus on the primal-dual approach for SIC and its application to CSD for the rest of the paper.

2.1 Preliminaries and IP Formulation

Definition 4. *Let* $\mathbf{e}^k$ *denote the* k*th unit vector in* $\mathbb{Z}_+^A$ *s.t.* $\mathbf{e}_i^k = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise} \end{cases}$ *for* $i \in A$. *For nondecreasing* g *we also assume it is* bounded, *i.e., there exists* $u \in \mathbb{R}$ *s.t.* $g(x) \le u$, $\forall x \in \mathbb{Z}_+^A$. *Then, there must exist an integer* u_i *for each* $i \in A$ *s.t., for any* x *with* $x_i = u_i$ *and* $x'_i \ge u_i$, $g(x') = g(x)$ *if* $x'_j = x_j$ *for* $j \in A - \{i\}$. *Let* u_i *be minimal such an integer for each* $i \in A$, *and let* χ^S *denote the vector in* $\mathbb{Z}_+^A$ *s.t.* $\chi_i^S = \begin{cases} u_i & \text{if } i \in S \\ 0 & \text{otherwise} \end{cases}$ *for* $S \subseteq A$. *Thus,* $g(\chi^A) = \sup_{x \in \mathbb{Z}_+^A} g(x)$.

Let L_i be a lattice for $i = 1, \cdots, n$, and L be a sublattice of $\prod_{i=1}^n L_i$. It was shown by Topkis that a submodular function on L has antitone (i.e., nonincreasing) differences on L:

Proposition 5 (Topkis [27]) *Let* $(x_1, \cdots, x_n)$ *be an element of* L. *If* g *is a real-valued submodular function on* L, *for all* $j \neq k$ *with each* x_i *fixed for* $i \neq j$ *and* $i \neq k$, $g(x_j, z) - g(x_j, x_k)$ *is nonincreasing in* x_j *on* $L^z \cap L^{x_k}$ *for each* $x_k < z$ *in* L_k, *where* $L^t = \{x \mid (x_1, \cdots, x_j = x, \cdots, x_k = t, \cdots, x_n) \in L\}$.

Lemma 6. *If* g *is submodular and nondecreasing,*

$$g(x) \le g(\chi^S) + \sum_{j \in A-S} (g(\chi^S + \mathbf{e}^j) - g(\chi^S))x_j$$

for $x \in \mathbb{Z}_+^A$ *and* $S \subseteq A$.

Proof. Let $A - S = \{j_1, \ldots, j_r\}$. Then,

$$\begin{aligned} g(x + \chi^S) - g(\chi^S) &= \sum_{t=1}^{r} \sum_{l=1}^{x_{j_t}} \{g(\chi^S + \sum_{k=1}^{t-1} x_{j_k}\mathbf{e}^{j_k} + l\mathbf{e}^{j_t}) \\ &\qquad - g(\chi^S + \sum_{k=1}^{t-1} x_{j_k}\mathbf{e}^{j_k} + (l-1)\mathbf{e}^{j_t})\} \\ &\le \sum_{t=1}^{r} \sum_{l=1}^{x_{j_t}} \{g(\chi^S + \mathbf{e}^{j_t}) - g(\chi^S)\} \\ &= \sum_{t=1}^{r} \{g(\chi^S + \mathbf{e}^{j_t}) - g(\chi^S)\}x_{j_t} \end{aligned}$$

where the inequality holds due to Proposition 5. Since g is also nondecreasing,

$$\begin{aligned} g(x) \le g(x + \chi^S) &= g(\chi^S) + (g(x + \chi^S) - g(\chi^S)) \\ &\le g(\chi^S) + \sum_{t=1}^{r} \{g(\chi^S + \mathbf{e}^{j_t}) - g(\chi^S)\}x_{j_t} \end{aligned}$$

□

Define function $\Delta_S : A - S \to \mathbb{Z}_+$ for $S \subseteq A$ s.t. $\Delta_S(i) = g(\chi^S + \mathbf{e}^i) - g(\chi^S)$.

Theorem 7. *If x is a solution for an SIC instance (A, w, g), it is feasible in the following integer program:*

$$\text{(IP)} \qquad \textit{subject to:} \quad \begin{array}{l} \min \sum_{i \in A} w_i x_i \\ x \in \mathbb{Z}_+^A \\ \sum_{i \in A-S} \Delta_S(i) x_i \geq g(\chi^A) - g(\chi^S),\ S \subseteq A \end{array}$$

Consequently, the optimum for (A, w, g) is lower bounded by that of the corresponding (IP).

Proof. If x is a solution for (A, w, g), $g(x) = g(\vec{\infty}) = g(\chi^A)$, and by Lemma 6,

$$g(\chi^A) = g(x) \leq g(\chi^S) + \sum_{i \in A-S} (g(\chi^S + \mathbf{e}^i) - g(\chi^S)) x_i = g(\chi^S) + \sum_{i \in A-S} \Delta_S(i) x_i$$

for any $S \subseteq A$; hence, x satisfies all the constraints of (IP). □

2.2 Primal-Dual Schema

To design a primal-dual based approximation algorithm for SIC, we begin with relaxing the integral constraints $x \in \mathbb{Z}_+^A$ of (IP) to the linear constraints $x \geq 0$. By taking the dual of the resulting LP relaxation (LP) of (IP), we next obtain

$$\text{(D)} \qquad \text{subject to:} \quad \begin{array}{ll} \max \sum_{S \subseteq A} \left(g(\chi^A) - g(\chi^S)\right) y_s & \\ \sum_{S: i \in A-S} \Delta_S(i) y_S \leq w_i, & i \in A \\ y_S \geq 0, & S \subseteq A \end{array}$$

The primal-dual schema for "set cover type" problems (i.e., "covering by a 0-1 vector" type) is by now a well established algorithmic technique (see surveys given in e.g. [14, 28]); we here extend it to the primal-dual pair of (IP) and (D) in designing an algorithm called PD. It consists of the following two phases; the phase in which a maximal dual solution y is constructed in a greedy fashion, and an integral primal solution x is correspondingly chosen s.t. it satisfies the primal complementary slackness conditions with y, followed by the phase called *reverse delete* in which x is ensured to satisfy minimality conditions in a certain order.

More specifically, starting with $F = \emptyset$ and the dual solution $y = 0$, a variable y_F in (D) is iteratively increased maximally without violating dual feasibility in the first phase, so that the dual constraint for i becomes newly binding for some i in $A - F$; that is, $\sum_{S: i \in A-S} \Delta_S(i) y_S = w_i$. This amounts to finding i in $A - F$ (say, i') at the lth iteration minimizing

$$\frac{w_i - \sum_{S: i \notin S, S \neq F} \Delta_S(i) y_S}{\Delta_F(i)} = \frac{w_i - \sum_{1 \leq t \leq l-1} \Delta_{F_t}(i) y_{F_t}}{\Delta_F(i)}$$

and setting y_F to

```
Initialize: $x = 0, y = 0, F = \emptyset, \mathsf{STACK} = \emptyset, \bar{w}_i \leftarrow w_i \ (\forall i \in A)$.
While $x$ is not an SIC solution for $(A, w, g)$ (i.e., $g(x) < g(\chi^A)$) do
    Let $y_F \leftarrow \min_{i \in A-F}\{\bar{w}_i / \Delta_F(i)\}$ and $i' \leftarrow \mathrm{argmin}_{i \in A-F}\{\bar{w}_i / \Delta_F(i)\}$.
    Add $i'$ into $F$, push it onto $\mathsf{STACK}$, and set $x_{i'} \leftarrow u_{i'}$.
    For each $i \in A - F$ do
        Reset $\bar{w}_i \leftarrow \bar{w}_i - \Delta_F(i) y_F \quad \left(= \bar{w}_i - \frac{\Delta_F(i)}{\Delta_F(i')}\bar{w}_{i'}\right)$.
While $\mathsf{STACK} \neq \emptyset$ do
    Let $i' \leftarrow \mathsf{pop}(\mathsf{STACK})$.
    Set $x_{i'} \leftarrow \min\{x_{i'} \mid g(x) = g(\chi^A)\}$.
Output $x$.
```

Fig. 1. Algorithm PD for SIC

$$\min_{i \in A-F}\left\{\frac{w_i - \sum_{S: i \notin S, S \neq F} \Delta_S(i) y_S}{\Delta_F(i)}\right\} = \frac{w_{i'} - \sum_{S: i' \notin S, S \neq F} \Delta_S(i') y_S}{\Delta_F(i')}$$
$$= \frac{w_{i'} - \sum_{1 \le t \le l-1} \Delta_{F_t}(i') y_{F_t}}{\Delta_F(i')}$$

where $F_0 = \emptyset \subseteq F_1 \subseteq \ldots \subseteq F_T$ denote (intermediate) F's constructed, in this order, by each iteration of the first phase (So, $y_S > 0$ iff S is one of these F's). This i' (or any other binding i's) is then added to F so that F remains as the set of all i's whose corresponding constraints are binding. At the same time it is kept track of in what order dual constraints become binding during this process (or, in what order i's enter F) in the first phase. Let x represent χ^F. Then, x eventually becomes an SIC solution as F may grow up to A if necessary.

In the second phase an actual SIC solution x is constructed based on F and the ordering of i's in F computed as above. Starting with $x = \chi^F$, each $i \in F$ is processed, one by one, in reversal of the order in which they were added to F during the first phase, and x_i is set to the minimal value needed for x to remain as a solution for (A, w, g). The SIC solution x constructed in this manner satisfies a certain minimality property: For all $t = 1, \ldots, T$, the values of x_i's with $i \in F - F_t$ are the ones "minimally" required, in addition to χ^{F_t}, to increase its g-value from $g(\chi^{F_t})$ up to the required $g(\chi^A)$; in other words, the value of $g(x + \chi^{F_t})$ drops below that of $g(\chi^A)$ whenever any x_i with $i \in F - F_t$ is decremented. The pseudo-code description of this algorithm is given in Fig. 1.

2.3 Analysis

In the algorithm PD any element $i \in A$ enters F (in the first phase) iff the corresponding dual constraint becomes binding; that is, $w_i = \sum_{S: i \in A-S} \Delta_S(i) y_S$. Therefore, the weight of computed x is

$$\sum_{i \in F} w_i x_i = \sum_{i \in F}\left(\sum_{S: i \in A-S} \Delta_S(i) y_S\right) x_i = \sum_{S \subseteq A}\left(\sum_{i \in F-S} \Delta_S(i) x_i\right) y_S.$$

When the RHS in this equation is compared with the objective function of (D), of which value gives a lower bound for the optimal weight, it can be seen that the ratio of the weight of x to the optimal weight is bounded above by

$$\max_{S\subseteq A, y_S>0} \frac{\sum_{i\in F-S}\Delta_S(i)x_i}{g(\chi^A)-g(\chi^S)} = \max_{1\le t\le T} \frac{\sum_{i\in F-F_t}\Delta_{F_t}(i)x_i}{g(\chi^A)-g(\chi^{F_t})}. \tag{2}$$

Recall now how an actual SIC solution x is constructed in the second phase.

Definition 8. *We call a solution x for (A, w, g)* minimal w.r.t. $S \subseteq A$ *if $x + \chi^{A-S}$ becomes a non-solution when any $x_i > 0$ with $i \in S$ is decremented.*

It is ensured that x be minimal w.r.t. $A - F_t$ for each t $(1 \le t \le T)$. Therefore, x can be restricted to such solutions for (A, w, g) in estimating the bound in (2), and we have the following general approximation bound for SIC:

Theorem 9. *When applied to an SIC instance (A, w, g), the algorithm* PD *computes a solution such that the ratio of its weight to the optimal weight is bounded by*

$$\max\left\{\frac{\sum_{i\in A-S}\Delta_S(i)x_i}{g(\chi^A)-g(\chi^S)}\right\}, \tag{3}$$

where max *is taken over any $S \subseteq A$ and such a solution x for (A, w, g) that is minimal w.r.t. $A - S$.*

3 Application to CSD Problem

To model CSD in the framework of SIC, let $u_i = \lceil c(\delta^+(i))/\kappa_i\rceil$, $\forall i \in A$, and define a function $\rho : \mathbb{Z}_+^A \to \mathbb{Z}_+$ s.t. $\rho(x) =$ (max flow value of $\mathcal{N}$ with $a_i = x_i\kappa_i, \forall i$) for a CSD instance of $(\mathcal{N}, \kappa, w)$. As we will need to consider $\mathcal{N}$ specified with flow capacities of our own choice, such a network with new capacities a', b', c' will be denoted by $\mathcal{N}(a = a', b = b', c = c')$. Let $\mathcal{N}_S$ denote the network $\mathcal{N}$ in which capacity a_i on $(s, i) = \infty$ if $i \in S$ and $a_i = 0$ otherwise, for $S \subseteq A$ (i.e., $\mathcal{N}_S = \mathcal{N}(a_i = \infty$ for $i \in S, a_i = 0$ for $i \in A - S))$. In estimating the bound of (3) in the context of CSD, the next lemma is useful:

Lemma 10. *Let f be any max flow in the network $\mathcal{N}_S$ for $S \subseteq A$. When f is augmented up to a max flow in $\mathcal{N}_A$, no augmenting path passes through any $i \in S$.*

Proof. Consider the residual network $\mathcal{N}_A^r$ with respect to $\mathcal{N}_A$ and f. Assuming that f is not yet a max flow in $\mathcal{N}_A$, observe that f saturates either arc (i, j) or (j, t) in $\mathcal{N}_S$ for each $i \in S$ and all j with $(i, j) \in \delta^+(i)$. Or, in other words, for each j reachable from some $i \in S$, either arc (j, t) is saturated, or all (i, j)'s going from S to j are saturated. Therefore, $\mathcal{N}_A^r$ has no s-t dipath in it passing through any $i \in S$, and it remains so even if f is augmented to a larger flow in $\mathcal{N}_A$; even after augmentations, the original f is nowhere decremented, and saturated arcs continue to block any augmentation through nodes in S, due to the network structure $(A \cup B \cup \{s, t, t'\}, E)$ of $\mathcal{N}$. □

The next is a key lemma of this paper, yet due to the space limitation, its proof is omitted here (will be given in the full version):

Lemma 11. *Let $x \in \mathbb{Z}_+^{\bar{A}}$ be a minimal CSD solution for $(\bar{\mathcal{N}} = (\bar{A} \cup \bar{B} \cup \{s, t, t'\}, \bar{E}, \bar{b}, c, \bar{b}_{\text{total}}), \kappa, w)$. Then,*

$$\frac{\sum_{i \in \bar{A}} \kappa_i x_i}{\bar{b}_{\text{total}}} \leq \max\left\{2, \max_{j \in \bar{B}} \frac{c(\delta^-(j))}{\bar{b}_j}\right\}. \tag{4}$$

Consider the network $\mathcal{N}'_S$ obtained from $\mathcal{N}_S$ and a max flow f in it by removing all the nodes in S along with incident arcs, and adjusting the capacities on (j, t) to $b_j - f(j, t)$. It follows from Lemma 10 that the values of $\Delta_S(i)$ and $\rho(\chi^A) - \rho(\chi^S)$ in (3) coincide with the max flow value in $\mathcal{N}'_S(a_i = \kappa_i)$ and $b_{\text{total}} - |f|$, respectively, where the former is always bounded by κ_i. Recall that, x being minimal w.r.t. $A - S$, the vector $(x_i)_{i \in A - S}$ specifies minimal capacities on arcs $(s, i), i \in A - S$, required to increase the max flow value $|f|$ of $\mathcal{N}_S$ to b_{total}; again from Lemma 10, it means that $(x_i)_{i \in A - S}$ is by itself a minimal CSD solution for $\mathcal{N}'_S$ with b_{total} adjusted to the necessary increments of $b_{\text{total}} - |f|$. Therefore, the value of (3) can be evaluated by taking the maximal value of (4) over all possible $\mathcal{N}'_S$ (subject to $y_S > 0$) used in place of $\bar{\mathcal{N}}$ of Lemma 11. It follows from these observations and Theorem 9 that

Theorem 12. *For any $D \subset \delta^-(j)$, let $\delta^{D-}(j) = \delta^-(j) - D$, $b_j^D = b_j - c(D)$, and $c_{ij}^D = \min\{c_{ij}, b_j^D\}$. The algorithm* PD *computes a CSD solution to an instance $(\mathcal{N} = (A \cup B \cup \{s, t, t'\}, E, b, c, b_{\text{total}}), \kappa, w)$, s.t. the ratio of its cost to the optimal cost is bounded above by*

$$\max\left\{2, \max_{j \in B,\, D \subset \delta^-(j)} \left\{\frac{c^D(\delta^{D-}(j))}{b_j^D}\right\}\right\}. \tag{5}$$

By assuming w.l.o.g. that $c_{ij} \leq b_j, \forall i, j$, it can be shown that (5) reduces to $\max\{2, \max_{j \in B} |\delta^-(j)|\}$. If $\max_{j \in B} |\delta^-(j)| < 2$, however, $|\delta^-(j)| = 1$, $\forall j \in B$, and this occurs only in a trivial case.

Corollary 13. *The algorithm* PD *computes a CSD solution to an instance $(\mathcal{N} = (A \cup B \cup \{s, t, t'\}, E, b, c, b_{\text{total}}), \kappa, w)$, s.t. the ratio of its cost to the optimal cost is bounded above by* $\max_{j \in B}\{|\delta^-(j)|\}$.

Running Time. Each iteration in the first phase (first while-loop) can be carried out in time $O(m)$, whereas time complexity of the second phase (second while-loop) is dominated by that of computing a max flow in each iteration. The running time of PD, when applied to CSD, is thus $O(nM(n, m))$, and $M(n, m) = O(nm \log(n^2/m))$, for instance, when the Goldberg-Tarjan's algorithm is used [15].

References

1. Bar-Yehuda, R.: Using homogeneous weights for approximating the partial cover problem. In: Proc. 10th SODA. ACM-SIAM (1999) 71–75

2. Bar-Yehuda, R., Even, S.: A linear time approximation algorithm for the weighted vertex cover problem. J. Algorithms **2** (1981) 198–203
3. Bshouty, N.H., Burroughs, L.: Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In: Proc. 15th STACS. (1998) 298–308
4. Carr, R.D., Fleischer, L.K., Leung, V.J., Phillips, C.A.: Strengthening integrality gaps for capacitated network design and covering problems. In: Proc. 11th SODA. ACM-SIAM (2000) 106–115
5. Chuzhoy, J., Naor, J.: Covering problems with hard capacities. In: Proc. 43rd FOCS. IEEE (2002) 481–489
6. Chvátal, V.: A greedy heuristic for the set-covering problem. Math. Oper. Res. **4**(3) (1979) 233–235
7. Dinur, I., Guruswami, V., Khot, S., Regev, O.: A new multilayered PCP and the hardness of hypergraph vertex cover. In: Proc. 35th STOC. ACM (2003) 595–601
8. Dobson, G.: Worst-case analysis of greedy heuristics for integer programming with nonnegative data. Math. Oper. Res. **7**(4) (1982) 515–531
9. Feige, U.: A threshold of $\ln n$ for approximating set cover. In: Proc. 28th STOC. ACM (1996) 314–318
10. Fujito, T.: A unified local ratio approximation of node-deletion problems. In: Proc. ESA'96. (1996) 167–178
11. Fujito, T.: On approximation of the submodular set cover problem. Oper. Res. Lett. **25** (1999) 169–174
12. Gale, D., Politof, T.: Substitutes and complements in network flow problems. Discrete Appl. Math. **3** (1981) 175–186
13. Gandhi, R., Khuller, S., Srinivasan, A.: Approximation algorithms for partial covering problems. In: Proc. 28th ICALP. (2001) 225-236
14. Goemans, M.X., Williamson, D.P.: The primal-dual method for approximation algorithms and its application to network design problems. In: Hochbaum, D. (ed.): Approximation Algorithm for NP-Hard Problems. PWS, Boston (1996) 144–191
15. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. J. ACM **35** (1988) 921–940
16. Guha, S., Hassin, R., Khuller, S., Or, E.: Capacitated vertex covering with applications. In: Proc. 13th SODA. ACM-SIAM (2002) 858–865
17. Hall, N.G., Hochbaum, D.S.: A fast approximation algorithm for the multicovering problem. Discrete Appl. Math. **15** (1986) 35–40
18. Halperin, E.: Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. In: Proc. 11th SODA. ACM-SIAM (2000) 329–337
19. Halperin, E., Srinivasan, A.: Improved approximation algorithms for the partial vertex cover problem. In: Proc. APPROX 2002. (2002) 161–174
20. Hochbaum, D.S.: Approximation algorithms for the set covering and vertex cover problems. SIAM J. Comput. **11**(3) (1982) 555–556
21. Hochbaum, D.S.: The t-vertex cover problem: Extending the half integrality framework with budget constraints. In: Proc. APPROX'98. (1998) 111–122
22. Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. System Sci. **9** (1974) 256–278
23. Kolliopoulos, S. G., Young, N. E.: Tight approximation results for general covering integer programs. In: Proc. 42nd FOCS. IEEE (2001) 522–528
24. Lovász, L.: On the ratio of optimal integral and fractional covers. Discrete Math. **13** (1975) 383–390
25. Peleg, D., Schechtman, G., Wool, A.: Randomized approximation of bounded multicovering problems. Algorithmica **18** (1997) 44–66

26. Slavík, P.: Improved performance of the greedy algorithm for partial cover. Inform. Process. Lett. **64**(5) (1997) 251–254
27. Topkis, D.M.: Minimizing a submodular function on a lattice. Operations Res. **26**(2) (1978) 305–321
28. Vazirani, V.: Approximation Algorithms. Springer, Berlin (2001)
29. Wolsey, L.A.: An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica **2**(4) (1982) 385–393