

Primal-dual schema for capacitated covering problems

Tim Carnes · David B. Shmoys

Received: 14 October 2011 / Accepted: 24 July 2014 / Published online: 15 August 2014
© Springer-Verlag Berlin Heidelberg and Mathematical Optimization Society 2014

Abstract Primal-dual algorithms have played an integral role in recent developments in approximation algorithms, and yet there has been little work on these algorithms in the context of LP relaxations that have been strengthened by the addition of more sophisticated valid inequalities. We introduce primal-dual schema based on the LP relaxations devised by Carr et al. for the minimum knapsack problem as well as for the single-demand capacitated facility location problem. Our primal-dual algorithms achieve the same performance guarantees as the LP-rounding algorithms of Carr et al. which rely on applying the ellipsoid algorithm to an exponentially-sized LP. Furthermore, we introduce new flow-cover inequalities to strengthen the LP relaxation of the more general capacitated single-item lot-sizing problem; using just these inequalities as the LP relaxation, we obtain a primal-dual algorithm that achieves a performance guarantee of 2. Computational experiments demonstrate the effectiveness of this algorithm on generated problem instances.

Keywords Approximation algorithms · Primal-dual schema · Knapsack-cover inequalities · Lot-sizing

A preliminary version of this work appeared in the Proceedings of the 13th MPS Conference on Integer Programming and Combinatorial Optimization, 2008. Research supported partially by NSF grants CCR-0635121, CCR-0430682, DMI-0500263, CCF-0832782 & CCF-1017688.

T. Carnes (✉)
KPMG, LLP, Seattle, WA, USA
e-mail: tcarnes@gmail.com

D. B. Shmoys
School of Operations Research and Information Engineering and Computer Science Department,
Cornell University, Ithaca, NY, USA
e-mail: david.shmoys@cornell.edu

Mathematics Subject Classification 90C27 · 90C05 · 90B05**1 Introduction**

Primal-dual algorithms have played an integral role in recent developments in approximation algorithms, and yet there has been little work on these algorithms in the context of LP relaxations that have been strengthened by the addition of more sophisticated valid inequalities. We introduce primal-dual schema based on the LP relaxations devised by Carr et al. [7] for the minimum knapsack problem as well as for the single-demand capacitated facility location problem. Our primal-dual algorithms achieve the same performance guarantees as the LP-rounding algorithms of Carr et al., which rely on applying the ellipsoid algorithm to an exponentially-sized LP. Furthermore, we introduce new flow-cover inequalities to strengthen the LP relaxation of the more general capacitated single-item lot-sizing problem; using just these inequalities as the LP relaxation, we obtain a primal-dual algorithm that achieves a performance guarantee of 2.

We say an algorithm is an approximation algorithm with a performance guarantee of α when the algorithm runs in polynomial time and always produces a solution with cost within a factor of α of optimal. Primal-dual algorithms are able to gain the benefits of LP-based techniques, such as automatically generating a new lower bound for each problem instance, but without having to solve an LP. Primal-dual approximation algorithms were developed by Bar-Yehuda and Even and Chvátal [4, 9] for the weighted vertex cover and set cover problems, respectively. Subsequently, this approach has been applied to many other combinatorial problems, such as results of Agrawal et al. [2], Goemans and Williamson [12], Bertsimas and Teo [5] and Levi et al. [14] for other related covering problems. Other recent work has been done on covering problems with capacity constraints by Even et al. [10], Chuzhoy and Naor [8] and Gandhi et al. [11]. In a similar vein, Bar-Noy et al. [3] gave a general framework for devising primal-dual approximation algorithms for a variety of packing problems (although phrased through an equivalent local-ratio lens); even more relevant, they consider a *loss minimization* problem, and their algorithm, when specialized to the minimum knapsack problem, is identical to the algorithm proposed here, and their analysis shows that the algorithm is a 4-approximation algorithm. It is interesting to note that although one can infer the LP-framework for this algorithm (and its analysis), it is non-trivial to see that this amounts to exactly the flow-cover inequalities of Carr et al.

In developing primal-dual (or any LP-based) approximation algorithms, it is important to have a strong LP formulation for the problem. However, there are cases when the LP relaxation of the natural IP formulation for a problem has a large integrality gap. When this happens, one can mend the situation by introducing extra *valid* inequalities that hold for any integral solution to the problem, but restrict the feasible region of the LP. One class of valid inequalities that has proved useful for a variety of problems are called *flow-cover inequalities*. Some of the early flow-cover style inequalities were introduced by Padberg et al. [15] for various fixed charge problems. Another large class of flow-cover inequalities was developed by Aardal et al. [1] for the capacitated

facility location problem. Carr et al. [7] developed a different style of inequalities for simpler capacitated covering problems, which are called *knapsack-cover inequalities*, and these we use in developing our primal-dual algorithms. Levi et al. [13] used a subset of the flow-cover inequalities of Aardal et al. [1] to develop a LP-rounding algorithm for the multiple-item lot-sizing problem with monotone holding costs. Our model is not a special case of theirs, however, since our result allows time-dependent order capacities whereas their result assumes constant order capacities across all periods. Additionally, our result allows for time-dependent, per-unit production costs, and it is not clear if their result can handle this added condition. Following this work and the development of our own knapsack-cover inequalities for lot-sizing problems, Sharma and Williamson demonstrated that the result of Levi et al. [13] has an analogue based on our knapsack-cover inequalities as well. Finally, it is worth noting that Van Hoesel and Wagelmans [17] have a FPTAS for the single-item lot-sizing problem that makes use of dynamic programming and input data rounding. The disadvantage of this result is that the running time of the algorithm can be quite slow for particular performance guarantees. Although the primal-dual algorithm we develop is a 2-approximation algorithm, this is a worst-case bound and we would expect it to perform much better on average in practice. Also this is the first LP-based result for the case of time-dependent capacities.

Our Results This paper presents primal-dual 2-approximation algorithms for the following three covering problems (defined below): the minimum knapsack problem, the single-demand facility location problem, and single-item lot-sizing problem. The three models studied in this paper are generalizations of one another. That is to say, the minimum knapsack problem is a special case of the single-demand capacitated facility location problem, which is a special case of the single-item lot-sizing problem. We present the result in order of generality with the aim of explaining our approach in the simplest setting first.

- The *minimum knapsack problem* gives a set of items, each with a weight and a value. The objective is to find a minimum-weight subset of items such that the total value of the items in the subset meets some specified demand.
- In the *single-demand facility location problem* there is a set of facilities, each with an opening cost and a capacity, as well as a per-unit serving cost that must be paid for each unit of demand a facility serves. The goal is to open facilities to completely serve a specified amount of demand, while minimizing the total service and facility opening costs.
- Finally the *single-item lot-sizing problem* considers a finite planning period of consecutive time periods. In each time period there is a specified level of demand, as well as a potential order with a given capacity and order opening cost. A feasible solution must open enough orders and order enough product so that in each time period there is enough inventory to satisfy the demand of that period. The inventory is simply the inventory of the previous time period plus however much is ordered in the current time period, minus the demand for that period. However, a per-unit holding cost is incurred for each unit of inventory held over a given time period.

Additionally, we may consider a per-unit production cost for the amount of demand requested in each order.

The straightforward LP relaxations for these problems have a bad integrality gap, but can be strengthened by introducing valid flow-cover inequalities. The inequalities Carr et al. [7] developed for the minimum knapsack problem are as follows

$$\sum_{i \in F \setminus A} u_i(A) y_i \geq D - u(A) \quad \forall A \subseteq F,$$

where the y_i are the binary decision variables indicating if item i is chosen, $u(A)$ is the total value of the subset of items A , and the $u_i(A)$ can be thought of as the effective value of item i with respect to A , which is the minimum of the actual value and the right-hand-side of the inequality. These inequalities arise by considering that if we did choose all items in the set A , then we still have an induced subproblem on all of the remaining items, and the values can be truncated since we are only concerned with integer solutions. Our primal-dual algorithm works essentially as a modified greedy algorithm, where at each stage the item is selected that has the largest value per cost. Instead of the actual values and costs, however, we use the effective values and the slacks of the dual constraints as costs. Similar to the greedy algorithm for the traditional maximum-value knapsack problem, the last item selected and everything selected beforehand, can each be bounded in cost by the dual LP value, yielding a 2-approximation.

The study of approximation algorithms is not just to prove theoretical performance guarantees, but also to gain sufficient understanding of the mathematical structure of a problem so as to design algorithms that perform well in practice. We examine the performance of the primal-dual algorithm for the single-item lot-sizing problem in a series of computational experiments. We demonstrate the effectiveness of the algorithm by comparing it to other natural heuristics, some of which make use of the optimal LP solution.

The remainder of this paper is organized as follows. In Sect. 2 we go over the minimum knapsack result in more detail. In Sect. 3 we generalize this result to apply to the single-demand capacitated facility location problem. In Sect. 4 we generalize the flow-cover inequalities to handle the lot-sizing problem, and then present and analyze a primal-dual algorithm for the single-item lot-sizing problem. Finally, in Sect. 5 we present computational results analyzing the performance of the primal-dual algorithm for lot-sizing compared to other heuristics, as well as exploring the effectiveness of utilizing the algorithm to create a better formulation.

2 Minimum knapsack

In the *minimum knapsack problem* one is given a set of items F , and each item $i \in F$ has a value u_i and a weight f_i . The goal is to select a minimum weight subset of items, $S \subseteq F$, such that the value of S , $u(S)$, is at least as big as a specified demand, D . The natural IP formulation for this problem is

$$\begin{aligned}
 \text{opt}_{MK} &:= \min \sum_{i \in F} f_i y_i && \text{(MK-IP)} \\
 \text{s.t.} & \sum_{i \in F} u_i y_i \geq D \\
 & y_i \in \{0, 1\} && \forall i \in F, \quad (1)
 \end{aligned}$$

where the y_i variables indicate if item i is chosen. The following example from [7] demonstrates that the integrality gap between this IP and the LP relaxation is at least as bad as D . Consider just 2 items where $u_1 = D - 1$, $f_1 = 0$, $u_2 = D$ and $f_2 = 1$. Any feasible integer solution must set $y_2 = 1$ and have a cost of 1, whereas the LP solution can set $y_1 = 1$ and $y_2 = 1/D$ and incurs a cost of only $1/D$. To remedy this situation we consider using the flow-cover inequalities introduced in [7].

The idea is to consider a subset of items $A \subseteq F$ such that $u(A) < D$, and let $D(A) = D - u(A)$. This means that even if all of the items in the set A are chosen, we must choose enough items in $F \setminus A$ such that the demand $D(A)$ is met. This is just another minimum knapsack problem where the items are restricted to $F \setminus A$ and the demand is now $D(A)$. The value of every item can be restricted to be no greater than the demand without changing the set of feasible integer solutions, so let $u_i(A) = \min\{u_i, D(A)\}$. This motivates the following LP

$$\begin{aligned}
 \text{opt}_{MKP} &:= \min \sum_{i \in F} f_i y_i && \text{(MK-P)} \\
 \text{s.t.} & \sum_{i \in F \setminus A} u_i(A) y_i \geq D(A) \quad \forall A \subseteq F \\
 & y_i \geq 0 && \forall i \in F, \quad (2)
 \end{aligned}$$

and by the validity of the flow-cover inequalities argued above we have that every feasible integer solution to (MK-IP) is a feasible solution to (MK-P). The dual of this LP is

$$\begin{aligned}
 \text{opt}_{MKD} &:= \max \sum_{A \subseteq F} D(A) v(A) && \text{(MK-D)} \\
 \text{s.t.} & \sum_{A \subseteq F: i \notin A} u_i(A) v(A) \leq f_i \quad \forall i \in F \quad (3) \\
 & v(A) \geq 0 && \forall A \subseteq F.
 \end{aligned}$$

Our primal-dual algorithm begins by initializing all of the primal and dual variables to zero, which produces a feasible dual solution and an infeasible primal integer solution. Taking our initial subset of items, A , to be the empty set, we increase the dual variable $v(A)$. Once a dual constraint becomes tight, the item corresponding to that constraint is added to the set A , and we now increase the new variable $v(A)$. Note that increasing $v(A)$ does not increase the left-hand-sides of dual constraints corresponding to items in A , so dual feasibility will not be violated. This process is

repeated as long as $D(A) > 0$, and once we finish we call our final set of items S , which is our integer solution. This is a feasible solution to (MK-IP) since $D(S) \leq 0$, which implies $u(S) \geq D$.

Algorithm 1: Primal-Dual for Minimum Knapsack

```

y, v ← 0;
A ← ∅;
while D(A) > 0 do
    Increase v(A) until a dual constraint becomes tight for some item i;
    y_i ← 1;
    A ← A ∪ {i};
S ← A;
    
```

Theorem 1 Algorithm 1 terminates with a solution of cost no greater than $2 \cdot \text{opt}_{MK}$.

Proof Let ℓ denote the final item selected by Algorithm 1. Then because the algorithm only continues running as long as $D(A) > 0$ we have that

$$D(S \setminus \{\ell\}) > 0 \Rightarrow D - u(S \setminus \{\ell\}) > 0 \Rightarrow u(S \setminus \{\ell\}) < D.$$

Also, the variable $v(A)$ is positive only if $A \subseteq S \setminus \{\ell\}$. Thus, since the algorithm only selects items for which the constraint (3) has become tight, we have that the cost of the solution produced is

$$\sum_{i \in F} f_i y_i = \sum_{i \in S} f_i = \sum_{i \in S} \sum_{A \subseteq F: i \notin A} u_i(A) v(A).$$

The expression on the right-hand side is summing over all items, i , in the final solution S , and all subsets of items, A , that do not contain item i . This is the same as summing over all subsets of items, A , and all items in the final solution that are not in A . Thus we can reverse the order of the summations to obtain

$$\begin{aligned} \sum_{i \in F} f_i y_i &= \sum_{A \subseteq F} v(A) \sum_{i \in S \setminus A} u_i(A) \\ &= \sum_{A \subseteq F} v(A) (u(S \setminus \{\ell\}) - u(A) + u_\ell(A)) \end{aligned} \tag{4}$$

$$< \sum_{A \subseteq F} v(A) (D - u(A) + u_\ell(A)), \tag{5}$$

where (4) follows since for each item i other than ℓ , $u_i(A) = u_i$. Also (5) holds by making use of our observation above that $u(S \setminus \{\ell\}) < D$. But we also have $u_\ell(A) \leq D(A)$ by definition, hence

$$\sum_{i \in F} f_i y_i \leq \sum_{A \subseteq F} 2D(A) v(A) \leq 2 \cdot \text{opt}_{MKD}.$$

□

3 Single-demand facility location

In the *single-demand facility location problem*, one is given a set of facilities F , where each facility $i \in F$ has capacity u_i , opening cost f_i , and there is a per-unit cost c_i to serve the demand, which requires D units of the commodity. The goal is to select a subset of facilities to open, $S \subseteq F$, such that the combined cost of opening the facilities and serving the demand is minimized. The natural IP formulation for this problem is

$$\text{opt}_{FL} := \min \sum_{i \in F} (f_i y_i + D c_i x_i) \tag{FL-IP}$$

$$\text{s.t. } \sum_{i \in F} x_i = 1 \tag{6}$$

$$u_i y_i \geq D x_i \quad \forall i \in F \tag{7}$$

$$\begin{aligned} y_i &\geq x_i && \forall i \in F \\ y_i &\in \{0, 1\} && \forall i \in F \\ x_i &\geq 0 && \forall i \in F, \end{aligned} \tag{8}$$

where each y_i indicates if facility $i \in F$ is open and each x_i indicates the fraction of D being served by facility $i \in F$. The same example from the minimum knapsack problem also demonstrates the large integrality gap of this IP. We once again turn to the flow-cover inequalities introduced by Carr et al. [7].

For these inequalities, we once again consider a subset of facilities $A \subseteq F$ such that $u(A) < D$, and let $D(A) = D - u(A)$. This means that even if all of the facilities in the set A are opened, we must open enough facilities in $F \setminus A$ such that we will be able to assign the remaining demand $D(A)$. But certainly for any feasible integer solution, a facility $i \in F \setminus A$ cannot contribute more than $\min\{D x_i, u_i(A) y_i\}$ towards the demand $D(A)$. So if we partition the remaining orders of $F \setminus A$ into two sets F_1 and F_2 , then for each $i \in F_1$ we will consider its contribution as $D x_i$, and for each $i \in F_2$ we will consider its contribution as $u_i(A) y_i$. The total contribution of these facilities must be at least $D(A)$, so if we let \mathcal{F} be the set of all 3-tuples that partition F into three sets, we obtain the following LP

$$\text{opt}_{FLP} := \min \sum_{i \in F} (f_i y_i + D c_i x_i) \tag{FL-P}$$

$$\begin{aligned} \text{s.t. } \sum_{i \in F_1} D x_i + \sum_{i \in F_2} u_i(A) y_i &\geq D(A) && \forall (F_1, F_2, A) \in \mathcal{F} \\ x_i, y_i &\geq 0 && \forall i \in F, \end{aligned} \tag{9}$$

and by the validity of the flow-cover inequalities argued above we have that every feasible integer solution to (FL-IP) is a feasible solution to (FL-P). The dual of this LP is

$$\begin{aligned}
 \text{opt}_{FLD} &:= \max \sum_{(F_1, F_2, A) \in \mathcal{F}} D(A)v(F_1, F_2, A) && \text{(FL-D)} \\
 \text{s.t.} & \sum_{(F_1, F_2, A) \in \mathcal{F}: i \in F_1} Dv(F_1, F_2, A) \leq Dc_i && \forall i \in F \quad (10) \\
 & \sum_{(F_1, F_2, A) \in \mathcal{F}: i \in F_2} u_i(A)v(F_1, F_2, A) \leq f_i && \forall i \in F \quad (11) \\
 & v(F_1, F_2, A) \geq 0 && \forall (F_1, F_2, A) \in \mathcal{F}.
 \end{aligned}$$

As in Sect. 2 the primal-dual algorithm begins with all variables at zero and an empty subset of facilities, A . Before a facility is added to A , we will require that it become tight on both types of dual constraints. To achieve this we will leave each facility in F_1 until it becomes tight on constraint (10), move it into F_2 until it is also tight on constraint (11), and only then move it into A . As before the algorithm terminates once the set A has enough capacity to satisfy the demand, at which point we label our final solution S .

Algorithm 2: Primal-Dual for Single-Demand Facility Location

```

x, y, v ← 0;
F1 ← F;
F2, A ← ∅;
while D(A) > 0 do
  Increase v(F1, F2, A) until a dual constraint becomes tight for facility i;
  if i ∈ F1 then /* i tight on (10) but not on (11) */
    Move i from F1 into F2;
  else /* else i tight on (10) and (11) */
    xi ← ui(A)/D;
    yi ← 1;
    Move i from F2 into A;
S ← A;

```

Clearly Algorithm 2 terminates with a feasible solution to (FL-IP) since all of the demand is assigned to facilities that are fully opened.

Theorem 2 *Algorithm 2 terminates with a solution of cost no greater than $2 \cdot \text{opt}_{FL}$.*

Proof Let ℓ denote the final facility selected by Algorithm 2. By the same reasoning as in Sect. 2 we have

$$D(S \setminus \{\ell\}) > 0 \Rightarrow D - u(S \setminus \{\ell\}) > 0 \Rightarrow u(S \setminus \{\ell\}) < D.$$

The variable $v(F_1, F_2, A)$ is positive only if $A \subseteq S \setminus \{\ell\}$. If a facility is in S then it must be tight on both constraints (10) and (11) so

$$\begin{aligned}
 \sum_{i \in F} (f_i y_i + Dc_i x_i) &= \sum_{i \in S} (f_i + Dc_i x_i) \\
 &= \sum_{i \in S} \left[\sum_{(F_1, F_2, A) \in \mathcal{F}: i \in F_2} u_i(A)v(F_1, F_2, A) + x_i \right] \sum_{(F_1, F_2, A) \in \mathcal{F}: i \in F_1} Dv(F_1, F_2, A),
 \end{aligned}$$

as in Sect. 2 we can simply reverse the order of summation to get

$$\sum_{i \in F} (f_i y_i + D c_i x_i) = \sum_{(F_1, F_2, A) \in \mathcal{F}} v(F_1, F_2, A) \left[\sum_{i \in S \cap F_2} u_i(A) + \sum_{i \in S \cap F_1} D x_i \right].$$

Recall that at the last step of Algorithm 2, facility ℓ was assigned $D(S \setminus \{\ell\})$ amount of demand. Since $D(A)$ only gets smaller as the algorithm progresses, we have that regardless of what summation above the facility ℓ is in, it contributes no more than $D(A)$. All of the other terms can be upper bounded by the actual capacities and hence

$$\begin{aligned} \sum_{i \in F} (f_i y_i + D c_i x_i) &= \sum_{(F_1, F_2, A) \in \mathcal{F}} v(F_1, F_2, A) [u(S \setminus \{\ell\}) - u(A) + D(A)] \\ &< \sum_{(F_1, F_2, A) \in \mathcal{F}} 2D(A)v(F_1, F_2, A) \leq 2 \cdot \text{opt}_{FLD}, \end{aligned}$$

where the inequality above follows from the observation made earlier. □

4 Single-item lot-sizing with linear holding costs

In the single-item lot-sizing problem, one is given a planning period consisting of time periods $F := \{1, \dots, T\}$. For each time period $t \in F$, there is a demand, d_t , and a potential order with capacity u_t , which costs f_t to place, regardless of the amount of product ordered. At each period, the total amount of product left over from the previous period plus the amount of product ordered during this period must be enough to satisfy the demand of this period. Any remaining product is held over to the next period, but incurs a cost of h_t per unit of product stored. Also, we may consider a per-unit production cost associated with the amount of demand requested for a given order, but for now we will ignore this possibility. If we let

$$h_{st} = \sum_{r=s}^{t-1} h_r$$

and set $h_{tt} = 0$ for all $t \in F$, then we obtain a standard IP formulation for this problem as follows

$$\text{opt}_{LS} := \min \sum_{s=1}^T f_s y_s + \sum_{s=1}^T \sum_{t=s}^T h_{st} d_t x_{st} \tag{LS-IP}$$

$$\text{s.t. } \sum_{s=1}^t x_{st} = 1 \quad \forall t \tag{12}$$

$$\sum_{t=s}^T d_t x_{st} \leq u_s y_s \quad \forall s \tag{13}$$

$$\begin{aligned}
 x_{st} &\leq y_s & \forall s \leq t \\
 y_s &\in \{0, 1\} & \forall s \\
 x_{st} &\geq 0 & \forall s \leq t.
 \end{aligned} \tag{14}$$

where the y_s variables indicate if an order has been placed at time period s , and the x_{st} variables indicate what fraction of the demand d_t is being satisfied from product ordered during time period s . This formulation once again suffers from a bad integrality gap, which can be demonstrated by the same example as in the previous two sections. We introduce new flow-cover inequalities to strengthen this formulation.

The basic idea is similar to the inequalities used in Sects. 2 and 3. We would like to consider a subset of orders, A , where even if we place all the orders in A and use these orders to their full potential, there is still unmet demand. In the previous cases, the amount of unmet demand was $D(A) = D - u(A)$. Now, however, that is not quite true, since each order s is capable of serving only the demand points t where $t \geq s$. Instead, we now also consider a subset of demand points B , and define $d(A, B)$ to be the total unmet demand in B , when the orders in A serve as much of the demand in B as possible. More formally

$$d(A, B) := \min d(B) - \sum_{s \in A} \sum_{t \geq s: t \in B} d_t x_{st} \tag{RHS-LP}$$

$$\text{s.t. } \sum_{s=1}^t x_{st} \leq 1 \quad \forall t \tag{15}$$

$$\sum_{t=s}^T d_t x_{st} \leq u_s \quad \forall s \tag{16}$$

$$x_{st} \geq 0 \quad \forall s \leq t.$$

As before, we would also like to restrict the capacities of the orders not in A . To do this, we define

$$u_s(A, B) := d(A, B) - d(A \cup \{s\}, B), \tag{17}$$

which is the decrease in remaining demand that would result if order s were added to A . (This reduces to the same $u_s(A)$ as defined in the previous sections when considered in the framework of the earlier problems.) We once again partition the remaining orders in $F \setminus A$ into two sets, F_1 and F_2 , and count the contribution of orders in F_1 as $\sum_t d_t x_{st}$ and orders in F_2 as $u_s(A, B) y_s$. This leads to the following LP, where once again \mathcal{F} is the set of all 3-tuples that partition F into three sets.

$$\text{opt}_{LSP} := \min \sum_{s=1}^T f_s y_s + \sum_{s=1}^T \sum_{t=s}^T h_{st} d_t x_{st} \tag{LS-P}$$

$$\begin{aligned}
 \text{s.t. } & \sum_{\substack{s \in F_1, \\ t \in B}} d_t x_{st} + \sum_{s \in F_2} u_s(A, B) y_s \geq d(A, B) \quad \forall (F_1, F_2, A) \in \mathcal{F}, B \subseteq F \quad (18) \\
 & x_{st}, y_s \geq 0 \quad \forall s \leq t.
 \end{aligned}$$

Before we demonstrate the validity of this LP, we must first introduce some notation and associated machinery. For a demand set, B , we define

$$e(B) := \sum_{t \in B} d_t - \sum_{t \in B} \sum_{s=1}^t d_t x_{st},$$

which is the amount of demand unsatisfied in B with respect to a given solution. We define $\text{Fill}(A, B)$ to be the following procedure that describes how to assign demand from B to orders in A . We consider the orders in A in arbitrary order, and for each order we serve as much demand as possible, processing demands from earliest to latest.

Unlike in the previous two sections, a given order is only able to serve a subset of demand points, but we will show that the assignment resulting from the Fill procedure is in fact a maximal assignment.

Lemma 1 *If we start from an empty demand assignment and run $\text{Fill}(A, B)$, then we obtain a demand assignment such that $e(B) = d(A, B)$. Thus, Fill produces an assignment that is optimal for (RHS-LP).*

Proof Consider the latest time period $t \in B$ where $e(\{t\}) > 0$. Let $F_1 := \{1, \dots, t\}$ and $F_2 := \{t + 1, \dots, T\}$. All orders $A \cap F_1$ must be serving up to capacity, since otherwise they could have served more of demand d_t . Furthermore, these orders must only be serving demand in $B \cap F_1$, since Fill would have them finish the demand at t before assigning later demand points. Since none of the orders in $A \cap F_2$ can serve demand in $B \cap F_1$, we can conclude that a maximal amount of this demand is being served by orders in A . Finally all of the demand in $B \cap F_2$ is being served by the way t was chosen, thus we have a maximal assignment. \square

Lemma 2 *For any order $s \in F$ and demand subset $B \subseteq F$, and sets $U \subseteq V \subseteq F \setminus \{s\}$ we have*

$$u_s(V, B) \leq u_s(U, B).$$

Proof We can calculate $u_s(U, B)$ by running $\text{Fill}(U \cup \{s\}, B)$ where order s is considered last. Then $u_s(U, B)$ is equal to the demand served by s , since the demand assigned to the orders in U is maximal. Similarly we can calculate $u_s(V, B)$ by running $\text{Fill}(V \cup \{s\}, B)$, again considering s last. If we start by considering all the orders in U in the same order as we did previously, then the orders in U will have the same demand assignment. This means that the demand available when s is considered will be a subset of the demand available when s was considered previously. Hence the demand s serves can be at most what s served previously, and so $u_s(V, B) \leq u_s(U, B)$. \square

Lemma 3 Any feasible solution to (LS-IP) is a feasible solution to (LS-P).

Proof Consider a feasible integer solution (x, y) to (LS-IP) and let $S := \{s : y_s = 1\}$. Now for any $(F_1, F_2, A) \in \mathcal{F}$ and $B \subseteq F$ we know

$$\sum_{\substack{s \in F_1, \\ t \in B}} d_t x_{st} \geq d((F_2 \cap S) \cup A, B),$$

since there is no way to assign demand from B to orders in $(F_2 \cap S) \cup A$ without leaving at least $d((F_2 \cap S) \cup A, B)$ amount of demand unfulfilled. Thus at least that amount of demand must be served by the other orders in S , namely those in F_1 . Let $k := |F_2 \cap S|$ and let s_1, \dots, s_k denote the elements of that set in some order. Furthermore let $S_i := \{s_1, \dots, s_i\}$ for each $1 \leq i \leq k$, so $S_k = F_2 \cap S$. Then by repeated use of (17) we have

$$\begin{aligned} \sum_{\substack{s \in F_1, \\ t \in B}} d_t x_{st} &\geq d((F_2 \cap S) \cup A, B) \\ &= d(((F_2 \cap S) \cup A) \setminus S_1, B) - u_{s_1}(((F_2 \cap S) \cup A) \setminus S_1, B) \\ &= d(((F_2 \cap S) \cup A) \setminus S_k, B) - \sum_{i=1}^k u_{s_i}(((F_2 \cap S) \cup A) \setminus S_i, B) \\ &= d(A, B) - \sum_{i=1}^k u_{s_i}(((F_2 \cap S) \cup A) \setminus S_i, B) \\ &\geq d(A, B) - \sum_{s \in F_2 \cap S} u_s(A, B) \\ &= d(A, B) - \sum_{s \in F_2} u_s(A, B) y_s, \end{aligned}$$

where the last inequality follows by Lemma 2. Thus (x, y) satisfies all of the flow-cover inequalities (18). □

The dual of (LS-P) is

$$\text{opt}_{\text{LSD}} := \max_{(F_1, F_2, A) \in \mathcal{F}} d(A, B) v(F_1, F_2, A, B) \tag{LS-D}$$

$$\text{s.t.} \quad \sum_{\substack{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F: \\ s \in F_1, t \in B}} v(F_1, F_2, A, B) \leq h_{st} \quad \forall s \leq t \tag{19}$$

$$\sum_{\substack{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F: \\ s \in F_2}} u_s(A, B) v(F_1, F_2, A, B) \leq f_s \quad \forall s \tag{20}$$

$$v(F_1, F_2, A, B) \geq 0 \quad \forall (F_1, F_2, A) \in \mathcal{F}, B \subseteq F,$$

where we simply divided constraint (19) by d_t .

Just as in the previous two sections, the primal-dual algorithm initializes the variables to zero and the set A to the empty set. As in Sect. 3, we initialize F_1 to be the set of all orders, and an order will become tight first on constraint (19), when it will be moved to F_2 , and then tight on (20), when it will be moved to A . Unlike in Sect. 3, however, constraint (19) consists of many different inequalities for the same order. This difficulty is averted since all the constraints (19) for a particular order will become tight at the same time, as is proved below in Lemma 4. This is achieved by slowly introducing demand points into the set B . Initially, B will consist only of the last demand point, T . Every time an order becomes tight on all constraints (19), it is moved from F_1 into F_2 , and the demand point of that time period is added to B . In this way we always maintain that F_1 is a prefix of F , and B is the complementary suffix. When an order s becomes tight on constraint (20), we move it to A and assign demand to it by running the procedure $\text{Fill}(s, B)$. Additionally we create a *reserve set* of orders, R_s , for order s , that consists of all orders earlier than s that are not in F_1 at the time s was added to A . Finally, once all of the demand has been assigned to orders, we label the set of orders in A as our current solution, S^* , and now enter a clean-up phase. We consider the orders in the reverse order in which they were added to A , and for each order, s , we check to see if there is enough remaining capacity of the orders that are in both the reserve set and our current solution, $S^* \cap R_s$, to take on the demand being served by s . If there is, then we reassign that demand to the orders in $S^* \cap R_s$ arbitrarily and remove s from our solution S^* . When the clean-up phase is finished we label the nodes in S^* as our final solution, S .

Lemma 4 *All of the constraints (19) for a particular order become tight at the same time, during the execution of Algorithm 3.*

Proof We instead prove an equivalent statement: when demand t is added to B , then for any order $s \leq t$ the slack of the constraint (19) corresponding to s and demand t' is h_{st} for any $t' \geq t$. This statement implies the lemma by considering $s = t$, which implies all constraints (19) for s become tight at the same time. We prove the above statement by (backwards) induction on the demand points. The case where $t = T$ clearly holds, since this demand point is in B before any dual variable is increased, and hence the slack of constraint (19) for order s and demand T is h_{sT} . Now assume the statement holds for some $t \leq T$. If we consider order $s = t - 1$ then by the inductive hypothesis the slack of all constraints (19) for s and demand $t' \geq t$ is h_{st} . Hence the slack of all constraints (19) for orders $s' \leq s$ decreases by h_{st} between the time t is added to B and when $t - 1$ is added to B . Then by the inductive hypothesis again, we have that for any order $s' \leq s$ and any demand $t' \geq t$, when $t - 1$ is added to B the slack of the corresponding constraint (19) is

$$h_{s't} - h_{st} = \sum_{r=s'}^{t-1} h_r - \sum_{r=s}^{t-1} h_r = \sum_{r=s'}^{t-1} h_r - h_{t-1} = \sum_{r=s'}^{t-2} h_r = h_{s',t-1}.$$

Hence the statement also holds for $t - 1$. □

Algorithm 3: Primal-Dual for Single-Item Lot-Sizing

```

 $x_{st}, y_s \leftarrow 0;$ 
 $F_1 \leftarrow F;$ 
 $F_2, A \leftarrow \emptyset;$ 
 $B \leftarrow \{T\};$ 
while  $d(A, F) > 0$  do
    Increase  $v(F_1, F_2, A, B)$  until dual constraint becomes tight for order  $s$ ;
    if  $s \in F_1$  then                                /*  $s$  tight on (19) but not on (20) */
        Move  $s$  from  $F_1$  into  $F_2$ ;
         $B \leftarrow F \setminus F_1$ ;
    else                                              /* else  $s$  tight on (19) and (20) */
         $y_s \leftarrow 1;$ 
        Fill( $s, B$ );
        Move  $s$  from  $F_2$  into  $A$ ;
         $R_s \leftarrow \{r \in F \setminus F_1 : r < s\};$ 
 $S^* \leftarrow A$                                      /* start clean-up phase */;
for  $s \leftarrow$  last order added to  $A$  to first order added to  $A$  do
    if remaining capacity of orders in  $S^* \cap R_s$  is enough to serve demand of  $s$  then
        Remove  $s$  from solution  $S^*$ ;
         $y_s, x_{st} \leftarrow 0$                         /* unassign demand of  $s$  */;
        Fill( $S^* \cap R_s, F$ )                          /* reassign to reserve orders */;
 $S \leftarrow S^*$ ;

```

Define ℓ to be

$$\ell = \ell(F_1, F_2, A) := \max\{s : s \in S \cap F_2\} \cup \{0\},$$

so ℓ is the latest order in the final solution that is also in F_2 , for a given partition, or if there is no such order then ℓ is 0, which is a dummy order with no capacity.

Lemma 5 Upon completion of Algorithm 3, for any F_1, F_2, A, B such that $v(F_1, F_2, A, B) > 0$, we have

$$\sum_{s \in S \cap F_2} u_s(A, B) + \sum_{s \in S \cap F_1} \sum_{t \in B: t \geq s} d_t x_{st} < d(A, B) + u_\ell(A, B).$$

Proof First we consider the case when $S \cap F_2 = \emptyset$. Here the first summation is empty, so we just need to show the bound holds for the second summation. We know that any order $s \in S \cap F_1$ is not in the reserve set for any order in A . This follows since F_1 decreases throughout the course of the algorithm, so since s is in F_1 at this point, then clearly s was in F_1 at any point previously, in particular when any order in A was originally added to A . Hence no demand that was originally assigned to an order in A was ever reassigned to an order in F_1 . But from Lemma 1 we know that only an amount $d(A, B)$ of demand from demand points in B is not assigned to orders in A , thus orders in F_1 can serve at most this amount of demand from demand points in B and we are done.

Otherwise it must be the case that $S \cap F_2 \neq \emptyset$, hence ℓ corresponds to a real order that was not deleted in the clean-up phase. By the way ℓ was chosen we know that any

other order in $S \cap F_2$ is in an earlier time period than ℓ , and since these orders were moved out of F_1 before ℓ was added to A , they must be in the reserve set R_ℓ . However, since ℓ is in the final solution, it must be the case that when ℓ was being considered for deletion the orders in the reserve set that were still in the solution did not have sufficient capacity to take on the demand assigned to ℓ . Thus if we let x' denote the demand assignment during the clean-up phase when ℓ was being considered, then

$$\sum_{s \in (S \cap F_2) \setminus \{\ell\}} u_s(A, B) \leq u((S \cap F_2) \setminus \{\ell\}) < \sum_{\substack{s \in S \cap F_2 \\ t \in B: t \geq s}} d_t x'_{st}.$$

None of the orders in A had been deleted when ℓ was being considered for deletion, so only an amount $d(A, B)$ of the demand in B was being served by orders outside of A . As argued previously, none of the orders in F_1 took on demand being served by orders in A , but they also did not take on demand being served by orders in $S \cap F_2$, since these orders were never deleted. Thus we can upper bound the amount of demand that orders from $S \cap F_2$ could have been serving at the time order ℓ was being considered for deletion as follows

$$\sum_{\substack{s \in S \cap F_2 \\ t \in B: t \geq s}} d_t x'_{st} \leq d(A, B) - \sum_{\substack{s \in S \cap F_1 \\ t \in B: t \geq s}} d_t x_{st}.$$

The desired inequality is obtained by rearranging terms and adding $u_\ell(A, B)$ to both sides. □

We are now ready to analyze the cost of the solution.

Theorem 3 *Algorithm 3 terminates with a solution of cost no greater than $2 \cdot \text{opt}_{LS}$.*

Proof As in the previous two sections, we can use the fact that all of the orders in the solution are tight on all constraints (19) and (20).

$$\begin{aligned} & \sum_{s \in S} \left[f_s + \sum_{t=s}^T d_t h_{st} x_{st} \right] \\ &= \sum_{s \in S} \left[\sum_{\substack{(F_1, F_2, A) \in \mathcal{F}, \\ B \subseteq F: s \in F_2}} u_s(A, B) v(F_1, F_2, A, B) \right. \\ & \quad \left. + \sum_{t=s}^T d_t x_{st} \sum_{\substack{(F_1, F_2, A) \in \mathcal{F}, \\ B \subseteq F: s \in F_1, t \in B}} v(F_1, F_2, A, B) \right] \\ &= \sum_{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F} v(F_1, F_2, A, B) \left[\sum_{s \in S \cap F_2} u_s(A, B) + \sum_{s \in S \cap F_1} \sum_{t \in B: t \geq s} d_t x_{st} \right]. \end{aligned}$$

Now we can apply Lemma 5 and achieve the desired result:

$$\begin{aligned} \sum_{s \in S} \left[f_s + \sum_{t=s}^T d_t h_{st} x_{st} \right] &< \sum_{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F} v(F_1, F_2, A, B) [d(A, B) + u_\ell(A, B)] \\ &\leq \sum_{(F_1, F_2, A) \in \mathcal{F}, B \subseteq F} 2d(A, B)v(F_1, F_2, A, B) \\ &\leq 2 \cdot \text{opt}_{LSD}. \end{aligned}$$

□

4.1 Time-dependent production costs

It is worth noting that Algorithm 3 and its corresponding analysis require almost no changes to handle the case of time-dependent, per-unit productions costs. The only change this introduces in the dual program (LS-D) is simply the addition of the term p_s to the right-hand-side of constraint (19). But this simply adds the same term to all constraints (19) for a particular order, so it is straightforward to adjust the algorithm so that Lemma 4 remains true, which is that all constraints (19) become tight for an order at the same time. We simply change the way that the set B changes over time, so that we add a demand point t to B precisely when the slack of all constraints (19) corresponding to the order at t , are equal to p_t . When $p_t = 0$, this is precisely what the algorithm did before and at the same time all constraints (19) become tight, and if $p_t > 0$ then they will all become tight p_t time units later in the algorithm. The rest of the analysis is the same as before.

5 Computational results

In this section we analyze the performance of Algorithm 3 compared to other reasonable heuristics for the single-item lot-sizing problem, as well as explore how well this algorithm can be used to strengthen the underlying formulation. We generated two different sets of problem data. The first set mirrors the experiments done by Pochet and Wolsey [16] for a nearly identical problem, with the one difference being that we do not require the capacities to be non-decreasing over time. This set of data formed problem instances where capacity was rather constrained, so we also used a second set of instances where capacity was much less-constrained. In both cases, all of the data distributions were uniform, with the ranges shown in Table 1. As done by Pochet and Wolsey, to ensure feasibility, we make the added assumption that the demand for any period is no more than the capacity for that period. This assumption is in essence without loss of generality, since any feasible instance can be expressed in this form by the following transformation. For any period where the demand exceeds the capacity, we know that this excess must be served by earlier orders, so we may simply move the excess demand to the previous period, and add the cost of holding this demand over

Table 1 Ranges for uniform data distributions for the two generated data sets

Distribution	Demand	Capacity	Order cost	Holding cost
(a)	[6, 35]	[16, 25]	[16, 20]	[0.01, 0.05]
(b)	[0, 9]	[16, 38]	[16, 33]	[1, 4]

this period to the objective function. We varied the number of time periods from 40 to 1000, and for each value generated ten instances according to the above distributions.

We now describe the heuristics that were used to compare against the performance of the primal-dual algorithm.

1. *GREEDY*, where orders are selected greedily based upon a notion of cost-effectiveness. At any given time, all orders that have not yet been selected are evaluated, and the one with the cheapest cost per demand served is added to the solution. To calculate this cost, we assume an order processes the remaining unsatisfied demand in order from earliest to latest, just as in the *Fill* procedure of Sect. 4. We then take the holding cost associated with this order serving all of this demand, add it to the ordering cost, and then divide by how much demand would be served. The order which has the lowest value is selected, and the demand it would serve is assigned to it.
2. *RR*, where orders are selected based upon a randomized rounding of the optimal LP solution. Here the LP is first solved, and then each order s is selected to be opened with probability y_s . The resulting ordering cost for this process is the same in expectation as the ordering cost of the LP, but this may lead to an infeasible solution. To fix this, we process the orders in reverse chronological order starting at time period T , and perform the randomized rounding as described above. If an order is added to the solution then it is assigned all of the demand that the *Fill* procedure would give it. If at any point we require an order to be open in order to be able to serve the remaining demand, then it is opened without any randomization.
3. *PART*, where the problem is partitioned into many subproblems, and for each subproblem the optimal integer solution is found. The main restriction with finding the optimal integer solution with commercial solvers is that the running time blows up very quickly as the number of time periods increases. Breaking a problem down into more manageable pieces makes finding the optimal integer solution for these parts a feasible option. To this end, first the optimal LP solution is found, and then the amount of inventory at the end of each period is computed. The time period is then broken into intervals, where each interval starts and ends with no inventory in the optimal LP solution. For each of these intervals, the optimal integer solution is found, and we combine all these solutions to produce a solution to the original problem.

The results of running each of the above heuristics along with the primal-dual algorithm on both sets of generated problem instances are summarized in Fig. 1. In addition a lower bound on the value of the optimal integer solution is shown which is simply the value of the dual solution generated by the primal-dual algorithm. Comput-

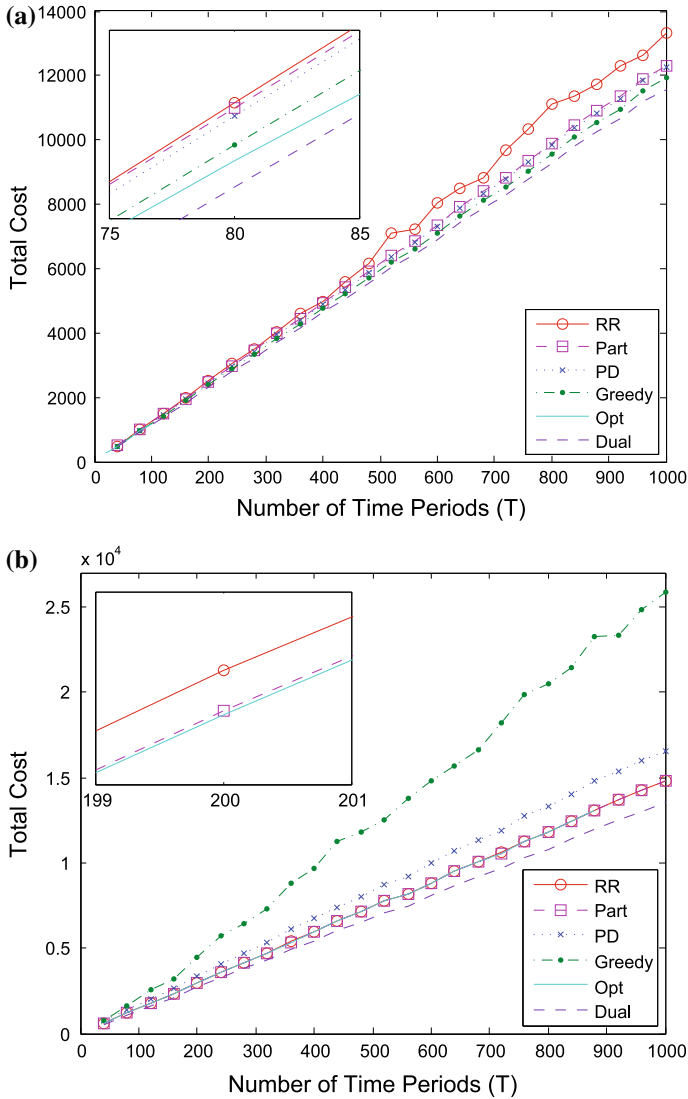


Fig. 1 The results of running the heuristics on all of the generated problem instances, along with the lower bound (Dual) generated by the primal-dual algorithm (PD). The optimal solution values (Opt) are shown for the number of time periods it could be reasonably computed, which is approximately 100 for (a) and 200 for (b)

ing the actual value of the optimal integer solution would have been computationally intractable for how large we took the number of time periods to be.

We see from Fig. 1a that surprisingly the *GREEDY* heuristic performed the best of all, though the primal-dual algorithm was fairly close. The good performance of *GREEDY* may be attributed to the way the data for these problems were distributed, which resulted in needing to open a large number of orders to satisfy all the demand.

Table 2 Numerical results for adding generated flow-cover inequalities to formulation

Formulation	Number of time periods				
	20	40	60	80	100
<i>Original</i>					
Gap	1.045	1.027	1.024	1.022	1.018
Nodes	49	232	5898	$4.802 \cdot 10^4$	$8.076 \cdot 10^5$
Time(s)	0.09531	0.5109	10.78	132.6	3142
<i>Flow-Cover</i>					
Gap	1.043	1.025	1.022	1.019	1.016
Nodes	7	139	3248	$1.999 \cdot 10^4$	$2.187 \cdot 10^4$
Time(s)	0.09844	0.7328	16.20	169.0	2871

It would be expected that in an optimal integer solution to such a problem, most of the orders opened would be serving close to their capacity. However, with problem instances that have capacity less constrained, it would seem likely that the myopic selection routine of *GREEDY* could backfire, which is exactly what we see in Fig. 1a. However, the primal-dual algorithm still manages to remain competitive even though, like the *GREEDY* heuristic, every order it opens was initially assigned all the demand available to it. This shows the robustness of the performance of the primal-dual algorithm over different types of problem instances.

An additional benefit of using the primal-dual algorithm is that during the course of its execution it generates a polynomial number of flow-cover inequalities. Even if one requires an optimal integer solution, this approximation-algorithm could still be useful in both providing a good initial feasible solution, as well as a number of inequalities to strengthen the formulation. This could potentially both decrease the integrality gap as well as reduce the number of branch-and-bound nodes and total time required by the solver. For time periods ranging from 20 to 100 we compared the result of adding the generated flow-cover inequalities to the original formulation, the results of which are summarized in Table 2.

The gap for the generated problem instances started off rather small, and the added flow-cover inequalities reduced it by only a very modest amount. There was, however, a fairly significant reduction in the number of branch-and-bound nodes required once the flow-cover inequalities had been added to the formulation, which amounted to a about a factor of 4 difference. This did not result in a great decrease in the amount of time required to solve, although some time savings did occur. It seems likely though that for instances with larger gaps (especially for those significantly larger than 2) the effect of adding the flow-cover inequalities found by the primal-dual algorithm will be much more dramatic.

Acknowledgments We would like to thank Retsef Levi for many helpful discussions.

References

1. Aardal, K., Pochet, Y., Wolsey, L.A.: Capacitated facility location: valid inequalities and facets. *Math. Oper. Res.* **20**(3), 562–582 (1995)
2. Agrawal, A., Klein, P., Ravi, R.: When trees collide: an approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.* **24**(3), 440–456 (1995)
3. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. *J. Assoc. Comput. Mach.* **48**(5), 1069–1090 (2001)
4. Bar-Yehuda, R., Even, S.: A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algorithms* **2**(2), 198–203 (1981)
5. Bertsimas, D., Teo, C.P.: From valid inequalities to heuristics: a unified view of primal-dual approximation algorithms in covering problems. *Oper. Res.* **46**(4), 503–514 (2003)
6. Carnes, T., Shmoys, D.: Primal-dual schema for capacitated covering problems. In: *Integer Programming and Combinatorial Optimization*, pp. 288–302 (2008)
7. Carr, R.D., Fleischer, L., Leung, V.J., Phillips, C.A.: Strengthening integrality gaps for capacitated network design and covering problems. In: *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 106–115 (2000)
8. Chuzhoy, J., Naor, J.: Covering problems with hard capacities. *SIAM J. Comput.* **36**(2), 498–515 (2006)
9. Chvátal, V.: A greedy heuristic for the set-covering problem. *Math. Oper. Res.* **4**(3), 233–235 (1979)
10. Even, G., Levi, R., Rawitz, D., Schieber, B., Shahar, S., Sviridenko, M.: Algorithms for capacitated rectangle stabbing and lot sizing with joint set-up costs. In: *Submitted to ACM Transactions on Algorithms* (2007)
11. Gandhi, R., Halperin, E., Khuller, S., Kortsarz, G., Srinivasan, A.: An improved approximation algorithm for vertex cover with hard capacities. *J. Comput. Syst. Sci.* **72**(1), 16–33 (2006)
12. Goemans, M.X., Williamson, D.P.: A general approximation technique for constrained forest problems. *SIAM J. Comput.* **24**(2), 296–317 (1995)
13. Levi, R., Lodi, A., Sviridenko, M.: Approximation algorithms for the multi-item capacitated lot-sizing problem via flow-cover inequalities. In: *Proceedings of the 12th International IPCO Conference, Lecture Notes in Computer Science*, vol. 4513, pp. 454–468, Springer, Berlin (2007)
14. Levi, R., Roundy, R., Shmoys, D.B.: Primal-dual algorithms for deterministic inventory problems. *Math. Oper. Res.* **31**(2), 267–284 (2006)
15. Padberg, M.W., van Roy, T.J., Wolsey, L.A.: Valid linear inequalities for fixed charge problems. *Oper. Res.* **33**(4), 842–861 (1985)
16. Pochet, Y., Wolsey, L.A.: Single item lot-sizing with non-decreasing capacities. *Math. Program.* **121**(1), 123–143 (2009)
17. van Hoesel, C.P.M., Wagelmans, A.P.M.: Fully polynomial approximation schemes for single-item capacitated economic lot-sizing problems. *Math. Oper. Res.* **26**(2), 339–357 (2001)