

Solving Multi-Item Capacitated Lot-Sizing Problems with Setup Times by Branch-and-Cut

Andrew J. Miller ¹, George L. Nemhauser ², and Martin W.P. Savelsbergh ²

July 2000

Abstract

Instances of the multi-item capacitated lot-sizing problem with setup times (MCL) often appear in practice, either in standard form or with additional constraints, but they have generally been difficult to solve to optimality. In MCL demand for multiple items must be met over a time horizon, items compete for a shared capacity, and each setup uses up some of this capacity. In this paper we use results concerning the polyhedral structure of simplified models obtained from a single time period of MCL to obtain strong valid inequalities for MCL. To the best of our knowledge, these inequalities are the first to consider demand for multiple items and the joint capacity restriction simultaneously. We also discuss how to implement these inequalities successfully in a branch-and-cut algorithm. Our computational results suggest that our contributions represent significant progress in solving instances of MCL.

KEYWORDS: Mixed integer programming, cutting planes, production planning, capacitated lot-sizing, setup times

¹CORE, Université Catholique de Louvain, Belgium

²Georgia Institute of Technology, School of Industrial and Systems Engineering, Atlanta, GA 30332-0205, USA

This paper presents research results of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsibility is assumed by the authors.

This research was also supported by NSF Grant No. DMI-9700285 and by Philips Electronics North America.

1 Introduction

We will refer to the multi-item capacitated lot-sizing problem with setup times as MCL. MCL is a standard model that occurs in many production planning applications, either in its standard form or with additional constraints and complications (such as conditional lower bounds on production and the possibility of backorders and/or overtime).

A standard formulation for MCL is

$$\min \quad \sum_{i=1}^P \sum_{j=1}^J p_j^i x_j^i + \sum_{i=1}^P \sum_{j=1}^J q_j^i y_j^i + \sum_{i=1}^P \sum_{j=1}^J h_j^i s_j^i \quad (1)$$

subject to

$$x_j^i + s_{j-1}^i - s_j^i = d_j^i, i = 1, \dots, P, j = 1, \dots, J, \quad (2)$$

$$\sum_{i=1}^P x_j^i + \sum_{i=1}^P t_j^i y_j^i \leq c_j, j = 1, \dots, J, \quad (3)$$

$$x_j^i \leq (c_j - t_j^i) y_j^i, i = 1, \dots, P, j = 1, \dots, J, \quad (4)$$

$$x_j^i, s_j^i \geq 0, i = 1, \dots, P, j = 1, \dots, J, \quad (5)$$

$$y_j^i \in \{0, 1\}, i = 1, \dots, P, j = 1, \dots, J. \quad (6)$$

The number of time periods in the problem is J . The number of items is P ; we let $\mathcal{P} = \{1, \dots, P\}$. The production capacity in period j is given by c_j . The setup time for item i in period j is given by t_j^i , and the demand for item i in period j is d_j^i . We assume that $0 \leq t_j^i < c_j, j = 1, \dots, J, i = 1, \dots, P$. The parameter p_j^i is the unit cost of production of item i during period j , h_j^i is the unit cost of holding stock or inventory of item i at the end of period j , and q_j^i is the fixed setup cost that must be paid before any production of item i can occur in period j . The nonnegative variable x_j^i is continuous and represents the amount of item i produced in period j . The variable s_j^i , also continuous, represents inventory or stock of item i held over at the end of period j , and its nonnegativity ensures that backorders are not permitted. The binary variable y_j^i indicates whether or not we produce item i in period j . In our analysis, we allow $s_0 \geq 0$ to ensure that a feasible solution exists.

Constraints (2) are inventory balance constraints that ensure that demand is met. Constraints (3) enforce the production capacity restriction in each time period, and (4) ensure that an appropriate setup cost is paid whenever production occurs. By letting $d_{jl}^i = \sum_{k=j}^l d_k^i, j \leq l$, we can tighten (4) to

$$x_j^i \leq \min\{c_j - t_j^i, d_{jJ}^i\} y_j^i, i = 1, \dots, P, j = 1, \dots, J. \quad (7)$$

Our study of this model was originally motivated by a production planning project with Philips Electronics North America. There are several areas within Philips in which

methods to solve this model efficiently could provide valuable decision support; these include tactical and strategic planning for facilities that manufacture lighting equipment and home appliances.

In addition, this model has been studied in the literature, and obtaining optimal and sometimes even feasible solutions has been challenging. When setup times are nonzero, even the problem of finding a feasible solution to MCL in which $s_0 = 0$, if such a solution exists, is $\mathcal{NP}$ -complete (see e.g. Garey and Johnson [1979]). Therefore, most research to date has focused on heuristic methods.

Among the first to try to solve MCL were Triguero, Thomas, and McClain [1989], who used a heuristic that employs Lagrangean relaxation to obtain near optimal solutions to MCL. Since the Lagrangean solutions they obtained were not always feasible, they used a production smoothing heuristic that sought to shift production from the Lagrangean solution in order to obtain a feasible production plan. They were able to solve many of their test problems this way; however, with problems that had a tight capacity restriction, they were not always able to find a feasible solution. Since the work of Triguero, Thomas, and McClain, many other researchers have tried to find near-optimal solutions for MCL using heuristic methods; these include Diaby et al. [1992], Tempelmeier and Derstoff [1996], and Katok, et al. [1998]

Pochet and Wolsey [1991] and Belvaux and Wolsey [1998, 2000] have solved instances of MCL and related problems to optimality by strengthening the LP formulation with valid inequalities and then invoking a branch-and-bound algorithm. There are two obvious advantages of using such an approach. The first is that the algorithm, if it has time to terminate, finds a provably optimal solution. The second is that a feasible solution is found if one exists; this characteristic is not shared by the many heuristic methods (such as that proposed by Triguero, Thomas, and McClain). A disadvantage of such an optimization approach is that it can require much time and memory, possibly an indefinite amount of both; however, this disadvantage has been mitigated somewhat in recent years by advances in computer technology and mathematical programming theory. In this paper, we seek to solve MCL by first defining new families of valid inequalities for it, and then using these families within a branch-and-cut algorithm.

All the research of which we are aware concerning valid inequalities for MCL has focused on first relaxing MCL to simpler models, and then defining strong valid inequalities for these relaxations. In the next section, we will summarize this research.

In Section 3, we present results for a single period relaxation of MCL that considers both demand and inventory entering this single period for each item. This relaxation is called PI, for preceding inventory, since the inventory entering the single period is from the *preceding* period of the instance of MCL for which PI is a relaxation. (There are also many other mixed integer programs for which PI provides a relaxation; a few of these are discussed in Miller, et al. [2000c].)

In Section 4, we discuss how to use our results for PI to obtain valid inequalities for

MCL. Our approach allows us to obtain inequalities for MCL that take into account the interaction between capacity and demand for multiple items over multiple time periods. To the best of our knowledge, these are the first such inequalities reported.

Given our results for PI, valid inequalities for MCL are not difficult to define. However, separation for these inequalities, and other factors involved in applying them in a branch-and-cut algorithm, involve several challenging issues. These are discussed in Section 5.

In Section 6, we present computational results on a test set of instances of MCL. We achieve our results by incorporating the developments of the earlier sections into a branch-and-cut algorithm. The results suggest that our contributions are essential for solving MCL to optimality as efficiently as possible.

In concluding, we briefly discuss possible extensions of the concepts explored in this paper to more general lot-sizing problems and other mixed integer programs.

2 Previously Studied Relaxations

2.1 Uncapacitated Relaxation

This relaxation is obtained by not considering the constraints (3), and by replacing (7) with $x_j^i \leq d_{jj}^i y_j^i, j = 1, \dots, J, i = 1, \dots, P$. In this case the convex hull is known. The problem separates by item into P instances of the uncapacitated lot-sizing problem (ULS). Thus the (l, S) inequalities for each item give the convex hull (Barany, Van Roy, and Wolsey [1984]). Van Roy and Wolsey [1987] defined the *path inequalities*, which generalize the (l, S) inequalities for more general lot-sizing and other fixed-charge network flow problems. Computational experience using (l, S) and path inequalities for MCL has been reported in the literature (e.g. Pochet and Wolsey [1991]).

2.2 Single-Item Relaxation

If an instance of MCL has just one product, then it becomes an instance of the single-item capacitated lot-sizing problem (CLS), which has been studied, for example, in Pochet [1988], Leung, et al. [1989], Pochet and Wolsey [1993, 1994], and Miller, et al. [2000b]. Some computational experience for applying the class of inequalities presented by Pochet [1988] for CLS to models with multiple items is reported in Pochet and Wolsey [1991]. To date, however, in solving multi-item models such as MCL, the (l, S) inequalities have often been the most effective known class.

We note here that if we replace (3) by

$$\sum_{i=1}^P y_j^i \leq 1, j = 1, \dots, J, \tag{8}$$

we obtain a model with a significantly different structure than MCL. Results from single-period relaxations such as CLS *have* proven effective in solving problems with a mode constraint (8); such problems are often called small bucket problems. Additional results for small bucket models and their relaxations can be found in Constantino [1996] and Belvaux and Wolsey [1998], as well as in the references cited just above.

2.3 Single-Period Relaxation

Constantino [1998] studied a multi-item model with variable lower bounds on production and derived inequalities, based on a single period relaxation, that can be applied to MCL. However, the inequalities he presented were tested only on instances of lower bound models, not on instances of MCL. While his contributions are helpful for solving models with variable lower bounds, their value in helping to solve instances of MCL is questionable, because they do not consider demand. Aside from this work, we are unaware of efforts to derive inequalities for MCL from relaxations that consider multiple items.

3 Single-Period Relaxation with Preceding Inventory

Here we present results obtained by considering a simplified model obtained from a single time period of MCL. We consider a period j with all s_{j-1}^i present (that is, *preceding inventory*); we call this model PI. PI takes into account both the demand for items in j , and the joint capacity constraint. We note here that it is also possible to consider a single period j with all s_j^i present (that is, with *succeeding inventory*). The analyses of these two models yield exactly the same results for MCL, so we will not discuss the latter. Also, for the sake of brevity we will focus our discussion of PI in this paper on results that are necessary to explain the methods we use to solve instances of MCL. Proofs of the propositions of this section, as well as a more complete polyhedral analysis for the general case of PI, can be found in Miller, et al. [2000c]. A detailed analysis of a polynomially solvable special case appears in Miller, et al. [2000a].

Throughout this section we suppress the subscripts $j-1$ and j . PI can be formulated as follows:

$$\min \quad \sum_{i=1}^P p^i x^i + \sum_{i=1}^P q^i y^i + \sum_{i=1}^P h^i s^i \quad (9)$$

subject to

$$x^i + s^i \geq d^i, i = 1, \dots, P, \quad (10)$$

$$\sum_{i=1}^P x^i + \sum_{i=1}^P t^i y^i \leq c, \quad (11)$$

$$x^i \leq (c - t^i)y^i, i = 1, \dots, P, \quad (12)$$

$$x^i, s^i \geq 0, i = 1, \dots, P, \quad (13)$$

$$y^i \in \{0, 1\}, i = 1, \dots, P, \quad (14)$$

We refer to the set of points defined by (10)–(14) as X^{PI} .

One simple family of nontrivial inequalities that induces facets of $\text{conv}(X^{PI})$ is given by

Proposition 1 *If $c > t^i + d^i$, the valid inequality*

$$s^i + d^i y^i \geq d^i \quad (15)$$

induces a facet of $\text{conv}(X^{PI})$, $i = 1, \dots, P$.

These inequalities correspond to the (l, S) inequalities for ULS from Barany, Van Roy, and Wolsey [1984].

In Miller, et al. [2000c] we present two families of valid inequalities for X^{PI} . To state these inequalities, we first define a *cover* of PI to be a set S such that $\lambda = \sum_{i \in S} (t^i + d^i) - c \geq 0$. Similarly, define a *reverse cover* of PI to be a set S such that $\mu = c - \sum_{i \in S} (t^i + d^i) > 0$.

If S is cover of PI and $y^i = 1, i \in S$, then clearly $\sum_{i \in S} s^i \geq \lambda$ is true. Moreover, if $y^{i'} = 0$ for exactly one $i' \in S$, then both

$$\sum_{i \in S} s^i \geq \lambda - t^i$$

and

$$\sum_{i \in S} s^i \geq s^{i'} \geq d^{i'}$$

must hold. Thus

$$\sum_{i \in S} s^i \geq \lambda + \sum_{i \in S} \max\{-t^i, d^i - \lambda\}(1 - y^i)$$

is valid for PI. This observation is the basis for

Proposition 2 (Cover Inequalities) *Given a cover S of PI, order the $i \in S$ such that $t^{[1]} + d^{[1]} \geq \dots \geq t^{[|S|]} + d^{[|S|]}$. Define $\mu^1 = (t^{[1]} + d^{[1]} - \lambda)^+$. Let (T, U) be any partition of $\mathcal{P} \setminus S$, and let (T', T'') be any partition of T . Also, define*

$$D' = \max\{t^{[2]} + d^{[2]}, \max_{i \in U}\{t^i + d^i\}\}.$$

Furthermore, let $j' = \max\{i : t^{[i]} + d^{[i]} \geq \lambda\}$, and let $f(\cdot, \cdot) : \mathbb{R}_+^2 \rightarrow \mathbb{R}$ be defined by

$$f(t, d) = \begin{cases} -d & \text{if } t + d \leq \mu^1, \\ t + (j-1)\lambda - [\mu^1 + \sum_{i=2}^j(t^{[i]} + d^{[i]})] & \text{if } \mu^1 + \sum_{i=2}^j(t^{[i]} + d^{[i]}) \leq t + d \leq \mu^1 + \lambda + \sum_{i=2}^j(t^{[i]} + d^{[i]}), \\ & 1 \leq j \leq j' - 1, \\ j\lambda - d & \text{if } \mu^1 + \lambda + \sum_{i=2}^j(t^{[i]} + d^{[i]}) \leq t + d \leq \mu^1 + \sum_{i=2}^{j+1}(t^{[i]} + d^{[i]}), \\ & 1 \leq j \leq j' - 1, \\ t + (j' - 1)\lambda - [\mu^1 + \sum_{i=2}^{j'}(t^{[i]} + d^{[i]})], & \text{if } t + d \geq \mu^1 + \sum_{i=2}^{j'}(t^{[i]} + d^{[i]}). \end{cases} \quad (16)$$

(In this definition we use the convention that $\sum_{i=2}^1 x_i = 0$, for all arguments x_i .) If $t^{[1]} + d^{[1]} \geq \lambda$, then

$$\begin{aligned} \sum_{i \in S \cup U} s^i &\geq \lambda + \sum_{i \in S} \max\{-t^i, d^i - \lambda\}(1 - y^i) + \\ &\quad \sum_{i \in U} f(t^i, d^i)y^i + \sum_{i \in U} d^i + \\ &\quad \frac{\sum_{i \in S} \min\{t^i + d^i, \lambda\} + \sum_{i \in U} (f(t^i, d^i) + d^i) - \lambda}{(|S| + |U| - 1)D'} \sum_{i \in T'} (x^i - (\mu^1 - t^i)y^i) \end{aligned} \quad (17)$$

is valid for X^{PI} .

Each of T' and U can be seen as a set of elements that are lifted into the simpler form of (17) that results from taking $T' = U = \emptyset$. The development necessary to define the complete form of (17) presented above is complex; details can be found in Miller, et al. [2000c]. We observe here that $f(t, d) \leq 0, \forall t \geq 0, d \geq 0$.

We obtain the second class of new inequalities for PI by considering reverse covers. To motivate this class, let S be a reverse cover of PI, and let i' be some element not in S . If $x^{i'} = (c - t^{i'})y^{i'}$, either

1. $y^{i'} = 0$; or
2. $y^{i'} = 1$ and no capacity is left for $i \in S$; therefore $y^i = 0, i \in S$, and $\sum_{i \in S} s^i \geq \sum_{i \in S} d^i$.

Such reasoning yields

Proposition 3 (Reverse Cover Inequalities) *Let S be a reverse cover of PI , let $T = \mathcal{P} \setminus S$, and let (T', T'') be any partition of T . Then*

$$\sum_{i \in S} s^i \geq \left(\sum_{i \in S} (t^i + d^i) \right) \sum_{i \in T'} y^i - \sum_{i \in S} t^i (1 - y^i) - \sum_{i \in T'} ((c - t^i) y^i - x^i) \quad (18)$$

is valid for X^{PI} .

In the context of the reasoning used to motivate Proposition 3, the set T' in (18) can be seen as a set of “candidates” for the choice of i' . Thus, inequalities (18) are tight only for points in which at most one of $y^i, i \in T'$, takes the value of 1.

4 New Valid Inequalities for MCL

In this section we show how to use the valid inequalities presented for PI to define valid inequalities for the multi-period model MCL .

One trivial way to define inequalities for MCL is simply to consider each time period separately as an instance of PI . However, given a fractional solution to an instance of MCL , it is possible that, for each period, considering only the demand in that period will yield no violated inequalities of the forms (17) and (18). It is certainly even possible that the sum of demand and setup times for all items may be less than the capacity, in which case it is impossible even to define a cover.

Therefore, for a given period $j < J$, we seek to fix some variables $y_k^i, k = j + 1, \dots, \sigma(i), i = 1, \dots, P$ to 0, where, for every $i \in S$, $j \leq \sigma(i) \leq J$. This has the effect of forcing the demand for item i in the periods $j + 1, \dots, \sigma(i)$ to be satisfied in or prior to period j . (Note that fixing x_k^i to 0 has the effect of forcing demand for i in j to be satisfied in an earlier time period, just as fixing y_k^i to 0 does.)

Given that either $x_k^i = 0$ or $y_k^i = 0, k = j + 1, \dots, \sigma(i)$, we can substitute $d_{j, \sigma(i)}^i$ for d_j^i in deriving inequalities of the forms (17) and (18). In order to maintain feasibility, we then “lift” the fixed variables back into these inequalities. To the best of our knowledge, there are no other known inequalities for multi-item models that consider multiple items and multiple time periods simultaneously.

More rigorously, given a time period j , let $\sigma(\cdot)$ be a function from $\{1, \dots, P\}$ to $\{j, \dots, J\}$. Then, given an instance of MCL , a time period j , and a function $\sigma(\cdot)$ defined as above, define a cover for period j with respect to $\sigma(\cdot)$ to be a set $S \subset \mathcal{P}$ such that

$$\lambda = \sum_{i \in S} (t_j^i + d_{j, \sigma(i)}^i) - c_j \geq 0.$$

Proposition 4 (Cover Inequalities for MCL) *Given a time period j and a function $\sigma(\cdot)$ defined as above, let S be a cover of j with respect to $\sigma(\cdot)$. Then order the $i \in S$ such that $t_j^{[1]} + d_{j,\sigma([1])}^{[1]} \geq \dots \geq t_j^{[|S|]} + d_{j,\sigma([|S|])}^{[|S|]}$. Define $\mu^1 = t_j^{[1]} + d_{j,\sigma([1])}^{[1]} - \lambda$. Let (T, U) be any partition of $\mathcal{P} \setminus S$, and let (T', T'') be any partition of T . Also, let*

$$D' = \max\{t_j^{[2]} + d_{j,\sigma([2])}^{[2]}, \max_{i \in U}\{t_j^i + d_{j,\sigma(i)}^i\}\}.$$

Finally, define $f(\cdot, \cdot)$ as in Proposition 2, with $t_j^{[i]} = t_j^{[i]}$ and $d_j^{[i]} = d_{j,\sigma([i])}^{[i]}$, $i = 1, \dots, |S|$. If $t_j^{[1]} + d_{j,\sigma([1])}^{[1]} \geq \lambda$, then

$$\begin{aligned} \sum_{i \in S \cup U} s_{j-1}^i &\geq \lambda + \sum_{i \in S} \max\{-t_j^i, d_{j,\sigma(i)}^i - \lambda\}(1 - y_j^i) + \\ &\quad \sum_{i \in U} f(t_j^i, d_{j,\sigma(i)}^i) y_j^i + \sum_{i \in U} d_{j,\sigma(i)}^i + \\ &\quad \frac{\sum_{i \in S} \min\{t_j^i + d_{j,\sigma(i)}^i, \lambda\} + \sum_{i \in U} (f(t_j^i, d_{j,\sigma(i)}^i) + d_{j,\sigma(i)}^i) - \lambda}{(|S| + |U| - 1)D'} \sum_{i \in T'} (x_j^i - (\mu^1 - t_j^i) y_j^i) \\ &\quad - \sum_{i \in S \cup U} \sum_{k=j+1}^{\sigma(i)} d_{k,\sigma(i)}^i y_k^i \end{aligned} \quad (19)$$

is valid for MCL.

Proof: Consider any feasible solution $(\bar{x}, \bar{y}, \bar{s})$ of MCL; we need to show that this solution satisfies (19). The proof proceeds by induction on $\sigma(i)$, $i \in S \cup U$. The basis step consists of showing that $(\bar{x}, \bar{y}, \bar{s})$ satisfies (19) if $\sigma(i) = j$, $i \in S \cup U$; but this follows from Proposition 2.

Now define a partition (S, T, U) of $\mathcal{P}$ as in the statement of Proposition 4. For each $i \in S \cup U$, choose $j(i)$ so that $j \leq j(i) \leq J$. As our inductive step, we need to show that $(\bar{x}, \bar{y}, \bar{s})$ satisfies (19) if $\sigma(i) = j(i)$, $i \in S \cup U$; this will complete the proof. For the inductive hypothesis, we assume that $(\bar{x}, \bar{y}, \bar{s})$ satisfies (19) whenever

1. $\sigma(i) \leq j(i)$, $i \in S \cup U$, and
2. for at least one $i \in S \cup U$, $\sigma(i) < j(i)$.

Let the inequality of the form (19) that has $\sigma(i) = j(i)$, $i \in S \cup U$, be denoted by I^1 . If $\bar{y}_k^i = 0$, $k = j + 1, \dots, \sigma(i)$, $i \in S \cup U$, then $(\bar{x}, \bar{y}, \bar{s})$ satisfies I^1 because of Proposition 2. Thus, in this case, the inductive step is valid and the proposition holds.

If $\bar{y}_k^i = 1$ for some k and i such that $j < k \leq j(i)$, then, for each $i \in S \cup U$, let $k(i) = \min_{k=j+1, \dots, j(i)} \{k : \bar{y}_k^i = 1\}$, or let $k(i) = j(i) + 1$ if $\{k : j + 1 \leq k \leq j(i), \bar{y}_k^i = 1\} = \emptyset$. Let I^2 be the inequality of the form (19) that has $\sigma(i) = k(i) - 1$, $i \in S \cup U$. Now $(\bar{x}, \bar{y}, \bar{s})$ satisfies I^2 by the induction hypothesis. However, the left hand sides of I^1 and I^2 are

the same, and for the point $(\bar{x}, \bar{y}, \bar{s})$, the right hand side of I^2 has value greater than or equal to the value of the right hand side of I^1 . Thus, in this case as well, $(\bar{x}, \bar{y}, \bar{s})$ satisfies I^1 , the inductive step is valid, and the result holds. $\square$

Example 1

Consider an instance of MCL with $P \geq 3$ and $J \geq 3$. Let this instance be defined in part by $c_2 = 13$, by $t_j^1 = t_j^2 = t_j^3 = 1, j = 2, 3$, and by

$$\begin{aligned} d_2^1 &= 7, & d_2^2 &= 4, & d_2^3 &= 6 \\ d_3^1 &= 3, & d_3^2 &= 6, & d_3^3 &= 4 \end{aligned}$$

Fix $j = 2$, and let $\sigma(1) = 2$, $\sigma(2) = 3$, and $\sigma(3) = 2$. Take $S = \{1, 2\}$; thus $\lambda = 1 + 7 + 1 + (4 + 6) - 13 = 6$, $t_j^{[1]} + d_{j, \sigma(i)}^{[1]} = t_2^2 + d_{23}^2 = 11$, and $\mu^1 = 5$. Take $U = \{3\}$ and $T' = \emptyset$; then $j' = 2$ (where j' is defined as in Proposition 2), and

$$f(t, d) = \begin{cases} -d & \text{if } t + d \leq 5, \\ t - 5 & \text{if } 5 \leq t + d \leq 11, \\ 6 - d & \text{if } 11 \leq t + d \leq 12, \\ t - 6 & \text{if } t + d \geq 12. \end{cases}$$

Thus $f(t_2^3, d_2^3) = t_2^3 - \mu^1 = 1 - 5 = -4$. Then (19) yields

$$\begin{aligned} s_1^1 + s_1^2 + s_1^3 &\geq 6 + \max\{-1, 7 - 6\}(1 - y_2^1) + \max\{-1, 10 - 6\}(1 - y_2^2) - 4y_2^3 + 6 - 6y_3^2 \\ &= 12 + (1 - y_2^1) + 4(1 - y_2^2) - 4y_2^3 - 6y_3^2, \end{aligned}$$

a valid inequality for MCL. If we again take $S = \{1, 2\}$ but instead take $T' = \{3\}$ and $U = \emptyset$, then $D' = t_2^1 + d_2^1 = 8$,

$$\frac{\sum_{i \in S} \min\{t_j^i + d_{j, \sigma(i)}^i, \lambda\} - \lambda}{(|S| - 1)D'} = \frac{6 + 6 - 6}{8} = \frac{6}{8},$$

and (19) yields

$$\begin{aligned} s_1^1 + s_1^2 &\geq 6 + \max\{-1, 7 - 6\}(1 - y_2^1) + \max\{-1, 10 - 6\}(1 - y_2^2) + \frac{6}{8}(x_2^3 - 4y_2^3) - 6y_3^2 \\ &= 6 + (1 - y_2^1) + 4(1 - y_2^2) + \frac{6}{8}(x_2^3 - 4y_2^3) - 6y_3^2, \end{aligned}$$

also valid for MCL. $\square$

More examples of cover inequalities will be given in the next section to illustrate certain computational issues. In (19) note that, for each $i \in S \cup T'$, $y_k^i, k = j+1, \dots, \sigma(i)$, are first fixed to 0 and then “lifted” back into the inequality. We can extend (19) to

$$\begin{aligned} \sum_{i \in S \cup U} s_{j-1}^i &\geq \lambda + \sum_{i \in S} \max\{-t_j^i, d_{j,\sigma(i)}^i - \lambda\}(1 - y_j^i) + \\ &\quad \sum_{i \in U} f(t_j^i, d_{j,\sigma(i)}^i) y_j^i + \sum_{i \in U} d_{j,\sigma(i)}^i + \\ &\quad \frac{\sum_{i \in S} \min\{t_j^i + d_{j,\sigma(i)}^i, \lambda\} + \sum_{i \in U} (f(t_j^i, d_{j,\sigma(i)}^i) + d_{j,\sigma(i)}^i) - \lambda}{(|S| + |U| - 1)D'} \sum_{i \in T'} (x_j^i - (\mu^1 - t_j^i) y_j^i) \\ &\quad - \sum_{i \in S \cup U} \sum_{k=j+1}^{\sigma(i)} \min\{d_{k,\sigma(i)}^i y_k^i, x_k^i\}. \end{aligned} \quad (20)$$

This allows us to take the minimum of $d_{k,\sigma(i)}^i \bar{y}_k^i$ and $\bar{x}_k^i$, for $k = j+1, \dots, \sigma(i)$, in separating a given point $(\bar{x}, \bar{y}, \bar{s})$.

Given an instance of MCL, a time period j , and a function $\sigma(\cdot)$ defined as above, define a reverse cover for period j with respect to $\sigma(\cdot)$ to be a set $S \subset \mathcal{P}$ such that

$$\mu = c_j - \sum_{i \in S} (t_j^i + d_{j,\sigma(i)}^i) > 0.$$

Proposition 5 (Reverse Cover Inequalities for MCL) *Given a time period j and a function $\sigma(\cdot)$ defined as above, let S be a reverse cover of j with respect to $\sigma(\cdot)$. Let $T = \mathcal{P} \setminus S$, and let (T', T'') be any partition of T . Then*

$$\begin{aligned} \sum_{i \in S} s_{j-1}^i &\geq (\sum_{i \in S} (t_j^i + d_{j,\sigma(i)}^i)) \sum_{i \in T'} y_j^i - \sum_{i \in S} t_j^i (1 - y_j^i) - \sum_{i \in T''} ((c_j - t_j^i) y_j^i - x_j^i) \\ &\quad - \sum_{i \in S} \sum_{k=j+1}^{\sigma(i)} d_{k,\sigma(i)}^i y_k^i \end{aligned} \quad (21)$$

is valid for MCL.

The proof of this proposition relies on the same observations as that of Proposition 4.

Example 1 (continued)

Again let $j = 2$, take $S = \{2\}$, and let $\sigma(2) = 3$, so that $\mu = 13 - 11 = 2$. Let $T' = \{1, 3\}$. Then (21) yields

$$\begin{aligned} s_1^2 &\geq 11(y_2^1 + y_2^3) - (1 - y_2^2) - (12y_2^1 - x_2^1) - (12y_2^3 - x_2^3) - 6y_3^2 \\ &= x_2^1 - y_2^1 + x_2^3 - y_2^3 - (1 - y_2^2) - 6y_3^2, \end{aligned}$$

a valid inequality for MCL. $\square$

As with cover inequalities, we can extend (21) to

$$\begin{aligned} \sum_{i \in S} s_{j-1}^i &\geq (\sum_{i \in S} (t_j^i + d_{j, \sigma(i)}^i)) \sum_{i \in T'} y_j^i - \sum_{i \in S} t_j^i (1 - y_j^i) - \sum_{i \in T'} ((c_j - t_j^i) y_j^i - x_j^i) \\ &\quad - \sum_{i \in S} \sum_{k=j+1}^{\sigma(i)} \min\{d_{k, \sigma(i)}^i y_k^i, x_k^i\}. \end{aligned} \quad (22)$$

5 Computational Issues and Separation

We now discuss issues that are important in the implementation of our results in a branch-and-cut algorithm. These include defining separation heuristics for the inequalities (19) and (21), which itself involves the development of methods to define $\sigma(\cdot)$ and to build covers and reverse covers, among other issues.

During the branch-and-cut algorithm, valid inequalities, or *cuts*, are stored in a cut pool. Violated cuts can be generated and placed into the cut pool at any node of the branch-and-bound tree. The special case of branch-and-cut in which violated inequalities are generated and added to the cut pool only at the root node is called cut-and-branch. At any node of the branch-and-bound tree, some subset of the cuts in the cut pool is included in the LP formulation that yields the lower bound for that node.

In addition to the valid inequalities that we have defined, in solving instances of MCL it is also helpful to use the well-known (l, S) inequalities (Barany, Van Roy, and Wolsey [1984]). Of course, this is to be expected, given earlier computational studies (for example, Pochet and Wolsey [1991] and Belvaux and Wolsey [1998, 2000]) and some of our results concerning PI, such as Proposition 1 (see also Miller, et al. [2000a]). The (l, S) inequalities can be expressed as

$$s_{k-1}^i + \sum_{j \in \hat{S}} d_{jl}^i y_j^i + \sum_{j \in [k, \dots, l] \setminus \hat{S}} x_j^i \geq d_{kl}^i, \quad (23)$$

where $i \in \mathcal{P}$, $1 \leq k \leq l \leq J$, and $\hat{S}$ is any subset of $[k, \dots, l]$. Separation for these inequalities takes $\mathcal{O}(PT^2)$, and it is thus easy to obtain a lower bound on the optimal solution value of an instance of MCL by adding all binding (l, S) inequalities to the LP formulation. This is important, since the trivial lower bound provided by the LP relaxation of (2)–(6) is often much weaker.

Recall that for (19), we can generalize the inequality by replacing $d_{k, \sigma(i)}^i y_k^i$ with x_k^i , for any subset of $\{(i, k) : k = j + 1, \dots, \sigma(i), i \in S \cup U\}$. A similar statement holds for (21). For the sake of simplicity, we will limit our discussion here to the simpler forms

(19) and (21). However, it is not difficult to extend the concepts we present so that the more general forms of the inequalities—(20) and (22)—can be applied, and we have done so in our computational tests.

When demand and setup times are not both constant, exact separation for cover and reverse cover inequalities for PI (the general single-period relaxation of MCL) involves solving a nonconvex quadratic program or worse. It is certainly no easier to separate for cover and reverse cover inequalities for MCL; thus, it is necessary to develop heuristic methods to separate for these inequalities. We discuss the methods we employ to try heuristically to find violated inequalities. We will assume first that $\sigma(i), i = 1, \dots, P$ has been defined, and discuss the heuristics we use, given this definition of $\sigma(\cdot)$. Finally we will discuss how we define $\sigma(\cdot)$, giving examples illustrating the importance of defining it “well”.

5.1 Separation Heuristic for Cover Inequalities

In separating for a violated cover inequality, we first order the items in some order $[1], \dots, [P]$ (we will specify how we define this order later). We then call the following separation heuristic.

Heuristic 1 *Separation Heuristic for Cover Inequalities*

Step 0 (Initialization) Let

$$i' = \min\{i : \sum_{k=1}^i (t_j^{[k]} + d_{j,\sigma([k])}^{[k]}) > c_j\}.$$

If no such i' exists, terminate; there exists no cover of j with respect to $\sigma(\cdot)$. Otherwise, set $S = \{[1], \dots, [i']\}$. Set

$$\lambda = \sum_{k=1}^{i'} (t_j^{[k]} + d_{j,\sigma([k])}^{[k]}) - c_j,$$

and set

$$\mu^1 = \max_{i \in S} \{t_j^i + d_{j,\sigma(i)}^i\} - \lambda.$$

Set $U = \emptyset$ and $T' = \emptyset$. Set $i = i' + 1$. If $i = P$, go to Step 4; otherwise, go to Step 1.

Step 1 If

$$f(t_j^{[i]}, d_{j, \sigma([i])}^{[i]}) \bar{y}_j^i + d_{j, \sigma([i])}^{[i]} - \bar{s}_{j-1}^{[i]} - \sum_{k=j+1}^{\sigma([i])} d_{k, \sigma(i)}^{[i]} \bar{y}_k^{[i]} > 0,$$

put $[i]$ into U , and go to Step 3. If this does not hold, go to Step 2.

Step 2 If

$$\bar{x}_j^{[i]} > (\mu^1 - t_j^{[i]}) \bar{y}_j^{[i]},$$

put $[i]$ into T' . Go to Step 3.

Step 3 If $i = P$, go to Step 4. Otherwise increment i and go to Step 1.

Step 4 Test if the inequality defined by S , U , and T' is violated. If it is, add it to the cut pool. Terminate. $\square$

Thus, after defining the cover S and defining λ and μ^1 , we loop through $i \in \mathcal{P} \setminus S$, testing to see if putting i first into U and then into T' increases the violation. If we find that doing so does increase the violation, then we put i into U or T' , whichever we have tested. We test U before T' because $(|S| + |U| - 1)D'$, the denominator of the coefficients of $i \in T'$ in cover inequalities, can be quite large. Thus, putting i into U seems likely to yield a stronger inequality, because it will increase the violation for more fractional points.

Clearly the ordering of $i \in \mathcal{P}$ determines the cover S that the heuristic finds. A greedy method to order $i \in \mathcal{P}$ is to order them by nonincreasing values of the function

$$\max\{-t_j^i, d_{j, \sigma(i)}^i - \lambda\}(1 - \bar{y}_j^i) - \bar{s}_{j-1}^i - \sum_{k=j+1}^{\sigma(i)} d_{k, \sigma(i)}^i \bar{y}_k^i, \quad (24)$$

since this is the contribution to the violation of the inequality by $i \in S$. However, we do not know λ a priori. Therefore, we first estimate λ , and then order $i \in \mathcal{P}$ by nonincreasing values of (24). In particular, we

1. order $i \in \mathcal{P}$ by nonincreasing values of

$$(t_j^i + d_{j, \sigma(i)}^i) \bar{y}_j^i \quad (25)$$

and call Heuristic 1; if we do not find a violated inequality, then let $\bar{\lambda}$ be the value of λ determined by the set S as defined in the call to Heuristic 1. Then

2. order $i \in \mathcal{P}$ by nonincreasing values of

$$\max\{-t_j^i, d_{j,\sigma(i)}^i - \bar{\lambda}\}(1 - \bar{y}_j^i) - \bar{s}_{j-1}^i - \sum_{k=j+1}^{\sigma(i)} d_{k,\sigma(i)}^i \bar{y}_k^i \quad (26)$$

and call Heuristic 1.

We typically find more violated inequalities with the second call than with the first. However, we do find violated inequalities with the first call, and since we have to achieve the estimate $\bar{\lambda}$ in order to make the second call, it makes sense to call Heuristic 1 to do so. Note that the first ordering can be computed without estimating λ .

Heuristic 1 runs in $\mathcal{O}(P)$ time. The time to sort the items takes $\mathcal{O}(P \log P)$, and thus dominates the running time of the heuristic.

5.2 Separation Heuristic for Reverse Cover Inequalities

Before describing the heuristic, we note that, given an inequality of the form (21), for each $i' \in T'$,

$$\sum_{i \in S} (t_j^i + d_{j,\sigma(i)}^i) - (c - t_j^{i'}) = t_j^{i'} - \mu.$$

Given an ordering $[1], \dots, [P]$ of $i \in \mathcal{P}$, we call the following heuristic to try to find a violated reverse cover inequality.

Heuristic 2 *Separation Heuristic for Reverse Cover Inequalities*

Step 0 (Initialization) Set $S = [1]$. Set

$$\mu = c - (t_j^{[1]} + d_{j,\sigma([1])}^{[1]}),$$

set $T' = \emptyset$, and set $i' = 1$. Go to Step 1.

Step 1 For $i = [i'] + 1, \dots, [P]$, if $(t_j^i - \mu)\bar{y}_j^i + \bar{x}_j^i > 0$, put i into T' . Go to Step 2.

Step 2 Test if the inequality (21) defined by S and T' is violated. If it is, add it to the cut pool and terminate. If it is not, go to Step 3.

Step 3 If $i' = P$, terminate; the heuristic has failed to find a violated inequality. Otherwise, increment i' , and go to Step 4.

Step 4 Put i' into S . If

$$\sum_{i \in S} (t_j^i + d_{j, \sigma(i)}^i) \geq c_j,$$

terminate; S is no longer a reverse cover, and the heuristic has failed to find a violated inequality. If this does not hold, set

$$\mu = c_j - \sum_{i \in S} (t_j^i + d_{j, \sigma(i)}^i),$$

and go to Step 1. $\square$

Again, the reverse covers formed in this heuristic are greatly affected by the ordering of the elements that the heuristic is given. The natural, greedy way to order $i \in \mathcal{P}$ is by nonincreasing values by which they would contribute to the violation of the inequality, were they in S :

$$t_j^i(\bar{y}_j^i - 1) - \bar{s}_{j-1}^i - \sum_{k=j+1}^{\sigma(i)} d_{k, \sigma(i)}^i \bar{y}_k^i. \quad (27)$$

This ordering enables the heuristic to find violated inequalities often; however, it penalizes items that have a very large value of $d_{j, \sigma(i)}^i$ and fractional values of $\bar{y}_j^i$ that are not integral but are very close to 1. The reason is that the value of $\bar{s}_j^i$ for such items can be comparatively large. Similarly, it penalizes items that have high setup times t_j^i and large but fractional values of $\bar{y}_j^i$. An item i' that is so penalized for either (or both) of these two reasons is often placed close to last in the order specified by (27); because of this, the heuristic puts other items into S and causes S to become a cover before it is able to put i' into S in Step 4. Sometimes, therefore, the heuristic never has the opportunity to put i' into S , even though its inclusion in S would yield a violated inequality. We have therefore also found it useful to order the items in nonincreasing values of

$$(t_j^i + d_{j, \sigma(i)}^i) \bar{y}_j^i. \quad (28)$$

In separating for reverse cover inequalities, we first order the items by (27) and call Heuristic 2; if we do not find and add a violated inequality, we then order $i \in \mathcal{P}$ by (28) and then call Heuristic 2.

Heuristic 2 runs in $\mathcal{O}(P^2)$ time. The time to sort the items is $\mathcal{O}(P \log P)$; thus the time to run Heuristic 2 dominates the sort time.

5.3 Defining $\sigma(\cdot)$

Defining $\sigma(i), i \in \mathcal{P}$, the function that determines how much demand is projected from future periods into the current period for each item, is clearly important in trying to find a violated cover or reverse cover inequality. This definition affects the ordering of the items that we use in the heuristics we have discussed, and therefore affects the covers and reverse covers defined, as well as the values λ , μ^1 , and μ . Clearly in defining $\sigma(\cdot)$, we should be careful of the following:

1. If $\bar{y}_{j+1}^i = \dots = \bar{y}_k^i = 0$, then we should set $\sigma(i) \geq k$. We should project as much demand into the period as possible if it is certain that doing so will not weaken the inequality (19) or (21) by the subtraction of the sum $\sum_{k=j+1}^{\sigma(i)} d_{k,\sigma(i)}^i \bar{y}_k^i$.
2. If, for some $l \geq j+1$,

$$\sum_{k=j+1}^l d_{kl}^i \bar{y}_k^i \geq d_{j+1,l}^i,$$

then we should set $\sigma(i) < l$. We should not project demand in periods $j+1$ through l into j if the sum, $\sum_{k=j+1}^{\sigma(i)} d_{k,\sigma(i)}^i \bar{y}_k^i$, by which this will weaken the resulting inequality is greater than $d_{j+1,l}^i$.

Therefore, a reasonable choice of $\sigma(i)$ is

$$\sigma(i) = \sigma_\alpha(i) = \operatorname{argmax}_{l=j,\dots,J} \{ \alpha d_{j+1,l}^i - \sum_{k=j+1}^l d_{kl}^i \bar{y}_k^i \}, \quad (29)$$

for some $\alpha \in R, 0 < \alpha \leq 1$. (To ensure that $\sigma_\alpha(i)$ is properly defined, we let $d_{j+1,j}^i = 0$ and $\sum_{k=j+1}^j d_{kl}^i \bar{y}_k^i = 0$. Also, if there is more than one maximizer l of the difference in (29), we define $\sigma_\alpha(i)$ to be the minimum (earliest) period l that maximizes this difference.) Note that, given an α , $\sigma_\alpha(i)$ is easy to compute using the above formula. The reason is that the difference to be maximized increases with l until $\sum_{k=j+1}^l \bar{y}_k^i \geq \alpha$,

at which point it stops increasing (as soon as this inequality is strict, the difference in (29) decreases with further increments of l). This last observation implies that an equivalent definition of $\sigma_\alpha(\cdot)$ is

$$\sigma_\alpha(i) = \max_{l=j, \dots, J} \{l : \alpha > \sum_{k=j+1}^l \bar{y}_k^i\}. \quad (30)$$

Defining $\sigma(i), i = 1, \dots, P$, by applying the above formula for some α allows us to find many violated inequalities and substantially close duality gaps. However, we often find violated inequalities for one value of α that we do not find with another. Which α to choose in order to find a violated inequality is not trivially evident, and depends not only on λ but also on the solution $(\bar{x}, \bar{y}, \bar{s})$; that is, the optimal choice of α is part of the separation problem. The next two examples illustrate the role of $\sigma(\cdot)$ and α in defining inequalities for MCL.

Example 2

Consider an instance of MCL with $P \geq 2$ and $J \geq 4$. Let this instance be defined in part by $c_2 = c_3 = 9$, by $t_j^1 = t_j^2 = 0, j = 2, 3, 4$, and by

$$\begin{aligned} d_2^1 &= 5, & d_2^2 &= 4, \\ d_3^1 &= 3, & d_3^2 &= 6, \\ d_4^1 &= 4, & d_4^2 &= 5. \end{aligned}$$

Then let a solution $(\bar{x}, \bar{y}, \bar{s})$ to the LP relaxation be defined in part by

$$\begin{aligned} \bar{y}_2^1 &= 1, & \bar{y}_2^2 &= \frac{3}{4}, \\ \bar{y}_3^1 &= 1, & \bar{y}_3^2 &= \frac{1}{2}, \\ \bar{y}_4^1 &= 1, & \bar{y}_4^2 &= 1, \end{aligned}$$

$$\begin{aligned} \bar{x}_2^1 &= 5, & \bar{x}_2^2 &= 4, \\ \bar{x}_3^1 &= 3, & \bar{x}_3^2 &= 3, \\ \bar{x}_4^1 &= 4, & \bar{x}_4^2 &= 5, \end{aligned}$$

$$\begin{aligned} \bar{s}_1^1 &= 0, & \bar{s}_1^2 &= 3, \\ \bar{s}_2^1 &= 0, & \bar{s}_2^2 &= 3, \\ \bar{s}_3^1 &= 0, & \bar{s}_3^2 &= 0. \end{aligned}$$

It can be checked that this solution satisfies all the (l, S) inequalities for $i = 1, 2$, $2 \leq k \leq l \leq 4$. If we fix $j = 2$ and take $\alpha = 1$, then $\sigma_\alpha(1) = 2$ and $\sigma_\alpha(2) = 3$. Thus we can take $S = \{1, 2\}$, and $\lambda = 5 + (4 + 6) - 9 = 6$. Take $U = T = \emptyset$. Then (19) yields

$$\begin{aligned} s_1^1 + s_1^2 &\geq 6 + \max\{0, 5 - 6\}(1 - y_2^1) + \max\{0, 10 - 6\}(1 - y_2^2) - 6y_3^2 \\ &= 6 + 4(1 - y_2^2) - 6y_3^2. \end{aligned}$$

For $(\bar{x}, \bar{y}, \bar{s})$, the left hand side of this inequality has value 3, while the right hand side has value $6 + 1 - 3 = 4$. Thus this inequality is violated.

Note that if we choose any $\alpha \leq \frac{1}{2}$, then we cannot find a violated cover inequality for this example. $\square$

We next give an example of where it is necessary to take $\alpha < 1$ in order to find a violated cover inequality.

Example 3

Consider an instance of MCL with $P \geq 2$ and $J = 4$. Let this instance be defined in part by $c_2 = c_3 = 15$, by $t_j^1 = t_j^2 = 0, j = 2, 3, 4$, and by

$$\begin{aligned} d_2^1 &= 6, & d_2^2 &= 4, \\ d_3^1 &= 5, & d_3^2 &= 5, \\ d_4^1 &= 4, & d_4^2 &= 6. \end{aligned}$$

Then let a solution $(\bar{x}, \bar{y}, \bar{s})$ to the LP relaxation be defined in part by

$$\begin{aligned} \bar{y}_2^1 &= \frac{9}{22}, & \bar{y}_2^2 &= 1, \\ \bar{y}_3^1 &= 0, & \bar{y}_3^2 &= 0, \\ \bar{y}_4^1 &= 1, & \bar{y}_4^2 &= \frac{3}{4}, \end{aligned}$$

$$\begin{aligned} \bar{x}_2^1 &= 4\frac{1}{2}, & \bar{x}_2^2 &= 10\frac{1}{2}, \\ \bar{x}_3^1 &= 0, & \bar{x}_3^2 &= 0, \\ \bar{x}_4^1 &= 4, & \bar{x}_4^2 &= 4\frac{1}{2}, \end{aligned}$$

$$\begin{aligned} \bar{s}_1^1 &= 6\frac{1}{2}, & \bar{s}_1^2 &= 0, \\ \bar{s}_2^1 &= 5, & \bar{s}_2^2 &= 6\frac{1}{2}, \\ \bar{s}_3^1 &= 0, & \bar{s}_3^2 &= 1\frac{1}{2}. \end{aligned}$$

It can be checked that this solution satisfies all the (l, S) inequalities for $i = 1, 2$, $2 \leq k \leq l \leq 4$. If we fix $j = 2$ and take $\alpha = \frac{1}{2}$, then $\sigma_\alpha(1) = 3$ and $\sigma_\alpha(2) = 3$. Thus we can take $S = \{1, 2\}$, and $\lambda = (6 + 5) + (4 + 5) - 15 = 5$. Take $U = T = \emptyset$. Then (19) yields

$$\begin{aligned} s_1^1 + s_1^2 &\geq 5 + \max\{0, 11 - 5\}(1 - y_2^1) + \max\{0, 9 - 5\}(1 - y_2^2) - 5y_3^1 - 5y_3^2 \\ &= 5 + 6(1 - y_2^1) + 4(1 - y_2^2) - 5y_3^1 - 5y_3^2. \end{aligned}$$

For $(\bar{x}, \bar{y}, \bar{s})$, the left hand side of this inequality has value $6\frac{1}{2}$, while the right hand has value $5 + 6 * \frac{13}{22} = 8\frac{6}{11}$. Thus this inequality is violated.

Now take $\alpha = \frac{4}{5}$. Then again $\sigma_\alpha(1) = 3$, but $\sigma_\alpha(2) = 4$. Thus taking $S = \{1, 2\}$ gives $\lambda = (6 + 5) + (4 + 5 + 6) - 15 = 11$. Again taking $U = T = \emptyset$, (19) yields

$$\begin{aligned} s_1^1 + s_1^2 &\geq 11 + \max\{0, 11 - 11\}(1 - y_2^1) + \max\{0, 15 - 11\}(1 - y_2^2) - 5y_3^1 - 11y_3^2 - 6y_4^2 \\ &= 11 + 4(1 - y_2^2) - 5y_3^1 - 11y_3^2 - 6y_4^2. \end{aligned}$$

For $(\bar{x}, \bar{y}, \bar{s})$, both sides of this inequality have value $6\frac{1}{2}$, and thus this inequality is *not* violated. $\square$

An additional problem complicating the choice of α is that there are some violated inequalities (19) and (21) in which demand from later periods is projected into j much more aggressively for some items than for others. Thus, there are inequalities that can only be found if α is chosen differently for each i . Because of this, we have found it useful to randomize this value for each i . Doing this allows us to compute $\sigma(i) = \sigma_{\alpha_i}(i)$ from (30), with a different value of α_i for each $i \in \mathcal{P}$.

Given a time period j , defining $\sigma_{\alpha_i}(i)$, $i = 1, \dots, P$ using (30) takes $\mathcal{O}(PT)$ time. We can now define the complete separation heuristic that we call at the root node to find violated inequalities for MCL:

Heuristic 3 *Separation Heuristic for Valid Inequalities for MCL*

Step 0 (Separation for (l, S) inequalities) Separate for and add violated (l, S) inequalities. Set $j = 1$, and go to Step 1.

Step 1 Set $ITER = 1$; go to Step 2.

Step 2 Let α_i be chosen randomly for each $i \in \mathcal{P}$ from the uniform distribution on $(0, 1]$. Go to Step 3.

Step 3 (Cover Inequality Separation Loop) Order $i \in \mathcal{P}$ by (25), and call Heuristic

1. If we find a violated inequality, go to Step 4. If we do not, reorder $i \in \mathcal{P}$ by (26) (using the estimate $\bar{\lambda}$ obtained from the first call to Heuristic 1), and call Heuristic 1 again. Go to Step 4.

Step 4 (Reverse Cover Inequality Separation Loop) Order $i \in \mathcal{P}$ by (27), and call Heuristic 2. If we find a violated inequality, go to Step 5. If we do not, reorder $i \in \mathcal{P}$ by (28) and call Heuristic 2. Go to Step 5.

Step 5 If a violated inequality was found in either of steps 3 and 4, or in both steps, go to step 6. If no violated cover or reverse cover inequality was found, go to Step 7.

Step 6 Add the violated inequalities into the cut pool, and go to Step 8.

Step 7 If $ITER = P$, go to Step 8. Otherwise, increment $ITER$, and to to Step 2.

Step 8 If $j = T$, terminate. Otherwise increment j and go to Step 1. $\square$

Thus we add at most one cover and one reverse cover inequality for each period during the execution of Heuristic 3. Note that, in searching for a violated cover or reverse cover inequality in each period, we try at most P assignments of α_i in which this parameter is chosen randomly for each item. Looping through steps 2 through 7 P times is motivated by the fact that the number of possible definitions of the functions of $\sigma(\cdot)$ increases with P . We thus allow the number of possible definitions $\sigma(\cdot)$ that we actually investigate to increase with P as well. While the increase in the number of definitions of $\sigma(\cdot)$ is, of course, not linear, increasing the number of randomized “guesses” for the appropriate value of $\sigma(\cdot)$ linearly with P still allows us to find many inequalities quickly. Moreover, we have also found that no matter how many times we iterate through Steps 2 to 7 at the root node, we do not measurably improve upon the lower bound that we obtain by iterating P times. That is, it seems that P passes through Steps 2 to 7 is sufficient to find “almost” all violated cover and reverse cover inequalities at the root node.

Step 0 of the heuristic takes $\mathcal{O}(PT^2)$. For each j , the most expensive of Steps 2 through 7 are steps 3 and 7, which take $\mathcal{O}(P^2 + PT)$ (that is, the maximum of defining $\sigma_{\alpha_i}(i), i = 1, \dots, P$ or of calling Heuristic 2). Thus, one loop of steps 1 through 8 executes in $\mathcal{O}(P(P^2 + PT))$ time, and the entire separation routine executes in $\mathcal{O}(PT^2 + PT(P^2 + PT)) = \mathcal{O}(PT(P^2 + PT))$ time.

In practice Heuristic 3 operates quickly; for example, it does not, in general, take much more time to call the above heuristic than to resolve the LP after violated cuts are added. This will become apparent from the results of the next section.

6 Computational Results

Here we present the results of our implementation of the concepts developed above. The results suggest that our contributions help to solve multi-item capacitated lot-sizing models more efficiently than was before possible.

The code in which our results are implemented uses the callable library of XPRESS-MP, Release 11.50. XPRESS-MP provides callback functions that allow the user to implement his own branch-and-cut algorithm. It stores valid inequalities, or cuts, in a cut pool. XPRESS-MP also allows the user, at any node of the tree, to add cuts from the cut pool to the matrix, to generate and add violated cuts to the cut pool, and otherwise manage the branch-and-cut process.

The best mathematical programming code that currently exists for solving lot-sizing problems is the academic software *bc-prod* (Belvaux and Wolsey [1998]). This code is built on top of *bc-opt* (Cordier et al. [1999]), which is a general purpose branch-and-cut MIP solver. *bc-opt* uses several classes of inequalities in its branch-and-cut algorithm, including, among others, flow cover inequalities (see Padberg, et al. [1985] and Gu, et al. [1999]), lifted knapsack cover inequalities (see Crowder, et al. [1983], Weismantel [1997], and Gu, et al. [1998]), and path inequalities (Van Roy and Wolsey [1987]—recall that there are a generalization of the (l, S) inequalities). *bc-prod* includes all of the cutting plane features of *bc-opt*. In addition, it has a number of cutting plane routines designed specifically for lot-sizing problems, including implementations of some of the research mentioned in Section 2. These cutting planes have proven to be very effective for the small bucket multi-item problems mentioned in Section 2, but as will be seen, they are not sufficient to solve problems such as MCL as efficiently as possible. Both *bc-prod* and *bc-opt* use the subroutine library of XPRESS-MP.

The best way to test the computational value of our contributions would be to test them within *bc-prod*. In that way we could determine the value added by our inequalities to what is currently known. Unfortunately, because of compatibility issues, we were not able to perform such tests. Instead, therefore, we have compared our implementation directly with *bc-prod* itself. In comparing our results with those of *bc-prod*, it should be remembered that *bc-prod* has many features specifically designed for lot-sizing problems that we have not been able to implement. Nevertheless, our computational results for MCL compare favorably with those of *bc-prod*, as will be seen.

6.1 Test Set

For our test set, we chose instances of MCL that are either taken directly from or modified from models in LOTSIZELIB, a library of lot-sizing problems (Belvaux and Wolsey [1998]). The first six of these MCL instances are taken from LOTSIZELIB and are known as the *trP-J* models. These instances first appeared in Triguero, et al. [1989]. They all have high setup costs and small setup times. Thus, feasible solutions are easy to

obtain, and, once (l, S) cuts are added to the formulation, the duality gaps become very small, even though none of the problems except *tr6-15* are easy to solve to optimality.

In addition to the problems from Triguero, et al., we modified slightly the LOT-SIZELIB models *set1ch* and *pp08a* so that the new models are instances of MCL. (Both *set1ch* and *pp08a* are also in the MIPLIB 3.0 library (see Bixby, Boyd, and Indovina [1992]).) The model *mset1ch* was created by removing the possibility of overtime production from *set1ch*, and the model *mpp08a* was created from *pp08a* by removing the possibility of backorders. In both of these last two models, the setup times for all items are 0.

We have also created new instances by modifying each of the above eight test problems. These new instances, whose names begin with an *x*, were created from their counterparts by setting the setup costs of each item to 0, and increasing the setup time of each item by a constant amount. In these new instances, the duality gaps are much larger, and feasible solutions are much harder to find. Our motivation in doing this is that, in many actual applications, setups complicate planning not primarily because setting up incurs specific costs, but because the time required to perform setups makes good, feasible plans difficult to find.

In creating these new instances, we sought to increase the setup time for each item in such a way that $P\delta t \approx .1 * c$, where δt is the amount of setup time increase and c is the capacity. (In each of our instances, capacity is constant for all time periods.) This yields a choice of 10 as the amount of increase for the setup time of each item to create each of the *xtrP-J* problems. For *xpp08a* and for *xset1ch*, this yields a setup time of 5 for each item (recall that neither *mpp08a* nor *mset1ch* has setup times). However, for *xset1ch*, eliminating the fixed costs and setting $t_j^i = 5, i = 1, \dots, P, j = 1, \dots, J$, yields an instance that solves almost trivially with both *bc-prod* and with our code. Therefore, to make this instance more interesting, we set the setup time to 10 for each item.

In evaluating the effectiveness of a branch-and-cut algorithm for our test set, the ratio $(UB - LB)/UB$, where UB and LB are the best upper and lower bounds found at a given point in the branch-and-bound tree, is not necessarily the best measure for judging how far from optimality the algorithm is. This is especially true for the *trP-J* instances, where the high fixed costs cause problems that require hours to solve to have very small values of $(UB - LB)/UB$ after (l, S) cuts are added. We therefore normalize the bounds and use the ratio $(UB - LB)/(UB - LB')$ as a relative measure of the duality gap, where LB' is the value of the LP relaxation that includes all the binding (l, S) cuts. Since the separation problem for the (l, S) inequalities is polynomial, and since both *bc-prod* and our code separate exactly for them, this ratio is always well-defined and a valid measure of the progress made in closing the duality gap left after the addition of the (l, S) cuts.

Some characteristics of each of the problems are listed in Table 1. These include the dimensions of the problem, the value of LB' , and the value of the optimal solution. A *

Table 1: Instances of MCL

Problem	P	T	LB'	Opt
tr6-15	6	15	37201	37721
tr6-30	6	30	60946	61746 [†]
tr12-15	12	15	73848	74634
tr12-30	12	30	130177	130596 [†]
tr24-15	24	15	136366	136509
tr24-30	24	30	287753	287929 [†]
mset1ch	20	12	54518	54538
mpp08a	8	8	7997	8430
xtr6-15	6	15	4035	5170
xtr6-30	6	30	5845	7880*
xtr12-15	12	15	6992	7678*
xtr12-30	12	30	13129	14785*
xtr24-15	24	15	13617	13978*
xtr24-30	24	30	17322	18772*
xset1ch	20	12	951	1006
xpp08a	8	8	461	505

indicates that the optimum is not known, and the value listed is that of the best known solution. A [†] indicates an instance for which the value of the optimal solution was not known before this research; that is, our contributions enable us to solve these previously intractable problems. For these problems *bc – prod* is not able to solve the problem on our computer in any length of time.

6.2 Algorithm Description

We have compared *bc – prod* with both cut-and-branch (C&B) and branch-and-cut (B&C) implementations of the inequalities described in this paper. These inequalities include the (l, S) inequalities and the two classes we have defined: the cover inequalities (20) and reverse cover inequalities (22). For C&B we call Heuristic 3 until we do not find any more violated inequalities; in B&C, we follow the same procedure at the root node.

In B&C, at every node of the branch-and-bound tree, we load all violated cuts in the cut pool into the matrix and resolve the LP. If there are no more violated cuts in the cut pool, we call Heuristic 3, with the modification that the loop from steps 2 through 7 is repeated *once*, rather than P times (as at the root node). We repeat this procedure and reoptimize until we do not find a violated inequality, either in the cut pool or with

our own separation routines.

Belvaux and Wolsey reported that having *bc-prod* generate these cuts in the tree did not appear to be very helpful for solving instances of lot-sizing problems; therefore, in running *bc-prod*, we have it generate cuts only at the root node of the tree. At the root node, *bc-prod* generates valid inequalities specifically designed for lot-sizing problems, and then it generates general MIP cuts (that is, inequalities from *bc-opt*—note that these include the path inequalities).

Before proceeding, we note again that while *bc-prod* uses many families of valid inequalities, we use only three: the (l, S) inequalities, and the two families defined for MCL. Moreover, *bc-prod* uses the (l, S) inequalities, since it uses path inequalities. Thus, the only inequalities we employ that are not used by *bc-prod* are the newly defined cover and reverse cover inequalities, while *bc-prod* uses several families of inequalities that we do not have access to.

6.3 Algorithm Comparison

Our computational results are displayed in Table 2. All the computations were carried out on a 350 megahertz PC with 128 megabytes of RAM, running under Windows NT 4.0.

For all algorithms, X_{LP} is the value of the LP relaxation after cut generation is complete, LB is the global lower bound at the termination of the algorithm, and UB is the value of the best integer feasible solution found. We also list the number of nodes processed and the number of cuts added (note that for B&C this is the number of cuts added throughout the tree, including the root node). We give the branch-and-bound time of each algorithm in CPU seconds; if any of the algorithms did not find the optimal solution after 3600 seconds, we terminated the algorithm. We indicate this with *. Finally, “Gap” refers to the ratio $\frac{UB-LB}{UB-LB^*}$.

Clearly C&B solves the test problems more effectively than *bc-prod*. Four instances (*tr6-15*, *mpp08a*, *mset1ch*, and *xpp08a*) solve easily with both C&B and *bc-prod*. For two instances (*xtr24-15* and *xtr24-30*), the lower bound on termination provided by C&B is better, but *bc-prod* finds a better feasible solution. For one instance (*xset1ch*, at termination both of the bounds found by *bc-prod* are at least as good as those found by our C&B, and the lower bound is strictly better. Thus *bc-prod* solves this instance better than C&B. For all the other instances, however, both bounds of C&B are at least as good as *bc-prod*, and at least one of the two is strictly better. In seven of these nine cases, *both* bounds found by C&B are strictly better than those found by *bc-prod*.

We can make other observations concerning the effects of using the inequalities defined in the earlier sections. With the exception of *xset1ch*, for each of the instances in the test set, the addition of the lot-sizing cuts in our code provides a better lower bound than the generation of *all* the families of cuts (both lot-sizing specific and those from

Table 2: Computational Results for MCL

Problem	code	X_{LP}	LB	UB	nodes	cuts	sec	Gap
tr6-15	$bc - prod$	37213	37721	37721	5851	242	21	0
	C&B	37319	37721	37721	1820	264	20	0
	B&C		37721	37721	476	1362	37	0
tr6-30	$bc - prod$	60946	61520	61746	392400	680	3600*	.28
	C&B	61150	61746	61746	201804	439	2617	0
	B&C		61746	61746	19048	6760	1824	0
tr12-15	$bc - prod$	73851	74627	74634	349100	499	3600*	.01
	C&B	73929	74634	74634	172963	458	3150	0
	B&C		74356	74634	11325	16488	3600*	.35
tr12-30	$bc - prod$	130177	130450	130639	193700	1411	3600*	.41
	C&B	130292	130485	130612	43700	974	3600*	.29
	B&C		130453	130617	2550	11728	3600*	.37
tr24-15	$bc - prod$	136366	136474	136537	255700	1071	3600*	.37
	C&B	136388	136486	136509	103925	809	3600*	.16
	B&C		136471	136537	10500	13097	3600*	.39
tr24-30	$bc - prod$	287753	287814	287998	76500	2906	3600*	.75
	C&B	287811	287870	287944	9850	1978	3600*	.39
	B&C		287872	287942	2250	6888	3600*	.37
mset1ch	$bc - prod$	54518	54538	54538	97	449	1	0
	C&B	54536	54538	54538	3	389	1	0
	B&C		54538	54538	3	389	1	0
mpp08a	$bc - prod$	8024	8430	8430	2668	100	7	0
	C&B	8078	8430	8430	1242	151	6	0
	B&C		8430	8430	407	661	11	0
xtr6-15	$bc - prod$	4053	5007	5210	491400	254	3600*	.17
	C&B	4409	5170	5170	232697	271	1622	0
	B&C		5170	5170	24185	5033	1121	0
xtr6-30	$bc - prod$	5882	6558	8067	400000	471	3600*	.68
	C&B	6599	7067	7880	185475	467	3600*	.39
	B&C		7116	7925	27625	6168	3600*	.39
xtr12-15	$bc - prod$	6997	7130	8036	339600	408	3600*	.87
	C&B	7077	7210	7889	214000	367	3600*	.76
	B&C		7199	7886	11775	23822	3600*	.77
xtr12-30	$bc - prod$	13159	13317	14950	208100	941	3600*	.90
	C&B	13337	13494	14835	79400	630	3600*	.79
	B&C		13471	14799	4375	20925	3600*	.80
xtr24-15	$bc - prod$	13630	13704	14094	167700	979	3600*	.82
	C&B	13663	13736	14120	41525	769	3600*	.76
	B&C		13722	14228	2925	14874	3600*	.83
xtr24-30	$bc - prod$	17337	17394	19007	100300	1870	3600*	.96
	C&B	17398	17453	19289	41800	1246	3600*	.93
	B&C		17449	19856	3175	10901	3600*	.95
xset1ch	$bc - prod$	972	1004	1006	425500	475	3600*	.02
	C&B	956	998	1006	356500	238	3600*	.15
	B&C		994	1007	131975	1172	3600*	.23
xpp08a	$bc - prod$	461	505	505	29938	113	67	0
	C&B	470	505	505	13250	130	38	0
	B&C		505	505	12808	737	130	0

$bc - opt$) used by $bc - prod$. Indeed, in the three $xtrP-J$ instances that have 30 periods, the lower bound provided by our cuts at the root node is higher than that provided by $bc - prod$ after adding cuts *and* performing branch-and-bound for an hour. The average value of “Gap” at termination for C&B, for the entire test set, is .29; for $bc - prod$ the average value is .39. (For B&C this value is .34.)

For some instances, the improvement of C&B over $bc - prod$ is especially noteworthy. For example, note that $tr6-30$, a problem that $bc - prod$ cannot solve to optimality on our computer in any length of time, solves in less than 45 minutes with C&B (and in only half an hour with B&C). Similarly C&B (as well as B&C) solves $xtr6-15$ to optimality long before $bc - prod$ can even identify the optimal solution, let alone prove its optimality.

Also, note that our cuts almost completely close the gap left between LB' and OPT for $mset1ch$, and they enable us to solve this instance in just 3 branch-and-bound nodes. Through additional experimentation we know that adding one round of Gomory cuts (see Gomory [1960]) after adding our cuts completely closes the duality gap and solves this instance at the root node, without branching. (In general using Gomory cuts does not enhance computational performance for lot-sizing problems; therefore neither $bc - prod$ nor our algorithms use them by default.)

The computational value of our contributions is clear from the performance of C&B. The algorithm B&C does not compare as well, although its performance is still generally better than $bc - prod$. The performance of B&C appears to be significantly retarded by the time necessary to resolve LP's at each node after adding cuts. (This will be discussed in more detail later). However, B&C solves many problems better than $bc - prod$ and solves several better than C&B (for example, $tr6-30$ and $xtr6-15$). Moreover, studying characteristics of its performance may help us to understand aspects of the polyhedral structure of MCL.

In Table 3 we list some additional statistics concerning the cuts generated by our code during the solution process of each of the test problems. These include the number of (l, S) cuts, cover cuts, and reverse cover cuts generated by our code at the root node—marked by (r)—and in the branch-and-bound tree—marked by (t). Also included is the percentage of time spent separating for cuts at the root node, the percentage of time spent separating for cuts, and the percentage of time spent in the whole branch-and-bound tree retrieving violated cuts from the cut pool and restoring them to the matrix. (The total time taken by our code—and by $bc - prod$ —at the root node never exceeds a few seconds for any instance, so we do not report it.)

Recall from Table 2 that B&C takes a long time to process each node compared to the other two. The data in Table 3 indicate that the extra time required at each node in B&C is certainly *not* required for separation—which appears to be efficient—and not primarily for retrieving old cuts. Rather, the time required is primarily for resolving LP's. This suggests that it might be possible to improve the performance of B&C by more sophisticated cut management strategies—for example, perhaps by adding cuts

Table 3: Cut Statistics for MCL

Problem	# $(l, S)(r)$	# cov(r)	# rc(r)	% sep(r)	# $(l, S)(t)$	# cov(t)	# rc(t)	% sep(t)	% ret(t)
tr6-15	196	39	29	76 %	311	387	400	2 %	12 %
tr6-30	329	66	44	75 %	3227	1805	1289	1 %	27 %
tr12-15	379	39	40	67 %	789	7408	7833	1 %	28 %
tr12-30	828	91	55	55 %	5325	3013	2416	1 %	13 %
tr24-15	697	67	45	55 %	920	5358	6010	2 %	27 %
tr24-30	1721	139	118	41 %	2137	1603	1170	1 %	9 %
mset1ch	371	8	10	75 %	0	0	0	0 %	0 %
mpp08a	114	16	21	68 %	100	129	281	3 %	15 %
xtr6-15	149	60	62	69 %	919	1726	2117	1 %	25 %
xtr6-30	242	132	93	62 %	1530	2074	2097	1 %	37 %
xtr12-15	247	70	50	65 %	1366	10701	11388	1 %	37 %
xtr12-30	500	156	74	55 %	5652	6981	7662	1 %	24 %
xtr24-15	543	124	102	53 %	2026	5504	6575	1 %	17 %
xtr24-30	999	151	96	50 %	5688	2734	3082	1 %	11 %
xset1ch	205	17	16	78 %	188	5948	9158	3 %	61 %
xpp08a	77	22	31	76 %	179	496	1075	3 %	24 %

only at specifically chosen nodes of the branch-and-bound tree. (Indeed, since C&B takes longer to process nodes than *bc – prod*, it might be possible to improve C&B through better cut management as well.)

Nevertheless, the data of this table, along with the results of Table 2, suggest that more classes of inequalities may be needed to solve MCL as efficiently as possible. For example, for some instances, many more cover and reverse cover inequalities are found in the tree than at the root node. This is probably due at least in part to the fact that branching explores many more regions of the polytope than are considered at the root node. However, this occurrence might also indicate that while some these inequalities support the convex hull closely, and while all of them support some integer points in the convex hull closely, some of the inequalities do not closely support many integer points in the convex hull and are thus only violated after integrality is forcibly imposed on several variables through branching. The large duality gaps left for the *xtrP-J* instances, as well as the relatively poor performance of our code on *xset1ch*, further suggest that, for some instances at least, this is indeed the case. Moreover, the idea that more classes of inequalities are needed is also suggested by the fact that our separation routines appear to be “good”, in the sense that, no matter how many different assignment of $\alpha_i, i = 1, \dots, P$ we try in each period, we do not achieve measurably better results (in terms of duality gap at the root node) than we do with the given settings.

6.4 Summary

The inequalities we have identified clearly help us to solve instances of MCL better than has been possible before. We can solve many problems easily that were once difficult, and we can obtain better bounds more quickly on problems that are still difficult. Indeed, we can now solve fairly easily some instances that before were intractable or almost intractable. Thus our contributions for MCL can provide a good complement to *bc – prod*’s capability in solving small bucket problems.

Although our results indicate that the cover and reverse cover inequalities for MCL are effective for solving MCL, it is almost certainly possible to improve on the results given here. For example, for both of our algorithms, a more sophisticated cut management strategy might enable nodes to be processed faster and thus improve performance. It also seems likely that we could improve upon the results already obtained by using the cover and reverse cover inequalities in conjunction with all of the classes used by *bc – prod*, both at the root node and in the branch-and-bound tree.

Finally, despite the computational evidence for the usefulness of our inequalities, the large gaps that remain for the *xtrP-J* problems, as well as the poor performance of our codes on *xset1ch*, suggest that there remains room for the identification of additional inequalities that capture aspects of the structure of MCL that have not yet been exploited.

7 Conclusions

We have derived new valid inequalities for MCL, a standard yet challenging lot-sizing model, by extending results for PI, a single-period relaxation of the original model. The inequalities we thus obtain for MCL consider demand for multiple items over several periods simultaneously, and they incorporate capacity restrictions in their definition as well. We have also discussed how effectively to implement these inequalities within a branch-and-cut algorithm. Our developments enable us to obtain significant improvement over *bc – prod*, a branch-and-cut code customized specifically for lot-sizing problems. Since these improvements were obtained even though we were not able to implement many of the features in *bc – prod*, we suspect that our developments could be much more computationally effective than they already are if implemented within such a system.

We have discussed areas of further computational interest, based on an analysis of our computational results, in Section 6. In addition, the ideas that we have implemented can be extended to some other multi-item lot-sizing models. In Miller [1999], a number of extensions to MCL are considered, including models in which variables lower bounds, setup carryovers, backorders and/or overtime production are added to the formulation of MCL. For all these extended models, many, and in some cases all, of the results obtained for PI carry over to the single period relaxations of these models. In addition,

PI itself provides a natural relaxation for many multi-level lot-sizing problems (Miller, et al. [2000c]). It therefore seems reasonable to expect that extending and applying our results to all of these models will be useful computationally.

Finally, both for MCL and for the extensions just mentioned, there remains the potential for further study of the polyhedral structure and identification of valid inequalities.

References

- I. Barany, T. Van Roy, and L.A. Wolsey. Uncapacitated lot-sizing: the convex hull of solutions. *Mathematical Programming Study*, 22:32–43, 1984.
- G. Belvaux and L.A. Wolsey. Lot-sizing problem: modelling issues and a specialized branch-and-cut system *bc – prod*. Technical Report CORE DP9849, Université Catholique de Louvain, Louvain-la-Neuve, September 1998.
- G. Belvaux and L.A. Wolsey. Modelling practical lot-sizing problems as mixed integer programs. Technical Report CORE DP2000/9, Université Catholique de Louvain, Louvain-la-Neuve, February 2000.
- R.E. Bixby, E.A. Boyd, and R.R. Indovina. Miplib: a test set of mixed integer programming problems. *SIAM News*, 16, March 1992.
- M. Constantino. A cutting plane approach to capacitated lot-sizing with start-up costs. *Mathematical Programming*, 75:353–376, 1996.
- M. Constantino. Lower bounds in lot-sizing models: a polyhedral study. *Mathematics of Operations Research*, 23:101–118, 1998.
- C. Cordier, H. Marchand, R. Laundy, and L.A. Wolsey. *bc – opt*: a branch-and-cut code for mixed integer programs. *Mathematical Programming, Series A*, 86:335–353, 1999.
- H. Crowder, E.L. Johnson, and M.W. Padberg. Solving large scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983.
- M. Diaby, H.C. Bahl, M.H. Karwan, and S. Zionts. A lagrangean relaxation approach for very large scale capacitated lot-sizing. *Management Science*, 38:1329–1339, 1992.
- M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Co., San Francisco, 1979.
- R.E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The RAND Corporation, 1960b.

- Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Cover inequalities for 0–1 linear programs: computation. *INFORMS Journal on Computing*, 10:427–437, 1998.
- Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh. Lifted flow cover inequalities for mixed 0–1 integer programs. *Mathematical Programming*, 85:439–467, 1999.
- E. Katok, H.S. Lewis, and T.P. Harrison. Lot–sizing in general assembly systems with setup costs, setup times, and multiple constrained resources. *Management Science*, 44:859–877, 1998.
- J. Leung, T.L. Magnanti, and R. Vachani. Facets and algorithms for capacitated lot–sizing. *Mathematical Programming*, 45:331–359, 1989.
- H. Marchand and L.A. Wolsey. Aggregation and mixed integer rounding to solve MIP’s. Technical Report CORE DP9839, Université Catholique de Louvain, Louvain-la-Neuve, June 1998.
- A.J. Miller. *Polyhedral Approaches to Capacitated Lot–Sizing Problems*. PhD thesis, Georgia Institute of Technology, 1999.
- A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. A multi–commodity flow model with setup times: Algorithms, reformulations, and polyhedral characterizations for a special case. Technical Report CORE DP, Université Catholique de Louvain, Louvain-la-Neuve, 2000a.
- A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. On the capacitated lot–sizing and continuous 0–1 knapsack polyhedra. *European Journal of Operational Research*, 125:298–315, 2000b.
- A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. On the polyhedral structure of a multi–commodity flow model with setup times. Technical Report CORE DP, Université Catholique de Louvain, Louvain-la-Neuve, 2000c.
- M.W. Padberg, T.J. Van Roy, and L.A. Wolsey. Valid linear inequalities for fixed charge problems. *Operations Research*, 33:842–861, 1985.
- Y. Pochet. Valid inequalities and separation for capacitated economic lot–sizing. *Operations Research Letters*, 7:109–116, 1988.
- Y. Pochet and L.A. Wolsey. Solving multi–item lot–sizing problems using strong cutting planes. *Management Science*, 37:53–67, 1991.
- Y. Pochet and L.A. Wolsey. Lot–sizing with constant batches: formulation and valid inequalities. *Mathematics of Operations Research*, 18:767–785, 1993.

- Y. Pochet and L.A. Wolsey. Polyhedra for lot-sizing with wagner-whitin costs. *Mathematical Programming*, 67:297–323, 1994.
- T.J. Van Roy and L.A. Wolsey. Solving mixed 0–1 problems by automatic reformulation. *Operations Research*, 35:45–57, 1987.
- H. Tempelmeier and M. Derstoff. A lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science*, 42:738–757, 1996.
- W.W. Triggiero, L.J. Thomas, and J.O. McClain. Capacitated lot-sizing with setup times. *Management Science*, 35:353–366, 1989.
- R. Weismantel. On the 0–1 knapsack polytope. *Mathematical Programming*, 77:49–68, 1997.