Solving Multi-Item Capacitated Lot-Sizing Problems with Setup Times by Branch-and-Cut [1]

Andrew J. Miller
CORE
Voie du Roman Pays, 34
B-1348 Louvain–la–Neuve
Belgium

George L. Nemhauser
Martin W.P. Savelsbergh
Georgia Institute of Technology
School of Industrial and Systems Engineering
Atlanta, GA 30332-0205
USA

Abstract

We study the multi–item capacitated lot–sizing problem with setup times (MCL). This model appears often in practice, either in its pure form or with additional constraints, but it is very 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. By analyzing the polyhedral structure of simplified models obtained from a single time period of MCL, we obtain strong valid inequalities for MCL. These inequalities are the first reported that consider demand for multiple items and the joint capacity restriction simultaneously. We report successful computational results with these inequalities in a branch–and–cut algorithm; these results suggest that our contributions significantly advance the state of the art in solving MCL.

KEYWORDS: Mixed integer programming, capacitated lot-sizing, setup times

October 1999

The current version of this document is a draft and is not intended for distribution.

[1]This research was 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

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

subject to

$$x_j^i + s_{j-1}^i - s_j^i = d_j^i, i = 1, ..., P, j = 1, ..., J \tag{2}$$

$$\sum_{i=1}^{P} x_j^i + \sum_{i=1}^{P} t_j^i y_j^i \le c_j, j = 1, ..., J \tag{3}$$

$$x_j^i \le (c_j - t_j^i) y_j^i, i = 1, ..., P, j = 1, ..., J \tag{4}$$

$$x_j^i, s_j^i \ge 0, i = 1, ..., P, j = 1, ..., J \tag{5}$$

$$y_j^i \in \{0, 1\}, i = 1, ..., P, j = 1, ..., J. \tag{6}$$

The number of time periods in the problem is T. The number of items is P; we let $\mathcal{P} = \{1, ..., P\}$. 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 $0 \le t_j^i < c_j$, $j = 1, ..., J$, $i = 1, ..., P$. The parameter p_j^i is the unit cost of production of item i during period j, h_j^i is the 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 $s_0 \ge 0$ 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 $d_{jl}^i = \sum_{k=j}^{l} d_k^i, j \le l$, we can tighten (4) to

$$x_j^i \le \min\{c_j - t_j^i, d_{jT}^i\} y_j, i = 1, ..., P, j = 1, ..., J. \tag{7}$$

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, although 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 Trigiero, Thomas, and McClain [1989], who used a heuristic that employs Lagrangean relaxation and production smoothing to obtain near optimal solutions to MCL. Since the Langangean solutions they obtained were not always feasible, they used a production smooothing 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 Trigiero, 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, Lewis, and Harrison [1998].

Pochet and Wolsey [1990] and Belvaux and Wolsey [1998] 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 Trigiero, 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 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 both these families and previously known classes 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 this relaxation. In the next section, we will summarize this research.

In Section 3, we present results for a new 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, Nemhauser, and Savelsbergh [1999a].) We present valid inequalities for PI, and give general conditions under which these inequalities define facets of the convex hull.

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, 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. Our 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 $x^i_j \leq d^i_{jT} y^i_j, j = 1, ..., J, i = 1, ..., P$. 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]). Computational experience using these 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, Magnanti, and Vachani [1989], Pochet and Wolsey [1993, 1994], Miller, Nemhauser, and Savelsbergh [1999b], and Miller [1999]. 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, the (l, S) inequalities have often been the most effective known class.

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 certainly 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 our goal is to derive strong valid inequalities for MCL by considering a simplified model obtained from a single time period of MCL. The submodel we analyze considers both inventory and demand. Through analysis of this single period submodel, we obtain the first valid inequalities for MCL that consider demand for multiple items simultaneously. Even if we relax MCL and set $t^i_j = 0$ for all i and j, these inequalities are still the first for this model—MCL *without* setup times—to consider demand for multiple items simultaneously.

We will consider a single period j with all s^i_{j-1} present (that is, *preceding inventory*). We call this model PI. We note here that it is also possible to consider a single period j with all s^i_j present (that is, with *succeeding inventory*). The analyses of these two models yield exactly the same results for MCL, so we will not present the analysis of the latter in this paper. Also, for the sake of brevity many of the proofs in the remainder of this paper are omitted. The interested reader can find many of these proofs in Miller, Nemhauser, and Savelsbergh [1999a]. Even more detail is given in Miller [1999].

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

$$\min \quad \sum_{i=1}^{P} p^i x^i + \sum_{i=1}^{P} q^i y^i + \sum_{i=1}^{P} h^i s^i \tag{8}$$

subject to

$$x^i + s^i \geq d^i, i = 1, ..., P, \tag{9}$$

$$\sum_{i=1}^{P} x^i + \sum_{i=1}^{P} t^i y^i \leq c, \tag{10}$$

$$x^i \leq (c - t^i) y^i, i = 1, ..., P, \tag{11}$$

$$x^i, s^i \geq 0, i = 1, ..., P, \tag{12}$$

$$y^i \in \{0, 1\}, i = 1, ..., P, \tag{13}$$

We refer to the set of points defined by (9)–(13) as X^{PI}.

If $h^i < 0$ for any $i \in \mathcal{P}$, PI has unbounded optimum. We will therefore assume that $h^i \geq 0, i = 1, ..., P$. Also, we assume that $q^i > 0, i = 1, ..., P$. Not only is this assumption often satisfied in every period of instances of MCL that occur in practice, but it ensures that in any optimal solution, an item is not set up for unless it is produced.

Under general conditions, inequalities from the LP formulation induce facets of $\text{conv}(X^{LP})$.

Proposition 1 *Assuming $c > t^i, i = 1, ..., P$, conv(X^{PI}) is full–dimensional. Moreover, for $i = 1, ..., P$, (9), (11), and the bound $x^i \geq 0$ induce facets of conv(X^{PI}).*

Proposition 2 *If $c > t^i + t^j, i \neq j$, then (10) and the bounds $y^i \leq 1, i = 1, ..., P$ induce facets of conv(X^{PI}).*

Another simple family of inequalities induces facets of $\text{conv}(X^{PI})$.

Proposition 3 *If $c > t^i + d^i, i = 1, ..., P$ the valid inequality*

$$s^i + d^i y^i \geq d^i \tag{14}$$

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

Note that (14) implies the bound $s^i \geq 0, i = 1, ..., P$; therefore this bound never induces a facet. Also note that these inequalities correspond to the (l, S) inequalities for ULS from Barany, Van Roy, and Wolsey [1984].

Now we define two new families of valid inequalities for X^{PI}. In order to define 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, then clearly $\sum_{i \in S} s^i \geq \lambda$ is true. Moreover, if we fix $y^{i'} = 0$ for some $i' \in S$,

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

must hold. Thus

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

is valid for PI. This observation is the basis for

Proposition 4 (Cover Inequalities) *Given a cover S of PI, order the $i \in S$ such that $t^{[1]} + d^{[1]} \geq ... \geq t^{[|S|]} + d^{[|S|]}$. 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. Finally, let*

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

If $t^{[1]} + d^{[1]} \geq \lambda$, then

$$\begin{gathered} \textstyle\sum_{i \in S \cup U} s^i \geq \lambda + \sum_{i \in S} \max\{-t^i, d^i - \lambda\}(1 - y^i) + \\ \textstyle\sum_{i \in U} \min\{t^i - \mu^1, \lambda - d^i\} y^i + \sum_{i \in U} d^i + \\ \frac{\sum_{i \in S} \min\{t^i + d^i, \lambda\} + \sum_{i \in U} \min\{t^i + d^i - \mu, \lambda\} - \lambda}{(|S| + |U| - 1) D'} \textstyle\sum_{i \in T'} (x^i - (\mu^1 - t^i)^+ y^i) \end{gathered} \tag{15}$$

is valid for X^{PI}.

When $T' = \emptyset$, the proof that (15) is valid follows from an argument similar to that used to motivate this proposition. If $T' \neq \emptyset$, the proof of validity becomes long, technical, and not very intuitive, and it is therefore omitted.

Example 1

Consider an instance of PI with $P = 3$, with $C = 10$, and with demand and setup times given by

i	1	2	3
t^i	2	2	1
d^i	5	4	6

Take $S = \{1, 2\}$ as our cover; then $\lambda = 3$, $t^{[1]} + d^{[1]} = 7$, and $\mu^1 = 4$. Take $U = 3$, and (15) yields

$$s^1 + s^2 + s^3 \geq 3 + 2(1 - y^1) + (1 - y^2) - 3y^3 + 6 = 12 - 2y^1 - y^2 - 3y^3, \tag{16}$$

a valid inequality for X^{PI}. $\square$

We can establish general conditions under which this class of inequalities induces facets of conv(X_{PI}).

Proposition 5 *If $\lambda > 0$, $t^{[1]} + d^{[1]} \geq t^{[2]} + d^{[2]} > \lambda$, $t^i \leq \mu^1, i \in U$, $t^{[1]} + d^{[1]} = t^i + d^i < \mu^1 + t^{[2]} + d^{[2]}, i \in U$, $t^i < \mu^1, i \in T$, and $T' = \emptyset$, then (15) induces a facet of conv(X^{PI}).*

Example 1 (continued)

The inequality (16) satisfies the conditions of Proposition 5, and is therefore a facet of conv(X^{PI}). $\square$

There are facets induced by inequalities of the form (15) with $T' \neq \emptyset$. We can extend Proposition 5 to establish conditions under which these inequalities induce facets.

Proposition 6 *If $\lambda > 0$, $t^{[1]} + d^{[1]} \geq t^{[2]} + d^{[2]} > \lambda$, $t^i \leq \mu^1, i \in U$, $t^{[1]} + d^{[1]} = t^i + d^i < \mu^1 + t^{[2]} + d^{[2]}, i \in U$, $t^i < \mu^1, i \in T$, and $t^i + d^i = t^{[2]} + d^{[2]}, \forall i \in S \setminus [1]$, then (15) induces a facet of conv(X^{PI}).*

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

1. $y^{i'} = 0$; or
2. $y^i = 0, i \in S$ and $\sum_{i \in S} s^i \geq \sum_{i \in S} d^i$.

Also, if $\sum_{i \in \hat{S}} s^i = 0$, where $\hat{S}$ is any subset of S, then either

1. $y^{i'} = 0$; or
2. $x^{i'} \leq c - t^{i'} - \sum_{i \in \hat{S}}(t^i + d^i) = (c - t^{i'})y^{i'} - \sum_{i \in \hat{S}}(t^i + d^i)y^{i'}$.

This reasoning yields

Proposition 7 (Reverse Cover Inequalities) *Let S be a reverse cover of PI, let $T = \mathcal{P} \backslash S$, and let (T', T'') be any partition of T. Then*

$$\sum_{i \in S} s^i \geq (\sum_{i \in S}(t^i + d^i)) \sum_{i \in T'} y^i - \sum_{i \in S} t^i(1 - y^i) - \sum_{i \in T'}((c - t^i)y^i - x^i) \tag{17}$$

is valid for X^{PI}.

Example 1 (continued)

Choose $S = 1$ as a reverse cover; thus $\mu = 1$. Let $T' = 3$; then (17) yields

$$s^1 \geq 7y^3 - 2(1 - y^1) - (9y^3 - x^3) = -2 + 2y^1 - 2y^3 + x^3, \tag{18}$$

a valid inequality for X^{PI}. $\square$

Under very general conditions, reverse cover inequalities induce facets of conv(X^{PI}).

Proposition 8 *If $S \neq \emptyset$, $T' \neq \emptyset$, and $t^i < \mu, i \in T$, (17) induces a facet of conv(X^{PI}).*

Example 1 (continued)

The inequality (18) satisfies the conditions of Proposition 8 and is therefore a facet of conv(X^{PI}). $\square$

While in general PI is $\mathcal{NP}$–complete (Miller [1999]), in certain special cases it can be solved in polynomial time. In these cases, the question of finding conv(X^{PI}), or of finding a set of complete set inequalities that allows PI to be solved by linear programming given the cost conditions $q^i > 0, h^i \geq 0, i = 1, ..., P$, is of theoretical interest. In particular, we have been able to show

Proposition 9 *When $t^i = t \geq 0, i = 1, ..., P$, and $d^i = d \geq 0, i = 1, ..., P$, then PI can be solved by optimizing (8) over (9)–(12), (14), (15), (17), and the bounds $y^i \leq 1, i = 1, ..., P$.*

In this sense, then, the set of inequalities that we have defined for PI is as complete a set we can reasonably hope for. The development needed to prove the above result, as well as many other results concerning the polyhedral structure of PI, can be found in Miller, Nemhauser, and Savelsbergh [1999a]. Even more detail is given in Miller [1999].

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.

A 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 (15) and (17). 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 $j' < T$, we seek to fix some variables $y^i_j, j = j'+1, ..., \sigma(i), i = 1, ..., P$ to 0, where, for every $i \in S$, $j'+1 \leq \sigma(i) \leq J$. This has the effect of forcing the demand for item i in the periods $j'+1, ..., \sigma(i)$ to be satisfied in or prior to period j'. (Note that fixing x^i_j to 0 has the effect of forcing demand for i in j to be satisfied in an earlier time period, just as fixing y^i_j to 0 does.)

Given that either $x^i_j = 0$ or $y^i_j = 0$, $j = j'+1, ..., \sigma(j')$, we can substitute $d^i_{j,\sigma(i)}$ for d^i_j in deriving inequalities of the forms (15) and (17). In order to prove 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 $\{1, ..., P\}$ to $\{j, ..., J\}$. 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

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

Proposition 10 (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 $t^{[1]}_j + d^{[1]}_{j,\sigma([1])} \geq ... \geq t^{[|S|]}_j + d^{[|S|]}_{j,\sigma([|S|])}$. Define $\mu^1 = (t^{[1]}_j + d^{[1]}_{j,\sigma([1])} - \lambda)^+$. Let (T, U) be any partition of $\mathcal{P} \setminus S$, and let (T', T'') be any partition of T. Finally, let*

$$D' = \max\{t^{[2]}_j + d^{[2]}_{j,\sigma([2])}, \max_{i \in U}\{t^i_j + d^i_{j,\sigma(i)}\}\}$$

If $t_j^{[1]} + d_{j,\sigma([1])}^{[1]} \geq \lambda$, *then*

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

is valid for MCL.

Proof: The proof proceeds by induction on $\sigma(i), i \in S \cup U$. The basis step consists of showing that 19 is valid when $\sigma(i) = j$, $i \in S \cup U$; but this follows from Proposition 4.

Now define a partition (S, T, U) of $\mathcal{P}$ as in the statement of Proposition 10. For each i, choose $j(i)$ so that $j \leq j(i) \leq J$. We need to show that (19) is valid when $\sigma(i) = j(i)$. For the inductive step, we assume that (19) is valid whenever

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

Then consider a solution $(\bar{x}, \bar{y}, \bar{s})$, of MCL. If $\bar{y}_k^i = 0$, $k = j+1, ..., \sigma(i), i \in S \cup U$, then validity holds for this point because of Proposition 4. If $\bar{y}_k^i = 1$ for some k and i such that $j < k \leq j(i)$, then, for each i, let $k(i) = \min_{k=j+1,...,j(i)}\{k : \bar{y}_k^i = 1\}$, or let $k(i) = j(i)+1$ if $\{k : j+1 \leq k \leq j(i), \bar{y}_k^i = 1\} = \emptyset$. Then (19) is implied by another inequality of the form (19), with $\sigma(i) = k(i) - 1, i \in S \cup T'$, which is valid because of our inductive assumption. $\square$

In (19), for each $i \in S \cup T'$, $y_k^i, k = j+1, ..., \sigma(j)$, are first fixed to 0 and then "lifted" back into the inequality. We can extend (19) to

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

This allows us to take the minimum of $d_{k,\sigma(i)}^i \bar{y}_k^i$ and $\bar{x}_k^i$ in separating a given point $(\bar{x}, \bar{y}, \bar{s})$. Examples of valid inequalities for MCL from this family are included in our discussion of computational issues in the next section.

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

$$\mu = c_j - \sum_{i \in S} (t^i_j + d^i_{j,\sigma(i)}) > 0.$$

Proposition 11 (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*

$$\begin{aligned} \textstyle\sum_{i \in S} s^i_{j-1} \geq (\sum_{i \in S} (t^i_j + d^i_{j,\sigma(i)})) \sum_{i \in T'} y^i_j - \sum_{i \in S} t^i_j (1 - y^i_j) - \sum_{i \in T'} ((c_j - t^i_j) y^i_j - x^i_j) \\ - \textstyle\sum_{i \in S} \sum_{k=j+1}^{\sigma(i)} d^i_{k,\sigma(i)} y^i_k \end{aligned} \tag{21}$$

is valid for MCL.

The proof of this proposition relies on the same observations as that of Proposition 10. As before, we can extend (21) to

$$\begin{aligned} \textstyle\sum_{i \in S} s^i_{j-1} \geq (\sum_{i \in S} (t^i_j + d^i_{j,\sigma(i)})) \sum_{i \in T'} y^i_j - \sum_{i \in S} t^i_j (1 - y^i_j) - \sum_{i \in T'} ((c_j - t^i_j) y^i_j - x^i_j) \\ - \textstyle\sum_{i \in S} \sum_{k=j+1}^{\sigma(i)} \min\{d^i_{k,\sigma(i)} y^i_k, x^i_k\} \end{aligned} \tag{22}$$

5 Computational Issues and Separation

We now discuss issues that are of crucial importance in the implementation of our results in a branch–and–cut algorithm. These include defining separation heuristics for the inequalities (15) and (17), 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. 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 provides the lower bound for that node. In addition, 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.

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 Propositions 3 and 9. We use only those (l, S) inequalities that have the form

$$s_{k-1}^i + \sum_{j=k}^{l} d_{jl}^i y_j^i \geq d_{kl}^i, \tag{23}$$

where $i \in \mathcal{P}$ and $1 \leq k \leq l \leq J$. This subset corresponds to that which is sufficient to solve the uncapacitated lot–sizing problem by LP in the presence of Wagner–Whitin costs (Pochet and Wolsey [1994]), and it has cardinality $\mathcal{O}(PT^2)$. Thus, separation by enumeration takes $\mathcal{O}(PT^3)$, and it is easy to obtain a lower bound on the optimal solution value of an instance of MCL by adding all of these 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 $\{(i,k) : k = j+1, ..., \sigma(i), i \in S \cup T\}$. 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 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. Thus, solving these problems to optimality is something that should not be attempted. We discuss the methods we employ to try heuristically to find violated inequalities. We will assume first that $\sigma(i), i = 1, ..., P$ has been defined, and discuss the heuristics we use, given this definition of $\sigma(\cdot)$. Then, we will discuss how we define $\sigma(\cdot)$.

5.1 Separation Heuristic for Cover Inequalities

In separating for a violated cover inequality, we first order the items in some order $[1], ..., [P]$ (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 $S = \{[1], ..., [i']\}$. 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

$$d_{j,\sigma([i])}^{[i]} + \min\{\lambda - d_{j,\sigma([i])}^{[i]}, t^i - \mu^1\}\bar{y}^i - \bar{s}_{j-1}^i - \sum_{k=j+1}^{\sigma([i])} d_{k,\sigma(i)} \bar{y}_k^i > 0,$$

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. □

Thus, after defining the cover S and defining λ and μ, 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, not only because it will increase the violation for more fractional points, but also because a large denominator makes it unlikely that many integer feasible points with $y_j^i = 1$ will lie in or close to the hyperplane defined by the inequality with $i \in T'$. Also, including items in T' is more likely to contribute to numerical stability challenges than including them in U.

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

$$\max\{-t, d^i_{j,\sigma(i)} - \lambda\}(1 - \bar{y}^i_j) - \bar{s}^i_{j-1} - \sum_{k=j+1}^{\sigma(i)} d^i_{k,\sigma(i)} \bar{y}^i_k, \tag{24}$$

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

$$(t^i_j + d^i_{j,\sigma(i)})\bar{y}^i_j \tag{25}$$

   Call Heuristic 1; if we do not find and add 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

$$\max\{-t^i_j, d^i_{j,\sigma(i)} - \bar{\lambda}\}(1 - \bar{y}^i_j) - \bar{s}^i_{j-1} - \sum_{k=j+1}^{\sigma(i)} d^i_{k,\sigma(i)} \bar{y}^i_k \tag{26}$$

   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; this ordering is also justified by some of our theoretical results for PI, which suggest that cover inequalities provide the deepest cuts when many of the elements in S have values of $\bar{y}^i$ that are close to or equal to 1 (see Miller, Nemhauser, and Savelsbergh [1999b]).

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

Given an ordering $[1], ..., [P]$ 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 $i = [i'] + 1, ..., [P]$, if $(t^i - \mu)\bar{y}^i + \bar{x}^i > 0$ and $i \notin T'$, then 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 If $i' \in T'$, go to Step 3. Otherwise, put i' into S, and go to Step 5.

Step 5 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. □

Again, the reverse covers formed in this heuristic are greatly affected by the ordering of the elements that the heuristic is given. They are not determined by this ordering, though, since if i is put into T', it is never again a candidate to be placed into S. The reason we leave $i \in T'$ is that the value of μ decreases each time Step 5 is reached, and if $(t^i - \mu)\bar{y}^i + \bar{x}^i > 0$ is true, then of course it is true for smaller values of μ.

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:

$$t^i_j(\bar{y}^i_j - 1) - \bar{s}^i_j - \sum_{k=j+1}^{\sigma(i)} d^i_{k,\sigma(i)}\bar{y}^i_k. \tag{27}$$

This ordering enables the heuristic to find violated inequalities often; however, it penalizes items that have a very large value of $d^i_{j,\sigma(i)}$ and fractional values of $\bar{y}^i_j$ that are not integral but are very close to 1. The reason is that the value of $\bar{s}^i_j$ for such items can be comparatively large. Similarly, it penalizes items that have high setup times t^i_j and large but fractional values of $\bar{y}^i_j$. An item i' that is so penalized for either (or both) of these two reasons are 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

$$(t^i_j + d^i_{j,\sigma(i)})\bar{y}^i_j. \tag{28}$$

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. Our experience suggests that in defining $\sigma(\cdot)$, we should be careful of the following:

1. If $\bar{y}^i_{j+1} = ... = \bar{y}^i_k = 0$, 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 $\sum_{k=j+1}^{\sigma(i)} d^i_{k,\sigma(i)}\bar{y}^i_k$.

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 want to 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

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

where $\alpha \in R, 0 < \alpha \leq 1$. (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

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

Unfortunately, which α to choose is not clear, and depends not only 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. Also, the best choice of α may be different for each i.

We have found that defining $\sigma(i), i = 1, \ldots, P$, by applying the above formula for some α allows us to find many violated inequalities and substantially close duality gaps. Interestingly, we often find violated inequalities for one value of α that we do not find with another. We give a couple of examples to 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 $T = 4$. 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

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

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

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

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

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

It can be checked that this solution satisfies all the (l, S) inequalities of the form (23), 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 take $\alpha \leq \frac{1}{2}$, then we cannot find a violated cover inequality for this example. □

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

Example 3

Consider an instance of MCL with $P \geq 2$ and $T = 4$. 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

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

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

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

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

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

It can be checked that this solution satisfies all the (l, S) inequalities of the form (23), 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

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

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

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

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. □

The problem with defining α a priori 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. In addition to using more than one value of α to find violated inequalities, where α is defined once for all items, we have also 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 $\sigma_{\alpha_i}(i), i = 1, ..., P$ using (30) takes $\mathcal{O}(PT)$ time. We can now define the complete separation heuristic that we call 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 all violated (l, S) inequalities of the form (23). Go to Step 1.

Step 1 Set $j = 1$, set $\alpha_i = 1, i \in \mathcal{P}$, and go to Step 2.

Step 2 (Cover Inequality Separation Loop) Define $\sigma(i) = \sigma_{\alpha_i}(i), i \in \mathcal{P}$. 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. If we find a violated inequality, go to Step 4; if not, go to Step 3.

Step 3 Repeat step 2 for $\alpha_i = .5, i \in \mathcal{P}$, for $\alpha_i = .1, i \in \mathcal{P}$, and then for α_i chosen randomly for each $i \in \mathcal{P}$ from the uniform distribution on $(0, 1]$. As soon as the first violated inequality is found, or if none are found, go to Step 4.

Step 4 Set $\alpha_i = 1, i \in \mathcal{P}$. Go to Step 5.

Step 5 (Reverse Cover Inequality Separation Loop) Define $\sigma(i) = \sigma_{\alpha_i}(i), i \in \mathcal{P}$. Order $i \in \mathcal{P}$ by (25), and call Heuristic 2. If we find a violated inequality, go to Step 4. If we do not, reorder $i \in \mathcal{P}$ by (26) and call Heuristic 2. If we find a violated inequality, go to Step 7; if not, go to Step 6.

Step 6 Repeat Step 5 for $\alpha_i = .5, i \in \mathcal{P}$, for $\alpha_i = .1, i \in \mathcal{P}$, and then for α_i chosen randomly for each $i \in \mathcal{P}$ from the uniform distribution on $(0, 1]$. As soon as the first violated inequality is found, or if none are found, go to Step 7.

Step 7 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. Step 0 takes $\mathcal{O}(PT^3)$. For each j, the most expensive of Steps 2 through 6 are Steps 5 and 6, which take $\mathcal{O}(PT + P^2)$ (that is, the maximum of defining $\sigma_{\alpha_i}(i), i = 1, ..., P$ or of calling Heuristic 2). Thus Steps 1 through 7 execute in $\mathcal{O}(T(PT + P^2))$ time, and the entire separation routine executes in $\mathcal{O}(PT^3 + T(PT + P^2)) = \mathcal{O}(PT(T^2 + P))$ time.

6 Computational Results

Here we present the results of our implementation of the concepts developed above. Our results suggest that our contributions significantly advance the state of the art in solving multi–item capacitated lot–sizing models.

The code in which our results are implemented uses the callable library of XPRESS–MP, Release 11.05. 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.

In addition, XPRESS–MP attempts to find and add several families of violated inequalities at the root node. The valid inequalities that XPRESS–MP generates include MIR inequalities (see Marchand and Wolsey [1998]), lifted knapsack cover inequalities (see Weismantel [1997] and Gu, Nemhauser, and Savelsbergh [1998]), and Gomory cuts (see Gomory [1960a] and [1960b]). Although XPRESS–MP allows the user to generate cuts that he has defined throughout the branch–and–bound tree, it only generates the above classes of cuts at the root node.

All the classes of inequalities that XPRESS–MP uses were first implemented in the experimental branch–and–cut software $bc - opt$ (Cordier et al. [1997]). In addition, $bc - opt$ uses classes of inequalities that are not used by XPRESS–MP (for example, path inequalities (Van Roy and Wolsey [1987]), which are a generalization of the (l, S) inequalities).

The best mathematical programming software currently available for solving lot–sizing problems is $bc - prod$ (Belvaux and Wolsey [1998]). Built on top of $bc - opt$, $bc - prod$ 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, and it also has preprocessing features designed specifically for lot–sizing problems. 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.

The best way to test the computational value of our contributions would be to test them within $bc - prod$. In that way we could discern what value using the inequalities that we have defined adds to what is already known. Unfortunately, we were not able to perform such tests. Instead, therefore, we have tested our implementation directly against $bc - prod$ 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 very favorably with those of $bc - prod$.

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–T* models; each has P items and T time periods. These instances first appeared in Trigiero, Thomas, and McClain [1989]. All of these models 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 all of the problems except *tr6–15* are hard to solve to optimality.

We have created new instances of the three smallest models. These new instances (*ytr6–15*, *ytr6–30*, and *ytr12–15*) were created from their *trP–T* 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.

Also, we modified slightly the LOTSIZELIB 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.

In evaluating the effectiveness of a branch–and–cut algorithm for our test set, the ratio $(UB - LB)/LB$, where UB and LB are the best lower and upper 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–T* instances, where the high fixed costs cause problems that require hours to solve to have very small values of $(UB - LB)/LB$ after (l, S) cuts are added. We therefore normalize the bounds and use the ratio $(UB - LB)/(LB - LB')$ as a relative measure of the duality gap, where LB' is the value of the LP relaxation that includes all the (l, S) cuts of the form (23). Since there are $\mathcal{O}(PT^2)$ inequalities of the form (23), the LP relaxation that includes these inequalities is easy to compute. Moreover, both $bc - prod$ and our code find all the binding cuts of the form (23), so for our tests this ratio is well–defined and positive.

We have compared $bc-prod$ with both cut–and–branch and branch–and–cut implementations of the cover and reverse cover inequalities defined in Propositions 10 and 11. We have $bc-prod$ generate at most 20 rounds of valid inequalities specifically designed for lot–sizing problems, and then 20 rounds of all its cuts (both lot–sizing specific and those from $bc-opt$). On all the instances tested, $bc-prod$ stops before generating 20 rounds of lot–sizing cuts, because the relative improvement in the objective function falls below the default tolerance necessary for it to continue. $bc-prod$ only generates cuts at the root node.

For our cut–and–branch routine we call Heuristic 3 until we do not find any more violated inequalities; when we do not find any more, we have XPRESS–MP generate 20 rounds of the cuts it uses. (Recall again that $bc-prod$ uses all the valid inequalities that XPRESS–MP does, and additional classes as well.) In our branch–and–cut routines, we follow the same procedure at the root node, and in both the cut–and–branch and branch–and–cut implementations, violated inequalities at the root node are added as rows into the matrix.

In the branch–and–cut implementation, 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 until there are no more violated cuts in the cut pool. We then call Heuristic 3 and reoptimize until we do not find a violated inequality. Thus, after the root node, we only add lot–sizing cuts to the cut pool, although we may reload violated cuts from other families into the LP matrix. In the branch–and–cut implementation, cuts generated in the tree are added to the cut pool, and we let XPRESS–MP remove inactive cuts from the matrix according to its default strategy.

Our computational results are displayed in Table 1. All the computations were carried out on a 450 megahertz PC with a dual Pentium processor, 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 give the branch–and–bound time of each algorithm in CPU seconds; if any of the algorithms did not find the optimal solution after 1800 seconds, we terminated the algorithm. We indicate this with $*$. In the last column of the table we list the optimal solution; if the optimal solution is not known, then we list the best known solution and mark it with $**$. We also list the number of nodes processed, and "Gap" refers to the ratio $\frac{UB-LB}{LB-LB'}$.

For our algorithms, we list X_{LS}, the objective function value at the root node after lot–sizing cuts (i.e., cover and reverse cover inequalities and inequalities of the form (23)) have been added. We do not list X_{LS} for $bc-prod$, because, with the exception of *mpp08a*, the addition of only 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 $bc-opt$) used by $bc-prod$. Optimal values marked with † are for optimal solutions that

Table 1: Computational Results for MCL

Problem	code	X_{LS}	X_{LP}	LB	UB	nodes	sec	Gap	LB'	Opt
tr6-15	$bc-prod$		37223	37721	37721	5756	22	0		
	C&B	37308	37390	37721	37721	2268	29	0		
	B&C			37721	37721	411	33	0	37213	37721
tr6-30	$bc-prod$		60973	61451	61765	219000	1800*	0.62		
	C&B	61065	61131	61569	61765	183300	1800*	0.31		
	B&C			61746	61746	30252	1175	0	60946	61746†
tr12-15	$bc-prod$		73881	74460	74644	172175	1800*	0.30		
	C&B	73926	73985	74464	74702	101975	1800*	0.39		
	B&C			74336	74634	8925	1800*	0.61	73848	74634
tr12-30	$bc-prod$		130177	130429	130658	85775	1800*	0.91		
	C&B	130263	130263	130459	130601	43700	1800*	0.50		
	B&C			130483	130604	5175	1800*	0.40	130177	130596†
tr24-15	$bc-prod$		136366	136457	136600	103625	1800*	1.57		
	C&B	136377	136377	136462	136562	70625	1800*	1.04		
	B&C			136474	136547	9025	1800*	0.68	136366	136509
tr24-30	$bc-prod$		287753	287807	287976	32350	1800*	3.02		
	C&B	287794	287794	287852	287939	20625	1800*	0.88		
	B&C			287878	287941	3350	1800*	0.50	287753	287929†
ytr6-15	$bc-prod$		5797	6049	6610	225075	1800*	1.98		
	C&B	5848	5880	6110	6507	159175	1800*	1.15		
	B&C			6182	6532	20265	1800*	0.84	5766	6465†
ytr6-30	$bc-prod$		19938	20404	24035	153850	1800*	7.73		
	C&B	20356	20397	20885	24669	93975	1800*	3.98		
	B&C			20989	23761	7525	1800*	2.76	19934	23409**
ytr12-15	$bc-prod$		6104	6169	6551	172875	1800*	4.78		
	C&B	6131	6143	6202	6581	152375	1800*	3.35		
	B&C			6207	6559	10900	1800*	2.98	6089	6524**
mset1ch	$bc-prod$		54518	54538	54538	97	1	0		
	C&B	54535	54538	54538	54538	1	1	0		
	B&C			–	–	–	–	–	54518	54538
mpp08a	$bc-prod$		8099	8430	8430	2501	8	0		
	C&B	8088	8143	8430	8430	1295	11	0		
	B&C			8430	8430	515	19	0	7997	8430

we have discovered and proven optimal for the first time (to the best of our knowledge); that is, $bc-prod$ is unable to solve these problems on our computer. For both of the instances in which the optimal value is not known, the values reported were found with long runs of our branch–and–cut code.

In general, our cut–and–branch implementation solves the test problems more effectively than $bc-prod$, and our branch–and–cut implementation performs better than either. This is especially true for those problems that do not solve in only a few seconds.

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 twenty minutes with the branch–and–cut algorithm. Also note that the new cuts we have defined, in conjunction with the (l, S) cuts and a few MIR and Gomory cuts, enable us to solve *mset1ch* at the root node.

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 listed here. For example, our branch–and–cut algorithm takes a very long time to process each node compared to the other two, and our cut–and–branch algorithm takes longer to process nodes than does $bc-prod$. For both of our algorithms, a more sophisticated cut management strategy, such as that employed in $bc-prod$, would probably enable nodes to be processed faster and thus improve performance.

Another possible way to improve upon these results is to see how the cover and reverse cover inequalities perform in conjunction with all the classes used by $bc-prod$. Also, it would be interesting to see what additional benefit that adding other families of cuts in the tree (in addition to (l, S) cuts and cover and reverse cover inequalities) would have.

7 Conclusion

We have derived new valid inequalities for MCL, a standard yet challenging lot–sizing model. These inequalities consider demand for multiple items over several periods simultaneously, and they incorporate capacity restrictions in their definition as well. 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 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, 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, Nemhauser, and Savelsbergh [1999a]). It therefore seems reasonable to expect that extending and applying our results to all of these models will be useful computationally.

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.

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. Technical Report CORE DP9778, Université catholique de Louvain, Louvain-la-Neuve, October 1997.

M. Diaby, H.C. Bahl, M.H. Karwan, and S. Zionts. A langrangean 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. Solving linear programs in integers. In R.E. Bellman and M. Hall, Jr, editors, *Combinatorial Analysis*, pages 211–216. American Mathematical Society, 1960a.

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.

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. 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. Facets, algorithms, and polyhedral characterizations for a multi–commodity flow model with setup times. Technical report, Georgia Institute of Technology, 1999a.

A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. On the capacitated lot–sizing and continuous 0–1 knapsack polyhedra. Technical Report TLI-LEC-98-11, Georgia Institute of Technology, March 1999b. To appear in European Journal of Operational Research.

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. Trigiero, 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.