This paper presents two new models to deal with different tooling requirements in the generic multiple-product assembly-system design (MPASD) problem and proposes a new branch-and-cut solution approach, which adds cuts at each node in the search tree. It employs the facet generation procedure (FGP) to generate facets of underlying knapsack polytopes. In addition, it uses the FGP in a new way to generate additional cuts and incorporates two new methods that exploit special structures of the MPASD problem to generate cuts. One new method is based on a principle that can be applied to solve generic 0-1 problems by exploiting embedded integral polytopes. The approach includes new heuristic and preprocessing methods, which are applied at the root node to manage the size of each instance. This paper establishes benchmarks for MPASD through an experiment in which the approach outperformed IBM's Optimization Subroutine Library (OSL), a commercially available solver.

(Programming: Integer, Cutting Planes; Production Scheduling: Flexible Manufacturing Line Balancing)

1. Introduction

To serve the specialized demands of customers in today’s highly competitive marketplace, most manufacturers assemble a variety of products, each in relatively low volume. In addition, the advancement of technology has initiated a trend in which product life cycles are becoming shorter. These and other competitive forces require manufacturers to design and redesign assembly systems frequently, so there is a widespread need for improved quantitative methods.

This paper describes a new branch-and-cut approach for the generic multiple-product assembly system design (MPASD) problem. The objective is to minimize the total cost of a system design,
including the variable cost of assembly operations and the fixed costs of activating assembly stations, purchasing machines, and providing tools. The problem is to prescribe the number of stations, the type of machine to be located at each station, the tooling to be provided at each machine, and the operations to be assigned to each machine. Each operation can be performed on any one of a (perhaps singleton) set of alternative machines and must be assigned to some machine at some station in accordance with precedence relationships. An appropriate set of tools must be provided at each machine to perform assigned operations. Each tool incurs a purchase cost and requires a certain storage space at a machine. Each machine has a finite time availability (i.e., capacity) and finite tool-storage space. The system design must provide sufficient capacity to assemble a specified set of products over the planning horizon.

The purpose of this paper is to present two new models, a branch-and-cut solution approach based on several new families of inequalities, and computational evaluation to establish benchmarks for MPASD. Our solution approach can be applied, with minor modifications, to both models. It consists of preprocessing at the root node and a branch-and-cut implementation that adds cuts at each node in the branch-and-bound (B&B) search tree. We generate cuts using the facet generation procedure (FGP) (Parija et al. 1999). In addition, we use the FGP in a new way to generate additional cuts and propose two new methods for generating strong cutting planes by exploiting special structures of the MPASD model. One of these new cut-generating methods is based on a principle that can be applied to solve generic 0-1 problems. Strong cutting planes are used to tighten the linear representation of the convex hull, \( \text{conv}(\bar{Q}) \), of feasible integer solutions, \( \bar{Q} \), to facilitate solution of an integer program using linear programming. The term "strong cutting planes" describes valid inequalities that represent high-dimensional faces of \( \text{conv}(\bar{Q}) \). The strongest possible cutting planes, which are of dimension \( \dim(\text{conv}(\bar{Q})) - 1 \), represent facets of \( \text{conv}(\bar{Q}) \) and provide a complete description of \( \text{conv}(\bar{Q}) \).

The Type I assembly line balancing (ALB) problem assigns tasks to a series of identical stations, minimizing the number of stations while observing task precedence relationships and a cycle time requirement (Baybars 1986). A number of studies have focused on the ALB problem and specialized
B&B algorithms have been shown to solve selected test problems effectively (e.g., Assche and Herroelen 1979, Johnson 1988, Hackman et al. 1989, Hoffmann 1992, Scholl 1995, Ugurdag et al. 1997, and Scholl and Klein 1997).

Assembly system design (ASD) is an extension of ALB in which a robot or other type of “machine” must be assigned to each station, so stations may not be identical. Each task may be performed by a set of alternative machine types, and selecting the type of machine to locate at each station entails an additional level of combinatorics, making ASD more challenging than ALB. Task precedence relationships and a cycle time requirement must still be observed. Pinnoi and Wilhelm (1997a) proposed a hierarchical family of models with the goal of dealing with a variety of ASD problems by exploiting the embedded ALB polytope that is common to a variety of formulations. This paper extends the family of models envisioned by Pinnoi and Wilhelm.

The single-product ASD (SPASD) problem is typically formulated with the objective of minimizing total cost (e.g., Ghosh and Gagnon 1989, Graves and Lamar 1983), since the design with the minimum number of stations need not minimize cost. Wilhelm (1999) developed a column-generation approach to the SPASD problem. His approach prescribes the sequence for performing assembly operations, explicitly addressing tool changes, which, for example, affect the productivity of some robotic assembly systems. Other research has proposed cutting-plane methods for the SPASD problem. Kim and Park (1995) addressed a version of the SPASD problem associated with robotic lines. They assigned tasks, parts, and tools to robotic cells (stations) with the objective of minimizing the total number of cells activated. Their cut-and-branch algorithm included preprocessing and added violated cover cuts (Nemhauser and Wolsey 1988) — all at the root node.

Pinnoi and Wilhelm (1998) devised a branch-and-cut approach that employed specialized preprocessing methods. They showed that the node-packing polytope is a relaxation of the ALB polytope and generated a set of cuts based on this relationship. Pinnoi and Wilhelm (1997b) proposed a related approach to the workload-smoothing problem, a variation of ALB that minimizes the maximum idle time on any station, “smoothing” workloads assigned to stations. They generated violated clique and cover
inequalities associated with an intersection graph, which they formed from precedence and cycle-time constraints. This prior work demonstrated the effectiveness of using the node-packing relaxation. In contrast, our model incorporates tooling requirements, and we develop strong cutting planes by exploiting special structures that result. Neither tooling requirements nor these new cut-generating methods were considered by Pinnoi and Wilhelm (1997b, 1998).

Gadidov and Wilhelm (2000) recently devised a new branch-and-cut approach for the SPASD problem. Their approach, which consisted of a heuristic, preprocessing techniques, and two types of cutting-planes, outperformed OSL in solving a set of test problems. The current paper extends these heuristic and preprocessing techniques to address multiple products and tooling requirements. It also proposes entirely different methods for generating strong cuts. Gadidov and Wilhelm (2000) devised one type of cut based on the time required to complete two tasks that are not related through precedence relationships and incorporated the facet-generation procedure (FGP) (Parija et al. 1999). They added cuts of the former type at the root node and those of the latter type at other nodes in the search tree. The current paper employs the FGP, but its primary contributions include a new way to implement the FGP to generate additional cuts and two new methods for generating cuts based on special structures of the MPASD problem that were not incorporated in the earlier SPASD model.

The FGP computes facets of an underlying polytope, \( \mathcal{R} \), which must be selected so that a subproblem involving a linear objective function can be optimized effectively over it. Parija et al. (1999) applied the FGP to a single constraint so that \( \mathcal{R} \) was a knapsack polytope and the subproblem can be solved in pseudopolynomial time. Given a fractional solution to the linear relaxation of an integer program, \( \mathbf{f}^* \) (an \( \mathbf{n} \) vector where \( \mathbf{n} \) is the number of decision variables in the relaxed problem), the FGP computes the coefficients of a hyperplane, \( H \), that represents a facet of \( \mathcal{R} \) and separates \( \mathbf{f}^* \) from \( \mathcal{R} \). The FGP uses column generation to solve the following linear program (LP):

**Problem (P):**

\[
\begin{align*}
\text{Problem (P): } & z^* = \min \left\{ \sum_{X \in \text{ext} \mathcal{R}} \alpha_X \mid \sum_{X \in \text{ext} \mathcal{R}} \alpha_X \mathbf{x} = \mathbf{f}^* ; \alpha_X \geq 0, \mathbf{x} \in \text{ext} \mathcal{R} \right\}.
\end{align*}
\]
Here, \( \text{ext } \mathcal{R} \) is the set of extreme points of \( \mathcal{R} \), each of which is represented by an \( \mathbf{n} \) vector, \( \mathbf{x} \). The subproblem prices out nonbasic columns, generating the column \( \mathbf{x} \in \text{ext } \mathcal{R} \) with the most negative reduced cost. If this reduced cost is nonnegative, the LP optimality criterion is satisfied and the current basis, \( \mathbf{B}^* \), which is composed of \( \mathbf{n} \) columns \( \mathbf{x} \in \text{ext } \mathcal{R} \), is optimal. The \( \mathbf{n} \) vector of coefficients that define the gradient of \( H \), \( \mathbf{w}^* \), may be determined by solving \( \mathbf{w}^* \mathbf{B}^* = 1 \). \( H = \{ \mathbf{x} : \mathbf{w}^* \mathbf{x} \leq 1, \mathbf{x} \in \mathbb{R}^n \} \) supports the \( \mathbf{n} \) linearly independent points (columns that comprise \( \mathbf{B}^* \)) so that it represents a facet of \( \mathcal{R} \). Further details may be found in Parija et al. (1999), who proposed the FGP and demonstrated its efficacy solving 0-1 problems in MIPLIB, and in Gadidov and Wilhelm (2000), who gave an intuitive description of the FGP and incorporated it in an approach for the SPASD problem.

The multiple-product ASD (MPASD) problem has received relatively little attention (Ghosh and Gagnon 1989). Variations have been addressed by Lagrangian relaxation (Kuroda and Tozato 1987), B&B (Pinto et al. 1983), dynamic programming (Chakravarty and Shtub 1986), mixed integer programming (Peters 1991), integer programming combined with queueing models (Lee and Johnson 1991), and column generation (Kimms 1998). Graves and Redfield (1988) enumerated feasible workstation designs and prescribed MPASD by solving a shortest-path problem.

We have organized the body of this paper in four sections. In the next section we introduce notation and our mathematical formulations of two versions of the MPASD problem. Section 3 describes our solution approach. We discuss implementation issues, test problems, and computational results in Section 4. The last section offers final remarks and conclusions.

2. Model Formulation

As the ALB model is a generic representation of line-balancing issues, our models are generic representations of MPASD. These models may be applied in a variety of settings, including robotic assembly of airframes e.g., Huber (1984), assembly of automotive subassemblies (e.g., Graves and Redfield 1988) and assembly of electronic products (e.g., Nof et al. 1997).
We adopt the traditional approach, combining the operation precedence relationships for the set of products, $P$, into one digraph (McCaskill 1972). In this digraph, each node represents an operation $o \in O$ that is required to assemble a subset of products $P_o \subseteq P$. Analogous to the cycle-time constraint that must be observed in SPASD, machine availability (i.e., capacity) must be observed in MPASD. This more general restriction is needed because products may be produced in different volumes and, thus, have different impacts on capacity.

This section presents models for two variations of MPASD that deal with different tooling requirements (e.g., Stecke 1983, Graves and Lamar 1983, Ammons et al. 1985, Kim and Park 1995, and Nof et al. 1997). These variations represent typical, practical requirements, and we have designed our cut-generating methods to exploit the resulting structures.

One set of assumptions is common to both of our models. We assume that all parameters are known with certainty: (a) $f_s$, the fixed cost of activating station $s \in S$; (b) $c_m$, the fixed cost of machine type $m \in M$; (c) $A_m$, the time availability of machine $m$; (d) $B_m$, the tool-storage space at machine $m$; (e) $\bar{c}_l$, the fixed cost of providing tool type $l \in L$; (f) $b_l$, the storage space required by tool type $l$; and (g) $V_p$, the production volume for product $p \in P$. We assume that operation $o \in O$ is completely specified, including the subset of products that require it, $P_o$; the set of operations that immediately precede it in the precedence digraph, $IP_o$; the set of associated tools $L_o \subseteq L$; the set of alternative machines that can perform it, $M_o$; and its processing time on each $m \in M_o$, $t_{mo}$, which incurs variable cost $v_{mo}$. We now summarize notation.

<table>
<thead>
<tr>
<th>Indices</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l$ = tool type</td>
</tr>
<tr>
<td>$m$ = machine type</td>
</tr>
<tr>
<td>$o$ = operation</td>
</tr>
<tr>
<td>$p$ = product type</td>
</tr>
<tr>
<td>$s$ = station</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_m$ = time availability of machine type $m$ (hours over the planning horizon)</td>
</tr>
<tr>
<td>$B_m$ = tool space provided on machine type $m$ (square inches or number of slots)</td>
</tr>
</tbody>
</table>
$b_l = $ space required by tool $l$ (square inches or number of slots)

$\tilde{c}_l = $ fixed cost of purchasing a tool of type $l$ ($)

c_m = $ fixed cost of purchasing a machine of type $m$ ($)

$f_S = $ fixed cost of activating station $s$ ($)

t_{mo} = $ processing time of operation $o$ on machine type $m$ (hours / operation)

$v_{mo} = $ variable cost of performing operation $o$ on machine type $m$ ($/operation)

$V_p = $ production volume required for product type $p$ (number of products over horizon)

Sets

$IP_o :$ set of immediate predecessors of operation $o$

$L :$ set of tools

$L_o :$ set of tools required to perform operation $o$

$M :$ set of machine types

$M_o :$ set of machine types to which operation $o$ can be assigned

$O :$ set of operations

$O_l :$ set of operations that require tool $l$

$O_m :$ set of operations that can be assigned to machine type $m$

$P :$ set of types of products

$P_o :$ set of products that require operation $o$ ($P_o \subseteq P$)

$S :$ set of stations

S = \{1,...,|S|\}

Binary Decision variables

$x_{mos} = 1$ if operation $o$ is assigned to a machine of type $m$ at station $s$, 0 otherwise

$y_{mx} = 1$ if machine type $m$ is assigned to station $s$, 0 otherwise

$z_{lms} = 1$ if tool $l$ is assigned to a machine of type $m$ at station $s$, 0 otherwise.

2.1 MPASD$_1$ Formulation

Our first model assumes that operation $o \in O$ requires a set of tools, $L_o \in L$, and that each tool can be used by several operations. MPASD$_1$ explores tradeoffs between alternative operation assignments and the fixed costs incurred for stations, machines, and tooling. For example, instead of assigning operations that require tool $l$ to different stations, and thus requiring the provision of tool $l$ at each of these stations, it may be less costly to explore alternatives that assign all operations that require tool $l$ to the same station.
\[
\text{MPASD}_i : \min \ Z_{BP} = \sum_{s \in S} \sum_{m \in M} (c_m + f_s) y_{ms} + \sum_{s \in S} \sum_{o \in O} \sum_{m \in M_o} \sum_{p \in P_o} V_p v_{mo} x_{mas} + \sum_{l \in L} \sum_{m \in M} \sum c_i z_{lms}
\]

Subject to:
\[\sum_{m \in M_o} \sum_{s \in S} x_{mos} = 1 \quad o \in O \quad (1)\]
\[\sum_{m \in M_o} \sum_{s \in S} s x_{mos} - \sum_{m \in M_o} \sum_{s \in S} s x_{mos'} \geq 0 \quad o \in O, o' \in IP_o \quad (2)\]
\[A_m y_{ms} - \sum_{o \in O_a} t_{mo} x_{mos} \sum_{p \in P_o} V_p \geq 0 \quad m \in M, s \in S \quad (3)\]
\[z_{lms} \leq y_{mos} \geq 0 \quad o \in O, m \in M_o, l \in L_o, s \in S \quad (4)\]
\[B_m y_{ms} - \sum_{l \in L} b_l z_{lms} \geq 0 \quad m \in M, s \in S \quad (5)\]
\[\sum_{m \in M} y_{ms} - \sum_{m \in M} y_{m(s+1)} \geq 0 \quad s \in S \quad (6)\]
\[\sum_{m \in M} y_{ml} \leq 1 \quad (7)\]
\[x_{mos}, y_{ms}, z_{lms} \in \{0,1\} \quad o \in O, m \in M_o, l \in L_o, s \in S \quad (8)\]

The objective function minimizes the total cost of the system design, including the variable cost of performing operations and the fixed costs of activating stations, purchasing machines, and providing tools. This model incorporates all logical issues involved in MPASD. Constraint (1) requires each operation to be processed at one station, and constraint (2) invokes operation precedence relationships. These constraints parallel those used in ALB except that operations in the multiple-product case are called tasks in the single-product ALB problem. Constraint (3) imposes machine availability (i.e., capacity) limitations and is the multiple-product analog of the ALB cycle-time requirement. The ALB polytope is embedded in MPASD through constraints (1)-(3) so that the types of cuts derived by Pinnoi and Wilhelm (1998) from the relationship between the node-packing and ALB polytopes could be applied to the MPASD. Since the efficacy of the former set of cuts is known, this paper focuses on the set of new cuts that are related to machine availability and tooling requirements.
Neither Pinnoi and Wilhelm (1998) nor Gadidov and Wilhelm (2000) dealt with tooling requirements, a central feature of MPASD. Constraint (4) is a stronger formulation for assuring that appropriate tooling is provided for each operation than was proposed in the models of Pinnoi and Wilhelm (1997a). It assures that the set of tools required by operation \( o, L_o \), are assigned to the machine-station \( (ms) \) combination at which \( o \) is assigned, and it permits tool \( l \) to be used by all operations that require that tool \( (o \in O; l \in L_o) \) if all are assigned to the same \( ms \) combination. Constraint (5) invokes the tool-area storage limitation at each machine.

Constraint (6) assures that station \( s \) will be activated before station \( s + 1 \), precluding symmetry. Together, constraints (6) and (7) assure that at most one machine will be located at each station. These constraints improve earlier models, relying on the the sequence \( \{\sum_{ms} y_{ms}\} \), being non-increasing.

The number of variables in an instance of \( MPASD_1 \) depends directly on the number of tools, \( |L| \); machine types, \( |M| \); operations, \( |O| \); and stations, \( |S| \). The number of constraints depends similarly on these parameters as well as on the number of predecessor relationships for each operation, \( |IP_o| \). The number of products, \( |P| \), does not affect problem size directly. However, as the number of products increases, the number of operations can also be expected to increase, giving rise to additional precedence relationships, \( IP_o \), tooling requirements, \( L_o \), and machine types, \( O_m \), to accommodate the additional operations.

2.2 \( MPASD_2 \) Formulation

Our second model assumes that tools are provided in pre-specified tool kits and that any one of several alternative tool kits, each requiring different space and fixed cost, can perform a given operation. However, each tool kit can perform just one operation. For cases in which each operation may require a number of tools, dealing with tool kits may avoid a combinatorial aspect of the problem. This model explores tradeoffs between the cost of tooling and the storage space required. For example, providing costly tool kits that require little space may allow the system to operate with fewer machines and stations.
This model is not overly restrictive because, if a tool kit could perform two operations, it may be preferable to combine them for technical reasons (e.g., Donohue et al. 1991). The notation for this model is the same as before, except that \( l \) now denotes a tool kit:

\[
\begin{align*}
  l & = \text{tool kit} \quad (l \in L) \\
  b_l & = \text{space required by tool kit} \ l \quad (\text{square inches or number of slots}) \\
  \tilde{c}_l & = \text{fixed cost of purchasing a tool kit of type} \ l \quad ($) \\
  L & = \text{set of tool kits} \quad (L = \bigcup_{o \in O} L_o) \\
  L_o & = \text{set of tool kits that can perform operation} \ o \quad (L_o \subseteq L)
\end{align*}
\]

\[
\text{MPASD}_2: \min Z_{BP} = \sum_{s \in S} \sum_{m \in M} (c_m + f_s) y_{ms} + \sum_{s \in S} \sum_{o \in O} \sum_{m \in M_o} \sum_{p \in P_o} V_{p}v_{mo}x_{mos} + \sum_{l \in L} \sum_{m \in M} \sum_{s \in S} \tilde{c}_l z_{lms}
\]

Subject to: constraints (1)-(3), (5)-(8) and

\[
\sum_{l \in L_o} z_{lms} - x_{mos} \geq 0 \quad o \in O, m \in M_o, s \in S.
\]

As in \( \text{MPASD}_1 \), the objective function minimizes the total cost of system design. Constraints (9) replace (4), ensuring that some tool kit for operation \( o \) is assigned to combination \( ms \) along with \( o \).

### 3. Solution Approach

This section describes our solution approach, including the heuristic and pre-processing methods, the two new methods for generating strong cuts, and the strategy for implementation, including the new way in which we use the FGP to generate additional cuts. We demonstrate our methods in relationship to \( \text{MPASD}_1 \); they can be adapted easily for \( \text{MPASD}_2 \).

#### 3.1 Heuristic and Pre-Processing Methods

Our heuristic prescribes an upper bound on the value of the optimal solution, and our pre-processing methods determine an upper bound on the number of stations and the earliest and latest stations to which operation \( o \in O \) can be assigned.

##### 3.1.1 Heuristic

We devised a simple heuristic to find an upper bound, \( Z_{BP}^{\ast} \), on the value of the optimal solution, \( Z_{BP}^{\ast} \), of \( \text{MPASD}_1 \) (i.e., \( Z_{BP}^{\ast} \leq Z_{BP} \)). At each iteration, the heuristic assigns the operation that can “fit” on a
station while incurring the least possible cost.

An operation is a candidate for assignment if all of its predecessors (if any) have been assigned. Indices of current candidates are maintained in the set $J$. When no candidate will fit on the current station, the next station, $s$, is activated. The machine type $m$ assigned to station $s$ is the one that can process a candidate at minimal cost: $m = \arg\min \left\{ c_{m'} + \left( \sum_{p \in P_{m'}} V_p \right) v_{m'o} : o \in J, m' \in M_o \right\}$. We determine the cost of assigning each candidate $o \in J \cap O_m$ to this $ms$ combination using $\tilde{p}_o = \left( \sum_{p \in P_m} V_p \right) v_{mo} + \sum_{l \in L_m \setminus L_{ms}} \tilde{c}_l$ where $L_{ms}$ is the set of tools already assigned to $ms$. If $o$ cannot be assigned ($m \notin M_o$) or does not fit (due to tool storage and/or availability limitations), $\tilde{p}_o = \infty$. At each iteration, we assign an operation from $\arg\min \left\{ \tilde{p}_o : o \in J \cap O_m \right\}$ and update $J$, $L_{ms}$, and $\tilde{p}_o$ appropriately. When no candidate will fit on $ms$ (i.e., $\min \left\{ \tilde{p}_o : o \in J \cap O_m \right\} = \infty$) we activate the next station as described above.

### 3.1.2 Pre-Processing Methods

We use $Z_{BPH}$ to obtain an upper bound on the optimal number of stations, $N_{opt}$. This allows us to fix some variables to zero, managing the size of the problem. We note that this bound is not as pivotal as it is in the ALB problem because the optimal MPASD solution does not necessarily use the minimum number of stations. For example, MPASD must determine an appropriate tradeoff between using a larger number of low-cost machines of lesser capability with using fewer higher-cost machines of greater capability.

Assuming that $N_{opt}$ stations have been activated, upper and lower bounds on $Z_{BP}$ are:

$$\sum_{s=1}^{N_{opt}} f_s + N_{opt} \min \left\{ c_m : m \in M \right\} + \sum_{o \in O} \sum_{p \in P_o} V_p \min \left\{ v_{mo} : m \in M_o \right\} + \sum_{l \in L} \tilde{c}_l \leq Z_{BP} \leq Z_{BPH} \quad (10)$$

The first term in (10) represents the cost to activate $N_{opt}$ stations, the second represents the lowest possible cost of purchasing $N_{opt}$ machines, the third represents the lowest possible variable cost of performing all the operations, and the fourth represents the cost of purchasing all required tools. Re-expressing (10), we can easily obtain an upper bound for $N_{opt}$ from

11
\[
\sum_{s=1}^{N_{opt}} f_s + N_{opt} \min \{ c_m : m \in M \} \leq Z_{\text{BPH}} - \sum_{o \in O} \sum_{p \in P} V_p \min \{ v_{mo} : m \in M_o \} - \sum_{l \in L} c_l.
\]

We used \( N_{opt} \) to manage problem size by defining the earliest (\( ES_o \)) and latest stations (\( LS_o \)) to which operation \( o \) can be assigned. Let \( PR_{os} \) (\( SU_{os} \)) denote the set of predecessors (successors) of operation \( o \) whose earliest (latest) station is \( s \) (\( s \in S \)):

\[
PR_{os} = \{ \tilde{o} : \tilde{o} = \text{predecessor of } o \text{ and } ES_{\tilde{o}} = s \} \text{ and }
SU_{os} = \{ \tilde{o} : \tilde{o} = \text{successor of } o \text{ and } LS_{\tilde{o}} = s \}.
\]

Now, \( ES_o \) and \( LS_o \) may be found recursively using the following rules:

\[
ES_o = \min \{ s_o : (1) \text{ there exists } m \in M_o , \ (2) \ ES_{\tilde{o}} \leq s_o \text{ for all predecessors } \tilde{o} \text{ of } o , \ (3) \ o \text{ fits on } ms_o \text{ with any predecessor } \tilde{o} \in PR_{os} \}
\]

(relative to tool-space and availability limitations)

\[
LS_o = \max \{ s_o \leq N_{opt} : (1) \text{there exists } m \in M_o , \ (2) \ LS_{\tilde{o}} \geq s_o \text{ for all successors } \tilde{o} \text{ of } o , \ (3) \ o \text{ fits on } ms_o \text{ with successor } \tilde{o} \in SU_{os} \}
\]

(relative to tool-space and availability limitations).

We now describe our two new methods for generating strong cutting planes.

### 3.2 New Methods for Generating Strong Cutting Planes

The first type of cut is based on the structure of our MPASD models. The second type is based on identifying an integral polytope that is embedded in a model. Both focus on tooling requirements. We demonstrate the latter type relative to MPASD, but underlying principles can be applied to integer programs in general.

#### 3.2.1 Cuts Based on Tooling Requirements

This section describes our new cutting planes, separation problems, lifting procedures, and implementation strategy. Our first new method for generating cuts exploits structures related to tooling constraints (4) or (9). The cuts may be non-trivial (i.e., some cannot be obtained from a single knapsack constraint (5)). Consider a
minimal cover inequality associated with $L \subseteq L$ in constraint (5) for a given machine type $m = m_0$ and station $s = s_0$:

$$\sum_{l \in L} z_{lm_{0} s_{0}} \leq |L| - 1. \quad (12)$$

Given $x^* = (x^*_1 \ y^*_1 \ z^*_1)$, a fractional solution to the linear relaxation of $MPASD_1$ including any applicable generated cuts, the most violated inequality of type (12) can be identified by solving the well-known separation problem (Nemhauser and Wolsey 1988):

**Problem (SP):** $\text{Min } Z_{\text{SP}} = \left\{ \sum_{l \in L} (1-z^*_{lm_{0} s_{0}}) u_l : \sum_{l \in L} b_l u_l \geq B_{m_0} + 1; u_l \in \{0,1\}, l \in L \right\}.$

If $Z_{SP}^* < 1$ Problem (SP) identifies the violated inequality $L^* = \left\{ l : u_l^* = 1, l \in L \right\}$.

### 3.2.1.1 Cutting Planes

In addition to the strong cutting planes that may be obtained by lifting cover inequalities (12), a new type of cut may be derived from minimal cover $\tilde{L}$ in inequality (12) by considering a relationship between $z_{lm_{0} s_{0}}$ and $x_{m_0 os_{0}}$ decision variables induced by tooling requirements. Let $\tilde{O} (\subseteq O)$ be the set of all operations that require some tool from set $L$, so that $\tilde{O} = \{ o \in O : \exists l \in \tilde{L} \text{ such that } o \in O_l \}$. That is, each operation $o \in \tilde{O}$ requires at least one tool $l \in \tilde{L}$. Note that each tool $l \in \tilde{L}$ may be required by a number of operations $o \in \tilde{O}$. From (12), not all the tools in $\tilde{L}$ can be assigned to machine type $m_0$. It follows that not all of the operations in $\tilde{O}$ can be assigned to machine type $m_0$, so

$$\sum_{o \in \tilde{O}} x_{m_0 os_{0}} \leq |\tilde{O}| - 1 \quad (13)$$

is an induced relationship, a cover that is valid for $MPASD_1$. The next subsection describes a separation problem to identify the most violated minimal cover (14) in which $\tilde{K} \subseteq \tilde{O}$:

$$\sum_{o \in \tilde{K}} x_{m_0 os_{0}} \leq |\tilde{K}| - 1. \quad (14)$$
3.2.1.2 Separation

Given \( \mathbf{f}^* = (x^* \ y^* \ z^*) \), \( \tilde{L} \), and the associated \( \tilde{O} \), the most violated inequality of type (14) may be found by solving a separation problem that is a set covering (SC) problem of the form:

\[
\text{Problem (SC)}: \quad Z_{SC} = \min \left\{ \sum_{\sigma \in \tilde{O}} (1-x_{m,o_\sigma}^*) w_o : \sum_{\sigma \in \tilde{O}} a_{l,o} w_o \geq 1, \ l \in \tilde{L} ; \ w_o \in \{0,1\}, \ o \in \tilde{O} \right\}
\]

in which \( a_{l,o} = 1 \) if \( l \in \tilde{L} \), 0 otherwise. The set-covering constraints assure that Problem (SC) will prescribe a subset of operations that require all tools in set \( \tilde{L} \). If \( Z_{SC}^* < 1 \), the optimal solution to Problem (SC) defines the inequality of type (14) that is most violated by the current fractional solution \( \mathbf{f}^* = (x^* \ y^* \ z^*) \), giving the minimal cover \( \tilde{K} = \{ o : w_o^* = 1, o \in \tilde{O} \} \).

We used the greedy heuristic to solve Problem SC (Nemhauser and Wolsey 1988). At each of its iterations, this heuristic accepts the column that satisfies the largest number of uncovered rows per unit cost. It stops when accepted columns cover all rows. The greedy heuristic is not new, but this application of the set-covering problem is, to our knowledge.

3.2.1.3 Lifting

Inequality (14) may be strengthened by lifting with respect to variables \( x_{m,o_\sigma} \) for \( o \in \tilde{O} \setminus \tilde{K} \) as follows.

First, we note that the greedy heuristic solution to Problem (SC) is “minimal” in the sense that, for any proper subset \( \tilde{I} \) of \( \tilde{K} \), \( \bigcup_{i=1}^{\tilde{I}} \tilde{L}_o \neq \tilde{L} \). If \( o \in \tilde{O} \setminus \tilde{K} \) is such that, for any subset \( \tilde{I} \) with

\[
|\tilde{I}| = |\tilde{K}| - 1, \ \tilde{L}_o \bigcup_{i=1}^{\tilde{I}} \tilde{L}_o = \tilde{L},
\]

where \( \tilde{L}_o = \tilde{L} \bigcap L_o \) is the subset of tools in \( \tilde{L} \) required by operation \( o \),

\[
\sum_{k \in K} x_{m,o_{k_o}} + x_{m,o_{\sigma}} \leq |\tilde{K}|-1 \quad (15)
\]

is also a valid inequality for \( MPASD_1 \). This logic can yield completely non-trivial valid inequalities for \( MPASD_1 \) as shown by the following example.
**Example.** Given $f^* = (x^*, y^*, z^*)$, $m_0$, and $s_0$, assume that Problem (SP) identifies the minimum cover $L = \{l_1, l_2, l_3\}$ so that these three tools require storage space that is larger than the space available on machine $m_0$. A valid inequality of type (12) is thus given by $\sum_{l \in L} z_{l m_0 s_0} \leq 2$. Assume that the associated set of operations $\tilde{O} = \{o_1, o_2, o_3\}$; that $L_{o_1} = \{l_1, l_2\}$, $L_{o_2} = \{l_2, l_3\}$ and $L_{o_3} = \{l_3, l_4\}$; and that, without loss of generality (wlog), the fractional solution to the linear relaxation satisfies

$$x_{m_0 o_1 s_0}^* \leq x_{m_0 o_2 s_0}^* \leq x_{m_0 o_3 s_0}^*.$$ 

The greedy heuristic yields the feasible solution $\begin{bmatrix} w_{o_1} & w_{o_2} & w_{o_3} \end{bmatrix} = [0 \ 1 \ 1]$, which is also optimal in this case, so that $\tilde{K} = \{o_2, o_3\} \subset \tilde{O} = \{o_1, o_2, o_3\}$; and inequality (14) becomes

$$x_{m_0 o_2 s_0} + x_{m_0 o_3 s_0} \leq 1. \quad (16)$$

We may select $\tilde{I} = \{o_2\} \subset \tilde{K} = \{o_2, o_3\}$ since $|\tilde{I}| = |\tilde{K}|-1$ and $\tilde{L}_{o_1} \cup \tilde{L}_{o_2} = \{l_1, l_2\} \cup \{l_2, l_3\} = \tilde{L} = \{l_1, l_2, l_3\}$ where $o_1 \in \tilde{O} \setminus \tilde{K}$. Inequality (16) may be lifted with respect to $x_{m_0 o_1 s_0}$ to obtain a valid inequality of type (15):

$$x_{m_0 o_1 s_0} + x_{m_0 o_2 s_0} + x_{m_0 o_3 s_0} \leq 1. \quad (17)$$

We note that (17) is a valid inequality, even though assigning operation $o_1 \in \tilde{O} \setminus \tilde{K}$ along with operations in set $\tilde{K}$ does not exceed the capacity of machine type $m_0$, as invoked by an inequality of type (3):

$$A_{m_0} - \sum_{p \in P_{o_1}} V_p t_{m_0 o_1} - \sum_{p \in P_{o_2}} V_p t_{m_0 o_2} - \sum_{p \in P_{o_3}} V_p t_{m_0 o_3} \geq 0.$$

### 3.2.1.4 Additional Cutting Planes
Instead of starting with a minimal cover (12) associated with constraint (5), we could begin with a minimal cover of type (14) associated with constraint (3). Starting with inequality (14) and invoking relationships induced by tooling requirements, we obtain a new cut in terms of $z_{lm_{1} o_{2} o_{3}}$ variables.

For each $o \in \tilde{O}$ let $\tilde{L}_{o} = L \setminus \bigcup_{o' \in O, o' \neq o} L_{o}$ denote the set of tools that are required by $o$ and not by any other operation $o' \in O \setminus \{o\}$, and assume that none of the sets $\tilde{L}_{o}$ is empty. From (13), not all the operations in $\tilde{O}$ can be assigned to machine type $m_{o}$, and it follows that not all the tools in $\bigcup_{o \in \tilde{O}} \tilde{L}_{o}$ can be assigned to machine type $m_{o}$, so an induced relationship that is valid for $MPASD_{1}$ is given by

$$\sum_{o \in \tilde{O}} \sum_{l \in \tilde{L}_{o}} z_{lm_{o} o_{2} o_{3}} \leq \min \{ |\tilde{L}_{o}| : o \in \tilde{O} \}.$$

### 3.2.2 Generating Cuts Based on Embedded, Integral Polytopes.

Our second new method is based on identifying an embedded integer polytope and lifting over it. The traditional method lifts over the knapsack polytope (e.g., Nemhauser and Wolsey 1988). One advantage of our method is that it lifts by optimizing an LP in polynomial time. A second advantage is that the integer polytope is defined by a number of constraints and may be expected to provide a tighter relaxation of the convex hull of feasible integer solutions than would, say, a single knapsack polytope.

We begin this subsection by introducing several claims and a lemma for $MPASD_{1}$ and an equivalent lemma for $MPASD_{2}$. Proofs are given in the Appendix. Subsequently, we describe how the integer polytopes can be used in lifting to generate strong cuts.

#### 3.2.2.1 Embedded Integer Polytopes

Lemma 1 establishes the integrality of embedded polytope $\mathcal{N}^{(1)}$:

**Lemma 1.** The polytope $\mathcal{N}^{(1)} = \{ (x^{T}, z^{T}) :$

$$\sum_{s=1}^{n} \sum_{m \in M} x_{m, o_{s}} + \sum_{s=n+1}^{S} \sum_{m \in M} x_{m, o_{s}} \leq 1 \quad n = 1, \ldots, |S| - 1, \quad o \in O, \quad o' \in IP_{o} \} \quad (18)$$
\[ x_{mos} - z_{lms} \leq 0 \quad o \in O, \ m \in M_o, \ l \in L_o, \ s \in S \]

(19)

\[-x \leq 0 \]  \quad (20)

\[-z \leq 0 \]  \quad (21)

\[ z \leq 1 \} \text{ is integral.} \]  \quad (22)

Inequalities (18) represent an alternative way of formulating precedence relationships (2), inequalities (19) reiterate inequalities (4), and inequalities (20)-(22) arise in the linear relaxation of binary restrictions (8). A relaxation of MPASD, the set of inequalities (18)-(22) admit all solutions that are feasible with respect to (1)-(8) and others as well. For example, (18)-(22) do not assure that each operation is assigned to a single ms combination as does (1) so that an operation can be unassigned or, in certain cases, assigned to two stations. In addition, inequalities (18)-(22) do not invoke machine-availability limitations (3) or tool-holding capacities (5) and do not require a single machine to be located at each station (6)-(7).

Using graph theory, Chaudhuri et al. (1994) proved that the polytope defined by (18) and (20) is integral (see also Ugurdag et al. 1997). We employ an alternative proof strategy and deal with inequalities (19), (21), and (22) as well. First, we introduce some new notation:

\[ \bar{n} = \bar{n}_x + \bar{n}_z = \text{total number of variables} \]

\[ \bar{n}_x = \text{number of } x_{mos} \text{ variables} \quad (\text{for } o \in O, m \in M_o, s \in S) \]

\[ \bar{n}_z = \text{number of } z_{lms} \text{ variables} \quad (\text{for } o \in O, l \in L_o, m \in M_o, s \in S) \]

\[ \mathbb{Q} = \text{set of integer points that are feasible with respect to (18)-(22).} \]

We define \( \bar{n} \) unit vectors, each of dimension \( \bar{n} \): \( e_{mos} \) (\( e_{lms} \)) is an \( \bar{n} \) vector of zeroes except it has 1 in the unique position determined by index mos (lms). We define an agreeable assignment as one that satisfies inequalities (18) with each operation assigned to at most one ms combination. We assume that operation precedence relationships are consistent; that is, if operation \( o \) is a predecessor of \( o' \) and if \( o' \) is a predecessor of \( o'' \), then \( o \) is a predecessor of \( o'' \). For example, the assembly tree in which each operation has at most one successor is consistent.
Our first claim implies that each extreme point in \( \mathcal{R}^{(1)} \) (i.e., each extreme point in \( \text{conv}(\mathcal{Q}) \)) is formed by the intersection of \( \mathbf{n} \) linearly independent, binding inequalities (Bazaraa et al. 1990) defining \( \mathcal{R}^{(1)} \) and \( \text{conv}(\mathcal{Q}) \), respectively.

**Claim 1** \( \dim(\mathcal{R}^{(1)}) = \mathbf{n} \) and \( \dim(\text{conv}(\mathcal{Q})) = \mathbf{n} \).

Each extreme point of \( \mathcal{R}^{(1)} \) is formed by a subsystem of \( \mathbf{n} \) linearly independent inequalities in (18)-(22) that hold at equality. We prove Lemma 1 using a strategy that shows that each feasible subsystem forms an extreme point that is integral, so that \( \mathcal{R}^{(1)} \) is integral. If more than \( \mathbf{n} \) inequalities hold at equality at an extreme point, \( \mathbf{n} \) linearly independent inequalities imply each of those in excess, whether it is essential or redundant. We show that, at each extreme point, each of the \( \mathbf{n} \) variables is related to a unique binding inequality and that this set of equalities is linearly independent. We begin by considering feasible subsets that require \( x_{mos} = 0 \) for all \( mos \) combinations.

**Claim 2** Each subsystem formed by fixing two types of inequalities to equalities
(a) all \( \mathbf{l} \times \mathbf{m} \) inequalities (20) and
(b) for each \( \mathbf{l} \times \mathbf{m} \), select an inequality of type (21) fixing \( z_{lms} = 0 \) or of type (22) fixing \( z_{lms} = 1 \) gives \( \mathbf{n} \) linearly independent, binding inequalities that define an extreme point, which is integral.

The null solution \( \left( x^T, \ z^T \right) = \left( 0^T, \ 0^T \right) \) is a special case in which inequalities (20) and (21) are all fixed at equality. \( MPASD_1 \) assumes that each operation requires a set of tools and that each tool may be used by more than one operation. Claim 2 highlights the fact that each \( z_{lms} \) is constrained on the range

\[
\{ x_{mos} \} \leq z_{lms} \leq 1,
\]

in which the operations that require tool \( l \) determine the lower bound; and inequality (22), the upper bound.

Next, we consider \( D_{mos} \), the index set of the subset of constraints (18) in which a selected variable, \( x_{mos} \), has the coefficient 1. If \( x_{mos} \) is set to 1, each of the other variables with coefficient 1 in at least one of these constraints must be set to 0. Variables associated with subset \( D_{mos} \) include:
(i) $x_{mos}$

two types of variables incorporated in the first summation of (18)

(ii) $x_{m'os'}$, which assigns operation $o$ to alternative machine $m' \in M_o \setminus \{m\}$ at station $s$, and

(iii) $x_{mos'}$, which assigns operation $o$ to some machine $m' \in M_o$ at station $s'$ ($1 \leq s' \leq |S| - 1$, $s' \neq s$),

variables associated with each immediate predecessor $o' \in IP_o$ of operation $o$

(iv) $x_{m'o's'}$, which assigns predecessor $o' \in IP_o$ to some machine $m' \in M_o$ at a downstream station $s' > s$,

two types of variables that have coefficient 1 in rows where $x_{mos}$ is in the second summation

(v) $x_{m''os}$, which assigns immediate successor operation $o \in IP_o$ to some machine $m'' \in M_o$,

at an upstream station $s'' < s$, and

(vi) $x_{mos}$, which assigns operation $o$ to some machine $m' \in M_o$ at another station $2 \leq s' \leq |S|, s' \neq s$.

We now characterize extreme points associated with a selected subset $D_{mos}$.

**Claim 3** Each subsystem formed by fixing four types of inequalities to equalities

(a) the selected subset $D_{mos}$ of inequalities (18),

(b) all $\overline{z}$ inequalities (20) except for the one associated with $x_{mos}$,

(c) inequality (22) for all tools $l \in L_o$ used by combination $mos$, and

(d) for each $lms$ combination not included covered by (c), select an inequality of type (21), fixing $z_{lms} = 0,

    or of type (22), fixing $z_{lms} = 1$,

gives $\overline{\lambda}$ linearly independent, binding inequalities that define an extreme point, which is integral.

Claim 3 may be readily extended to deal with all agreeable assignments.

**Claim 4** Each subsystem formed by fixing four types of inequalities to equalities

(a) selected subsets $D_{mos}$ of inequalities (18) associated with $mos$ combinations in the agreeable assignment,

(b) all $\overline{z}$ inequalities (20) except for the ones associated with the $x_{mos}$ in the agreeable assignment,

(c) inequality (22) for all tools $l \in L_o$ used by combinations $mos$ in the agreeable assignment, and

(d) for each $lms$ combination not included covered by (c), select an inequality of type (21), fixing $z_{lms} = 0,

    or of type (22), fixing $z_{lms} = 1$,

gives $\overline{\lambda}$ linearly independent, binding inequalities that define an extreme point, which is integral.

We now extend Claim 4 to deal with *extended agreeable assignments* for which extreme points that represent feasible integer solutions allow an operation to be assigned to two stations. This possibility arises because the first summation in inequality (18) is for stations $1 \leq n \leq |S| - 1$. Thus, it is feasible for $x_{mos} = 1$.
even when \( x_{\text{mon}} = 1 \). Upstream assignment \( \text{mon} \) is limited only by assignments of predecessors \( o' \in \mathcal{I}P_o \).

Assignment \( \text{mo} \mid \mathcal{S} \) is limited only by the assignments of immediate successors \( o'' \) where \( o \in \mathcal{I}P_{o''} \).

**Claim 5** If each immediate predecessor of combination \( \text{mos} \), \( o' \in \mathcal{I}P_o \), is either unassigned or assigned agreeably (i.e., at station \( s' \) where \( s' < s \), given \( \text{mos} \)) and each successor \( o'' \) (i.e., where \( o \in \mathcal{I}P_{o''} \)) is unassigned or assigned to station \( |\mathcal{S}| \), then \( x_{\text{mos} \mid \mathcal{S}} \) may also be set to 1 for any \( m \in M_o \) in addition to \( x_{\text{mon}} = 1 \). This revises certain inequalities of two types, (b) and (d), that are fixed to equalities in Claim 4:

\[
(b') \text{ instead of inequality (20), use inequality (19) for each designated combination } \text{mo} \mid \mathcal{S} \text{ for any } m \in M_o
\]

\[(d') \text{ instead of inequality (21) or (22), use inequality (22) for all tools } l \in L_o \text{ used by combination mo} \mid \mathcal{S} \text{.}
\]

This gives \( \mathcal{N} \) linearly independent, binding inequalities that define an extreme point, which is integral.

Claims 6 and 7 and Corollary 1 show that \( \mathcal{N}^{(1)} \) has no fractional extreme point solution.

**Claim 6** If \( \mathcal{N}_k^f \) variables of type \( x_{\text{mos}} \) are set to fractional values \( 1 \leq \mathcal{N}_k^f \leq \mathcal{N}_s^f \), they cannot be associated with a set of \( \mathcal{N}_k^f \) linearly independent binding inequalities of type (18).

The implication of Claim 6 is that no set of \( \mathcal{N} \) linearly independent, binding inequalities intersect to form a fractional extreme point. Note that a fractional \( x_{\text{mos}} \) variable could be related to one of the \( |L_o| \) inequalities of type (19), but then one of the associated \( z_{\text{ins}} \) variables for \( l \in L_o \) could not be related to an independent, binding inequality because \( z_{\text{ins}} = x_{\text{mos}} \) for all \( l \in L_o \) so that \( z_{\text{ins}} \) could not be related to an inequality of type (21) or (22). Corollary 1 identifies the important result that combination \( os \) can be assigned to at most one machine \( m \in M_o \). It limits both integral and fractional solutions.

**Corollary 1** If \( 0 < x_{\text{mos}} \leq 1 \), \( x_{\text{ins}} \) for \( m' \in M_o \setminus \{m\} \) may be associated with a binding inequality that is linearly independent of others if and only if inequality (20) holds at equality, that is \( x_{m'\text{os}} = 0 \).

Now, consider values of \( z_{\text{ins}} \) that are not determined by fixing an inequality of type (19), (21), or (22).

**Claim 7** If variable \( z_{\text{ins}} \) is assigned a fractional value on \( z_{\text{ins}} = \max_{o \in \mathcal{O} \in L_o} \{x_{\text{mos}}\} < z_{\text{ins}} < 1 \), it cannot be associated with a binding inequality.

Because of the relationships imposed by inequalities (18), relatively few subsystems must be considered in the proof of Lemma 1. In addition, a related integral polytope is defined by Corollary 2.

**Corollary 2** Inequalities (19)-(22) and
\[
\sum_{s=1}^{n} \sum_{m \in M} x_{ms} + \sum_{s=n+1}^{||S||} \sum_{m \in M} x_{ms} \leq 1, \quad n = 1, \ldots, ||S||, \quad o \in O, \quad o' \in IP_o,
\]  

(23)

In which the second summation is vacuous when \( n = ||S|| \), define an integral polytope.

Inequality (23) allows \( n = ||S|| \) in inequalities (18), assuring that each operation will be assigned to at most one \( m_s \) combination, precluding the assignments described in Claim 5.

In \( MPASD_2 \), each operation requires one tool kit to be selected from a set of alternatives and each tool kit may be used by only one operation. The polytope that corresponds to \( \mathcal{R}^{(1)} \) replaces (19) with (9).

**Lemma 2** The polytope \( \mathcal{R}^{(2)} = \{ (x^T, z^T) : (18), (9), (20), (21), and (22) \} \) is integral.

The proof of Lemma 2 relies on Claim 8, which is the \( MPASD_2 \) version of Claim 5 for extended agreeable assignments. Otherwise, the proof follows that of Lemma 1 so that details are not presented.

**Claim 8** Each subsystem formed by fixing six types of inequalities to equalities

(a) selected subsets \( D_{mos} \) of inequalities (18) for \( mos \) combinations in the extended agreeable assignment,

(b') inequality (9) for each designated combination \( mos \) \( |S| \) for any \( m \in M_o \),

(b) all \( \bar{\Pi}_z \) inequalities (20) except for the ones specified in (a) and (b'),

(c) inequality (9) for one selected tool \( l \in L_o \) used by each \( mos \) combination in (a) and (b'),

(c') inequality (21) for each tool required by each \( mos \) combination in (a) and (b') that is not selected in (c):

\( l' \in L_o \setminus \{l\} \) , and

(d) for each \( lms \) combination not covered by (c) and (c'), select an inequality of type (21), fixing \( z_{lms} = 0 \),

or of type (22), fixing \( z_{lms} = 1 \),

gives \( \bar{\Pi} \) linearly independent, binding inequalities that define an extreme point, which is integral.

Finally, an integral polytope is also defined if inequalities (18) are tightened as described in Corollary 2. Again, details of the proof are omitted.

**Corollary 3** Inequalities (9) and (20)-(23) define an integral polytope.

### 3.2.2.2 Generating Cuts

Now, fix a \( m_0 \in M \) and \( s_0 \in S \) and assume that a valid inequality for \( MPASD_3 \) is given by

\[
\sum_{l \in L} \alpha_i z_{lm_0s_0} \leq b y_{ms_0},
\]
where \( \tilde{L} \subseteq L \) such that, for each \( o \in O_m \), \( \tilde{L} \cap L_o \neq \emptyset \). In an optimal solution, no station can be activated without assigning at least one operation to it. Thus,

\[
\tilde{\alpha} y_{m, o_0} \leq \sum_{i \in \tilde{L}} \alpha_i z_{m, i, o_0} \tag{24}
\]

is also a valid inequality, in which \( \tilde{\alpha} = \min \left\{ \sum_{i \in \tilde{L}} \alpha_i : o \in O_m \right\} \).

Using Lemma 1 for \( MPASD_1 \) (or Lemma 2 for \( MPASD_2 \)), we can improve (24) by lifting with respect to variables \( x_{m, o_0} \), \( o \in O \) over embedded polytope \( \mathcal{R}^{(1)} \) (\( \mathcal{R}^{(2)} \)). So, assume that

\[
\tilde{\alpha} y_{m, o_0} \leq \sum_{i \in \tilde{L}} \alpha_i z_{m, i, o_0} - \sum_{o \in \tilde{O}} \beta_o x_{m, o_0} \tag{25}
\]

is an inequality obtained from (24) by lifting with respect to variables \( x_{m, o_0} \), \( o \in O \), where \( \tilde{O} \) is a subset of \( O \), and \( \beta_o > 0 \) for all \( o \in \tilde{O} \). Moreover, let \( o \in O_m \setminus \tilde{O} \) and let the lifting problem associated with variable \( x_{m, o_0} \) be defined as

\[
\beta_o^* = \min \left\{ \sum_{i \in \tilde{L}} \alpha_i z_{m, i, o_0} - \sum_{o \in \tilde{O}} \beta_o x_{m, o_0} : x \in \mathcal{R}^{(1)} \right\}.
\]

The LP solution to this problem prescribes binary values for \( x \) decision variables and the lifting coefficient \( \beta_o^* \). If \( \beta_o^* > \tilde{\alpha} \), another inequality that is valid for \( MPASD_1 \) is given by

\[
\tilde{\alpha} y_{m, o_0} \leq \sum_{i \in \tilde{L}} \alpha_i z_{m, i, o_0} - \sum_{o \in \tilde{O}} \beta_o x_{m, o_0} - (\beta_o^* - \tilde{\alpha}) x_{m, o_0} \tag{26}
\]

since the integral polytope associated with inequalities (2)-(5) is a subset of \( \mathcal{R}^{(1)} \). A similar argument holds for \( MPASD_2 \), and we omit details.

Lifted inequalities (25) and (26) translate precedence relationships for operations to tools. The following example demonstrates the cut-generating procedure described in this section (3.2.2).

**Example** For simplicity, assume there is only one machine, \( L = \{l_1, l_2, l_3\} \), \( O = \{o_1, o_2, o_3\} \), and \( o_2 \) must be performed after \( o_1 \) and \( o_3 \), after \( o_2 \). Let \( s_o = 1 \) and assume that inequality (5) is
\[
2 z_{t_{h_0}} + 3 z_{l_{z_{t_0}}} + 5 z_{l_{x_{t_0}}} \leq 7 y_{t_{0}}.
\]

In this case, (24) becomes

\[
2 y_{t_{0}} \leq 2 z_{t_{h_0}} + 3 z_{l_{z_{t_0}}} + 5 z_{l_{x_{t_0}}}.
\]

Now, one may lift with respect to \( x_{o_{2_{t_0}}} \) in the following manner:

\[
\min\{2 z_{t_{h_0}} + 3 z_{l_{z_{t_0}}} + 5 z_{l_{x_{t_0}}} : x_{o_{2_{t_0}}} = 1, x \in \{1, 2, \ldots, n\}\} = 5
\]

since \( x_{o_{2_{t_0}}} = 1 \) implies \( x_{o_{2_{t_0}}} = 1 \) (so \( z_{t_{h_0}} = z_{l_{x_{t_0}}} = 1 \) from (4)). Therefore,

\[
2 y_{t_{0}} \leq 2 z_{t_{h_0}} + 3 z_{l_{z_{t_0}}} + 5 z_{l_{x_{t_0}}} - 3 x_{o_{2_{t_0}}}
\]

is also a valid inequality for \( MPASD_1 \). We now describe our implementation strategy.

### 3.3 Implementation Strategy

We apply our heuristic and preprocessing methods at the root node. The heuristic determines an upper bound, \( Z_{BPH} \), on the value of the optimal solution. In turn, our preprocessing methods use \( Z_{BPH} \) to determine an upper bound on the number of stations, \( N_{opt} \), as well as the earliest (\( ES_o \)) and latest (\( LS_o \)) stations to which operation \( o \in O \) may be assigned. These bounds limit the size of each instance.

We solved \( MPASD_1 \) (\( MPASD_2 \)) to obtain the optimal binary solution, \( Z^*_{BP} \), using the branch-and-cut capabilities of OSL. To implement branch and cut, OSL uses its supernode routine to manage cuts generated by user-supplied routines. OSL’s supernode routine (IBM 1995) invokes pre-processing methods (e.g., bound tightening and coefficient reduction using probing techniques) at nodes of a B&B search tree to strengthen the linear relaxation. It is called “supernode” because it may fix several variables while analyzing one node in the search tree; in this way it is comparable to the analysis done at a number of nodes in the traditional search tree. It implements branch and cut, adding certain inequalities, for example, derived from the implication lists that result from probing. It also calls a routine through which the user may add generated cuts. Supernode manages cuts; for example, it may discard an
inequality if it becomes nonbinding. Supernode also employs a heuristic that attempts to determine a feasible solution, and it may eliminate cuts after invoking its heuristic.

Our routines apply the FGP to individual knapsacks defined by inequalities (3), generating inequalities of type (14) using separation Problem (SC), and apply Lemma 1 for $MPASD_1$ (Lemma 2 for $MPASD_2$) to lift inequality (24). We applied the solutions prescribed by the FGP in several ways. We used FGP solutions to generate cuts associated with inequalities (3) directly. In addition, we used FGP solutions in a new way, lifting the vectors that comprise an optimal basis for Problem P to obtain cover inequalities (12) and (14), which were subsequently strengthened by our new lifting methods.

4. Computational Evaluation

To establish computational benchmarks, we ran a set of tests on an IBM RISC 6000 model 550 using OSL release 3. We are aware of no other special-purpose algorithms to solve $MPASD_1$ and $MPASD_2$, so we compared our approach with OSL using its capable supernode routine to implement B&B. Thus, the only difference between the two approaches is that one employs our cut-generating methods. This section describes the set of random test instances we generated as well as test results.

4.1 Test Instances

We designed an experiment with four factors to gain insight into the influence they have on the run time to solve $MPASD_1$ and $MPASD_2$. Resource limitations did not allow us to replicate each treatment; however, experiments of this design are commonly used to evaluate mathematical programming algorithms. The factors we considered and the levels for each were:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) machine availability</td>
<td>4000 6000</td>
</tr>
<tr>
<td>(2) precedence relationships</td>
<td>(4, 7) (3, 5)</td>
</tr>
<tr>
<td>(3) number of alternative machine types</td>
<td>2 4 6</td>
</tr>
<tr>
<td>(4) number of products</td>
<td>3 4</td>
</tr>
</tbody>
</table>
We defined the levels for factor (1) by $A_m = c_i / r_m$ for $i = 1, 2$ for each $m \in M$. We specified $c_1 = 4000$ and $c_2 = 6000$ and drew $r_m$ randomly from $U[0.5, 1.0]$. The levels for factor (3) were 2, 4, and 6, and those for factor (4) were 3 and 4.

The levels for factor (2) were contingent on factor (4). Instances that involved 3 products employed precedence digraphs with either 4 (Figure 1) or 7 (Figure 3) rows of nodes. Instances that involved 4 products employed precedence relationships with either 3 (Figure 2) or 5 (Figure 4) rows of nodes. We used the precedence digraphs in Figures 1-4 for both $MPASD_1$ and $MPASD_2$ instances. Each node in these figures represents an operation, an arc represents an immediate precedence relationship, and the number(s) within each node indicate the product(s) that require the associated operation. We designed digraphs 1 and 2 (3 and 4) so that about 50% (75%) of all possible arrows from one layer to the next were generated, and about 50% (75%) of the operations are common to two or more products. Precedence relationships are relatively dense, contributing to the difficulty of the instances.

We used the same factors in testing model $MPASD_2$, generating a set of 24 instances of $MPASD_1$ and another 24 of $MPASD_2$. We generated tooling constraints in $MPASD_1$ so that 30%,
Figure 2: Precedence Diagram 2 for MPASD Problems

Figure 3: Precedence Diagram 3 for MPASD Problems
60%, and 10% of the operations require 1, 2, and 3 tools, respectively, and 60%, 30%, 6%, and 4% of the tools were required by 1, 2, 3, and 4 operations, respectively. Generated tool-kit constraints in $MPASD_2$ assigned a unique tool kit to each of 10% of the operations and allowed 50%, 30%, and 10% of the operations to select from 1 of 2, 1 of 3, or 1 of 4 tool kits, respectively.

Tables 1 and 2 describe our test problems. Each row describes an instance. The first column specifies the model and level of each factor associated with an instance. Notation is self-explanatory, for example, mpasd/c1d4m2p3 represents an $MPASD_1$ instance with $c_1 = 4000$, four rows in the precedence digraph, two types of alternative machines, and three products. Other columns give the number of constraints; number of variables; optimal values for the initial LP relaxation, $Z_{LP}^*$; optimal binary solution, $Z_{BP}^*$; and percent gap, $\%\text{Gap} = 100 \left( \frac{Z_{BP}^* - Z_{LP}^*}{Z_{BP}^*} \right)$. We obtained $Z_{BP}^*$, the optimal binary solution to each $MPASD_1$ ($MPASD_2$) instance using the strategy described in Section 3.3. The

![Figure 4: Precedence Diagram 4 for MPASD Problems](image)
average %Gap for the 24 $MPASD_1$ instances was 13.75%, and that for the 24 $MPASD_2$ instances was 9.30%. We now describe test results.

### 4.2 Test Results

Tables 3 and 4 summarize test results. Each row describes an instance. The first column specifies the model and level of each factor associated with an instance. Columns 2-4 describe the performance of our approach, giving, respectively, the number of cuts generated during the branch-and-cut tree search, the number of B&B nodes required to find and verify an optimal solution, and the CPU run time (in seconds) for solving the instance. We adopted CPU run time as the primary measure of our approach and the number of nodes required to find and confirm an optimal solution as a secondary measure. Columns 5-6 describe the performance of OSL, giving, respectively, the number of B&B nodes to find and verify the optimal solution and the CPU run time (in seconds). We set a run-time limit of five hours for each instance. Our heuristic and pre-processing methods serve only to manage the sizes of instances, so we document their effects only in columns 2 and 3 of Tables 1 and 2. On average, our heuristic prescribed a solution that was 1 to 2 stations higher than the optimal number of stations. However, even in the few instances in which the heuristic prescribed the optimal number of stations, both our branch-and-cut approach and OSL took some time to find and verify an optimal solution.

Our approach was able to solve 46 of these 48 instances within run-time and storage limitations, but OSL was able to solve only 32 instances. On these 32 instances, our approach required, on average, significantly less run time and significantly fewer nodes in the search tree.
It is difficult to generalize the effect of each factor on run time. It appears that increasing machine availability tended to make instances somewhat easier to solve. Increasing the number of alternative machines for each operation from 2 to 4 to 6 tended to increase the number of assignment combinations and the level of difficulty. As expected, results showed that run time increased with the number of operations. Even though model $MPASD_2$ is somewhat tighter than $MPASD_1$, most larger instances (i.e., those with 5 or 7 layers in their precedence relationships) remained challenging, requiring, on average, more than one hour to obtain an optimal solution.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Rows</th>
<th>Variables</th>
<th>$Z_{LP}^*$</th>
<th>$Z_{BP}^*$</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mpasd/c1 4m2p3</td>
<td>441</td>
<td>423</td>
<td>358755.800</td>
<td>448299.311</td>
<td>19.97</td>
</tr>
<tr>
<td>Mpasd/c1 3m2p4</td>
<td>415</td>
<td>396</td>
<td>402261.900</td>
<td>435740.790</td>
<td>7.68</td>
</tr>
<tr>
<td>Mpasd/c1 4m4p3</td>
<td>738</td>
<td>738</td>
<td>355497.100</td>
<td>438847.722</td>
<td>18.99</td>
</tr>
<tr>
<td>Mpasd/c1 3m4p4</td>
<td>684</td>
<td>702</td>
<td>361957.800</td>
<td>428264.502</td>
<td>15.48</td>
</tr>
<tr>
<td>Mpasd/c1 4m6p3</td>
<td>1035</td>
<td>1080</td>
<td>318314.300</td>
<td>436693.404</td>
<td>27.11</td>
</tr>
<tr>
<td>Mpasd/c1 3m6p4</td>
<td>1000</td>
<td>1026</td>
<td>335774.800</td>
<td>402742.255</td>
<td>16.63</td>
</tr>
<tr>
<td>Mpasd/c2 4m2p3</td>
<td>459</td>
<td>405</td>
<td>358761.600</td>
<td>417007.557</td>
<td>13.97</td>
</tr>
<tr>
<td>Mpasd/c2 3m2p4</td>
<td>433</td>
<td>405</td>
<td>347605.600</td>
<td>426133.741</td>
<td>18.43</td>
</tr>
<tr>
<td>Mpasd/c2 4m4p3</td>
<td>729</td>
<td>720</td>
<td>364156.400</td>
<td>414689.917</td>
<td>12.19</td>
</tr>
<tr>
<td>Mpasd/c2 3m4p4</td>
<td>667</td>
<td>684</td>
<td>310854.300</td>
<td>409076.253</td>
<td>24.01</td>
</tr>
<tr>
<td>Mpasd/c2 4m6p3</td>
<td>1026</td>
<td>1098</td>
<td>321218.300</td>
<td>395627.739</td>
<td>18.81</td>
</tr>
<tr>
<td>Mpasd/c2 3m6p4</td>
<td>1000</td>
<td>1089</td>
<td>325703.600</td>
<td>394885.599</td>
<td>17.52</td>
</tr>
<tr>
<td>Mpasd/c1 7m2p3</td>
<td>838</td>
<td>975</td>
<td>729758.700</td>
<td>832577.857</td>
<td>12.35</td>
</tr>
<tr>
<td>Mpasd/c1 5m2p4</td>
<td>775</td>
<td>1005</td>
<td>732501.200</td>
<td>799873.179</td>
<td>8.42</td>
</tr>
<tr>
<td>Mpasd/c1 7m4p3</td>
<td>1468</td>
<td>1845</td>
<td>751194.400</td>
<td>824941.650</td>
<td>8.94</td>
</tr>
<tr>
<td>Mpasd/c1 5m4p4</td>
<td>1360</td>
<td>1800</td>
<td>698182.000</td>
<td>781576.445</td>
<td>10.67</td>
</tr>
<tr>
<td>Mpasd/c1 7m6p3</td>
<td>1788</td>
<td>2400</td>
<td>728098.000</td>
<td>786504.736</td>
<td>7.43</td>
</tr>
<tr>
<td>Mpasd/c1 5m6p4</td>
<td>2185</td>
<td>2820</td>
<td>650681.200</td>
<td>792423.100</td>
<td>17.89</td>
</tr>
<tr>
<td>Mpasd/c2 7m2p3</td>
<td>928</td>
<td>1155</td>
<td>770942.400</td>
<td>824838.760</td>
<td>6.53</td>
</tr>
<tr>
<td>Mpasd/c2 5m2p4</td>
<td>805</td>
<td>1020</td>
<td>729075.100</td>
<td>762525.201</td>
<td>4.39</td>
</tr>
<tr>
<td>Mpasd/c2 7m4p3</td>
<td>1528</td>
<td>1965</td>
<td>742920.200</td>
<td>813259.131</td>
<td>8.65</td>
</tr>
<tr>
<td>Mpasd/c2 5m4p4</td>
<td>1585</td>
<td>2085</td>
<td>654187.500</td>
<td>739959.028</td>
<td>11.59</td>
</tr>
<tr>
<td>Mpasd/c2 7m6p3</td>
<td>1888</td>
<td>2535</td>
<td>700416.500</td>
<td>799716.174</td>
<td>12.42</td>
</tr>
<tr>
<td>Mpasd/c2 5m6p4</td>
<td>2065</td>
<td>2715</td>
<td>626691.400</td>
<td>696597.748</td>
<td>10.04</td>
</tr>
</tbody>
</table>

*Not optimal*
### Table 2: MPASD\textsubscript{2} Test Instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>Rows</th>
<th>Variables</th>
<th>$Z_{LP}^*$</th>
<th>$Z_{BP}^*$</th>
<th>% Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpsd\textsubscript{2}/c1 \ell 4\textsubscript{m}2p3</td>
<td>244</td>
<td>603</td>
<td>422046.300</td>
<td>485713.914</td>
<td>13.11</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c1 \ell 4\textsubscript{m}4p4</td>
<td>378</td>
<td>999</td>
<td>415798.900</td>
<td>473070.047</td>
<td>12.11</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c1 \ell 4\textsubscript{m}6p3</td>
<td>540</td>
<td>1485</td>
<td>427301.600</td>
<td>452714.600</td>
<td>5.61</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c1 \ell 4\textsubscript{m}6p4</td>
<td>559</td>
<td>1611</td>
<td>382274.100</td>
<td>442762.028</td>
<td>13.66</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 4\textsubscript{m}2p3</td>
<td>243</td>
<td>612</td>
<td>408345.800</td>
<td>458761.237</td>
<td>10.99</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 4\textsubscript{m}4p4</td>
<td>396</td>
<td>1143</td>
<td>405385.600</td>
<td>435463.744</td>
<td>7.09</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 4\textsubscript{m}6p3</td>
<td>540</td>
<td>1260</td>
<td>704812.700</td>
<td>790281.321</td>
<td>10.81</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 4\textsubscript{m}6p4</td>
<td>550</td>
<td>1156</td>
<td>399938.600</td>
<td>441549.025</td>
<td>9.42</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c1 \ell 7\textsubscript{m}2p3</td>
<td>613</td>
<td>1405</td>
<td>749766.500</td>
<td>824602.733</td>
<td>9.08</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c1 \ell 7\textsubscript{m}4p3</td>
<td>928</td>
<td>2475</td>
<td>758145.000</td>
<td>823767.762</td>
<td>7.97</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c1 \ell 7\textsubscript{m}6p3</td>
<td>1184</td>
<td>3285</td>
<td>742821.500</td>
<td>821238.000</td>
<td>9.37</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c1 \ell 7\textsubscript{m}6p4</td>
<td>1135</td>
<td>2940</td>
<td>691959.800</td>
<td>759719.700</td>
<td>8.92</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 7\textsubscript{m}2p3</td>
<td>552</td>
<td>1470</td>
<td>710704.700</td>
<td>819446.214</td>
<td>13.27</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 7\textsubscript{m}4p3</td>
<td>565</td>
<td>1380</td>
<td>727657.500</td>
<td>802395.500</td>
<td>9.31</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 7\textsubscript{m}6p3</td>
<td>987</td>
<td>2580</td>
<td>701197.300</td>
<td>775012.310</td>
<td>9.52</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 7\textsubscript{m}6p4</td>
<td>850</td>
<td>2130</td>
<td>683543.400</td>
<td>797080.400</td>
<td>14.24</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 7\textsubscript{m}6p3</td>
<td>1228</td>
<td>3375</td>
<td>697886.700</td>
<td>791351.341</td>
<td>11.81</td>
</tr>
<tr>
<td>mpsd\textsubscript{2}/c2 \ell 5\textsubscript{m}6p4</td>
<td>1150</td>
<td>3060</td>
<td>693993.100</td>
<td>714803.445</td>
<td>2.91</td>
</tr>
</tbody>
</table>

*Not optimal*

Our test instances were not large relative to parameters, $|L|$, $|M|$, $|O|$, and $|S|$. However, MPASD is challenging because precedence relationships, alternative machines, machine capacities, tooling requirements, and tooling-space limitations interact combinatorially to increase difficulty. As the theory of computational complexity would suggest, run time was related to problem size as measured by the number of decision variables. Instances with fewer than 700 variables tended to be relatively easy to solve. However, OSL (without our new methods) could not solve most problems with over 2000 variables. A few relatively small instances were particularly challenging, while a few of the largest
instances were solved easily. The two problems that our new methods did not solve within the run-time limit were the largest $MPASD_1$ instance with 2820 variables and the largest $MPASD_2$ instance with 3285 variables.

Table 3: $MPASD_1$ Test Results

<table>
<thead>
<tr>
<th>Instance</th>
<th>Branch and Cut</th>
<th>OSL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cuts</td>
<td>Nodes</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 4m2p3$</td>
<td>340</td>
<td>12</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 3m2p4$</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 4m4p3$</td>
<td>538</td>
<td>25</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 3m4p4$</td>
<td>297</td>
<td>7</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 4m6p3$</td>
<td>2513</td>
<td>54</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 3m6p4$</td>
<td>2905</td>
<td>53</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 4m2p3$</td>
<td>74</td>
<td>0</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 3m2p4$</td>
<td>125</td>
<td>7</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 4m4p3$</td>
<td>72</td>
<td>3</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 3m4p4$</td>
<td>430</td>
<td>18</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 4m6p3$</td>
<td>1286</td>
<td>52</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 3m6p4$</td>
<td>1116</td>
<td>30</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 7m2p3$</td>
<td>4922</td>
<td>63</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 5m2p4$</td>
<td>355</td>
<td>7</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 7m4p3$</td>
<td>3780</td>
<td>64</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 5m4p4$</td>
<td>4078</td>
<td>155</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 7m6p3$</td>
<td>1165</td>
<td>27</td>
</tr>
<tr>
<td>$Mpasd_1/c1 \ell 5m6p4$</td>
<td>3832</td>
<td>176</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 7m2p3$</td>
<td>147</td>
<td>0</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 5m2p4$</td>
<td>201</td>
<td>0</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 7m4p3$</td>
<td>1300</td>
<td>33</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 5m4p4$</td>
<td>1057</td>
<td>36</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 7m6p3$</td>
<td>4143</td>
<td>81</td>
</tr>
<tr>
<td>$Mpasd_1/c2 \ell 5m6p4$</td>
<td>11051</td>
<td>456</td>
</tr>
</tbody>
</table>

* All times are in seconds
** Time or Memory over limit
Table 4: $MPASD_2$ Test Results

<table>
<thead>
<tr>
<th>Instance</th>
<th>Branch and Cut</th>
<th>OSL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cuts</td>
<td>Nodes</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>Mpasd/c1 l4m2p3</td>
<td>582</td>
<td>19</td>
</tr>
<tr>
<td>Mpasd/c1 l3m2p4</td>
<td>54</td>
<td>0</td>
</tr>
<tr>
<td>Mpasd/c1 l4m4p3</td>
<td>717</td>
<td>52</td>
</tr>
<tr>
<td>Mpasd/c1 l3m4p4</td>
<td>2249</td>
<td>60</td>
</tr>
<tr>
<td>Mpasd/c1 l4m6p3</td>
<td>385</td>
<td>19</td>
</tr>
<tr>
<td>Mpasd/c1 l3m6p4</td>
<td>6024</td>
<td>421</td>
</tr>
<tr>
<td>Mpasd/c2 l4m2p3</td>
<td>144</td>
<td>2</td>
</tr>
<tr>
<td>Mpasd/c2 l3m2p4</td>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>Mpasd/c2 l4m4p3</td>
<td>1244</td>
<td>104</td>
</tr>
<tr>
<td>Mpasd/c2 l3m4p4</td>
<td>629</td>
<td>40</td>
</tr>
<tr>
<td>Mpasd/c2 l4m6p3</td>
<td>1042</td>
<td>111</td>
</tr>
<tr>
<td>Mpasd/c2 l3m6p4</td>
<td>229</td>
<td>14</td>
</tr>
<tr>
<td>Mpasd/c1 l7m2p3</td>
<td>866</td>
<td>21</td>
</tr>
<tr>
<td>Mpasd/c1 l5m2p4</td>
<td>1540</td>
<td>8</td>
</tr>
<tr>
<td>Mpasd/c1 l7m4p3</td>
<td>5344</td>
<td>159</td>
</tr>
<tr>
<td>Mpasd/c1 l5m4p4</td>
<td>505</td>
<td>6</td>
</tr>
<tr>
<td>Mpasd/c1 l7m6p3</td>
<td>6150</td>
<td>137</td>
</tr>
<tr>
<td>Mpasd/c1 l5m6p4</td>
<td>571</td>
<td>12</td>
</tr>
<tr>
<td>Mpasd/c2 l7m2p3</td>
<td>292</td>
<td>0</td>
</tr>
<tr>
<td>Mpasd/c2 l5m2p4</td>
<td>347</td>
<td>0</td>
</tr>
<tr>
<td>Mpasd/c2 l7m4p3</td>
<td>16566</td>
<td>400</td>
</tr>
<tr>
<td>Mpasd/c2 l5m4p4</td>
<td>7520</td>
<td>192</td>
</tr>
<tr>
<td>Mpasd/c2 l7m6p3</td>
<td>3185</td>
<td>36</td>
</tr>
<tr>
<td>Mpasd/c2 l5m6p4</td>
<td>1304</td>
<td>8</td>
</tr>
</tbody>
</table>

* All times are in seconds  
** Time or Memory over limit

5. Conclusions

This paper presents two new models to deal with different tooling requirements in the generic multiproduct, assembly-system design (MPASD) problem; describes a branch-and-cut solution approach based on several new families of inequalities, and establishes computational benchmarks. Our solution approach can be applied, with minor modifications, to both models. Our approach includes new heuristic and
preprocessing methods that manage the size of each instance. It employs the facet generation procedure (FGP) to generate facets of underlying knapsack polytopes. In addition, it uses the FGP in a new way to generate additional cuts and incorporates two new methods that exploit special structures of the MPASD problem to generate cuts. One of our new methods is based on a principle that exploits embedded, integral polytopes and can be applied to solve generic 0-1 programs.

This paper establishes computational benchmarks for the MPASD problem by describing an experiment that involved 24 test instances of each of the two MPASD models. Test results show that our method performed particularly well in comparison to OSL; it had better run times on three instances, required fewer nodes in one instance, and posted both better run times and fewer nodes in 40 of the 48 instances. OSL performed somewhat better on two instances (mpasd1/c2 \ell 4m2p3 and mpa1/c2 \ell 3m2p4), which were relatively easy for both approaches, and neither approach was able to solve the two largest instances (mpasd1/c1 \ell 5m6p4 and mpa1/c1 \ell 7m6p3) within the run-time limit.

Increasing levels of factors (1) machine availability and (3) number of alternative machine types for each operation did not affect run time in a consistent way. However, increasing levels of factors (2), precedence relationships, and (4), number of products, increased the number of operations and resulted in increased run times.

Results indicate that the MPASD problem is amenable to our new cutting-plane methods, even though tooling considerations entail an additional level of combinatorics, making our MPASD problems more challenging than the ASD problems studied by Pinnoi and Wilhelm (1997b, 1998) and Gadidov and Wilhelm (2000). It would be possible to combine our new cut-generating methods with those devised by Pinnoi and Wilhelm (1998), which rely upon relationships between the node-packing and assembly-line balancing polytopes, but focusing on our new cut-generation methods allowed us to assess their efficacy. Test results show that the new methods presented in this paper collectively form a credible solution approach. Future research may, however, investigate the relative efficacy of each applicable family of inequalities, including those based on ASD (Pinnoi and Wilhelm 1998) and MPASD (as described in
this paper) as well as those related to more generic problems such as the precedence-constrained knapsack, multiple knapsack, and generalized assignment problem.

Appendix

This Appendix presents proofs of Claims, Corollaries, and Lemmas.

Proof of Claim 1:
(a) \( \dim(\mathcal{R}(1)) \leq \overline{n} \) and \( \dim(\text{conv}(\mathcal{Q})) \leq \overline{n} \) since both incorporate \( \overline{n} \) decision variables.
(b) \( \dim(\mathcal{R}(1)) \geq \overline{n} \) and \( \dim(\text{conv}(\mathcal{Q})) \geq \overline{n} \) since each polytope contains \( \overline{n} + 1 \) affinely independent points:
   (i) one point: \((x^T, z^T) = (\theta^T, \theta')\)
   (ii) \(\overline{n}_z\) points: \((x^T, z^T) = (\overline{e}_{lms}^T)\), one for each of the \(\overline{n}_z\) lms combinations
   (iii) \(\overline{n}_s\) points: \((x^T, z^T) = \left(\overline{e}_{mos}^T - \sum_{l \in L} \overline{e}_{lms}^T\right)\), one for each of \(\overline{n}_s\) mos combinations.
(c) Combining (a) and (b), \( \dim(\mathcal{R}(1)) = \overline{n} \) and \( \dim(\text{conv}(\mathcal{Q})) = \overline{n} \). Q.E.D.

Proof of Claim 2. A basis matrix representing an extreme point of Claim 2 may be composed of rows
(a) \(-e_{mos}^T\) for all mos \((\overline{n}_x\) binding inequalities (20))
(b) \(-e_{lms}^T\) or \(e_{lms}^T\) for each lms \((\overline{n}_z\) binding inequalities of type (21) or (22))
representing \(\overline{n}\) linearly independent, binding inequalities. Each of these extreme points is integral. Q.E.D.

Proof of Claim 3. A basis matrix representing an extreme point of Claim 3 may be composed of rows
(a) \(e_{mos}^T\) for combination mos, upon which subset \(D_{mos}\) is defined
(b) \(-e_{m'os'}^T\) for each combination \(m'o's'\) other than mos
(c) \(e_{lms}^T\) for each \(l \in L_o\) required by combination mos
(d) \(-e_{l'm'o's'}^T\) or \(e_{l'm'o's'}^T\) for each \(l' \in L_o\) associated with combinations \(m'o's'\) other than mos
representing \(\overline{n}\) linearly independent, binding inequalities. Each of these extreme points is integral. Elementary row operations (EROs) can be performed on each row in subset \(D_{mos}\), adding (or subtracting) the appropriate row in (b)-(d) to reduce that row to \(e_{mos}^T\). Any one of these reduced rows may be used in (a); the others are implied by it. Q.E.D.

Proof of Claim 4. A basis matrix representing an extreme point of Claim 4 may be composed of rows
(a) \(-e_{mos}^T\) for each combination mos associated with the agreeable assignment
(b) \(-e_{m'o's'}^T\) for each combination \(m'o's'\) other than combinations mos associated with the agreeable assignment
(c) \(e_{lms}^T\) for each \(l \in L_o\) required by an mos combination associated with the agreeable assignment
(d) \(-e_{l'm'o's'}^T\) or \(e_{l'm'o's'}^T\) for each \(l' \in L_o\) associated with combinations \(m'o's'\) other than mos combinations associated with the agreeable assignment representing \(\overline{n}\) linearly independent, binding inequalities. Each of these extreme points is integral.
Subsets $D_{mos}$ for each pair of combinations in the agreeable assignment have no rows or columns in common. Elementary row operations (EROs) may thus be performed on each row in each subset $D_{mos}$, adding (or subtracting) the appropriate rows in (b)-(d) to reduce that row to $e^T_{mos}$ as in the proof to Claim 3. Q.E.D.

Proof of Claim 5. A basis matrix representing an extreme point of Claim 5 may be composed of the rows described in the proof of Claim 4 but modified by using

(b') $e^T_{mos} - e^T_{mos}$ for each designated combination $mo | S |$ for any $m \in M_o$ and $l \in L_o$

(d') $e^T_{mos}$ for each $l \in L_o$ associated with combination $mo | S |$.

The revised set represents $\pi$ linearly independent, binding inequalities. Each of these extreme points is integral. Q.E.D.

Proof of Claim 6. If $\pi^f_x = 1$, a single fractional $x_{mos}$ cannot make any inequality (18) hold at equality because each has RHS equal to 1. Now, suppose that $\pi^f_x$ variables of type $x_{mos}$ are fractional and they appear in $\pi^f_{18}$ inequalities of type (18), $\pi^f_{18}$ of which hold at equality. Clearly, if $\pi^f_x < \pi^a_{18}$, the rows are linearly dependent. If $\pi^f_x > \pi^a_{18} \geq \pi^e_{18}$ or $\pi^f_x = \pi^a_{18} = \pi^e_{18}$, $(\pi^f_x - \pi^a_{18}) \geq 1$ so that the $\pi^f_x$ variables cannot be associated with $\pi^f_x$ binding inequalities (18). Now, consider three cases in which $\pi^f_x = \pi^a_{18} = \pi^e_{18}$.

(Case 1): If the same $\pi^f_x$ variables appear in all $\pi^e_{18} = \pi^a_{18}$ binding inequalities, EROs can be used to show that these rows are linearly dependent. Thus, in case 1, the $\pi^f_x$ variables cannot be associated with $\pi^f_x$ linearly independent, binding inequalities. (Case 2): If the row of (18) in which $x_{mos}$ first appears contains $x_{mos}$ in the first summation, successive inequalities add variables that assign operation $o$ downstream of station $s$ (i.e., at station $s'$ where $s' > s$) as $n$ increases and/or different predecessors (i.e., $o' \in IP_o$) downstream of station $s$. (Case 3): If the row of (18) in which $x_{mos}$ first appears contains $x_{mos}$ in the second summation, successive inequalities add variables that assign successors at other stations upstream of station $s$. In Case 2 or 3, the row in which $x_{mos}$ first appears must contain fewer than $\pi^f_x$ fractional variables that cannot add to 1. This is true because each of the other rows contains at least one additional fractional variable and the fractions must add to 1 in each of those rows. If not, the row in which $x_{mos}$ first appears must contain the same set of variables that appear in each of the $\pi^e_{18} = \pi^a_{18}$ rows and this set of rows is not linearly independent). Q.E.D.

Proof of Corollary 1. ($\Rightarrow$) A basis matrix representing an extreme point of Corollary 1 may be composed using the rows of Claim 5, modified by

(a'') $-e^T_{mos}$ for each combination $mos$ where it is desired that $x_{mos}$ be on the range $0 < x_{mos} \leq 1$

(b'') $-e^T_{mos}$ for each combination $m'os$ where $m' \in M_o \setminus \{m\}$ and it is desired that $x_{mos} = 0$

The revised subsystem represents $\pi$ linearly independent, binding inequalities. Each of these extreme points is integral.
(⇐) Given \( 0 < x_{\text{max}} \leq 1 \), suppose \( \alpha < x_{\text{max}} \) (for \( m' \in M_o \setminus \{m\} \)). Claim 6 establishes that both \( x_{\text{max}} \) and \( x_{\text{max}}' \) cannot be fractional and inequalities (18) do not allow both to equal 1. Therefore, \( x_{\text{max}}' = 0 \) must hold. \( Q.E.D. \)

Proof of Claim 7. \( z_{\text{ins}} \) may only be associated with binding inequalities (19) \( x_{\text{max}} = z_{\text{ins}} \) (for \( \alpha \in \{l \in L_o \} \)) (21) \( z_{\text{ins}} = 0 \), or (22) \( z_{\text{ins}} = 1 \) so that, if it is fractional and \( z_{\text{ins}}' < z_{\text{ins}} \), it cannot be associated with a binding inequality. Furthermore, no set of \( \overline{n} \) fractional variables because each constraint of type (19), (21) or (22) includes only one \( z_{\text{ins}} \) variable. \( Q.E.D. \)

Proof of Lemma 1. Claims 1-5 identify subsystems of \( \overline{n} \) inequalities to hold at equality to represent feasible extreme points, each of which is integral. Claims 6 (along with Corollary 1) and 7 show that it is not possible to identify a subsystem of \( \overline{n} \) inequalities to hold at equality to represent a fractional solution, so no extreme point is fractional. These claims enumerate all feasible subsystem selections, so that \( \mathcal{R}^{(1)} \) is integral (i.e., \( \mathcal{R}^{(1)} = \text{conv}(\mathcal{Q}) \)) because each feasible selection of \( \overline{n} \) linearly independent, binding inequalities forms an extreme point that is integral. \( Q.E.D. \)

Proof of Corollary 2. The proof parallels that for \( \mathcal{R}^{(1)} \) except that an operation cannot be assigned to station \( s \) and well as station \( |S| \) as described in Claim 5.

Proof of Claim 8. A basis matrix representing an extreme point of Claim 8 may be composed of rows

(a) \(-e_{m_{\text{max}}}^T\) for each \( mos \) combination in the extended agreeable assignment

(b') \( e_{m_{\text{max}}}^T - e_{m_{\text{ms}}}^T \) for each designated \( mo \setminus S \) combination for any \( m \in M_o \) and \( l \in L_o \)

(b) \(-e_{m'o's'}^T\) for each \( m'o's' \) combination except for the ones specified in (a) and (b')

(c) \( e_{l_{\text{ms}}}^T - e_{l_{\text{max}}}^T \) for one selected tool \( l \in L_o \) used by each \( mos \) combination in (a) and (b')

(c') \( -e_{l_{\text{ms}}}^T \) for each tool \( l' \in L_o \) \( \setminus \{l\} \) used by each \( mos \) combination in (a) and (b') that is not selected in (c): \( l' \in L_o \ \setminus \{l\} \)

(d) \(-e_{l'm's'}^T\) or \( e_{l'm's'}^T \) for each \( l' \in L_o \) associated with \( m'o's' \) combinations other than those covered in (c) and (c') giving \( \overline{n} \) linearly independent, binding inequalities. Each of these extreme points is integral.

Since tool \( l \) can be used by only one operation, it is not possible for \( z_{l_{\text{ms}}} \) to appear in more than one constraint of type (9) so that no \( z_{l_{\text{ms}}} \) variable at a fractional value can be associated with an additional binding inequality. \( Q.E.D. \)

36
Acknowledgements

This material is based upon work supported by the National Science Foundation on Grants DDM-9114396 and DMI-9500211. The authors are indebted to two anonymous referees whose comments allowed an earlier version of this paper to be strengthened.

References


*Accepted by John Hooker; received May 2001; revised May 2002; accepted October 2002.*