Product Assignment in Flexible Multilines
Part 2: Single Stage Systems with No Demand Splitting

Udatta S. Palekar
*Department of Mechanical
and Industrial Engineering
University of Illinois*

Narayan Raman
*Department of Business Administration
University of Illinois*

BEBR

Product Assignment in Flexible Multilines
Part 2: Single Stage Systems with No Demand Splitting

Udatta S. Palekar
Department of Mechanical and Industrial Engineering

Narayan Raman
Department of Business Administration

Product Assignment in Flexible Multilines
Part 2: Single Stage Systems with No Demand Splitting

Udatta S. Palekar

Department of Mechanical and Industrial Engineering

University of Illinois at Urbana-Champaign

Urbana, Illinois

Narayan Raman

Department of Business Administration

University of Illinois at Urbana-Champaign

Champaign, Illinois

January 1993

ABSTRACT

This study deals with the multiline design problem in an automated manufacturing system. This system is identical to one considered in a companion work (Raman and Palekar 1993); however, we deal with the case in which each product is required to be assigned to exactly one line. We show that this requirement renders the multiline design problem NP-complete. We construct several lower bounds for the problem. In so doing, we show that a greedy procedure presented in the companion study, for the case in which a product can be assigned to multiple lines, yields a strong lower bound to the problem considered here. We construct heuristic and exact solution methods for this problem and report our computational experience with these methods. These experiments show the efficacy of the suggested heuristic approach as well as the proposed lower bounds.

We first recapitulate the manufacturing system addressed in the companion study (Raman and Palekar 1993) that extends to this paper. We consider a facility that manufactures a set $\mathcal{N}$ of N products in medium to large volumes. Each product has an associated processing time and a demand rate. These products are processed on one or more lines, each line comprising of one or more identical machines operating in parallel. These machines are flexible in the sense that they can switch from one product to another with negligible changeover time. Products assigned to the same line share the same cycle time that equals the longest processing time of the products assigned to that line. Given the costs for opening a line and procuring a machine, the objective of the flexible multiline design problem is to determine a minimum cost partition of $\mathcal{N}$ that assigns each subset to exactly one line. In this paper, we consider the problem in which the subsets are mutually exclusive which corresponds to the requirement that each product be produced on exactly one line. Such a constraint is imposed in practice for reasons of technological requirements, ease of supervision and limitations on available tooling.

This paper is organized as follows. The problem is formulated in §1. We show that the multiline design problem is NP-complete. Note that the **FMD1** problem considered in Raman and Palekar (1993) which permits the demand of a product to be split across multiple lines is solvable in polynomial time. In §2, we present an efficient graph representation of the problem. In §3, we develop several lower bounds. We show that a greedy procedure presented in Raman and Palekar to solve **FMD1** yields a lower bound to the problem considered here. We construct a heuristic solution method in §4, and an exact algorithm based on implicit enumeration in §5. Our computational experience with these solution approaches as well as the lower bounds is discussed in §6. We conclude in §7 with a summary discussion of the main results of this paper.

1   Introduction

An integer programming formulation of the multiline design problem is given below. First, we restate the notation used in Raman and Palekar (1993). It is useful to note that the cycle time of any line in a feasible solution must equal the processing time of its *pivot* product, i. e., the product with the longest processing time that is assigned to that line.

$\mathcal{N}$ = the set of products, and $|\mathcal{N}| = N$

F_1 = fixed cost per line

$$F_2 \quad = \quad \text{fixed cost per machine}$$

$$p_j \quad = \quad \text{processing time of product } j, \quad j \in \mathcal{N}$$

$$\mathcal{J}_i \quad = \quad \text{set of products with processing times greater than or equal to } i, \{j|p_j \geq p_i, \quad j \in \mathcal{N}\}$$

$$\mathcal{I}_j \quad = \quad \text{set of products with processing times less than or equal to } j, \{i|p_i \leq p_j, \quad i \in \mathcal{N}\}$$

$$d_j \quad = \quad \text{per period demand of product } j, \quad j \in \mathcal{N}$$

$$A \quad = \quad \text{available time per period on any machine}$$

$$\tau_l \quad = \quad \text{cycle time of line } l$$

$$\pi(l) \quad = \quad \text{index of the pivot of line } l$$

$$\lambda(j) \quad = \quad \text{index of the line for which } j \text{ is the pivot}$$

$$n_j \quad = \quad \text{the number of machines required at line } \lambda(j)$$

As in Raman and Palekar (1993), we assume that $A \gg p_j, \quad \forall j \in \mathcal{N}$ so that $\lfloor A/p_j \rfloor \approx A/p_j$. The multiline design problem can be stated as

FMD2

$$\text{Minimize} \quad Z = \sum_{j=1}^{N}(F_1 y_j + F_2 n_j) \tag{1}$$

subject to

$$\sum_{j \in \mathcal{J}_i} x_{ji} = 1, \quad i \in \mathcal{N} \tag{2}$$

$$p_j\left(\sum_{i=1}^{N} d_i x_{ji}\right) \leq A n_j, \quad j \in \mathcal{N} \tag{3}$$

$$x_{ji} \leq y_j, \quad i,j \in \mathcal{N} \tag{4}$$

$$x_{ji} \in \{0,1\}, \quad i,j \in \mathcal{N} \tag{5}$$

$$y_j \in \{0,1\}; n_j \geq 0, \text{ integer}, \quad j \in \mathcal{N} \tag{6}$$

where

$$y_j \quad = \quad \begin{cases} 1, & \text{if a line is opened with pivot } j \\ 0, & \text{otherwise} \end{cases}$$

and

$$x_{ji} \quad = \quad \begin{cases} 1, & \text{if product } i \text{ is assigned to a line } \lambda(j) \\ 0, & \text{otherwise.} \end{cases}$$

Equation (2) insures that the demand of each product is fully assigned, and a product is assigned only to lines with cycle times no less than the processing time of the product. Constraint (3) requires that all product-to-line assignments be capacity feasible. Constraint (4) insures that the fixed cost of opening a line is accounted for. Finally, constraints (5) and (6) specify the nature of the variables.

FMD2 differs from the **FMD1** problem discussed in Raman and Palekar (1993) in that it requires each product to be assigned to exactly one line. It is easy to see that the total number of machines required on any line l with pivot j is given by

$$n_j = \left\lceil \frac{\sum_{i \in \mathcal{N}} d_i x_{ji} p_j}{A} \right\rceil,$$

where $\lceil f \rceil$ is the smallest integer greater than or equal to f. The unused capacity on this line, hereafter the *remnant*, is

$$R_j = A n_j / p_j - \left(\sum_{i=1}^{N} d_i x_{ji} \right).$$

While Raman and Palekar show that **FMD1** is polynomially solvable, the following result indicates that it is unlikely that an efficient solution exists for **FMD2**.

Theorem 1. **FMD2** *is NP-complete in the strong sense.*

PROOF: **FMD2** is clearly in NP. We show that the 3-PARTITION problem which is known to be NP-complete in the strong sense (Garey and Johnson 1979) is reducible to **FMD2**. Consider an arbitrary instance of 3-PARTITION given by a set Q of $3q$ elements of size s_i for each $i \in Q$, and a positive integer B such that i) $B/4 < s_i < B/2$, $\forall i \in Q$, and ii) $\sum_{i \in Q} s_i = qB$. The recognition version of the 3-PARTITION problem is stated as: Is there a partition of Q into q disjoint subsets $(Q_1, Q_2, \ldots, Q_q)$ such that $\sum_{i \in Q_j} s_i = B$, for $1 \leq j \leq q$?

The equivalent instance of **FMD2** is:

$$|\mathcal{N}| = 4q$$

$$p_j = \begin{cases} j, & j = 1, 2, \ldots, q \\ 1, & j = q+1, q+2, \ldots, 4q. \end{cases}$$

3

$$d_j = \begin{cases} qB, & j = 1, 2, \ldots, q \\ s_{j-q}, & j = q+1, q+2, \ldots, 4q. \end{cases}$$

$$A = qB + B$$

$$F_1 = 0.$$

Given an instance of 3-PARTITION, this instance of **FMD2** can be constructed in polynomial time. We show that for this instance, $Z \leq F_2 q(q+1)/2$, if and only if 3-PARTITION has a solution.

Denote the set of products $j = 1, 2, \ldots, q$ by $\mathcal{N}_1$, and let $\mathcal{N}_2 = \mathcal{N}\backslash\mathcal{N}_1$. First suppose that 3-PARTITION has a solution. The required solution for **FMD2** is constructed by forming q lines such that each product $j \in \mathcal{N}_1$ is a pivot. The number of machines required individually for any product assigned to a line with pivot j is

$$\mu_j = \left\lceil \frac{jqB}{(qB+B)} \right\rceil = j \tag{7}$$

and the resulting remnant on this line is

$$R_j = \frac{\mu_j A - p_j d_j}{p_j} = j\{(qB+B) - qB\}/j = B \text{ units.} \tag{8}$$

The total number of machines required under this assignment for products in $\mathcal{N}_1$ is $\sum_{l=1}^{q} l = q(q+1)/2$. Clearly, if 3-PARTITION has a solution, $\mathcal{N}_2$ can be partitioned into q disjoint subsets $\mathcal{N}_{21}, \mathcal{N}_{22}, \ldots, \mathcal{N}_{2q}$ such that $\sum_{j \in \mathcal{N}_{2l}} d_j = B$, $l = 1, 2, \ldots, q$. Each of the q lines is assigned one of these subsets to give the desired solution with cost $Z = F_2 q(q+1)/2$.

Next, we show that if $Z \leq F_2 q(q+1)/2$ under an assignment σ (say), then 3-PARTITION has a solution. First, note that because $p_i \geq p_j, i \in \mathcal{N}_1, j \in \mathcal{N}_2$, there must be at least one line in σ that has a pivot which belongs to $\mathcal{N}_1$. [In particular, q must be a pivot.] More generally, suppose that σ has $L + T$ lines, of which the pivots for the first L lines belong to $\mathcal{N}_1$. Clearly, $L \leq q$. Let $\mathcal{L}_l$ be the set of products assigned to line l, and let $\mathcal{L}_l^1 = \{j | j \in \mathcal{N}_1 \cap \mathcal{L}_l\}$, $l = 1, 2, \ldots, L$, denote a subset of $\mathcal{L}_l$ that belongs to set $\mathcal{N}_1$ as well. Also, let γ_l be the cardinality of $\mathcal{L}_l^1$. First consider the assignment of products in $\mathcal{N}_1$ in these L lines.

Lemma 1. $\mathcal{L}_l^1 = \{l\}$, $l = 1, 2, \ldots, L$, and hence, $L = q$.

PROOF: Let the products assigned to line l under σ be

$$\mathcal{L}_l^1 = \{i_1, i_2, \cdots, i_{\gamma_l}\}$$

4

such that $i_1 < i_2 < \cdots < i_{\gamma_l}$. Since the processing times of products in $\mathcal{N}_1$ are the same as their index and since a product in $\mathcal{N}_1$ can only be assigned to a line with a pivot with an equal or higher index, it follows that $i_{\gamma_l} \leq l$. Now consider an alternative assignment σ' in which products $i_1, i_2, \ldots, i_{\gamma_l-1}$ are pivots for their individual lines. From (7), the total number of machines required under σ' for these γ_l products is

$$
\begin{aligned}
n_l' &= i_1 + i_2 + \cdots + i_{\gamma_l-1} + i_{\gamma_l} \\
&\leq (l-1)(\gamma_l - 1) + l.
\end{aligned} \tag{9}
$$

Note that (9) is satisfied as an equality only if either $\mathcal{L}_l^1 = \{l\}$ or $\mathcal{L}_l^1 = \{l-1, l\}$. The number of machines required on line l under σ is given by

$$
\begin{aligned}
n_l &= \left\lceil \frac{lqB\gamma_l}{qB+B} \right\rceil \\
&= \left\lceil l\gamma_l - \frac{l\gamma_l}{q+1} \right\rceil \\
&\geq l\gamma_l - (\gamma_l - 1), \quad \text{because } l < q+1 \\
&\geq (l-1)(\gamma_l - 1) + l \\
&\geq n_l'.
\end{aligned}
$$

Hence,

$$
\sum_{l=1}^{L} n_l \geq \sum_{l=1}^{L} n_l' = 1 + 2 + \ldots + q = q(q+1)/2.
$$

But $Z \leq F_2 q(q+1)/2$ implies that

$$
\sum_{l=1}^{L} n_l \leq q(q+1)/2,
$$

and therefore,

$$
\sum_{l=1}^{L} n_l = q(q+1)/2.
$$

Furthermore, $T = 0$, and either $\mathcal{L}_l^1 = \{l\}$, or $\mathcal{L}_l^1 = \{l-1, l\}$, $l = 1, 2, \ldots, L$.

Now, suppose that $\mathcal{L}_j^1 = \{j-1, j\}$ for some j, $1 \leq j \leq L$. Then

$$
\begin{aligned}
n_j &= \left\lceil \frac{2jqB}{qB+B} \right\rceil \\
&= \left\lceil 2j - \frac{2j}{q+1} \right\rceil.
\end{aligned}
$$

Because $j < q + 1$, either $n_j = 2j$ or $n_j = 2j - 1$. But if $n_j = 2j$, then

$$\sum_{l=1}^{L} n_l \geq \sum_{l=1}^{j-2} n'_l + 2j + \sum_{l=j+1}^{q} n'_l$$
$$= 1 + 2 + \ldots + (j - 2) + 2j + (j + 1) + \ldots + q$$
$$> q(q + 1)/2$$

which is not feasible. Hence, it must be true that $n_j = (2j - 1)$. The capacity available on line j for assigning products in $\mathcal{N}_2$ is

$$CAP(j) = [(2j - 1)(qB + B) - 2jqB]/j < 2B \text{ units.} \tag{10}$$

From (8), the capacity available on any line l, such that $\mathcal{L}_l = \{l\}$ is B units. Hence, the total capacity available on all L lines for products in $\mathcal{N}_2$

$$\sum_{l=1}^{L} CAP(l) < 2B + (L - 2)B \leq qB.$$

But the minimum capacity required for products in $\mathcal{N}_2$ is

$$\sum_{j=q+1}^{4q} d_{j-q} - \sum_{i=1}^{3q} s_i = qB,$$

which implies that $\mathcal{L}_j \neq \{j - 1, j\}$, and the only feasible configuration under σ is $\mathcal{L}_l = \{l\}$, $l = 1, 2, \ldots, L$, and therefore, $L = q$. $\square$

Now consider the products in $\mathcal{N}_2$. Given that there are q lines, each with a residual capacity of B after assigning products in $\mathcal{N}_1$, a feasible solution to **FMD2** can exist only if $\mathcal{N}_2$ can be partitioned into q disjoint subsets $\mathcal{N}_{21}, \mathcal{N}_{22}, \ldots, \mathcal{N}_{2q}$ such that $\sum_{j \in \mathcal{N}_{2l}} d_j = B$, $l = 1, 2, \ldots, q$. But this implies that 3-PARTITION has a solution. $\square$

Note that **FMD2** is NP-complete even when there are no line costs, i.e., $F_1 = 0$.

2 Problem Representation

Without any loss of generality, we assume in the rest of this paper that products are indexed such that if $i < j$, then $p_i \geq p_j$. Similar to the representation of **FMD1** in Raman and Palekar (1993), it is possible to represent **FMD2** on graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ shown in Figure 1. In this graph, node V_{ij}, which is depicted as ij, represents an assignment in which product j is produced on a line with

pivot i. Because V_{ij} is feasible only if $p_j \leq p_i$, the graph is upper triangular. We append a dummy source node S and a dummy sink node T to $\mathcal{G}$.

Arc-set $\mathcal{E} = \{E_{ij}^{u,j+1}\}$ consists of arcs connecting contiguous nodes V_{ij} and $V_{u,j+1}$. $\mathcal{E}$ can be partitioned into disjoint subsets $\mathcal{H}$, $\mathcal{B}$ and $\mathcal{F}$ where $\mathcal{H}$ is the set of *horizontal* arcs *(h-arcs)* of the form $E_{ij}^{i,j+1}$, while $\mathcal{B}$ is the set of *backward* arcs *(b-arcs)* of the form $E_{ij}^{u,j+1}$ where $u < i$. $\mathcal{F}$ comprises the *forward* arcs *(f-arcs)* $E_{ij}^{u,j+1}$ such that $u > i$. Without any loss of generality, we assume that all arcs leading from S and leading into T are f-arcs.

<div align="center">

┌─────────────────────────┐
│ INSERT FIGURE 1 HERE │
└─────────────────────────┘

</div>

Raman and Palekar (1993) show that the following result holds for **FMD1**.

Sequential Assignment Property (SAP): There exists an optimal solution with the property that if $x_{ji} > 1$, *then* $x_{jq} = 1$ *for* $q = j+1, j+2, \ldots, i-1$.

However, if a product can be assigned to only one line, the remnant available at any line cannot be utilized for meeting the partial demand of a product from another line. In such a case, it is easy to show that the *sequential assignment property* is no longer dominant for **FMD2**. In the absence of this property, there is no "natural" order of products assigned to any line because the optimal solution can consist of one or more b-arcs. However, $\mathcal{G}$ is a sufficient representation of **FMD2** in the sense that any solution to **FMD2** can be depicted as a path from S to T with no cycles, and the optimal solution can be posed as a shortest path problem on $\mathcal{G}$. Consequently, in order to keep the notation simple, we assume without loss of generality in the rest of the paper that any product is assigned only after all lower numbered products are assigned, and define the arc costs as well as the remnants accordingly. This essentially enables us to scan $\mathcal{G}$ from left to right.

Following this approach, we now determine the cost of each arc in $\mathcal{G}$. Consider node V_{jk} in $\mathcal{G}$. The number of machines M_{jk} required at line $\lambda(j)$ corresponding to this node will only consider products $j, j+1, \ldots, k$; hence

$$M_{jk} = \left\lceil \frac{p_j \left(\sum_{u=j}^{k} d_u x_{ju} \right)}{A} \right\rceil.$$

The remnant available at V_{jk} is

$$r_{jk} = \frac{A M_{jk}}{p_j} - \left(\sum_{u=j}^{k} d_u x_{ju} \right).$$

The cost of h-arc $E_{jk}^{j,k+1}$ is

$$c_{jk}^{j,k+1} = F_2 \left\lceil \frac{p_j\,(d_{k+1} - r_{jk})}{A} \right\rceil, \tag{11}$$

and the cost of f-arc $E_{jk}^{k+1,k+1}$ is

$$c_{jk}^{k+1,k+1} = F_1 + F_2 \left\lceil \frac{p_{k+1} d_{k+1}}{A} \right\rceil. \tag{12}$$

Note that the costs of all f-arcs incident on node $V_{k+1,k+1}$ are the same. It is also easy to see that a pivot product must be assigned to its own line in an optimal solution because otherwise, the products assigned to this line are being produced at higher than required cycle time. Consequently,

Remark 1. *If an optimal path consists of an f-arc $E_{jk}^{u,k+1}$, $u > j$, or a b-arc $E_{jk}^{u,k+1}$, $u < j$, then this path must pass through V_{uu} and V_{jj} as well.*

It follows that

Corollary 1. *If the f-arc $E_{jk}^{l,k+1}$, $l \neq k+1$ is contained in any optimal path, then such a path must include one or more b-arcs as well.*

The cost of arc $E_{jk}^{l,k+1}$, $l \neq k+1$ then equals

$$c_{jk}^{l,k+1} = F_2 \left\lceil \frac{p_l\,(d_{k+1} - r_{lk})}{A} \right\rceil. \tag{13}$$

3 Lower Bounds on FMD2

In this section, we construct four lower bounds for **FMD2**. The first bound is obtained by relaxing the integrality requirements on the number of machines required. The second bound is obtained by allowing a product to be produced on multiple machines, and solving the resulting **FMD1** problem by the **Greedy** heuristic discussed in Raman and Palekar (1993). The other two bounds are derived by relaxing constraints (2) and (3), respectively, and solving the resulting Lagrangean problems.

3.1 Fractional Machines

The first bound $LB1$ is obtained by relaxing the integrality requirement for n_j in constraint (6). In any optimal solution, constraint (3) is then satisfied as an equality. In particular,

$$n_j = \sum_i c_{ji} x_{ji}, \quad \text{where } c_{ji} = d_i p_j / A.$$

8

FMD2 then reduces to

FMD2-FM

$$LB1 = min \left(\sum_j F_1 y_j + \sum_j \sum_i F_2 c_{ji} x_{ji} \right)$$

subject to

$$\sum_{j \in \mathcal{J}_i} x_{ji} = 1$$

$$x_{ji} \leq y_j$$

$$x_{ji}, y_j \in \{0, 1\}.$$

Proposition 1. *There exists an optimal solution to* **FMD2-FM** *that follows SAP.*

PROOF: Let σ_1 be an optimal solution to **FMD2-FM** that does not have this property. Then there must be a pair of products s and t, $s < t$, such that s is assigned to line $\lambda(j)$ and t is assigned to line $\lambda(k)$ with $k < j$. Let σ_2 be identical to σ_1 except in that t is produced on line $\lambda(j)$. Let $Z(\cdot)$ denote the cost of solution $(\cdot)$. Then

$$Z(\sigma_1) - Z(\sigma_2) = \frac{d_t p_k}{A} - \frac{d_t p_j}{A} = (p_k - p_j)\frac{d_t}{A} \geq 0.$$

Hence, σ_2 is optimal as well. Repeating this argument for all such pairs (s, t) gives the desired result. □

Now consider the following problem discussed by Corneujols, Nemhauser and Wolsey (1990).

Tree Partitioning Problem: Suppose $\overline{G} = (\mathcal{N}, \overline{\mathcal{E}})$ is a tree graph, and $\mathbf{D}$ *is a $|\mathcal{N}|$ by $|\mathcal{N}|$ matrix with elements D_{ij} that represent the distance between nodes v_i and v_j, for all $v_i, v_j \in \overline{\mathcal{N}}$. Let the weight of any subtree $\overline{G}_j = (\overline{\mathcal{N}}_j, \overline{\mathcal{E}}_j)$ be $w(\overline{G}_j) = max_{v_k \in \overline{\mathcal{N}}_j} \left(\sum_{v_i \in \overline{\mathcal{N}}_j} D_{ki} \right)$. Determine the partition of $\overline{G}$ into subtrees that minimizes the sum of subtree weights.*

Corneujols et al. present a dynamic programming algorithm that solves the tree partitioning problem in $O(|\mathcal{N}|^2)$ time. Given the sequential assignment policy, **FMD2-FM** can be modeled as a tree partitioning problem as follows. Let $\overline{\mathcal{N}} = \{v_1, \ldots, v_N\}$ denote products 1 through N, and let $\overline{\mathcal{E}} = \{E_{v_i}^{v_i+1} : i = 1, 2, \ldots, N - 1\}$ be the set of arcs that connects consecutively numbered products. Note that, in any feasible solution, there is a unique path, comprising one or more pivots,

from v_1 to v_N. Furthermore, in any solution to this problem, each pivot generates a subtree. The equivalence is complete if we assign

$$D_{ii} = F_1 + F_2 \frac{d_i p_i}{A}, \quad \text{and}$$

$$D_{ij} = \frac{F_2 p_i \sum_{t=j}^{i} d_t}{A}, \quad \text{if } i < j$$

$$= \infty \quad \text{otherwise.}$$

for any $i, j \in \mathcal{N}$. $LB1$ can, therefore, be determined in $O(N^2)$ time.

3.2 Demand Splitting

The second bound is obtained by relaxing constraints (5) to read $x_{ji} \geq 0$, and solving the resulting **FMD1** problem using algorithm **Greedy** described in Raman and Palekar (1993). Let the resulting solution value be $LB2$.

Proposition 2. *LB2 is a lower bound on the optimal solution value for* **FMD2**.

PROOF: In the following, $\mathcal{G}^G$ denotes the subgraph followed by the **Greedy** algorithm when demand splitting is permitted (see Raman and Palekar 1993). We show that the cost of the optimal path Ω^* in $\mathcal{G}$ from V_{11} to T is no less than the cost of at least one path between these two nodes in $\mathcal{G}^G$.

Let $\mathcal{C}(\cdot)$ $[\mathcal{C}^G(\cdot)]$ denote the cost, and $\eta(\cdot)[\eta^G(\cdot)]$ denote the number of machines required by path $(\cdot)$ in $\mathcal{G}$ $[\mathcal{G}^G]$. Let σ_0 be the longest segment of Ω^* that originates at V_{11} and consists only of f- and h-arcs, and let V_{uv} be the node at which σ_0 terminates. Such a segment must exist because only an f-arc and a b-arc emanate from V_{11}; also, $u \geq 2$. In Figures 2a and 2b, σ_0 is the path $E_{11}^{12} - E_{12}^{33}$ $- E_{44}^{33}$, and V_{uv} is V_{44}. Because it does not include any b-arcs, σ_0 must exist in $\mathcal{G}^G$ as well and is, therefore, considered by **Greedy**.

$$\boxed{\text{INSERT FIGURES 2a AND 2b HERE}}$$

Let $\mathcal{P}_0$ be the set of pivots visited by σ_0. At the end of σ_0, for any line $\lambda(l)$, $l \in \mathcal{P}_0$, let R_l denote the remnant in $\mathcal{G}$, R_l^G the remnant in $\mathcal{G}^G$, n_l the number of machines in $\mathcal{G}$ and n_l^G be the number of machines in $\mathcal{G}^G$. Recall that because demand splitting is permitted in $\mathcal{G}^G$, the remnant R_l^G at any line will be used for partially meeting the demands of products from subsequent lines. Then, for $\lambda(l) = 1$,

10

$$n_1 = n_1^G,$$

and for $\lambda(l) \geq 2$,

$$
\begin{aligned}
n_l^G &= \left\lceil \frac{p_l \left(\sum_{t=l}^{\pi(\lambda(l)+1)-1} d_t - R_{l-1}^G \right)}{A} \right\rceil \\
&\leq \left\lceil \frac{p_l \left(\sum_{t=l}^{\pi(\lambda(l)+1)-1} d_t \right)}{A} \right\rceil \\
&= n_l.
\end{aligned}
\tag{14}
$$

Hence, the number of machines saved by permitting demand splitting is

$$
\begin{aligned}
\xi &= \eta(\sigma_0) - \eta^G(\sigma_0) \\
&= \sum_{l \in \mathcal{P}_0} n_l - \sum_{l \in \mathcal{P}_0} n_l^G \\
&\geq 0.
\end{aligned}
\tag{15}
$$

Since the number of pivots in σ_0 is the same in both $\mathcal{G}$ and $\mathcal{G}^G$, $\mathcal{C}(\sigma_0) \geq \mathcal{C}^G(\sigma_0)$ and the proof is complete if $\Omega^* = \sigma_0$. Otherwise, suppose that Ω^* consists of one or more b-arcs. At the end of σ_0 in $\mathcal{G}$, the remnant at any line $\lambda(l), l \in \mathcal{P}_0$ is

$$R_l = \frac{A n_l}{p_l} - \sum_{t=l}^{\pi(\lambda(l)+1)-1} d_t$$

and the total idle capacity available for possibly absorbing demands of subsequent products is $\sum_{l \in \mathcal{P}_0} R_l$. On the other hand, the total idle capacity available at the end of σ_0 in $\mathcal{G}^G$ is R_u^G which is given by

$$R_u^G = \frac{A n_u^G}{p_u} - \sum_{t=u}^{v} d_t + R_\tau^G$$

where $\tau = \pi(\lambda(u) - 1)$ is the pivot of the line immediately preceding $\lambda(u)$. From induction, it is easy to show that

$$
\begin{aligned}
R_u^G &= \sum_{l \in \mathcal{P}_0} \frac{A n_l^G}{p_l} - \sum_{t=1}^{v} d_t \\
&= \sum_{l \in \mathcal{P}_0} R_l - A \left[\sum_{l \in \mathcal{P}_0} \left(n_l - n_l^G \right) / p_l \right] \\
&\geq \sum_{l \in \mathcal{P}_0} R_l - \frac{\xi A}{p_u}.
\end{aligned}
\tag{16}
$$

11

because $p_u \leq p_l$, $l \in \mathcal{P}_0$. Let σ_1 denote the segment on Ω^* that originates at V_{uv} (with a b-arc) and ends with an f-arc incident upon a node (say) $V_{\theta,v+k}$ such that $\theta \geq u$. Note that there will always be such a segment given that the terminal node T is reachable only with an f-arc. Figure 2a depicts the case in which $\theta = u$, and σ_1 is the path $E_{44}^{35} - E_{35}^{36} - E_{36}^{17} - E_{17}^{48}$. Figure 2b depicts the case in which $\theta > u$, and σ_1 is the path $E_{44}^{35} - E_{35}^{36} - E_{36}^{17} - E_{17}^{88}$. Let the set of pivots visited by σ_1 be $\mathcal{P}_1$. Note that

$$p_u < p_l, \quad \forall l \in \mathcal{P}_1 \backslash \{\theta\}. \tag{17}$$

Let $\mathcal{A}_l$ denote the set of products assigned to pivot $l, l \in \mathcal{P}_1$ along σ_1. Consider the following cases:

Case I: $\theta = u$.

From Remark 1, we have in this case

$$\mathcal{P}_1 \subsetneq \mathcal{P}_0. \tag{18}$$

Let σ_1' denote the path $E_{u,v}^{u,v+1} - - \cdots - E_{u,v+k-1}^{u,v+k}$. In Figure 2a, σ_1' is the path $E_{44}^{45} - E_{45}^{46} - E_{46}^{47} - E_{47}^{48}$. We will show that

$$\mathcal{C}(\sigma_0 \cup \sigma_1) \geq \mathcal{C}^G(\sigma_0 \cup \sigma_1').$$

Note that

$$\eta^G(\sigma_1') = \left\lceil \frac{p_u \left[\left(\sum_{l \in \mathcal{P}_1} \sum_{t \in \mathcal{A}_l} d_t \right) - R_u^G \right]}{A} \right\rceil$$

and

$$
\begin{aligned}
\eta(\sigma_1) &= \sum_{l \in \mathcal{P}_1} \left\lceil \frac{p_l \left(\sum_{t \in \mathcal{A}_l} d_t - R_l \right)}{A} \right\rceil \\
&\geq \sum_{l \in \mathcal{P}_1} \left\lceil \frac{p_u \left(\sum_{t \in \mathcal{A}_l} d_t - R_l \right)}{A} \right\rceil \quad \text{from (17)} \\
&\geq \left\lceil \frac{\sum_{l \in \mathcal{P}_1} p_u \left(\sum_{t \in \mathcal{A}_l} d_t - R_l \right)}{A} \right\rceil \tag{19} \\
&= \left\lceil \frac{p_u \left[\left(\sum_{l \in \mathcal{P}_1} \sum_{t \in \mathcal{A}_l} d_t \right) - \sum_{l \in \mathcal{P}_1} R_l \right]}{A} \right\rceil \\
&\geq \left\lceil \frac{p_u \left[\left(\sum_{l \in \mathcal{P}_1} \sum_{t \in \mathcal{A}_l} d_t \right) - R_u^G \right]}{A} - \xi \right\rceil \tag{20} \\
&= \eta^G(\sigma_1') - \xi \\
&= \eta^G(\sigma_1') - \left[\eta(\sigma_0) - \eta^G(\sigma_0) \right]
\end{aligned}
$$

12

where (19) follows from the well-known inequality

$$\sum_{y=1}^{Y} \lceil a_y \rceil \geq \left\lceil \sum_{y=1}^{Y} a_y \right\rceil$$

and (20) follows from (16) and (18). Hence, $\eta(\sigma_0) + \eta(\sigma_1) \geq \eta^G(\sigma_0) + \eta^G(\sigma_1')$. Since no new pivots are opened under both σ_1 and σ_1', $\mathcal{C}(\sigma_0 \cup \sigma_1) \geq \mathcal{C}^G(\sigma_0 \cup \sigma_2)$.

Case II: $\theta > u$.

In this case, $\theta = v + k$ and

$$\mathcal{P}_1 \backslash \{\theta\} \subseteq \mathcal{P}_0. \tag{21}$$

Let σ_1'' denote the path $E_{u,v}^{u,v+1} - - \cdots - - E_{u,v+k-1}^{\theta,v+k}$. In Figure 2b, σ_1'' is the path $E_{44}^{45} - E_{45}^{46} - E_{46}^{47} - E_{47}^{88}$. We will show that

$$\mathcal{C}(\sigma_0 \cup \sigma_1) \geq \mathcal{C}^G(\sigma_0 \cup \sigma_1'').$$

We have

$$\eta(\sigma_1) = \sum_{l \in \mathcal{P}_1 \backslash \{\theta\}} \left\lceil \frac{p_l \left(\sum_{t \in \mathcal{A}_l} d_t - R_l \right)}{A} \right\rceil + \left\lceil \frac{p_\theta d_\theta}{A} \right\rceil.$$

Using arguments similar to those used in case I, it can be shown that

$$\sum_{l \in \mathcal{P}_1 \backslash \{\theta\}} \left\lceil \frac{p_l \left(\sum_{t \in \mathcal{A}_l} d_t - R_l \right)}{A} \right\rceil \geq \left\lceil \frac{p_u \left[\left(\sum_{l \in \mathcal{P}_1 \backslash \{\theta\}} \sum_{t \in \mathcal{A}_l} d_t \right) - R_u^G \right]}{A} - \xi \right\rceil.$$

Hence

$$\begin{aligned}
\eta(\sigma_1) &\geq \left\lceil \frac{p_u \left[\left(\sum_{l \in \mathcal{P}_1 \backslash \{\theta\}} \sum_{t \in \mathcal{A}_l} d_t \right) - R_u^G \right]}{A} - \xi \right\rceil + \left\lceil \frac{p_\theta \left(d_\theta - R_u^G \right)}{A} \right\rceil \\
&= \eta^G(\sigma_1'') - \xi.
\end{aligned}$$

Since σ_1 and σ_1'' have the same number of pivots, $\mathcal{C}(\sigma_0 \cup \sigma_1) \geq \mathcal{C}^G(\sigma_0 \cup \sigma_1'')$.

Therefore, in both cases, the proof is complete if $\Omega^* = \sigma_0 \cup \sigma_1$. Otherwise, we identify the next segment σ_2 in Ω^* that originates at $V_{\theta,v+k}$ with a b-arc and ends with an f-arc incident upon a node $V_{\theta',v+k+k'}$ such that $\theta' \geq \theta$. Note that both σ_1, and σ_1' and σ_1'' terminate on the same node $V_{\theta,v+k}$. Set $\sigma_0 = \sigma_0 \cup \sigma_1$, and repeat the above arguments for σ_2. $\quad\square$

The following result shows that this bound is at least as strong as $LB1$.

13

Proposition 3. $LB2 \geq LB1$.

PROOF: We show that the cost of any path Ω from V_{11} to T under **Greedy** is at least as large as the cost of the same path when fractional machines are permitted. Let the set of pivots on Ω be $\mathcal{P}$, and let $P = |\mathcal{P}|$. Also, let $\mathcal{L}_{\lambda(l)}, l \in \mathcal{P}$ denote the set of products assigned to pivot l. Then the total cost of Ω given fractional machines is

$$C_1(\Omega) = PF_1 + F_2 \sum_{l \in \mathcal{P}} \frac{p_l \sum_{t \in \mathcal{L}_{\lambda(l)}} d_t}{A} \tag{22}$$

and the cost of Ω under **Greedy**, with $R_0^G = 0$, is

$$
\begin{aligned}
C_2(\Omega) &= PF_1 + F_2 \sum_{l \in \mathcal{P}} \left\lceil \frac{p_l \left(\sum_{t \in \mathcal{L}_{\lambda(l)}} d_t - R_{l-1}^G \right)}{A} \right\rceil \\
&= PF_1 + F_2 \sum_{l \in \mathcal{P}} \frac{p_l \left(\sum_{t \in \mathcal{L}_{\pi(l)}} d_t - R_{l-1}^G + R_l^G \right)}{A} \\
&= C_1(\Omega) + F_2/A \sum_{l \in \mathcal{P}} p_l (R_l^G - R_{l-1}^G) \\
&= C_1(\Omega) + F_2/A \left[\sum_{l \in \mathcal{P} \backslash \{P\}} R_l^G (p_l - p_{l+1}) + R_P^G \right] \\
&\geq C_1(\Omega)
\end{aligned}
$$

because $p_l \geq p_{l+1}$ for all $l \in \mathcal{P} \backslash \{P\}$ and $R_l^G \geq 0, \forall l$. $\quad\square$

3.3 Lagrangean Relaxation 1

The third bound $LB3$ is obtained by Lagrangean relaxation (see, for example, Geoffrion 1974). We dualize constraints (2) with nonnegative multipliers u_i. The resulting problem is

LR1

$$\text{Minimize} \quad Z(\mathbf{u}) = \sum_{j=1}^{N} (F_1 y_j + F_2 n_j) - \sum_{i=1}^{N} u_i \left(\sum_{j \in \mathcal{J}_i} x_{ji} - 1 \right) \tag{23}$$

subject to

$$p_j \left(\sum_{i=1}^{N} d_i x_{ji} \right) \leq A n_j, \quad j \in \mathcal{N}$$

$$x_{ji} \leq y_j, \quad i, j \in \mathcal{N}$$

$$x_{ji} \in \{0, 1\}, \quad i, j \in \mathcal{N}$$

$$y_j \in \{0, 1\}; n_j \geq 0, \text{ integer}, \quad j \in \mathcal{N}$$

14

For given multipliers, **LR1** separates into N independent problems, where the jth problem is

LR1$_j$

$$\text{Minimize} \quad Z_j(\mathbf{u}) = F_1 y_j + F_2 n_j - \sum_{i \in \mathcal{I}_j} u_i x_{ji} \tag{24}$$

subject to

$$p_j \left(\sum_{i \in \mathcal{I}_j} d_i x_{ji} \right) \leq A n_j, \tag{25}$$

$$x_{ji} \leq y_j, \tag{26}$$

$$x_{ji} \in \{0, 1\}, \quad i \in \mathcal{I}_j \tag{27}$$

$$y_j \in \{0, 1\}; n_j \geq 0, \text{ integer}, \tag{28}$$

LR1$_j$ is separable and it can be solved by solving the following knapsack problem

KP$_j$

$$Y_j(\mathbf{u}) = min \left(F_2 n_j - \sum_{i \in \mathcal{I}_j} u_i x_{ji} \right)$$

subject to

$$p_j \left(\sum_{i \in \mathcal{I}_j} d_i x_{ji} \right) \leq A n_j,$$

$$x_{ji} \in \{0, 1\}, \quad i \in \mathcal{I}_j$$

Note that $Y_j \leq 0$, in an optimal solution since $n_j = 0, x_{ji} = 0, \quad \forall i$ is a feasible solution to **KP$_j$**. If $Y_j < -F_1$, then $y_j = 1$ in an optimal solution to **LR1$_j$** with a solution value of $Z_j^* = F_1 - Y_j$. Otherwise, $y_j = n_j = x_{ji} = 0$ in the optimal solution for all i, yielding the optimum solution value $Z_j^* = 0$.

The Lagrangean dual problem is

LRD1

$$LB3 = \quad max \quad \sum_{i=1}^{N} Z_i(\mathbf{u}) + \sum_{i=1}^{N} u_i \tag{29}$$

with u_i unrestricted. However, we need consider only nonnegative values of these multipliers, because if $u_i < 0$ for any i, then $x_{ji} = 0$ in an optimal solution to **LR1**. While it is possible to use subgradient optimization, it is often more effective to first use a dual ascent technique that can

guarantee bound improvement at each iteration. We use an ascent approach that exploits violation of constraints (2). Define

$$\mathcal{X}^1 = \{i | \sum_{j \in \mathcal{J}_i} x_{ji} = 0\}$$

and

$$\mathcal{X}^2 = \{i | \sum_{j \in \mathcal{J}_i} x_{ji} \geq 2\}$$

First consider the products in set $\mathcal{X}^1$, i. e., those products that have not been assigned to any line. The minimum increase required in u_i for any $i \in \mathcal{X}^1$ before it can be assigned to a line depends upon the status of the line. Based upon the solution to problems $\mathbf{LR1}_j$, $j \in \mathcal{N}$, a line corresponding to pivot j can either be

i) open; i.e., $y_j = 1$, or

ii) closed, but with assignable products; i.e., $y_j = 0$, and $-F_1 < Y_j(\mathbf{u}) < 0$. Note that in this case $\sum_{i \in \mathcal{I}_j} x_{ji} \geq 1$, or

iii) closed with no assignable products; i.e., $y_j = 0$, and $Y_j(\mathbf{u}) = 0$. In this case, $\sum_{i \in \mathcal{I}_j} x_{ji} = 0$.

In case i), the number of machines required to include i in line $\lambda(j)$ is $\lceil p_j(d_i - R_j)/A \rceil$. Therefore, an increase of

$$v_j = F_2 \lceil p_j(d_i - R_j)/A \rceil - u_i$$

in u_i will guarantee that $x_{ji} = 1$. However, this increase may result in a decrease in the value of Z_j. To insure that this does not occur, we first solve $\mathbf{KP}_j$ with multiplier $u_i' = u_i + v_j$. If the resulting solution value is Y_{new}, then the largest increase in u_i that does not result in any increase in Z_j is

$$\delta_j = (v_j - u_i) - (Y_j - Y_{new}).$$

Consequently, based only on the open lines, the maximum permissible increase in u_i to insure $\sum_{j \in \mathcal{J}_i} x_{ji} = 1$ is

$$\Delta_1 = min \ \{\delta_j | y_j = 1\}.$$

Using similar arguments in case ii), the maximum permissible increase in u_i based upon lines $\lambda(j)$ that are currently closed with assignable products is

$$\Delta_2 = min \ \{F_1 + (F_2 \lceil p_j(d_i - R_j)/A \rceil - u_i + Y_j) - (Y_j - Y_{new}) | y_j = 0, \ Y_j < 0\},$$

16

and in case iii), the maximum permissible increase in u_i based upon lines $\lambda(j)$ that is currently closed with no assignable products is

$$\Delta_3 = min \ \{F_1 + (F_2\lceil p_j d_i/A\rceil - u_i) - (Y_j - Y_{new})\,|y_j = 0, \ Y_j = 0\}.$$

Therefore, the overall maximum permissible increase in u_i to insure that $x_{ji} = 1$ is $\Delta = min \ \{\Delta_1, \Delta_2, \Delta_3\}$ and the bound value $LB3$ increases by Δ.

Next consider the set $\mathcal{X}^2$, i. e., the set of products that have been assigned to more than one line. For any $i \in \mathcal{X}^2$, we first compute the minimum amount by which u_i needs to be decreased in order to be excluded from any line $\lambda(j)$ that it is currently assigned to. If u_i is decreased by

$$w_j = u_i - F_2 \left\{ \left\lceil \frac{p_j \sum_{q\in\mathcal{I}_j} d_q x_{jq}}{A} \right\rceil - \left\lceil \frac{p_j \left(\sum_{q\in\mathcal{I}_j} d_q x_{jq} - d_i\right)}{A} \right\rceil \right\}$$

then i's contribution to Y_j becomes positive, and consequently, $x_{ji} = 0$ in an optimal solution to **LR1**$_j$. However, this increase may lead to a decrease in Z_j as well. In order to determine the maximum decrease in u_{i_2} that will retain the same value of Z_j, we next solve **KP**$_j$ with multiplier $u_i - w_j$. If the resulting solution is Y_{new}, then the largest decrease in u_i that does not result in a decrease in Z_j is

$$\delta_j = (u_i - w_j) - (Y_j - Y_{new}).$$

The second largest value of such decreases across all open lines is

$$\Delta = max_{\{j\in\mathcal{J}_i|j\neq j^*\}} \ \{\delta_j\}$$

where $j^* = arg \ \ max_{j\in\mathcal{J}_i}\{\delta_j\}$. If u_i is decreased by Δ, then an alternative optima to **LR1** is obtained in which i is assigned to exactly one line. Note that this results in an overall bound increase of $\left(|\sum_{j\in\mathcal{J}_i} x_{ji}| - 1\right)\Delta$.

It is always possible to further improve this bound by using a subgradient method (see, for example, Held, Wolfe and Crowder 1974) to search for multipliers after the ascent stops. This was done in the computational study reported in §6.

3.4 Lagrangean Relaxation 2

We add the following inequalities to the formulation (1) – (6) of **FMD2**.

$$\sum_{j=1}^{l}\sum_{i\in\mathcal{J}_i} p_j d_i x_{ji} \leq A\sum_{j=1}^{l} n_j \quad l = 1, 2, \ldots, N$$

17

Clearly, any solution that satisfies (3) will satisfy the above inequalities as well. Now we relax constraints (3) by dualizing them with nonnegative multipliers u_j. The resulting Lagrangean problem is

LR2

$$\text{Minimize} \quad Z(\mathbf{u}) = \sum_{j=1}^{N}(F_1 y_j + F_2 n_j) + \sum_{j=1}^{N} u_j \left(p_j \sum_{i \in \mathcal{I}_j} p_j d_i x_{ji} - A n_j \right) \tag{30}$$

subject to

$$\sum_{j \in \mathcal{J}_i} x_{ji} = 1, \quad i \in \mathcal{N} \tag{31}$$

$$\sum_{j=1}^{l} \sum_{i \in \mathcal{J}_i} p_j d_i x_{ji} \leq A \sum_{j=1}^{l} n_j \quad l = 1, 2, \ldots, N \tag{32}$$

$$x_{ji} \leq y_j, \tag{33}$$

$$x_{ji} \in \{0, 1\}, \quad i, j \in \mathcal{N} \tag{34}$$

$$y_j \in \{0, 1\}; n_j \geq 0, \text{ integer}, \quad j \in \mathcal{N} \tag{35}$$

The objective function can be restated to read

$$Z(\mathbf{u}) = \sum_{j=1}^{N} F_1 y_j + \sum_{j=1}^{N} (F_2 - A u_j) n_j + \sum_{j=1}^{N} u_j p_j \sum_{i \in \mathcal{I}_j} d_i x_{ji}$$

Note that Z is unbounded from below for any value of u_j such that $F_2 < A u_j$. Therefore, in order to obtain a meaningful bound, we enforce the constraints

$$F_2 - A u_j \geq 0 \quad \forall j \tag{36}$$

Proposition 4. *Suppose that the multipliers $u_j, j = 1, 2, \ldots, N$ are selected such that for any $j, k \in \mathcal{N}, p_j \geq p_k$ implies $u_j p_j \geq u_k p_k$. Then there is at least one optimal solution to* **LR2** *that follows the sequential assignment policy.*

PROOF: From (32), it follows that the surplus capacity κ_l available at line $\lambda(l)$ is carried forward to the next line. In particular, for the first line,

$$\kappa_1 = A n_1 - \sum_{i \in \mathcal{I}_1} d_i x_{1i}$$

and in general, for any pivot l,

$$\kappa_l = A n_l - \sum_{i \in \mathcal{I}_l} d_i x_{li} + \kappa_q$$

18

where q is the pivot of the line immediately preceding $\lambda(l)$. Let σ_1 be an optimal solution to **LR2** that does not possess the sequential assignment property. Then there must be a product t that is assigned to line $\lambda(k)$ while another product s, $s < t$, is assigned to line $\lambda(j)$ with $k < j$. Let σ_2 be identical to σ_1 in the number of machines as well as the products assigned to each line, except in that t is produced on line $\lambda(j)$. This solution is feasible because it results in an increase of $p_k d_t$ in the surplus capacity of all lines $\lambda(k)$ through $\lambda(j) - 1$, and an increase of $(p_k - p_j)d_t > 0$ in the surplus capacity of all the subsequent lines. If $Z(\cdot)$ denotes the cost of solution $(\cdot)$ to **LR2**, then

$$Z(\sigma_1) - Z(\sigma_2) = (u_k p_k - u_j p_j) \geq 0$$

and if σ_1 is optimal, then so is σ_2. Repeating the above argument whenever necessary gives the desired result. □

Similar to the demand splitting problem **FMD1** discussed in Raman and Palekar (1993), **LR2** can then be posed as a shortest path problem on graph $\widehat{\mathcal{G}} = \left(\widehat{\mathcal{V}}, \widehat{\mathcal{E}} \right)$. Following arguments analogous to Raman and Palekar, it can be shown that the number of machines required, as well as the surplus capacity available at any node depends upon the path selected to reach that node. For a path Ω in which i and j, $j > i$, are adjacent pivots, the number of machines required at line $\lambda(j)$ corresponding to node V_{jk} is

$$M_{jk} = \left\lceil \frac{p_j \left(\sum_{u=j}^{k} d_u \right) - \varrho_{i,j-1}}{A} \right\rceil,$$

where $\varrho_{i,j-1}$ is the surplus capacity at $V_{i,j-1}$. At any node V_{it}, $t = 1, 2, \ldots, N$, the surplus capacity $\varrho_{1t} = r_{1t}$. In general, the surplus capacity at V_{jk} is

$$\varrho_{jk} = A M_{jk} - \left(\sum_{u=j}^{k} d_u \right) + \varrho_{i,j-1}.$$

The cost of h-arc $E_{jk}^{j,k+1}$ is

$$c_{jk}^{j,k+1} = (F_2 - A u_j) \left\lceil \frac{p_j d_{k+1} - \varrho_{jk}}{A} \right\rceil + u_j p_j d_{k+1}, \tag{37}$$

and the cost of the f-arc $E_{jk}^{k+1,k+1}$ is

$$c_{jk}^{k+1,k+1} = F_1 + (F_2 - A u_j) \left\lceil \frac{p_j d_{k+1} - \varrho_{jk}}{A} \right\rceil + u_{k+1} p_{k+1} d_{k+1}, \tag{38}$$

LR2 can be solved in polynomial time using the algorithm described in Raman and Palekar (1993). The Lagrangean dual problem is

$$LB4 = \quad max \quad Z(\boldsymbol{u})$$

subject to

$$F_2 - Au_j \geq 0 \quad \forall j$$

$$u_{j+1} - u_j p_j / p_{j+1} \leq 0 \quad j = 1, 2, \ldots, N-1 \tag{39}$$

$$u_j \geq 0. \quad \forall j$$

We solve for the multipliers using the subgradient optimization method with a simple modification to account for the upper bounds given by (39).

4 A Heuristic Algorithm

Given the strong NP-completeness of **FMD2**, it is likely that most real problems will need to be solved using heuristic approaches. In this section, we construct an efficient solution method that is an improvement heuristic. Note that while the optimal solution need not satisfy the sequential assignment property (SAP), such a policy can be enforced to obtain a heuristic solution. The proposed method starts with the SAP solution and iteratively improves upon it.

Given the sequential assignment policy, **FMD2** reduces to a tree partitioning problem by assigning

$$D_{ii} = F_1 + \left\lceil \frac{d_i p_i}{A} \right\rceil, \quad \text{and}$$

$$
\begin{aligned}
D_{ij} &= F_2 \left\lceil \frac{p_i \sum_{t=j}^{i} d_t}{A} \right\rceil, \quad \text{if } i < j \\
&= \infty \quad \text{otherwise.}
\end{aligned}
$$

Following Corneujols, Nemhauser and Wolsey (1990), the optimal SAP solution can be found in $O(N^2)$ time. Note that

Remark 2. *If the path corresponding to an optimal solution to* **FMD2**
does not contain any b-arcs, then the SAP solution is optimal.

PROOF: From Corollary 1, it follows that if there are no b-arcs in the optimal path, then all f-arcs in that path must be of the form $E_{jk}^{k+1,k+1}$. Since the SAP solution considers all such f-arcs and all resulting h-arcs, it must be optimal. □

Clearly, any improvement on the SAP solution is predicated on the existence of one or more b-arcs. Consider an optimal path Ω^* from S to T in $\mathcal{G}$ that consists of a b-arc $E_{j,k-1}^{uk}$. From Remark 1, both j and u must be pivots. Let

$$\mu_{uk} = \frac{p_u d_k}{A}$$

be the (possibly) fractional number of machines required on line $\lambda(u)$ to produce k ignoring any available remnant. Also, let $\mu_{jk} = \alpha_{jk} + \beta_{jk}$, where α_{jk} is the integer part of μ_{jk}. $\beta_{jk} = \mu_{jk} - \lfloor \mu_{jk} \rfloor$, where $\lfloor f \rfloor$ is the largest integer no greater than f, is the purely fractional part of μ_{jk}. Then

Proposition 5. $\alpha_{uk} \leq \alpha_{jk} + 1$.

PROOF: If Ω^* does not have this property, then construct an alternative path Ω' which differs from Ω^* only in that product k is assigned to line $\lambda(j)$. Let n'_j denote the resulting number of machines on $\lambda(j)$. Then

$$
\begin{aligned}
n_u &= \left\lceil \frac{p_u \sum_i d_i x_{ui}}{A} \right\rceil \\
&\geq \left\lceil \frac{p_u \left(\sum_i d_i x_{ui} - d_k \right)}{A} \right\rceil + \left\lceil \frac{p_u d_k}{A} \right\rceil - 1 \qquad (40) \\
&= n'_u + \lceil \alpha_{uk} + \beta_{uk} \rceil - 1 \qquad (41)
\end{aligned}
$$

where (40) follows from the inequality

$$\lceil a + b \rceil \geq \lceil a \rceil + \lceil b \rceil - 1.$$

It can similarly be shown that,

$$n_j \geq n'_j - \lceil \alpha_{jk} + \beta_{jk} \rceil. \qquad (42)$$

Because Ω^* is optimal, $n_u + n_j \leq n'_u + n'_j$. From (41) and (42), it then follows that

$$\lceil \alpha_{uk} + \beta_{uk} \rceil \leq \lceil \alpha_{jk} + \beta_{jk} \rceil + 1$$

Since, $0 \leq \beta_{jk}, \beta_{uk} < 1$, this is possible only if $\alpha_{uk} \leq \alpha_{jk} + 1$. $\quad \square$

The following corollary extends the above result to include b-arcs involving a subset of products.

Corollary 2. *Let Γ_{jt} denote a subset of products that is assigned to line j in the SAP solution. If Γ_{jt} is assigned to line $\lambda(l)$, $l < j$, in an optimal solution then $\gamma_{lt} \leq \gamma_{jt} + 1$, where*

$$\gamma_{qt} = \left\lfloor \frac{p_q \sum_{i \in \Gamma_{jt}} d_i}{A} \right\rfloor.$$

There are two kinds of b-arcs imbedded in any SAP solution. The first kind, hereafter referred to as a *Type 1* b-arc, exists *within* each line. Identifying such b-arcs and updating the SAP solution accordingly results in the single line being split into two or more lines. Consider the following 3-product problem: $F_1 = 250$; $F_2 = 130$; $A = 800$; $p_1 = 10$; $p_2 = 5$; $p_3 = 2$; $d_1 = 100$; $d_2 = 280$; $d_3 = 60$. The SAP solution, $x_{11} = x_{12} = x_{13} = 1$, results in a single line. However, the optimal solution to this problem is $x_{11} = x_{22} = x_{13} = 1$ which induces the b-arc E_{22}^{13} as shown in Figure 3. Updating the SAP solution results in an additional pivot at product 2.

INSERT FIGURE 3 HERE

The second kind of b-arc, hereafter a *Type 2* b-arc, exists across lines; these b-arcs result in re-assigning a product to a different line. However, such b-arcs do not create any new lines. Consider the following 4-product example illustrated in Figure 4: $F_1 = 1000$; $F_2 = 100$; $A = 800$; $p_1 = 10$; $p_2 = 9$; $p_3 = 2$; $p_4 = 1$; $d_1 = 100$; $d_2 = 200$; $d_3 = 1590$; $d_4 = 20$. The SAP solution is $x_{11} = x_{12} = x_{33} = x_{34} = 1$ with two lines having pivots 1 and 3 respectively. The optimal solution is $x_{11} = x_{12} = x_{33} = x_{14} = 1$ that induces the b-arc E_{33}^{14} resulting in product 4 being re-assigned from line 2 to line 1.

INSERT FIGURE 4 HERE

The proposed solution method is an improvement heuristic that modifies the SAP solution by iteratively identifying Type 1 and Type 2 b-arcs and updating the solution accordingly. A formal statement of the algorithm is given below.

Algorithm **MultilineDesign**

Step 0: Initialization: Determine the optimal SAP solution. In case of multiple optima, select the solution with the largest number of pivots; break ties arbitrarily. Go to Step 1.

Step 1: Identification of Type 1 b-arc and Solution Update: Determine if a Type 1 b-arc exists that can result in a cost reduction. Update the current solution accordingly. Go to Step 2.

Step 2: Identification of Type 2 b-arc and Solution Update: If a Type 2 b-arc exists that can result in a cost reduction, then update the current solution accordingly and go to Step 1. Else, stop.

Steps 1 and 2 are now discussed in detail.

4.1 Identification of Type 1 b-arcs

Suppose that the SAP solution results in L lines. Starting with line 1, we consider each line in order, and attempt to determine the optimal solution with respect to the products allocated to that line. This is done by identifying the b-arcs, if any, that are imbedded within that line. Note that any improvement at this step is contingent upon the existence of one or more b-arcs, and any such b-arc must result in the generation of one or more additional pivots.

Consider an arbitrary line l, and let $\mathcal{L}_l = \{\pi(l), \ldots, \pi(l+1) - 1\}$ be the set of Q (consecutively numbered) products assigned to this line under SAP. Note that in the SAP solution, $x_{li} = 1 \ \forall \ i \in \mathcal{L}_l$. Let $\mathbf{SP2}_l$ denote the subproblem of $\mathbf{FMD2}$ that considers only those products that are in $\mathcal{L}_l$, and let $\mathbf{SP1}_l$ denote the corresponding problem in which demand splitting is permitted. Since each line can be treated independently, hereafter in this subsection we renumber the products in $\mathcal{L}_l$ as $1, \ldots, Q$ and suppress subscript l. Note that the SAP solution to $\mathbf{SP2}$ consists of a sequence of h-arcs $E_{11}^{12} - \ldots - E_{1,Q-1}^{1,Q}$.

First we state the condition under which no b-arcs exist in a given line, and consequently, the SAP solution is optimal to $\mathbf{SP2}$. From Proposition 2, recall that the **Greedy** solution to $\mathbf{SP1}$ is a lower bound on $\mathbf{SP2}$. Therefore,

Remark 3. *The SAP solution is optimal to* $\mathbf{SP2}$, *if it is the* **Greedy** *solution to* $\mathbf{SP1}$ *as well.*

When this above condition is not satisfied, the optimal solution to $\mathbf{SP2}$ may consist of pivots in addition to 1. We now construct a heuristic method that solves $\mathbf{SP2}$ through a 2-stage approach. In the first stage, we identify the additional pivots required, and in the second stage the various products are assigned to these pivots. These two stages are now discussed.

4.1.1 Determination of Pivots

Suppose that the optimal solution to $\mathbf{SP2}$ consists of K lines with pivots from the set $\mathcal{K} = \{\pi(1), \pi(2), \ldots, \pi(K)\}$. Consider the path that passes through these pivots and that satisfies the sequential assignment property, i.e., the path $E_{\pi(1),\pi(1)}^{\pi(1),\pi(1)+1} - \cdots - E_{\pi(1),\pi(2)-1}^{\pi(2),\pi(2)} - \cdots - E_{\pi(K),Q-1}^{\pi(K),Q}$.

23

Let $C^G(\sigma)$ be the cost of this path in $\mathcal{G}^G$. Let $Z^*(\cdot)$ and $Z^S(\cdot)$ denote the cost of the optimal and the SAP solutions, respectively, to problem $(\cdot)$. Then from the proof of Proposition 2, it follows that

Remark 4. $C^G(\sigma) \leq Z^*(\mathbf{SP2}) \leq Z^S(\mathbf{SP2})$.

Remark 4 suggests that the set of optimal pivots can be found by first constructing a graph $\mathcal{G}^G$ corresponding to **SP2**, identifying a path in this graph that has a lower cost than Z^S, and selecting the pivots corresponding to this path. However, there are two problems with such an approach. First, the number of paths to be investigated can be large. This problem can be circumvented by considering a limited number of paths; in the computational experiments, we consider only the **Greedy** solution to **FMD1**. Second, a path in $\mathcal{G}^G$ may contain one or more "spurious" pivots, i.e., pivots that may be efficient for **FMD1** but that lead to inferior solutions for **FMD2**. For example, consider the following 3-product problem: $F_1 = 250$; $F_2 = 130$; $A = 800$; $p_1 = 10$; $p_2 = 5$; $p_3 = 2$; $d_1 = 100$; $d_2 = 280$; $d_3 = 80$. The optimal solution to **SP2** is the SAP solution $x_{11} = x_{12} = x_{13} = 1$. However, the **Greedy** solution to this problem is $x_{11} = x_{22} = x_{23} = 1$ which generates an additional, spurious pivot at product 2.

It is, however, possible to eliminate one or more spurious pivots by the following result which requires that under a relatively weak condition, each pivot other than 1, must be associated with one or more b-arcs in at least one optimal solution. Suppose that the optimal set of pivots for **SP2** is $\mathcal{K} = \{\pi(1), \pi(2), \ldots, \pi(K)\}$, and let σ be the path that passes through these pivots and that satisfies the sequential assignment property. Define a pivot in σ as a *donor* if one or more products assigned to it is reassigned to a lower numbered pivot in the optimal solution to **SP1**. Alternatively, pivot k is a *receiver* if a subset of products assigned to a higher numbered pivot in σ is reassigned to it in the optimal solution to **SP1**. Figure 5 shows σ in broken lines and the optimal path in bold lines. With respect to this figure, pivots 5 and 7 are donors while 3 is a receiver. A pivot can be both a donor and a receiver. Let ψ_{jk} denote the integer part of the number of machines required to produce products $k, k+1, \ldots, Q$ on line $\lambda(j)$, i.e.,

$$\psi_{jk} = \left\lfloor \frac{p_j \sum_{q=k}^{Q} d_q}{A} \right\rfloor.$$

Remark 5. *Suppose that $k \in \mathcal{K} \backslash \{1\}$ is a pivot in an optimal solution to* **SP2** *such that $\psi_{1k} > \psi_{kk} + 1$. Then k must be either a donor pivot or a receiver pivot.*

PROOF: First note that Step 0 of **MultilineDesign** insures that the SAP solution selected σ^S (say) is one with the maximum number of pivots in case of multiple SAP solutions with the same value. Therefore, with respect to the products in $\mathcal{L}_l$, σ^S uniquely has the lowest solution value among all SAP solutions. σ^S is the path connecting V_{11}, $V_{12} - \cdots - V_{1,Q-1}$ and V_{1Q} in Figure 6. Suppose there exists an optimal path σ^* that contains a pivot k which is neither a donor nor a receiver, but $\psi_{1k} > \psi_{kk} + 1$. σ^* is shown with bold lines in Figure 6. Let σ_1 be another path that is identical to σ^* except that all products assigned to pivot k in σ^* are now assigned to pivot 1. Then

$$\mathcal{C}(\sigma_1) - \mathcal{C}(\sigma^*)$$
$$= -F_1 + F_2 \left\{ \left\lceil \frac{p_1 \left(\sum_{i \in \mathcal{L}_1 \cup \mathcal{L}_{\lambda(k)}} d_i \right)}{A} \right\rceil - \left\lceil \frac{p_1 \left(\sum_{i \in \mathcal{L}_1} d_i \right)}{A} \right\rceil - \left\lceil \frac{p_k \left(\sum_{i \in \mathcal{L}_{\lambda(k)}} d_i \right)}{A} \right\rceil \right\} \geq 0. \qquad (43)$$

Now construct another path σ_2 in which products 1 through $k - 1$ are produced on line 1, and products k through Q are produced on line 2 with pivot k. Clearly, σ_2 is another SAP solution, and

$$\mathcal{C}(\sigma^S) - \mathcal{C}(\sigma_2) = -F_1 + F_2 \left\{ \left\lceil \frac{p_1 \left(\sum_{i=1}^{Q} d_i \right)}{A} \right\rceil - \left\lceil \frac{p_1 \left(\sum_{i=1}^{k-1} d_i \right)}{A} \right\rceil - \left\lceil \frac{p_k \left(\sum_{i=k}^{Q} d_i \right)}{A} \right\rceil \right\} < 0 \qquad (44)$$

because σ^S has uniquely the lowest cost among all SAP solutions. Note that $\mathcal{L}_1 \bigcup \mathcal{L}_{\lambda(k)} \subseteq \{1, 2, \ldots, Q\}$, and $\mathcal{L}_{\lambda(k)} \subseteq \{k, k+1, \ldots, Q\}$. Following arguments similar to those used in the proof of Proposition 5, it can then be shown that Equations (43) and (44) are consistent only if $\psi_{1k} \leq \psi_{kk} + 1$. But this contradicts the original assumption. $\quad\square$

<div align="center">

INSERT FIGURE 6 HERE

</div>

Note that this result may not hold if the integer number of machines required for *all* products $k, k+1, \ldots, Q$ at line $\lambda(k)$ differs from the number of machines required for these products on line $\lambda(1)$ by no more than one machine. However, this is a relatively weak condition that is likely to be satisfied in most real problems. Also note that the sets of donors and receivers cannot be determined exactly unless the optimal solution to **SP2** is known. However, we make use of the fact that such a solution must satisfy the condition stated in Proposition 5.

We now formally state the procedure that determines the "best" set of pivots for **SP2**. In the following, $\mathcal{D}$ and $\mathcal{R}$ denote the set of donor and receiver pivots, respectively.

Algorithm **SetPivots**

Step 0: Set $j = 0$.

Step 1: Solve **SP1** using the **Greedy** approach. If the optimal **Greedy** solution is the same as the SAP solution to **SP2**, stop. Else, let $\mathcal{O} = \{\sigma | \mathcal{C}^G(\sigma) < Z^S(SP2)\}$ be the set of candidate paths to be investigated. Arrange these paths in the nondecreasing order of their costs. Go to step 2.

Step 2: Select the path Ω from the top of the list; let $\mathcal{K} =$ be the set of pivots in Ω, and $K = |\mathcal{K}|$. Set $l = 1$; $\mathcal{D} = \emptyset$; $\mathcal{R} = \{1\}$. Go to step 3.

Step 3: a) Set $l \leftarrow l + 1$.

b) If $\alpha_{\pi(l),i} \leq \alpha_{\pi(t),i} + 1$ for some i and t such that $\pi(l) < i \leq Q$, and $t \in \mathcal{R} \bigcup \mathcal{D}$, then set

$$\mathcal{D} = \mathcal{D} \bigcup \{\pi(l)\}$$

$$\mathcal{R} = \mathcal{R} \bigcup \{t\}$$

$$\mathcal{K} = \mathcal{K} \backslash \{\pi(l)\}$$

and go to step 4. Otherwise, delete Ω from $\mathcal{O}$ and go to step 5.

Step 4: If $l \neq K - 1$, go to step 3a), else go to step 5.

Step 5: Stop if all paths in $\mathcal{O}$ are scanned. Else, go to step 2.

Using Remark 3, Step 1 checks for the optimality of the SAP solution by solving **SP1** with the **Greedy** approach. If this test fails, then **SetPivots** first identifies the stack of paths with solution values less than $Z^S(SP2)$. For any path Ω from this stack, step 3 enforces the condition stated in Remark 5 to check if each pivot, other than 1, is either a donor or a receiver. If any path fails this test, then it is removed from the candidate list of paths to be pursued further.

4.1.2 Product Assignment to Sublines

At the end of **SetPivots**, we have a stack $\mathcal{O}$ of paths; each path in this stack decomposes what was originally one line into K, $K \geq 2$ sublines. Furthermore, we have identified the sets $\mathcal{D}$ and $\mathcal{R}$ of donor and receiver pivots, respectively, within each path Ω in this stack. Note that Ω consists of only f-arcs and h-arcs resulting in sequential assignment of all products. While this path is suboptimal for **SP2**, we now attempt to improve it by reassigning products and in so doing, generate b-arcs. Let n_k, $k = 1, 2, \ldots, K$ denote the number of machines required and R_k be the remnant at line $\lambda(k)$ in Ω.

Let $j \in \mathcal{D}$ be a donor pivot in Ω. Let Γ_{jt}, $t = 1, 2, \ldots$ denote the subsets of products that are currently assigned to line $\lambda(j)$ but that are eligible for reassignment to another line. Γ_{jt} is *eligible* if there exists a line $\lambda(l)$, $l \in \mathcal{R} \backslash \{j\}$ such that assigning Γ_{jt} to $\lambda(l)$ results in a reduction in the number of machines $n_j + n_l$ at lines $\lambda(j)$ and $\lambda(l)$. This is possible only if Γ_{jt} makes use of the remnant R_l available at line $\lambda(l)$. In particular, the fractional part of the machines required by Γ_{jt} at $\lambda(l)$ should be no more than R_l, i.e.,

$$\delta_{jt}^l \stackrel{\text{def}}{=} \left(\frac{p_l \sum_{i \in \Gamma_{jt}} d_i}{A} - \left\lfloor \frac{p_l \sum_{i \in \Gamma_{jt}} d_i}{A} \right\rfloor \right) \frac{A}{p_l} \leq R_l.$$

Let $\phi_{jt}^l = 1$ if Γ_{jt} can be reassigned to $\lambda(l)$, zero otherwise. Also, let $\mathcal{D}_j = \{\Gamma_{jt}\}$ denote the collection of product subsets that are assigned to pivot j in Ω and that are eligible for reassignment. The product reassignment problem can then be formulated as

GAP

$$\text{Maximize} \quad \sum_{l \in \mathcal{R}} \sum_{j \in \mathcal{D}} \sum_{\Gamma_{jt} \in \mathcal{D}_j} \phi_{jt}^l Y_{jt}^l \tag{45}$$

subject to

$$\sum_{l \in \mathcal{R}} \sum_{\Gamma_{jt} \in \mathcal{D}_j} Y_{jt}^l \leq 1 \quad j \in \mathcal{D} \tag{46}$$

$$\sum_{j \in \mathcal{D}} \sum_{\Gamma_{jt} \in \mathcal{D}_j} \phi_{jt}^l \delta_{jt}^l Y_{jt}^l \leq R_l \quad \forall l \in \mathcal{R} \tag{47}$$

$$Y_{jt}^l \in \{0, 1\} \quad \forall \Gamma_{jt} \in \mathcal{D}_j, \ \forall j \in \mathcal{D}, \ \text{and} \ \forall l \in \mathcal{R} \tag{48}$$

where Y_{jt}^l equals 1 if subset Γ_{jt} is reassigned to line l, 0 otherwise. The objective function (45) maximizes the number of reassigned subsets. Constraints (46) specify that no more than one subset is reassigned from any line. In particular, because a job can belong to more than one subset, these

constraints guarantee that job reassignments are not duplicated. Constraint (47) is a knapsack constraint that insures that the fractional machine required for any reassigned product subset is no more than the available remnant.

GAP is a generalized assignment problem. [See, for example, Ross and Soland 1975; Fisher, Jaikumar and van Wassenhove 1986; Guignard and Rosenwein 1989; Martello and Toth 1990]. While this problem is NP-complete, there are reasonably efficient solution methods available for it. This problem is easily solved in our case in particular because in **SP2**, the number of alternative lines that a product subset can be reassigned to is usually very small which yields a sparse $\Phi = [\phi_{jt}^l]$ matrix.

4.2 Identification of Type 2 b-arcs

Let the incumbent solution at the end of Step 1 comprise L lines. At the second step, we consider each line in order, and try to minimize the idle capacity on this line by reassigning a subset of products that is currently assigned to higher numbered lines (and thereby, generating b-arcs). Suppose that the line being considered is $\lambda(l)$. We enumerate all subsets Γ_{jt} of products that are currently assigned to some other line $\lambda(j)$, $j \neq l$, and that are eligible for being reassigned to $\lambda(l)$. Let Θ_{jl} denote the set of feasible product-subsets that are currently assigned to line q and which are candidates for assigning to line l. The resulting problem to be solved for line l is formulated as:

KP$_l$

$$\text{Maximize} \quad \sum_{j=l+1}^{L} \sum_{\Gamma_{jt} \in \Theta_{jl}} w_{jt}^j Y_{jt} \tag{49}$$

subject to

$$\sum_{\Gamma_{jt} \in \Theta_{jl}} Y_{jt} \leq 1 \quad j = 1, \ldots, L; j \neq l \tag{50}$$

$$\sum_{j=l+1}^{L} \sum_{\Gamma_{jt} \in \Theta_{jl}} \delta_{jt}^l Y_{jt} \leq R_l \tag{51}$$

$$Y_{jt} \in \{0, 1\} \quad \forall \, \Gamma_{jt} \in \Theta_{jl}, \text{ and } l = 1, 2, \ldots, L \tag{52}$$

where Y_{jt} equals 1 if subset Γ_{jt} is reassigned to line l, 0 otherwise. The objective function (49) maximizes the weighted number of reassigned subsets. The weight w_{jt}^j assigned to subset Γ_{jt} is selected such that a higher weight is given to subsets at lines with smaller indices so that ties are broken in the favor of these subsets. Because the lines are considered sequentially starting with the

smallest indexed line, this objective facilitates better reassignment in the subsequent steps. Also, among the various subsets on the same line, higher weight is given to the subset that frees up more capacity. In the computational experiments reported in §6, the functional form $w_{jt}^j = exp(\delta_{jt}^j/j)$ was used. Constraints (50) and (51) parallel constraints (46) and (47), respectively.

Problem $\mathbf{KP}_l$ is a knapsack problem with generalized upper bound constraints (50). This is a well-studied problem (see the related references in Martello and Toth 1990). Dyer, Kayal and Walker (1984) provide an effective algorithm for solving it.

5 An Exact Algorithm

We now present an implicit enumeration method for solving $\mathbf{FMD2}$ exactly. Branching in the solution tree shown in Figure 7 is based on the assignment of products to lines. For greater clarity, the vertices shown in the tree are labeled according to the nodes V_{ij} that they correspond to in $\mathcal{G}$. The root vertex corresponds to the assignment of product 1 to line 1; clearly line 1 must be opened in any feasible solution because product 1 has the largest processing time. Each subsequent level in the enumeration tree corresponds to the assignment of a product considered sequentially in the order of its index. At level j of the tree, we generate vertices in the solution tree corresponding to each feasible assignment of product j to a line. Since each product $i \leq j$ is potentially a pivot, this step requires generating j vertices corresponding to each of the nodes $V_{1j}, V_{2j}, \ldots, V_{jj}$ in $\mathcal{G}$. A depth-first strategy is used to search the enumeration tree.

$$\boxed{\text{INSERT FIGURE 7 HERE}}$$

Lower Bounding

The lower bound at any vertex in the enumeration tree is determined by using $LB2$ discussed in §3.3. Consider vertex u in the tree that corresponds to V_{ij} in $\mathcal{G}$. Let Ω denote the path leading from the root vertex to u. Ω represents the partial solution comprising of assignments of products $1, 2, \ldots, j$. Let $\mathcal{C}(u)$ be the cost of Ω, and let ρ_u denote the sum of the remnants on all lines opened in Ω. The lower bound $LB(u)$ at vertex u is determined by first netting ρ_u units from the demands of products $j + 1, \ldots, N$ considered in that order. Suppose that ρ_u can completely meet the demands of products $j + 1$ through $k - 1$, and partially satisfy the demand of k. Let d_t', $k \leq t \leq N$ denote the net demands remaining to be satisfied. Then

29

$$d'_k \leq d_k, \quad \text{and}$$

$$d'_t = d_t \text{ for } k+1 \leq t \leq N. \tag{53}$$

Define $\mathbf{P}^1_u$ as a relaxed subproblem of **FMD2** that considers only products $k, k+1, \ldots, N$, and that permits demand splitting. Product demands and processing times are given, respectively, by d'_t, and $p'_t = p_t$, $k \leq t \leq N$. Let $Z^G(\mathbf{P}^1_u)$ denote the **Greedy** solution value for $\mathbf{P}^1_u$. Then, from (53) and Proposition 2, $Z^G(\mathbf{P}^1_u)$ is a lower bound on any completion of path Ω if *product k generates a new line.*

Consider another subproblem $\mathbf{P}^2_u$ that differs from $\mathbf{P}^1_u$ only in that $p'_k = p_i$. This essentially insures that product k is assigned to the line with pivot i. Let $Z^G(\mathbf{P}^2_u)$ denote the **Greedy** solution value for $\mathbf{P}^2_u$. Then, as above, it follows that $Z^G(\mathbf{P}^2_u) - F_1$ is a lower bound on any completion of path Ω if *product k does not generate a new line.*

A valid lower bound on u is then given by

$$LB(u) = \mathcal{C}(\Omega) + min \ [Z^G(\mathbf{P}^1_u), Z^G(\mathbf{P}^2_u) - F_1].$$

Upper Bounding

The upper bound at vertex u in the tree is similarly derived by solving two subproblems. Let R_i denote the remnant at line $\lambda(i)$ after assigning product j. We first net R_i units from the demands of products $j+1, \ldots, N$ considered in that order. At the end of this step, suppose that the products with positive net demands are $k, k+1, \ldots, N$. Let d''_t denote the net demand of product t, $k \leq t \leq N$. Then, $d''_k \leq d_k$, and $d''_t = d_t$, $k+1 \leq t \leq N$. We now consider two subproblems of **FMD2** at u. In the first subproblem $\mathbf{P}^3_u$, we assign k to line $\lambda(i)$. $\mathbf{P}^3_u$ is defined for products $k, k+1, \ldots, N$, with demands d''_t, $k \leq t \leq N$, and processing times $p''_k = p_i$, and $p''_t = p_t$ for $k+1 \leq t \leq N$. If $Z^S(\mathbf{P}^3_u)$ is the SAP solution value for $\mathbf{P}^3_u$, then $Z^S(\mathbf{P}^3_u) - F_1$ is a feasible upper bound on any completion of path Ω that assigns product k to line $\lambda(i)$.

We define $\mathbf{P}^4_u$ as the other subproblem of **FMD2** in which k generates its own line. $\mathbf{P}^4_u$ considers products $k, k+1, \ldots, N$, with demands d_t and processing times p_t, $k \leq t \leq N$. Then its SAP solution value $Z^S(\mathbf{P}^4_u)$ provides another feasible upper bound on any completion of path Ω. An overall upper bound at u is then given by

$$UB(u) = \mathcal{C}(\Omega) + min \ [Z^G(\mathbf{P}^3_u) - F_1, Z^G(\mathbf{P}^4_u),]$$

Branching Strategy

While determining the upper bound at u if it can be determined that there are no b-arcs in an optimal completion of the partial solution at u, then from Remark 2, u is fathomed with solution value $UB(u)$. Noting that u corresponds to node V_{ij} in $\mathcal{G}$, the non-existence of b-arcs in any optimal completion at u is guaranteed from Proposition 5 if $\alpha_{lt} > \alpha_{it} + 1$ for all $j + 1 \leq t \leq N$, and for any l that is a pivot on path Ω. This approach is implemented in the algorithm by scanning the pivots along Ω in the decreasing order of their indices. Since $\alpha_{lt} \geq \alpha_{kt}$ if $l < k$, scanning can be terminated whenever this condition is satisfied at any pivot.

Similar arguments are used to limit the number of descendants generated at an vertex. Consider a descendant v of u that represents the assignment of product $j + 1$ to line $\lambda(g)$ such that $1 < g < i$. The generation of v augments Ω with a b-arc $E_{ij}^{g,j+1}$. Clearly, if $\alpha_{g,j+1} > \alpha_{i,j+1} + 1$, then this augmentation can never be a part of an optimal solution, and v is fathomed. In order to exploit this dominance, we generate the descendants in the decreasing order of the pivot indices. The generation of additional descendants is terminated whenever a descendant is dominated.

Further efficiency is gained by using the notion of remnant dominance. Consider the vertices corresponding to the assignment of product j. Let $Z^G\left(\mathbf{P}_{u1}^1\right)$ and $Z^G\left(\mathbf{P}_{u2}^1\right)$ correspond to the assignment of j to lines $k1$ and $k2$, $k2 > k1$, respectively. If $\rho_{u1} \leq \rho_{u2}$, then $Z^G\left(\mathbf{P}_{u1}^1\right) < Z^G\left(\mathbf{P}_{u2}^1\right)$. Thus, if the vertices are scanned in the decreasing order of the line assignments, then it is possible to avoid bound calculation at any vertex by first attempting to fathom against previously calculated bounds.

6   Computational Experience

Our computational experience comprises two sets of experiments. In the first set, we use the optimum solution obtained by the enumeration procedure described in §5 to evaluate the performance of algorithm **MultilineDesign** as well as the various lower bounds discussed in §3. Because of the computational burden involved, this set is restricted to small problems. The second set deals with larger problems of the size that we have encountered in an existing system in the automotive industry. In this set, we use the tightest lower bound value as the benchmark for evaluating **MultilineDesign** and the lower bounds. The product demand and processing time values used in

both sets reflect the characteristics of this facility. However, additional scenarios are generated to provide a wider range of problem instances.

The fixed cost per line F_1 is randomly generated from the discrete uniform distribution $U_D(0, 50)$, while the machine cost F_2 is sampled from $U_D(100, 1000)$. Part processing times are generated from the continuous uniform distribution $U(0.1, 5.0)$. The total time available per period per machine A is retained at 800 across all experiments. Finally, product demands are sampled from two demand streams with equal probability. The demand for large-volume products is generated from the discrete distribution $U_D(100, 2000)$ while the demand for medium-volume products is generated from $U_D(5, 100)$. In the first set, the number of products N is sampled from $U_D(10, 20)$, while in the second set N is generated from $U_D(10, 50)$. All experiments were conducted on an IBM RS 6000 workstation.

The first set consists of 20 different problem scenarios generated randomly. These are shown in Table 1 together with the solution value $Z(MLD)$ obtained using algorithm **MultilineDesign** (MLD), as well as the lower bound values $LB2$ and $LB3$. $LB3$ is computed by the dual ascent method described in §3.3 augmented by a subgradient optimization procedure with 30 iterations. $LB4$ was dominated in all instances by at least one of $LB2$ and $LB3$, and therefore, it is not reported. All values are normalized with respect to the optimum solution. These results indicate that MLD yields excellent solution values; it is optimal is 18 instances. $Z(MLD)$ is 0.4% more on average, and 4.5% more in the worst case, than the optimal solution. Both $LB2$ and $LB3$ are seen to be strong bounds; neither one dominates the other. $LB2$ is, on average, 3.8% less than the optimal value; in the worst case, it is 8.1% less. $LB3$ has a somewhat better average performance, although is more variable. It is 3.1% and 13.2% away from the optimal solution, on average and in the worst case, respectively.

<div align="center">

INSERT TABLE 1 HERE

</div>

The second set consists of 40 problem instances; these are shown in Table 2. As with the first experiment, $LB4$ was dominated in all instances by at least one of $LB2$ and $LB3$, and it is, therefore, not reported. The lower bound and solution values shown in Table 2 are normalized with respect to the largest lower bound $L_{max} = max(LB2, LB3)$.

These results indicate that the strong performance of **MultilineDesign**, and $LB2$ and $LB3$ extend to larger problems as well. MLD is about 1.1% more than L_{max} on average, and 3.0% in the worst case, across the 40 problems. It required an average computational time of 0.334 seconds. Between $LB2$ and $LB3$, neither is dominant across all problems. $LB2$ is, on average, 1.0% less than L_{max}; in the worst case, it is 5.3% less. The corresponding values for $LB3$ are 2.2% and 14.8%, respectively. The average computational times for $LB2$ and $LB3$ are 0.02 seconds and 147.6 seconds, respectively.

In both sets, we also monitor the intermediate SAP solution values obtained while implementing **MultilineDesign**. In the first set, the SAP value is found to be 3.1% more on average, and 6.6% more in the worst case, than the optimal solution. In the second set, it is 3.8% more than L_{max} on average. However, in the worst case, it is about 27% more than L_{max} (which is also the optimal solution value in this instance). Thus, while the average SAP solution value is reasonably good, it has high variability.

7 Conclusion

This paper addresses the design of a single-stage manufacturing system with parallel lines comprising flexible workcenters. For a given set of products with their processing requirements and demands, the objective of the design problem is to determine the product-to-line allocation that minimizes the total investment in the lines and the workcenters. In this paper, we consider the problem in which each product is required to be assigned to exactly one line.

We show that the flexible multiline design problem **FMD2** is NP-complete. In view of its complexity, we develop an implicit enumeration approach for solving this problem. We present several lower bounding procedures for this problem based on various relaxations. In so doing, we show that an effective lower bound for this problem is obtained by *heuristically* solving a relaxed problem in which a product can be assigned to multiple lines. As shown in our computational study, this approach generates fairly strong bounds quite rapidly. We also construct a heuristic solution method which efficiently exploits certain properties of the optimal solution. Our computational experience reports the effectiveness of this method under a variety of problem scenarios.

REFERENCES

1. Corneujols, G., G. L. Nemhauser and L. A. Wolsey (1990), "The Uncapacitated Plant Location Problem," in *Discrete Location Theory*, edited by R. L. Francis and P. Mirchandani, John Wiley and Sons, New York, NY.

2. Dyer, M. E., N. Kayal and J. Walker (1984), "A Branch and Bound Algorithm for Solving the Multiple-Choice Knapsack Problem," *Journal of Computational and Applied Mathematics*, Vol. 11, 231–249.

3. Fisher, M. L., R. Jaikumar and L. Van Wassenhove (1986), "A Multiplier Adjustment Method for the Generalized Assignment Problem," *Management Science*, Vol. 32, 1095–1103.

4. Garey, M. R. and D. S. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, NY.

5. Geoffrion, A. M. (1974), "Lagrangean Relaxation for Integer Programming," *Mathematical Programming Study*, Vol. 2, 82–114.

6. Guignard, M. and M. Rosenwein (1989), "An Improved Dual-Based Algorithm for the Generalized Assignment Problem," *Operations Research*, Vol. 37, 658–663.

7. Held, M., P. Wolfe and H. P. Crowder (1974), "Validation of Subgradient Optimization," *Mathematical Programming*, Vol. Vol. 6, 62–88.

8. Martello, S. and P. Toth (1990), *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, New York, NY.

9. Raman, N. and U. S. Palekar (1993), "Product Assignment in Flexible Multilines, Part 1: Single-Stage Systems with Demand Splitting," Working Paper, Bureau of Economic and Business Research, University of Illinois at Urbana-Champaign, Champaign, IL.

10. Ross, G. T. and R. M. Soland (1975), "A Branch and Bound Algorithm for the Generalized Assignment Problem," *Mathematical Programming*, Vol. 8, 91–103.

Table 1

Computational Results – Set I

No.	N	F_1	F_2	Lower Bounds		$Z(MLD)$
				LB2	LB3	
1	15	32	302	1.000	0.969	1.000
2	15	42	131	0.987	0.957	1.000
3	16	47	135	0.947	0.990	1.000
4	14	46	712	0.923	0.952	1.038
5	15	44	570	0.962	0.986	1.000
6	10	15	932	0.935	0.915	1.000
7	16	31	751	0.932	0.957	1.000
8	14	36	417	0.953	0.951	1.000
9	16	40	965	0.998	0.952	1.045
10	15	13	861	0.959	0.997	1.000
11	15	5	340	0.962	0.959	1.000
12	16	26	295	0.968	0.946	1.000
13	17	30	508	0.936	0.997	1.000
14	15	44	312	0.958	0.989	1.000
15	15	48	376	1.000	0.868	1.000
16	12	18	670	0.953	1.000	1.000
17	14	42	639	0.919	0.988	1.000
18	15	48	531	0.959	0.949	1.000
19	14	42	145	0.981	0.998	1.000
20	16	2	685	1.000	0.964	1.000

Table 2

Computational Results – Set 2

| No.	N	F_1	F_2	Lower Bounds		Z(MLD)	No.	N	F_1	F_2	Lower Bounds		Z(MLD)
				LB2	LB3						LB2	LB3	
1	10	15	980	1.000	0.852	1.000	21	21	9	836	0.975	1.000	1.015
2	35	39	189	1.000	0.992	1.012	22	24	32	490	0.993	1.000	1.005
3	21	49	107	1.000	0.911	1.011	23	40	10	100	1.000	0.909	1.028
4	10	11	362	0.960	1.000	1.007	24	45	50	880	1.000	0.914	1.015
5	40	9	110	1.000	0.959	1.029	25	13	48	577	1.000	0.989	1.025
6	35	22	796	0.991	1.000	1.015	26	21	37	455	0.992	1.000	1.009
7	41	47	363	1.000	0.953	1.021	27	14	12	657	0.947	1.000	1.003
8	48	24	951	1.000	0.955	1.016	28	10	29	324	1.000	0.986	1.000
9	30	24	932	0.989	1.000	1.017	29	21	47	733	1.000	0.993	1.024
10	18	17	680	0.984	1.000	1.003	30	28	36	211	1.000	0.975	1.014
11	39	3	859	0.993	1.000	1.008	31	22	29	282	0.999	1.000	1.018
12	37	18	866	1.000	0.972	1.024	32	25	23	559	0.990	1.000	1.025
13	15	7	513	0.940	1.000	1.003	33	36	25	288	1.000	0.904	1.025
14	17	4	904	0.967	1.000	1.000	34	48	47	439	1.000	0.956	1.009
15	12	49	209	1.000	0.993	1.030	35	45	38	312	0.996	1.000	1.022
16	11	9	113	0.985	1.000	1.014	36	35	25	824	1.000	0.972	1.000
17	24	29	679	0.975	1.000	1.005	37	46	10	891	1.000	0.988	1.020
18	24	12	900	0.987	1.000	1.001	38	30	47	479	1.000	0.890	1.000
19	26	10	658	1.000	0.998	1.012	39	34	34	389	1.000	0.867	1.000
20	23	26	963	0.964	1.000	1.002	40	47	49	107	1.000	0.979	1.000

Figure 1 – Graph $\mathcal{G}$

37
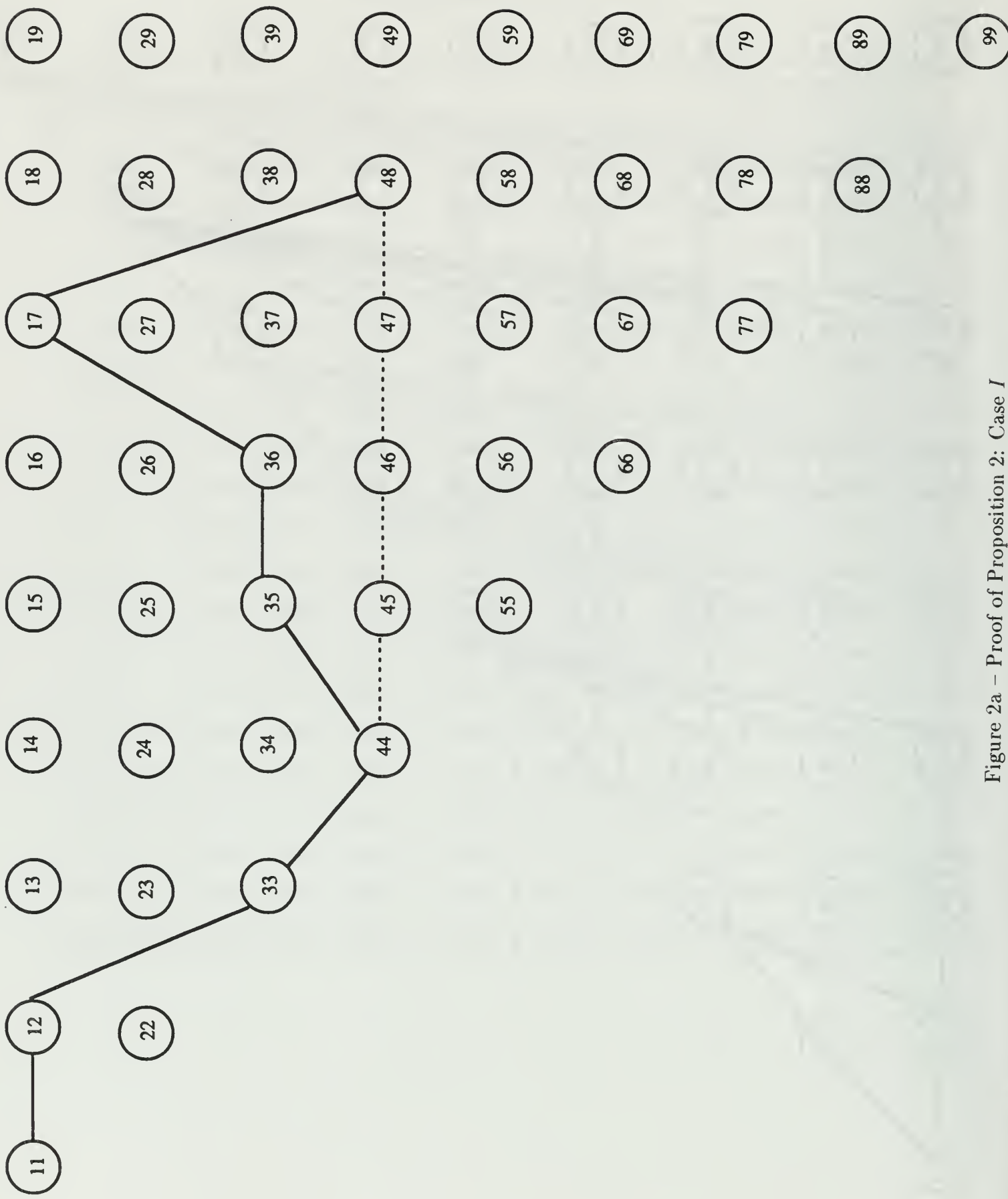
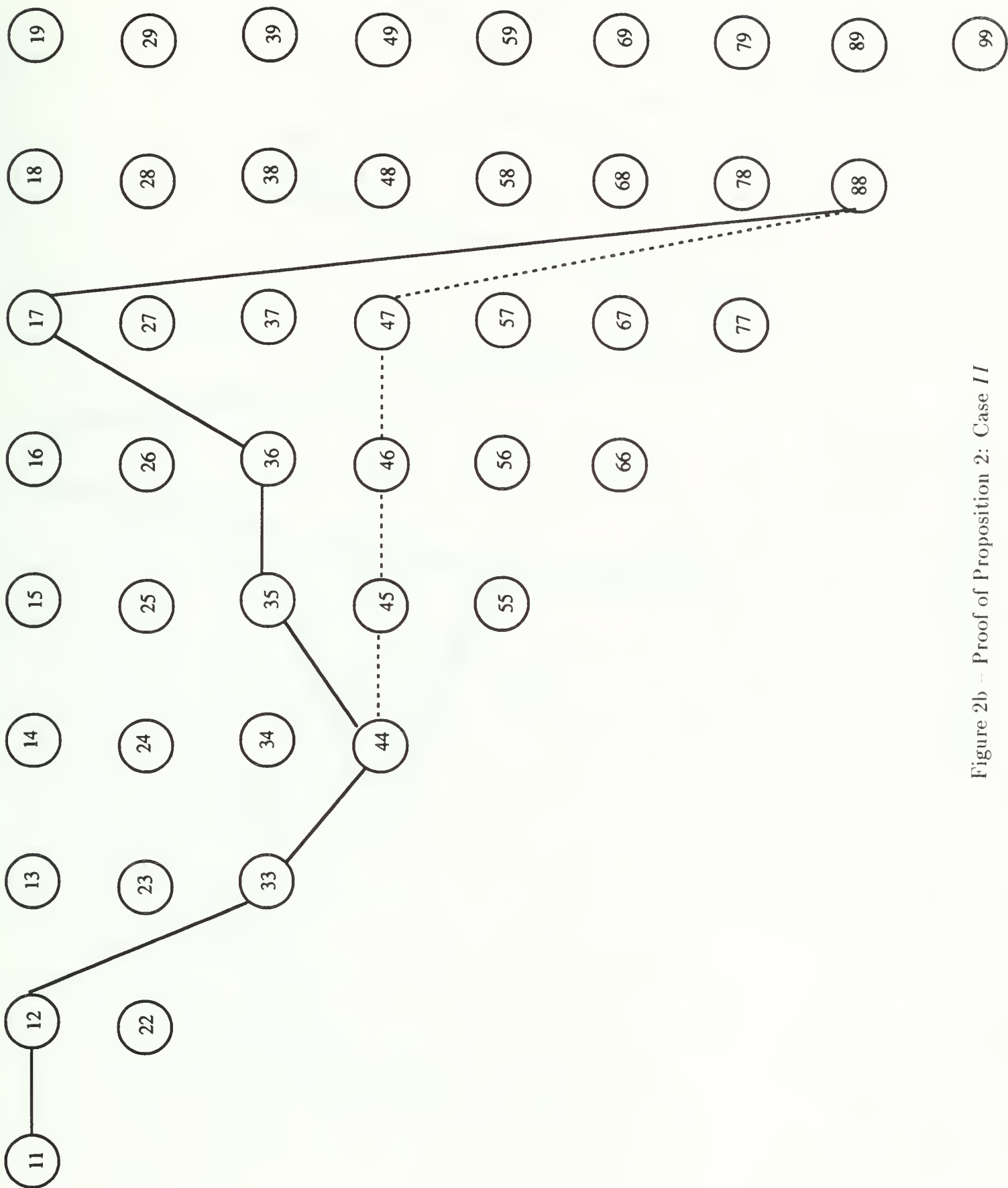Figure 2a – Proof of Proposition 2: Case I
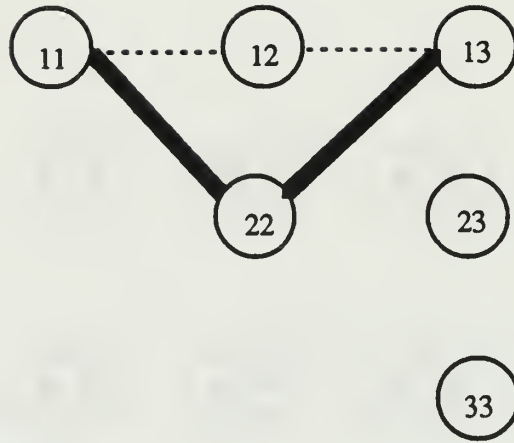
Figure 2b – Proof of Proposition 2: Case II
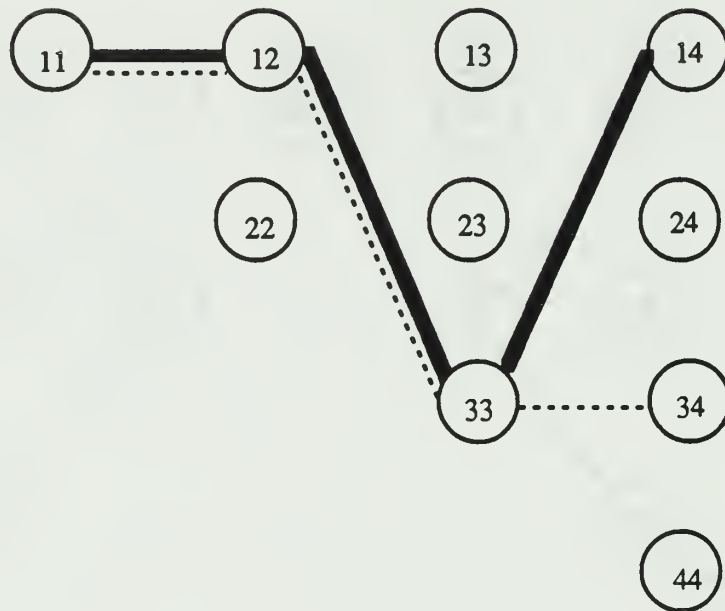
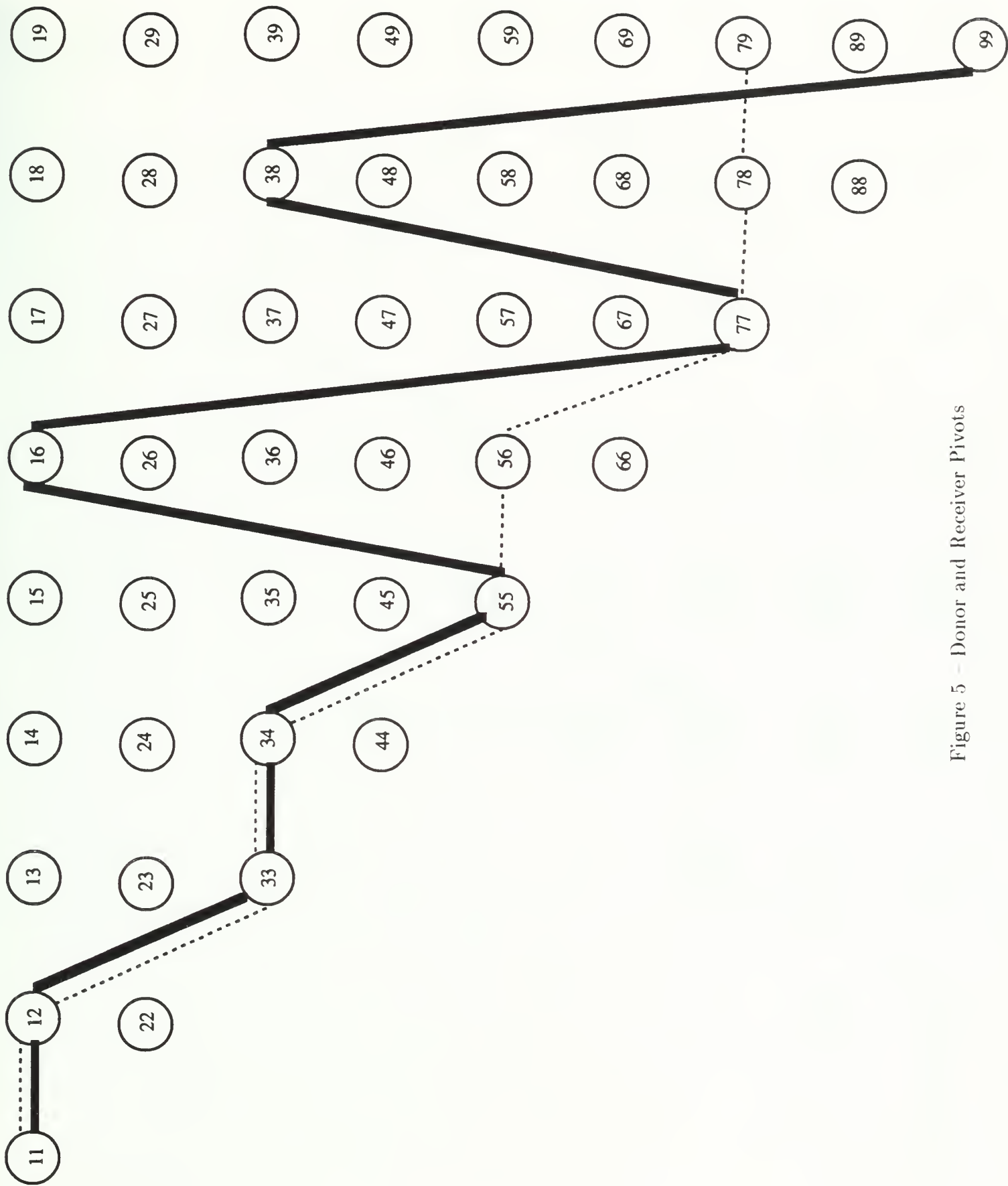Figure 3 – A Type 1 b-arc

Figure 4 – A Type 2 b-arc

40

Figure 5 – Donor and Receiver Pivots
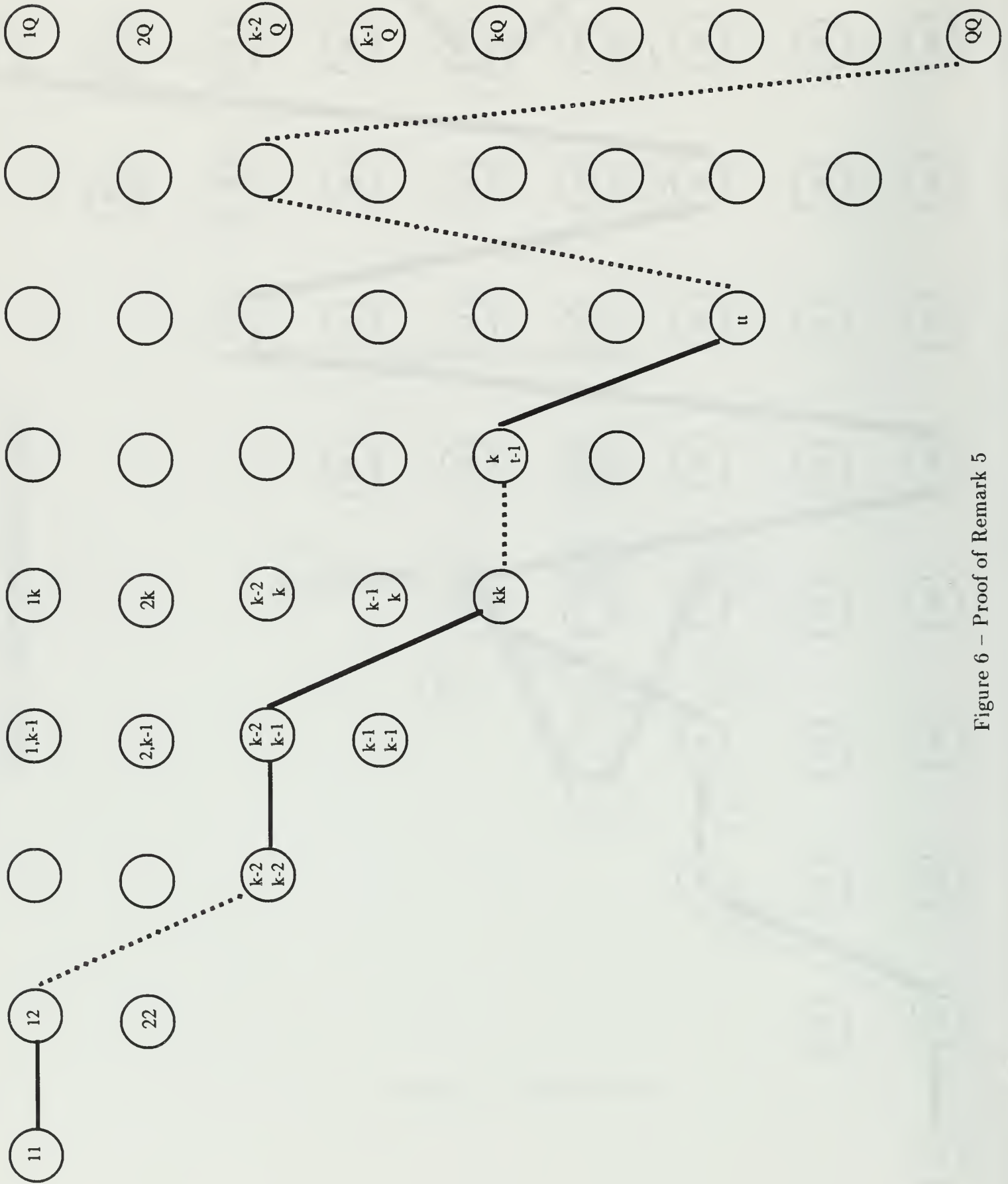
Figure 6 – Proof of Remark 5

42
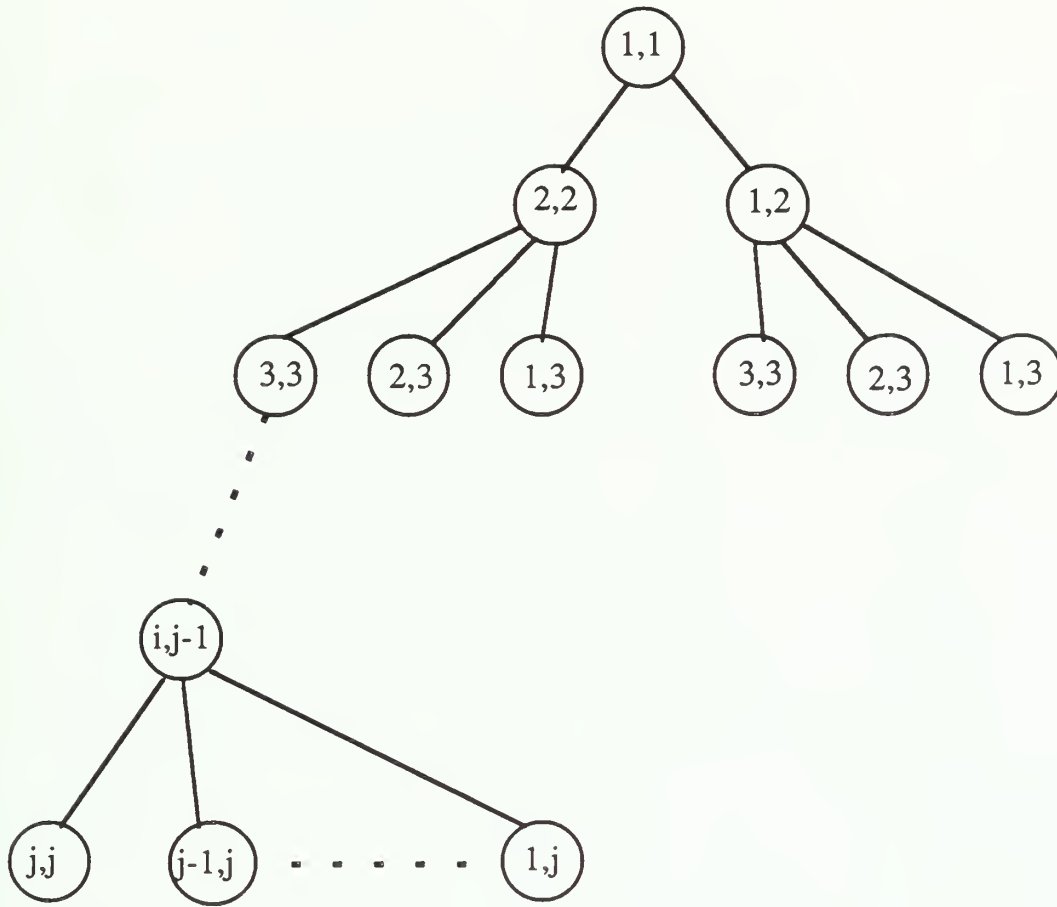
Figure 7 – The Enumeration Tree