.ULATIO AND MIXED INTEGE LINEAR PROGRAMMING MODELS FOR ANALYSIS OF SEMI-AU OMATED MAIL PROCESSING

by

STEVEN DOUGLAS WERT, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University f Texas at Austin

i Parti Fulf llment

f the quirements

for th Degree of

TER OF SCIENCE IN ENGINEERING

HE UNIVE SITY OF TEXAS AT A STIN

December, 1989

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFIT/CI/CIA- 89-184
6a. NAME OF PERFORMING ORGANIZATION AFIT STUDENT AT UNIV OF TEXAS/AUSTIN; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFIT/CIA
6c. ADDRESS (City, State, and ZIP Code)	7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6583
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification) (UNCLASSIFIED)
SIMULATION AND MIXED INTERGER LINEAR PROGRAMMING MODELS FOR ANALYSIS OF SEMI-AUTMATED MAIL PROCESSING

12. PERSONAL AUTHOR(S)
STEVEN DOUGLAS WERT

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
THESIS/~~DISSERTATION~~	FROM ____ TO ____	1989	215

16. SUPPLEMENTARY NOTATION
APPROVED FOR PUBLIC RELEASE IAW AFR 190-1
ERNEST A. HAYGOOD, 1st Lt, USAF
Executive Officer, Civilian Institution Programs

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL ERNEST A. HAYGOOD, 1st Lt, USAF	22b. TELEPHONE (Include Area Code) (513) 255-2259; 22c. OFFICE SYMBOL AFIT/CI

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

AFIT/CI "OVERPRINT"

SIMULATION AND MIXED INTEGER LINEAR PROGRAMMING MODELS FOR ANALYSIS OF SEMI-AUTOMATED MAIL PROCESSING

APPROVED:

Co-supervisors:

[illegible]
Jonathan F. Bard

[illegible]
Thomas A. Feo

Dedicated to my father and *mi esposa bonita*

ACKNOWLEDGMENTS

This document is a significant compendium of model formulations and results from ongoing research. Much of the basis for this work was laid out by Dr. Jonathan Bard, Dr. Thomas Feo, and Dr. Anura deSilva. Their continuous support and sagely advice were the mainstays of this project. Neither rain, nor sleet, nor massive primal degeneracy lessened Dr. Feo's positive commitment to this work and I am most grateful for his help and the benefit of his wisdom.

Sincere appreciation is deserved by Geraldo Veiga, Department of Industrial Engineering and Operations Research, at the University of California, Berkeley, who solved several of our linear programming models using MINOS.

My graduate program in mechanical engineering was fully supported by the Air Force Institute of Technology. That my specialty area became operations research was a fortuitous accident, precipitated in part by Dr. J. Wesley Barnes.

I was indebted to my wife before starting this project. In that regard, I am now operating at a huge deficit. As a token of my appreciation, I have promised not to discuss postal processing at the dinner table.

This thesis was presented for approval on November 12, 1989.

ABSTRACT

SIMULATION AND MIXED INTEGER LINEAR PROGRAMMING MODELS
FOR ANALYSIS OF SEMI-AUTOMATED MAIL PROCESSING

by

STEVEN DOUGLAS WERT, B.S.

SUPERVISING PROFESSORS: JONATHAN F. BARD AND THOMAS A. FEO

Over the last decade, much attention has been focused on the development of automated letter mail processing systems for postal sorting. Optical character readers and bar-code sorters have begun to augment mechanized processing that has been in use since the mid-1960s. Continuing automated mail processing programs are aimed at minimizing growth in labor costs, which at $30.5 billion accounted for 83 percent of the total United States Postal Service (USPS) operating costs in fiscal year 1988. Simulation and mixed integer linear programming (MILP) models are developed in this thesis with the objective of assisting postal managers in designing automated systems for the general mail facilities (GMFs) of the USPS.

[illegible]

The simulation model utilizes a probabilistic structure to channel processed mail between stations. Processing equipment and associated personnel are modelled as resources. The arrival process can accurately model daily input mail profiles and variability, as well as seasonal loads and secular trends such as changes in mail address quality. Such a detailed probabilistic model provides a realistic test of proposed equipment selections and resource schedules for strategic planning and operations management.

A simulation model of a specific medium-sized GMF is constructed using the SLAMII simulation language [illegible] A feasible resource schedule is determined by iterative simulation and this baseline is compared with various "what if" scenarios. The simulation approach is shown to be successful in modelling and evaluating a proposed GMF automation configuration which includes automatically sequencing each carrier's letter mail in delivery order. Nationwide implementation of automatic sequencing has been estimated at a potential annual savings in excess of six billion dollars.

The baseline model results are compared with those from scenarios in which mail volume and address readability are varied from their nominal levels. The simulation results clearly demonstrate the processing system's sensitivity to relatively modest changes in mail volume and mail type profiles. The effects of altering between-station handling and transport times are also investigated. Certain of these results underscore the need to address changes such as inter-process material handling times from a systems model approach.

Formulations of MILP models to optimize the equipment selection and resource schedules for letter mail sorting are also presented. Such deterministic models are proposed as a tool for identifying the types, numbers, and schedules for automated equipment in order to minimize acquisition, operation, and maintenance costs over a postulated operating period. The GMF letter mail processing system can be modelled as a large multistage network where mail is processed during each period and forwarded to the next machine or is retained in inventory.

These general MILP formulations are the basis for generating the systems of constraints and testing the tractability of the models. Experience with solving linear programming relaxations of the MILPs indicates that these multistage networks are solvable, but exhibit degeneracy.

Since implicit enumeration for solution of the MILPs using simplex-based methods could be costly, the postal GMF models are good candidates for solution using an interior point method. A discussion of possible extended capabilities of the MILP models is included.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1. Introduction

Only twenty-five years ago, the United States Postal Department's inveterate mail processing operations had not changed much from eye-shaded clerks hand-sorting letters into pigeonhole racks. As a consequence, over one percent of the national labor force was employed by the Postal Department in 1967 (Tierney, 1981). In the years following the establishment of the United States Postal Service (USPS), mechanization and consolidation of operations have become the keys to major gains in efficiency. By automating, the USPS sought to take advantage of the huge economies of scale. A brief background of the USPS automation developments and mail processing at central postal facilities is provided here for historical perspective. A discussion of the interesting problems expounded by the need to implement automated technologies effectively and efficiently follows in Chapter 2.

1.1 Background

Since the early 1970s, a major objective of the USPS has been to achieve complete fiscal self-sufficiency. In fact, the USPS can be considered to be a huge corporation owned by the federal government and chartered by Congress. Granted relative autonomy by the Postal Reorganization Act (PRA) in 1971, the USPS has been gradually weaned from federal subsidies. Yet, public perception aside, mail postage has paced the Consumer Price Index (CPI). First class mail bears the greatest percentage of the USPS institutional costs (Sherman 1980), but its average annual cost inflation has been about 7.4 percent from 1973 to 1989. The CPI rate has averaged 6.74 percent over the same period. At the same time, overall mail volume has been growing at a staggering rate.

In the fiscal year ending September 1988 over 160 billion pieces of mail were delivered, up 59 percent since 1978 (USPS 1989). The annual volume is expected to exceed 240 billion pieces by the end of this century (ELSAG, 1986). The burgeoning mail volume, coupled with more and more delivery points each year, has challenged the USPS to rapidly implement new automation technologies to limit cost increases.

In the face of expanding requirements, most USPS cost containment strategies can be tied to limiting labor force growth. In 1988, the annual $30.5 billion labor costs represented over 83 percent of the total USPS operating expenses (USPS 1989). The USPS is currently the nation's largest civilian employer at roughly 760,000 employees, but the USPS labor force has remained relatively constant (around 700,000) since 1967 (Tierney 1981). Obviously, mail handling efficiency has significantly increased.

The USPS management strategies to achieve self-sufficiency are subject to Congress, powerful special interest groups, postal unions, and sometimes public outcry. For example, the USPS is obliged to maintain relatively inefficient rural offices which "lose money" (Tierney 1981) and continue door-to-door or curb-delivery routes (Congressional Hearing, July 1985), when there are obviously more efficient alternatives which could provide comparable service. In this constrained managerial environment, much of the relative success in becoming self-supporting must be attributed to automation programs for mail processing operations, primarily at the USPS general mail facilities (GMFs).

The GMF is a processing and distribution center which receives stamped and metered mail from the local area (referred to as originating mail) and mail from other distant GMFs (called destinating mail). From the GMF, sorted destinating mail is sent to local delivery units.

Processed originating mail is dispatched to other destinating GMFs. While automation is also being developed for the sorting of flat mail and parcels, the focus of this effort is on letter mail processing.

1.2 Current USPS Automation Programs

As late as the mid-1960s, most of the nation's mail was sorted by hand. During this period, the Postal Department had developed and implemented what is known as mechanized equipment. These machines mechanized portions of the sorting process by automating the channeling of letters to separate bins or stackers. This represented a significant advance in efficiency over manual sorting, but the USPS still relied on skilled operators to recognize the address field on each letter and key a code sequence at consoles. Much of the mechanized equipment is still in operation in GMFs today.

This mechanization relied on mailers to provide the Zone Improvement Plan (ZIP) code. The original five-digit ZIP code program was initiated in 1963 (Tierney 1981), and the ZIP is still the primary code used for mechanized processing. In 1978, the USPS announced a program to expand the five-digit code to nine digits. Known as ZIP+4, the program was delayed by Congress until 1983. While the five-digit ZIP code identifies the destination of a letter to the delivery area of a local post office, the ZIP+4 identifies the destination to a block face (roughly one side of a city block, a building in the case of apartments, or a firm). The USPS widely advertised the ZIP+4 program and offered incentives for its use, but compliance fell far short of established goals (Congressional Hearing, October 1987). The nine-digit ZIP code concept is important in the implementation of automated mail processing in the GMFs.

The cornerstones of the latest automation programs are the optical character reader (OCR) and the bar-code sorter (BCS). The original single-line OCR machines were procured in 1981 (Congressional Hearing, April 1986) and operated as shown in Figure 1. At a net throughput rate of about 27,500 letters/hour (assuming 65 percent efficiency) these machines read the last line of a machine-printed address field, access a regional directory, and spray a bar-code on each letter. The bar-code or POSTNET code is related to the ZIP code. Letters are then channeled to separate bins depending on their POSTNET code. This single-pass sorting process requires a total of two operators to feed letter mail into the OCR and to empty the bins as they become full.

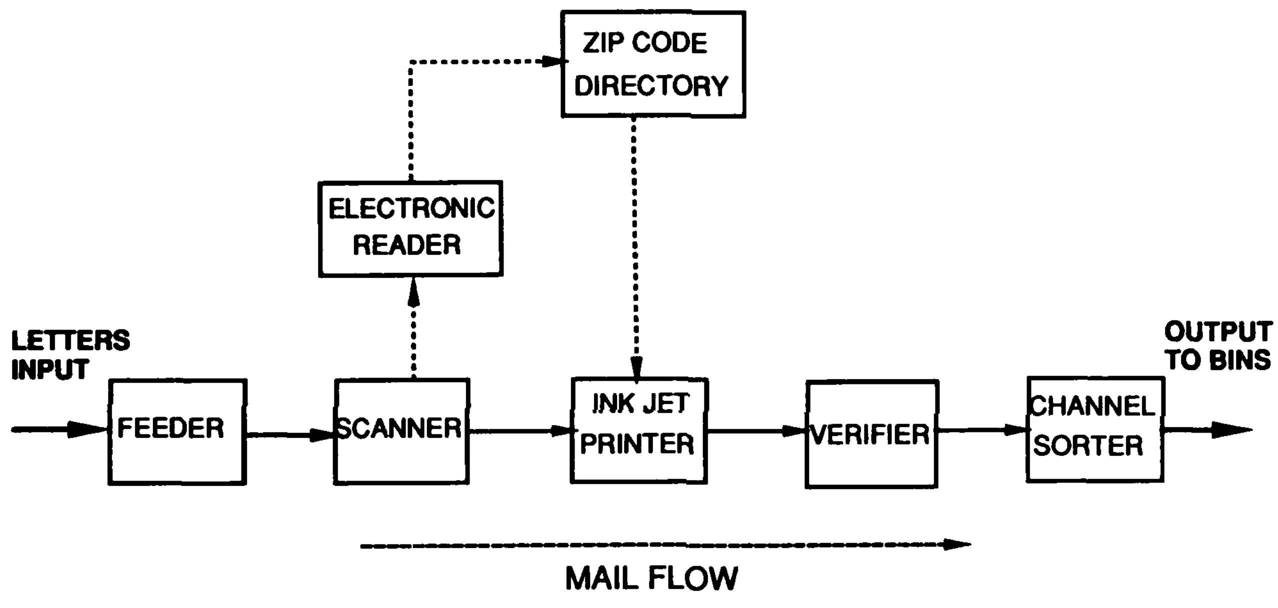


Figure 1. OCR Operation Schematic

The USPS is currently upgrading its inventory of automated sorting equipment by converting single-line OCR machines to multi-line OCRs (MLOCRs). These upgraded machines have access to a larger regional directory of addresses and are capable of reading the complete address field on machine-printed mail. Most mail processed on an MLOCR is sprayed with a bar-code corresponding to the nine-digit ZIP code. Some mail, destinating through another distant GMF, is only five-digit bar-coded, depending on the extent of the MLOCR's regional address directory. A limited amount of handwritten addresses can also be read by the OCRs. Current USPS development efforts are aimed at implementing pattern recognition for certain handwritten address fields and building a national address directory so that virtually all mail can be bar-coded to nine digits at the originating GMF. Currently, a national directory with ZIP+4 level of detail is available at a few GMFs.

The bar-code sorter (BCS) is capable of reading the POSTNET code (much like a bar-code scanner at a grocery store) and sorting letters to separate bins. A BCS processes mail at roughly the same rate as an OCR. Because the degree of separation at the OCRs is limited by the number of OCR bins (typically 44 or 60), the BCS can be used following an OCR to provide a secondary sort on local destinating or dispatch mail. In fact, mail bar-coded to nine digits at an originating GMF can be directly passed through a BCS at the destinating GMF. Therefore, with a national address directory for all MLOCR machines, all OCR-machinable mail can be completely bar-coded in a single pass at the originating GMF and directly sorted by a BCS at the destinating GMF.

1.3 Automated Processing at the GMF

Throughout the 1970s, before the advent of OCR and BCS machines, all physically machinable letter mail was processed by a multi-position letter sorting machine (MPLSM). All other letters were sorted manually. While the mechanized MPLSMs were more efficient than manual sorting, they earned a reputation for a high error rate (Tierney, 1981). And while productivity was increased, the MPLSM was still quite labor-intensive, requiring a total of 17 operators and mail handlers to achieve the nominal processing rate of 34,600 letters per hour (USPS Publication 150, 1988). The MPLSMs are still in use today, but are expected to be displaced by the more efficient automated equipment.

Effectively implementing automation in the postal GMFs presents an interesting and challenging problem. In FY88, 34 percent of all letter mail was processed on automated equipment (USPS, 1989). The remaining mail, which was either not machinable or not OCR-readable, required manual or mechanized processing. Currently, GMFs must incorporate the new automated technologies while meeting the remaining requirements for mechanized and manual processing. A simplified diagram of the flow through the sorting processes in an automated GMF of today is shown in Figure 2.

In this system, all physically machinable, machine-printed letter mail is passed through the OCRs. Some outgoing mail (destinating at another GMF) is sorted sufficiently at the OCR and can be sent directly to the destinating GMF. The remaining OCR output requires a secondary pass through a BCS. Local mail is sent directly to the local delivery units and outgoing mail is sent to the destinating GMFs. Machinable mail which the OCR cannot read must now be processed using the MPLSMs. Mechanized rejects and nonmachinable mail are processed manually.

As Figure 2 indicates, the MPLSMs still play a significant role in letter mail processing. The estimated sorting costs for letter mail processed on automated

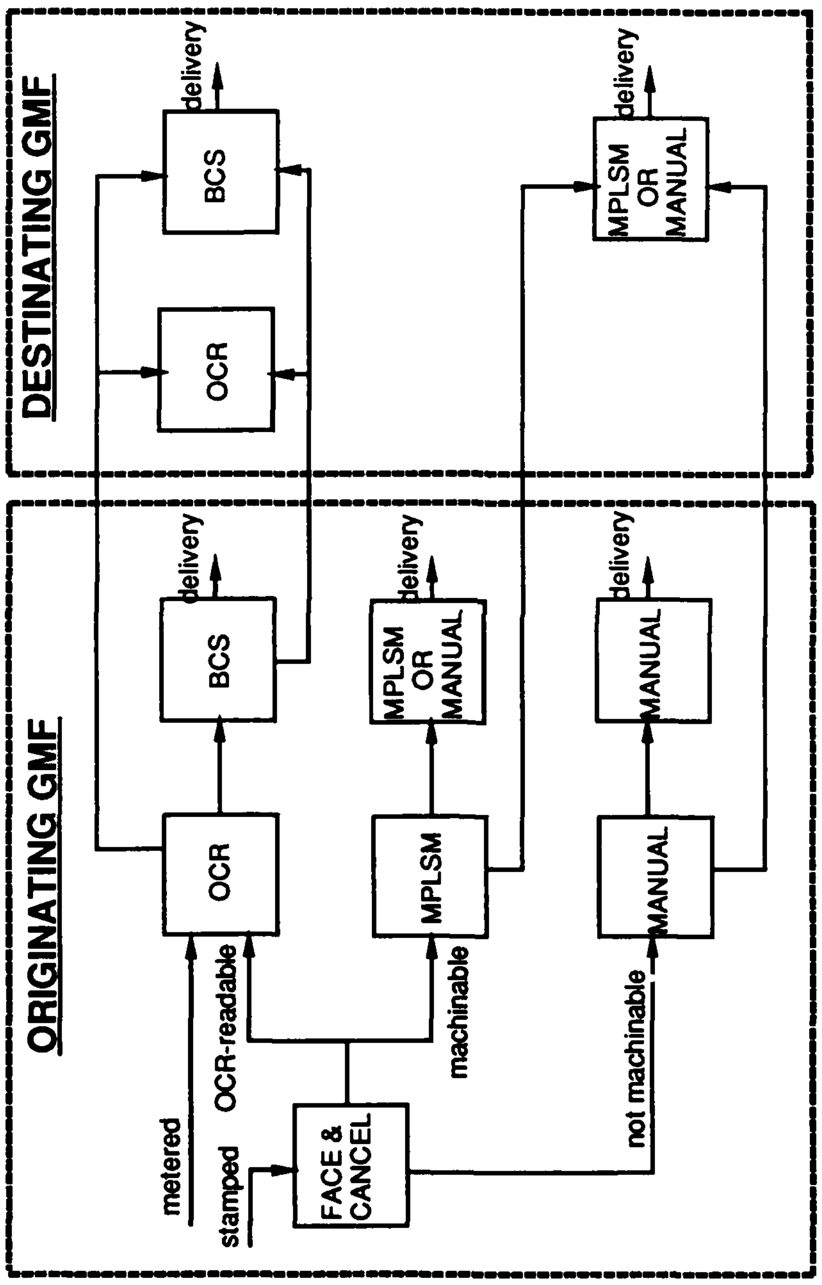


Figure 2. Current GMF Letter Mail Operations

equipment is about 0.3 cents per piece. This is compared with a cost of 1.5 cents per piece for mail processed on mechanized equipment and 3.3 cents per piece for manual mail sorting (USPS, 1989). The USPS hopes to bar-code virtually all letter mail by 1995 (USPS, 1989). Even with the new MLOCRs fully deployed in 1990, other new equipment may be required in order for the USPS to reach that goal. The following two proposed machines are included in the system we have modelled.

The facer-canceller enricher (FCE) is a potential option for processing stamped collection mail. The FCE would serve the same purpose as the existing facer-canceller in finding and cancelling the stamp on each letter, while facing the letters in one direction (based on the stamp location). In addition, the proposed FCE would also be capable of separating out most of the script-addressed letters which the OCRs cannot read. This mail would be sent to the MPLSMs or manual processing. While the FCE is often considered part of proposed future mail processing systems, stamped letter mail may simply be run through the MLOCRs instead. Whether the FCE is economically attractive in the GMF is an interesting equipment selection problem of the type discussed in Chapter 9.

To meet its automated processing goal, the USPS must also develop equipment to bar-code mechanized mail which the OCRs cannot read. These machines, under development at the time this analysis was conducted, are known as remote video encoding systems (RVES). All machinable letter mail which the OCR cannot read would be processed through the RVES. In RVES sorting, each letter is tagged with an identifier code and an image of the address field is stored (USPS, 1988). The stored information is sent to an internal image processing unit or remote video terminal operators for key entry of an extraction code. The entered code is then stored in temporary memory linked with the identifier code for each letter. As the mail piece reaches the sorting stage, its identifier is matched with the POSTNET code in memory and the letter is sprayed with the delivery point bar-code. The letter is then immediately sorted to bins behind the RVES.

The RVES is considered mechanized equipment because its operation would be dependent on operator recognition of the address field. However, the error rate on the RVES is expected to be much lower than that of the MPLSM. Because the RVES's operation is asynchronous, the terminal operators will have more flexibility in response time for each letter.

It is the RVES (or a similar system) which will enable the USPS to achieve high levels of automated flow since the mail stream leaving the RVES is bar-coded. As the image processing becomes more effective, the dependence on operator recognition will be curtailed. While the RVES is still developmental, an Italian firm (ELSAG) has conceived a similar system (without the automated image processing capability) called a video coding desk (VCD). The VCD's operating parameters are used to represent the RVES in our models.

1.4 Delivery Sequencing Concept

In the past, mail had been sorted by five-digit ZIP code to the local delivery unit. More recently, the ZIP+4 code has been used to sort mail to carrier route. However, the nine-digit code contains more information than merely carrier route. Using the ZIP+4 code and finer levels of automated sortation, mail can be prepared in a more ordered fashion prior to dispatch to the delivery carrier. Further efforts in this area are extremely attractive due to the large target of opportunity in reducing carrier labor costs. Delivery carriers comprised 36 percent of the USPS work force for FY88 and spend 25 to 45 percent of their daily work time sequencing route mail to walk order prior to starting their deliveries. Efforts to reduce carrier-in-office activities (sequencing) are directly aimed at a potential multi-billion dollar annual cost reduction.

Currently, automated sorting to carrier route has had no impact on carrier-in-office activity. The further extension of the POSTNET code where necessary to 11 digits, either as a ZIP+6 or a different internal delivery point recognition code, has been proposed. With the additional two digits, the bar code would identify mail piece destination to a specific address and could be used to facilitate automated mail sequencing to carrier route walk order. This automated sequencing would require multiple passes through improved BCS machines with an expanded number of bins.

The delivery point code could be implemented at the GMFs and would not depend on mailers to address letters with a nine-digit or 11-digit ZIP code. Nationwide implementation of such a sequencing configuration has been estimated at a potential annual cost savings of $6.85 billion (ELSAG 1986). This simulation and mathematical programming modelling effort examines the operational scheduling of equipment in a proposed automated GMF processing configuration which includes FCEs, RVESs, and the delivery sequencing concept.

Chapter 2. Implementing Automation

Automated technologies may be the most important factor in the continued progress by the USPS in curbing cost growth. However, analyzing and implementing these new technologies on such a vast scale has created a need for sophisticated tools for postal managers. Simulation and mixed integer linear programming (MILP) models are proposed as comprehensive approaches for improving automated equipment selection and scheduling, and for exploring possible scenarios for future operations.

2.1 General Applications

The USPS general mail facilities are confronted with a variety of both strategic planning and operations management problems which are good applications for simulation and MILP modelling. The expense of the automated equipment and the huge economies of scale make equipment selection and operational resource scheduling important. A single MLOCR is priced at about $650,000 and the RVES cost may exceed $700,000. USPS equipment acquisition decisions apply to over 100 GMFs nationwide. At the projected 240 billion pieces per year, small per piece reductions in operating costs become significant. However, effective and efficient resource level, selection strategy, and scheduling at GMFs are problematic.

Mail processing at a GMF is more complex than indicated in Figure 2. In general, letter mail differs by:

(1) type (collection stamped, collection metered, managed mail, etc.)
(2) class (preferential, non-preferential)
(3) physical size (machinable or non-machinable)
(4) quality of address (machine-printed, handwritten, windowed envelopes)
(5) rate category (presort, bulk, incentive discounts)
(6) degree of processing (originating, partial bar-coded, completed bar-code)

Additionally, all automated and mechanized machines have rejection rates and form side streams of mail which must be processed at a lower level of automation. And letter mail processing is just part of the operation in the GMF.

In addition to the complexity of the core processing system, problem formulation requires estimated costs for the future labor force and for equipment still in development. Input mail volumes and profiles vary daily, weekly, and seasonally. Secular trends of volume growth by mail category depend on both USPS automation programs and outside technological developments (such as electronic mail and networking). Postal managers must also understand and anticipate various political and institutional constraints, as well as labor force and ergonomic considerations. These uncertainties and constraints, and the high level of risk involved, emphasize the need for comprehensive models for assisting the decision-making in resource selection and scheduling.

Heuristic methods have traditionally been applied to USPS equipment selection problems. Operational scheduling is widely accomplished using spreadsheet methods. These approaches may be adequate for approximate results, but cannot address the dynamic interaction of random mail stream arrivals at processing equipment. The representation of such probabilistic and dynamic interactions in a processing network are the major strengths of simulation.

Recent simulation models for facility design and control problems have been constructed by Kiran, Schloffer, and Hawkins (1989) and by Fan and Sackett (1988). Work by Pulat and Pulat (1989) employed simulation as the primary method of investigating throughput of an automated electrical and electronics manufacturing handling system. Taha and Goforth (1988) provide a sketchy description of a simulation of a postal sectional center facility using SIMNET and SLAMII.

From a process view perspective, simulation permits a modular block approach to formulating the complex mail sorting network. Input mail volumes and arrivals can follow a range of probability distributions to address various scenarios. Discrete event capabilities enable the modelling of variable system details and the collection of comprehensive system statistics.

Results of the simulation model include statistics on machine throughput, queue sizes, mail waiting times, proportions of mail volume not meeting critical sequencing and dispatch windows, output volumes, and resource utilizations. Simulation using the SLAMII high-level language offers straightforward modelling, portability, and flexibility for modelling different facilities and sort schemes. While simulation does not guarantee an optimal solution for some objective function, it provides for more robust modelling of the arrival process and complex random interactions of mail being processed through the system.

On the otherhand, our ongoing efforts to develop mixed integer linear programming (MILP) formulations are to enable the optimization of the equipment and processing configuration selection problem. The proposed MILP formulations seek to identify the optimal numbers and schedules of specified equipment types for a given facility with known input streams, operating parameters, and critical dispatch deadlines. The fundamentals of implicit enumeration for integer linear programming appear in Nemhauser and Woolsey (1988) and numerous other references.

An example of an integer linear programming model for an equipment selection problem is provided by Kusiack (1987) using binary purchase decision variables. The advantages and disadvantages to binary variable models are discussed in this document in Chapter 9. Similar work with zero-one modelling for evaluating manufacturing systems is documented by Sarin and Chen (1986). A non-linear cost minimization algorithm with general integer variables has been proposed and solved by Bard and Feo (to appear 1990). While examples of similar mixed integer programming problems for different applications exist, no specific postal processing applications have been found.

Since the mathematical program does not model the probabilistic structure of the system and cannot address the same level of dynamic detail as simulation without becoming unmanageable, simulation analysis will be employed to test the operational feasibility of the MILP solution. The simulation and MILP formulations are separate, but are intended to serve as complementary tools to assist in management decision-making.

2.2 Problem Statement

A GMF letter mail sorting system model was developed to demonstrate simulation's utility for analyses of postal processing operations. A proposed operational configuration for a specific, medium-sized GMF (labelled GMF-A), including its delivery sequencing processing, was selected. GMF-A is somewhat representative of the medium-sized GMFs of the USPS in that all letter mail sorting components are used. Inputs to the simulation model include mail arrival stream volumes and profiles, equipment types and availabilities, operator and mail-handler availabilities, and critical dispatch and sequencing deadlines. Probabilistic stream separation and reject proportions for processing at each machine are specified. The simulation model addresses the questions:

(1) For given numbers, types, and schedules of processing equipment and operators, can the mail be sorted in time to meet critical dispatch times and the sequencing window?

(2) How sensitive is the processing schedule to perturbations in system parameters?

The first question is within the framework of the equipment selection problem. The second question addresses the operations management and planning issues of variations in arrival stream profiles, mail volumes, secular mail type distribution trends, equipment efficiency and maintenance scenarios, and mail handling transfer times between stations.

The following sections introduce a proposed GMF processing configuration and the SLAMII network model of the facility (Chapter 3). SLAMII offers a process modelling view which can be interfaced with discrete event user-written FORTRAN subroutines. Chapter 4 describes the simulation experiment design. Simulation results demonstrating the model's capabilities are given in Chapter 5. Baseline model results which address question (1) above are compared with the results from variations in input values to demonstrate the model's flexibility to address question (2).

The MILP formulations are aimed at providing an optimal solution for the equipment selection and schedule. For given daily mail input volumes and profiles, known operating characteristics, and specified deadlines, these deterministic models seek to minimize the total cost of the processing system over a ten year period. The resultant linear programming (LP) relaxations are fairly large problems which are shown to be solvable using primal and dual simplex. However, an interior point method of solution may be justified given the durations experienced in solving the LP relaxations. Such LP solutions would be required many times over to achieve the optimum integer solution by implicit enumeration.

A general overview of the MILP model formulation approach is given in Chapter 6 along with more detailed information on the notation and constructs employed. The development of the system of linear constraints and the objective function cost factors is provided in Chapter 7. Approaches to solution are discussed in Chapter 8 and include experience with solution of the GMF system models with the integrality constraints relaxed. Further modelling capabilities are discussed in Chapter 9. Chapter 10 documents extensions and conclusions drawn from this effort.

Chapter 3. Simulation Network Model

The proposed GMF-A letter mail sorting configuration is shown in Figure 3. The major elements in the system are shown as arcs for the mail streams and as nodes for the processing stations. There are three main categories of input mail: (1) local collection, (2) managed mail from other GMFs, and (3) large mailer presort. This system is an acyclic processing network which lends itself to process-oriented simulation. The network model formulation and user-written FORTRAN event utilities for the GMF-A simulation model are described in the following sections.

3.1 Network Model Formulation

The diagram in Figure 3 is a simplified view of the system showing the main mail streams and the equipment stations. "Local collection mail" is comprised of locally originating stamped and metered mail, some of which must be processed manually because of physical size. "Incoming/ managed mail," sometimes called managed mail program (MMP) letters, originated at some other GMF and should destinate in the local area. "Large mailer presort" is the business mail which has been presorted and will destinate in the

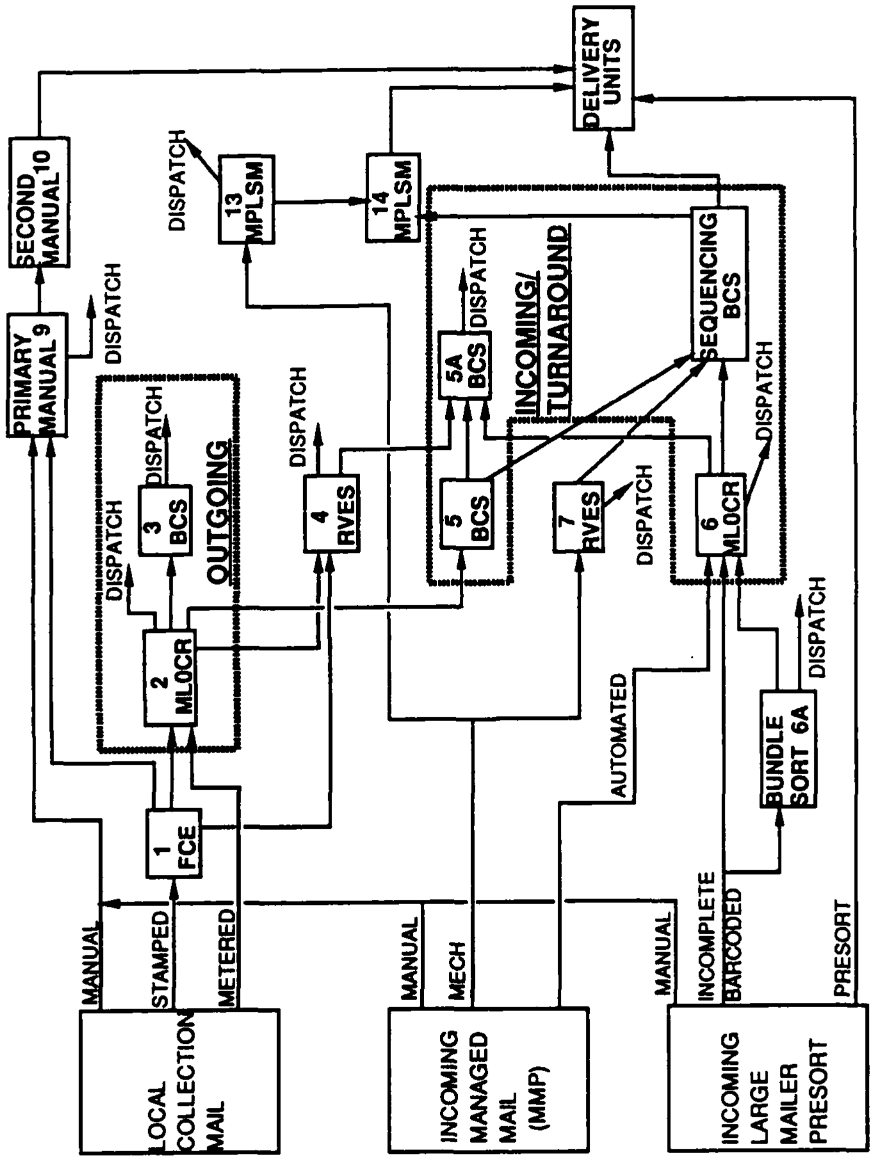


Figure 3. Block Diagram of GMF-A Processing

local area. Automated processing at the GMF is separable into outgoing processing and incoming/turnaround processing, as emphasized in Figure 3. Outgoing OCRs and BCSs process the outgoing portion of the local collection mail. Locally originating collection mail destinating in the local area is called turnaround mail and is processed by the incoming/turnaround equipment as are the destinating and presort mail.

Figure 4 is a more representative of the complexity of the stream separations as mail is input and processed through the system. In this diagram, the blocks are the processing stations and the arcs are the transfers of mail from station to station. For the GMF-A model, 20 input streams and eight categories of output letter mail are defined. The formulation of the simulation model requires the definition of a static framework of queues, servers, and transfer streams to describe the network structure, as well as input stream profiles, branching probabilities, and resource allocations to model the dynamic flow of entities through the system.

In the SLAMII process view network, entities flow through the system with certain attributes attached to them (Pritsker, 1986). In this model, entities are classified as

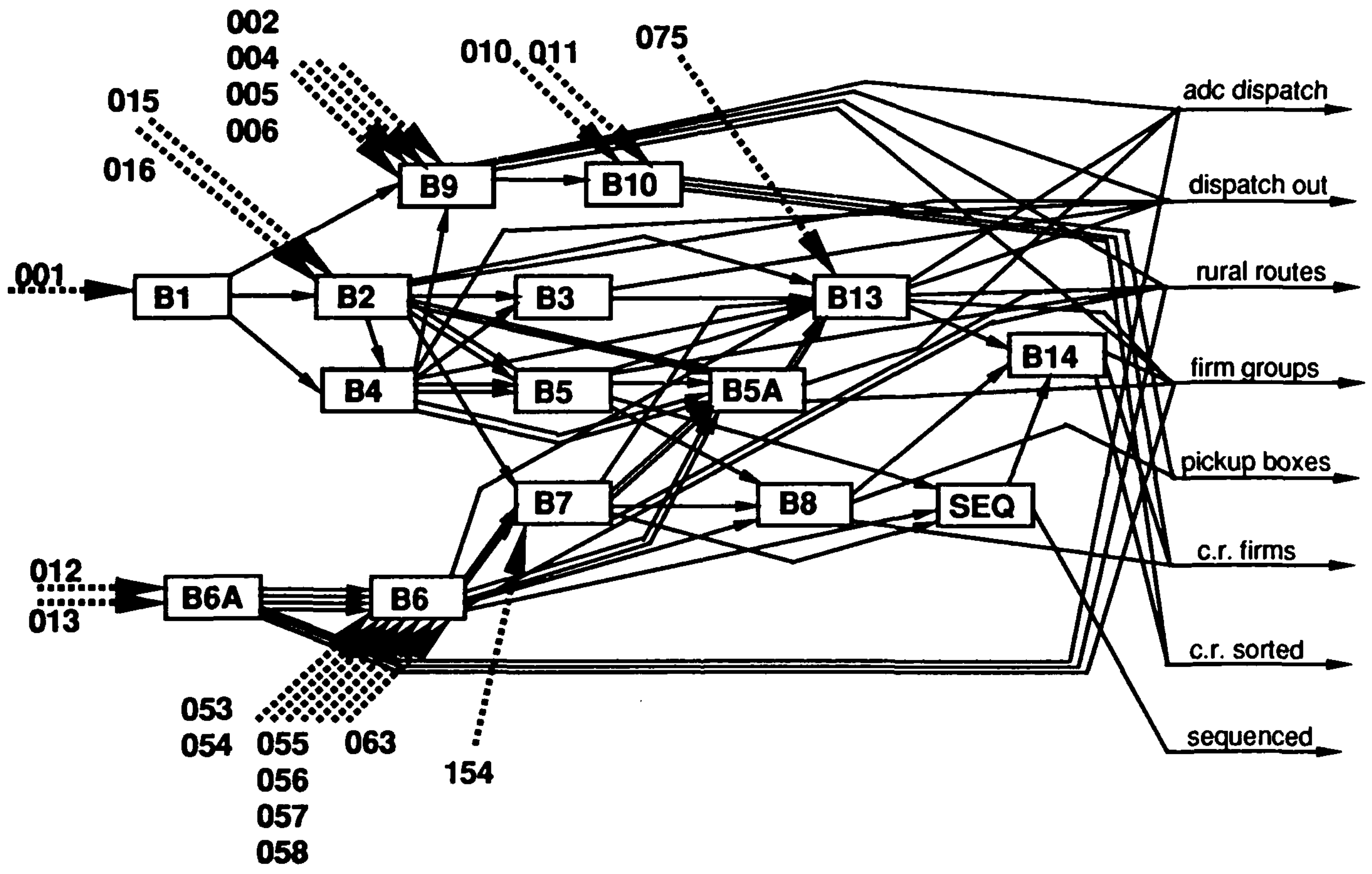


Figure 4. GMF-A Network Diagram

either 1000 piece (1 kp) or 500 piece (1 tray) bundles of letters. The number of letters represented by an entity can be varied. Three attributes are used:

ATRIB(1)	MARKTIME	the time the entity entered the system
ATRIB(2)	MAILSTREAM	the type of letter mail the entity represents
ATRIB(3)	CURRENT_ID	the current mail stream identifier

For example, if a tray of local collection stamped mail entered the system at 330 minutes into the simulation and was then rejected by the FCEs, on its way to manual sorting the entity's attributes would be:

ATRIB(1) = 330
ATRIB(2) = 1 (local collection stamped mail)
ATRIB(3) = 3 (nonenriched FCE reject mail stream)

In order to introduce the SLAMII network elements used in the model, processing stations B1 (FCEs) and B2 (collection mail MLOCRs) are described here.

An entity arriving at block 1 for FCE processing must be local collection stamped mail. Figure 5 shows the portion of the SLAMII network diagram for modelling this processing station. An arriving entity will first reach AWAIT node B1 where it will wait in queue for the next available RESOURCE (an FCE machine) if none are currently available. As one RESOURCE becomes available, the entity

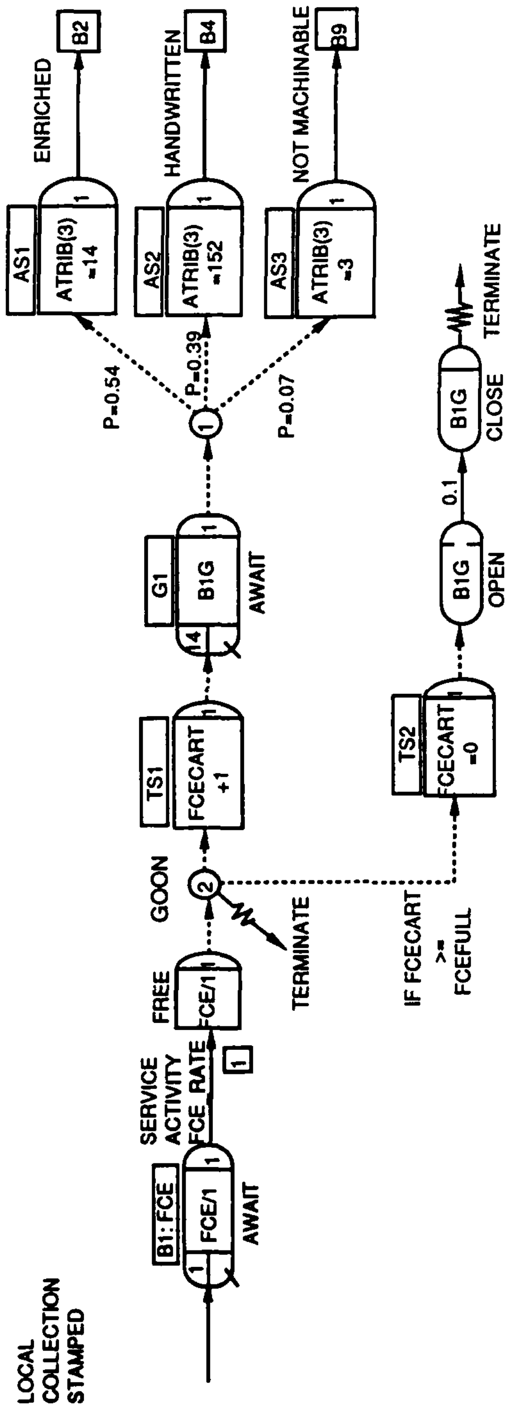


Figure 5. SLAMII Network Diagram of FCE Station

leaves the AWAIT node with the resource committed to it. The entity passes through an activity of duration reflecting the processing rate for the FCE (minutes/tray or minutes/kp). When processing is complete, the single FCE RESOURCE is released at the FREE node for another entity to use. Several entities can be processed at once if several machines are committed to this process.

While the entity proceeds to an ASSIGN node (TS1), a duplicate entity is created by the GOON node. If more than FCEFULL entities have been processed, the duplicate entity branches to TS2. Otherwise, the duplicate is destroyed. At TS1, the counter FCECART, which accounts for the number of entities processed and awaiting release, is incremented by one and the entity is passed to another AWAIT node (G1), to wait for GATE B1G to be opened. When FCEFULL entities have been processed, the GATE is opened for a short time by the duplicate entity allowing all entities in G1 to proceed. This gate structure simulates the delays involved in filling a mail cart with processed mail trays and proceeding when the cart is full.

Once released from G1, the entities follow a probabilistic branching according to the given proportions for the machine. Assumed branching proportions are shown explicitly here, but are actually stored in a SLAMII global variable array. As shown, about 54 percent of the FCE output is sent to the ASSIGN node AS1 where ATRIB(3) is changed to 14. It is now labeled as FCE enriched collection automated mail and is sent to the collection mail MLOCRs (station 2). The entity's ATRIB(2) does not change. The other probabilistic branches represent handwritten mail (ATRIB(3)=152), which is sent to the RVES (39 percent), and facer rejects (ATRIB(3)=3) sent to manual processing (7 percent).

The FCE is the simplest of the processing stations because it has only one input stream so the processing rate and probabilistic branching are independent of the mail type (the entity's attributes). Figure 6 shows the complexity involved when there are three input streams and the nine output streams have probabilistic branching dependent on the entity's mail type. This is representative of the collection MLOCR station (block 2).

The MLOCRs receive the collection automated and collection metered mail, apply the bar-code, and divide the mail into outgoing streams and destinating (incoming or turnaround) streams. Some of the outgoing mail is finalized (i.e. no further processing is needed) at the MLOCRs while

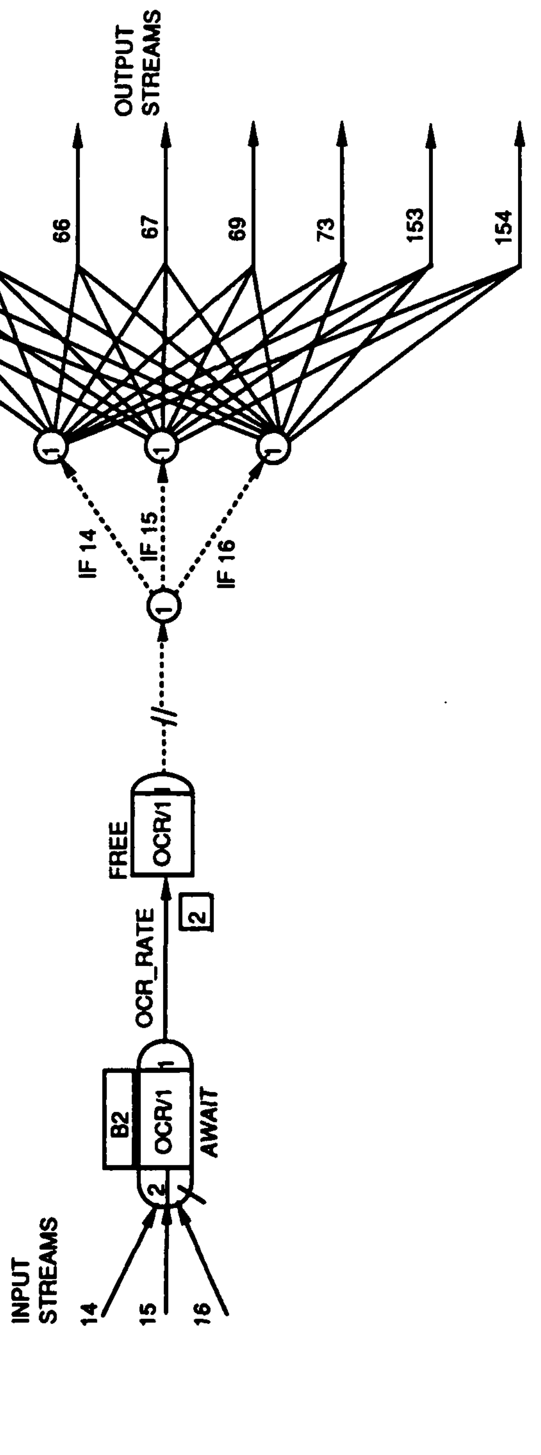


Figure 6. SLAMII Network Diagram of MLOCR Station

the remaining volume must complete a secondary pass through the outgoing BCSs (block 3). Rejects are sent to the mechanized systems (MPLSM, incoming RVES, or outgoing RVES). Assigning a numeric code to identify each stream, the input streams to station 2 are:

Label	Stream Description
(14)	collection automated
(15)	collection metered
(16)	collection metered with ZIP+4

Collection MLOCR output streams are:

(19)	outgoing secondary to BCS for sorting (block 3)
(21)	dispatch outgoing to destinating GMFs
(65)	incoming automated to incoming BCS (block 5)
(66)	incoming automated rural routes to incoming BCS (block 5)
(67)	outgoing to local ADC for secondary sort (block 5A)
(69)	5-digit coded firms mail secondary sort (block 5A)
(73)	primary rejects to MPLSM (block 13)
(153)	partially coded rejects to outgoing RVES (block 4)
(154)	partially coded turnaround rejects to incoming RVES (block 7)

The input stream dependent branching proportions can be shown in matrix form, where the rows are output streams, the columns are input streams, and the elements are the transfer probabilities:

		Input Streams		
		14	15	16
	19	17.5	15.4	15.4
	21	36.5	32.1	32.1
	65	12.0	10.5	11.8
Output	66	2.7	2.4	2.4
Streams	67	0.7	0.6	0.6
	69	0.5	0.5	0.5
	73	3.8	3.4	3.4
	153	23.3	32.5	32.5
	155	3.0	2.6	1.3
		100.0	100.0	100.0

Based on these probabilities, 17.5 percent of the enriched collection automated mail (14) is sent for secondary sort to the outgoing BCSs (block 3).

The three input mail streams must be separated after processing and each stream follows probabilistic branching based on a column of the branching matrix. If these branching proportions are all placed in the SLAMII global variable ARRAY(I,J), the model structure is simplified. The columns of the ARRAY will be related to the entity's ATRIB(3) which is the current mail stream of the entity. Rows of the ARRAY are related to the output streams. Each element in ARRAY(I,J) is the proportion of the entities with ATRIB(3) equal I which will be branched to output stream J. The simulation network can be simplified as shown in Figure 7.

The MLOCR block 2 is more complex than the FCE, but the processing rate of an OCR is not mail stream dependent. For the MPLSM and RVES stations where the processing rate is dependent on the mail type, the streams must be separated prior to processing.

Constructing the complete network for the GMF processing model required building small modules for each machine station (like the diagrams in Figures 4 and 6), and then ensuring that the transfer activities were connected correctly. Processing equipment and manual mail handlers were modelled as resources. The types, numbers, and schedules of resource availabilities are fully described by the process view SLAMII network elements. The scheduling of resources was accomplished using a series of ACTIVITIES and ALTER nodes for each resource type. The entire simulation model could have been accomplished using SLAMII network

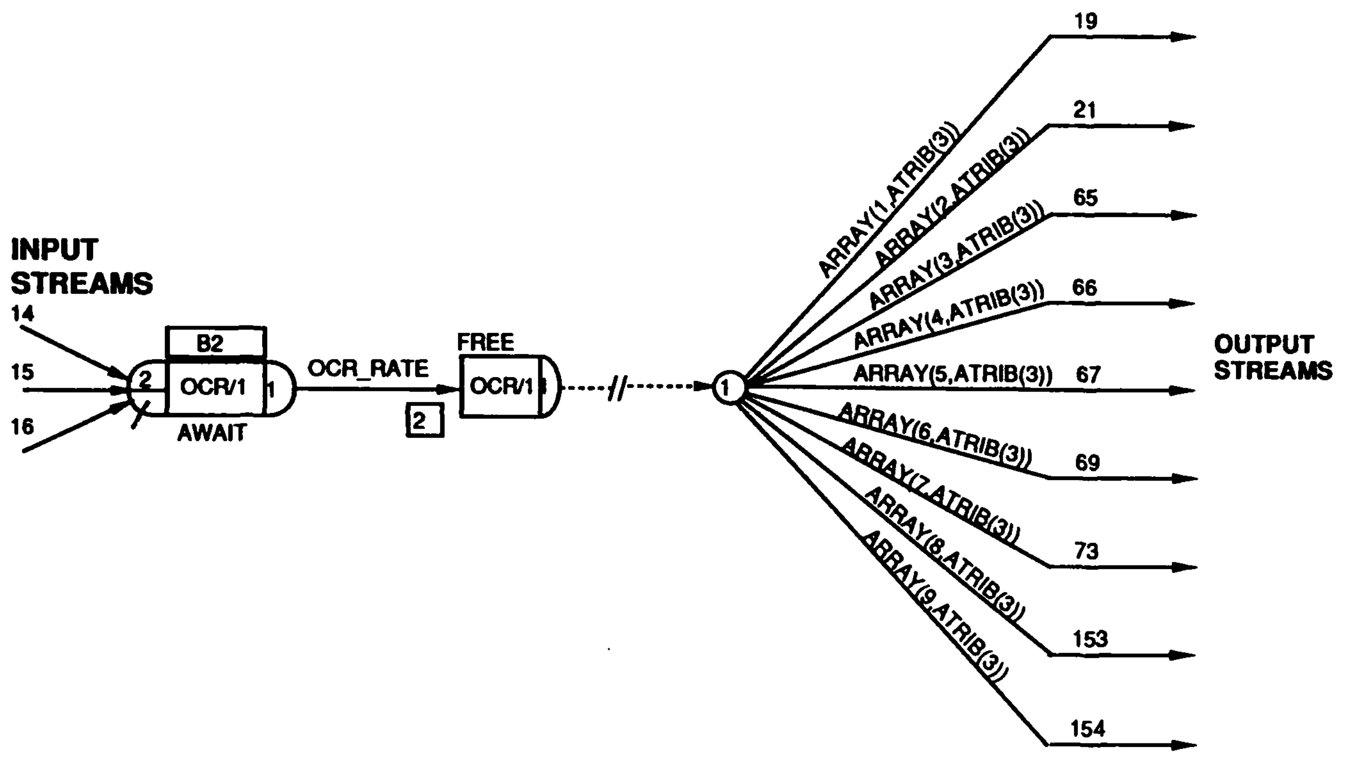


Figure 7. Simplified MLOCR Network Diagram

elements. However, there are many advantages to using SLAM's user-written FORTRAN event capabilities.

3.2 Discrete User-written Event Modelling

The fundamental utility employed here is that discrete user-coded events can be initiated by entities in the processing network. This permits the modeller, while working in the process view, to access SLAM's variables and functions, and to change the variable values or the state of the system. In our postal GMF model, user-coded events are used to:

(1) Initiate SLAMII global variables from local data files (processing rates, batch sizes, branching proportion values);
(2) Input arrival stream volumes at intervals to match arrival profiles;
(3) Generate intermediate SLAMII summary reports;
(4) Clear SLAMII statistical arrays; and
(5) Collect detailed resource utilization statistics.

Figure 8 is a representation of the file structure used for the GMF model. The SLAMII network model (POST.DAT) is actually executed as a subroutine (SLAM) by a main FORTRAN driver (MAIL.FOR). Entity arrivals at EVENT nodes in the network trigger execution of the EVENT subroutines of the main program which access the global variables for initializing, updating, or statistics reporting.

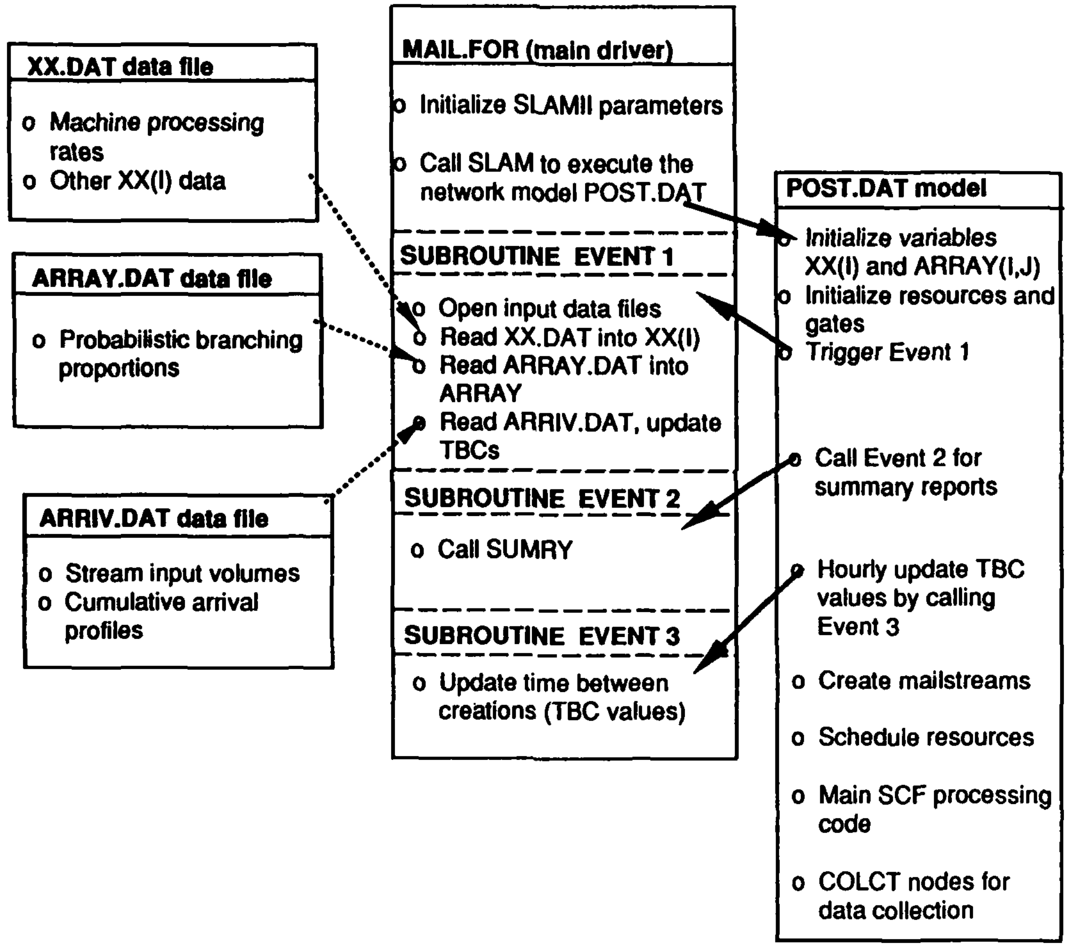

XX.DAT data file
o Machine processing rates
o Other XX(I) data
ARRAY.DAT data file
o Probabilistic branching proportions
ARRIV.DAT data file
o Stream input volumes
o Cumulative arrival profiles
MAIL.FOR (main driver)
o Initialize SLAMII parameters
o Call SLAM to execute the network model POST.DAT
SUBROUTINE EVENT 1
o Open input data files
o Read XX.DAT into XX(I)
o Read ARRAY.DAT into ARRAY
o Read ARRIV.DAT, update TBCs
SUBROUTINE EVENT 2
o Call SUMRY
SUBROUTINE EVENT 3
o Update time between creations (TBC values)
POST.DAT model
o Initialize variables XX(I) and ARRAY(I,J)
o Initialize resources and gates
o Trigger Event 1
o Call Event 2 for summary reports
o Hourly update TBC values by calling Event 3
o Create mailstreams
o Schedule resources
o Main SCF processing code
o COLCT nodes for data collection

Figure 8. File Structure for Simulation Model

The capability to initialize SLAMII global variables facilitates construction and clarity of the model since variable names can be equivalenced to mnemonic labels. More importantly, the ability to update variables provides the capability to modify the model without altering the network structure or recompiling the SLAM network file. In fact, changes to global variables throughout the simulation allow tailoring of input streams to match a given arrival profile or to simulate different mail volumes each day.

SLAM's statistic summary reports provide comprehensive information on machine throughput, mail waiting times, times of arrival, and times in system. Certain information in the summary reports, resource utilizations for example, may not provide the detail needed. The discrete event capability allows access to the SLAM statistical functions such as the average resource utilizations, RRAVG(I), and periods of availability, RRPRD(I). The variable arrays RNUSE(I) and RAVAIL(I), the current resource utilization and availability of resource I, can also be read from user-written subroutines. Periodic access to this information permits computation of detailed resource utilization profiles. Complete listings of the computer codes MAIL.FOR and

POST.DAT, as well as the three input data files, are included in Appendix A.

Chapter 4. Experimental Design

The application of simulation for analysis of processing systems frequently involves the comparison of the present method of operation (PMO) versus various proposed future methods of operation (FMO). The approach here is quite similar, but the baseline is a potential future operation of interest that assesses the adequacy of a proposed equipment schedule. This baseline model provides the capability for comparison of several "what if" scenarios (FMO).

4.1 Baseline

The baseline scenario is the proposed 1990 GMF-A processing configuration with implemented automation including carrier route sequencing BCSs (ELSAG, 1986). As previously discussed, this layout differs from current GMF sorting by employing FCEs for enriching the stamped local collection mail, RVES to bar-code non-OCR readable letters, and improved BCSs (BCS2) for the walk order sequencing scheme. Estimates for the daily input mail stream volumes and profiles are based on actual volumes and recent trends for the GMF-A. Nominal input mail stream volumes are shown in Table 1 with the corresponding daily arrival profiles illustrated in Figure 9. A total of 3707 kp of letter mail is forecast to be processed in this system (an additional 448 kp of local destinating presort mail will also pass through the GMF-A in 1990, but is sufficiently sorted by the mailer and does not require processing).

Table 2 provides details of the six resource types required. The baseline model examines whether the mail can be processed to meet the fixed sequencing and dispatch deadlines for given input volumes, arrival profiles, probabilistic branching, and specified resource scheduling and processing rates. Since the resource schedule is predetermined, the acquisition costs, maintenance costs, operator requirements, and floorspace issues can be addressed a priori.

The sequencing BCS operation is not explicitly modelled in the simulation but is based on a two pass sort sequence involving N_{cg} carrier groups. The choice of N_{cg} is made by preliminary presimulation calculations to yield a starting value, followed by one or more iterations using a separate simulation process (deSilva et al, to appear 1990). Based on this study, with a choice of N_{cg}=36, for the assumed mail volume at GMF-A, the proposed sequencing BCS2s

Table 1: Arriving Stream Volumes and Descriptions

Stream Number	Stream ID	Description	Daily Volume (kp)	Daily Profile*
1	(001)	Local Collection Stamped	800	C
2	(015)	Collection Metered	692	C
3	(016)	Collection Metered ZIP+4	52	C
4	(012)	Presort Bundles	287	M
5	(013)	Non-pref Presort Bundles	163	B
6	(053)	Managed Incomplete #1	489	M
7	(054)	Managed Incomplete #2	29	M
8	(055)	Incoming 3-D Mech	19	M
9	(056)	Non-pref Incoming 3-D Mech	319	B
10	(057)	Incoming 3-D Mech ZIP+4	2	M
11	(058)	Non-pref 3-D Mech ZIP+4	28	B
12	(063)	Managed Bar-coded	101	M
13	(154)	Managed Mech RVES	475	M
14	(002)	Local Collection Manual	56	C
15	(004)	Managed Manual	83	M
16	(005)	Presort 3-D Manual	1	M
17	(006)	Presort 3-D Manual Non-pref	20	B
18	(010)	Presort 5-D Manual	15	M
19	(009)	Presort 5-D Manual Non-pref	9	B
20	(075)	Managed Mech MPLSM	67	M

* see Figure 9 for daily cumulative arrival profiles

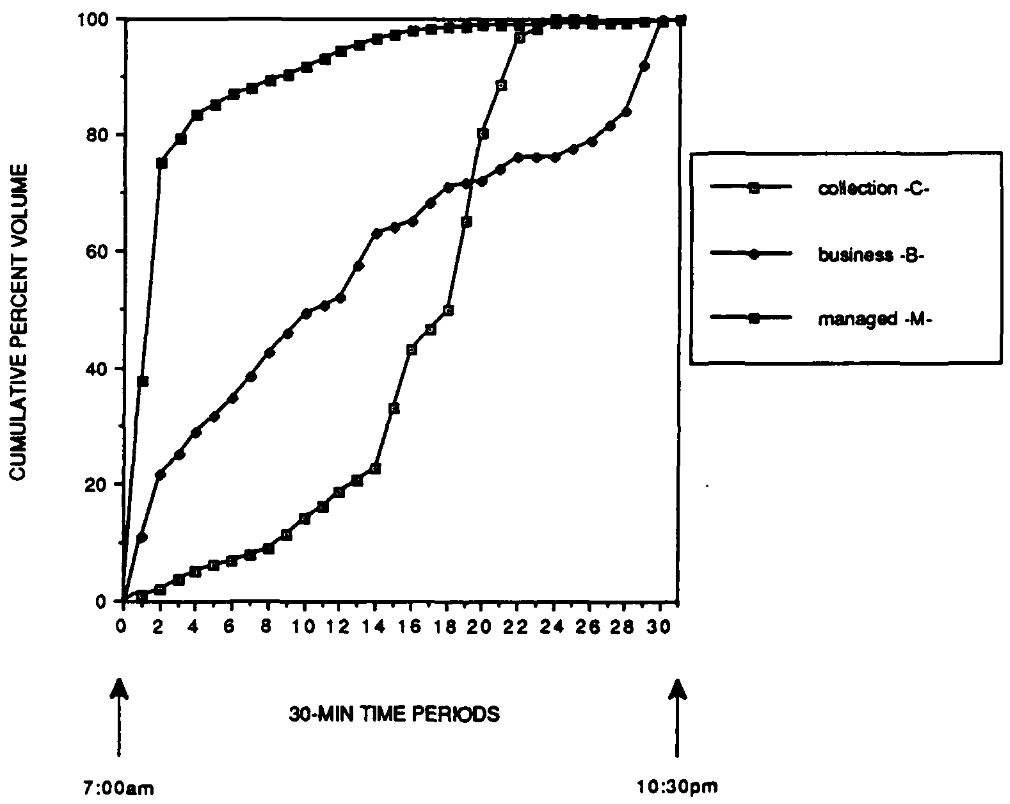


Figure 9. GMF-A Arrival Stream Profiles

Table 2: Processing Resource Parameters.

Type of Resource	Number of Operators	Acquisition Cost ($K)	Annual Maint. Cost($K)	Daily Maint. Hours	Required Floorspace* (Sq ft)
FCE	2	400	40	4	1995
MLOCR	2	650	65	4	3349
BCS1	2	175	17.5	4	2993
RVES	18	700	70	4	3848
MPLSM	17	400	40	8	3420
Manual	1	-	-	-	-

* Floorspace requirements include operating, staging, handling, and aisle space.

require two additional 16-bin modules and increased memory compared with standard 96-bin BCSs. Obtaining the machine parameters for achieving the finer levels of sortation is not straightforward. The minimum number of sequencing BCSs (N_m) can be obtained from:

$$N_m = N_L / (W \cdot R_{BCS})$$

where	N_L	the daily number of letters to be sequenced (roughly 1,600,000)
	W	the processing window for sequencing (hours)
	R_{BCS}	nominal BCS processing rate (27,000 letters/hour)

The minimum number of bins per machine (N_b) is a function of the total number of delivery points (N_d), the number of carrier groups (N_{cg}), and the number of sequencing passes (p), as described by the relation

$$N_b = (N_d / N_{cg})^{1/p}$$

For GMF-A, N_d is estimated at 480,000 delivery points per day in 1990.

At this point, some parametric dimensioning is required since W, N_{cg}, and p must be selected. We follow a previous study (ELSAG, 1986) in selecting W = 5 hours, N_{cg} = 36, and p = 2. Hence, the minimum number of BCSs would be

12 and the minimum number of bins per machine would be 116. This scenario is further complicated, since each machine must perform multiple sort schemes. This necessitates additional machines to compensate for the sweeping time when changing sort schemes. Additionally, since bins are available in 16-bin modules, each BCS would have 128 bins.

For the baseline operation at GMF-A, the proposed sequencing operation is based on:

Number of passes (p)	2
Total number of BCSs (N_m)	18
Number of carrier groups (N_{cg})	36
Number of bins per machine	128
Sort schemes per BCS	2
Total sort time	3 hr 10 min
Total sequencing window	5 hr

These parameters are used to set the critical entry time for the sequencing operation window. The simulation objective is to test whether letter mail meets this deadline for the given inputs and resource schedules.

Table 3 shows more detailed information on the processing at each of the sort stations (i.e. the nodes in Figure 6). The operating window durations are limited by processing deadlines or block maintenance time requirements. Processing rates can be simply constant (as for the FCE and OCR) or may vary with the type of mail being processed. The MPLSMs, for example, process each input stream at a

Table 3: Processing Station Equipment Operation

Processing Block	Description	Processing Rate (letters/hr)	Available Operating Window
B1	FCE for collection	21,000	0700 - 2400
B2	Collection MLOCRs	27,500	0700 - 2400
B3	Outgoing BCS Secondary	28,000	1100 - 2400
B4	Collection RVES	13,440-54,880[1]	0730 - 2400
B5	Turnaround BCS	28,000	1100 - 2400
B5A	Firms/ADC Secondary BCS	28,000	1100 - 0530
B6A	Manual Bundle Sort	12,500	0700 - 2400
B6	Incoming MLOCRs	27,500	0700 - 2400
B7	Incoming RVES	13,440-33,600[1]	0700 - 2400
B8	Firms Sort BCS	28,000	1100 - 0630
B9	Primary Manual	895	0700 - 2400
B10	Secondary Manual	895	0700 - 0530
B13	Primary MPLSM	28,080-32,700[2]	1500 - 0530
B14	Secondary MPLSM	27,800	1500 - 0630

[1] Processing rate for the RVES depends on the mail type of the entity being processed.

[2] Rate for the MPLSM depends on incoming mail stream type.

differing rate. The RVES processing rate is dependent on the actualletter being sorted (effectively rate dependence on the output stream).

The first step in the experiment was to construct and execute the network model and perform verification tests. Since the mail volumes are fixed and the probabilistic branching proportions known, an estimate of the mail volume through each station is possible and provides a verification check for the baseline model flows. Equipment scheduling based on pre-simulation calculations was then tested. The key indicators in evaluating the adequacy of the resource schedule are entity times of completion and the number of entities remaining in the system to be processed the next day. Detailed resource utilization statistics were periodically collected to iteratively reschedule resources in order to achieve a feasible solution for the baseline model.

4.2 "What if?" Scenarios

The baseline model described above provides a vehicle for exploring several important postal processing issues. Three fundamental areas of interest were examined; (1) overall daily mail volume growth; (2) a secular trend to increasingly automated mail; and, (3) the impact of between-station transfer times. The key statistics for comparison with the baseline are entity times in system, completion times, and the number of entities remaining in the system for more than one day.

Evaluating the system's sensitivity, to either increasing volume or more automated mail, addresses strategic planning issues for equipment selection and scheduling decisions. Our first scenarios examine the impact of overall daily mail volume increases of five percent (to 3892 kp) and 10 percent (to 4078 kp). On the otherhand, increasing percentages of OCR-readable and bar-coded mail by 10 to 20 percent reflect trends which must be anticipated by the postal manager if the USPS is to reach its goal of automating all letter mail. Our second set of scenarios increase the percent automated mail in these areas by:

(1) Increasing the probabilistic branching of local collection stamped enriched mail from the FCE by 10 and 20 percent; and
(2) Increasing the percent of managed mail which enters the GMF completely bar-coded by 10 and 20 percent.

Another investigation, this of various between-station delay times, emphasizes simulation's capacity to model realistic processing systems, including material handling delays. In the baseline, batch mail is modelled to be representative of the transport of mail in carts. The delays attributable to mail cart movement are simulated as uniformly distributed 10 to 15 minute delay times between stations. This last scenario examines the impact of neglecting transfer delay times, or of modelling these delays as exponentially distributed with mean 15 minutes (and, of course, greater variance than the uniform case). These scenarios are not an examination of observed transfer times, but an evaluation of the system's sensitivity to mail handling delays.

Chapter 5. Simulation Results

One of the important outcomes of simulating complex operations is the insight gained in the effort of constructing the model. Another is the ability to perturb the model and examine the impact of transient or steady changes on a system which one could ill afford to actually disturb. Postal GMFs fit the above criteria eminently. In the following presentation of simulation results, presimulation estimates are first used to verify the network structure of the model. A successful resource schedule for the baseline model is achieved by successive iteration. Results from "what if" scenarios are summarized in 5.2.

5.1 Baseline Model Results

Presimulation analysis was employed to determine the expected mail flow through each processing station for comparison with observed values from the model. By giving the sorting processes unlimited resources (infinite processing rates) and neglecting mail batching and delays, the daily mail throughput can be efficiently simulated for verification of the overall network structure. For the nominal stream volumes (Table 1) and specified arrival profiles (Figure 9), a daily total of 3707 kp (entities) are input for the simulation. The total throughput volumes for each processing station for three daily simulation runs are compared with the deterministic presimulation estimates in Table 4. A comparison of output letter volumes by destinating category is provided as Table 5. As these results demonstrate, the use of probabilistic branching in the network system incorporates a randomness which would be observed in practice.

Presimulation estimates of the resource requirements are more troublesome. Table 3 prescribes the bounds for the possible operating windows for each of the processing stations in the model. The operating windows are limited by: (1) the sequencing deadline; (2) dispatch deadlines on outgoing mail; and (3) daily routine maintenance requirements. Given the total process volume and the operating window for an equipment station, a lower bound resource requirement can be computed as throughput volume divided by the product of the processing rate times the operating time. Such a value, however, may be infeasible for the system because of the implicit assumption of constant and deterministic rate of mail arrival at each station. Moreover, it may be desirable for the same

Table 4: Simulation Throughput Results

Processing Block		Throughput Volumes (kp) Day 1	Day 2	Day 3	Presimulation Estimate
B1	FCE	800	800	800	800
B2	MLOCR	1182	1176	1178	1176
B3	BCS	327	336	318	336
B4	RVES	630	670	649	654
B5	BCS	310	297	307	308
B5A	BCS	97	88	95	100
B6A	Bundles	450	450	450	450
B6	MLOCR	1349	1346	1351	1350
B7	RVES	831	851	841	848
B8	BCS	192	192	196	192
B9	Prim.Manual	231	235	225	229
B10	Sec. Manual	131	127	126	134
B13	Prim.MPLSM	210	231	214	215
B14	Sec. MPLSM	170	160	136	160
SEQ	First Pass	1599	1569	1576	1559

Table 5: Simulation Destinating Volumes Result

Output Mail Category	------ Daily Output Volumes (kp) ------- Day 1	Day 2	Day 3	Presimulation Estimate
Outgoing Dispatch	1187	1205	1185	1185
Dispatch to ADCs	44	50	41	48
Rural Routes	376	355	412	383
5D Firms Dispatch	79	83	96	86
Firms Pickup/Boxes	175	209	186	208
Firms in Carrier Route	7	11	8	12
Sorted Carrier Route	277	255	233	265
Sequenced Mail	1562	1539	1546	1520

resources to be shared by several processes and scheduling of machine reallocations is required.

An investigation of equipment scheduling for two resource types is provided here for comparison. As shown in Table 4, the two MLOCR stations (B2 and B6 as shown in Figure 4) process a nominal daily volume of 2526 kp of letter mail daily. From Table 3, the operating windows for the two stations are similar, so a lower bound value for the number of OCRs required is:

$$\frac{2526 \text{ kp daily mail}}{27.5 \, \frac{\text{kp/hour}}{\text{machine}} \cdot 16.5 \text{ hours}} = 5.567 \text{ OCRs}$$

However, this estimate assumes that the machines do not suffer starvation and that fractions of machine hours can be reallocated without penalty. Even if these conditions were met, since the simulation processing volume is not deterministic, this presimulation estimate could only be approximate.

Another problem is the difficulty in determining the arrival profile (entities versus time) for the processing stations. For example, the FCE station is required to process a total of 800 kp of mail over a maximum conceivable operating window of 17 hours. Given that the net processing rate of the FCE is approximated by 10.5 minutes per kp, then only 2.24 machines may be required. However, the simulation results will show that three FCEs are not adequate. The reasons involve the sequencing deadline and the arrival profile of collection stamped mail (see Figure 9). Entities processed by the FCE proceed through the system on a variety of paths depending on their attributes and the probabilistic branching. Some of these entities must be processed by several machines before reaching the sequencing BCSs. Therefore, some mail processed during the last part of the FCE operating window will not meet the critical entry time for sequencing.

Moreover, the local collection stamped arrival profile shows that over 77 percent of the total daily volume arrives at the FCE after the first seven operating hours. Therefore, 77.3 percent of the total daily volume must be processed after the seventh hour but before the last few periods before the sequencing deadline. If the processing window is effectively limited to 618 kp (77.3 percent of 800 kp) in the period from 1400 to 2200 hours, 3.681 FCEs are needed. The simulation results show that four FCEs are adequate to cover the nominal daily processing requirements.

This type of reasoning becomes more difficult downstream from the mail entry points as the arrival profiles are no longer exogenous and predictable. It is quite evident that resource scheduling at one station is dependent on the resource schedules at other processing stations. Achieving a feasible schedule for the baseline model was an iterative process of adjusting the resource schedule, executing the model, and examining the results. The adequacy of the schedule is determined from statistics on entities not completed processing or missing the deadlines. Detailed resource utilization and queue length statistics provide indications of possible directions for improvement.

Results for one type of resource are shown here as an example. For a particular unsuccessful schedule, over three successive days of the simulation many entities remained in queue for processing at the incoming MLOCRs. Detailed resource utilizations for one day, obtained from the event subroutine output, are plotted in Figures 10 and 11. As these figures show, the incoming OCRs are fully utilized before the sequencing window deadline (at 2400 hours). The collection MLOCRs, however, are not fully utilized and may be reallocated for incoming mail processing.

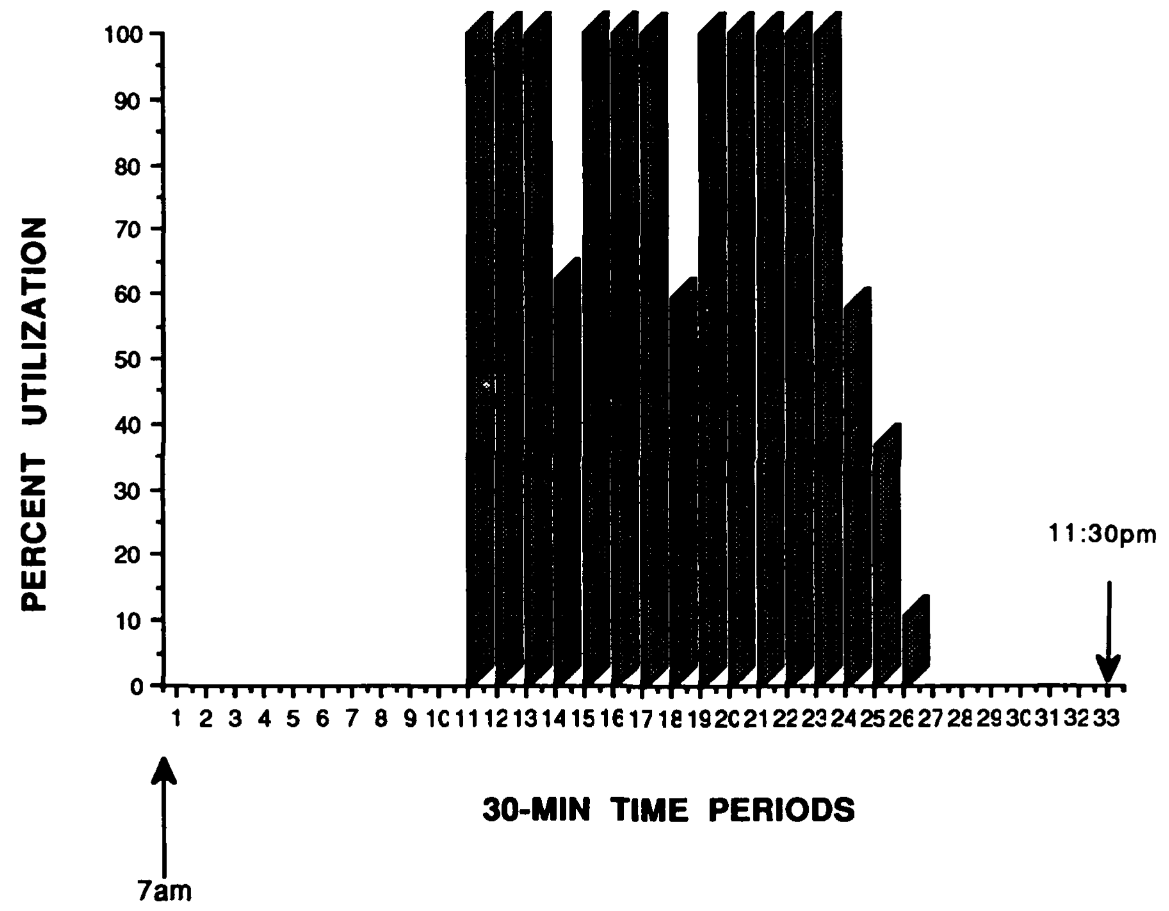


Figure 10. Collection OCR Utilization Results

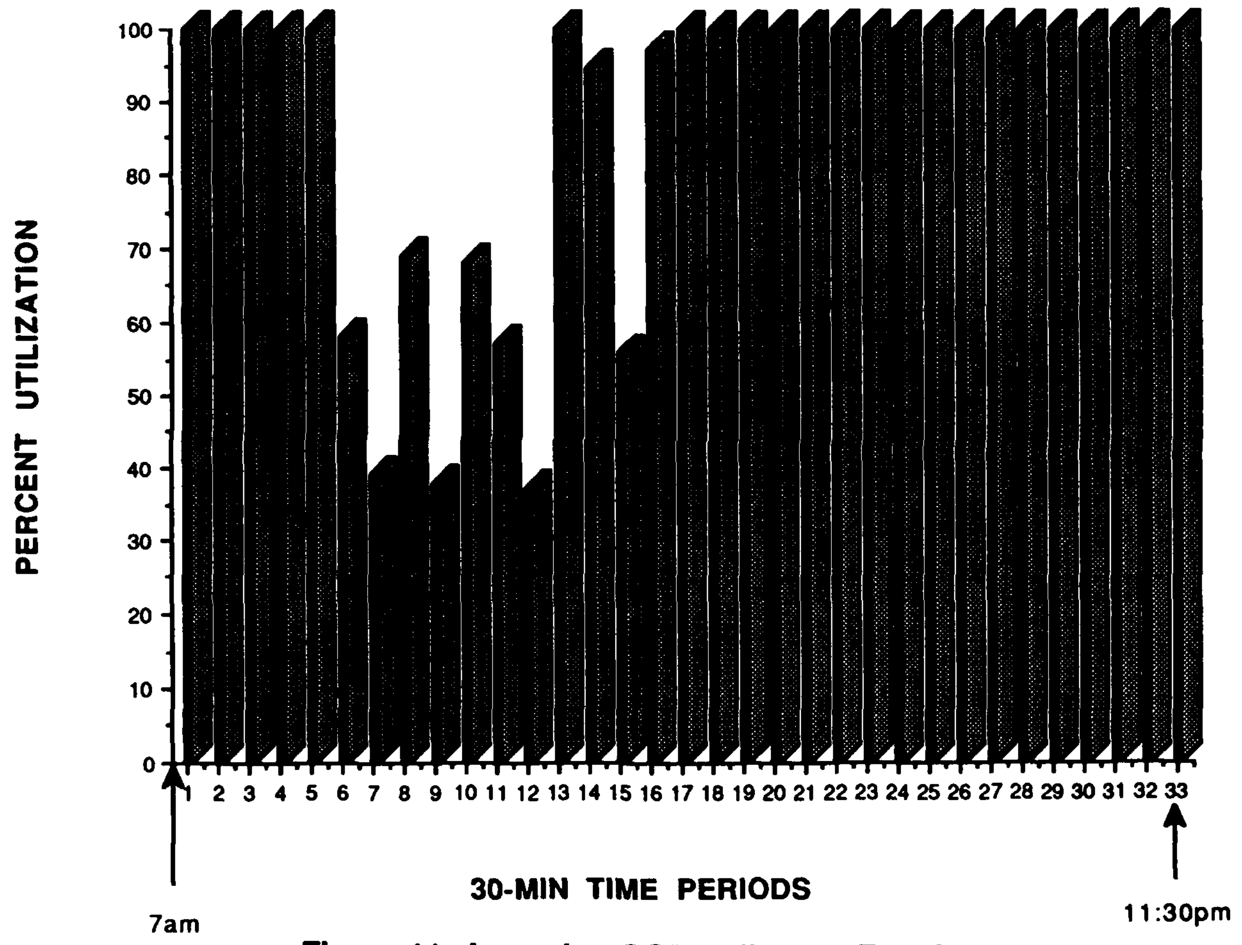


Figure 11. Incoming OCR Utilization Results

Several iterations resulted in the baseline resource schedule result shown in Figure 12. In this schedule, for each of 14 consecutive days, an average of about 23.3 kp of mail remains in the system beyond the day it entered. Of the total daily volume processed, about 0.6 percent remains in the system to be processed the following day. The total resources shown in Figure 12 may not be the absolute minimum required equipment schedule for the process, but as the overall utilizations in Table 6 indicate, the resources are heavily utilized.

It bears noting at this point, that achieving this feasible schedule by iterative simulation is a fairly difficult process unless one has experience with the processing system or is provided a good starting point. Obtaining this feasible resource schedule, of course, is not the main objective of the simulation model (the MILP models are aimed at providing a good starting schedule). The purpose of GMF simulation is to provide a testbed for experimentation and for gaining insight into the system's behavior.

Given a feasible resource schedule may be implemented, many other simulation statistics are available to assist the postal manager in planning and operations. For example,

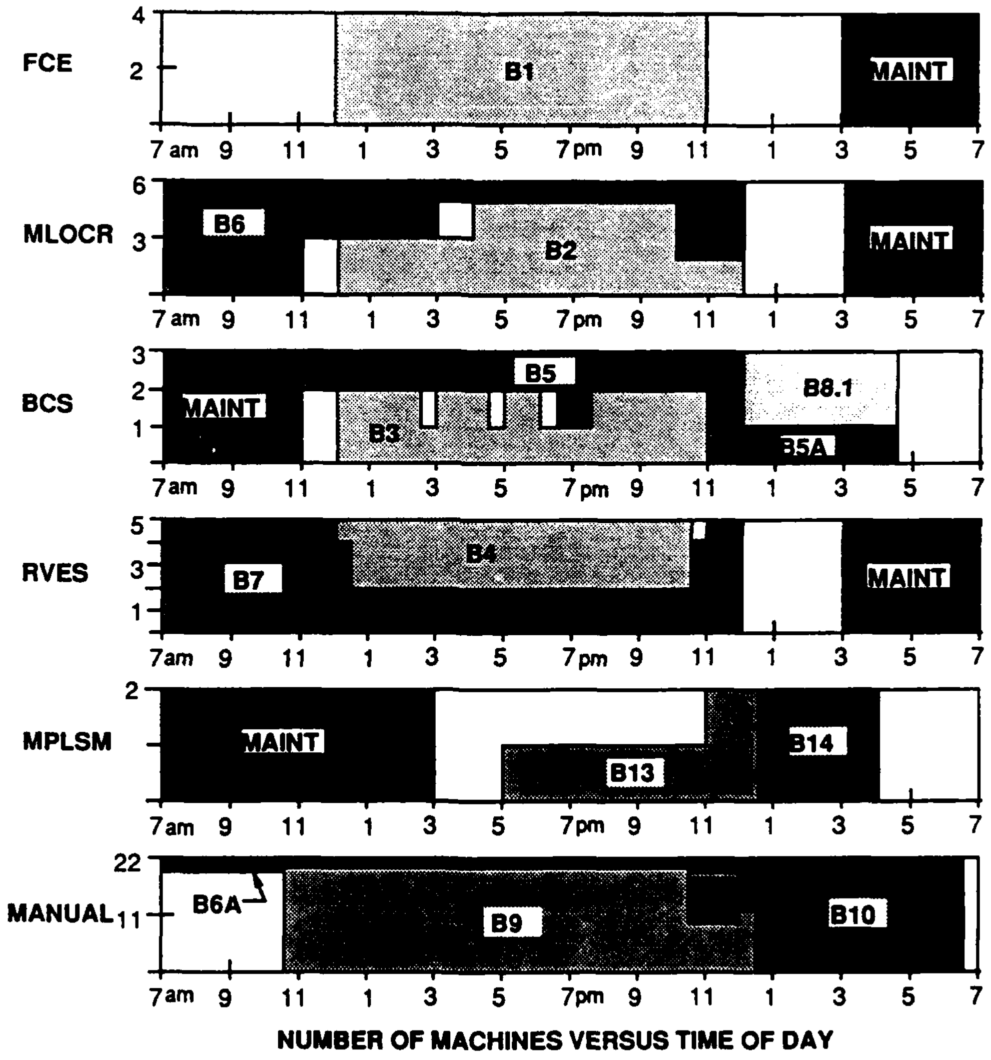


Figure 12. Feasible Resource Schedule Result

Table 6: Overall Resource Utilizations Result

Resource Type	Scheduled Machine-Hrs	Utilized Machine-Hrs	Percent Utilization	Total Processed (kp)
FCE	44.0	38.1	86.6	800
MLOCR	97.0	91.8	94.6	2524
BCS	48.0	33.4	69.6	935
RVES	84.0	81.2	96.7	1489
MPLSM	18.0	12.9	71.7	380
Manual	450.0	436.0	96.9	809

maximum queue lengths at each station may be used for sizing inventory floorspace requirements. For the baseline model, the specific maximum queue length each day was about 415 kp at the incoming MLOCRs. Mail waiting times in the queues and entity times in system provide insight into the processing system and may simply be used to validate the simulation against historical data. On the otherhand, simulation may provide interesting statistics which would be difficult to measure in the actual operating environment. For example, the average letter in the model required 350 minutes in the system to reach sequencing or dispatch.

Detailed histograms of arrival profiles by mail type are possible for virtually any station in the model. The simulation model provides an abstraction of the real system; a representation which can be experimented with simply by altering parameters or the order of activities. Detailed statistics can be gleaned from any event or process of interest.

5.2 Sensitivity to Increasing Volume

The degree of sensitivity of the processing system to increased daily mail volume can be surmized from the results in Table 6. The MLOCRs, RVES, and manual stations are being heavily utilized and are likely to be bottlenecks in the system for even modest increases in flow. Figure 13 shows the mail volume (kp) remaining overnight in the system for the 14 simulated consecutive days at the baseline volume, a five percent increase, and a ten percent increase in mail volume. A similar trend for the mean times in system results is shown in Figure 14.

At an increased daily volume of five percent, the system quickly becomes unable to handle the flow. By the eleventh day at five percent increased volume, about 19 percent of the daily letter mail is waiting overnight for processing the next day and the mean time in system has increased by more than 60 percent. The resource statistics for this scenario are shown in Table 7. The MLOCRs and RVES, which are scheduled over their entire possible operating windows, are 100 percent utilized. Reallocation cannot be used fix this problem. The only feasible alternative is the acquisition of additional resources.

The nominal volumes for the arrival streams are not a worst case scenario. Increases of five and ten percent volume are probably modest compared with holiday mail volumes experienced by the USPS. With the simulation model, the daily stream volumes can be selected from probability

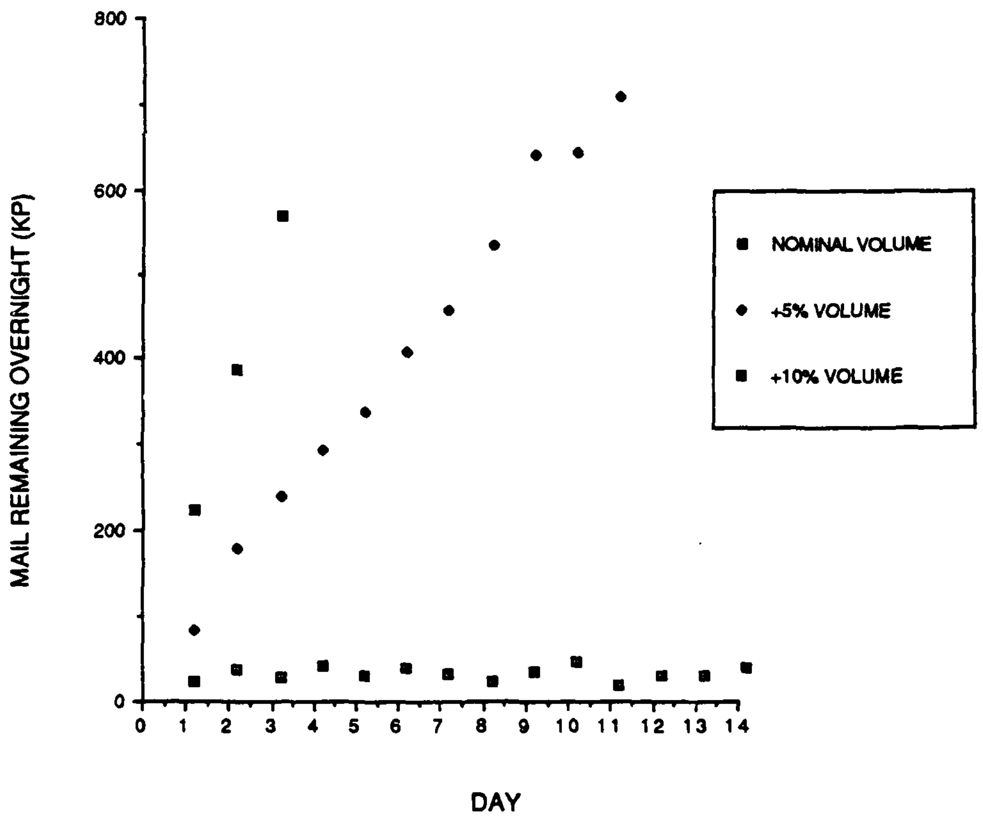


Figure 13. Sensitivity to Increasing Mail Volume

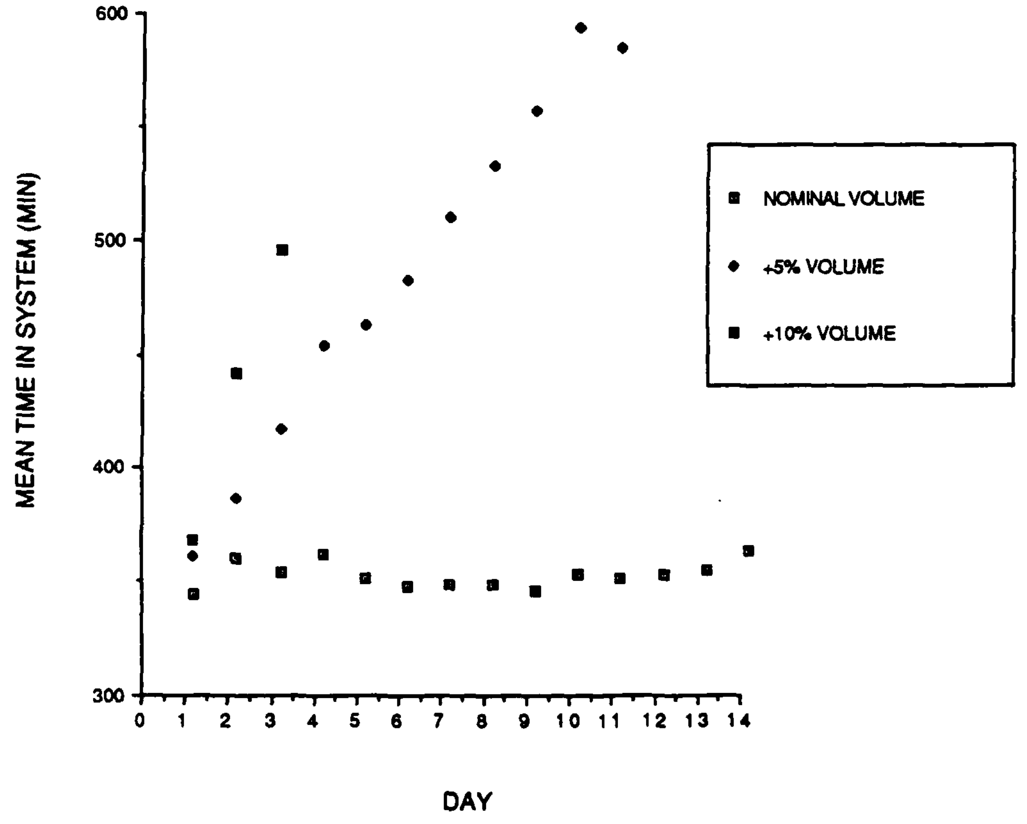


Figure 14. Mean Time-In-System for Increasing Volume

Table 7: Resource Utilization Affected by Increased Mail Volumes

Resource Type	Scheduled Machine-Hrs	Utilized Machine-Hrs	Percent Utilization	Total Processed (kp)
FCE	44.0	39.9	90.6	837
MLOCR	97.0	96.3	99.3	2652
BCS	48.0	33.1	69.0	926
RVES	84.0	84.0	100.0	1572
MPLSM	18.0	14.1	78.3	415
Manual	450.0	432.0	96.0	822

distributions, permitting daily variations as well as weekly or seasonal patterns. For an example of daily variation, if the nominal daily volume (3707 kp) is 1σ above the mean of a normal distribution, the total daily volume would exceed 3707 kp about once per week. The simulation of such arrival processes can assist postal managers in preparing for resource acquisitions and planning the implementation of the additional equipment, especially during rush periods.

5.3 Increasing Automated Mail

Suppose that instead of increasing the total volume, the percent of the daily nominal volume which is automated (able to be processed by MLOCRs and BCSs) is increased. This scenario is representative of a secular trend toward increasingly automated letter mail as the USPS seeks to achieve 100 percent letter mail automation. Such a trend would follow from improved MLOCR capabilities at originating GMFs and improved address field quality. For this scenario, the probabilistic branching proportion of FCE processed mail which is sent to the collection MLOCRs is increased. The percent of input managed mail which is completely bar-coded is also increased while MMP mechanized and manual mail stream volumes are decreased.

As a result, the volumes of mail left in system overnight during the 14 consecutive simulated days are compared with the baseline in Figure 15. As with overall mail volume increases, the system quickly becomes unable to handle the modest changes in these input parameters. At 10 percent increased automated mail, by day 14 over 15 percent of the daily mail is held overnight. Although the trends in Figure 15 are quite similar to the results in Figure 13, the bottlenecks in the flow have changed. In this case, of the cumulative total of 4872 kp held in system overnight, 84 percent was left in the incoming MLOCR queue.

As the percent automated increases by 20%, the system backs up much more quickly. Almost 25 percent of the daily mail volume is held over in system by day four. In this case, the outgoing MLOCR queues also begin to grow. Of the 2431 kp held overnight in this simulation, 33 percent was left at the outgoing MLOCRs and 57 percent remained in the incoming MLOCR queues. More MLOCRs would be required to offset these increases in automated mail and then additional BCSs will also be required.

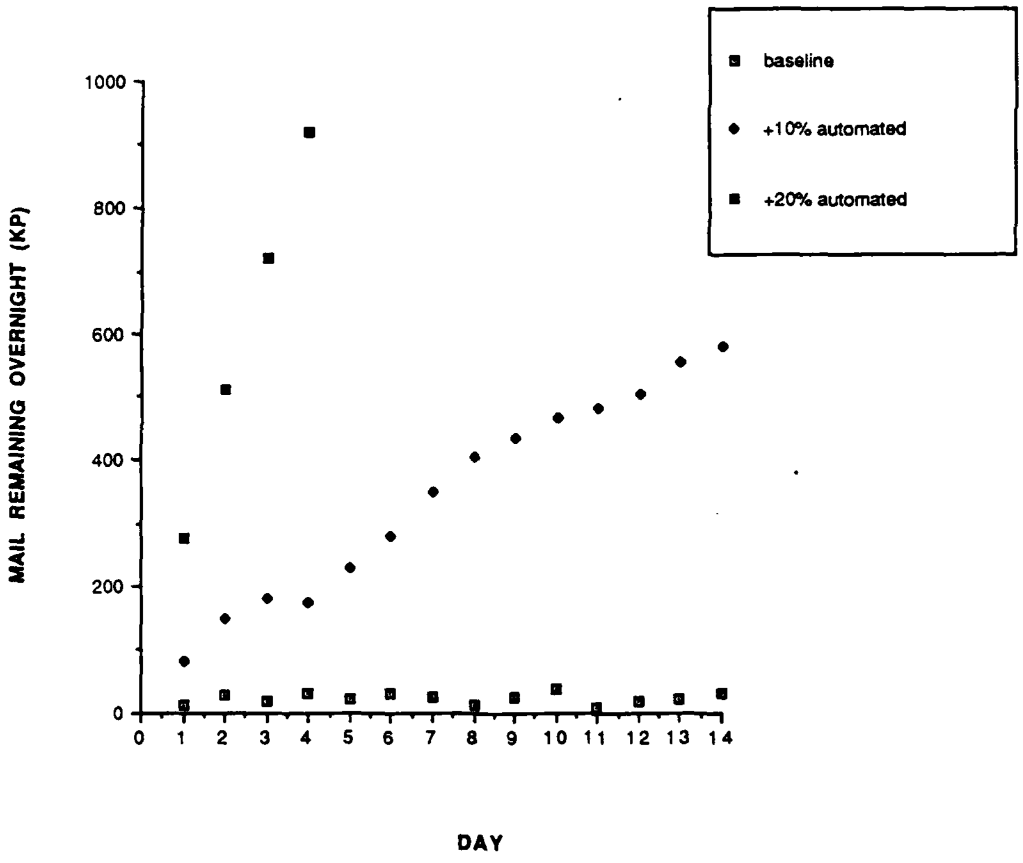


Figure 15. Effect of Increasing Automated Mail

5.4 Between-Station Transfer Times

Given the objective of determining whether letter mail meets the sequencing and dispatch times, handling delay times are important. The between-station transfer time distribution installed in the baseline model (uniform 10 to 15 minute delays) is a discretionary estimate. By changing this transfer time parameter in the simulation model, some interesting system complexities are revealed.

Table 8 provides mean time in system (TIS) and the number of remaining entities resulting from the baseline model with uniformly distributed transfer delays from 10 to 15 minutes. These results are compared to exponentially distributed delay times with a mean of 15 minutes, and to the system modelled without any transfer delays between stations. The data for mean time in system do not include the processing times for the entities which do not complete processing. The trend in the mean time in system is as one would expect. As the mean delay time increases, the mean time in system increases.

What is seemingly paradoxical is that as the mean transfer delay time (and variance) are reduced to zero, more mail remains overnight than in the baseline model, even though the mean time in the system has been reduced. This

Table 8: Effect of Transfer Time Changes

	Uniform(10,15)		No Delays		Exponential(15)	
Day	Queue Vol[1] (kp)	Mean TIS[2] (min)	Queue Vol (kp)	Mean TIS (min)	Queue Vol (kp)	Mean TIS (min)
1	13	340.1	54	325.5	8	338.2
2	27	356.2	59	352.2	25	350.3
3	19	350.0	35	353.7	21	356.9
4	32	357.9	48	340.6	47	363.2
5	21	347.2	36	342.5	29	354.8
6	31	344.1	46	337.5	36	348.6
7	24	344.7	14	341.2	30	339.8
8	13	345.3	11	330.9	46	352.8
9	26	341.9	13	327.7	41	363.4
10	38	349.6	19	324.3	37	361.4
11	9	347.4	12	327.8	34	355.6
12	20	349.7	23	327.8	52	359.6
13	22	350.9	15	328.0	87	365.6
14	31	360.2	36	338.4	16	366.7
	---	-----	---	-----	---	-----
Mean	23.3	348.9	30.1	335.6	36.4	355.5
Variance	63.2	32.4	265.6	86.6	337.2	73.6

[1] The mail volume remaining in the system queues and held over until the next day to complete processing.

[2] Mean time in system for all entities completed processing during that day.

is because the processing resources were first scheduled by iterative simulations of the baseline model which included uniformly distributed delays with mean 12.5 minutes. By reducing the delay times, in this case, letters which can be processed quickly by the RVES now arrive at the incoming RVES queue behind interposing mail which requires a longer processing time. The equipment schedule must now be altered if benefit is to be gained from the reduced delay times; the moral being that material handling issues, even relatively innocuous ones, can and should be examined as part of the processing system, and not independently.

Chapter 6. MILP Modelling

While simulation provides for detailed modelling of the GMF-A letter mail processing system and the evaluation of various scenarios, the GMF-A simulation cannot in itself select optimal automation strategies with regard to some objective function. By iterative simulation modelling, a feasible resource selection and schedule may be found, but this process is tedious and the solution may be a poor one compared with alternative solutions. While simulation can be employed to ensure that a chosen system configuration and schedule is practical or even feasible, a mathematical programming model and solution method is required to provide the most efficient strategy.

As a fundamental objective, one would like to be able to select equipment numbers and schedules which would minimize acquisition and operating costs over some appropriate time period while still meeting the imposed processing deadlines. Providing that the automated letter mail system can be represented as a set of linear constraints, a mixed integer linear programming (MILP) approach is necessitated (as opposed to a linear programming model) by the obvious constraint that the number of machines must be integer. In this section, the general approach to casting the GMF-A processing system as a multistage network is followed by a discussion of the major constraint equations and the objective function cost factors in forming the MILP model.

6.1 Multistage Network Model

Determining the optimal number of machines to install in an automated GMF represents a challenging problem with long range economic and operational consequences. Accounting for relevant cost factors, floorspace limitations, labor requirements, maintenance schedules, complex mail flow with output stream branching, and the processing rate dependence on the many mail types requires a rigorous MILP model formulation. From a network view, the GMF activities are represented by a set of inventory, processing, and destinating nodes. To represent the operations over a 24-hour period, inventories must be conserved from one period to the next. Mail is either processed and passed to a downstream node or remains in inventory until the next period.

The MILP model presented here represents the system over one day as a set of 30-minute periods or stages. During a single period, a piece of mail is processed on no more than one machine. All mail arriving to the system or to a processing station, arrives at the beginning of a period. Each processing station (for GMF-A there are 14 stations) is modelled by an inventory node and a processing node as shown in Figure 16. In this figure, the dashed arrows indicate transfers from period to period and can be thought of as connecting the 48 stages. Solid arrows represent processing transfers which occur within the period.

Figure 16 is representative of the basic elements employed in the formulation of the MILP. Input mail to a processing station at period t includes all relevant mail that the preceding stations processed during t-1, plus mail remaining in inventory after t-1 (for the moment, exogenous arrival streams are not considered). Input mail is either processed (transferred to the processing node) or remains in inventory for the next period (t+1). Processed mail is transferred to the next station's inventory node, or to a destinating node if the mail is finalized, arriving at period t+1.

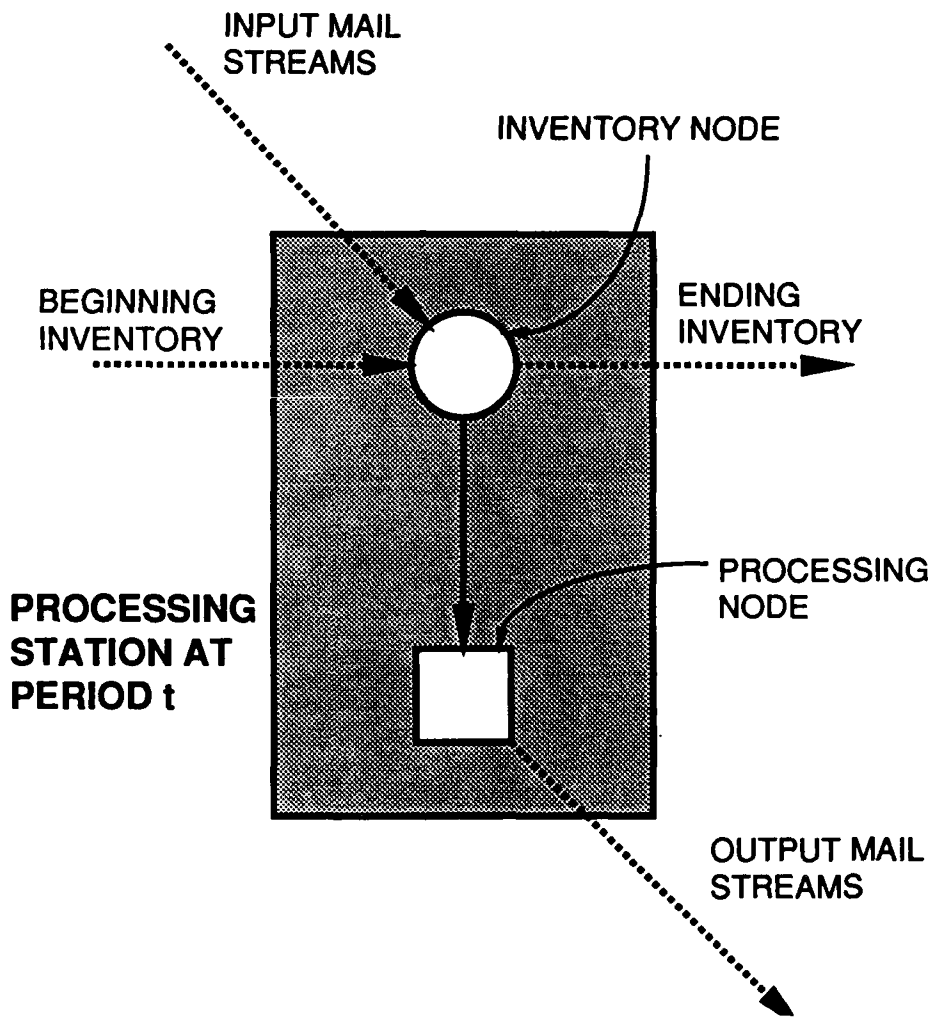


Figure 16. MILP Processing Station Model

The MILP model is deterministic. The branching proportions for processed output are fixed at the assumed values (based on data for actual and estimated GMF-A system behavior). In the simulation model output stream branching fractions are the elements of a stochastic matrix, whereas for the MILP these proportions are fixed coefficients in constraint equations. Likewise, mail arriving at the GMF is inserted as a deterministic volume at each 30 minute period in the MILP model in order to approximate the arrival profiles shown in Figure 9. Between-station delays are not probabilistically modelled since the MILP has no variable activity durations per se. Mail processed at period t is simply available at the next station at period t+1.

It is evident by intuitive reasoning that when the stations are operating at capacity, the multistage MILP approach incorporates an average delay of about 15 minutes. The mail processed immediately at the start of period t will arrive at the following station at the beginning of period t+1, effectively delayed 30-minutes between stations. On the otherhand, a letter processed at the end of period t has virtually no delay. When stations are operating below capacity, the average transfer times are effectively longer.

The general processing station portrayed by Figure 16 simultaneously handles several streams. The input volume is comprised of separate mail types, possibly from several different preceding stations, and may include some arriving mail to the GMF. In order to introduce the modelling notation and level of complexity, the total input mail volume arriving at inventory node n at period t is written

$$\sum_{i\epsilon I_n} \quad \sum_{k\epsilon L_n} (x_{ik}(t)) + (a_i(t)|i\epsilon A)$$

where L_n	the set of processing nodes immediately preceding n in processing of stream i
k	a particular node preceding n
I_n	the set of input streams processed at n
i	a particular input stream
t	the time period
$x_{ik}(t)$	the mail volume of stream i from preceding processing node k arriving at period t
A	the set of exogenous arriving mail streams which are initially entering the GMF
$a_i(t)$	volume of arrival stream i during period t.

The expression $(a_i(t) | i \epsilon A)$ is the volume of arrival stream i during t, given i is an element of the set A.

The inventory volume for each mail stream must be conserved separately and each processed stream at a station is separated into several output streams. Therefore, the number of constraints required to enforce conservation of flow at a single inventory node n for 24-hours is product of the number of input mail streams (I_n) and the number of 30-minute periods (T_n) in the operating window at processing station n. The number of conservation constraints needed for all inventory nodes in the model will be

$$\sum_{n=1}^{N} (I_n \cdot T_n)$$

A similar structure is applied for conservation of flow at the processing nodes and is employed in the model formulations described in Chapter 7.

6.2 System of Constraints

The underlying framework for modelling the GMF-A letter mail processing system consists of:

(1) ensuring mail volume conservation (by stream) at the inventory and processing nodes which comprise the processing stations;

(2) fully describing the input mail flow to the GMF according to the defined arrival profiles (Figure 9);

(3) imposing machine availability and processing rate constraints on the volume processed at each station each period;

(4) constraining all mail to complete processing in time to meet the critical sequencing window or outgoing dispatch times; and,

(5) accurately quantifying floorspace requirements and separate cost factors in order to formulate the objective function to minimize costs over some planned operating period.

The following chapter provides a detailed presentation of the system of variables and constraints used to fc .nulate a MILP model for optimizing the automated equipment "sizing" problem (the number of machines which must be procured) and the optimal scheduling of these resources. The objective is to minimize the acquisition, maintenance, additional floorspace, and operator (labor) costs while still meeting the sequencing and dispatch deadlines for sorting completion. Additional formulations, extensions to this MILP model, are presented in Chapter 9.

Chapter 7. GMF MILP Model Formulation

The system of linear constraints for composing the GMF MILP formulations is described in this section. In some cases alternative approaches are included in this presentation. Not only do these alternative considerations assist in the understanding of the system being modelled, but may be superior for extending the model as discussed in Chapter 9. As pointed out in the following presentation, the selection of constraints for the final MILP was sometimes a matter of correctness. In other cases, the preferred constraints were a compromise to make the model more tractable.

7.1 General Conservation Constraints

Consider the conservation of mail volume at the inventory node n of a general processing station embedded in the system (Figure 17). As noted earlier, node n may be processing several different types of mail such that the input volume at period t may be of various mail streams (i). For that matter, each stream may have contributions from several nodes preceding n (preceding node k is an element of the set of nodes L_n which precede n). Let $v_i(t)$ represent

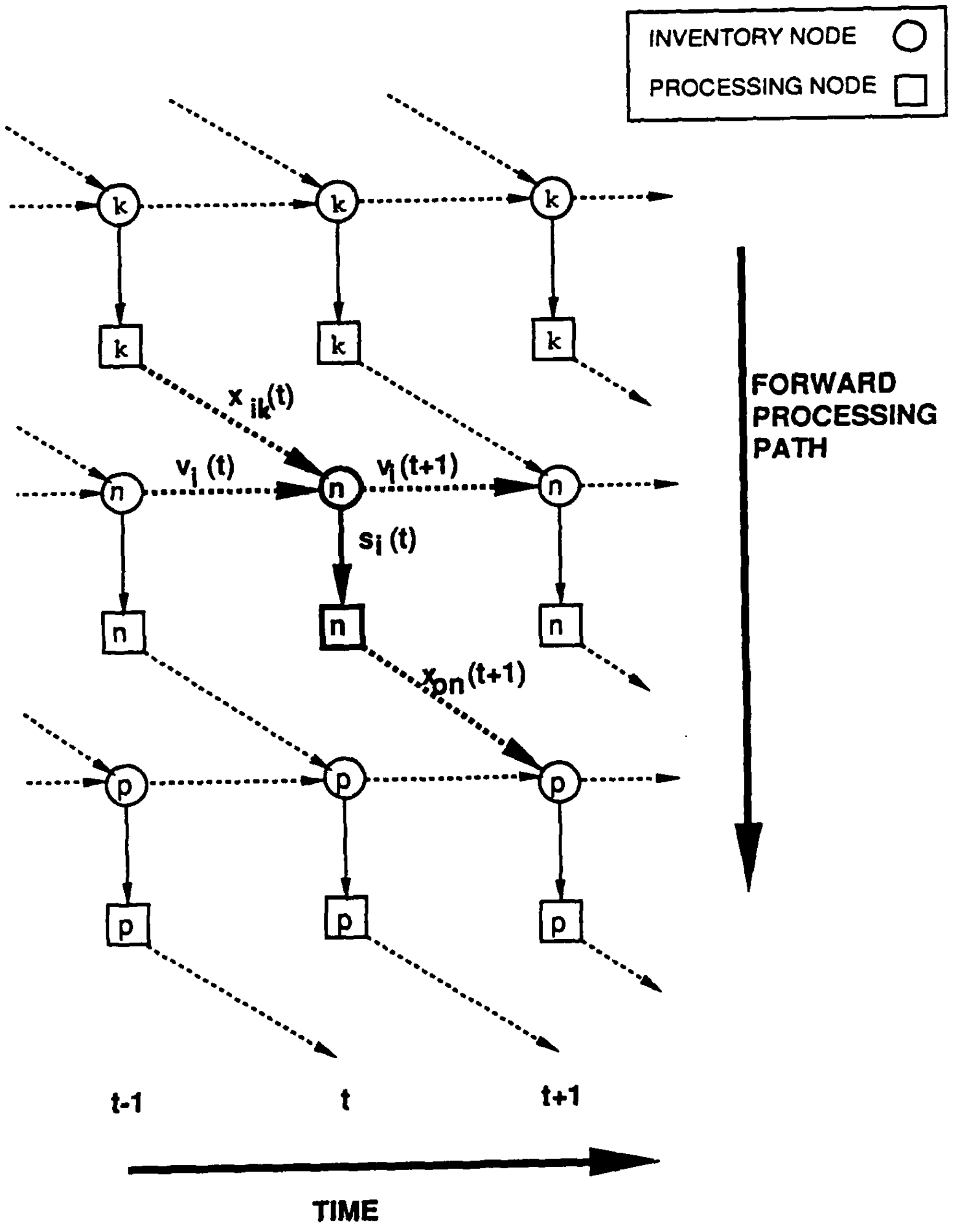


Figure 17. MILP Multi-station Diagram

the inventory volume of stream i at the beginning of period t and $s_i(t)$ denote the volume processed during t. The conservation of mail stream i at node n at period t can be expressed

$$\sum_{k \in L_n} x_{ik}(t) + v_i(t) = s_i(t) + v_i(t+1), \text{ where } i \in I_n \quad (1)$$

In fact, if we denote the set of possible operating periods for station n to be T_n, the set of conservation constraints for the N inventory nodes in the system can be represented by

$$\sum_{k \in L_n} x_{ik}(t) + v_i(t) = s_i(t) + v_i(t+1), \quad \begin{array}{l} n=1,\ldots,N \ (2) \\ \text{all } i \in I_n \\ \text{all } i \in T_n \end{array}$$

Each output stream, designated $x_{on}(t+1)$ in Figure 17, must be some fraction of $s_i(t)$. Here $x_{on}(t+1)$ is the volume of output stream o processed at n during period t and transferred to a following node p at period t+1. That is, $x_{on}(t+1)$ is the output stream from station n at stage t and the input stream to station p at t+1. The output volume depends on the machine processing rate at n, the separation fraction (branching proportion), and the number of machines employed at n during t. Let r_{in} denote the processing rate at n for stream i (the rate may depend on the type of mail being processed). Let f_{ion} represent the proportion of stream i converted to output stream o at station n. The conservation constraints at the processing nodes can be expressed by Eq. 3.

$$x_{on}(t+1) = y_n(t)\cdot\sum_{i\epsilon I_n} [f_{ion}\cdot r_{in}\cdot s_i(t)], \quad \begin{array}{l} n=1,\ldots,N \\ \text{all } o\epsilon O_n \\ \text{all } t\epsilon T_n \end{array} \qquad (3)$$

Where $y_n(t)$ is the number of machines available at station n during period t.

While this equation imposes resource availability and processing rate bounds on the output mail flow, it is unfortunately nonlinear. The constraints can be linearized by introducing the following transformation and an intermediate variable. Let $w_i(t)$ represent the number of machines at node n devoted to processing a particular stream i during t, such that Eq. 3 can be expressed as a set of conservation, processing rate, and resource availability constraints. The conservation at processing nodes is then

$$x_{on}(t+1) = \sum_{i\epsilon I_n} [f_{ion} \cdot s_i(t)], \quad \begin{array}{l} n=1,\ldots,N \\ \text{all } o\epsilon O_n \\ \text{all } t\epsilon T_n \end{array} \qquad (3.1)$$

with separate processing rate constraints

$$x_{on}(t+1) \leq \sum_{i \in I_n} [f_{ion} \cdot r_{in} \cdot w_i(t)], \quad \begin{array}{l} n=1,\ldots,N \\ \text{all } o \epsilon O_n \\ \text{all } t \epsilon T_n \end{array} \qquad (3.2)$$

and processing availability constraints

$$y_n(t) \geq \sum_{i \in I_n} w_i(t), \quad \begin{array}{l} n=1,\ldots,N \\ \text{all } t \epsilon T_n \end{array} \qquad (3.3)$$

Since $y_n(t)$ will be minimized by the objective function, this effectively minimizes $w_i(t)$ and Eqs. 3.2 and 3.3 can be inequalities. Eqs. 3.1, 3.2, and 3.3 are linear and can replace Eq. 3.

However, there is a more subtle problem with this formulation. While Eqs. 3.2 and 3.3 appear to correctly model the rate and availability constraints, their effect in the MILP serves to inaccurately describe the system. Because the objective function will minimize $y_n(t)$ and therefore $w_i(t)$, $x_{on}(t+1)$ is indeed bounded. However, the solution method will capitalize on the relationship described by Eq. 3.2. In solving the linear program, larger values of $w_i(t)$ will be selected in cases where the product of f_{ion} and r_{in} is large, thereby maximizing total output without preserving the integrity of the relationship between input and output stream volumes. In result, the sum of input proportions intended to comprise a particular output stream o may not be equal to the volume of stream o. In fact, the initial Eq. 3 introduces the same problem.

A more appropriate replacement for Eq. 3 is the set of constraints described by Eq. 3.1 and Eq. 3.3 supplemented with an upper bound constraint on $s_i(t)$ of the form of Eq. 4.

$$s_i(t) \leq r_{in} \cdot w_i(t), \; n=1,\ldots,N \tag{4}$$

To complete the conservation constraints, the coefficient r_{in} must be defined. The rate at which letters are processed at a station n may be constant (such as for the MLOCRs). The processing rate may instead be dependent on the input stream type of the letter (the case for the MPLSMs), or may depend on the time to process a particular type of letter. When the processing rate is dependent on particular separation types of mail, such as at the RVES stations, the processing rate is effectively dependent on the output streams. The coefficient r_{in} can be defined as,

$$r_{in} = \begin{cases} r_n & \text{for constant rate processing} \\ r_{in} & \text{for input stream dependent rate} \\ \sum\limits_{o \in O_n} (f_{ion} \cdot r_{on}) & \text{for output dependent rate} \end{cases}$$

where r_{on} is the processing rate for output stream o at station n. The expression in the last condition requires the limitation that mail will actually branch according to the fixed proportions, an assumption already stated.

The set of constraints described by Eqs. 2, 3.1, 3.3, and 4 provides for the conservation of flow at the inventory and processing nodes, as well as machine availability and processing rate bounds. These are constraints like those employed in the GMF-A model formulations. To this point we have neglected the mail flows arriving to the GMF from outside the system. The modelling of this exogenous mail is considered next.

7.2 Arriving Mail Profiles

Two alternative methods for handling the arriving mail to the system are presented here. The first approach provides for a separate "receiving area" inventory, while the second permits a simpler structure of inserting arriving mail directly at the inventory nodes of the processing stations.

Suppose that incoming mails (of all types) are first received at a separate inventory area on arrival at the GMF. Consider further, that mail arriving at period t will be forwarded to its first processing station either at t+1 (at the earliest) or some time thereafter. Let $a_i(t)$ represent the deterministic volume of stream i arriving at the GMF at period t. Let $d_i(t)$ denote the volume of stream i forwarded to its first processing station at period t. Another set of inventory nodes is required with associated inventories $u_i(t)$, the volume of arriving stream i in receiving inventory at the beginning of period t.

Arriving mail profiles can be imposed on the system by Eq. 5.

$$a_i(t) + u_i(t) = d_i(t+1) + u_i(t+1), \quad \text{all } i \in A, \; t = 1,2,\ldots,T_i \qquad (5)$$

where A is the set of arrival streams to the GMF and T_i is the latest nonzero volume period in the arrival profile for i. Eq. 1 must then be amended to include the mail forwarded from the receiving inventory as shown in Eq. 6.

$$\sum_{k \in L_n} x_{ik}(t) + [d_i(t) \mid i \in A] + v_i(t) = s_i(t) + v_i(t+1), \quad n=1,\ldots,N, \; \text{all } i \in I_n, \; \text{all } t \in T_n \qquad (6)$$

Eqs. 5 and 6 provide for enforcing arriving mail profiles on the incoming mail to the GMF. Additionally, inventory considerations, both at the receiving area and at the processing stations, can be modelled using this approach. The alternative is to insert the arrival volumes directly at the processing stations. In this case, Eq. 7 replaces Eqs. 5 and 6.

$$\sum_{k\epsilon L_n} x_{ik}(t) \;+\; [a_i(t)|i\epsilon A] \;+\; v_i(t) \;=\; s_i(t) \;+\; v_i(t+1), \quad \begin{array}{l} n=1,\ldots,N \\ \text{all } i\epsilon I_n \\ \text{all } t\epsilon T_n \end{array} \qquad (7)$$

This second approach is less comprehensive but eliminates the need for Eq. 5 and the variables $d_i(t)$ and $u_i(t)$. If one wished to examine inventory issues (for example, placing an upper bound on the processing station inventory), the first approach (Eqs. 5 and 6) should be employed. In the absence of appropriate inventory storage requirements and facility limitations for GMF-A, including Eqs. 5 and 6 in our model could have only been demonstrative.

7.3 Sequencing Operations

As discussed in the presentation of the simulation model, the parameters for the sequencing process depend on the number of carrier groups, delivery points, sequencing machines, bins per machine, sequencing passes, and the total sequencing volume. Because the effectiveness of these parameters and the sequencing operation itself were speculative at the time of this modelling effort, parameters from a separate study (deSilva, 1990) are used to set the sequencing window for the MILP model.

Sequencing proceeds based on the two-pass scheme described in Section 4. Each sequencing pass for each set of carrier groups requires about two hours. It is assumed that if at least half of the total sequencing volume arrives at the sequencing station, sequencing of the first set of carrier groups can start prior to the deadline period (T_d). All mail to be sequenced must arrive prior to period T_d. The more geographically distant carrier groups are sequenced first starting at T_d-3 for the first pass and T_d+1 for the second pass. The set of carrier groups nearer the GMF is sequenced from T_d+5 through T_d+12.

If we assume that the total sequencing volumes (X_s) for the two sets of carrier groups are equivalent, X_s can be determined by

$$\frac{1}{2} \sum_{t=1}^{T_d-1} \sum_{i \in I_n} \sum_{k \in L_n} x_{ik}(t) = X_s \qquad (8)$$

where n is the node representing the sequencing block(n=15).

Ensuring that at least half the mail arrives prior to T_d-3 in order to start sequencing the first pass is accomplished by Eq. 9 which also limits the volume of new arrivals during the first pass sequencing periods.

$$\sum_{i \in I_n} \sum_{k \in L_n} x_{ik}(t) \leq \frac{1}{3} X_s, \quad t = T_d-3,\ T_d-2,\ T_d-1 \qquad (9)$$

According to Eq. 9, the greatest arriving volume possible in the last three periods prior to T_d will be one third of X_s or a sixth of the total mail to be sequenced. Therefore, at least one half of the total sequencing volume must arrive before T_d-3. It is reasonable to expect that Eq. 9 will not usually be a tight constraint in the optimal solution since the objective function is intended to level resource requirements over time in order to minimize the number of machine acquisitions.

The remaining requirement for modelling the sequencing operations is a set of constraints defining the flow from first pass to second pass, from second pass to destinating nodes, and from sequencing to the MPLSMs which handle the rejected mail. Let $f_{p,r}$ be the reject rate for sequencing pass p. Then we model the first sequencing pass by

$$x_{on}(T_d + 1) = (1 - f_{p,r}) X_s \qquad (10)$$

where output stream o is input to the second pass, and

$$x_{on}((T_d-3) + b) = \frac{1}{4} f_{p,r} \cdot X_s, \; b=1,2,3,4 \qquad (11)$$

where stream o is the input to the MPLSMs.

Constraints of the form of Eq. 10 and 11 are needed for each pass to prescribe the flow of successfully sequenced mail to the next pass, or to destinating nodes, and rejected mail to the MPLSMs.

7.4 Meeting Dispatch Deadlines

There are two differing approaches to ensuring that all mail is finalized in time to meet dispatch deadlines. Critical dispatch times may be enforced by requiring the sum of all completed mail arriving at the destinating nodes prior to their deadlines be equal to the total GMF input volume. On the otherhand, requiring the ending inventory at each processing station to be zero can enforce the same condition. Each approach has inherent advantages and drawbacks for modelling the GMF system.

Considering the former approach, by requiring the total volume of mail at the destinating nodes, prior to the appropriate deadlines, to equal the total input volume is analogous to "pulling" mail through the system. Let D represent the set of destinating nodes and $T_d(n)$ be the deadline period at destinating node n. An intermediate variable V_n is used to represent the destinating volume at each $n \in D$. Then Eqs. 12 and 13 enforce the dispatch deadlines.

$$\sum_{t=1}^{T_d(n)} \sum_{k \in L_n} \sum_{i \in I_n} xik(t) = V_n, \quad \text{all } n \in D \tag{12}$$

$$\sum_{n \in D} V_n = \sum_{t=1}^{T} \sum_{i \in A} a_i(t) \tag{13}$$

The second approach, depleting ending inventories to zero throughout the system, can be thought of as "pushing" the mail through processing in order to meet the dispatch deadlines. A more detailed structure is needed to implement this approach. Each processing station has a operating window comprised of the periods in the set T_n. Let the first period in T_n be t_{n1} and the last be t_{nr}. That is, a specific operating period for station n is within the set,

$$T_n = [t_{n1}, t_{n1}+1, \ldots, t-1, t, t+1, \ldots t_{nr}-1, t_{nr}]$$

Now if the dispatch deadline at a destinating node n is $T_d(n)$, then all preceding stations sending finalized mail to n must have a last period of operation of $T_d(n)-1$. That is, for each destinating node n,

$$t_{kr} \leq T_d(n) - 1, \quad \text{all } k \epsilon L_n \tag{14}$$

where, at each preceding node k

$$v_i(t_{kr} + 1) = 0, \quad \text{all } i \epsilon I_k \tag{15}$$

This precedence relationship is carried throughout the processing network. For any station k preceding n, t_{kr} must be less than or equal to $t_{nr}-1$ and $v_i(t_{kr}+1)$ must be zero.

The first approach (Eq. 12 and 13) is far simpler to implement than this second methodology. However, the precedence relations established in the "inventory push" approach facilitate the pruning of equations and variables because the operating windows at the processing stations must be well-defined. Also, all of the output streams at t must be connected (stated in the appropriate equation) at t+1 in the model generation. In a problem of this size and complexity, the initial formulations may have subtle errors where a few output streams are not connected. If this condition occurs, Eq. 12 would be violated and no feasible solution found (and the errors difficult to find). With regard to this type of error, the second approach could be considered more robust in the sense that a feasible solution could still be found and the errors more easily detected.

7.5 Objective Function

The objective function seeks to minimize: (1) equipment acquisition and maintenance costs, (2) machine operating costs, (3) manual mail processing costs, and (4) *additional floorspace cost*. Acquisition and maintenance cost estimates per machine are shown in Table 2. The application of these factors in the MILP objective function requires a set of integer variables.

The processing availability constraint (Eq. 3.3) provides $y_n(t)$, the number of machines required at station n during period t, a sum of the number of machines processing the separate streams at n. In order to formulate a more readily-tractable MILP, the $y_n(t)$ must be a real-valued variable (otherwise this large model would also have several hundred integer-valued variables). However, the number of machines to be installed must be integer. This can be achieved by defining Y_m, the total integer number of machines of type m to be installed in the GMF, in the relation

$$Y_m \geq \sum_{n \in N_m} y_n(t), \quad \begin{array}{l} m=1,\ldots,M \\ \text{all } t \in T_n \end{array} \qquad (16)$$

where N_m is the set of stations employing machines of type m and M is the number of types of machines available in the model. Note that several stations utilize the same type of machines which, according to this formulation, may be readily reallocated.

Another approach is to define Y_n as the integer number of machines required at station n and replace Eq. 16 by:

$$Y_n \geq y_n(t), \quad \begin{array}{l} n=1,\ldots,N \\ \text{all } t \in T_n \end{array} \qquad (17)$$

For the GMF-A scenario, M=6 and N=14, so either (16) or (17) may be used to formulate a reasonable model. While the former permits virtually unlimited machine reallocations, the latter does not permit shared resources between any stations.

The objective function costs are stated in terms of 1990 ("current") dollars. Therefore acquisition costs are simply the 1990 estimated cost per machine, $C_a(n)$, times Y_n. Ongoing maintenance costs are expressed in current dollars by computing the present value of the annual cost over ten years at an eight percent discount rate (maintenance and labor costs are assumed to pace inflation). Therefore, the present cost of maintenance over the ten year period is the product of Y_n, the annual maintenance costs in current dollars ($C_m(n)$), and present value multiplier of 6.71 (Newnan, 1988).

The only direct equipment operating costs included in the GMF model are operator costs. It is recognized that the differing utility consumption of the automated equipment versus mechanized machines or manual processing could have bearing on various equipment selection problems, however these data were not obtainable. Labor costs for operating the equipment, as well as for manual sorting operatings are based on total operator hours (H_m) on each machine of type m derived from Eq. 18.

$$H_m \leq \frac{1}{2} \cdot \sum_{n \in N_m} \sum_{t \in T_n} [P_m \cdot y_n(t)], \quad m=1,\ldots,M \qquad (18)$$

Let P_m designate the estimated 1990 wage rate for operators on equipment of type m. The approximate cost of labor over a ten year period is then the product of H_m, P_m, the number of working days per year (assumed 315), and the present value multiplier 6.71, summed over the M machine types.

Additional floorspace beyond that already available is costed at a current dollar amount (represented by CPSF). The floorspace requirements per machine (F_m) are stated in Table 2 and can be directly applied to the Y_n to obtain the total floorspace required to accommodate the processing equipment. If the variable S designates the additional floorspace beyond that available, the cost will be CPSF times S.

The objective function can be stated.

$$\text{Minimize } Z = \sum_{n=1}^{N} [(C_a(n)+6.71\ C_m(n))\cdot Y_n]+CPSF\cdot S + \sum_{m=1}^{M} [(315\cdot 6.71\cdot P_m)\cdot Hm] \qquad (19)$$

Scaling should be applied to the coefficients in Eq. 19 to mitigate numerical stability problems. For example, the total objective functions costs (Z) may be expressed in units of $10,000.

7.6 GMF MILP Model Statement

A summary statement of the equipment selection and scheduling optimization problem modelled is as follows:

Problem Statement: Given the available type of equipment with known operating characteristics for each processing station, minimize the total cost of equipping and operating the GMF letter mail system over a ten-year period with a nominal daily mail profile and fixed processing flow stream branching.

Model Formulation:

Minimize Total Costs: Eq. 19

Subject to:

Conservation at Inventory Nodes: Eq. 7
Conservation at Processing Nodes: Eq. 3.1
Processing Rate Limitations: Eq. 4
Resource Availability: Eq. 3.3
Sequencing Window: Eqs. 8 and 9
Sequencing Operations: Eqs. 10 and 11
Dispatch Deadlines: Eqs. 14 and 15
Total Resource Acquisitions: Eq. 17
Total Operator Hours: Eq. 18
Additional Floorspace Constraints

where all variables are nonnegative and real

except Y_n $(n=1,2,\ldots,N)$ which are nonnegative and integer-valued

The example formulation described above inserts the arrival streams directly at the processing stations and does not permit equipment sharing between stations. In the following sections, by GMF-A MILP model, we refer to this formulation or to the similar cases where machine reallocations are permitted or receiving area inventory is included. In discussing the tractability of the MILP models in Chapter 8, this specific class of similar-structured formulations is being considered.

Chapter 8. Approach to Solution of MILP Models

In the initial formulation stages, the equipment selection and scheduling model was expected to consist of up to 10,000 linear constraints and 12,000 variables. Through the kinds of modelling considerations presented in the previous chapter, the final GMF-A MILP model was reduced to under 7,000 constraints and 8,000 variables. It is evident that some form of model generator program is necessary to create the input file for the solution of such a large model. The following sections describe the approach to solution of the optimization problems formulated in Chapter 7.

A C-language program was written to create a formatted model file for this specific application and is described in 8.1 and 8.2. This model file is translated to MPS-standard format as shown in 8.3. A discussion of experience to date with solving the linear programming (LP) relaxations of the GMF-A MILP is included in 8.4. The results of the LP relaxation efforts lend confidence to the validity of the model formulations and the generation process, but indicate that the mathematical programming models exhibit degeneracy.

8.1 A Format for Model Generation

From the start, we wished to take advantage of the widely-used and readily-available programming library of XMP routines (original release by Marsten, 1981). The XMP programming library is an integrated collection of FORTRAN subroutines for solving optimization problems. XMP's ability to solve large sparse problems is tied to routines which manage a lower triangular, upper triangular (LU) factorization of the basis matrix (Reid, 1982). The basic XMP library is a hierarchical structure of subroutines for performing the necessary functions involved in solving LP problems.

The XMP routines are one example of a class of large LP solvers, virtually all of which accommodate a MPS-standard format input file. This input file fully defines the model variables, constraints, and objective. While a model generator could certainly be written to create an MPS-standard file for the GMF-A model, this format is somewhat unwieldy. In MPS format, the constraint coefficients must be listed column by column. For illustration, a small LP problem is postulated (Hillier and Lieberman, 1987).

MINIMIZE $Z = 8x_1 + 10x_2 + 7x_3 + 6x_4 + 11x_5 + 9x_6$

subject to:

$$12x_1 + 9x_2 + 25x_3 + 20x_4 + 17x_5 + 13x_6 \geq 60$$

$$35x_1 + 42x_2 + 18x_3 + 31x_4 + 56x_5 + 49x_6 \geq 150$$

$$37x_1 + 53x_2 + 28x_3 + 24x_4 + 29x_5 + 20x_6 \geq 125$$

$$x_j \leq 1 \quad \text{for } j = 1,2,\ldots,6$$

$$x_j \geq 0 \quad \text{for } j = 1,2,\ldots,6$$

The MPS-standard format representation of this problem is shown in Figure 18. Given that the matrix of coefficients is defined, this simple problem can be quickly expressed in MPS format. For large problems however, this matrix is not obvious until the model is first generated so that MPS format is both difficult to generate and to verify.

The selected alternative approach was to create the input file in XML language format and then translate the file to MPS. XML (a pseudo-acronym for eXperimental Modelling Language) is a simple algebraic modelling language for which translation to MPS format is expedient. Unlike for MPS format, with XML the constraints actually appear as equations in the model file.

```
NAME          EXAMPLE PROBLEM
ROWS
 G  ROW1
 G  ROW2
 G  ROW3
 N  OBJZ
COLUMNS
    X1        OBJZ        8.
    X1        ROW3        37.
    X1        ROW2        35.
    X1        ROW1        12.
    X2        OBJZ        10.
    X2        ROW3        53.
    X2        ROW2        42.
    X2        ROW1        9.
    X3        OBJZ        7.
    X3        ROW3        28.
    X3        ROW2        18.
    X3        ROW1        25.
    X4        OBJZ        6.
    X4        ROW3        24.
    X4        ROW2        31.
    X4        ROW1        20.
    X5        OBJZ        11.
    X5        ROW3        29.
    X5        ROW2        56.
    X5        ROW1        17.
    X6        OBJZ        9.
    X6        ROW3        20.
    X6        ROW2        49.
    X6        ROW1        13.
RHS
    RHS       ROW1        60.
    RHS       ROW2        150.
    RHS       ROW3        125.
BOUNDS
 UP BOUNDS    X1          1.
 UP BOUNDS    X2          1.
 UP BOUNDS    X3          1.
 UP BOUNDS    X4          1.
 UP BOUNDS    X5          1.
 UP BOUNDS    X6          1.
ENDATA
```

Figure 18. MPS Format File for Example Problem

The XML format representation of the small example problem is shown in Figure 19. The first record is the problem title and is followed by the declaration of all variables. The constraints and objective function are then expressed in a straightforward algebraic manner where an equation name appears on the left. Limits are then placed on the constraints by limiting the range of the equation name expression. Bounds can be placed on the variables in the BOUNDS section of the file. The objective function equation is identified by the last record.

For generating models of the size of the GMF-A problem, the XML format is far easier to create and to verify. It is simply more obvious to construct the matrix of coefficients by proceeding one constraint at a time, then by generating the columns of coefficients. The process of translating from XML to MPS also provides a further measure of model verification.

The XML format model generation was accomplished by writing a C-language program specifically for this MILP problem. The author chose C because it offers the advantages of (1) simple character string manipulation, (2) direct file pointer control, and (3) portability across operating systems. The first two advantages come into play

```
$ EXAMPLE PROBLEM
$ DECISIONS
X1  X2  X3  X4  X5  X6
$ EQUATIONS
ROW1 = 12X1 + 9X2 + 25X3 + 20X4 + 17X5 + 13X6
ROW2 = 35X1 + 42X2 + 18X3 + 31X4 + 56X5 + 49X6
ROW3 = 37X1 + 53X2 + 28X3 + 24X4 + 29X5 + 20X6
OBJZ = 8X1 + 10X2 + 7X3 + 6X4 + 11X5 + 9X6
$ LIMITS
ROW1 > 60
ROW2 > 150
ROW3 > 125
$ BOUNDS
X1 < 1
X2 < 1
X3 < 1
X4 < 1
X5 < 1
X6 < 1
$ MINIMIZE OBJZ
```

Figure 19. XML Language File for Example Problem

because model generation is largely a matter of rigidly formatted output with only small amounts of input and computation. Portability was also important since the generator was constructed and tested on a microcomputer before being transferred to an IBM-3081 mainframe computer.

In order to achieve the required XML-language format, only minor changes to the formulated constraint equations are needed. For example, the flow conservation at inventory nodes (Eq. 7) is reexpressed in the following form.

$$\text{EQNNAME} = \sum_{k\in L_n} x_{ik}(t) + v_i(t) - s_i(t) - v_i(t+1), \quad n=1,\ldots N, \text{ all } i\in I_n \text{ all } t\in T_n \qquad (20)$$

where EQNNAME = $-a_i(t)$ if $i\in A$ and 0 otherwise

As an example of the notation, the variable $x_{ik}(t)$ for $i=14$, $k=1$, and $t=21$ appears as "X0140121" in the XML form of the model.

For illustration, suppose that the 1107th constraint in the generated model happens to be conservation of inventory (Eq. 7) at station three ($n=3$) during the 11th period ($t=11$) for input stream 14 ($i=14$). If 14 is not an exogenous arrival stream ($i\notin A$) and is comprised of the outputs from preceding stations 1 and 2 ($L_n = \{1,2\}$), then in XML format this constraint would appear as follows.

EQ71107 = X0140111 + X0140211 + V01411 - S01411 - V01412

Having demonstrated the notation and format for the variables and constraints, a description of the model generator program for creating the complete model file follows.

8.2 The Model Generator Program

A listing of the *model generator* program (MGP1) for the GMF problem described in Chapter 7 is given in Appendix B and comments appear throughout the code. MGP1's various functions can be grouped into three main categories which are:

(1) reading sets of input parameters, indices, and ranges including cost, floorspace, wage rates, sets of input and output streams, arrival mail profiles, etc. from an input file;

(2) computing coefficients and performing file management functions as required; and

(3) writing data to a set of output files according to the XML format requirements.

The input file for MGP1, called GMFA.DAT, is also included in App B and is described briefly here. GMFA.DAT input data are formatted in five main blocks which are summarized as follows.

Block 1: Stream numbers for mail arriving to the GMF, total volumes per stream (kp), each stream arrival profile identifier, and the station into which each stream is to be inserted.

Block 2: The three arrival profiles as a percentage of total stream volume per period (see Figure 9).

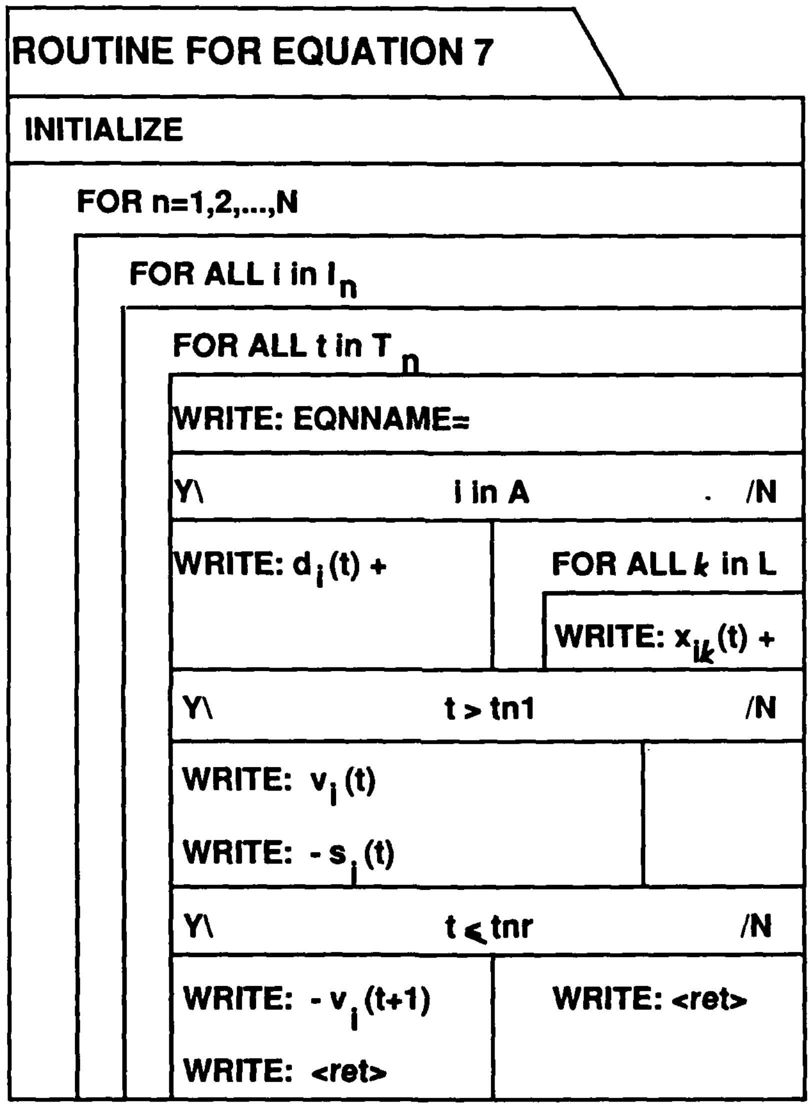
Block 3: Data for each processing station, where line 1 includes the station number, machine type number, first possible operating period (t_{n1}), last possible operating period (t_{nr}), the number of input streams, and the number of output streams. This line is followed by a short table of input streams and their preceding node of origin k (where if $k=0$, $i \epsilon A$), and output streams and their following processing node p (where if $p > 15$, then p is a destinating node).

Block 4: The set of matrices giving the fixed flow branching proportions in input stream (row) by output stream (column) format.

Block 5: The set of branching matrices are restated with machine processing rates included in either the first column (if rate is input stream dependent) or last row (if processing rate is output stream dependent).

GMFA.DAT is about 8.3 kbytes in size having 327 records. The branching matrix data are required twice because only one station matrix is held in program memory at a time. This makes final model generation possible on a microcomputer (most of this work was done on an IBM AT compatible).

The majority of the programming effort was in structuring the subroutines which print the variables, constraints, limits, and bounds in XML format. Of these, the constraints are the most difficult to generate. The flowchart in Figure 20 provides a diagram of the main structural components for generating the set of constraints represented by Eq. 7 in the model formulation. For GMF-A, over 1600 constraint equations are required to impose inventory conservation over the 15 processing stations for the processing windows specified in GMFA.DAT.



WRITE: "a" = print the character string representing "a"

Y\ a=b /N is an "if" statement:
if a=b follow logic under Y
if a=b follow logic under N

Figure 20. Flowchart for a Model Generation Subroutine

Much of the program coding is similar to that described in Figure 20. Each equation in the model formulation becomes an expanded set of constraints generated by looping over specified ranges or sets. Some additional structure is required in order to ensure that certain precedence relations are not violated and to manage the pointer position in the output file. The precedence considerations overlay the model formulation equations effectively causing the printing of some variables and constraints to be conditional. The pointer position can be thought of as carriage control required to maintain the right margin for constraints which exceed 72 characters in length (a limit imposed by the XML translator program).

Program output is written to four different files (XML1.MOD through XML4.MOD) which are small enough to be opened by the IBM-3081 editor (XEDIT) for examination. XML1.MOD contains the variable declarations. XML2.MOD is comprised of the sets of conservation equations at the inventory and processing nodes (Eq. 7 and Eq. 3.1). The rest of the constraints are written to XML3.MOD and the limits and bounds appear in XML4.MOD. These smaller separate files permit verification checks on the XML form of the model. The different methods employed to examine the correctness of the XML file included:

(1) ongoing examination of resultant output as the separate variable declaration and equation generating modules were written;

(2) specific inspections of identified difficult sets of constraints; and,

(3) browsing and random checking of the XML files.

The four XML.MOD files are combined to form the complete XML-language model file.

8.3 Model Translation to MPS Format

A FORTRAN program called XML is part of the XMP programming library and provides for efficient translation of the model. Only a few modifications to the XML program were made to change the output to create the MPS format as a family of separate files, and add diagnostic messages to supplement the programmed error messages. These diagnostics enhance the model verification aspects of the translation.

The process of translating the model from XML-language to MPS format actually provides a level of error checking, because the process identifies:

(1) excess (unused) declared variables;

(2) undeclared variables used in the constraint equations; and,

(3) general format errors.

Any of these, even minor format errors, may indicate a more serious error in logic in the constraint construction.

A complete MILP model described in Chapter 7, for the parameters shown in the GMFA.DAT input file (App.B), is 542 kb in size (15,519 records) when expressed in XML language. The same model requires 2.17 Mb (over 33,000 records) in MPS format. This particular formulation has 6629 constraints (4013 equality and 2616 inequality equations) and 7806 variables. Not including slack variables, there are 24,965 nonzeroes in the coefficient matrix for a sparsity of about 99.95 percent.

8.4 Model Tractability

The planned method of solution was to employ XMP to solve the linear relaxations of the MILP model in an implicit enumeration (branch and bound) scheme. This methodology was an attractive option because the XMP program library facilitates the construction of a main program to manage the enumeration and expedites successive LP solutions by adding the integer constraints as cuts and continuing from the previous LP solution.

To solve the LP relaxations of the MILP GMF-A model the XMP routines for the primal simplex method were employed. A relatively simple main program (XMPOST.FOR) was constructed to manage the XMP routines and was fashioned after LPDEMO, a XMP library routine driver program. XMPOST requires an additional input file (SPEC.MPS) which defines the objective function and output frequency settings. The following sequential series of functions and subroutine calls describes the procedure driven by XMPOST.FOR.

1. Read SPEC.MPS and initialize parameters
2. Call XMAPS to set up the XMP memory maps
3. Call MPSIN to read MPS-format input
4. Call XSLACK to set up the starting basis with slack and artificials included
5. Set the partial and multiple pricing parameters
6. Call XPRIML to manage the primal simplex:
   (a) Calls XFEAS to obtain phase 1 feasible solution
   (b) Calls XPHAS2 to execute primal simplex phase 2
7. Call MPSOUT to create the MPS-style output report

An initial solution attempt for an early form of the model (which included the receiving area considerations described by Eqs. 5 and 6), resulted in abandonment of the problem in XFEAS due to slow convergence of the phase 1 objective function. Such a result is most often indicative of degeneracy, but in a model of this size it was difficult to be assured that degeneracy was the only problem. The following brief discussion pertains to countermeasures and experiments to mitigate the degeneracy and to further verify the formulation.

Using XMP primal simplex routines, a problem is abandoned in XFEAS if more than a certain number of iterations occur without improvement. In phase 1 of the GMF-A problem, the sum of the artificial variables is initially the negative of the total daily arriving mail volume (the sum of the right hand sides of the set of constraints described by Eq. 7). Therefore, the initial phase 1 objective function is $Z = -3707$, which must be driven to zero to successfully complete phase 1.

The XMP parameter FACTOR is the basis refactorization frequency. If the basis is refactored five consecutive times without improvement in the objective function value, the problem is abandoned. FACTOR is advisedly set higher than the default of 50 for degenerate problems such as networks, allowing for slower convergence, but setting FACTOR too high can lead to numerical stability problems. These considerations can be skirted by overriding the improvement check and observing the objective function value over many iterations. For the GMF-A multistage network model, in over 18,000 iterations the objective function had only improved from -3707 to -3566 and the convergence rate did not appear to be increasing.

Two avenues of study were followed in order to further test the model structure. A scaled model comprised of five processing stations and eight operating periods was constructed. This model was created by changing the input data (GMFA.DAT) and modifying the generator program (MGP1) as required. This scaled model was representative of the full problem in that all of the constraint equations shown in the Chapter 7 model were used. The fewer nodes and shorter operation duration made for a smaller system of constraints which could quickly be solved by XMP primal simplex.

It was also possible to test parts of the complete model by zeroing all inputs except for one specific arrival stream, in effect activating only limited sets of nodes in the network. This approach was successful for downstream arrivals which insert mail for processing to only a few stations. These efforts assisted in the verification of the model formulation but also indicated that the generated complete models are largely degenerate. Two alternative solution approaches have shown that the LP relaxation of the MILP model is indeed tractable.

8.5 Alternative Solution Approaches

Successful LP solutions were obtained by applying the XMP dual simplex and MINOS, a different primal simplex solver. Quite basically, a significant property of the simplex method is that a LP problem can be converted to an equivalent dual formulation which, if feasible, has the same optimal solution value. Whereas primal simplex seeks to minimize costs subject to resource availability, the dual simplex approach maximizes the value of resource utilization subject to the resources' contributions to costs. One well-known advantage of most dual simplex solvers is that entering basic variables follow an order of entry and cycling can be reduced in solving degenerate problems. A complete discussion of this subject can be found in numerous texts including Nemhauser and Woolsey, 1988.

A separate dual model need not be generated since the MPS-format file contains all the information describing the dual problem. Except for the routines accessed, whether one is solving the dual or primal problem is virtually transparent when employing the XMP programming library. A main program, quite similar to XMPOST.FOR is constructed and the following sequential steps describe the procedure.

1. Read SPEC.MPS and initialize parameters
2. Call XMAPS to set up the XMP memory maps
3. Call MPSIN to read MPS-format input
4. Call XSLACK to set up the starting basis with slack and artificials included
5. Set the partial and multiple pricing parameters
6. Call XDUAL to manage the dual simplex:
   (a) XDUAL pivots to obtain a dual feasible starting solution
   (b) Calls XDUAL2 to execute dual simplex pivots to obtain an optimal solution
7. Call MPSOUT to create the MPS-style output report

Because the objective function is now to be maximized, the optimal solution to the dual formulation is approached from below as shown in Figure 21. Although the starting solution was achieved quickly, over 12,500 dual simplex iterations were required to reach the optimal solution. This corresponded to about two cpu hours on the IBM-3081D, a significant improvement over the XMP primal which, based on a linear extrapolation of the convergence rate, could be expected to only reach a phase 1 feasible solution in 12 cpu hours.

Another solver, MINOS, was able to converge to the optimal solution of a GMF-A model more rapidly (Geraldo

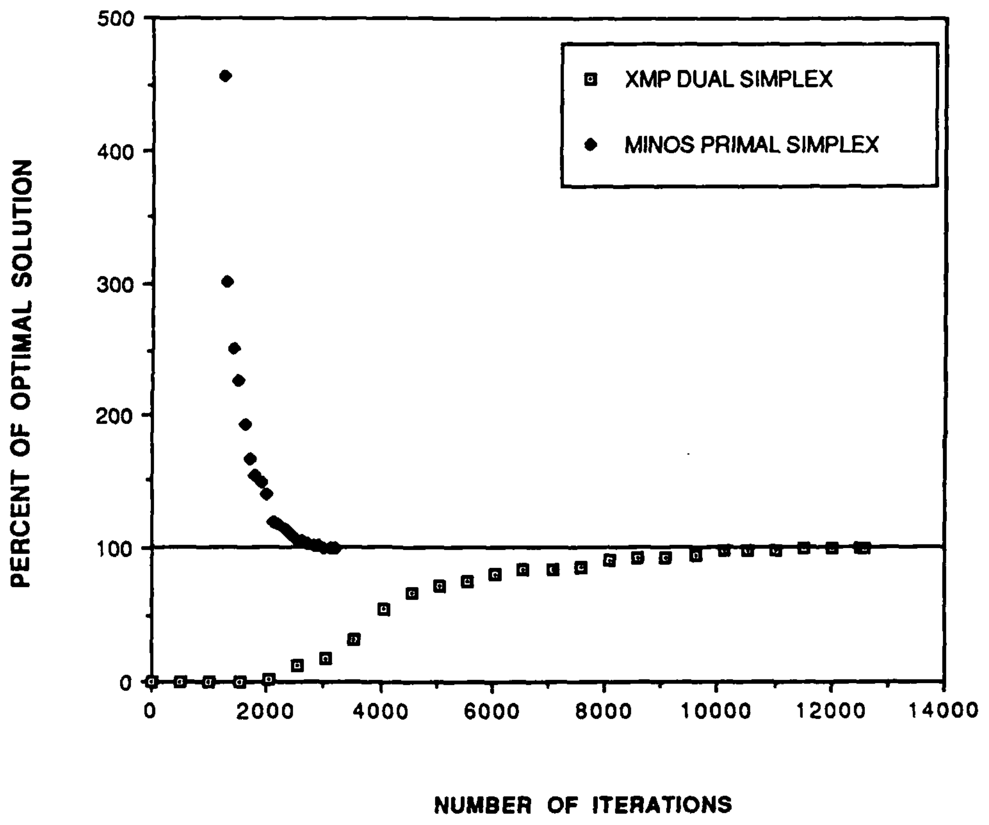


Figure 21. Linear Programming Relaxation Convergence

Veiga, University of California, Berkeley, must be credited with applying the MINOS code to our GMF-A problems). MINOS is a FORTRAN-based large-scale linear or nonlinear programming solver which employs the primal simplex method. It also uses the sparse LU factorization of the basis matrix, but has additional Markowitz ordering schemes and improved Bartels-Golub updates (Gill, et al. 1987). These and many other features make the MINOS program superior to the 1986 version of XMP. The primal simplex iterations seek to minimize the objective function as shown in Figure 21. This graphical comparison also reflects the faster convergence of the MINOS implementation which achieved the optimal solution in 3,208 iterations. The MINOS effort, while encouraging, also revealed that the basis matrix was ill-conditioned indicating that such convergence should not always be expected.

The optimal solution to the LP relaxation does not answer the equipment selection and scheduling problem. It is the repeated solution of these relaxations with successively imposed integrality constraints which form the working elements of the required implicit enumeration to achieve an optimal solution for the MILP. Because the MINOS package does not support a MILP branch and bound technique and the XMP dual solution requires such long computing times, more work will be needed in order to demonstrate that these MILP formulations can be efficiently solved. The LP results indicate, however, that the MILP models are indeed tractable.

Chapter 9. Further MILP Modelling Capabilities

Two further modelling capabilities, directly related to the presented formulations, are documented in this section. These formulations were explored but not implemented. First, the MILP problems may be converted to equivalent binary integer linear programming (BILP) models. The latter are solvable using standard BILP packages to perform the implicit enumeration. The GMF problems may be good candidates for this type of approach as discussed in the following section. The second capability is to incorporate equipment type selection, where the MILP solution indicates the *optimal type* of equipment to install at a processing station. This is discussed in 9.2.

9.1 Conversion to a BILP

The conversion of a MILP problem to BILP involves the replacement of the general integer variables with equivalent binary variable expressions. This is easily done if the range on each of the original integer variables is tightly bounded. The GMF equipment selection and scheduling problems are attractive candidates for BILP modelling because reasonable bounds can be provided on the integer number of machines to be installed. For instance, a postal manager states, by means of a simple analysis, that at least three FCEs are needed but certainly no more than seven will be required. For this processing station, the resource acquisition constraint (Eq. 17) is restated as

$$1Y_{n1} + 2Y_{n2} + 4Y_{n3} \geq y_n(t), \text{ all } t \epsilon T_n \qquad (21)$$

$$\text{and } 3 \leq 1Y_{n1} + 2Y_{n2} + 4Y_{n3} \leq 7$$

$$\text{where } Y_{ni} = 0,1 \text{ for } i = 1, 2, 3$$

For example, a solution requiring six machines would have $Y_{n1}=0$, $Y_{n2}=1$, and $Y_{n3}=1$. These binary variables and their coefficients must also appear in the objective function.

Such a mixed binary problem can be directly considered by a BILP package such as XMP's Zero One Optimization Methods (ZOOM, see Singhal, Marsten, and Marin, 1989). These packages automatically manage the binary branching and solve the associated LP relaxation for bounds.

9.2 Equipment and Configuration Choice Model

In the formulations presented in this report, equipment types with identified operating parameters have been specified for each processing station. Suppose now that at one or more processing stations there are several alternative types of equipment which are competitive options for installation. The GMF MILP formulation can be extended to enable the selection of the optimal equipment choice. For a pertinent example, consider the functions performed by the facer-canceller enricher which can also be performed by the combined functions of the existing facer-cancellers and the MLOCRs. The problem is to choose the optimal equipment mix and schedule, with respect to total cost during some defined period. Such a problem may be represented as a MILP by extending the proposed GMF model.

If, for example, one desired to choose the optimal equipment type at some station n, given that the operating parameters for each of α alternatives have been specified, the processing rate constraint (Eq. 4) can be replaced by the set of equations

$$s_{i\alpha}(t) \le r_{in\alpha} \cdot w_{i\alpha}(t), \text{ all } \alpha, \text{ all } i \epsilon I_n, \text{ all } t \epsilon T_n \qquad (22)$$

$$s_{i\alpha}(t) \le \mu \, Z_{n\alpha}, \text{ all } \alpha, \text{ all } i \epsilon I_n, \text{ all } t \epsilon T_n \qquad (23)$$

$$\sum_{\text{all } \alpha} Z_{n\alpha} \le 1 \quad Z_{n\alpha} = 0,1 \text{ all } \alpha, \text{ all } n$$

where μ is a very large constant. At each station n, only one $s_{i\alpha}(t)$ will be nonzero. $Z_{n\alpha}$ is a binary decision variable which indicates the selected alternative (α*) at n (where $Z_{n\alpha} = 1$ for α*, = 0 otherwise). Since we are minimizing the number of machines (and therefore all

$w_{i\alpha}(t)$), the corresponding selected $w_{i\alpha*}(t)$ will also be the only nonzero valued of the $w_{i\alpha}(t)$. The original term $w_i(t)$ must be replaced by the sum, over α, of the $w_i(t)$ in Eq. 3.3 and the sum of the $s_{i\alpha}(t)$ must replace $s_i(t)$ in Eq. 7.

Since the alternative equipment may have different branching proportions, conservation of processed mail volume (Eq. 3.1) is restated as

$$x_{on}(t+1) = \sum_{i \in I_n} \sum_{\text{all } \alpha} [f_{ion\alpha} \cdot s_{i\alpha} t)], \quad \underset{\text{all } o \in O_n}{n=1,\ldots,N} \quad (24)$$

The following equation is also needed in order to compose the objective function.

$$Y_{n\alpha} \leq \mu Z_{n\alpha}$$

The $Y_{n\alpha}$ are then included in the objective function with their appropriate cost coefficients.

Obviously, many additional constraints and variables have been added for this formulation. The benefit is additional modelling capability for optimizing trade-off problems in equipment selection and operating configuration. The same type of modelling structure as described can be applied to processing schemes where the equipment characteristics may be the same, but mail stream branchings differ. The core GMF MILP model can, thus, be readily expanded to enable optimal selection of processing resource types and processing schemes.

Chapter 10. Conclusions and Extensions

The comprehensive models developed in this effort have immediate applicability for assisting in the strategic planning and day-to-day operations management of semi-automated mail processing systems. The simulation model offers a testbed for candidate equipment selection and scheduling solutions, and incorporates both the realistic randomness and the dynamic interaction of mail flowing through the system. Various "what if?" scenarios may be analyzed to investigate the impact of future changes in input volumes, automation technologies, or other variations.

The scenarios and results documented are demonstrative of the wide range of possibilities afforded by the simulation model. For given arrival processes, equipment types, and operating parameters, the model provides an effective method for testing the feasibility of resource schedules. A successful resource schedule is achieved by iteratively simulating the system and adjusting the equipment allocations based on the results.

In the simulation analysis, the baseline resource schedule for the GMF was shown to be sensitive to perturbations in both daily mail volume and mail type profile. Even modest increases in mail volume overwhelmed the automated equipment and the RVES stations. Increases in percent automated mail, with corresponding decreases in mechanized and manual mail, resulted in bottlenecks at both the incoming and outgoing MLOCRs. Such increases in the percent of automated mail will surely be experienced if the USPS is to bar-code all letter mail by 1995.

By employing user-written event subroutines linked with the simulation model, changes to system parameters are possible by adjusting values in the input files without altering the process network. This suggests that an effective user-interface should be explored to enable the model to be executed by facilities managers who need not be simulation experts. The result will be a real-time tool for addressing on-site operational scheduling and trade-off studies. A single general model structure could be customized for different GMFs.

By casting the letter mail processing system as a multiperiod network, an extensive mathematical programming formulation has been developed which has few integer variables. The large sparse linear relaxations of these MILPs were solved to optimality by XMP dual simplex and MINOS primal simplex. MINOS appears to be a more effective solver for these degenerate, ill-conditioned problems if appropriate pre-conditioning is applied. The results of the MILP studies indicate that, with some effort, this class of difficult MILPs could become an effective tool for long-term planning and implementation of automation at postal GMFs.

The MILP and simulation models are complementary approaches in the study of these processing systems. Simulation should be used to test the scheduling results given by the deterministic model to ensure feasibility and stochastic stability.

Modelling greater system detail, including the sequencing process and materials handling methods, is within the range of current simulation capabilities. The degree of detail represented in the current models is mainly constrained by the availability of data to describe the system. Better modelling of postal GMF operations and validation of the results will require more information than is currently published in the public domain. Consequently, such improvements will require an effort more closely coordinated with the relevant USPS entities.

Letter mail processing is only one part of the GMF operations. Further, the GMFs are encompassed by the regional and national distribution systems. While no small task, deterministic and probabilistic modelling of the regional distribution scheme is of value. These more ambitious models could address issues such as the trade-offs between operations at the centralized GMFs versus a division of functions at local delivery units. The boundary conditions for the GMF models, for example the arrival profiles and dispatch times, could be subject to change as the GMF models are subsumed into a regional processing model.

Quantitative results aside, perhaps the most beneficial outcome of such modelling efforts are the insights gained from the formulation of, and experimentation with, a complex system which one can ill-afford to disturb in actuality. Such models, having been shown solvable, should not be shelved in the haste to tackle other challenging problems. Extensive experience with these mathematical abstractions and their interesting solutions will lead to many practical cost-saving heuristics for postal processing and similar operations.

Appendix A. Simulation Model Computer Code*

*see Figure 8 for simulation program structure

```
      PROGRAM MAIN
C
C  program MAIL.FOR is the main user-written code for the SLAMII network model
C  POSTAL.DAT.  The main program below calls SLAM to execute the network
C  model.  EVENT node calls from SLAM to SUBROUTINE EVENT initialize the
C  global XX(I) variables for machine rates and delay times (XX.DAT), the
C  time between arrivals for the mailstreams - the TBC values (ARRIV.DAT), and
C  ARRAY values (branching probabilities at the various machines) from
C  ARRAY.DAT {.DAT files are local data files}.  All of the TBC-values
C  are changed every 30 min to match the arrival profiles for the mail streams
C  by calls to EVENT node 3.  Intermediate SUMMARY reports are generated by
C  scheduled calls to EVENT node 2 {Wert, UT OR/IE Dept}
C
C      event nodes:
C        Node 1:  initialize XX(I) and ARRAY(I,J)--go to node 3
C        Node 2:  intermediate SUMRY reports
C        Node 3:  update TBC values each period
C        Node 4:  output resource utilization data
C        Node 5:  clear statistical arrays
C        Node 6:  periodic bookkeeping for resource statistics
C
C---------------------------------------------------------------------------
C  MAIN PROGRAM
C
C
      DIMENSION NSET(50000)
      PARAMETER (KVACP1=1,KVACP2=2,KVALEN=3,KVACAP=4,KVADIR=5,KVANTX=6,
     $ KVANCO=7,KVASTO=8,KVAMNO=9,KVATLU=10,KVAPFE=11,KVAPLE=12,
     $ KVCSPR=1,KVCNWL=2,KVCPFE=3,KVCPLE=4,KVCMXL=5,KVCPPL=6,
     $ KVCCRC=7,KVCNTE=8,KVCCST=9,KVCSTB=10,KVCTLU=11,KVCVCO=12,
     $ KVCRRC=13,KVCCPI=14,KVSESP=1,KVSLSP=2,KVSACC=3,KVSDEC=4,KVSLEN=5,
     $ KVSBUF=6,KVSCKZ=7,KVSIFL=8,KVSJRQ=9,KVSINI=10,KVSREP=11,
     $ KVSNTL=12,KVSNTU=13,KVSNUL=14,KVSNUU=16,KVSNUE=18,KVSNUF=20,
     $ KVSNUC=22,KVSNUS=24,KVSSTE=26,KVSSTF=27,KVSTLU=28,KVSPFE=29,
     $ KVSPLE=30,KVSPAL=31,KVSNOV=32,KVUPVS=1,KVUCSG=2,KVUCCP=3,
     $ KVUICP=4,KVUDCP=5,KVUPCL=6,KVUVMO=7,KVUCSI=8,KVUSPD=9,KVUCBF=10,
     $ KVUCBT=11,KVUTLU=12,KVUSCP=13,KVUSHT=14,KVUNTL=15,KVUSPT=16,
     $ KVFVSI=4,KVWIFL=4,KVWVSI=5,KVWVRQ=6,KVWVRE=7,
     $ KVWCPI=8,KVMCPI=4,KVANWF=13,KVCNWF=15,KVFNWF=5,KVMNWF=4,
     $ KVSNWF=33,KVUNWF=15,KVWNWF=8)
      PARAMETER (MXMSG=250)
      PARAMETER (MXPOUT=6)
      INCLUDE 'PARAM.INC'
      COMMON/SCOM1/ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II, MFA,
     1MSTOP,NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
     2SSL(MEQT),TNEXT, TNOW, XX(MMXXV)
      COMMON QSET(50000)
      COMMON ISWITCH
      EQUIVALENCE (NSET(1),QSET(1))
      NNSET=50000
      NCRDR=5
      NPRNT=6
      NTAPE=7
      ISWITCH=0
      CALL SLAM
      STOP
      END
C-----------------------------------------------------------------------
C
      SUBROUTINE EVENT(I)
C
      INCLUDE 'PARAM.INC'
C
      COMMON/SCOM1/ ATRIB(MATRB), DD(MEQT), DDL(MEQT), DTNOW, II, MFA,
     1MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(MEQT),
     2SSL(MEQT), TNEXT, TNOW, XX(MMXXV)
C
C
      REAL DUMMY(9,47),XPRO(36,3),ITVOL(20),IPROF(20),SUBTOT(20)
      REAL RUTIL1(14),RTIME1(14),PERUTIL,XTEMPOR,TOTUTIL(14)
      INTEGER ICAP,TOTCAP(14)
C
C                                   six event nodes
      GO TO (1,2,3,4,5,6),I
C
C************** node 1 initializes the SLAMII XX(I) & ARRAY    ***************
C************** and sneaks through node 3 code below to set    ***************
C************** TBC values for the first hour before returning ***************
C
C                                    XX(I) data are in local file XX.DAT
C                                    ARRAY data are in local file ARRAY.DAT
C                                    ARRIVAL TBC data are in file ARRIV.DAT
C
1     OPEN(3,FILE='XX.DAT',STATUS='OLD')
      OPEN(4,FILE='ARRAY.DAT',STATUS='OLD')
      OPEN(8,FILE='ARRIV.DAT',STATUS='OLD')
C
C                if not first time...skip over reading most stuff...
      IF (ISWITCH.GT.0) GO TO 17
C                first time through open resource data output file
      OPEN(9,FILE='RESOUR.OUT',STATUS='NEW')
C
C                                    skip file header on XX.DAT
      READ(3,*)
      READ(3,*)
      READ(3,*)
      READ(3,*)
C                                    first XX(I) values are the machine
C                                    processing rates (min/kp)
      DO 100 I=1,15
      READ(3,700) XX(I)
100   CONTINUE
C
      DO 110 I=63,68
      READ(3,700) XX(I)
110   CONTINUE
C
      READ(3,*)
      READ(3,*)
C                                  read in cart (BATCH) size for each machine
      DO 300 I=40,52
      READ(3,700) XX(I)
```

```
300   CONTINUE
C
C                                   XX(I) variables are initiali~su
C
C      WRITE(*,*) 'XX(I) values read by MAIL.FOR from XX.DAT'
C
C            ------------------------ branching array data -------------
C
      READ(4,*)
      READ(4,*)
C                                     first, read array data into DUMMY
      DO 400 J=1,47
        READ(4,*)
        READ(4,*)
        DO 450 I=1,9
          READ(4,710) DUMMY(I,J)
450     CONTINUE
400   CONTINUE
C
C                                    ARRAY data should sum to 1 for each column
C                                    if not, print a nasty error message...
      DO 500 J=1,47
        TEMP=0.0
        DO 550 I=1,9
          TEMP=TEMP+DUMMY(I,J)
550     CONTINUE
C
        IF(TEMP.GT.0.99999.AND.TEMP.LT.1.00001) GO TO 500
C       WRITE(*,*) 'ERROR IN DATA FROM ARRAY.DAT'
C       WRITE(*,*) 'COL ',J,' PROB SUMS TO ',TEMP
500   CONTINUE
C
C      WRITE(*,*) 'ARRAY DATA read by subr EVENT from ARRAY.DAT'
C
C                          call SLAM subrt PUTARY to put DUMMY data into ARRAY
      DO 600 J=1,47
        DO 650 I=1,9
          CALL PUTARY(I,J,DUMMY(I,J))
650     CONTINUE
600   CONTINUE
C
C            ---------------------- arrival stream profiles ------------
C
17    CONTINUE
C                          ready to read ARRIV.DAT -- skip big header
      READ(8,*)
      READ(8,*)
      READ(8,*)
      READ(8,*)
      READ(8,*)
      READ(8,*)
      READ(8,*)
      READ(8,*)
      READ(8,*)
      READ(8,*)
C                           IENT switch means entity is kp (1) or tray (2)
```

```
C                              IMET switch means deterministic input volumes (1)
C                                   or daily volumes are random variables (2)
      READ(8,*)
      READ(8,*) IENT
      READ(8,*)
      READ(8,*) IMET
      READ(8,*)
      IF (ISWITCH.GT.0) GO TO 18
C
C                              adjust processing rates to reflect entity value
C                              the first time through here only...
      DO 345 I=1,15
        XX(I)=XX(I)/IENT
345   CONTINUE
C
      DO 347 I=40,52
        XX(I)=XX(I)*IENT+IENT-1
347   CONTINUE
C
      DO 346 I=63,68
        XX(I)=XX(I)/IENT
346   CONTINUE
C
18    CONTINUE
      ISWITCH=5
C
C                              read in total daily mail stream volumes from
C                              ARRIV.DAT and put in ITVOL(20)...compute total
      CHKVOL=0.
      DO 630 I=1,20
        READ(8,*) ITVOL(I),IPROF(I)
        ITVOL(I)=ITVOL(I)*IENT
        IF (IMET.EQ.1) GO TO 629
C                                      here's where the total volume of each
C                                      input stream is selected from a prob.
C                                      distribution each day (if IMET=2)...
C                                      any SLAMII distrib. function can be
C                                      used here...
          XMED=0.8*ITVOL(I)
          XSTD=0.25*XMED
          IVL=RNORM(XMED,XSTD,5)
          ITVOL(I)=IVL
C
629     CHKVOL=ITVOL(I)+CHKVOL
630   CONTINUE
C                                      print total daily mail volume
C
      WRITE(*,*) 'arrival profile data read from ARRIV.DAT'
      WRITE(*,*) 'TOTAL MAIL VOLUME= ',CHKVOL
C
C      DO 447 I=1,20
C        WRITE(*,*) 'IPROF(',I,')=  ',IPROF(I)
C447   CONTINUE
C
      READ(8,*)
      READ(8,*)
      READ(8,*)
      READ(8,*)
C                                read in the cumulative percentage profiles from
C                                ARRIV.DAT into XPRO array
      DO 750 J=1,34
        READ(8,*) (XPRO(J,I),I=1,3)
750   CONTINUE
C                                zero out the SUBTOT vector for use in EVENT node 3
      DO 751 I=1,20
        SUBTOT(I)=0.0
751   CONTINUE
C                                zero out resource data vectors for EVENT 4
      DO 752 I=1,14
        RUTIL1(I)=0.0
        RTIME1(I)=0.0
752   CONTINUE
C
      CLOSE(3)
      CLOSE(4)
      CLOSE(8)
C                                fix IND to set initial TBC-values and go to node 3
      IND=1
      GO TO 444
C
700   FORMAT(F8.4)
710   FORMAT(F6.4)
C
C
C********** node 2 causes intermediate summary reports to be printed *********
C
2     CALL SUMRY
      WRITE(*,*) 'SUMMARY REPORT OUTPUT AT TIME ',TNOW
      RETURN
C
C
C********** node 3 updates the TBC values for the stream profiles ************
C
C  The ATRIB(3) of the entity setting off the EVENT node 3 will be the
C  current time period.  Just before each period, the TBC values are reset.
C  Coding here is a little tricky...
C
C
3     IND=ATRIB(1)
444   CONTINUE
C
      DO 997 J=1,20
C
          IF(IND.LT.34) GO TO 13
          XX(J+79)=35.
          GO TO 997
C
13        TMP=ITVOL(J)*XPRO(IND,IPROF(J))-SUBTOT(J)
          IF(TMP.LT.1) XX(J+79)=35.
          IF(TMP.LT.1) GO TO 997
C
          ITMP=TMP
          XX(J+79)=30./ITMP
C
          IF(XX(J+79).GT.30) GO TO 997
C
          SUBTOT(J)=SUBTOT(J)+ITMP
C
997   CONTINUE
C
C
C      WRITE(*,637) TNOW,IND
C637   FORMAT(//,' TBC VALUES RESET AT TNOW=',F8.1,'  IND=',I3,/)
C      WRITE(*,*) 'CUMULATIVE TOTAL STREAM VOLUMES BY TNOW+30:'
C      WRITE(*,*) '       STREAM    CUMM VOL        TOTAL VOL'
C      WRITE(*,*) '       ------    --------        ---------'
C
C      DO 993 I=1,20
C       WRITE(*,*) I,SUBTOT(I),ITVOL(I)
C993   CONTINUE
C
1000  RETURN
C
C
C********************EVENT NODE 4********************************
C
C  Kleaver Skeems accessing SLAMII resource statistical functions to
C  compute detailed resource utilization data for intervals throughout
C  the day...
C
C  calls are made to SLAMII variables:
C     NNRSC(I)  number of resource I currently available
C     NRUSE(I)  number of resource I currently in use
C
C  and to SLAMII statistical calculation functions:
C     RRAVG(I)  average utilization of resource I
C     RRPRD(I)  time period for statistics on resource I
C
C  Local subroutine EVENT variables store the values from
C  the previous period:
C     RTIME1(I) = RRAVG(I) at t-1
C     RUTIL1(I) = RRPRD(I) at t-1
C
C
4     CONTINUE
      T1=TNOW-30+0.02
      T2=TNOW+0.02
      WRITE(9,241) T1,T2
241   FORMAT(//,' RESOURCE DATA FROM 'F7.2,' TO ',F7.2,/)
      WRITE(9,*) ' RES #   CAPACITY  AVG UTIL   %UTIL '
C
      DO 243 I=1,14
        UTIL=(RRAVG(I)*RRPRD(I)-RUTIL1(I)*RTIME1(I))/29.98
        ICAP=NNRSC(I)+NRUSE(I)
        IF(ICAP.EQ.0) THEN
           PERUTIL=0.0
         ELSE
           PERUTIL=UTIL/ICAP
         ENDIF
        WRITE(9,244) I,ICAP,UTIL,PERUTIL
244     FORMAT(3X,I2,5X,I3,5X,F7.4,5X,F7.4)
        RUTIL1(I)=RRAVG(I)
        RTIME1(I)=RRPRD(I)
        TOTCAP(I)=TOTCAP(I)+ICAP
        TOTUTIL(I)=PERUTIL*ICAP+TOTUTIL(I)
243   CONTINUE
C
      RETURN
C
C***************************EVENT NODE 5*********************************
C
C  Start a brand new day!  Clears SLAM statistics arrays after summary
C  report is generated.  Writes start of new day to resource data
C  output file...
C
5     CONTINUE
      WRITE(9,591)
591   FORMAT(/,' ########################################## ',/,
     X  ' DAILY RESOURCE STATISTICS: ',//,' RESOURCE   AVAILABLE',
     X  '   UTILIZED    OVERALL ',/,' NUMBER   MACHINE-HRS ',
     X  'MACHINE-HRS %UTILIZED ')
C
      DO 552 I=1,14
        XTEMPOR=TOTUTIL(I)/TOTCAP(I)
        TOTCAP(I)=TOTCAP(I)/2
        TOTUTIL(I)=TOTUTIL(I)/2
        WRITE(9,553) I,TOTCAP(I),TOTUTIL(I),XTEMPOR
553     FORMAT(3X,I2,9X,I5,8X,F7.1,5X,F7.4)
        TOTCAP(I)=0
        TOTUTIL(I)=0.0
552   CONTINUE
C
      CALL CLEAR
C
      WRITE(9,245) XX(71)+2
245   FORMAT(//,' ============================== DAY ',F4.1)
C
      RETURN
C
C*************************** EVENT NODE 6 ******************************
C
C  Periodic bookkkeeping needed to reset resource stat variables
C    RUTIL1(I) and RTIME(I) at the beginning of every 30-minute period
C
6     CONTINUE
      DO 571 I=1,14
        RUTIL1(I)=RRAVG(I)
        RTIME1(I)=RRPRD(I)
571   CONTINUE
C
      RETURN
      END
```

```
GEN,SDWERT,GMFA POSTAL FACILITY,11/4/89,1;                         POST.DAT
;
;****  The SLAMII network model for simulation of a USPS sectional      ****
;****  center facility.  This model is executed using the user-written  ****
;****  main FORTRAN driver MAIL.FOR.  SLAMII global variables XX(I) and ****
;****  ARRAY are initialized by MAIL.FOR and the EVENT subroutine.      ****
;****  Data are read in from files XX.DAT, ARRAY.DAT, and ARRIV.DAT.    ****
;
; [limits: largest file #, # of atribs, max concurrent entities]
;
LIMITS,27,3,2500;
;
;                    initialize ARRAY(9,47)
;
ARRAY(1,47)
ARRAY(2,47)
ARRAY(3,47)
ARRAY(4,47)
ARRAY(5,47)
ARRAY(6,47)
ARRAY(7,47)
ARRAY(8,47)
ARRAY(9,47)
;
;  labels assigned for attributes...MAILSTREAM designates the original
;  stream from which the entity originated; CURRENT_ID is used to
;  distinguish later separation characteristics (i.e. a entity may be
;  labeled as "OCR reject" in current id, but may have originated from
;  one of several different mailstreams)...
;
EQUIVALENCE/ATRIB(1),MARKTIME/ATRIB(2),MAILSTREAM/ATRIB(3),CURRENT_ID;
;
;  machine processing rates (min/entity) are assigned labels based on the
;  machine type and network block number...values are read from XX.DAT
;
EQUIVALENCE/XX(1),FCEB1_RATE/XX(2),OCRB2_RATE/XX(3),BCSB3_RATE/
      XX(4),VCDB4_SLOW/XX(5),VCDB4_FAST/XX(6),BCSB5_RATE/
      XX(7),BCSB5A_RATE/XX(8),MANB6A_RATE/XX(9),OCRB6_RATE/
      XX(10),VCDB7_RATE/XX(11),MANB9_RATE/XX(12),MANB10_RATE/
      XX(13),LSMB13_OUT/XX(14),LSMB13_INC/XX(15),LSMB14_RATE;
EQUIVALENCE/XX(63),VCDB4_PROB/XX(64),VCDB4_OUTG/XX(65),VCDB4_SEQN/
      XX(66),VCDB4_RURA/XX(67),VCDB4_FIRM/XX(68),BCSB81_RATE;
;
;
;  cart counters are used to permit a batching of entities before forwarding
;  processed mail to the next machine (using GATE nodes)...initial values
;  are read from XX.DAT.
;
EQUIVALENCE/XX(20),FCEB1CART/XX(21),OCRB2CART/XX(22),BCSB3CART/
      XX(23),VCDB4CART/XX(24),BCSB5CART/XX(25),BCSB5ACART/
      XX(26),MANB6ACART/XX(27),OCRB6CART/XX(28),VCDB7CART/
      XX(29),MANB9CART/XX(30),MANB10CART/XX(31),LSMB13CART/
      XX(32),LSMB14CART;
;
;  cart sizes (number of entities batched before sending mail to the next
;  machine)...variable values are read from XX.DAT
```

```
;
EQUIVALENCE/XX(40),FCEB1FULL/XX(41),OCRB2FULL/XX(42),BCSB3FULL/
      XX(43),VCDB4FULL/XX(44),BCSB5FULL/XX(45),BCSB5AFULL/
      XX(46),MANB6AFULL/XX(47),OCRB6FULL/XX(48),VCDB7FULL/
      XX(49),MANB9FULL/XX(50),MANB10FULL/XX(51),LSMB13FULL/
      XX(52),LSMB14FULL;
;
;  a few more variables for delay time between machines and counter for
;  incoming mail entities...
;
EQUIVALENCE/UNFRM(10,15,1),DELAY/XX(61),IN_CNT/XX(71),DAYNUM/XX(72),TCUR;
;
;
;  and the variables for arriving profiles are "TBC" followed by the stream
;  number...these values are updated in the EVENT node 3...
;
EQUIVALENCE/XX(80),TBC1/XX(81),TBC15/XX(82),TBC16/XX(83),TBC12/
      XX(84),TBC13/XX(85),TBC53/XX(86),TBC54/XX(87),TBC55;
EQUIVALENCE/XX(88),TBC56/XX(89),TBC57/XX(90),TBC58/XX(91),TBC83/
      XX(92),TBC154/XX(93),TBC2/XX(94),TBC4/XX(95),TBC5/
      XX(96),TBC6/XX(97),TBC10/XX(98),TBC11/XX(99),TBC75;
;
TIMST,XX(61),INPUT ENTITIES;
;
;##############################################################################
;##############################################################################
;
NETWORK;
;
;************************************************************ resources ********
;
      RESOURCE/FCEB1(0),1;                FCE machines in B1
      RESOURCE/OCRB2(0),2;                OCR machines in B2
      RESOURCE/BCSB3(0),3;                BCS1 machines in B3
      RESOURCE/VCDB4(0),4;                outgoing VCD in B4
      RESOURCE/BCSB5(0),5;                incoming primary BCS1 in B5
      RESOURCE/BCSB5A(0),6;               incoming secondary BCS1 in B5A
      RESOURCE/MANB6A(0),7;               manual bundle sort in B6A
      RESOURCE/OCRB6(0),8;                incoming OCR in B6
      RESOURCE/VCDB7(0),9;                incoming VCD in B7
      RESOURCE/MANB9(0),10;               primary manual sort in B9
      RESOURCE/MANB10(0),11;              secondary manual sort B10
      RESOURCE/LSMB13(0),12;              LSM machines in block B13
      RESOURCE/LSMB14(0),13;              LSM machines in block B14
      RESOURCE/BCSB81(0),27;              BCS1 machines in block B81
;
;
;
;
;************************************************************ gates ************
;  gates are used to batch entities in carts behind each machine...based on the
;  input cart sizes ("machineFULL"), carts are emptied when full.
;
      GATE/B1G,CLOSE,14;                  FCE cart B1
      GATE/B2G,CLOSE,15;                  OCR cart B2
      GATE/B3G,CLOSE,16;                  BCS cart B3
```

```
        GATE/B4G,CLOSE,17;                   outgoing VCD cart B4
        GATE/B5G,CLOSE,18;                   incoming primary BCS cart B5
        GATE/B5AG,CLOSE,19;                  incoming secondary BCS cart B5A
        GATE/B6AG,CLOSE,20;                  manual bundle sort in cart B6A
        GATE/B6G,CLOSE,21;                   incoming OCR in cart B6
        GATE/B7G,CLOSE,22;                   incoming VCD in cart B7
        GATE/B9G,CLOSE,23;                   primary manual sort in cart B9
        GATE/B10G,CLOSE,24;                  secondary manual sort cart B10
        GATE/B13G,CLOSE,25;                  LSM machine sort cart B13
        GATE/B14G,CLOSE,26;                  LSM machine sort cart B14
;
;
;*********************************************** set off the EVENT node  ***
;
;  create one quiet entity to activate EVENT node 1 and read in ARRAY values
;  from the user's ARRAY.DAT file, read in XX(I) values from XX.DAT, and
;  read in TBC values for the first hour from ARRIV.DAT.  Cycle through EVENT
;  node 2 to trigger summary reports.
;
        CREATE,1439.99,0,1,1,1;
NDE     GOON,1;
        ASSIGN,IN_CNT=0,1;
        EVENT,1,1;
        ACT,1440,,NDE;
;
;
;**************************************** reset the TBC values each hour  ****
;
;  create another lonely entity to reset the TBC values each period to match
;  stream arrival profiles...data were read into FORTRAN vectors in EVENT node
;  1 which set TBC values for the first period.  From period 2 on, EVENT 3
;  resets the TBC values just before each 30-min period ends.
;
        CREATE,1440,0,1,100,1;
        ASSIGN,ATRIB(1)=1,1;
GWY     ASSIGN,ATRIB(1)=ATRIB(1)+1,1;
        ACT,29.99,,;
        EVENT,3,1;
        ACT,0.01,ATRIB(1).LT.34,GWY;
        ACT,0,,TTTT;
;
;
;****************************************  move faster near window  *****
;
;       another loop which opens the GATE at each processing block during the
;       last hour before the sequencing deadline...
;
        CREATE,1440,1020,1,100,1;
        ASSIGN,ATRIB(2)=TNOW+60,1;
        ASSIGN,ATRIB(3)=TNOW+420,1;
GW2     GOON,14;
        ACT,0,,TS2;
        ACT,0,,TS4;
        ACT,0,,TS6;
        ACT,0,,TS8;
        ACT,0,,TS10;
```

```
      ACT,0,,TS12;
      ACT,0,,TS14;
      ACT,0,,TS16;
      ACT,0,,TS18;
      ACT,0,,TS20;
      ACT,0,,TS22;
      ACT,0,,TS24;
      ACT,0,,TS26;
      ACT,UNFRM(5,10,2),TNOW.LT.ATRIB(2),GW2;
      ACT,30,TNOW.LT.ATRIB(3),GW2;
      ACT,0,,TTTT;
;
;
;***************************************************** create mailstreams ***
;
; The start of a brand new day!
;   First, set up the outer loop to set the DAY and trigger initial entities
;   to stream creation blocks below.  DAYNUM is the current day (where day 1
;   is 0, day 2 is 1, etc)...this is used to compute a current time (TCUR)
;   which is the time after 0700 each morning (in min).  TBC values are reset
;   every 30-min in the EVENT node 3 to match arrival profiles.
;
      CREATE,0,0,1,1,1;
      ASSIGN,DAYNUM = -1.0,1;
XEN   GOON,2;
      ACT,1440,,NEX;
      ACT,0,,;
      ASSIGN,DAYNUM=DAYNUM+1.0,1;
      GOON,20;
      ACT,0,,N1;
      ACT,0,,N2;
      ACT,0,,N3;
      ACT,0,,N4;
      ACT,0,,N5;
      ACT,0,,N6;
      ACT,0,,N7;
      ACT,0,,N8;
      ACT,0,,N9;
      ACT,0,,N10;
      ACT,0,,N11;
      ACT,0,,N12;
      ACT,0,,N13;
      ACT,0,,N14;
      ACT,0,,N15;
      ACT,0,,N16;
      ACT,0,,N17;
      ACT,0,,N18;
      ACT,0,,N19;
      ACT,0,,N20;
;
;                          (EVENT 2 summary report, EVENT 5 clears stats)
NEX   EVENT,2,1;
      EVENT,5,1;
      ACT,0,,XEN;
;
;
```

```
;************************************************************* output resource stats ***
;
;
        CREATE,0,0,1,1,1;
DAT     GOON,1;
        ACT,29.98,,;
        EVENT,4,1;
        ACT,0.02,,;
        EVENT,6,1;
        ACT,0,,DAT;
;
;
;
;-----------------------------------------------------------------------
;
;  creating COLLECTION STAMPED mail stream...MAILSTREAM=1
;
N1      ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
        ACT,0,TBC1.LE.30.AND.TCUR.LT.840,SA1;
        ACT,TBC1,TBC1.LE.30.AND.TCUR.LT.840,N1;
        ACT,30,TBC1.GT.30.AND.TCUR.LT.840,N1;
        ACT,0,,TTTT;
        ACT,0,,;
TTTT    TERMINATE;
;
SA1     ASSIGN,MARKTIME=TNOW,MAILSTREAM=1,CURRENT_ID=1,IN_CNT=IN_CNT+1,1;
        ACTIVITY,0,,B1;                        sent to the FCE (block 1)
;
;.......................................................................
;
;  creating METERED COLLECTION (BY-PASS) #1 mail stream...MAILSTREAM=15
;
N2      ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
        ACT,0,TBC15.LE.30.AND.TCUR.LT.840,SA2;
        ACT,TBC15,TBC15.LE.30.AND.TCUR.LT.840,N2;
        ACT,30,TBC15.GT.30.AND.TCUR.LT.840,N2;
        ACT,0,,TTTT;
        ACT,0,,TTTT;
;
SA2     ASSIGN,MARKTIME=TNOW,MAILSTREAM=15,CURRENT_ID=15,IN_CNT=IN_CNT+1,1;
        ACTIVITY,0,,B2;                        sent to the OCR (block 2)
;
;.......................................................................
;
;  creating METERED COLLECTION (BY-PASS) #2 mail stream...MAILSTREAM=16
;
N3      ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
        ACT,0,TBC16.LE.30.AND.TCUR.LT.840,SA3;
        ACT,TBC16,TBC16.LE.30.AND.TCUR.LT.840,N3;
        ACT,30,TBC16.GT.30.AND.TCUR.LT.840,N3;
        ACT,0,,TTTT;
        ACT,0,,TTTT;
;
SA3     ASSIGN,MARKTIME=TNOW,MAILSTREAM=16,CURRENT_ID=16,IN_CNT=IN_CNT+1,1;
        ACTIVITY,0,,B2;                        sent to the OCR (block 2)
;
```

```
;.................................................................
;
; creating METERED PRESORT 5-D BUNDLES #1 mail stream...MAILSTREAM=12
;
N4    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
      ACT,0,TBC12.LE.30.AND.TCUR.LT.1020,SA4;
      ACT,TBC12,TBC12.LE.30.AND.TCUR.LT.1020,N4;
      ACT,30,TBC12.GT.30.AND.TCUR.LT.1020,N4;
      ACT,0,,TTTT;
      ACT,0,,TTTT;
;
SA4   ASSIGN,MARKTIME=TNOW,MAILSTREAM=12,CURRENT_ID=12,IN_CNT=IN_CNT+1,1;
      ACTIVITY,0,,B6A;                      manual bundle sort block 6A
;
;.................................................................
;
; creating METERED PRESORT 5-D BUNDLES #2 mail stream...MAILSTREAM=13
;
N5    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
      ACT,0,TBC13.LE.30.AND.TCUR.LT.960,SA5;
      ACT,TBC13,TBC13.LE.30.AND.TCUR.LT.960,N5;
      ACT,30,TBC13.GT.30.AND.TCUR.LT.960,N5;
      ACT,0,,TTTT;
      ACT,0,,TTTT;
;
SA5   ASSIGN,MARKTIME=TNOW,MAILSTREAM=13,CURRENT_ID=13,IN_CNT=IN_CNT+1,1;
      ACTIVITY,0,,B6A;                      manual bundle sort block B6A
;
;.................................................................
;
; creating INCOMING 3-D MECHANIZED #1 mail stream...MAIL STREAM=55
;
N6    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
      ACT,0,TBC55.LE.30.AND.TCUR.LT.1020,SA6;
      ACT,TBC55,TBC55.LE.30.AND.TCUR.LT.1020,N6;
      ACT,30,TBC55.GT.30.AND.TCUR.LT.1020,N6;
      ACT,0,,TTTT;
      ACT,0,,TTTT;
;
SA6   ASSIGN,MARKTIME=TNOW,MAILSTREAM=55,CURRENT_ID=55,IN_CNT=IN_CNT+1,1;
      ACTIVITY,0,,B6;                       incoming OCR block B6
;
;.................................................................
;
; creating INCOMING 3-D MECHANIZED #2 mail stream...MAIL STREAM=56
;
N7    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
      ACT,0,TBC56.LE.30.AND.TCUR.LT.960,SA7;
      ACT,TBC56,TBC56.LE.30.AND.TCUR.LT.960,N7;
      ACT,30,TBC56.GT.30.AND.TCUR.LT.960,N7;
      ACT,0,,TTTT;
      ACT,0,,TTTT;
;
SA7   ASSIGN,MARKTIME=TNOW,MAILSTREAM=56,CURRENT_ID=56,IN_CNT=IN_CNT+1,1;
      ACTIVITY,0,,B6;                      incoming OCR block B6
;
```

```
;.....................................................................
;
;  creating INCOMING 3-D MECHANIZED #3 mail stream...MAIL STREAM=57
;
N8     ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
       ACT,0,TBC57.LE.30.AND.TCUR.LT.1020,SA8;
       ACT,TBC57,TBC57.LE.30.AND.TCUR.LT.1020,N8;
       ACT,30,TBC57.GT.30.AND.TCUR.LT.1020,N8;
       ACT,0,,TTTT;
       ACT,0,,TTTT;
;
SA8    ASSIGN,MARKTIME=TNOW,MAILSTREAM=57,CURRENT_ID=57,IN_CNT=IN_CNT+1,1;
       ACTIVITY,0,,B6;                      incoming OCR block B6
;
;.....................................................................
;
;  creating INCOMING 3-D MECHANIZED #4 mail stream...MAIL STREAM=58
;
N9     ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
       ACT,0,TBC58.LE.30.AND.TCUR.LT.960,SA9;
       ACT,TBC58,TBC58.LE.30.AND.TCUR.LT.960,N9;
       ACT,30,TBC58.GT.30.AND.TCUR.LT.960,N9;
       ACT,0,,TTTT;
       ACT,0,,TTTT;
;
SA9    ASSIGN,MARKTIME=TNOW,MAILSTREAM=58,CURRENT_ID=58,IN_CNT=IN_CNT+1,1;
       ACTIVITY,0,,B6;                      incoming OCR block B6
;
;.....................................................................
;
;  creating OCR MANAGED BAR-CODED mail stream...MAILSTREAM=63
;
N10    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
       ACT,0,TBC63.LE.30.AND.TCUR.LT.1020,SA10;
       ACT,TBC63,TBC63.LE.30.AND.TCUR.LT.1020,N10;
       ACT,30,TBC63.GT.30.AND.TCUR.LT.1020,N10;
       ACT,0,,TTTT;
       ACT,0,,TTTT;
;
SA10   ASSIGN,MARKTIME=TNOW,MAILSTREAM=63,CURRENT_ID=63,IN_CNT=IN_CNT+1,1;
       ACTIVITY,0,,B6;                      incoming OCR block B6
;
;.....................................................................
;
;  creating MANAGED VCD mail stream...MAILSTREAM=154
;
N11    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
       ACT,0,TBC154.LE.30.AND.TCUR.LT.1020,SA11;
       ACT,TBC154,TBC154.LE.30.AND.TCUR.LT.1020,N11;
       ACT,30,TBC154.GT.30.AND.TCUR.LT.1020,N11;
       ACT,0,,TTTT;
       ACT,0,,TTTT;
;
SA11   ASSIGN,MARKTIME=TNOW,MAILSTREAM=154,CURRENT_ID=154,IN_CNT=IN_CNT+1,1;
       ACTIVITY,0,,B7;                      to VCD block 7
;
```

```
;..................................................................
;
; creating COLLECTION MANUAL mail stream...MAILSTREAM=2
;
N12   ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
      ACT,0,TBC2.LE.30.AND.TCUR.LT.840,SA12;
      ACT,TBC2,TBC2.LE.30.AND.TCUR.LT.840,N12;
      ACT,30,TBC2.GT.30.AND.TCUR.LT.840,N12;
      ACT,0,,TTTT;
      ACT,0,,TTTT;
;
SA12  ASSIGN,MARKTIME=TNOW,MAILSTREAM=2,CURRENT_ID=2,IN_CNT=IN_CNT+1,1;
      ACTIVITY,0,,B9;                        to manual primary
;
;..................................................................
;
; creating 3-D MANUAL #1 mail stream...MAILSTREAM=5
;
N13   ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
      ACT,0,TBC5.LE.30.AND.TCUR.LT.1020,SA13;
      ACT,TBC5,TBC5.LE.30.AND.TCUR.LT.1020,N13;
      ACT,30,TBC5.GT.30.AND.TCUR.LT.1020,N13;
      ACT,0,,TTTT;
      ACT,0,,TTTT;
;
SA13  ASSIGN,MARKTIME=TNOW,MAILSTREAM=5,CURRENT_ID=5,IN_CNT=IN_CNT+1,1;
      ACTIVITY,0,,B9;                        manual sort
;
;..................................................................
;
; creating 3-D MANUAL #2 mail stream...MAILSTREAM=6
;
N14   ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
      ACT,0,TBC6.LE.30.AND.TCUR.LT.960,SA14;
      ACT,TBC6,TBC6.LE.30.AND.TCUR.LT.960,N14;
      ACT,30,TBC6.GT.30.AND.TCUR.LT.960,N14;
      ACT,0,,TTTT;
      ACT,0,,TTTT;
;
SA14  ASSIGN,MARKTIME=TNOW,MAILSTREAM=6,CURRENT_ID=6,IN_CNT=IN_CNT+1,1;
      ACTIVITY,0,,B9;                       manual sort
;
;..................................................................
;
; creating MANAGED MANUAL mail stream...MAILSTREAM=4
;
N15   ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
      ACT,0,TBC4.LE.30.AND.TCUR.LT.1020,SA15;
      ACT,TBC4,TBC4.LE.30.AND.TCUR.LT.1020,N15;
      ACT,30,TBC4.GT.30.AND.TCUR.LT.1020,N15;
      ACT,0,,TTTT;
      ACT,0,,TTTT;
;
SA15  ASSIGN,MARKTIME=TNOW,MAILSTREAM=4,CURRENT_ID=4,IN_CNT=IN_CNT+1,1;
      ACTIVITY,0,,B9;                       manual sort
;
```

```
;..................................................................
;
;  creating 5-D MANUAL MAIL #1 ...MAILSTREAM=10
;
N16    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
       ACT,0,TBC10.LE.30.AND.TCUR.LT.1020,SA16;
       ACT,TBC10,TBC10.LE.30.AND.TCUR.LT.1020,N16;
       ACT,30,TBC10.GT.30.AND.TCUR.LT.1020,N16;
       ACT,0,,TTTT;
       ACT,0,,TTTT;
;
SA16   ASSIGN,MARKTIME=TNOW,MAILSTREAM=10,CURRENT_ID=10,IN_CNT=IN_CNT+1,1;
       ACT,0,,B10;                                    manual sort
;
;..................................................................
;
;  creating 5-D MANUAL MAIL #2 ...MAILSTREAM=11
;
N17    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
       ACT,0,TBC11.LE.30.AND.TCUR.LT.960,SA17;
       ACT,TBC11,TBC11.LE.30.AND.TCUR.LT.960,N17;
       ACT,30,TBC11.GT.30.AND.TCUR.LT.960,N17;
       ACT,0,,TTTT;
       ACT,0,,TTTT;
;
SA17   ASSIGN,MARKTIME=TNOW,MAILSTREAM=11,CURRENT_ID=11,IN_CNT=IN_CNT+1,1;
       ACT,0,,B10;                                    manual sort
;
;..................................................................
;
;  creating MANAGED INCOMPLETE #1...MAILSTREAM=53
;
N18    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
       ACT,0,TBC53.LE.30.AND.TCUR.LT.1020,SA18;
       ACT,TBC53,TBC53.LE.30.AND.TCUR.LT.1020,N18;
       ACT,30,TBC53.GT.30.AND.TCUR.LT.1020,N18;
       ACT,0,,TTTT;
       ACT,0,,TTTT;
;
SA18   ASSIGN,MARKTIME=TNOW,MAILSTREAM=53,CURRENT_ID=53,IN_CNT=IN_CNT+1,1;
       ACT,0,,B6;                                  incoming OCR B6
;
;..................................................................
;
;  creating MANAGED INCOMPLETE #2...MAILSTREAM=54
;
N19    ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
       ACT,0,TBC54.LE.30.AND.TCUR.LT.1020,SA19;
       ACT,TBC54,TBC54.LE.30.AND.TCUR.LT.1020,N19;
       ACT,30,TBC54.GT.30.AND.TCUR.LT.1020,N19;
       ACT,0,,TTTT;
       ACT,0,,TTTT;
;
SA19   ASSIGN,MARKTIME=TNOW,MAILSTREAM=54,CURRENT_ID=54,IN_CNT=IN_CNT+1,1;
       ACT,0,,B6;                                  incoming OCR B6
;
```

```
;..........................................................................
;
;  creating LSM MECHANIZED MMP...MAILSTREAM=75
;
N20   ASSIGN,TCUR=TNOW-1440*DAYNUM,2;
      ACT,0,TBC75.LE.30.AND.TCUR.LT.1020,SA20;
      ACT,TBC75,TBC75.LE.30.AND.TCUR.LT.1020,N20;
      ACT,30,TBC75.GT.30.AND.TCUR.LT.1020,N20;
      ACT,0,,TTTT;
      ACT,0,,TTTT;
;
SA20  ASSIGN,MARKTIME=TNOW,MAILSTREAM=75,CURRENT_ID=75,IN_CNT=IN_CNT+1,1;
      ACT,0,,B13;                          to block 13 LSM
;
;..........................................................................
;
;*************************************************** resource scheduling ***
;
;
;  here, one entity is created and split up to the resource scheduling
;  subnetworks (ALTER networks) to schedule machines each day...
;
      CREATE,1440,0,1,100,1;
      GOON,14;
      ACT,0,,RB1;                        FCE machines block 1
      ACT,0,,RB2;                        OCR machines block 2
      ACT,0,,RB3;                        BCS machines block 3
      ACT,0,,RB4;                        VCD machines block 4
      ACT,0,,RB5;                        BCS machines block 5
      ACT,0,,RB5A;                       BCS machines block 5A
      ACT,0,,RB6A;                       manpower for bundle sort block 6A
      ACT,0,,RB6;                        OCR machines block 6
      ACT,0,,RB7;                        VCD machines block 7
      ACT,0,,RB9;                        manpower for primary manual sort B9
      ACT,0,,RB10;                       secondary manual sort manpower B10
      ACT,0,,RB13;                       LSM machines block 13
      ACT,0,,RB14;                       LSM machines block 14
      ACT,0,,RB81;                       BCS machines block 81
;
;-------------------------------------------------------------------------
;  schedule FCE machines for B1
;
RB1   GOON,1;
      ACTIVITY,270,;        not available until 1130
      ALTER,FCEB1,+4,1;
      ACTIVITY,660,;        operating from 1130 to 2230
      ALTER,FCEB1,-4,1;
      ACT,0.1,,TS2;         open gate after machine shuts down....
;
;-------------------------------------------------------------------------
;  schedule OCR for block B2 (collection mail)
;
RB2   GOON,1;
      ACTIVITY,300,;
      ALTER,OCRB2,+3,1;         not available until 1200
      ACTIVITY,240,;
```

```
          ALTER,OCRB2,+2,1;       add more at 1600
          ACTIVITY,360,;
          ALTER,OCRB2,-3,1;       take machines away at 2200
          ACTIVITY,120,;
          ALTER,OCRB2,-2,1;       last machine gone at 2400
          ACT,0.1,,TS4;
;
;----------------------------------------------------------------------
; schedule BSC for block B3 (collection mail/outgoing secondary)
;
RB3       GOON,1;
          ACTIVITY,300,;
          ALTER,BCSB3,+2,1;        1200 hrs
          ACTIVITY,150,;
          ALTER,BCSB3,-1,1;        1430 hrs
          ACTIVITY,30,;
          ALTER,BCSB3,+1,1;        1500 hrs
          ACTIVITY,90,;
          ALTER,BCSB3,-1,1;        1630 hrs
          ACTIVITY,30,;
          ALTER,BCSB3,+1,1;        1700 hrs
          ACTIVITY,60,;
          ALTER,BCSB3,-1,1;        1800 hrs
          ACTIVITY,90,;
          ALTER,BCSB3,+1,1;        1930 hrs
          ACTIVITY,210,;
          ALTER,BCSB3,-2,1;        2300 hrs
          ACT,0.1,,TS6;
;
;----------------------------------------------------------------------
; schedule outgoing VCD machines in B4
;
RB4       GOON,1;
          ACTIVITY,300,,;
          ALTER,VCDB4,+1,1;        1200 hrs
          ACTIVITY,30,,;
          ALTER,VCDB4,+2,1;        1230 hrs
          ACTIVITY,600,,;
          ALTER,VCDB4,-3,1;        2230 hrs
          ACT,0.1,,TS8;
;
;----------------------------------------------------------------------
; schedule BCS1 machines in B5 (incoming primary)
;
RB5       GOON,1;
          ACTIVITY,240,,;
          ALTER,BCSB5,+1,1;        1100 hrs
          ACTIVITY,450,,;
          ALTER,BCSB5,+1,1;        1830 hrs
          ACTIVITY,60,,;
          ALTER,BCSB5,-1,1;        1930 hrs
          ACTIVITY,210,,;
          ALTER,BCSB5,+2,1;        2300 hrs
          ACTIVITY,60,,;
          ALTER,BCSB5,-3,1;        2400 hrs
          ACT,0.1,,TS10;
```

```
;
;--------------------------------------------------------------------------
;  schedule secondary BCS1 machines in block 5A
;
RB5A  GOON,1;
      ACTIVITY,1020,,;
      ALTER,BCSB5A,+1,1;        2400 hrs
      ACTIVITY,270,,;
      ALTER,BCSB5A,-1,1;        0430 hrs
      ACT,0.1,,TS12;
;
;
;--------------------------------------------------------------------------
;  schedule personnel for manual bundle sort block 6A
;
RB6A  GOON,1;
      ACT,0,,;
      ALTER,MANB6A,+3,1;        0700 hrs
      ACT,210,,;
      ALTER,MANB6A,-1,1;        1030 hrs
      ACT,720,,;
      ALTER,MANB6A,+1,1;        2230 hrs
      ACT,90,,;
      ALTER,MANB6A,-3,1;        2400 hrs
      ACT,0.1,,TS14;
;
;
;--------------------------------------------------------------------------
;  schedule incoming OCR3 machines for block 6
;
RB6   GOON,1;
      ACTIVITY,0,,;
      ALTER,OCRB6,+6,1;         0700 hrs
      ACTIVITY,240,,;
      ALTER,OCRB6,-3,1;         1100 hrs
      ACTIVITY,240,,;
      ALTER,OCRB6,-2,1;         1500 hrs
      ACTIVITY,420,,;
      ALTER,OCRB6,+3,1;         2200 hrs
      ACTIVITY,120,,;
      ALTER,OCRB6,-4,1;         2400 hrs
      ACT,0.1,,TS16;
;
;
;--------------------------------------------------------------------------
;  schedule incoming VCD machines for block 7
;
RB7   GOON,1;
      ACTIVITY,0,,;
      ALTER,VCDB7,+5,1;         0700 hrs
      ACTIVITY,300,,;
      ALTER,VCDB7,-1,1;         1200 hrs
      ACTIVITY,30,,;
      ALTER,VCDB7,-2,1;         1230 hrs
      ACTIVITY,600,,;
      ALTER,VCDB7,+2,1;         2230 hrs
```

```
          ACTIVITY,30,,;
          ALTER,VCDB7,+1,1;           2300 hrs
          ACTIVITY,60,,;
          ALTER,VCDB7,-5,1;           2400 hrs
          ACT,0.1,,TS18;
;
;
;--------------------------------------------------------------------------
;  schedule personnel for primary manual sort block 9
;
RB9       GOON,1;
          ACTIVITY,210,,;
          ALTER,MANB9,+20,1;          1030 hrs
          ACTIVITY,720,,;
          ALTER,MANB9,-11,1;          2230 hrs
          ACTIVITY,90,,;
          ALTER,MANB9,+3,1;           2400 hrs
          ACTIVITY,30,,;
          ALTER,MANB9,-12,1;          2430 hrs
          ACT,0.1,,TS20;
;
;
;--------------------------------------------------------------------------
;  schedule personnel for the secondary manual sort block 10
;
RB10      GOON,1;
          ACTIVITY,930,,;
          ALTER,MANB10,+10,1;         2230 hrs
          ACTIVITY,120,,;
          ALTER,MANB10,+12,1;         2430 hrs
          ACTIVITY,360,,;
          ALTER,MANB10,-22,1;         0630 hrs
          ACT,0.1,,TS22;
;
;
;--------------------------------------------------------------------------
;  schedule LSM machines for block 13
;
RB13      GOON,1;
          ACTIVITY,600,,;
          ALTER,LSMB13,+1,1;          1700 hrs
          ACTIVITY,360,,;
          ALTER,LSMB13,+1,1;          2300 hrs
          ACTIVITY,150,,;
          ALTER,LSMB13,-2,1;          2430 hrs
          ACT,0.1,,TS24;
;
;
;--------------------------------------------------------------------------
;  schedule LSM machines for block 14
;
RB14      GOON,1;
          ACTIVITY,1050,,;
          ALTER,LSMB14,+2,1;          2430 hrs
          ACTIVITY,210,,;
          ALTER,LSMB14,-2,1;          0400 hrs
```

```
     ACT,0.1,,TS26;
;
;-------------------------------------------------------------------------
; schedule BCS machines for B8.1 firm groups sort
;
RB81 GOON,1;
     ACTIVITY,1020,,;
     ALTER,BCSB81,+2,1;          2400 hrs
     ACTIVITY,270,,;
     ALTER,BCSB81,-2,1;          0430 hrs
     TERMINATE;
;
;
;****************************************************************************
;************************ main scf network ***********************************
;****************************************************************************
;
;########## B1 is the FCE for collection stamped mail ######################
;
;    inputs: {1} local collection stamped arrivals
;
B1   AWAIT(1),FCEB1/1,,1;                   waiting for FCE resource
     ACTIVITY/1,FCEB1_RATE,;FCEB1
;                                           FCE_RATE is min/entity
     FREE,FCEB1/1,1;                        done...free the machine
;
     GOON,2;
     ACT,0,,TS1;                            entity is always put in the cart
     ACT,0,FCEB1CART.GE.FCEB1FULL,TS2;      and if the cart is full send an
     ACT,0,,TTTT;                           entity to TS2 to dump the cart
;                                           (OPEN gate B1G).
;
TS1  ASSIGN,FCEB1CART=FCEB1CART+1,1;        putting entity in the cart...track
G1   AWAIT(14),B1G,,1;                      number of entities in the cart
;
     GOON,1;                                dumping the cart...entities
     ACT,0,ARRAY(1,1),AS1;                  will branch to three output streams
     ACT,0,ARRAY(2,1),AS2;
     ACT,0,ARRAY(3,1),AS3;
;
AS1  ASSIGN,CURRENT_ID=14;                  collection automated (OCR-readable)
     ACT,DELAY,,B2;OCR-B2                       sent to the OCR
;
AS2  ASSIGN,CURRENT_ID=152;                 handwritten unreadable sent to
     ACT,DELAY,,B4;VCD                          outgoing VCD
;
AS3  ASSIGN,CURRENT_ID=3;                   physical rejects (oversize) are sent
     ACT,DELAY,,B9;MANUAL                       to manual processing
;
TS2  ASSIGN,FCEB1CART=0.,;                  dumping the full FCE cart (zero the
     OPEN,B1G;                              counter, open the gate for a short
     ACT,0.1,;                              time...
     CLOSE,B1G;
     TERMINATE;
;
;################## B2 is the collection mail OCR ###########################
```

```
;
;       inputs:  {14} collection enriched from FCE_B1, {15} collection metered
;                arrivals #1, and {16} collection metered arrivals #2
;
B2      COLCT,ALL,TOA AT MLOCR_B2,24/60/60,1;
        AWAIT(2),OCRB2/1,,1;                      waiting for the machine if busy
        ACTIVITY/2,OCRB2_RATE,;OCRB2
;                                                 processing according to users
        FREE,OCRB2/1,1;                           input OCRB2_RATE (min/entity)
;
        ASSIGN,CURRENT_ID=CURRENT_ID-12,2;        index id to use ARRAY columns
;                                                 (different ids are preserved)
;
        ACT,0,,TS3;                               entity always put in the cart and
        ACT,0,OCRB2CART.GE.OCRB2FULL,TS4;         if the cart is full another unit
        ACT,0,,TTTT;                              is sent to TS4 to open gate (dump
;                                                 the cart)
;
TS3     ASSIGN,OCRB2CART=OCRB2CART+1,1;          track the number of entities in cart
G2      AWAIT(15),B2G,,1;                        and wait for the gate to open
;
        GOON,1;                                  branching output streams depend
        ACT,0,ARRAY(1,CURRENT_ID),AS4;           on user input probabilities
        ACT,0,ARRAY(2,CURRENT_ID),AS5;
        ACT,0,ARRAY(3,CURRENT_ID),AS6;
        ACT,0,ARRAY(4,CURRENT_ID),AS7;
        ACT,0,ARRAY(5,CURRENT_ID),AS8;
        ACT,0,ARRAY(6,CURRENT_ID),AS9;
        ACT,0,ARRAY(7,CURRENT_ID),AS10;
        ACT,0,ARRAY(8,CURRENT_ID),AS11;
        ACT,0,ARRAY(9,CURRENT_ID),AS12;
;
AS4     ASSIGN,CURRENT_ID=19,;
        ACT,DELAY,,B3;                       secondary OCR to BCS
;
AS5     ASSIGN,CURRENT_ID=21,;
        ACT,0,,OUT;                          dispatch outgoing
;
AS6     ASSIGN,CURRENT_ID=65,;
        ACT,DELAY,,B5;                       inc. auto. BCS for sequencing
;
AS7     ASSIGN,CURRENT_ID=66,;
        ACT,DELAY,,B5;                       inc. auto. BCS mostly rural
;
AS8     ASSIGN,CURRENT_ID=67,;
        ACT,DELAY,,B5A;                      ADC secondary to BCS 5A
;
AS9     ASSIGN,CURRENT_ID=69,;
        ACT,DELAY,,B5A;                      firms 5D to BCS 5A
;
AS10    ASSIGN,CURRENT_ID=73,;
        ACT,DELAY,,B13;                      primary LSM
;
AS11    ASSIGN,CURRENT_ID=153;
        ACT,DELAY,,B4;                       outgoing VCD
;
```

```
AS12  ASSIGN,CURRENT_ID=155;
      ACT,DELAY,,B7;                  incoming VCD
;
TS4   ASSIGN,OCRB2CART=0.,1;        cart is full...zero the counter and dump
      OPEN,B2G;                     the cart (OPEN the gate B2G for a short
      ACT,0.1,,;                    time).
      CLOSE,B2G;
      TERMINATE;
;
;
;#################### B3 is the collection mail BCS #########################
;
;     inputs:  {19}  secondary outgoing from the OCR_B2
;              {20}  secondary outgoing from the VCD_B4
;
B3    AWAIT(3),BCSB3/1,,1;                      waiting on the BCS machines
;                                               processed at _RATE (min/entity)
      ACTIVITY/3,BCSB3_RATE,,;BCSB3
      FREE,BCSB3/1,1;
;
      GOON,2;                                   entity always sent to cart, if
      ACTIVITY,0,,TS5;                          cart is full another unit is
      ACTIVITY,0,BCSB3CART.GE.BCSB3FULL,TS6;    sent to TS6 to dump the cart...
      ACTIVITY,0,,TTTT;
;
TS5   ASSIGN,BCSB3CART=BCSB3CART+1,1;           track number of entities in cart
G3    AWAIT(16),B3G,,1;                         when cart is full gate B3G will
      ACTIVITY,0,ARRAY(1,5),AS13;               be opened and output will
      ACTIVITY,0,ARRAY(2,5),AS14;               branch...
;
AS13  ASSIGN,CURRENT_ID=22,1;
      ACT,0,,OUT;                   outgoing dispatch
;
AS14  ASSIGN,CURRENT_ID=76,1;
      ACT,DELAY,,B13;               primary LSM
;
TS6   ASSIGN,BCSB3CART=0.,1;                    cart is full, zero counter and
      OPEN,B3G,;                                open gate for a short time to
      ACT,0.1,;                                 dump the cart.
      CLOSE,B3G;
      TERMINATE;
;
;
;######################## B4 is the outgoing VCD ############################
;
;     inputs:  {152} machinable non-OCR readable from FCE_B1
;              {153} processing rejects from the OCR_B2
;
B4    AWAIT(4),VCDB4/1,,1;                 waiting for machine availability
      ACTIVITY/11,0,,;VCDB4
      GOON,1;
      ACT,VCDB4_PROB,ARRAY(1,6),AS15;      cart and proceed to be processed
      ACT,VCDB4_OUTG,ARRAY(2,6),AS16;
      ACT,VCDB4_OUTG,ARRAY(3,6),AS17;      processing rate depends on type
      ACT,VCDB4_SEQN,ARRAY(4,6),AS18;      of entity (either _FAST or _SLOW)
      ACT,VCDB4_RURA,ARRAY(5,6),AS19;
```

```
        ACT,VCDB4_FIRM,ARRAY(6,6),AS20;
        ACT,VCDB4_FIRM,ARRAY(7,6),AS21;
        ACT,VCDB4_SEQN,ARRAY(8,6),AS22;
;
AS15    ASSIGN,CURRENT_ID=7,1;              address problems
        ACT,0,,F4;
;
AS16    ASSIGN,CURRENT_ID=20,1;             outgoing BCS B3
        ACT,0,,F4;
;
AS17    ASSIGN,CURRENT_ID=25,1;             outgoing dispatch
        ACT,0,,F4;
;
AS18    ASSIGN,CURRENT_ID=65,1;             BCS turnaround for sequencing
        ACT,0,,F4;
;
AS19    ASSIGN,CURRENT_ID=66,1;             BCS turnaround mostly rural
        ACT,0,,F4;
;
AS20    ASSIGN,CURRENT_ID=67,1;             BCS 5D firms
        ACT,0,,F4;
;
AS21    ASSIGN,CURRENT_ID=69,1;             BCS 5D firms
        ACT,0,,F4;
;
AS22    ASSIGN,CURRENT_ID=73,1;             LSM read but not coded
        ACT,0,,F4;
;
F4      FREE,VCDB4/1,1;
        GOON,2;
        ACT,0,,TS7;
        ACT,0,VCDB4CART.GE.VCDB4FULL,TS8;
        ACT,0,,TTTT;
;
TS7     ASSIGN,VCDB4CART=VCDB4CART+1,1;
G4      AWAIT(17),B4G,,1;
        ACT,DELAY,CURRENT_ID.EQ.7,B9;
        ACT,DELAY,CURRENT_ID.EQ.20,B3;
        ACT,DELAY,CURRENT_ID.EQ.25,OUT;
        ACT,DELAY,CURRENT_ID.EQ.65,B5;
        ACT,DELAY,CURRENT_ID.EQ.66,B5;
        ACT,DELAY,CURRENT_ID.EQ.67,B5A;
        ACT,DELAY,CURRENT_ID.EQ.69,B5A;
        ACT,DELAY,CURRENT_ID.EQ.73,B13;
;
TS8     ASSIGN,VCDB4CART=0.,1;              entity sent here only if cart is
        OPEN,B4G,;                          full...open the gate and dump
        ACT,0.1,,;                          the cart (reset counter to zero)
        CLOSE,B4G,;
        TERMINATE;
;
;
;##################  B5 is Incoming Primary BCS1  ############################
;
;       inputs:  {65} inc automated for sequencing from OCR_B2
;                {65} inc automated for sequencing from VCD_B4
;            {66} inc automated mostly rural routes from OCR_B2
;            {66} inc automated mostly rural routes from VCD_B4
;
B5    AWAIT(5),BCSB5/1,,1;                    wait for machine
      ACTIVITY/4,BCSB5_RATE,,;BCSB5
;                                             processing rate is min/entity
      ASSIGN,CURRENT_ID=CURRENT_ID-58,1;      index counter to find column
;                                               of ARRAY
      FREE,BCSB5/1,2;                         entity always sent on to cart and
      ACT,0,,TS9;                             if the cart is full a copy is
      ACT,0,BCSB5CART.GE.BCSB5FULL,TS10;      sent to TS10 to dump the cart...
      ACT,0,,TTTT;
;
TS9   ASSIGN,BCSB5CART=BCSB5CART+1,1;       track the number of entities in cart
G5    AWAIT(18),B5G,,1;                     and wait for full cart to proceed
      ACT,0,ARRAY(1,CURRENT_ID),AS24;       branching by probabilities
      ACT,0,ARRAY(2,CURRENT_ID),AS25;       placed in the ARRAY by user...
      ACT,0,ARRAY(3,CURRENT_ID),AS26;
      ACT,0,ARRAY(4,CURRENT_ID),AS27;
      ACT,0,ARRAY(5,CURRENT_ID),AS28;
;
AS24  ASSIGN,CURRENT_ID=51,1;
      ACT,0,,RURA;                                rural routes
;
AS25  ASSIGN,CURRENT_ID=70,1;
      ACT,DELAY,,B5A;                             BCS1 secondary sort
;
AS26  ASSIGN,CURRENT_ID=74,1;
      ACT,DELAY,,B13;                             code rejects
;
AS27  ASSIGN,CURRENT_ID=158,1;
      ACT,DELAY,,B81;                             firm groups sort B81
;
AS28  ASSIGN,CURRENT_ID=178,1;
      ACT,DELAY,,B82;                             sequence pass 1 - B82
;
TS10  ASSIGN,BCSB5CART=0,1;                   cart is full, zero counter and
      OPEN,B5G;                               dump the cart by opening the gate
      ACT,0.1,,;                              for a short time...
      CLOSE,B5G;
      TERMINATE;
;
;##########  B5A is the incoming secondary (firms sort) BCS1  ##############
;
;     inputs:  {67}  ADCs secondary from OCR_B2
;              {67}  ADCs secondary from VCD_B4
;              {67}  ADCs secondary from OCR_B6
;              {67}  ADCs secondary from VCD_B7
;              {69}  firms 5D secondary from OCR_B2
;              {69}  firms 5D secondary from VCD_B4
;              {69}  firms 5D secondary from OCR_B6
;              {69}  firms 5D secondary from VCD_B7
;              {70}  secondary sort from BCS_B5
;
B5A   AWAIT(6),BCSB5A/1,,1;                    waiting for the machine
      ACTIVITY/5,BCSB5A_RATE,,;BCSB5A
```

```
;                                                processing at _RATE assigned
      ASSIGN,CURRENT_ID=CURRENT_ID-58,1;         index ID to correct column of
      FREE,BCSB5A/1,1;                              ARRAY
      GOON,2;
      ACT,0,,TS11;                               entity will always be sent thru
      ACT,0,BCSB5ACART.GE.BCSB5AFULL,TS12;       and if the cart is full a copy of
      ACT,0,,TTTT;                               the entity will be sent to TS12 to
;                                                dump the cart
;
TS11  ASSIGN,BCSB5ACART=BCSB5ACART+1,1;       track the number of entities in cart
G5A   AWAIT(19),B5AG,,1;                      wait til the cart is full
      ACT,0,ARRAY(1,CURRENT_ID),AS29;         and branch using ARRAY p's
      ACT,0,ARRAY(2,CURRENT_ID),AS30;
;
AS29  GOON,1;
      ACT,0,CURRENT_ID.GT.10,TT1;
      ACT,0,,;
      ASSIGN,CURRENT_ID=30,1;
      ACT,0,,ADC;                             ADC dispatch
;
TT1   ASSIGN,CURRENT_ID=33,1;
      ACT,0,,FIRM;                            5D firms dispatched
;
AS30  GOON,1;
      ACT,0,CURRENT_ID.GT.10,TT2;
      ACT,0,,;
      ASSIGN,CURRENT_ID=71,1;
      ACT,DELAY,,B13;                         ADC rejects
;
TT2   ASSIGN,CURRENT_ID=72,1;
      ACT,DELAY,,B13;                         5D firms rejects
;
TS12  ASSIGN,BCSB5ACART=0.,1;                    cart is full, zero counter and
      OPEN,B5AG;                                 open the gate for a short time
      ACT,0.1,,;                                 to dump the cart...
      CLOSE,B5AG;
      TERMINATE;
;
;########### B6A is the manual bundle sort (5-D mailer presort  ##############
;
;     inputs:  {12} large mailer presort 5D bundles
;              {13} large mailer presort 5D bundles
;
B6A   AWAIT(7),MANB6A/1,,1;                      waiting for manual processing
      ACTIVITY/6,MANB6A_RATE,;MANB6A
;                                                at user specified _RATE
      FREE,MANB6A/1,1;
      ASSIGN,CURRENT_ID=CURRENT_ID+1,1;
;
      GOON,2;                                 entity will always be placed in cart
      ACT,0,,TS13;                            and if the cart is full a copy
      ACT,0,MANB6ACART.GE.MANB6AFULL,TS14;    will go to TS14 to dump the cart
      ACT,0,,TTTT;                            (open the gate)
;
TS13  ASSIGN,MANB6ACART=MANB6ACART+1,1;       track the number of entities in cart
G6A   AWAIT(20),B6AG,,1;                      and wait for full cart to send
```

```
        ACT,0,ARRAY(1,CURRENT_ID),AS31;
        ACT,0,ARRAY(2,CURRENT_ID),AS32;
        ACT,0,ARRAY(3,CURRENT_ID),AS33;
        ACT,0,ARRAY(4,CURRENT_ID),AS34;
        ACT,0,ARRAY(5,CURRENT_ID),AT35;
        ACT,0,ARRAY(6,CURRENT_ID),AT36;
        ACT,0,ARRAY(7,CURRENT_ID),AT37;
;
AS31    ASSIGN,CURRENT_ID=28,1;
        ACT,0,,ADC;                            dispatch outgoing
;
AS32    ASSIGN,CURRENT_ID=31,1;
        ACT,0,,FIRM;                           dispatch 5D firms
;
AS33    ASSIGN,CURRENT_ID=49,1;
        ACT,0,,RURA;                           rural routes
;
AS34    ASSIGN,CURRENT_ID=59,1;
        ACT,DELAY,,B6;                         incoming OCR
;
AT35    ASSIGN,CURRENT_ID=60,1;
        ACT,DELAY,,B6;                         incoming OCR
;
AT36    ASSIGN,CURRENT_ID=61,1;
        ACT,DELAY,,B6;                         incoming OCR
;
AT37    ASSIGN,CURRENT_ID=62,1;
        ACT,DELAY,,B6;                         incoming OCR
;
TS14    ASSIGN,MANB6ACART=0.,1;       cart is full, reset counter and open the
        OPEN,B6AG;                    gate for a short time to dump the cart
        ACT,0.1,,;
        CLOSE,B6AG;
        TERMINATE;
;
;
;##############  B6 is the incoming OCR3 machine in block 6  ################
;
;       inputs:  {53} managed incomplete barcode arrivals
;                {54} managed incomplete barcode arrivals
;                {55} ¦
;                {56} ¦  incoming 3D arrivals
;                {57} ¦
;                {58} ¦
;                {59}      ¦
;                {60}      ¦  from bundle sort B6A
;                {61}      ¦
;                {62}      ¦
;                {63} managed bar-coded
;
B6      AWAIT(8),OCRB6/1,,1;                 waiting for free OCR machine.
        ACTIVITY/7,OCRB6_RATE,,;OCRB6
;                                            processed at user assigned _RATE
        FREE,OCRB6/1,1;
;
        GOON,2;                              processed entities will always be
        ACTIVITY,0,,TS15;                           sent to the cart...if cart is full
        ACTIVITY,0,OCRB6CART.GE.OCRB6FULL,TS16;         a copy is sent to TS16 to
        ACTIVITY,0,,TTTT;                           dump the cart (open the gate)
;
TS15    ASSIGN,OCRB6CART=OCRB6CART+1,1;           track the number of entities in the
G6      AWAIT(21),B6G,,1;                         cart and wait til cart is full
        ASSIGN,CURRENT_ID=CURRENT_ID-38,1;
;
        GOON,1;                                     branching based on user specified
        ACT,0,ARRAY(1,CURRENT_ID),AS44;             probabilities
        ACT,0,ARRAY(2,CURRENT_ID),AS45;
        ACT,0,ARRAY(3,CURRENT_ID),AS46;
        ACT,0,ARRAY(4,CURRENT_ID),AS47;
        ACT,0,ARRAY(5,CURRENT_ID),AS48;
        ACT,0,ARRAY(6,CURRENT_ID),AS49;
        ACT,0,ARRAY(7,CURRENT_ID),AS50;
        ACT,0,ARRAY(8,CURRENT_ID),AS51;
;
AS44    ASSIGN,CURRENT_ID=51,1;
        ACT,0,,RURA;                   rural routes
;
AS45    ASSIGN,CURRENT_ID=67,1;
        ACT,DELAY,,B5A;                ADC secondary BCS
;
AS46    ASSIGN,CURRENT_ID=69,1;
        ACT,DELAY,,B5A;                5D firms secondary BCS
;
AS47    ASSIGN,CURRENT_ID=75,1;
        ACT,DELAY,,B13;                LSM primary rejects
;
AS48    ASSIGN,CURRENT_ID=156,1;
        ACT,DELAY,,B7;                 incoming VCD -- 3D
;
AS49    ASSIGN,CURRENT_ID=157,1;
        ACT,DELAY,,B7;                 incoming VCD -- 5D
;
AS50    ASSIGN,CURRENT_ID=159,1;
        ACT,DELAY,,B81;                firm groups sort B81
;
AS51    ASSIGN,CURRENT_ID=179,1;
        ACT,DELAY,,B82;                sequencing first pass B82
;
TS16    ASSIGN,OCRB6CART=0.,1;       cart is full, reset counter CART, open the
        OPEN,B6G,;                   gate for a short time to send mail on...
        ACT,0.1,,;
        CLOSE,B6G,;
        TERMINATE;
;
;
;####################### B7 is the incoming VCD ############################
;
;       inputs:  {154} managed VCD partial coded arrivals
;                {155} incoming partial coded OCR_B2 rejects
;                {156} OCR_B6 code rejects (3D)
;                {157} incomplete bar-code from OCR_B6 (5D)
;
```

```
B7      COLCT,ALL,TOA AT RVES_B7,24/60/60,1;
        AWAIT(9),VCDB7/1,,1;               waiting on VCD availability
        ACTIVITY/8,0,;VCDB7
;                                          processed at _RATE (min/entity)
        ASSIGN,CURRENT_ID=CURRENT_ID-128,1;  adjust _ID to match ARRAY columns
        ACT,VCDB4_FIRM,ARRAY(1,CURRENT_ID),AS60;
        ACT,VCDB4_FIRM,ARRAY(2,CURRENT_ID),AS61;
        ACT,VCDB4_FIRM,ARRAY(3,CURRENT_ID),AS62;
        ACT,VCDB4_SEQN,ARRAY(4,CURRENT_ID),AS63;
        ACT,VCDB4_FIRM,ARRAY(5,CURRENT_ID),AS64;
        ACT,VCDB4_SEQN,ARRAY(6,CURRENT_ID),AS65;
;
AS60    ASSIGN,CURRENT_ID=51,1;            rural routes
        ACT,0,,F7;
;
AS61    ASSIGN,CURRENT_ID=67,1;            ADC secondary BCS
        ACT,0,,F7;
;
AS62    ASSIGN,CURRENT_ID=69,1;            5D firms secondary
        ACT,0,,F7;
;
AS63    ASSIGN,CURRENT_ID=75,1;            LSM primary rejects
        ACT,0,,F7;
;
AS64    ASSIGN,CURRENT_ID=159,1;           firm groups sort
        ACT,0,,F7;
;
AS65    ASSIGN,CURRENT_ID=179,1;           sequencing first pass
        ACT,0,,F7;
;
F7      FREE,VCDB7/1,1;
;
        GOON,2;                            processed entities are placed in cart
        ACT,0,,TS17;                       and if the cart is full, a copy of
        ACT,0,VCDB7CART.GE.VCDB7FULL,TS18; the entity is sent to TS18 to dump
        ACT,0,,TTTT;                       the cart (open the gate).
;
TS17    ASSIGN,VCDB7CART=VCDB7CART+1,1;    track the number of entities in cart
G7      AWAIT(22),B7G,,1;                  and wait for a full cart
        ACT,DELAY,CURRENT_ID.EQ.51,RURA;
        ACT,DELAY,CURRENT_ID.EQ.67,B5A;
        ACT,DELAY,CURRENT_ID.EQ.69,B5A;
        ACT,DELAY,CURRENT_ID.EQ.75,B13;
        ACT,DELAY,CURRENT_ID.EQ.159,B81;
        ACT,DELAY,CURRENT_ID.EQ.179,B82;
;
TS18    ASSIGN,VCDB7CART=0.,1;              cart is full, reset counter and
        OPEN,B7G;                           open the gate for a short time to
        ACT,0.1,,;                          dump the cart...
        CLOSE,B7G;
        TERMINATE;
;
;
;######################## B81 is the firms sort ###############################
;
;   input:  {158} firm groups from BCS_B5
;          {159} firm groups from VCD_B7
;          {159} firm groups from OCR_B6
;
B81   COLCT,ALL,TOA FIRMS SORT;
      AWAIT(27),BCSB81/1,,1;
      ACTIVITY/14,BCSB81_RATE,;BCSB81
      ASSIGN,CURRENT_ID=CURRENT_ID-112,1;
      FREE,BCSB81/1,1;
      ACT,0,ARRAY(1,CURRENT_ID),TS27;
      ACT,0,ARRAY(2,CURRENT_ID),TS28;
      ACT,0,ARRAY(3,CURRENT_ID),TS29;
;
TS27  ASSIGN,CURRENT_ID=78,1;
      ACT,DELAY,,B14;                5D LSM inc B14
;
TS28  ASSIGN,CURRENT_ID=36,1;
      ACT,0,,PICK;                   firms pickup/boxes
;
TS29  ASSIGN,CURRENT_ID=40,1;
      ACT,0,,CRFM;                   carrier route firms
;
;
;################### B82 is the first sequencing pass ######################
;
; all entities arriving for the first sequencing pass come here
; inputs:  {178} c.g. mail from BCS_B5
;          {179} c.g. mail from OCR_B6
;          {179} c.g. mail from VCD_B7
;
B82   COLCT,ALL,TOA SEQ PASS 1,24/60/60,1;
      GOON,1;
      ACT,0,ARRAY(1,41),TS30;
      ACT,0,ARRAY(2,41),TS31;
;
TS30  ASSIGN,CURRENT_ID=79,1;
      ACT,DELAY,,B14;                5D LSM inc B14
;
TS31  ASSIGN,CURRENT_ID=258,1;
      ACT,DELAY,,B83;                second pass groups
;
;
;#################### B83 is the second sequencing pass #####################
;
; all entities arriving for the second sequencing pass come here
; input:  {258} from first pass B82
;
B83   COLCT,ALL,TOA SEQ PASS 2;
      GOON,1;
      ACT,0,ARRAY(1,42),TS32;
      ACT,0,ARRAY(2,42),TS33;
;
TS32  ASSIGN,CURRENT_ID=41,1;
      ACT,0,,SEQO;                   sequenced mail
;
TS33  ASSIGN,CURRENT_ID=79,1;
      ACT,DELAY,,B14;                rejects to LSM B14
```

```
;
;
;################  B9 is the primary manual sort block  #####################
;
;     inputs:  {2} collection manual arrivals
;              {3} FCE rejects from B1
;              {4} managed manual arrivals
;              {5} 3D manual arrivals
;              {6} 3D manual arrivals
;              {7} primary rejects from the VCD_B4
;
B9    AWAIT(10),MANB9/1,,1;                  waiting for personnel availability
      ACTIVITY/9,MANB9_RATE,,;MANB9
;                                            processed at _RATE (min/entity)
      FREE,MANB9/1,1;
;
      GOON,2;                                entity is always placed in cart and
      ACT,0,,TS19;                           if the cart is full, a copy entity is
      ACT,0,MANB9CART.GE.MANB9FULL,TS20;     sent to TS20 to dump the cart (open
      ACT,0,,TTTT;                           the gate)...
;
TS19  ASSIGN,MANB9CART=MANB9CART+1,1;        track the number of entities in cart
G9    AWAIT(23),B9G,,1;                      and wait for a full cart
      ACTIVITY,0,CURRENT_ID.EQ.2,GN2;
      ACTIVITY,0,CURRENT_ID.EQ.3,GN2;        then branch according to user-
      ACTIVITY,0,CURRENT_ID.EQ.4,GN3;        assigned probabilities
      ACTIVITY,0,CURRENT_ID.EQ.5,GN3;
      ACTIVITY,0,CURRENT_ID.EQ.6,GN3;
      ACTIVITY,0,CURRENT_ID.EQ.7,GN2;
;
GN2   GOON,1;
      ACT,0,ARRAY(1,30),AS70;
      ACT,0,ARRAY(2,30),AS71;
      ACT,0,ARRAY(3,30),AS72;
      ACT,0,ARRAY(4,30),AS73;
      ACT,0,ARRAY(5,30),AS74;
;
GN3   GOON,1;
      ACT,0,ARRAY(1,31),AS70;
      ACT,0,ARRAY(2,31),AS71;
      ACT,0,ARRAY(3,31),AS72;
      ACT,0,ARRAY(4,31),AS73;
      ACT,0,ARRAY(5,31),AS74;
;
AS70  ASSIGN,CURRENT_ID=27,1;
      ACT,0,,OUT;                               dispatch outgoing
;
AS71  ASSIGN,CURRENT_ID=28,1;
      ACT,0,,ADC;                               dispatch ADC
;
AS72  ASSIGN,CURRENT_ID=31,1;
      ACT,0,,FIRM:                              dispatch 5D firms
;
AS73  ASSIGN,CURRENT_ID=49,1;
      ACT,0,,RURA;                              rural routes
;
```

```
AS74  ASSIGN,CURRENT_ID=9,1;
      ACT,DELAY,,B10;                        secondary manual sort (inc)
;
TS20  ASSIGN,MANB9CART=0.,1;               cart is full, reset counter and
      OPEN,B9G;                            open gate for a short time to
      ACT,0.1,,;                           dump the cart
      CLOSE,B9G;
      TERMINATE;
;
;################  B10 is the secondary manual sort  ######################
;
;     inputs:  {9}  incoming manual secondary 5D from MAN_B9
;              {10} 5D manual presort #1 arrivals
;              {11} 5D manual presort #2 arrivals
;
B10   AWAIT(11),MANB10/1,,1;                waiting for personnel to
      ACTIVITY/10,MANB10_RATE,,;MANB10
;                                           process at _RATE (min/entity)
      FREE,MANB10/1,1;
;
      GOON,2;                               entity will go to cart (TS21) and
      ACT,0,,TS21;                          if the cart is full a copy of the
      ACT,0,MANB10CART.GE.MANB10FULL,TS22;  entity is sent to TS22 to empty the
      ACT,0,,TTTT;                          cart (open the gate)
;
TS21  ASSIGN,MANB10CART=MANB10CART+1,1;     track the number of entity in the
G10   AWAIT(24),B10G,,1;                    cart and wait til full...
      ACT,0,ARRAY(1,32),AS80
      ACT,0,ARRAY(2,32),AS81                branching by user's ARRAY values
      ACT,0,ARRAY(3,32),AS82
;
AS80  ASSIGN,CURRENT_ID=34,1;
      ACT,0,,PICK;                             firms pickup/boxes
;
AS81  ASSIGN,CURRENT_ID=37,1;
      ACT,0,,CRFM;                             carrier route - firms
;
AS82  ASSIGN,CURRENT_ID=44,1;
      ACT,0,,CRST;                             carrier route - sorted
;
TS22  ASSIGN,MANB10CART=0.,1;              cart is full...reset the counter,
      OPEN,B10G,;                          open the gate for a short time
      ACT,0.1,,;                           to empty the cart
      CLOSE,B10G,;
      TERMINATE;
;
;
;##################  B13 is the LSM block 13  ##############################
;
;     inputs:  {71} ADC rejects from the BCS_B5A
;              {72} 5D firms rejects from the BCS_B5A
;              {73} OCR rejects from the OCR_B2
;              {73} code rejects from VCD_B4
;              {74} code rejects from the BCS_B5
;              {75} code rejects from the OCR_B6
;              {75} code rejects from the VCD_B7
;               {75} mechanized LSM managed arrivals
;               {76} code rejects from the BCS_B3
;
B13   AWAIT(12),LSMB13/1,,1;                       waiting on machine
      ACTIVITY/12,0,,;LSMB13
      GOON,1;
      ACTIVITY,LSMB13_INC,CURRENT_ID.EQ.71,FR13;
      ACTIVITY,LSMB13_INC,CURRENT_ID.EQ.72,FR13;     processing at either OUT or
      ACTIVITY,LSMB13_OUT,CURRENT_ID.EQ.73,FR13;     INC rate dependent on _ID
      ACTIVITY,LSMB13_INC,CURRENT_ID.EQ.74,FR13;
      ACTIVITY,LSMB13_INC,CURRENT_ID.EQ.75,FR13;
      ACTIVITY,LSMB13_OUT,CURRENT_ID.EQ.76,FR13;
FR13  FREE,LSMB13/1,1;                             finished processing
      ASSIGN,CURREN  .D=CURRENT_ID-38,1;           adjust _ID for accessing ARRAY
;
      GOON,2;                                      entity is always placed in cart
      ACT,0,,TS23;                                 and if the cart is full, an
      ACT,0,LSMB13CART.GE.LSMB13FULL,TS24;         entity is sent to TS24 to
      ACT,0,,TTTT;                                 empty the cart (open the gate
;                                                  for a short time)
;
TS23  ASSIGN,LSMB13CART=LSMB13CART+1,1;            track the number of entities in
G13   AWAIT(25),B13G,,1;                           the cart and wait for the cart
      ACT,0,ARRAY(1,CURRENT_ID),AS84;              to be full...branch according
      ACT,0,ARRAY(2,CURRENT_ID),AS85;              to user-specified ARRAY
      ACT,0,ARRAY(3,CURRENT_ID),AS86;              values...
      ACT,0,ARRAY(4,CURRENT_ID),AS87;
      ACT,0,ARRAY(5,CURRENT_ID),AS88;
;
AS84  ASSIGN,CURRENT_ID=26,1;
      ACT,0,,OUT;                              dispatch outgoing
;
AS85  ASSIGN,CURRENT_ID=29,1;
      ACT,0,,ADC;                              dispatch to ADCs
;
AS86  ASSIGN,CURRENT_ID=32,1;
      ACT,0,,FIRM;                             dispatch 5D firms
;
AS87  ASSIGN,CURRENT_ID=50,1;
      ACT,0,,RURA;                             rural routes
;
AS88  ASSIGN,CURRENT_ID=77,1;
      ACT,DELAY,,B14;                          secondary LSM
;
TS24  ASSIGN,LSMB13CART=0.,1;                      cart is full, reset counter
      OPEN,B13G;                                   and open the gate briefly
      ACT,0.1,,;
      CLOSE,B13G;
      TERMINATE;
;
;
;################# B14 is the carrier-route sorting LSM ###################
;
;     inputs:  {77} incoming secondary LSM_B13
;              {78} code rejects from firms sort BCS_B81
;              {79} secondary sort rejects from B82
;             {79} secondary sort rejects from B82
;
B14   AWAIT(13),LSMB14/1,,1;                  waiting for LSM machine
      ACTIVITY/13,LSMB14_RATE,,;LSMB14
;                                             processed at user-specified _RATE
      FREE,LSMB14/1,1;
      ASSIGN,CURRENT_ID=CURRENT_ID-34,2;       the entity is always placed in cart
      ACT,0,,TS25;                             and if the cart is full an entity
      ACT,0,LSMB14CART.GE.LSMB14FULL,TS26;     is sent to TS26 to dump the cart
      ACT,0,,TTTT;                             (open the gate)
;
TS25  ASSIGN,LSMB14CART=LSMB14CART+1,1;        track the number of entities in cart
G14   AWAIT(26),B14G,,1;                       and wait until cart is full
      ACT,0,ARRAY(1,CURRENT_ID),AS90;
      ACT,0,ARRAY(2,CURRENT_ID),AS91;          branch based on ARRAY
      ACT,0,ARRAY(3,CURRENT_ID),AS92;
;
AS90  ASSIGN,CURRENT_ID=35,1;
      ACT,0,,PICK;                            firms pickup/boxes
;
AS91  ASSIGN,CURRENT_ID=38,1;
      ACT,0,,CRFM;                            carrier route firms
;
AS92  ASSIGN,CURRENT_ID=45,1;
      ACT,0,,CRST;                            carrier route sorted
;
TS26  ASSIGN,LSMB14CART=0.,1;                  the cart is full, reset counter
      OPEN,B14G,;                              and open the gate for a short time
      ACT,0.1,,;                               to dump the cart...
      CLOSE,B14G,;
      TERMINATE;
;
;
;
;******************** output streams by destination **********************
;
; all mail leaving SCF must pass through one of these COLCT nodes below
;
;
ADC   COLCT,ALL,TOA ADCS DIS;
      ACT,0,,DATA;
RURA  COLCT,ALL,TOA RURAL RT;
      ACT,0,,DATA;
FIRM  COLCT,ALL,TOA FIRMS 5D;
      ACT,0,,DATA;
PICK  COLCT,ALL,TOA PICKUP5D;
      ACT,0,,DATA;
CRFM  COLCT,ALL,TOA CR FIRMS;
      ACT,0,,DATA;
CRST  COLCT,ALL,TOA CR SORTD;
      ACT,0,,DATA;
SEQO  COLCT,ALL,TOA SEQUENCD;
      ACT,0,,DATA;
OUT   COLCT,ALL,TOA OUTGOING;
      ACT,0,,DATA;
;
```

```
;************************** TIME IN SYSTEM DATA *****************************
;
DATA  COLCT,INT(1),TIS:ALL MAIL;
      ACT,0,MAILSTREAM.EQ.1,M1;
      ACT,0,MAILSTREAM.EQ.15,M15;
      ACT,0,MAILSTREAM.EQ.16,M16;
      ACT,0,MAILSTREAM.EQ.12,M12;
      ACT,0,MAILSTREAM.EQ.13,M13;
      ACT,0,MAILSTREAM.EQ.53,M53;
      ACT,0,MAILSTREAM.EQ.54,M54;
      ACT,0,MAILSTREAM.EQ.55,M55;
      ACT,0,MAILSTREAM.EQ.56,M56;
      ACT,0,MAILSTREAM.EQ.57,M57;
      ACT,0,MAILSTREAM.EQ.58,M58;
      ACT,0,MAILSTREAM.EQ.63,M63;
      ACT,0,MAILSTREAM.EQ.154,M154;
      ACT,0,MAILSTREAM.EQ.2,M2;
      ACT,0,MAILSTREAM.EQ.4,M4;
      ACT,0 MAILSTREAM.EQ.5,M5;
      ACT,0,MAILSTREAM.EQ.6,M6;
      ACT,0,MAILSTREAM.EQ.10,M10;
      ACT,0,MAILSTREAM.EQ.11,M11;
      ACT,0,MAILSTREAM.EQ.75,M75;
;
M1    COLCT,INT(1),TIS:STREAM 1;
      ACT,0,,TTTT;
M15   COLCT,INT(1),TIS:STREAM 15;
      ACT,0,,TTTT;
M16   COLCT,INT(1),TIS:STREAM 16;
      ACT,0,,TTTT;
M12   COLCT,INT(1),TIS:STREAM 12;
      ACT,0,,TTTT;
M13   COLCT,INT(1),TIS:STREAM 13;
      ACT,0,,TTTT;
M53   COLCT,INT(1),TIS:STREAM 53;
      ACT,0,,TTTT;
M54   COLCT,INT(1),TIS:STREAM 54;
      ACT,0,,TTTT;
M55   COLCT,INT(1),TIS:STREAM 55;
      ACT,0,,TTTT;
M56   COLCT,INT(1),TIS:STREAM 56;
      ACT,0,,TTTT;
M57   COLCT,INT(1),TIS:STREAM 57;
      ACT,0,,TTTT;
M58   COLCT,INT(1),TIS:STREAM 58;
      ACT,0,,TTTT;
M63   COLCT,INT(1),TIS:STREAM 63;
      ACT,0,,TTTT;
M154  COLCT,INT(1),TIS:STREAM 154;
      ACT,0,,TTTT;
M2    COLCT,INT(1),TIS:STREAM 2;
      ACT,0,,TTTT;
M4    COLCT,INT(1),TIS:STREAM 4;
      ACT,0,,TTTT;
M5    COLCT,INT(1),TIS:STREAM 5;
      ACT,0,,TTTT;
```

```
M6    COLCT,INT(1),TIS:STREAM 6;
      ACT,0,,TTTT;
M10   COLCT,INT(1),TIS:STREAM 10;
      ACT,0,,TTTT;
M11   COLCT,INT(1),TIS:STREAM 11;
      ACT,0,,TTTT;
M75   COLCT,INT(1),TIS:STREAM 75;
      ACT,0,,TTTT;
;
      ENDNETWORK;
INITIALIZE,0,1500;                    1440 min = 24 hrs
FIN;
```

ARRIV.DAT contains the total mailstream volumes and arrival profile percentages for the mail profiles C, B, or M. Total mail volumes will be changed for your if you alter the entity aggregation. The stream volumes must be integer (the machine processing rates will also be automatically changed if the entity aggregation is reset...no changes are needed in the network model POSTAL.DAT).
The main FORTRAN program MAIL.FOR reads the total stream volumes into a vector ITVOL(20). Profile data are proportion of total stream volume per hour (cumulative) and are read into XPRO(36,3) by MAIL.FOR.

```
switch set for entity aggregate: (1) kpiece  (2) tray...(and so on)
1
switch set for total input vol: (1) deterministic or  (2) randomize
1
Total stream volumes (format I4): i.d. / type /  profile
 800      1     collection stamped         { 1}  local col.          C
 692      1     collection metered #1      {15}  metered by-pass     C
  52      1     collection metered #2      {16}  metered by-pass     C
 287      3     presort 5-D bundles #1     {12}  mailer presort      M
 163      2     presort 5-D bundles #2     {13}  mailer presort      B
 489      3     managed incomplete #1      {53}  MMP                 M
  29      3     managed incomplete #2      {54}  MMP                 M
  19      3     incoming 3-D mech. #1      {55}  MMP                 M
 319      2     incoming 3-D mech. #2      {56}  MMP                 B
   2      3     incoming 3-D mech. #3      {57}  MMP                 M
  28      2     incoming 3-D mech. #4      {58}  MMP                 B
 101      3     managed barcoded 11-D      {63}  MMP                 M
 475      3     managed VCD               {154}  MMP                 M
  56      1     collection manual          { 2}  local col           C
  83      3     managed manual             { 4}  MMP                 M
   1      3     3-D manual #1              { 5}  large mailer        M
  20      2     3-D manual #2              { 6}  large mailer        B
  15      3     5-D manual presort #1      {10}  mailer presort      M
   9      2     5-D manual presort #2      {11}  mailer presort      B
  67      3     LSM managed mech.          {75}  MMP                 M


ARRIVAL PROFILES (CUMULATIVE PERCENTAGE OF TOTAL STREAM VOLUME PER HOUR):
   C        B        M            TIME        PERIOD     MIN
0.0105   0.1090   0.3765      0700 - 0730       1       0 -  30
0.0210   0.2180   0.7530      0730 - 0800       2      30 -  60
0.0370   0.2525   0.7935      0800 - 0830       3      60 -  90
0.0530   0.2870   0.8340      0830 - 0900       4      90 - 120
0.0605   0.3165   0.8510      0900 - 0930       5     120 - 150
0.0680   0.3460   0.8680      0930 - 1000       6     150 - 180
0.0780   0.3855   0.8810      1000 - 1030       7     180 - 210
0.0880   0.4250   0.8940      1030 - 1100       8     210 - 240
0.1140   0.4595   0.9050      1100 - 1130       9     240 - 270
0.1400   0.4940   0.9160      1130 - 1200      10     270 - 300
0.1620   0.5090   0.9310      1200 - 1230      11     300 - 330
0.1840   0.5240   0.9460      1230 - 1300      12     330 - 360
0.2055   0.5785   0.9550      1300 - 1330      13     360 - 390
0.2270   0.6330   0.9640      1330 - 1400      14     390 - 420
0.3300   0.6430   0.9710      1400 - 1430      15     420 - 450
0.4330   0.6530   0.9780      1430 - 1500      16     450 - 480
0.4675   0.6825   0.9825      1500 - 1530      17     480 - 510
```

0.5020	0.7120	0.9870	1530 - 1600	18	510 - 540
0.6525	0.7170	0.9875	1600 - 1630	19	540 - 570
0.8030	0.7220	0.9880	1630 - 1700	20	570 - 600
0.8855	0.7420	0.9880	1700 - 1730	21	600 - 630
0.9680	0.7620	0.9880	1730 - 1800	22	630 - 660
0.9835	0.7620	0.9900	1800 - 1830	23	660 - 690
0.9990	0.7620	0.9920	1830 - 1900	24	690 - 720
0.9995	0.7770	0.9920	1900 - 1930	25	720 - 750
1.0000	0.7920	0.9920	1930 - 2000	26	750 - 780
1.0000	0.8165	0.9920	2000 - 2030	27	780 - 810
1.0000	0.8410	0.9920	2030 - 2100	28	810 - 840
1.0000	0.9205	0.9960	2100 - 2130	29	840 - 870
1.0000	1.0000	0.9960	2130 - 2200	30	870 - 900
1.0000	1.0000	1.0000	2200 - 2230	31	900 - 930
1.0000	1.0000	1.0000	2230 - 2300	32	930 - 960
1.0000	1.0000	1.0000	2300 - 2330	33	960 - 990
1.0000	1.0000	1.0000	2330 - 2400	34	990 - 1020
1.0000	1.0000	1.0000	2400 - 0030	35	1020 - 1050
1.0000	1.0000	1.0000	0030 - 0100	36	1050 - 1080
1	2	3			

```
XX.DAT is a data file for initializing the SLAM II global variables XX(I),
called by MAIL.FOR, used with network model POSTAL.DAT (format F8.4)
machine processing rates in min/kp (per unit resource)
{regardless of entity aggregation, leave rates in min/kp}
  2.8572              XX(1)    FCEB1_RATE    21,000
  2.1818              XX(2)    OCRB2_RATE    27,500
  2.1428              XX(3)    BCSB3_RATE    28,000
  2.8484              XX(4)    VCDB4_SLOW    21,065
  2.4228              XX(5)    VCDB4_FAST    24,765
  2.1428              XX(6)    BCSB5_RATE    28,000
  2.1428              XX(7)    BCSB5A_RATE   28,000
  4.8000              XX(8)    MANB6A_RATE   12,500
  2.1818              XX(9)    OCRB6_RATE    27,500
  2.8484              XX(10)   VCDB7_RATE    21,065
 67.039               XX(11)   MANB9_RATE       895
 67.114               XX(12)   MANB10_RATE      894
  2.1368              XX(13)   LSMB13_OUT    28,079
  1.8348              XX(14)   LSMB13_INC    32,700
  2.1582              XX(15)   LSMB14_RATE   27,800
  1.0933              XX(63)   VCDB4_PROB    54,880
  2.0292              XX(64)   VCDB4_OUTG    29,568
  4.4643              XX(65)   VCDB4_SEQN    13,440
  0.0000              XX(66)   VCDB4_RURA      inf
  1.7857              XX(67)   VCDB4_FIRM    33,600
  2.1428              XX(68)   BCSB81_RATE   28,000

specified number of entities when carts are full minus one...
 14.0000              XX(40)   FCEB1FULL
 14.0000              XX(41)   OCRB2FULL
 14.0000              XX(42)   BCSB3FULL
 14.0000              XX(43)   VCDB4FULL
 14.0000              XX(44)   BCSB5FULL
 14.0000              XX(45)   BCSB5AFULL
 14.0000              XX(46)   MANB6AFULL
 14.0000              XX(47)   OCRB6FULL
 14.0000              XX(48)   VCDB7FULL
 14.0000              XX(49)   MANB9FULL
 14.0000              XX(50)   MANB10FULL
 14.0000              XX(51)   LSMB13FULL
 14.0000              XX(52)   LSMB14FULL
```

```
ARRAY.DAT includes all of the ARRAY values for POSTAL.DAT SLAM II network,
called by MAIL.FOR by the EVENT subroutine (ARRAY= branching probabilities)
                                                         (format F6.4)
column 1: branching from the FCE block 1
0.5400              ARRAY(1,1)          to B2 OCR           {14}
0.3900                   (1,2)          to B4 VCD           {152}
0.0700                   (1,3)          to B9 manual        {3}
0.0000
0.0000
0.0000
0.0000
0.0000
0.0000

column 2: branching from the OCR block 2 if CURRENT_ID=14 (COL AUTOMATED)
0.1750              ARRAY(2,1)          to B3 BCS           {19}
0.3650                   (2,2)          to dispatch         {21}
0.1200                   (2,3)          to B5 BCS           {65}
0.0270                   (2,4)          to B5 BCS           {66}
0.0070                   (2,5)          to B5A BCS          {67}
0.0050                   (2,6)          to B5A BCS          {69}
0.0380                   (2,7)          to B13 LSM          {73}
0.2330                   (2,8)          to B4 VCD           {153}
0.0300                   (2,9)          to B7 VCD           {155}

column 3: branching from the OCR block 2 if CURRENT_ID=15 (COL METERED)
0.1540              ARRAY(3,1)          to B3 BCS           {19}
0.3210                   (3,2)          to dispatch         {21}
0.1050                   (3,3)          to B5 BCS           {65}
0.0240                   (3,4)          to B5 BCS           {66}
0.0060                   (3,5)          to B5A BCS          {67}
0.0050                   (3,6)          to B5A BCS          {69}
0.0340                   (3,7)          to B13 LSM          {73}
0.3250                   (3,8)          to B4 VCD           {153}
0.0260                   (3,9)          to B7 VCD           {155}

column 4: branching from the OCR block 2 if CURRENT_ID=16 (COL METERED)
0.1540              ARRAY(4,1)          to B3 BCS           {19}
0.3210                   (4,2)          to dispatch         {21}
0.1180                   (4,3)          to B5 BCS           {65}
0.0240                   (4,4)          to B5 BCS           {66}
0.0060                   (4,5)          to B5A BCS          {67}
0.0050                   (4,6)          to B5A BCS          {69}
0.0340                   (4,7)          to B13 LSM          {73}
0.3250                   (4,8)          to B4 VCD           {153}
0.0130                   (4,9)          to B7 VCD           {155}

column 5: branching from the BCS block 3

0.9800              ARRAY(5,1)          to dispatch         {22}
0.0200                   (5,2)          to B13 LSM          {76}
0.0000                   (5,3)
0.0000                   (5,4)
0.0000                   (5,5)
0.0000                   (5,6)
0.0000                   (5,7)
0.0000                   (5,8)
```

```
0.0000                    (5,9)

column 6: branching from the VCD block 4

0.0190              ARRAY(6,1)            to B9 manual       {7}
0.2230                    (6,2)           to B3 BCS          {20}
0.4660                    (6,3)           to dispatch        {25}
0.1910                    (6,4)           to B5 BCS          {65}
0.0350                    (6,5)           to B5 BCS          {66}
0.0090                    (6,6)           to B5A BCS         {67}
0.0070                    (6,7)           to B5A BCS         {69}
0.0500                    (6,8)           to B13 LSM         {73}
0.0000                    (6,9)

column 7: branching from the BCS block 5 if CURRENT_ID=65

0.0000              ARRAY(7,1)
0.0350                    (7,2)           to B5A BCS         {70}
0.0200                    (7,3)           to B13 LSM         {74}
0.1040                    (7,4)           to B81 seq         {158}
0.8410                    (7,5)           to B82 seq         {174}
0.0000                    (7,6)
0.0000                    (7,7)
0.0000                    (7,8)
0.0000                    (7,9)

column 8: branching from the BCS block 5 if CURRENT_ID=66
0.9800              ARRAY(8,1)            to dispatch        {51}
0.0000                    (8,2)
0.0200                    (8,3)           to B13 LSM         {74}
0.0000                    (8,4)
0.0000                    (8,5)
0.0000                    (8,6)
0.0000                    (8,7)
0.0000                    (8,8)
0.0000                    (8,9)

column 9: branching from the BCS block 5A if CURRENT_ID=67
0.9800              ARRAY(9,1)            to dispatch        {30}
0.0200                    (9,2)           to B13 LSM         {71}
0.0000                    (9,3)
0.0000                    (9,4)
0.0000                    (9,5)
0.0000                    (9,6)
0.0000                    (9,7)
0.0000                    (9,8)
0.0000                    (9,9)

column 10: branching from the BCS block 5A if CURRENT_ID=68
0.9900              ARRAY(10,1)           to dispatch        {30}
0.0100                    (10,2)          to B13 LSM         {71}
0.0000                    (10,3)
0.0000                    (10,4)
0.0000                    (10,5)
0.0000                    (10,6)
0.0000                    (10,7)
0.0000                    (10,8)
0.0000                    (10,9)
```

```
column 11: branching from the BCS block 5A if CURRENT_ID=69
0.9800          ARRAY(11,1)          to dispatch       {30}
0.0200               (11,2)          to B13 LSM        {71}
0.0000               (11,3)
0.0000               (11,4)
0.0000               (11,5)
0.0000               (11,6)
0.0000               (11,7)
0.0000               (11,8)
0.0000               (11,9)

column 12: branching from the BCS block 5A if CURRENT_ID=70
0.9900          ARRAY(12,1)          to dispatch       {30}
0.0100               (12,2)          to B13 LSM        {71}
0.0000               (12,3)
0.0000               (12,4)
0.0000               (12,5)
0.0000               (12,6)
0.0000               (12,7)
0.0000               (12,8)
0.0000               (12,9)

column 13: branching from MAN BUNDLE SORT block 6A if CURRENT_ID=12
0.0150          ARRAY(13,1)          to ADC dispatch   {28}
0.0300               (13,2)          5D firm dispatch  {31}
0.1480               (13,3)          rural routes      {49}
0.7430               (13,4)          to B6 OCR         {59}
0.0000               (13,5)          to B6 OCR         {60}
0.0640               (13,6)          to B6 OCR         {61}
0.0000               (13,7)          to B6 OCR         {62}
0.0000               (13,8)
0.0000               (13,9)

column 14: branching from MAN BUNDLE SORT block 6A if CURRENT_ID=13
0.0150          ARRAY(14,1)          to ADC dispatch   {28}
0.0300               (14,2)          5D firm dispatch  {31}
0.1480               (14,3)          rural routes      {49}
0.0000               (14,4)          to B6 OCR         {59}
0.7430               (14,5)          to B6 OCR         {60}
0.0000               (14,6)          to B6 OCR         {61}
0.0640               (14,7)          to B6 OCR         {62}
0.0000               (14,8)
0.0000               (14,9)

column 15: branching from the INC OCR B if CURRENT_ID=53
0.1450          ARRAY(15,1)          to dispatch       {51}
0.0150               (15,2)          to B5A BCS        {67}
0.0290               (15,3)          to B5A BCS        {69}
0.0200               (15,4)          to B13 LSM        {75}
0.0000               (15,5)          to B7 VCD         {156}
0.1580               (15,6)          to B7 VCD         {157}
0.0690               (15,7)          to B81 seq        {159}
0.5640               (15,8)          to B82 seq        {179}
0.0000               (15,9)
```

column 16: branching from the INC OCR B6 if CURRENT_ID=54

0.1450	ARRAY(16,1)	to dispatch	{51}
0.0150	(16,2)	to B5A BCS	{67}
0.0290	(16,3)	to B5A BCS	{69}
0.0200	(16,4)	to B13 LSM	{75}
0.0000	(16,5)	to B7 VCD	{156}
0.0790	(16,6)	to B7 VCD	{157}
0.0780	(16,7)	to B81 seq	{159}
0.6340	(16,8)	to B82 seq	{179}
0.0000	(16,9)		

column 17: branching from the INC OCR B6 if CURRENT_ID=55

0.1110	ARRAY(17,1)	to dispatch	{51}
0.0110	(17,2)	to B5A BCS	{67}
0.0220	(17,3)	to B5A BCS	{69}
0.0390	(17,4)	to B13 LSM	{75}
0.2110	(17,5)	to B7 VCD	{156}
0.1210	(17,6)	to B7 VCD	{157}
0.0530	(17,7)	to B81 seq	{159}
0.4320	(17,8)	to B82 seq	{179}
0.0000	(17,9)		

column 18: branching from the INC OCR B6 if CURRENT_ID=56

0.1060	ARRAY(18,1)	to dispatch	{51}
0.0110	(18,2)	to B5A BCS	{67}
0.0210	(18,3)	to B5A BCS	{69}
0.0380	(18,4)	to B13 LSM	{75}
0.2420	(18,5)	to B7 VCD	{156}
0.1160	(18,6)	to B7 VCD	{157}
0.0510	(18,7)	to B81 seq	{159}
0.4150	(18,8)	to B82 seq	{179}
0.0000	(18,9)		

column 19: branching from the INC OCR B6 if CURRENT_ID=57

0.1110	ARRAY(19,1)	to dispatch	{51}
0.0110	(19,2)	to B5A BCS	{67}
0.0220	(19,3)	to B5A BCS	{69}
0.0390	(19,4)	to B13 LSM	{75}
0.2110	(19,5)	to B7 VCD	{156}
0.0610	(19,6)	to B7 VCD	{157}
0.0600	(19,7)	to B81 seq	{159}
0.4850	(19,8)	to B82 seq	{179}
0.0000	(19,9)		

column 20: branching from the INC OCR B6 if CURRENT_ID=58

0.1060	ARRAY(20,1)	to dispatch	{51}
0.0110	(20,2)	to B5A BCS	{67}
0.0210	(20,3)	to B5A BCS	{69}
0.0380	(20,4)	to B13 LSM	{75}
0.2420	(20,5)	to B7 VCD	{156}
0.0580	(20,6)	to B7 VCD	{157}
0.0570	(20,7)	to B81 seq	{159}
0.4670	(20,8)	to B82 seq	{179}
0.0000	(20,9)		

column 21: branching from the INC OCR B6 if CURRENT_ID=59

```
0.0000          ARRAY(21,1)          to dispatch      {51}
0.0000               (21,2)          to B5A BCS       {67}
0.0000               (21,3)          to B5A BCS       {69}
0.0390               (21,4)          to B13 LSM       {75}
0.2110               (21,5)          to B7 VCD        {156}
0.1500               (21,6)          to B7 VCD        {157}
0.0660               (21,7)          to B81 seq       {159}
0.5340               (21,8)          to B82 seq       {179}
0.0000               (21,9)

column 22: branching from the INC OCR B6 if CURRENT_ID=60
0.0000          ARRAY(22,1)          to dispatch      {51}
0.0000               (22,2)          to B5A BCS       {67}
0.0000               (22,3)          to B5A BCS       {69}
0.0380               (22,4)          to B13 LSM       {75}
0.2420               (22,5)          to B7 VCD        {156}
0.1440               (22,6)          to B7 VCD        {157}
0.0630               (22,7)          to B81 seq       {159}
0.5130               (22,8)          to B82 seq       {179}
0.0000               (22,9)

column 23: branching from the INC OCR B6 if CURRENT_ID=61
0.0000          ARRAY(23,1)          to dispatch      {51}
0.0000               (23,2)          to B5A BCS       {67}
0.0000               (23,3)          to B5A BCS       {69}
0.0390               (23,4)          to B13 LSM       {75}
0.2110               (23,5)          to B7 VCD        {156}
0.0750               (23,6)          to B7 VCD        {157}
0.0740               (23,7)          to B81 seq       {159}
0.6010               (23,8)          to B82 seq       {179}
0.0000               (23,9)

column 24: branching from the INC OCR B6 if CURRENT_ID=62
0.0000          ARRAY(24,1)          to dispatch      {51}
0.0000               (24,2)          to B5A BCS       {67}
0.0000               (24,3)          to B5A BCS       {69}
0.0380               (24,4)          to B13 LSM       {75}
0.2420               (24,5)          to B7 VCD        {156}
0.0720               (24,6)          to B7 VCD        {157}
0.0710               (24,7)          to B81 seq       {159}
0.5770               (24,8)          to B82 seq       {179}
0.0000               (24,9)

column 25: branching from the INC OCR B6 if CURRENT_ID=63
0.1450          ARRAY(25,1)          to dispatch      {51}
0.0150               (25,2)          to B5A BCS       {67}
0.0290               (25,3)          to B5A BCS       {69}
0.0200               (25,4)          to B13 LSM       {75}
0.0000               (25,5)          to B7 VCD        {156}
0.0000               (25,6)          to B7 VCD        {157}
0.0870               (25,7)          to B81 seq       {159}
0.7040               (25,8)          to B82 seq       {179}
0.0000               (25,9)

column 26: branching from the INC VCD B7 if CURRENT_ID=154
0.1450          ARRAY(26,1)          to dispatch      {51}
```

```
0.0150              (26,2)          to B5A BCS        {67}
0.0280              (26,3)          to B5A BCS        {69}
0.0200              (26,4)          to B13 LSM        {75}
0.0870              (26,5)          to B81 seq        {159}
0.7050              (26,6)          to B82 seq        {179}
0.0000              (26,7)
0.0000              (26,8)
0.0000              (26,9)

column 27: branching from the INC VCD B7 if CURRENT_ID=155
0.0000         ARRAY(27,1)          to dispatch       {51}
0.0000              (27,2)          to B5A BCS        {67}
0.0000              (27,3)          to B5A BCS        {69}
0.0240              (27,4)          to B13 LSM        {75}
0.1070              (27,5)          to B81 seq        {159}
0.8690              (27,6)          to B82 seq        {179}
0.0000              (27,7)
0.0000              (27,8)
0.0000              (27,9)

column 28: branching from the INC VCD B7 if CURRENT_ID=156
0.1400         ARRAY(28,1)          to dispatch       {51}
0.0140              (28,2)          to B5A BCS        {67}
0.0280              (28,3)          to B5A BCS        {69}
0.0500              (28,4)          to B13 LSM        {75}
0.0840              (28,5)          to B81 seq        {159}
0.6840              (28,6)          to B82 seq        {179}
0.0000              (28,7)
0.0000              (28,8)
0.0000              (28,9)

column 29: branching from the INC VCD B7 if CURRENT_ID=157
0.0000         ARRAY(29,1)          to dispatch       {51}
0.0000              (29,2)          to B5A BCS        {67}
0.0000              (29,3)          to B5A BCS        {69}
0.0000              (29,4)          to B13 LSM        {75}
0.1100              (29,5)          to B81 seq        {159}
0.8900              (29,6)          to B82 seq        {179}
0.0000              (29,7)
0.0000              (29,8)
0.0000              (29,9)

column 30: branching from the MANUAL PRIMARY B9 if CURRENT_ID=2,3,or 7
0.7400         ARRAY(30,1)          to dispatch       {27}
0.0100              (30,2)          to dispatch       {28}
0.0070              (30,3)          to dispatch       {31}
0.0380              (30,4)          to dispatch       {49}
0.2050              (30,5)          to B10 manual 2nd {9}
0.0000              (30,6)
0.0000              (30,7)
0.0000              (30,8)
0.0000              (30,9)

column 31: branching from the MANUAL PRIMARY B9 if CURRENT_ID=4,5,6,or 8
0.0000         ARRAY(31,1)          to dispatch       {27}
0.0150              (31,2)          to dispatch       {28}
```

```
0.0300              (31,3)          to dispatch        {31}
0.1480              (31,4)          to dispatch        {49}
0.8070              (31,5)          to B10 manual 2nd  {9}
0.0000              (31,6)
0.0000              (31,7)
0.0000              (31,8)
0.0000              (31,9)

column 32: branching from the MANUAL SECONDARY B10
0.1040         ARRAY(32,1)          to dispatch        {34}
0.0060              (32,2)          to CIO             {37}
0.8900              (32,3)          to CIO             {44}
0.0000              (32,4)
0.0000              (32,5)
0.0000              (32,6)
0.0000              (32,7)
0.0000              (32,8)
0.0000              (32,9)

column 33: branching from the LSM B13 if CURRENT_ID=71
0.0000         ARRAY(33,1)          to dispatch        {26}
1.0000              (33,2)          to dispatch        {29}
0.0000              (33,3)          to dispatch        {32}
0.0000              (33,4)          to dispatch        {50}
0.0000              (33,5)          to B14 LSM         {77}
0.0000              (33,6)
0.0000              (33,7)
0.0000              (33,8)
0.0000              (33,9)

column 34: branching from the LSM B13 if CURRENT_ID=72
0.0000         ARRAY(34,1)          to dispatch        {26}
0.0000              (34,2)          to dispatch        {29}
1.0000              (34,3)          to dispatch        {32}
0.0000              (34,4)          to dispatch        {50}
0.0000              (34,5)          to B14 LSM         {77}
0.0000              (34,6)
0.0000              (34,7)
0.0000              (34,8)
0.0000              (34,9)

column 35: branching from the LSM B13 if CURRENT_ID=73
0.7400         ARRAY(35,1)          to dispatch        {26}
0.0100              (35,2)          to dispatch        {29}
0.0080              (35,3)          to dispatch        {32}
0.0380              (35,4)          to dispatch        {50}
0.2040              (35,5)          to B14 LSM         {77}
0.0000              (35,6)
0.0000              (35,7)
0.0000              (35,8)
0.0000              (35,9)

column 36: branching from the LSM B13 if CURRENT_ID=74
0.0000         ARRAY(36,1)          to dispatch        {26}
0.0000              (36,2)          to dispatch        {29}
0.0000              (36,3)          to dispatch        {32}
```

```
0.1550              (36,4)          to dispatch     {50}
0.8450              (36,5)          to B14 LSM      {77}
0.0000              (36,6)
0.0000              (36,7)
0.0000              (36,8)
0.0000              (36,9)

column 37: branching from the LSM B13 if CURRENT_ID=75
0.0000         ARRAY(37,1)          to dispatch     {26}
0.0150              (37,2)          to dispatch     {29}
0.0300              (37,3)          to dispatch     {32}
0.1480              (37,4)          to dispatch     {50}
0.8070              (37,5)          to B14 LSM      {77}
0.0000              (37,6)
0.0000              (37,7)
0.0000              (37,8)
0.0000              (37,9)

column 38: branching from the LSM B13 if CURRENT_ID=76
1.0000         ARRAY(38,1)          to dispatch     {26}
0.0000              (38,2)          to dispatch     {29}
0.0000              (38,3)          to dispatch     {32}
0.0000              (38,4)          to dispatch     {50}
0.0000              (38,5)          to B14 LSM      {77}
0.0000              (38,6)
0.0000              (38,7)
0.0000              (38,8)
0.0000              (38,9)

column 39: branching from the LSM B14
0.3500         ARRAY(39,1)          to dispatch     {35}
0.0210              (39,2)          to CIO          {38}
0.6290              (39,3)          to CIO          {45}
0.0000              (39,4)
0.0000              (39,5)
0.0000              (39,6)
0.0000              (39,7)
0.0000              (39,8)
0.0000              (39,9)

column 40: branching from the B8.1 FIRMS SORT
0.0150         ARRAY(40,1)          to B14 LSM      {78}
0.9305              (40,2)          firms pickup    {36}
0.0545              (40,3)          c.r.firms       {40}
0.0000              (40,4)
0.0000              (40,5)
0.0000              (40,6)
0.0000              (40,7)
0.0000              (40,8)
0.0000              (40,9)

column 41: branching from the B8.2 FIRST SEQUENCING PASS
0.0150         ARRAY(41,1)          to B14 LSM      {78}
0.9850              (41,2)          second pass B83 {38}
0.0000              (41,3)
0.0000              (41,4)
```

```
0.0000              (41,5)
0.0000              (41,6)
0.0000              (41,7)
0.0000              (41,8)
0.0000              (41,9)

column 42: branching from the B8.3 SECOND SEQUENCING PASS
0.9900         ARRAY(42,1)          sequenced mail    {41}
0.0100              (42,2)          rejects to LSM B14 {79}
0.0000              (42,3)
0.0000              (42,4)
0.0000              (42,5)
0.0000              (42,6)
0.0000              (42,7)
0.0000              (42,8)
0.0000              (42,9)

column 43: branching from the B14 LSM if CURRENT_ID=77
0.1040         ARRAY(43,1)          firms pickup      {35}
0.0060              (43,2)          c.r. firms        {38}
0.8900              (43,3)          c.r. sorted       {45}
0.0000              (43,4)
0.0000              (43,5)
0.0000              (43,6)
0.0000              (43,7)
0.0000              (43,8)
0.0000              (43,9)

column 44: branching from the B14 LSM if CURRENT_ID=78
0.9440         ARRAY(44,1)          firms pickup      {35}
0.0560              (44,2)          c.r. firms        {38}
0.0000              (44,3)
0.0000              (44,4)
0.0000              (44,5)
0.0000              (44,6)
0.0000              (44,7)
0.0000              (44,8)
0.0000              (44,9)

column 45: branching from the B14 LSM if CURRENT_ID=79
0.0000         ARRAY(45,1)          firms pickup      {35}
0.0000              (45,2)          c.r. firms        {38}
1.0000              (45,3)          c.r. sorted       {45}
0.0000              (45,4)
0.0000              (45,5)
0.0000              (45,6)
0.0000              (45,7)
0.0000              (45,8)
0.0000              (45,9)

column 46: branching from the BCS_B81 firms sort if CURRENT_ID=158
0.0100              (46,1)          5D LSM to B14     {78}
0.9350              (46,2)          firms/PO boxes    {36}
0.0550              (46,3)          c.r. firms        {40}
0.0000              (46,4)
0.0000              (46,5)
```

```
0.0000                  (46,6)
0.0000                  (46,7)
0.0000                  (46,8)
0.0000                  (46,9)

column 47: branching from the BCS_B81 firms sort if CURRENT_ID=159
0.0200                  (47,1)              5D LSM to B14       {78}
0.9260                  (47,2)              firms/PO boxes      {36}
0.0540                  (47,3)              c.r. firms          {40}
0.0000                  (47,4)
0.0000                  (47,5)
0.0000                  (47,6)
0.0000                  (47,7)
0.0000                  (47,8)
0.0000                  (47,9)
```

Appendix B. MILP Model Generation Computer Codes

```
#define D_MAX 21        /*  number of arrival streams plus one        */
#define DTIMES 48       /*  total number of time periods              */
#define INMAX 12        /*  max input or output streams for any node  */
#define NODES 16        /*  number of processing nodes plus two       */
#define TWINDOW 34      /*  last period before sequencing window (Td) */
#define MACHTOT 7       /*  number of different machine types+1       */
#define SPACE 178.5     /*  maximum free floor space (1000 sf)        */
#define CPSF 11.5       /*  cost ($10K/1000 sf) additional floorspace */
#define MAX(x,y)  (((x) > (y)) ? (x) : (y))
#include <stdio.h>


    /***********************************************************/
    /*  MGP1.C is a problem-specific model generator for the   */
    /*  GMF-A equipment selection and scheduling optimization  */
    /*  problem.  MGP1 requires input data from GMFA.DAT and   */
    /*  writes the XML-language model to a family of output    */
    /*  files (XML#.DAT).  Output may be sizable (possibly     */
    /*  1 Mb or more for certain input parameters).            */
    /*                              UT OR Dept, Wert, 1 Nov 89 */
    /***********************************************************/


int  dstrm[D_MAX], prof[D_MAX], mach[NODES], tn1[NODES], tnr[NODES];
int  inum[NODES], onum[NODES], linum[NODES], istrm[INMAX][NODES];
int  lnode[INMAX][NODES], ostrm[INMAX][NODES], pnode[INMAX][NODES];
int  anode[D_MAX], cntr15;
int  cntr2, cntr3, cntr4, cntr5, cntr7, cntr12, cntr13, cntr14;
int  p[MACHTOT] = {0, 2, 2, 2, 18, 17, 1 };

/*  Description of integer variable arrays:

      dstrm[]     stream id numbers for arriving mail
      prof[]      input stream profile ids
      mach[]      type of machin available at station n
      tn1[]       first possible operating period at n
      tnr[]       last possible operating period at n
      inum[]      number of input streams at n
      onum[]      number of output streams at n
      linum[]     contains the max(inum,onum) for each n
      istrm[][]   input stream numbers at each station
      lnode[][]   preceding nodes corresponding to istrm[][]
      ostrm[][]   output stream numbers at each station
      pnode[][]   following nodes corresponding to ostrm[][]
      anode[]     station number to which input streams are fed
      cntr#       counters for number of equations generated
      p[]         number of operators required for each machine      */

float  fsp[MACHTOT] = { 0, 1.995, 3.349, 2.993, 3.848, 3.42, 0};
float  cacq[7] = { 0, 40., 65., 17.5, 70., 40., 0.001};
float  cmnt[7] = { 0., 4., 6.5, 1.75, 7., 4., 0 };
float  dvol[D_MAX], arrivd[4][DTIMES];
float pay[7] = {0.0, 21.11, 21.11, 21.11, 21.11, 21.11, 21.79 };
long  pos, numeqn, numvar;      /*  accounting variables  */

/*  Description of real variable arrays:
      fsp[]      floorspace required for each machine type (1000 sf)
      cacq[]     acquisition costs ($10K) per machine ot type m
      cmnt[]     fixed annual maintenance cost ($10K) per machine
      dvol[]     input mail stream volumes
      arrivd[][] arrival stream percent volume per period profiles
      pay[]      hourly wage rate for operators on machine m      */

FILE  *infp,*outfp1,*outfp2,*outfp3,*outfp4,*outfp5,*outfp6,*fopen();

main ()
{
           /*  open I/O files and assign pointers  */

      infp=fopen("GMFA.DAT","r");
      outfp5=fopen("XML5.DAT","w");

           /* read in input stream data from "GMFA.DAT"  */
      rddstrm ();
           /* read in node information from "GMFA.DAT"  */
      rdnode ();

           /*-----------------------------------------------------*/
           /* write decision variables for equation i in VARDi () */
           /*-----------------------------------------------------*/

      outfp1=fopen("XML1.MOD","w");
      fprintf(outfp1,"$  POSTAL SORTING PROBLEM\n$  DECISIONS\n");
      vard2 ();
      vard3 ();
      vard4 ();
      vard5 ();
      vard6 ();
                   /* variables for eqn 7 already declared */
      vard8 ();
      vard13 ();
      vard14 ();
      fclose(outfp1);

           /*-----------------------------------------------------*/
           /* write output equations by routine EQNi ()           */
           /*-----------------------------------------------------*/

      outfp2=fopen("XML2.MOD","w");
      fprintf(outfp2,"$  EQUATIONS\n");
      eqn2 ();
      eqn3 ();
      fclose(outfp2);
      outfp3=fopen("XML3.MOD","w");
      eqn4 ();
      eqn5 ();
      eqn6 ();
      eqn7 ();
      eqn8 ();
      eqn13 ();
      eqn14 ();
```

```
        eqn15 ();
        eqn16 ();
        fclose(outfp3);

            /*-------------------------------------------------*/
            /* write limits for equation i by routine LIMi ()   */
            /*-------------------------------------------------*/

        outfp4=fopen("XML4.MOD","w");
        outfp5=fopen("AIT.DAT","w");
        fprintf (outfp4,"$  LIMITS\n");
        lim2();
        lim3();
        lim4();
        lim5();
        lim6();
        lim7();
        lim8();
        lim13();
        lim14();
        lim15();

            /*-------------------------------------------------*/
            /*  header for the bounds section                   */
            /*-------------------------------------------------*/
        fprintf(outfp4,"$  BOUNDS\n");

            /*-------------------------------------------------*/
            /*  finish off the output file by assigning objective */
            /*  and closing the files                            */
            /*-------------------------------------------------*/
        fprintf(outfp4,"$ MAXIMIZE TOTCOSTS\n");
        fclose(outfp4);
        fclose(infp);

        printf ("\n\nALL DONE !!!");
        printf ("\n\noutput files XML.MOD about %ld bytes...",pos);
        printf ("\n\nthere are %ld constraint equations...",numeqn);
        printf ("\n\nthere are %ld decision variables...",numvar);
        printf ("\nHAVE A NICE DAY !!!!!");
}

/*--------------------------------------------------------------------*/

rddstrm                          /* read arrival stream data */
{                                /* from GMFA.DAT            */

        int  icnt=0, tcnt=0;
        float  tmptot=0;

        while (++icnt < D_MAX) {
          fscanf(infp,"%d  %d  %f  %d\n",&dstrm[icnt],&prof[icnt],
                                     &dvol[icnt],&anode[icnt]);
          tmptot += dvol[icnt];
        }
```

```
                                    /* print incoming stream volume */

      printf("\n\ntmptot= %f  kpieces\n\n\n",tmptot);
      fscanf(infp,"\n");

                                    /* read stream profile data */

      while (++tcnt < DTIMES) {
        fscanf(infp,"%f  %f  %f\n",&arrivd[1][tcnt],&arrivd[2][tcnt],
                                   &arrivd[3][tcnt]);
      }
}
/*-----------------------------------------------------------------------*/

rdnode ()                      /* read in the stream data for */
{                              /* processing nodes            */
      int  n=0, jcnt=0, in, totrd;

      fscanf(infp,"\n");
                          /* read in machine type, tnl(n), tnr(n),
                             number of input streams, and the
                             number of output streams          */
      while (++n < NODES) {
        fscanf(infp,"%d  %d  %d  %d  %d  %d\n",
             &in,&mach[n],&tnl[n],&tnr[n],&inum[n],&onum[n]);
        totrd = MAX (inum[n],onum[n]);

                          /* to read the stream data for this node
                             need to know how many lines to
                             read (the max of #i or #o)...     */
          while (++jcnt <= totrd) {
            fscanf(infp," %d  %d  %d  %d\n",&istrm[jcnt][n],
                &lnode[jcnt][n],&ostrm[jcnt][n],&pnode[jcnt][n]);
          }
          jcnt=0;
      }
}

/*-----------------------------------------------------------------------*/


vard2 ()        /* declares the variables needed for eqn 2: x(i,l,t),
                   v(i,t), and s(i,t)                         */
{
      int  ncnt=0, icnt=0, tcnt=0, ocnt=0, iret=0;

                /* To ensure that a variable x(i,l,t) exists only
                   if some x(o,n,t) creates it...
                   x(i,l,t) taken care of by declaring
                   all of the x(o,n,t+1) possible.            */

      while (++ncnt < (NODES))  {
        while (++ocnt <= onum[ncnt]) {
          tcnt=tnl[ncnt];
        /* Can only send mail up until tnr of node p    */

      while (++tcnt <= (tnr[ncnt]+1))  {
        if(( tcnt <= tnr[pnode[ocnt][ncnt]]) &&
           (pnode[ocnt][ncnt] <= 14))
          {
          fprintf(outfp1,"x%03d%02d%02d  ",ostrm[ocnt][ncnt],
                    ncnt,tcnt);
          ++iret;
          ++numvar;
          if (((iret/7) >= 1.0) && ((iret % 7) == 0))
             fprintf(outfp1,"\n");
          }
      }
    }
    ocnt=0;
  }
  if ((iret % 7) != 0)
     {
     fprintf(outfp1,"\n");
     iret=0;
     }
  ncnt=0;
  while (++ncnt < NODES-1) {
    icnt=0;
    while (++icnt <= inum[ncnt]) {
      tcnt=tnl[ncnt];
      while (++tcnt <= tnr[ncnt]) {
        fprintf(outfp1,"v%03d%02d  ",istrm[icnt][ncnt],tcnt);
        ++iret;
        ++numvar;
        if (((iret/9) >= 1.0) && ((iret % 9) == 0))
          fprintf(outfp1,"\n");
      }
      if (istrm[icnt+1][ncnt] == istrm[icnt][ncnt]) ++icnt;
      if (istrm[icnt+1][ncnt] == istrm[icnt][ncnt]) ++icnt;
      if (istrm[icnt+1][ncnt] == istrm[icnt][ncnt]) ++icnt;
    }
  }

  if ((iret % 9) != 0)
     {
     fprintf(outfp1,"\n");
     iret=0;
     }
  ncnt=0;
  while (++ncnt < (NODES-1)) {
    icnt=0;
    while (++icnt <= inum[ncnt]) {
      tcnt=tnl[ncnt]-1;
        while (++tcnt <= tnr[ncnt]) {
          fprintf(outfp1,"s%03d%02d  ",istrm[icnt][ncnt],tcnt);
          ++numvar;
          ++iret;
          if (((iret/9) >= 1.0) && ((iret % 9) == 0))
             fprintf(outfp1,"\n");
```

```
            }
            if (istrm[icnt+1][ncnt] == istrm[icnt][ncnt]) ++icnt;
            if (istrm[icnt+1][ncnt] == istrm[icnt][ncnt]) ++icnt;
            if (istrm[icnt+1][ncnt] == istrm[icnt][ncnt]) ++icnt;
        }
      }

      if ((iret % 9) != 0)
      fprintf(outfp1,"\n");
      pos=ftell(outfp1);
      printf("\nOUTPUT FILE 1 IS %ld BYTES SO FAR (vard2)\n",pos);
}

/*-------------------------------------------------------------------*/

vard3 ()          /* must declare the output streams going to */
{                 /* destinating nodes (15 through 24)...    */

      int  ncnt, jcnt, tcnt, icarp=0;

      ncnt=0;
      while (++ncnt < NODES-1)    {
        jcnt=0;
        while (++jcnt <= onum[ncnt])  {
          tcnt = tnl[ncnt] - 1;
          while (++tcnt <= tnr[ncnt]) {
            if (pnode[jcnt][ncnt] > 14) {
              fprintf(outfp1,"x%03d%02d%02d  ",ostrm[jcnt][ncnt],
                                ncnt,tcnt+1);
              icarp += 10;
              ++numvar;
              if (icarp > 60)
                {
                fprintf(outfp1,"\n");
                icarp=0;
                }
            }
          }
        }
      }
      if (icarp != 0) fprintf(outfp1,"\n");
      pos=ftell(outfp1);
      printf("\nOUTPUT FILE 1 IS %ld BYTES SO FAR (vard3)\n",pos);
}

/*-------------------------------------------------------------------*/

vard4 ()         /* declare variables associated with eqn 4...only
                    w(i,t) are not yet declared...                 */
{
      int  ncnt=0, icnt=0, tcnt=0, iret=0;

      while (++ncnt < (NODES-1)) {
        icnt=0;
        while (++icnt <= inum[ncnt]) {
          tcnt=tnl[ncnt]-1;
          while (++tcnt <= tnr[ncnt]) {
            fprintf(outfp1,"w%03d%02d  ",istrm[icnt][ncnt],tcnt);
            ++iret;
            ++numvar;
            if (((iret/9) >= 1.0) && ((iret % 9) == 0))
              fprintf(outfp1,"\n");
          }
          if (istrm[icnt+1][ncnt] == istrm[icnt][ncnt])  ++icnt;
          if (istrm[icnt+1][ncnt] == istrm[icnt][ncnt])  ++icnt;
          if (istrm[icnt+1][ncnt] == istrm[icnt][ncnt])  ++icnt;
        }
      }
      if ((iret % 9) != 0)
      fprintf(outfp1,"\n");
      pos=ftell(outfp1);
      printf("\nOUTPUT FILE 1 IS %ld BYTES SO FAR (vard4)\n",pos);
}

/*--------------------------------------------------------------------*/

vard5()          /*  declare variables for eqn 5:  y(n,t)  */
{
      int ncnt=0, tcnt=0, iret=0;

      while (++ncnt < NODES-1) {
        tcnt=tn1[ncnt]-1;
        while (++tcnt <= tnr[ncnt]) {
          fprintf(outfp1,"y%02d%02d  ",ncnt,tcnt);
          ++numvar;
          if (((++iret/10) >= 1.0) && ((iret % 10) == 0))
            fprintf(outfp1,"\n");
        }
      }
      if ((iret % 10) != 0)
      fprintf(outfp1,"\n");
      pos=ftell(outfp1);
      printf("\nOUTPUT FILE 1 IS %ld BYTES SO FAR (vard5)\n",pos);
}

/*--------------------------------------------------------------------*/

vard6 ()         /* only one new variable needed for eqn 6  */
{
      fprintf(outfp1,"x15\n");
      ++numvar;
}

/*--------------------------------------------------------------------*/

vard8 ()         /* declare variables for sequencing node      */
{                /* equations 8 through 11...                   */

      fprintf (outfp1,"x2581536  ");
      fprintf (outfp1,"x0241640  ");
      fprintf (outfp1,"x2581544  ");
      fprintf (outfp1,"x0241648  ");
      numvar+=4;
      fprintf (outfp1,"\n");
      pos=ftell(outfp1);
      printf("\nOUTPUT FILE 1 IS %ld BYTES SO FAR (vard8)\n",pos);
}

/*---------------------------------------------------------------------*/

vard13()        /* additional floorspace required             */
{
      fprintf(outfp1,"fsadded  \n");
      numvar += 1;
}

/*---------------------------------------------------------------------*/

vard14()        /* yy(n): the total number of machines required    */
{               /*        at each processing station (these will   */
                /*        be the integer variables.                */

      int mcnt=0, ncnt=0, icarp=0;

      while (++ncnt < (NODES-1)) {
        fprintf(outfp1,"yy%02d  ",ncnt);
        icarp += 6;
        if (icarp >= 60)
          {
          fprintf(outfp1,"\n");
          icarp=0;
          }
        ++numvar;
      }
                /* ptot(m): the total number of operator hours     */
                /*          on all machines of type m              */

      mcnt=0;
      while (++mcnt < MACHTOT) {
        fprintf(outfp1,"ptot%01d  ",mcnt);
        ++numvar;
      }
      fprintf(outfp1,"\n");
      pos=ftell(outfp1);
      printf("\nOUTPUT FILE 1 IS %ld BYTES SO FAR (vard14)\n",pos);
}

/*-------------------------------------------------------------------*/

eqn2 ()
{     /*  write the set of conservation constraints for the
          inventory nodes:
             dos(row) = x(i,1,t) + v(i,t) - s(i,t) - v(i,t+1)
               v(i,tn1)=0 for all i              for all n
               v(i,tnr+1)=0 for all i                all In
               dos(row)= a(i,t) for i in A           all Tn
                       = 0      for i not in A                  */
```

```
int icnt=0, tcnt=0, ncnt=0, irow=0, itoe=0, idum=0, icarp=0;
int tbum=0;

while (++ncnt < (NODES-1)) {
  icnt=0;
  while (++icnt <= inum[ncnt]) {
    tcnt = tnl[ncnt]-1;
    while (++tcnt <= tnr[ncnt]) {
      itoe=0;
      idum=1;
      fprintf(outfp2,"dos%04d = ",++irow);
      icarp += 10;

       /*  a little tricky here...if lnode is 0 this is an
           arrival stream...d(i,t) not x(i,l,t)...          */

      if((lnode[icnt][ncnt]) == 0) goto lab1;

      if((tcnt == tnl[ncnt]) && (tcnt <= tnr[lnode[icnt][ncnt]]+1)
                     && (tcnt > tnl[lnode[icnt][ncnt]]+1))
        {
        tbum=tnl[lnode[icnt][ncnt]];
        while(++tbum < tnl[ncnt])  {
          fprintf(outfp2,"x%03d%02d%02d + ",istrm[icnt][ncnt],
                         lnode[icnt][ncnt],tbum);
          icarp += 11;
          if(icarp > 60)
            {
            fprintf(outfp2,"\n          ");
            icarp = 10;
            }
        }
      }
     if((tcnt <= tnr[lnode[icnt][ncnt]] + 1) &&
           (tcnt >= tnl[lnode[icnt][ncnt]] + 1))
       {
       fprintf(outfp2,"x%03d%02d%02d + ",istrm[icnt][ncnt],
                      lnode[icnt][ncnt],tcnt);
       icarp += 9;
       if (icarp > 60)
         {
         fprintf(outfp2,"\n          ");
         icarp = 10;
         }
       }

     itoe = 0;
     idum = 1;

     lab77: if((istrm[icnt+idum-1][ncnt] == istrm[icnt+idum][ncnt])
                 && (tcnt == tnl[ncnt])
                 && (tcnt > tnl[lnode[icnt+idum][ncnt]] + 1)
                 && (tcnt <= tnr[lnode[icnt+idum][ncnt]] + 1))
        {
        tbum=tnl[lnode[icnt+idum][ncnt]];
        while(++tbum < tnl[ncnt])  {
```

```
        fprintf(outfp2,"x%03d%02d%02d + ",istrm[icnt+idum][ncnt],
                    lnode[icnt+idum][ncnt],tbum);
        icarp += 11;
        if (icarp > 60)
          {
          fprintf(outfp2,"\n          ");
          icarp = 10;
          }
      }
   }

   if((istrm[icnt+idum-1][ncnt] == istrm[icnt+idum][ncnt])
       && (tcnt <= tnr[lnode[icnt+idum][ncnt]] + 1) &&
       (tcnt >= tnl[lnode[icnt+idum][ncnt]] + 1))
     {
     fprintf(outfp2,"x%03d%02d%02d + ",
           istrm[icnt+idum][ncnt],lnode[icnt+idum][ncnt],
           tcnt);
     icarp += 10;
     if (icarp > 60)
       {
       fprintf(outfp2,"\n          ");
       icarp = 10;
       }
     }

     if (istrm[icnt+idum-1][ncnt] == istrm[icnt+idum][ncnt])
       {
       idum +=1;
       itoe +=1;
       }
      else
        idum=4;
     if (idum < 4) goto lab77;

 lab1: if(tcnt > tnl[ncnt])
   {
   fprintf(outfp2,"v%03d%02d ",istrm[icnt][ncnt],tcnt);
   icarp += 9;
   if(icarp > 60)
     {
     fprintf(outfp2,"\n          ");
     icarp = 10;
     }
   }
 if (tcnt < tnr[ncnt])
   {
   fprintf(outfp2,"- v%03d%02d ",istrm[icnt][ncnt],tcnt+1);
   icarp += 9;
   if (icarp > 60)
     {
     fprintf(outfp2,"\n          ");
     icarp = 10;
     }
 }
 fprintf(outfp2,"- s%03d%02d\n",istrm[icnt][ncnt],tcnt);
```

```
          icarp = 0;
         }
         icnt +=itoe;
        }
        icnt=0;
       }
       cntr2=irow;
       numeqn += irow;
       pos=ftell(outfp2);
       printf("\nOUTPUT FILE 2 IS %ld BYTES SO FAR (eqn2)\n",pos);
}

/*---------------------------------------------------------------------------*/

eqn3 ()  /*  write the set of eqn 3 (output conservation)         */
         /*   tres(row) = sum all In (fion*s(i,t)) - x(o,n,t+1)  */
         /*                            for all n, all On, all Tn  */
         /*   read values for the fion to matrix fion[i][o]       */
         /*   which is temporary used (one node at a time).       */

         /*  Logic goes:                                  */
         /*    1.  read fion[i][o] for current node n     */
         /*    2.  write eqn for the current node n       */
         /*    3.  n=n+1; return to 1.                    */
{
      int icnt=0, ncnt=0, tcnt=0, jcnt=0, ir3=0, icarp=0;
      float  fion[12][10], tchk=0.0;

      fscanf(infp,"\n");
      while (++ncnt < (NODES-1)) {
        while (++icnt <= inum[ncnt]) {
          fscanf(infp,"\n");
          jcnt=0;
          tchk=0.0;
          while (++jcnt <= onum[ncnt]) {
            fscanf (infp,"  %f",&fion[icnt][jcnt]);
            tchk += fion[icnt][jcnt];
          }
          if ((tchk < 0.99999) || (tchk > 1.00001))
          printf("\nERROR IN BRANCH AT NODE %02d,STREAM %02d\n",
                  ncnt,istrm[icnt][ncnt]);
        }
        icnt=0;
        jcnt=0;

        while (++jcnt <= onum[ncnt]) {
          tcnt = tnl[ncnt] - 1;
          while (++tcnt <= tnr[ncnt]) {
            if ((tcnt > tnr[pnode[jcnt][ncnt]] - 1) &&
                (pnode[jcnt][ncnt] <= 14)) goto lab99;

            fprintf(outfp2,"tres%04d =",++ir3);
            icarp += 10;
            icnt=0;
            while (++icnt <= inum[ncnt]) {
              fprintf(outfp2," %5.3fs%03d%02d ",fion[icnt][jcnt],
                istrm[icnt][ncnt],tcnt);
            icarp += 13;
            if(istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
            if(istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
            if(istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;

            if (icarp > 60)
              {
              fprintf(outfp2,"\n           ");
              icarp=11;
              }

          if (icnt < inum[ncnt])
          fprintf(outfp2,"+");
          ++icarp;
        }
        if (( tcnt <= tnr[pnode[jcnt][ncnt]]-1) ||
                (pnode[jcnt][ncnt] > 14))
        fprintf(outfp2,"- x%03d%02d%02d\n",ostrm[jcnt][ncnt],
                   ncnt,tcnt+1);
          else
          fprintf(outfp2,"\n");
        lab99: icarp=0;
       }
     }
     icnt=0;
     jcnt=0;
    }
    cntr3=ir3;
    numeqn += ir3;
    pos=ftell(outfp2);
    printf("\nOUTPUT FILE 2 IS %ld BYTES SO FAR (eqn3)\n",pos);
}

/*-------------------------------------------------------------------*/

eqn4 ()        /* write the set of eqn 4 (processing rate    */
{              /* constraints...                             */

    int icnt=0, ncnt=0, tcnt=0, jcnt=0, ir3=0, ocnt=0;
    int iclick=0;
    float  fion[12][INMAX], rate[INMAX], factr, oorate[INMAX];

    fscanf(infp,"\n\n");
    while (++ncnt < (NODES-1)) {
      iclick=0;
      while (++icnt <= inum[ncnt]) {
        fscanf(infp,"\n");
        jcnt=0;
        fscanf(infp,"  %f",&rate[icnt]);
        while (++jcnt <= onum[ncnt]) {
          fscanf (infp,"  %f",&fion[icnt][jcnt]);
        }
      }
      ocnt=0;
      if (rate[1] != 0) goto lab2;
```

```
          iclick=7;  /* if iclick=7, processing rate depends on output */
          while (++ocnt <= onum[ncnt]) {
            fscanf (infp,"  %f",&oorate[ocnt]);
          }
        lab2: icnt=0;
        icnt=0;
        ocnt=0;
        while (++icnt <= inum[ncnt]) {
          tcnt = tnl[ncnt] - 1;
          while (++tcnt <= tnr[ncnt])  {
            fprintf(outfp3,"quat%04d = ",++ir3);
            if (iclick == 7)
              {
              jcnt=0;
              factr=0.0;
              while (++jcnt <= onum[ncnt])  {
                factr=fion[icnt][jcnt] * oorate[jcnt] + factr;
               }
              fprintf(outfp3,"%8.3fw%03d%02d ",factr,
                       istrm[icnt][ncnt],tcnt);
              }
            else
              {
              factr=rate[icnt];
              fprintf(outfp3,"%8.3fw%03d%02d ",factr,
                       istrm[icnt][ncnt],tcnt);
              }
            fprintf(outfp3,"- s%03d%02d\n",istrm[icnt][ncnt],tcnt);
          }
          if(istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
          if(istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
          if(istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
        }
       icnt=0;
       jcnt=0;
      }
      cntr4=ir3;
      numeqn += ir3;
      pos=ftell(outfp3);
      printf("\nOUTPUT FILE 3 IS %ld BYTES SO FAR (eqn4)\n",pos);
}

/*--------------------------------------------------------------------*/

eqn5 ()
{           /*    Equation 5 (computing the y(n,t)):          */
            /*       cinc(row) = sum of I {w(i,t)} - y(n,t)   */
            /*                  for all n, all Tn             */
            /*       processing availability constraints      */

      int icnt=0, tcnt=0, ncnt=0, ir5=0, icarp=0;

      while (++ncnt < NODES-1) {
         tcnt=tnl[ncnt]-1;
         while (++tcnt <= tnr[ncnt]) {
           fprintf(outfp3,"cinc%03d =",++ir5);
```

```
            icarp += 9;
            icnt=0;
            while (++icnt <= inum[ncnt]) {
              fprintf(outfp3," w%03d%02d",istrm[icnt][ncnt],tcnt);
              icarp += 7;
              if (istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
              if (istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
              if (istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
              if (icarp > 60)
                {
                fprintf(outfp3,"\n        ");
                icarp=7;
                }
              if (icnt == inum[ncnt])
              fprintf(outfp3," -");
              else
                fprintf(outfp3," +");
              icarp += 2;
            }
          fprintf(outfp3," y%02d%02d\n",ncnt,tcnt);
          icarp=0;
        }
      }
      cntr5=ir5;
      numeqn += ir5;
      pos=ftell(outfp3);
      printf("\nOUTPUT FILE 3 IS %ld BYTES SO FAR (eqn5)\n",pos);
}

/*---------------------------------------------------------------------*/

eqn6 ()         /* sequencing volume constraint  */
{
      int icnt=0, tcnt=0, icarp=0, itmp=0;

      fprintf(outfp3,"seis = 2x15");
      icarp += 11;
      while (++icnt <= inum[15]) {
        itmp=lnode[icnt][15];
        tcnt=tnl[itmp];
        while (++tcnt <= (tnr[itmp]+1)) {
          fprintf(outfp3," - x%03d%02d%02d",istrm[icnt][15],
                           itmp,tcnt);
          icarp += 11;
          if (icarp > 60)
            {
            fprintf(outfp3,"\n  ");
            icarp=2;
            }
        }
      }
      if (icarp != 2) fprintf(outfp3,"\n");
      numeqn += 1;
      pos=ftell(outfp3);
      printf("\nOUTPUT FILE 3 IS %ld BYTES SO FAR (eqn6)\n",pos);
}
```

```
/*------------------------------------------------------------------*/

eqn7 ()         /* constraint to ensure that half the sequencing
                   volume arrives before the first pass period and
                   to limit late arrivals...                        */
{
      int  icnt=0, tcnt=0, irow=0;

      tcnt=TWINDOW-3;
      while (++tcnt <= TWINDOW) {
        fprintf (outfp3,"cien%02d = 0.333x15",++irow);
        while (++icnt <= inum[15]) {
          if(tcnt <= tnr[lnode[icnt][15]]+1)
          fprintf (outfp3," - x%03d%02d%02d",istrm[icnt][15],
                          lnode[icnt][15],tcnt);
        }
        icnt=0;
        fprintf(outfp3,"\n");
      }
      cntr7=irow;
      numeqn += irow;
      pos=ftell(outfp3);
      printf("\nOUTPUT FILE 3 IS %ld BYTES SO FAR (eqn7)\n",pos);
}

/*------------------------------------------------------------------*/

eqn8 ()         /* sequencing processing constraints      */
{
      int  b=0;

      fprintf (outfp3,"p1cg1suc = 0.985x15 - x2581536\n");
      while (++b <= 4)  {
        fprintf(outfp3,"p1cg1rj%d = 0.00375x15 - x07915%02d\n",b,b+31);
      }
      b=0;
      fprintf (outfp3,"p2cg1suc = 0.990x2581536 - x0241840\n");
      while (++b <= 4)  {
        fprintf(outfp3,"p2cg1rj%d = 0.0025x2581536 - x07915%02d\n",
                                   b,b+35);
      }
      b=0;
      fprintf (outfp3,"p1cg2suc = 0.985x15 - x2581544\n");
      while (++b <= 4)  {
        fprintf(outfp3,"p1cg2rj%d = 0.00375x15 - x07915%02d\n",b,b+39);
      }
      b=0;
      fprintf (outfp3,"p2cg2suc = 0.990x2581544 - x0241848\n");
      while (++b <= 4)  {
        fprintf(outfp3,"p2cg2rj%d = 0.0025x2581544 - x07915%02d\n",
                                   b,b+43);
      }
      numeqn += 20;
      pos=ftell(outfp3);
      printf("\nOUTPUT FILE 3 IS %ld BYTES SO FAR (eqn8)\n",pos);
```

```
}
/*------------------------------------------------------------------*/

eqn13 ()       /* constraint for additional floorspace      */
{
      int  ncnt=0, icarp=0;
      fprintf(outfp3,"floor = - fsadded");
      icarp += 17;
      while (++ncnt < (NODES-1)) {
        fprintf(outfp3," + %5.3fyy%02d",fsp[mach[ncnt]],ncnt);
        icarp += 12;
        if (icarp > 60)
          {
          fprintf(outfp3,"\n     ");
          icarp = 5;
          }
      }
      fprintf(outfp3,"\n");
      numeqn += 1;
}
/*------------------------------------------------------------------*/

eqn14 ()       /* the required number of machines at each n is
                  yy(n), > or = y(n,t) for all n all Tn       */
{
      int  ncnt=0, tcnt=0, irow=0;

      while (++ncnt < (NODES-1)) {
        tcnt = tnl[ncnt] - 1;
        while (++tcnt <= tnr[ncnt]) {
          fprintf(outfp3,"mnum%03d = ",++irow);
          fprintf(outfp3,"y%02d%02d ",ncnt,tcnt);
          fprintf(outfp3,"- yy%02d\n",ncnt);
        }
      }

      cntr14=irow;
      numeqn += irow;
      pos=ftell(outfp3);
      printf("\nOUTPUT FILE 3 IS %ld BYTES SO FAR (eqn14)\n",pos);
}
/*------------------------------------------------------------------*/

eqn15 ()         /* computing the total operator hours for each
                    type of machine m....                    */
{
      int  mcnt=0, ncnt=0, tcnt=0, irow=0, icarp=0, icnthg=0;

      while (++mcnt < MACHTOT) {
      icnthg=0;
      tcnt=0;
      while (++tcnt < DTIMES) {
        ncnt=0;
        while (++ncnt < (NODES-1))  {
          if((mach[ncnt] == mcnt) && (tcnt >= tn1[ncnt]) &&
                 (tcnt <= tnr[ncnt]))
            icnthg += 1;
        }
      }
      if (icnthg > 0)
        {
        tcnt = 0;
        fprintf (outfp3,"pers%01d = ",++irow);
        icarp += 8;
        while (++tcnt < DTIMES) {
          ncnt=0;
          while (++ncnt < NODES-1) {
            if ((mach[ncnt] == mcnt) && (tcnt >= tn1[ncnt]) &&
                   (tcnt <= tnr[ncnt]))
              {
              fprintf(outfp3,"%dy%02d%02d",p[mcnt],ncnt,tcnt);
              icnthg -= 1;
              if (icnthg > 0) fprintf(outfp3," + ");
                icarp += 10;
              if (icarp > 60)
                {
                fprintf(outfp3,"\n        ");
                icarp=8;
                }
              }
          }
        }
        fprintf (outfp3," - 2.0ptot%01d\n",mcnt);
        icarp=0;
       }
      }
      cntr15=irow;
      numeqn += irow;
      pos=ftell(outfp3);
      printf("\nOUTPUT FILE 3 IS %ld BYTES SO FAR (eqn15)\n",pos);
}

/*-------------------------------------------------------------------*/

eqn16 ()        /* and at last, the objective functions         */
{
      int  mcnt=0, icarp=0, ncnt=0;
      float costmp=0.0;

      fprintf(outfp3,"totcosts = ");
      printf("totcosts = ");
      icarp += 11;
      while (++ncnt < (NODES-1)) {
        costmp = cacq[mach[ncnt]] + 6.71*cmnt[mach[ncnt]];
        fprintf(outfp3," - %7.3fyy%02d",costmp,ncnt);
        printf(" - %7.3fyy%02d",costmp,ncnt);
        icarp += 14;
        if (icarp > 60)
          {
            fprintf(outfp3,"\n      ");
            printf("\n      ");
            icarp = 6;
            }
        }
        mcnt=0;
        while (++mcnt <= 6) {
          costmp = 0.20935 * pay[mcnt];
          fprintf(outfp3," - %7.4fptot%d",costmp,mcnt);
          printf(" - %7.4fptot%d",costmp,mcnt);
          icarp += 15;
          if (icarp > 60)
            {
            fprintf(outfp3,"\n      ");
            printf("\n      ");
            icarp = 6;
            }
        }
        fprintf(outfp3," - %4.1ffsadded\n",CPSF);
        printf(" - %4.1ffsadded\n",CPSF);
        numeqn += 1;
        pos=ftell(outfp3);
        printf("\nOUTPUT FILE 3 IS %ld BYTES SO FAR (eqn16)\n",pos);
}

/*---------------------------------------------------------------------*/
/*  The following set the limits on each of the equations              */
/*---------------------------------------------------------------------*/

lim2()
{
        int  icnt=0, tcnt=0, ncnt=0, irow=0, tt=0, ii=0;
        float  tempd=0.0;

        while (++ncnt < (NODES-1)) {
          icnt=0;
          while (++icnt <= inum[ncnt]) {
            tcnt=tnl[ncnt] - 1;
            while (++tcnt <= tnr[ncnt]) {
              tempd=0.0;
              fprintf(outfp4,"dos%04d = ",++irow);
              if (lnode[icnt][ncnt] != 0) goto lab12;

              if (tcnt == tnl[ncnt])
                {
                tt=0;
                while (++tt <= tnl[ncnt]) {
                  ii=0;
                  while (++ii < D_MAX) {
                    if (dstrm[ii] == istrm[icnt][ncnt])
                    tempd = tempd - dvol[ii]*arrivd[prof[ii]][tt]*0.01;
                  }
                }
              }
              else
                {
```

```
                ii=0;
                while(++ii < D_MAX) {
                  if(dstrm[ii] == istrm[icnt][ncnt])
                  tempd = -dvol[ii]*arrivd[prof[ii]][tcnt]*0.01;
                 }
                }
             lab12: fprintf(outfp4,"%8.3f\n",tempd);
            }
            if(istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
            if(istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
            if(istrm[icnt][ncnt] == istrm[icnt+1][ncnt]) ++icnt;
          }
        }
}

/*---------------------------------------------------------------------*/

lim3()
{
        int icnt=0;
        while (++icnt <= cntr3) {
          fprintf(outfp4,"tres%04d = 0\n",icnt);
        }
        pos=ftell(outfp4);
        printf("\nOUTPUT FILE 4 IS %ld BYTES SO FAR (lim3)\n",pos);
}

/*---------------------------------------------------------------------*/

lim4()
{
        int icnt=0;
        while (++icnt <= cntr4) {
          fprintf(outfp4,"quat%04d > 0\n",icnt);
        }
        pos=ftell(outfp4);
        printf("\nOUTPUT FILE 4 IS %ld BYTES SO FAR (lim4)\n",pos);
}

/*---------------------------------------------------------------------*/

lim5()
{
        int icnt=0;
        while (++icnt <= cntr5) {
          fprintf(outfp4,"cinc%03d < 0\n",icnt);
        }
        pos=ftell(outfp4);
        printf("\nOUTPUT FILE 4 IS %ld BYTES SO FAR (lim5)\n",pos);
}

/*---------------------------------------------------------------------*/

lim8()
{
        fprintf(outfp4,"seis = 0\n");
```

```
      pos=ftell(outfp4);
      printf("\nOUTPUT FILE 4 IS %ld BYTES SO FAR (lim6)\n",pos);
}

/*-------------------------------------------------------------------*/

lim7()
{
      int icnt=0;
      while (++icnt <= cntr7) {
        fprintf(outfp4,"cien%02d > 0\n",icnt);
      }
      pos=ftell(outfp4);
      printf("\nOUTPUT FILE 4 IS %ld BYTES SO FAR (lim7)\n",pos);
}

/*-------------------------------------------------------------------*/

lim8()
{
      int b=0;
      fprintf(outfp4,"p1cg1suc = 0\n");
      while (++b <= 4) {
        fprintf (outfp4,"p1cg1rj%d = 0\n",b);
      }
      b=0;
      fprintf(outfp4,"p2cg1suc = 0\n");
      while (++b <= 4) {
        fprintf (outfp4,"p2cg1rj%d = 0\n",b);
      }
      b=0;
      fprintf(outfp4,"p1cg2suc = 0\n");
      while (++b <= 4) {
        fprintf (outfp4,"p1cg2rj%d = 0\n",b);
      }
      b=0;
      fprintf(outfp4,"p2cg2suc = 0\n");
      while (++b <= 4) {
        fprintf(outfp4,"p2cg2rj%d = 0\n",b);
      }
      pos=ftell(outfp4);
      printf("\nOUTPUT FILE 4 IS %ld BYTES SO FAR (lim8)\n",pos);
}

/*-------------------------------------------------------------------*/

lim13 ()
{
      fprintf(outfp4,"floor < %5.1f\n",SPACE);
}

/*-------------------------------------------------------------------*/

lim14 ()
{
      int icnt=0;
      while (++icnt <= cntr14) {
        fprintf(outfp4,"mnum%03d < 0\n",icnt);
      }
      pos=ftell(outfp4);
      printf("\nOUTPUT FILE 4 IS %ld BYTES SO FAR (lim14)\n",pos);
}

/*------------------------------------------------------------------------*/

lim15 ()
{
      int icnt=0;

      while (++icnt <= cntr15) {
        fprintf(outfp4,"pers%01d < 0\n",icnt);
      }
      pos=ftell(outfp4);
      printf("\nOUTPUT FILE 4 IS %ld BYTES SO FAR (lim15)\n",pos);
}

/*--------------------------------------------------------------------*/
/******                     end of MGP1.C                      ******/
/*--------------------------------------------------------------------*/
```

```
001  001  0800.0  1          ****************************************
015  001  0692.0  2          *  GMFA.DAT is the input file for MGP1 *
016  001  0052.0  2          *  the postal GMF-A model generator.   *
012  002  0287.0  7          *  There are five data blocks.         *
013  003  0163.0  7          ****************************************
053  002  0489.0  8
054  002  0029.0  8
055  002  0019.0  8          ¦
056  003  0319.0  8          ¦  Block 1:
057  002  0001.7  8          ¦
058  003  0028.0  8          ¦  This first block is the arrival stream
063  002  0101.0  8          ¦  identification.  Format is stream i.d.,
154  002  0475.0  9          ¦  which arrival profile, stream volume in
002  001  0056.0  10         ¦  kp, and station number for insertion.
004  002  0083.0  10         ¦
005  002  0001.3  10         ¦  Format must be followed throughout.
006  003  0020.0  10         ¦
010  002  0015.0  11         ¦
011  003  0009.0  11         ¦
099  002  0067.0  12         ¦

01.05  10.90  37.65          ¦¦
01.05  10.90  37.65          ¦¦
01.60  03.45  04.05          ¦¦  Second block is the arriving mail
01.60  03.45  04.05          ¦¦  stream profiles for collection,
00.75  02.95  01.70          ¦¦  managed, and business mail.  Data
00.75  02.95  01.70          ¦¦  are proportions of the total stream
01.00  03.95  01.30          ¦¦  volume each 30 minute period.
01.00  03.95  01.30          ¦¦
02.60  03.45  01.10          ¦¦  First row is 0700-0730.
02.60  03.45  01.10          ¦¦
02.20  01.50  01.50          ¦¦
02.20  01.50  01.50          ¦¦
02.15  05.45  00.90          ¦¦
02.15  05.45  00.90          ¦¦
10.30  01.00  00.70          ¦¦
10.30  01.00  00.70          ¦¦
03.45  02.95  00.45          ¦¦
03.45  02.95  00.45          ¦¦
15.05  00.50  00.05          ¦¦
15.05  00.50  00.05          ¦¦
08.25  02.00  00.00          ¦¦
08.25  02.00  00.00          ¦¦
01.55  00.00  00.20          ¦¦
01.55  00.00  00.20          ¦¦
00.05  01.50  00.00          ¦¦
00.05  01.50  00.00          ¦¦
00.00  02.45  00.00          ¦¦
00.00  02.45  00.00          ¦¦
00.00  07.95  00.40          ¦¦
00.00  07.95  00.40          ¦¦
00.00  00.00  00.00          ¦¦
00.00  00.00  00.00          ¦¦
00.00  00.00  00.00          ¦¦
00.00  00.00  00.00          ¦¦
00.00  00.00  00.00          ¦¦
```

```
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !
00.00  00.00  00.00                    ! !

01  01  01  32  1  3                   !  Block 3: Processing station input
  01  00  003  10                      !  and output streams.  Each station
  00  00  014  02                      !  has first line data in the form:
  00  00  152  04                      !  station #, machine type #, first
02  02  01  33  3  9                   !  possible operating period, last
  14  01  019  03                      !  possible operating period, number
  15  00  021  18                      !  of input streams, and number of
  16  00  065  05                      !  output streams.
  00  00  066  05                      !
  00  00  067  06                      !  This first line is followed by
  00  00  069  06                      !  four columns:
  00  00  073  12                      !    Col 1: Input stream numbers
  00  00  153  04                      !    Col 2: Source (node) of input
  00  00  155  09                      !    Col 3: Output stream numbers
03  03  08  33  2  2                   !    Col 4: Destination (node) of
  19  02  022  18                      !              output stream
  20  04  076  12                      !
04  04  02  33  2  8                   !
  152  01  007  10                     !  Node 1: FCE
  153  02  020  03                     !  Node 2: Collection MLOCRs
  000  00  025  18                     !  Node 3: Outgoing BCS Secondary
  000  00  065  05                     !  Node 4: Collection RVES
  000  00  066  05                     !
  000  00  067  06                     !
  000  00  069  06                     !
  000  00  073  12                     !
05  03  08  33  4  5                   !  Node 5: Turnaround BCSs
  65  02  051  19                      !
  65  04  070  06                      !
  66  02  074  12                      !
  66  04  158  14                      !
  00  00  178  15                      !
06  03  08  44  9  4                   !  Node 6: Firms/ADC Secondary BCSs
  67  02  030  17                      !
  67  04  033  20                      !
  67  08  071  12                      !
  67  09  072  12                      !
  69  02  000  00                      !
  69  04  000  00                      !
  69  08  000  00                      !
  69  09  000  00                      !
  70  05  000  00                      !
07  06  01  33  2  7                   !  Node 7: Manual Bundle Sort
```

```
   12  00  028  17          ¦
   13  00  031  20          ¦
   00  00  049  19          ¦
   00  00  059  08          ¦
   00  00  060  08          ¦
   00  00  061  08          ¦
   00  00  062  08          ¦
08  02  01  33  11  8       ¦  Node 8: Incoming MLOCRs
   59  07  051  19          ¦
   60  07  067  06          ¦
   61  07  069  06          ¦
   62  07  075  12          ¦
   53  00  156  09          ¦
   54  00  157  09          ¦
   55  00  159  14          ¦
   56  00  179  15          ¦
   57  00  000  00          ¦
   58  00  000  00          ¦
   63  00  000  00          ¦
09  04  01  33  4  6        ¦  Node 9: Incoming RVES
   155  02  051  19         ¦
   156  08  067  06         ¦
   157  08  069  06         ¦
   154  00  075  12         ¦
   000  00  159  14         ¦
   000  00  179  15         ¦
10  06  01  33  6  5        ¦  Node 10. Manual Primary
   03  01  009  11          ¦
   07  04  027  18          ¦
   02  00  028  17          ¦
   04  00  031  20          ¦
   05  00  049  19          ¦
   06  00  000  00          ¦
11  06  01  44  3  3        ¦  Node 11. Manual Secondary
   09  10  034  21          ¦
   10  00  037  22          ¦
   11  00  044  23          ¦
12  05  16  44  9  5        ¦  Node 12. Primary MPLSMs
   71  06  026  18          ¦
   72  06  029  17          ¦
   73  02  032  20          ¦
   73  04  050  19          ¦
   74  05  077  13          ¦
   75  08  000  00          ¦
   75  09  000  00          ¦
   76  03  000  00          ¦
   99  00  000  00          ¦
13  05  16  47  3  3        ¦  Node 13. Secondary MPLSMs
   77  12  35  21           ¦
   78  14  38  22           ¦
   79  15  45  23           ¦
14  03  08  47  3  3        ¦  Node 14. Firms Route Sort BCSs
   158  05  38  21          ¦
   159  08  40  22          ¦
   159  09  78  13          ¦
15  00  31  46  3  2        ¦  Node 15. Sequencing Block
```

```
178  05  079  13               |
179  08  258  16               |
179  09  000  00               |

.070  .540  .390                                               ||
                                                               || Block 3:
.175  .365  .120  .027  .007  .005  .038  .232  .031           ||
.154  .321  .105  .024  .006  .005  .034  .325  .026           || Transfer matrix
.154  .321  .118  .024  .006  .005  .034  .325  .013           || data, one node
                                                               || at a time.
.980  .020                                                     || Elements are
.980  .020                                                     || the branching
                                                               || proportions
.019  .223  .466  .191  .035  .009  .007  .050                 || for input stream
.019  .223  .466  .191  .035  .009  .007  .050                 || (row) to output
                                                               || stream (column)
.000  .035  .020  .104  .841                                   ||
.000  .035  .020  .104  .841                                   || Rows must sum
.980  .000  .020  .000  .000                                   || to one.
.980  .000  .020  .000  .000

.980  .000  .020  .000
.980  .000  .020  .000
.980  .000  .020  .000
.980  .000  .020  .000
.000  .980  .000  .020
.000  .980  .000  .020
.000  .980  .000  .020
.000  .980  .000  .020
.990  .000  .010  .000

.015  .030  .148  .743  .000  .064  .000
.015  .030  .148  .000  .743  .000  .064

.000  .000  .000  .039  .211  .150  .066  .534
.000  .000  .000  .039  .242  .144  .063  .512
.000  .000  .000  .039  .211  .075  .074  .601
.000  .000  .000  .038  .242  .072  .071  .577
.145  .015  .029  .020  .000  .158  .069  .564
.145  .015  .029  .020  .000  .079  .078  .634
.111  .011  .022  .039  .211  .121  .053  .432
.106  .011  .021  .038  .242  .116  .051  .415
.111  .011  .022  .039  .211  .061  .060  .485
.106  .011  .021  .038  .242  .058  .057  .467
.145  .015  .029  .020  .000  .000  .087  .704

.000  .000  .000  .024  .107  .869
.140  .014  .028  .050  .084  .684
.000  .000  .000  .000  .110  .890
.145  .015  .029  .020  .087  .704

.205  .740  .010  .008  .037
.205  .740  .010  .008  .037
.205  .740  .010  .008  .037
.808  .000  .015  .030  .147
```

```
.808   .000   .015   .030   .147
.808   .000   .015   .030   .147

.104   .006   .890
.104   .006   .890
.104   .006   .890

.000   1.00   .000   .000   .000
.000   .000   1.00   .000   .000
.740   .010   .008   .038   .204
.740   .010   .008   .038   .204
.000   .000   .000   .155   .845
.000   .015   .030   .148   .807
.000   .015   .030   .148   .807
1.00   .000   .000   .000   .000
.000   .015   .030   .148   .807

.104   .006   .890
.944   .056   .000
.000   .000   1.00

.935   .055   .010
.926   .054   .020
.926   .054   .020


10.50   .070   .540   .390                                                 | Block 5:
                                                                            |
13.75   .175   .365   .120   .027   .007   .005   .038   .232   .031       | Transfer
13.75   .154   .321   .105   .024   .006   .005   .034   .325   .026       | matrices
13.75   .154   .321   .118   .024   .006   .005   .034   .325   .013       | repeated
                                                                            | with added
14.00   .980   .020                                                         | processing
14.00   .980   .020                                                         | rates in
                                                                            | column 1
00.00   .019   .223   .466   .191   .035   .009   .007   .050              | (if input
00.00   .019   .223   .466   .191   .035   .009   .007   .050              | stream
27.44   14.78  14.78  6.72   999.99  16.80  16.80  6.72                     | dependent)
                                                                            |
14.00   .000   .035   .020   .104   .841                                     | If output
14.00   .000   .035   .020   .104   .841                                     | dependent
14.00   .980   .000   .020   .000   .000                                     | processing
14.00   .980   .000   .020   .000   .000                                     | rate, first
                                                                            | column will
14.00   .980   .000   .020   .000                                            | be zero and
14.00   .980   .000   .020   .000                                            | processing
14.00   .980   .000   .020   .000                                            | rates are
14.00   .980   .000   .020   .000                                            | added as a
14.00   .000   .980   .000   .020                                            | last row.
14.00   .000   .980   .000   .020                                            |
14.00   .000   .980   .000   .020
14.00   .000   .980   .000   .020
14.00   .990   .000   .010   .000

06.25   .015   .030   .148   .743   .000   .064   .000
06.25   .015   .030   .148   .000   .743   .000   .064
```

```
13.75  .000  .000  .000  .039  .211  .150  .066  .534
13.75  .000  .000  .000  .039  .242  .144  .063  .512
13.75  .000  .000  .000  .039  .211  .075  .074  .601
13.75  .000  .000  .000  .038  .242  .072  .071  .577
13.75  .145  .015  .029  .020  .000  .158  .069  .564
13.75  .145  .015  .029  .020  .000  .079  .078  .634
13.75  .111  .011  .022  .039  .211  .121  .053  .437
13.75  .106  .011  .021  .038  .242  .116  .051  .415
13.75  .111  .011  .022  .039  .211  .061  .060  .485
13.75  .106  .011  .021  .038  .242  .058  .057  .467
13.75  .145  .015  .029  .020  .000  .000  .087  .704

00.00  .000  .000  .000  .024  .107  .869
00.00  .140  .014  .028  .050  .084  .684
00.00  .000  .000  .000  .000  .110  .890
00.00  .145  .015  .029  .020  .087  .704
16.80  16.80  16.80  6.72  16.80  6.72

.4473  .205  .740  .010  .008  .037
.4473  .205  .740  .010  .008  .037
.4473  .205  .740  .010  .008  .037
.4473  .808  .000  .015  .030  .147
.4473  .808  .000  .015  .030  .147
.4473  .808  .000  .015  .030  .147

.4473  .104  .006  .890
.4473  .104  .006  .890
.4473  .104  .006  .890

16.35  .000  1.00  .000  .000  .000
16.35  .000  .000  1.00  .000  .000
14.04  .740  .010  .008  .038  .204
14.04  .740  .010  .008  .038  .204
16.35  .000  .000  .000  .155  .845
16.35  .000  .015  .030  .148  .807
16.35  .000  .015  .030  .148  .807
14.04  1.00  .000  .000  .000  .000
16.32  .000  .015  .030  .148  .807

13.90  .104  .006  .890
13.90  .944  .056  .000
13.90  .000  .000  1.00

14.00  .935  .055  .010
14.00  .926  .054  .020
14.00  .926  .054  .020
```

REFERENCES

J.F. Bard and T.A. Feo, "An Algorithm for the Manufacturing Equipment Selection Problem," to appear in IIE Transactions, 1990.

Congressional Hearing before the Subcommittee on Postal Operations and Services, on H.R. 1980, Prohibit the Use of Clusterboxes for Certain Residential Mail Delivery, 99th Congress, First Session, July 11, 1985, SN 99-24, U.S. Government Printing Office, 1985.

Congressional Hearing before the Subcommittee on Postal Operations and Services, Postal Service Automation Program, 99th Congress, First and Second Sessions, November 7, 1985, and April 29, 1986, SN 99-57, U.S. Government Printing Office, 1986.

Congressional Joint Hearings before the Subcommittee on Postal Operations and Services and the Subcommittee on Postal Personnel and Modernization, Oversight on the Reorganization of the United States Postal System, 100th Congress, First Session, May 6, June 25, and July 15, 1987, SN 100-9, U.S. Government Printing Office, 1987.

A. deSilva, G. deLeo, and N. Gennari, European Journal of Operations Research, Special Issue on the Management of Technology, Elsevier Science Publishers, Amsterdam, Netherlands, to appear 1990.

ELSAG, Inc., Carrier In Office Project, Concept Development and Evaluation Final Report to the United States Postal Service, 1986.

I.S. Fan and P.J. Sackett, "A PROLOG Simulator for Interactive Flexible Manufacturing Systems Control," Simulation, Vol. 51, No. 6, pp 239-247, June 1988.

P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright, Two Step-Length Algorithms for Numerical Optimization, Report SOL 79-25, Department of Operations Research, Stanford University, Stanford, California, 1979.

F.S. Hillier and G.J. Liebermann, Introduction to Operations Research, 4th Ed., Holden-Day Inc., Oakland, California, pp 43-46, 1986.

A.S. Kiran, A. Schloffer, and D. Hawkins, "An Integrated Simulation Approach to the Design of Flexible Manufacturing Systems," Simulation, Vol. 52, No. 2, pp 47-52, February 1989.

A. Kusiack, "The Production Equipment Requirements Problem," International Journal of Production Research, Vol. 22, No. 6, pp 949-968, 1984.

R.E. Marsten, "The Design of the XMP Linear Programming Library," Transactions on Mathematical Software, Vol. 7, No. 4, December 1981.

G.L. Nemhauser and L.A. Woolsey, Integer and Combinatorial Optimization, John Wiley and Sons, New York, 1988.

D.G. Newnan, Engineering Economic Analysis, 3rd Ed., Engineering Press Inc., California, 1988.

A.A. Pritsker, Introduction to Simulation and SLAMII, 3rd Ed., Systems Publishing Company, 1986.

P.S. Pulat and B.M. Pulat, "Throughput Analysis in an Automated Material Handling System," Simulation, Vol. 52, No. 5, pp 195-198, May 1989.

J.K. Reid, "A Sparsity-exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Bases," Mathematical Programming, No. 24, pp 55-69, 1982.

S.C. Sarin and C.S. Chen, "A Model for Manufacturing System Selection," Flexible Manufacturing Systems: Methods and Studies, North-Holland, Amsterdam, pp 99-112, 1986.

R. Sherman (ed.), Perspectives on Postal Service Issues, American Enterprise Institute for Public Policy Research, 1980.

J. Singhal, R.E. Marsten, and T.L. Morin, "Fixed Order Branch-and-Bound Methods for Mixed Integer Programming: The ZOOM System," ORSA Journal on Computing, Vol. 1, No. 1, pp 44-51, 1989.

H.A. Taha and R.R. Goforth, "Unbiased Statistics and Simulation of Continuous Mail Processing," Simulation, Vol. 50, No. 6, pp 267-287, June 1988.

J.T. Tierney, Postal Reorganization -- Managing the Public's Business, Auburn House Publishing, Massachusetts, 1981.

United States Postal Service, Automation and Retail Equipment, USPS Publication 150, May 1988.

United States Postal Service, Postal Facts for Fiscal Year Ended September 30, 1988, 1989 Ed., Published by the Communications Department, U.S. Postal Service, 1989.

VITA

Steven Douglas Wert, [redacted] He was graduated from Auburndale Senior High School in Auburndale, Florida, in 1977 and entered the University of Florida at Gainesville where his major field of study was mechanical engineering. He was awarded a Bachelor of Science degree in May, 1982, and was commissioned by the United States Air Force. He served first as a research officer, then as Chief, Atmospheric Phenomenology Section, at the Air Force Weapons Laboratory (AFWL), Kirtland Air Force Base, New Mexico from 1982 until 1987. During that period, his work in the study of high explosive detonation physics, far field airblast from buried explosive simulators, and convention weapons test methodology was presented at international symposia and published in numerous technical reports. In 1987 he was selected Executive Officer of the AFWL Nuclear Technology Office where he coordinated the research programs of six diverse divisions. He was selected in 1988 for the Air Force Institute of Technology Masters Degree Program and entered the Graduate School of the University of Texas at Austin in September 1988.

[redacted]

[redacted]

This thesis was typed by Steven Douglas Wert and formatted by Papers-To-Go.