

VARIABLE SELECTION RULES FOR IMPLICIT ENUMERATION

A THESIS

Presented to

The Faculty of the Division of Graduate
Studies and Research

By

Charles Lemuel Carroll III

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

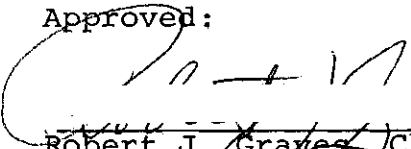
In the School of Industrial and Systems Engineering


Georgia Institute of Technology

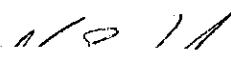
March, 1977

VARIABLE SELECTION RULES FOR IMPLICIT ENUMERATION

Approved:


Robert J. Graves, Chairman


Ronald L. Rardin


V. E. Unger

Date approved by Chairman 2/22/77

ACKNOWLEDGEMENTS

I have received encouragement from many in the course of my graduate studies at Georgia Institute of Technology. In particular, I wish to acknowledge the academic and financial support provided by Dr. R. N. Lehrer and Dr. W. W. Hines.

I also wish to acknowledge Dr. R. J. Graves. His insight into the problem kindled the interest in the area which eventually led to this work. Dr. R. L. Rardin and Dr. V. E. Unger as thesis advisors also provided technical assistance without which this research would be sorely lacking.

Also, I wish to thank all of my friends who at one time or another encouraged me to complete this work. A very special thanks to my lifetime good friend, David Flynn, is also noted. Lastly, I am thankful to Him whose name is above every other name.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	ii
LIST OF TABLES	v
LIST OF ILLUSTRATIONS	vi
Chapter	
I. RESEARCH GOALS	1
1.1 Introduction	1
1.2 The Purpose of the Research	1
1.3 Scope and Limitations	2
1.4 The Thesis Plan	2
II. A REVIEW OF THE PROBLEM AND OF SIGNIFICANT LITERATURE	4
2.1 The Office Building Problem - An Example Capacitated Quadratic Location Problem	4
2.2 The Model	6
2.3 A Capacitated Quadratic Assignment Problem Solution Procedure	9
2.4 A Review of the Importance of Free Variable Entering Rules	22
2.5 Some Comments on the Analysis Procedure	26
III. THE ANALYSIS PROCEDURE	38
3.1 Introduction	38
3.2 The Experimental Treatments	39
3.3 Some Preliminary Statistical Treatments	49
3.4 The Random Problem Generator	50
3.5 The Experimental Design	59
IV. ANALYSIS OF THE EXPERIMENTAL RESULTS	71
4.1 An Examination of Assumptions	71
4.2 An Analysis of Variance	76

TABLE OF CONTENTS (Continued)

Chapter	Page
IV. ANALYSIS OF THE EXPERIMENTAL RESULTS (Continued)	
4.3 Final Analysis of Variance	80
4.4 Examination of Main Effects and Interactions	81
V. CONCLUSIONS AND RECOMMENDATIONS	86
5.1 Summary of Results	86
5.2 Recommendations for Further Research .	87
APPENDICES	
A. The Solution Procedure Code	89
B. The Random Problem Generator Code	121
C. The Experimental Results	143
BIBLIOGRAPHY	160

LIST OF TABLES

Table	Page
3.1 Expected Mean Squares	69
4.1 An ANOVA of the Transformed Observations . . .	78
4.2 An ANOVA with the N Effect Pooled	79
4.3 The Statistically Significant Effects	82
4.4 Estimates of the Main Effects	82

LIST OF ILLUSTRATIONS

Figure		Page
3.1	The Experimental Design	62
4.1	Plot of Residual Cell Variance vs. Cell Mean	72
4.2	Plot of Residual Standard Deviation vs. Cell Mean	73
4.3	Plot of the Residual Variances of the Transformed Observation	75
4.4	Histogram of Error Residuals of Transformed Observation	77
4.5	Graphical Representation of Some Two Way Interactions	84

CHAPTER I

RESEARCH GOALS

1.1 Introduction

In recent years interest in integer programming, or problems in which some of the decision variables are required to be integer, has been increasing in all disciplines involved in decision making. One class of integer programming problems known as capacitated quadratic assignment problems deals with the allocation of interrelated variables among regions. The optimal assignment is that arrangement which minimizes the total interaction between the given regions.

Graves [13] formulated the problem of site selection for college departments in a multi-campus university as a capacitated quadratic assignment problem. His work went on to develop an implicit enumeration solution procedure and to examine one particular university's location decisions within a planning context. Graves [13] besides demonstrating the wide area of applicability emphasized that present solution procedures drastically limit the size of problem which can practically be solved.

1.2 The Purpose of the Research

The work done in this research centers around investigating possible improvements in a least bound implicit enumeration solution procedure. In particular the performance

of two free variable entering strategies are compared for various levels of factors which affect the general structure of the problem. Consideration will also be given to the question of whether or not certain factors examined significantly affect problem solving difficulty.

1.3 Scope and Limitations

The scope of this research is to examine two free variable entering strategies in light of a class of capacitated quadratic assignment problem referred to as the office building problem. The office building problem differs from a more general capacitated quadratic assignment problem in that it has a given number of variables that have a fixed value. All results are to be viewed in light of the following limitations:

- (1) The two free variables entering strategies are only compared for one solution algorithm.
- (2) Only problems with a comparatively large number of fixed variables, i.e., the office building problem, are considered.

1.4 The Thesis Plan

The plan of the thesis can be seen by the following brief discussion of the contents of each chapter.

- (1) Chapter 1: An Introduction. This chapter is concerned with the goals and scope of the research effort.
- (2) Chapter 2: A Discussion of the Problem and of Related Literature. This chapter discusses the problem, the solution procedure, and the method of analysis in detail.
- (3) Chapter 3: Development of the Experimental Design. This chapter deals with a discussion of the experi-

mental treatments, the analysis of variance assumptions, the random problem generator, and the development of a good experimental design.

- (4) Chapter 4: The Analysis. This chapter is an analysis of the data collected from an experiment run according to the design developed in Chapter 3.
- (5) Chapter 5: Conclusions and Recommendations. This chapter concludes the paper. It contains the conclusions and findings of the research and offers some additional areas in which further work could be done along these same lines.

CHAPTER II

A REVIEW OF THE PROBLEM AND OF SIGNIFICANT LITERATURE

2.1 The Office Building Problem - An Example
Capacitated Quadratic Location Problem

Currently there is very little literature available on the capacitated quadratic assignment problem. Graves [13] discusses this problem and has developed an exact solution procedure. Lin [18] has established a framework for statistical analysis of integer type problems. Both of these contributions will be examined in detail and placed in context as the problem examined in this work is developed.

An example of the problem of interest is how to locate or assign interrelated items to regions where each item has a size, S_i , and where each region to which items can be assigned has a size constraint. The best assignment is that which minimizes the total amount of interaction between regions. One real world application of this problem would be locating a production corporation's supervisory, support and service groups' offices in a multi-floor production center. Each group's office could be viewed as related to or interacting with other offices or work areas. Another example of the problem might be to find the best locations within a large corporation's multi-floor office

building for the offices of a newly-organized section. The measure of interaction or flow between various offices in these problems could be formulated as the number of trips between office i and office j during some given time period. In these types of problems, referred to from now on as the office building problem, all flow would be considered non-directional and each movement from one office to another would represent one unit of flow. For example, if someone moved from office i to office j , or vice versa, 32 times in the given time period, the flow between i and j and j and i would equal 32.

The space available on each floor could be viewed as representing the size constraint of each region. Likewise every office would have some size associated with it. The general problem, then, stated in terms of the office building problem would be to locate the N offices on the M floors such that the size constraints are satisfied. By including the cost of movements between the different floors, the model can be reformulated in terms of cost. Having done this, locations can be determined which minimize the total cost of movement between floors for given size constraints.

The following assumptions are made: (1) the offices are non-divisible, that is, each office must be located in its entirety on one and only one floor; and (2) the flow between each office is deterministic and known. The size

of each floor and each office must also be known. (Note that no restrictions have been put on the shape of the floors or offices.) (3) The cost associated with a unit movement from one floor to another is the same over time.

2.2 The Model

The following information is necessary for the formulation of a capacitated quadratic model as representative of the problem of locating interacting offices on a multi-floor office building.

- I. A description of the offices:
 - i) the number of offices
 - ii) how many offices are free to be assigned by the model, and how many are fixed
 - iii) whether a given office's location is fixed or free
 - iv) the size of each office
 - v) the amount of interaction between each office
- II. A description of the floors:
 - i) the number of floors
 - ii) the size of the floors
- III. A description of the cost:
 - i) the cost of movement between the various floors

Mathematically, the model appears as follows:

minimize $Z =$

$$\sum_{k=1}^M \sum_{r=1}^M \sum_{i=1}^M \sum_{j=1}^M C_{kr} F_{ij} \left[[Y_{jk}(1-Y_{ik})] [Y_{ir}(1-Y_{jr})] \right] \quad (2.1)$$

subject to the following three constraints:

$$\sum_{i=1}^N S_i Y_{ik} \leq A_k \quad \text{for all } k = 1, 2, \dots, M \quad (2.2)$$

$$\sum_{k=1}^M Y_{jk} = 1 \quad \text{for each } j = 1, 2, \dots, N \quad (2.3)$$

$$Y_{jk} = 0, 1 \quad \text{for all } j, k \quad (2.4)$$

where the terms are defined as follows:

- (1) Z = the value of the objective function
- (2) M = the number of floors
- (3) N = the number of offices
- (4) k, r = floor indices; $k, r = 1, 2, \dots, M$
- (5) i, j = office indices; $i, j = 1, 2, \dots, N$
- (6) $Y_{jk} = 1$, if office j is located on floor k ;
0, otherwise.
- (7) F_{ij} = the flow between office i and office j ; in this case, the number of times someone moves per day from office i to office j or vice versa.
- (8) A_k = the size of floor k . This represents a constraining size and limits the number of offices that can be located on any one floor.
- (9) S_i = the size of office i . This represents the space required by office i .
- (10) C_{kr} = the cost associated with traveling from floor k to floor r .

The model's objective is to minimize the total cost of movement between floors in the context of the N offices, M floors office building location problem. The component

of the objective function, $[Y_{ir}(1-Y_{jr})]$ will have the value of one when office i is on floor r and office j is not on floor r . Likewise, $[Y_{jk}(1-Y_{ik})]$ will have a value of one when office j is on floor k and when office i is not on floor k . The product of these components will produce a value of one only if office i is located on floor r and office j is located on floor k , otherwise the product's value is zero. The product of these decision variables enables the model to consider and assign an appropriate cost for every unit of between-floor movement. By summing over all floor and office indices contained within the objective function, the total value of Z or the total weighted cost of movement between floors is obtained. It is precisely this quantity for which a minimum value is desired.

The first constraint set is of the following form:

$$\sum_{i=1}^N S_i Y_{ik} \leq A_k \quad \text{for all } k = 1, 2, \dots, M$$

This constraint, in essence, sums the size of each office located on each of the M floors. This constraint requires that no more office space will be assigned to a floor than that floor can hold. The constant A_k in this case represents the initial area available for office assignment on floor k . Note that this A_k may or may not be the same for different floors.

The second constraint set is of the following form:

$$\sum_{k=1}^M Y_{jk} = 1 \quad \text{for each } j = 1, 2, \dots, N$$

This second constraint requires that every single office be located somewhere among the M floors. This constraint assures, for each office, exactly one location in the final feasible solution.

The last constraint requires that the decision variable have only values of zero or one. In practical terms, this insures that a single office will not be split among several floors.

2.3 A Capacitated Quadratic Assignment Problem Solution Procedure

The thrust of this paper is to examine the effect free variable entering rules have on the time required to obtain an optimal solution using an implicit enumeration scheme. A general review of this procedure as used by Graves [13] follows.

Implicit Enumeration

Implicit enumeration algorithms operate by implicitly searching all possible solutions through examining certain well-chosen solutions and using the information obtained from the examination of these solutions, coupled with dominance concepts and constraint infeasibility arguments to eliminate non-optimal solutions from investigation in the search for the optimal. The investigation of these

solutions serves also to establish stricter optimal solution criteria. Although from a practical point of view it can be proven that implicit enumeration searching techniques will converge on the optimal solution, it has been demonstrated that the computer resources required can be very large and usually are unpredictable. However, implicit enumeration, due to its emphasis on the logic of the problem, is recognized by many to be a most promising integer programming solution technique [13].

A brief description of the general procedure used in implicit enumeration and branch and bound techniques is given by Zionts [27].

1. Given a list of active partial solutions that have not yet been explored, select one to examine next. If appropriate, terminate the search and conclude that the optimal solution is the best one found so far, or that no optimal feasible solution exists. Otherwise, go to step 2.
2. Examine the partial solution selected, and draw some conclusions about its possible continuations that may exclude certain successor partial solutions from consideration. This includes adding the successor partial solution, as appropriate, to the list. Then go to step 1.

As noted from Zionts' description, there are two fundamental questions which need to be recognized and answered. First, how does one select which sub-problem, or partial solution (as they are appropriately called in this example), to examine next. Secondly, how does one draw inference about the desirability of continuing or discarding certain partial solutions.

In answering the first question, it is important to realize that the "order" is a function of two things: (1) the structure of the searching tree, and (2) the rules by which the free variables are entered into the searching tree. A brief discussion of each of these is in order. The term "structure" is used to denote the actual mechanical way in which the decision variables are enumerated in the solution process. For instance, one, two, or more free variables could be assigned in each stage or step of the solution procedure. Obviously, the possible number of structures is very large; therefore, it is important that a structure be chosen which is advantageous to the actual solution procedure used. A simple structure which is relatively straightforward to develop in an algorithmic way is quite desirable, but perhaps more desirable in a structure is the attribute of hierarchical dominance. A structure is hierarchically dominant if every partial solution which branches or flows from partial solution A is dominated by partial solution A. The ideas of dominance and the structure of the partial solutions search tree will be further discussed later. The second factor affecting the ordering of the partial solutions is how one decides in what order the free variables are to be entered into the searching process. Practical experience shows that the order in which the free variables are placed into partial solutions can significantly affect how rapidly an implicit enumera-

tion technique converges to the optimal solution [21]. However, it is extremely difficult to know the exact order in which the variables should be entered into the searching tree. The importance of free variable entering rules is discussed more fully in Section 2.4.

The second basic question deals with the problem of obtaining information as to whether or not continuation of a partial solution could result in an optimal solution. The approach used in dealing with this problem is that of a simple branch and bound scheme. In brief, the approach operates on the premise that the optimum value of the objective function Z^* lies between some upper bound (UB) and some lower bound (LB).

For this minimization problem, the value of UB, the upper bound, is always set equal to the smallest feasible objective function value of all the known feasible solutions. The lower bound, LB, can be calculated by summing the contribution to the objective function of those variables already assigned locations and the absolute minimum contribution to the objective function of all those variables which are not presently assigned. Note that the second portion of the lower bound is not required to be constraint feasible. For this reason the lower bound value obtained for each partial solution will not always equal the optimal location objective function value; however, the optimal value of the objective function Z^* is guaranteed to

be no lower than the lower bound value.

Knowing that the optimal solution to a partial solution can be no lower than the lower bound value for that partial solution and remembering that the optimal solution to the whole problem has to be less than or equal to the upper bound value allows for the following test and inference to be drawn. Upon examination of each partial solution, a lower bound LB is calculated and that LB is compared with the current upper bound UB of the problem. Whenever the lower bound LB is greater than the upper bound UB, it is certain that continuation of that partial solution would not result in an optimal solution. A partial solution, then, that has a lower bound LB greater than the upper bound UB can be excluded from further consideration. A second test that should be performed is to determine if those active partial solutions satisfy all of the size constraints, namely constraint set 1.

$$\sum_{i=1}^N S_i Y_{ir} \leq A_r \quad \text{for all } r = 1, 2, \dots, M$$

If any partial solution does not satisfy this constraint set, it and all the nodes emanating from it can be eliminated from further investigation. These two tests are to be performed at each partial solution and in combination represent the total inference and testing mechanism.

The Decision Tree

As noted from Zionts' description, implicit enumeration is a sequential process. In this problem the combinatorial tree which is sequentially examined is developed by expanding outward from an origin into a series of subsequent generations. The origin or beginning of the process includes only those variables that have an original value of one. These variables, referred to as fixed variables, are those which must be at a certain location. Each subsequent generation or stage from its predecessor represents a set of mutually exclusive and collectively exhaustive descendants of the previous generation; and each partial solution in generation, stage T , differs from its immediate predecessor, stage $T-1$, by having one previously unassigned free variable now fixed at some location.

For example, in the multi-story office building problem, 15 out of 30 offices might have a fixed location due to a policy determination. Consider that there are three floors available on which to locate new offices and five offices must be on the first floor, three on the second floor, and seven on the third floor. A total of 15 offices are left to be located. These variables which have no predetermined locations are called free variables.

The implicit enumeration procedure starts from an origin with the 15 variables which have a fixed location assigned to them. It then makes three branches from that ori-

gin. These three branches represent the assignment of the first free variable to be considered to either the first, second, or third floor. This branching process continues, with the decision being always to branch from the node or partial solution with the smallest lower bound value. Each stage or generation of branching represents the addition of one more free variable to the partial solution. This branching scheme continues until all of the free variables have been included for consideration. If no inferences resulting in elimination of some partial solution are drawn, this branching scheme will totally enumerate the solutions, and a combinatorial tree would be developed which would contain every possible location of the free variables. In the example regarding the 15 free offices and the three floors on which to locate them, there is a maximum of partial solutions to examine if all partial solutions are totally enumerated. Equation 2.5 relates the total number of partial solutions in a totally enumerated search tree.

$$N_p = \sum_{k=1}^n M^k \quad (2.5)$$

where N_p = the maximum number of partial solutions
 n = the number of free variables
 M = the number of locations

The termination of the branching process does assure, however, that constraint set 2 or

$$\sum_{k=1}^M Y_{ir} = 1 \quad \text{for all } i = 1, 2, \dots, N$$

will be satisfied by not allowing offices to be assigned to more than one floor in any one partial solution.

Dominance

Looking at Equation 2.1 and remembering that all of the cost C_{rk} and all of the flows F_{ij} are non-negative, it is evident that Z^* or the optimal value of any partial solution can not decrease as more variables are included in the problem. $Z =$

$$\sum_{k=1}^M \sum_{r=1}^M \sum_{i=1}^M \sum_{j=1}^M C_{kr} F_{ij} \left[[Y_{jk}(1-Y_{ik})] [Y_{ir}(1-Y_{jr})] \right] \quad (2.1)$$

$$\sum_{i=1}^N S_i Y_{ik} \leq A_k \quad \text{for all } k = 1, 2, \dots, M \quad (2.2)$$

If each partial solution $T+1$ is viewed as differing from its parent partial solution T only by the addition of one free variable, it is clear that the value of partial solution $T+1$ is always greater than or equal to the value of partial solution T . For this reason, if it can be determined that the best solution to partial solution T is non-optimal, then investigation of any partial solutions branching from partial solution T would be fruitless.

Equation 2.2 simply requires that the total amount of space required by the offices assigned to each floor be

no greater than the space each floor has available. Remembering once again that each partial solution differs from the predecessor partial solution by the assignment of one additional free variable, and that the area each floor has available A_k is constant, and that S_j , the size required by each office, is non-negative, allows for the following statement concerning feasibility: If any partial solution T is found to be infeasible because of size or space constraints, then all partial solutions branching from partial solution T are also space or size infeasible. This is due to the non-decreasing aspect of the size function, i.e., Equation 2.2.

Inference Techniques

In view of the dominance characteristics of the search tree, the importance of the relationship between the order or structure of the tree of partial solutions and the ability to draw inferences about each partial solution's feasibility and optimality is more clearly understood. It is easy to understand why strong inference techniques and hierarchical dominance in the tree structure are so desirable.

Of course, even with a well-structured problem, unless the inference techniques are strong, a relatively small problem may become unsolvable due to the large number of partial solutions to examine. For instance, for the problem discussed previously with three floors on which to locate offices, given that there are 17 free variables there would

be a total of 193,710,244 partial solutions to examine in the absence of inference techniques. The large number of partial solutions that has to be explored also points toward the need for bound calculations that require very little calculation effort and time. There existed two key criteria to keep in mind concerning bounds and their development. The first was speed or simplicity of calculation. The second criterion was strength or the ability to distinguish partial solutions whose continuation would result in sub-optimal solutions from those whose continuation would result in an optimal or near optimal solution.

As is often the case, these two criteria were found to be not directly complementary in that there exists a trade-off between speed and strength. This trade-off can be seen by realizing that basically the time required to find the optimal solution is a function of the time required to explore and calculate bounds for partial solutions investigated. So the time required to find an optimal solution might be reduced either by reducing the time required to calculate each bound or by reducing the number of partial solutions examined.

Having reviewed the purpose behind the structure and inference techniques and the relationship between them, along with the importance of strong bounding techniques, there remains to be discussed the actual details of the

bound calculations. Associated with each partial solution are the upper and lower bounds. Each bound's significance lies in the fact that if the lower bound LB is greater than the upper bound UB for any partial solution, then that partial solution and all the partial solutions which branch from it can be disregarded in the search for the optimal solution. The test is structured in such a way that a good estimate or "strong bound" is needed for both the lower and upper bound. The bounds used in Graves [13] will now be briefly discussed.

The Upper Bound

Generally, an upper bound can initially be set at infinity, or a solution can be discovered randomly, or a good solution can be obtained from a heuristic method. This heuristic method consists of two phases. The first is a constructive process in which the offices are located in a logical way. The second phase is an improvement process where simple changes are investigated to determine if the constructed solution can be improved.

The Construction Phase

This phase begins by separating the problem's free and fixed variables. The free variables are assigned, one at each stage, to the location or floor that obtains the smallest increase in the objective function value. This initial phase in which all of the free variables are assigned does not, however, return to re-examine the placement deci-

sion of a free variable at a later stage when another free variable is under scrutiny. This process of re-examination occurs in the interchange or improvement portion of the heuristic once the first portion has exhausted the free variables list and a feasible solution is at hand.

The interchange phase attempts in a very simple way to find any improvements of the feasible solution constructed in the first phase.

The Interchange Process

The first portion of the interchange phase of the heuristic examines the possible exchange of single free variables located at some location by the construction phase of the heuristic with slack capacity at another location. Similarly, the second portion of the interchange phase examines the possible exchange of a pair of located free variables plus slack, one at each of two locations, with each other. In both parts of this improvement phase an exchange is performed only if both a feasible solution results and the value of the objective function is improved. The improvement process concludes when some number of iterations are completed.

The Lower Bound

The lower bound, simply, is an estimate of the lowest value a partial solution's objective function could reach should that partial solution be carried forward to completion. Obviously, one could estimate a value of zero in each

instance and indeed zero would be a lower bound, but this type of estimation achieves little, for no real inference can be drawn about a partial solution. As discussed previously, strength is one of the most significant attributes a bound technique can have. Strength, in this case, implies the ability of a bounding procedure to provide as high a value as possible, and still guarantee that each value is a lower bound.

The lower bound used in this problem views the objective function estimate as being composed of two major components: the contribution of those variables already fixed in a given partial solution to the objective function; and the estimated contribution of the remaining free variables to the objective function given the continuation of a given partial solution. This first component is relatively easy to calculate because all of the variables which are considered have fixed and known values.

The second component represents an estimate of the cost associated with the free variables given their optimal placement. It is important to determine this component in such a way that its value represents a minimum or lower bound value of the objective function. In this second component, all free variables are considered in pairs. Each one of the pairs has an implied placement at a location as a result of viewing the individual contributions to the objective function, and choosing the location for each which

would minimize this objective function's growth. If these locations are the same for each individual of the pair, then only the individual objective function contribution is added to the estimate. However, if the implied locations for the individuals of the pair are different, then it may be possible to also add the activity of the flow between these individuals as well as the individual contributions to the estimate. This latter feature strengthens the estimating component, but will result only in a lower bound where locating pairs at the same location despite different implied locations when considered individually, will not result in a lower estimate of the objective function value than that of the implied locations. As each pair is then considered and the aggregate estimates are summed, the estimated component value is developed and added to the first component for a strengthened, estimated lower bound value.

2.4 A Review of the Importance of Free Variable Entering Rules

Free variable entering rules are those rules which govern the order in which free variables are considered in the implicit enumeration solution procedure. As mentioned earlier in this chapter, the order in which the free variables are assigned can drastically effect the time required to obtain the optimal solution.

When any partial solution can be found to be one which would result in a non-optimal partial solution if it

were continued to completion, it and all nodes branching from it can be excluded from further consideration. When this occurs the search tree is said to have been pruned at the given partial solution. Because a single prune eliminates more partial solutions from investigation the closer it takes place to the origin, it is desirable to have as many prunes as possible take place as quickly as possible in the decision tree. The problem arises, however, in that it is often difficult to know how to enter the free variables in order that the most effective pruning can take place. Whether or not a partial solution is pruned depends not only on the free variable entering into that partial solution but also on the problem itself and all of the previously assigned free variables. The problem of determining the most effective pruning strategy is compounded because there are two different reasons for pruning a partial solution from further consideration. The first reason is that the given partial solution's lower bound is greater than the current upper bound and would consequently result in a non-optimal solution. The second justification for pruning is that of a given partial solution not satisfying the constraints, particularly the size constraints. Such a partial solution and all partial solutions branching from it can be excluded from further consideration because they all are infeasible.

One designing an algorithm is left then with heuristic procedures at best, by which to estimate a good free variable entering scheme. However, as is the case with most heuristic procedures, these heuristic schemes do not always perform equally well on all problems. In general, a heuristic procedure is developed around some aspect or structure of the general problem that may not be exhibited in all examples. Perhaps this is the case with the two heuristics examined in this study.

The first ranking scheme orders the free variables by their constraint attribute. In the office building problem this would be by office size or space requirement. This first free variable ranking scheme examines each of the free offices' size and ranks those offices accordingly from largest to smallest. It was mentioned previously that there are two basic reasons for pruning a partial solution from further investigation, one being non-feasibility. It was also pointed out that the closer a prune is to the origin, the more effective it is or the more partial solutions it eliminates. With these points in mind, the logic of the first ranking scheme becomes apparent. The free offices are ranked by largest size to smallest so that partial solutions that result in infeasible location assignments occur quickly. This procedure is expected to perform well especially when the size constraints are very active, but the question should be addressed as to how well it will perform

if the size constraints are not really active. Another area of concern is its performance in view of the problem's changing attributes.

The Second Ranking Scheme

The second ranking scheme orders the free variables by their contribution to the lower bound calculation. In the office building the free offices are ranked from largest to smallest fixed flow, where the fixed flow of any free variable is the sum of that variable's flow to all variables which already have a fixed location at that stage in the tree. The fixed flow of a variable then is a rough measure of how much the addition of that variable will add to the first component of the lower bound calculation. This ranking scheme is attempting to quickly reach partial solutions where non-optimal location assignments have been made, i.e., the LB is greater than UB, the current upper bound.

Both ranking schemes are designed to work well on problems with certain structures and characteristics. Both are designed to improve the likelihood of early pruning for either non-feasibility or non-optimality. Though they are designed primarily for the occurrence of one type of prune, this does not exclude the other type of pruning from taking place during the solution process. In fact both types of pruning are very likely to be employed in most solution procedures. This only adds to the complication of knowing a priori which ranking scheme will result in the most effective pruning strategy.

The problem then becomes one of selection. If each of these free variables entering rules can be used, which one should be chosen so that the optimal solution can be obtained and proven to be optimal in a minimum amount of time? The use of each of these two ranking schemes in the same implicit enumeration solution procedure now allows the problem solver two algorithms from which to pick a solution procedure. The question then becomes, given certain conditions which generally characterize the problem, which of these two algorithms should be chosen.

There are many ways that algorithms have been compared in the past. One of the most recently developed methods involves statistical analysis. The remainder of this chapter will deal with comparison methods and, in particular, a statistical procedure developed by Lin [18]. A detailed review of Lin's work is in order because it is this approach which will be used in analyzing the two solution algorithms under varying problem conditions in the remainder of this work.

2.5 Some Comments on the Analysis Procedure

Within the past 15 years, numerous integer programming solution techniques have been developed and proposed. Generally, these solution procedures, when they are reported, are accompanied by some computational experience relating the performance of the proposed algorithm to some existing solution techniques. The way various algorithms'

performances are compared vary, but in general, some measure of performance is chosen and collected for a series of test problems. These measures of performance are then averaged over the number of test problems for each solution procedure. Finally, a comparison of means is used to indicate which algorithm on the average performed in a superior manner for those test problems. Often the measure of performance may differ among researchers; however, the computer time required or the number of basic algorithmic steps, such as the number of simplex iterations, are often used. Conclusions that are drawn from computational results may differ according to the measure of performance or the type of test problems used, so the choice of these can be crucial. Test problems fall into the following three groups: (1) either they are representations of real world problems, (2) they are developed specifically to exploit or test a certain algorithm's behavior, or (3) they are developed randomly.

The power and significance of any conclusions drawn from examination of the results depends upon the one of these three groups from which the test problems are drawn. In a strict sense, any conclusions drawn from tests where the test problems came, the real world or developed test problems, are valid only for the chosen test problems. The implication of this weakness is made evident by the inability of many researchers to report anything but vague find-

ings from such comparisons [26]. When the test problems are randomly generated, however, a more precise conclusion can be drawn. When test problems are generated from a random process, the measure of performance itself becomes a random variable. The results of the test problems then are realizations of a random variable and from these realizations strong statistical inferences can be made.

Although the statistical analysis is straightforward, until recently there had been very little work done in this area. In 1974, Rardin [22], and Rardin and Unger [23] made the first known attempts to draw such statistical inferences by applying an analysis of variance in a computational comparison of various fixed charge network algorithms. Some very strong conclusions were able to be drawn for some very common sets of problem distributions, but a rigorous analysis as to whether or not all of the necessary analysis of variance assumptions were satisfied was not performed.

Ben Lin's Intent

Lin [18] realized the potential power available to those interested in LP algorithm development and analysis work through the use of statistical analysis. The primary thrust of his work was devoted to examining whether or not analysis of variance can be applied and, if at all, in what way it can best be applied to the comparison of integer programming solution techniques. Issues investigated included whether or not all of the necessary assumptions for a

valid analysis of variance were satisfied and if, in fact, comparisons between different algorithms are desired, what kind of design an investigation should use to yield the most complete and desired information. Because Lin's work is so applicable to the analysis done in this research on the capacitated quadratic location problem, a brief review of his work will be given here.

A Review of Lin's Findings

Lin had primarily two objectives to his dissertation in 1975. First, he wanted to verify the validity of analysis of variance as a means of investigating different algorithms' performances and the effect certain problem parameters had on integer problem difficulty. Secondly, he wanted to develop a broad methodology which would facilitate such an analysis. He approached the problem by examining the basic statistical issues involved and working an example. From this work he was able to summarize the basic problem by addressing two questions:

- (1) Is an analysis of variance valid?
- (2) How can it best be applied?

Is an ANOVA Valid? Analysis of Variance (ANOVA) is one of the most commonly used tools in the analysis of data. The principle of the technique is that if observations can be classified according to one or more criteria, then the total variation between the members of the observations can be broken up into components which can be attributed

to the different criteria of the classifications. By testing the significance of these components, it is possible to determine which of the criteria are associated with a significant proportion of the overall variation.

The accuracy of the assumptions in the model underlying the procedure is one factor which can drastically affect the power of the ANOVA technique. Equation 2.6 is an example of the form of model assumed for an ANOVA.

$$X_{ij} = M + T_i + E_{ij} \quad (2.6)$$

where X = an observation
 M = an overall mean
 T_i = an effect due to class i
 E_{ij} = a residual representing the variation of observation X_{ij} from the average value of the i class

It can be seen from Equation 2.6 that the models assumed in an ANOVA are all linear in the parameters. Ellwein [11] in work done in 0-1 programming observed that solution times of some integer linear programs (ILP) solution procedures grow exponentially with the number of discrete variables. In light of this, Lin realized that an examination of ILP experimental data was necessary before an ANOVA could be performed to determine whether or not all of the assumptions of ANOVA were sufficiently satisfied. Besides being linear in all parameters, it is also assumed that the

residuals of the model are independent, normally distributed, and have constant variance. A brief examination of each of these three assumptions and their importance now follows.

The Assumption of Independence of Residuals. The ANOVA procedure assumes that the residuals (E_{ij}) are mutually independent random variables. When the test problems are chosen randomly, this should always result; failure of this assumption to hold, however, invalidates the procedure completely.

The Assumption of Normality of the Residuals. The ANOVA procedure also assumes that the residuals are normally distributed. It is well known, however, that the F-test used in the ANOVA procedure is very robust to the normality assumption. The result of using non-normal residuals is that the true significance of the hypothesis tests will not be exactly the one indicated. Several tests such as the Chi-square or the Kolmogorov-Smirnov test are available to test the normal distribution adequacy to describe the residuals' characteristics. Lin [18] points out that even though the normality assumption is not critical, a remedial step for correction of serious departures from the assumption is found in the use of data transformation.

The Assumption of the Equality of Residual Variance. The third assumption about the residuals is that they have equal variance at all levels of the experimental classifi-

cation. If the variance of the residuals differs from one observation to another, the usual ANOVA method of analysis leads to a loss of efficiency in the estimation of effects and a distortion of the significance level of analysis of various comparisons. This assumption, unlike the normality assumption, is crucial to the ANOVA method. Lin points out that where the number of discrete variables affected classifications, it could be expected that residual variance via expected solution time would increase as the problem size increased.

When the problem of inequality of residual variance is encountered, Lin proposes three alternatives. First, it may be necessary to proceed with the ANOVA in the usual manner, interpreting the apparent conclusions with more or less reserve. This alternative, as Lin points out, is not particularly appealing. The second alternative is to weight each observation in proportion to the inverse of its error variance. This approach may be reasonable when there are a large number of replicates in each cell; however this approach has a presupposition knowledge of the relative variances of any two observations and this is in practice seldom known. A third alternative, and one which is often used to obtain a constant error variance, is the use of transformations. Remembering that a transformation may be necessary to normalize the error distribution, Lin suggests that this third approach appears to be the most promising.

The problem of determining the best transform was generally treated first by Box and Cox [6]. The principle of the Box and Cox procedure is to find a transformation which maximizes the likelihood that the transformed data arose from a process with normally, independently, and identically distributed residuals. A family of Box and Cox transformations called the power transforms is of particular importance.

$$Y(\lambda) = \begin{cases} Y^\lambda, & \lambda \neq 0 \\ \text{Log } Y, & \lambda = 0 \end{cases}$$

For the details of how λ is chosen, see Box and Cox [6].

General Methodology

After finding that the ANOVA assumption can be satisfied with little or no data manipulation, Lin turned to the problem of developing a general methodology for statistical investigation of integer programming problems. The steps involved result from practical as well as theoretical considerations. The basic steps of the methodology are as follows:

- (1) Selection of the Relevant Parameters
- (2) Generation of Test Problems
- (3) Selection of Experimental Design
- (4) Experimentation
- (5) A Pre-Analysis of the Realizations
- (6) The Analysis of Variance

What follows are brief statements of some of the key components of each step of the methodology.

Selection of the Relevant Parameters. This step is primarily involved with determining which effects are to be included in the investigation. Included in the effects chosen to be investigated would be those of the different algorithms to be compared as well as properties of the problem which could affect the performance of either of the algorithms. Selection of the response or measure of performance by which to compare effects is also done in this first step.

Generation of the Test Problems. Because it is necessary to have the recorded measures of performance for the individual runs be the realization of random variables, it is necessary to generate the test problems randomly from some designated distribution. The first step in this stage is to specify the distribution from which problems are to be selected randomly. Selection of the proper distribution is important because any inferences made can only be extended to problems from that same distribution. The second step is to develop a random problem generator, and validate its performance. The final step entails the generation of a specified number of test problems.

Selection of an Experimental Design. In this step the actual experimental design is decided. Lin states probably the factorial designs are the most promising. Questions should be answered concerning how many levels of the effects are needed, should a full or fractional factorial design

be used, and how many replicates are desired if indeed replicates are desired. Once the design has been specified, the proposed model can be developed and the expected mean squares determined.

The Experimentation. The experimentation consists of running the experiment at all of the desired levels. The levels of the parameters and resulting measure of performance are collected and recorded. The experiments in Lin's work would consist of solving different integer programming problems on a computer.

A Pre-Analysis of the Realizations. This step deals with two common problems likely to be encountered with the realization from the experimental runs. The first problem is how should censored data be treated. The second problem deals with transforming the realizations that do satisfy all of the critical ANOVA assumptions.

A realization becomes censored when the final measure of performance can not be obtained because of the limited length of the experiment. Perhaps an integer programming problem can not be solved within some time limit set on the computer. In each such case, the experimental response value is not obtained, but some information is gained. The researcher knows the missing value falls outside some limit or bound. A number of alternatives have been proposed for treating censored data in the experimental design context. The most common three are the following:

- (1) Consider the censored data point as a missing value and use the usual least square error approach to estimate it.
- (2) Use the bound or limit value as though it were really the realization itself.
- (3) Try to estimate the true value of the censored point.

Lin points out that by far the third approach is the more desirable. Work by Sampford and Taylor [24] is the only known application of estimating the value of censored data using the maximum likelihood method done in an experimental setting, similar to the one encountered in Lin's work. Lin adapted their work and proposed its use for censored data. Further information on the iterative estimating procedure can be found in Sampford and Taylor [24].

The second area of concern in this step of the methodology deals with whether or not the realizations satisfy all of the necessary assumptions for an ANOVA to be calculated. The first phase of this step is to determine whether or not the realizations do actually satisfy every assumption, namely that the realizations are from a random variable and that the residuals are independent, normally distributed random variables with equal variance. If the realizations do satisfy all of these criteria, then the ANOVA may be performed directly on these realizations. However, if these criteria are not satisfied, then the realizations will have to be transformed so that they do. Lin suggests using the Box and Cox power transform family with

the maximum likelihood estimate for the appropriate power transform. For more on the Box and Cox procedure, see Box and Cox [6].

The Analysis of Variance. In this section the amount of variance that can be attributed to the different classifications is determined. The variance due to each classification is then compared with the variance due to the residuals by an F-test. This F-test for a given confidence level can be used to assess the significance of different treatments.

CHAPTER III

THE ANALYSIS PROCEDURE

3.1 Introduction

This chapter deals with the two pre-experimentation phases of the analysis methodology: (1) description of the experiment, and (2) development of the design. The first phase deals largely with the definition and description of the problem and of the factors to be varied. The information obtained from any experiment significantly affects which factors are varied and the manner in which they are varied. It is important then to understand the general treatments of interest and how these treatments affect the treatment of principal interest. In this work the treatments of direct interest are the two different free variables ranking rules as they affect the implicit enumeration solution algorithm of the capacitated quadratic assignment problem. The fundamental logic of each ranking strategy suggests that several parameters affecting the structure of the problem might also affect the superiority of either of the two free variable entering rules. The purpose of this paper is not only to investigate the relative performances of these ranking strategies, but also to point future algorithm developers toward selection of the better scheme under varied conditions; therefore several of these param-

eters will be varied in an attempt to determine how their levels influence the selection of the better free variable entering strategy.

3.2 The Experimental Treatments

The Response Variable

What response variable should be used as the measure of a procedure's performance is perhaps the first question encountered in the development of an empirical comparative study of integer solution procedures. Although many different measures have been suggested and reported, none are accepted as the standard. Perhaps the two most commonly used response measures are (1) the number of partial solutions explored, and (2) the computer time required to obtain and demonstrate that a solution is indeed optimal.

Both of these measures are good indicators of performance and, as could be expected, are generally very highly correlated. The two measures, however, differ in the type of units used to report performance. The number of partial solutions explored is an absolute measure, whereas the computer time is relative to and dependent upon the operating system in use.

Because of this distinction, the number of partial solutions for which a lower bound is calculated will be used as the measure of performance. This choice not only eliminates machine dependency of the results but also serves to reduce the error variation by eliminating the variance in

computer time between identical problems. The number of partial solutions examined before the optimal solution was determined and demonstrated, using a given free variable entering rule under varying problem conditions, then will be the measure of that free variable entering strategy's performance.

The Experimental Treatments

As discussed in Chapter II, the primary thrust of this work is to examine the effect certain free variable entering strategies have on the number of partial solutions investigated before it is demonstrated that a solution is optimal under varying problem conditions. The following is a brief description of the two fundamental ranking strategies and the three problem parameters or characteristics viewed as possibly interactive with the performance of these two fundamental ranking schemes.

The Ranking Scheme. Two fundamental ranking schemes have been proposed for an implicit enumeration solution procedure. The difficulty of determining the best ranking has been previously discussed, as has been the rationale behind the use of either of these two basic ranking strategies. Presented here is a brief review of each rule.

The first strategy ranks all free variables by the magnitude of its value in the constraint space, e.g., in the office building problem by the office's required size, from largest to smallest. Computationally, this scheme de-

termines which of the variables are free and then links with that free variable its associated constraint related value. These free variables are then ordered by that constraint attribute from the largest value to the smallest.

The second ranking scheme ranks all free variables by their contribution to the first component of the lower bound calculation process. In the office building problem the free offices would be ranked by their interactions to all offices already assigned a location. From a computational point of view, this ranking procedure determines which of the variables are free and then determines that variable's total interactions to all variables with a fixed location. The free variables are then ranked by that value from largest to smallest.

These procedures were coded in Fortran as was the entire solution procedure. Listings of these codes are included in Appendix A under subroutines SORT and SORT FX.

The Correlation Between the Optimized Attribute Ver-
the Constrained Attribute. Upon investigating the structure of the capacitated quadratic problem, one immediately becomes aware of the importance of two blocks of characteristics which, to a large degree, define the problem. One of these data blocks describes the amount of space associated with each variable within the constraint space. In the office building problem, this first data attribute would represent the size required by each office, remembering that

each floor has a limited amount of space available on which to locate offices. The second data characteristic describes the relationship between the free and fixed variables. It is some component of this block of the data for which an optimum value is being sought. Once again in the office building problem example, this second characteristic can be viewed as the interactions between offices which have free and fixed locations. The significance of each of these two problem attributes is brought into perspective by realizing that the first of these attributes influences the feasibility of a solution and the second affects the optimality of a solution.

The correlation between these two attributes is being used as the first parameter treatment for several reasons. First of all, Ahrens [1] demonstrated that the level of correlation between the weight and value vectors for the 0-1 knapsack problem using implicit enumeration can affect the time required to obtain the optimal solution. Secondly, the level of correlation between the free variable's fixed flow and size yields information about the relationship between the free variable entering strategies resulting from the two free variable ranking schemes. For instance, if a high positive correlation existed between these two data characteristics, then one expects that the ranking of the free variables from both of the ranking schemes might be very similar. However, if a high negative correlation ex-

isted then one expects the ranking rules to yield quite opposite results. This last situation seems to indicate that indeed one ranking scheme, given certain conditions, might perform in a superior manner. The point is that in many cases one would naturally expect a high correlation between these two attributes; i.e., in the office building problem, big offices would more probably have more flow than small offices, but this is not always the case in other examples of the capacitated quadratic assignment problem.

The Constraint Tightness. The second parameter treatment is the tightness of the constraints. Recall constraining Equation 2.2:

$$\sum_{i=1}^N S_i Y_{ir} \leq A_r \quad \text{for all } r = 1, 2, \dots, M$$

Because the constraints represented by Equations 2.3 and 2.4 are in fact maintained by the solution process itself, this equation or relationship determines whether or not a solution is feasible. The importance of pruning by infeasibility, one of the two basic reasons for the elimination of a given partial solution from further consideration, becomes obvious by remembering that the first ranking scheme enters free variables by size from largest to smallest in an attempt to facilitate effective pruning by infeasibility. However, it is apparent that unless the constraints are in-

deed active, i.e., unless the A_r are sufficiently small, pruning due to infeasibility will not occur. For this reason, the degree of tightness of the constraints represented by Equation 2.2 could very easily influence which free variable entering strategy would perform in a superior manner.

If one were to examine the partial solutions explored to determine an optimal solution with respect only to the pruning mechanism, then it would appear that the number of solutions explored would increase as the probability of pruning by infeasibility decreased; however, this view disregards the importance of the heuristic solution's upper bound value. It might very well be that as more slack is incorporated into the problem, that the heuristic solution would find a solution which is closer to the optimal value, thereby increasing the probability of pruning by non-optimality. It appears then that very likely there is some trade off between the pruning mechanisms as the problem slack is changed.

Of course there exist many measures of constraint tightness that could be used, but it is important to keep in mind that the purpose here is to investigate which free variable entering rule is best in light of characteristics of the problem which can be determined a priori to any solution generation. One manner in which the constraint tightness might be manipulated is seen by looking at Equation 3.3.

$$A_t = \sum_{r=1}^M A_r = \sum_{i=1}^N S_i Fx_i + \sum_{i=1}^N S_i Fo_i + L \quad (3.3)$$

where A_t = the total available floor space

A_r = the floor space of floor r

S_i = the size of office i

Fx_i = office i [1, if fixed; 0, otherwise]

Fo_i = office i [1, if free; 0, otherwise]

L = the slack in the system

M = the number of floors

N = the total number of offices

Equation 3.3 in regard to the office building problem expresses the total amount of space on the M floors on which to locate the N offices. This equation does not, however, give information as to how the total space is to be allocated among the M floors. Ideally, the slack could be controlled to the point that the slack on each floor would be equal. Equation 3.4 relates the space available on each floor with its components.

$$A_r = \sum_{i=1}^N S_i Fx_{ir} + \sum_{i=1}^N S_i Fo_{ir} + L_r \quad (3.4)$$

where A_r = the total floor space on floor r

S_i = the size of office i

Fx_{ir} = 1 if office i is fixed on floor r ;
0, otherwise

Fo_{ir} = 1 if free office i is to be located on floor r ;
0, otherwise

L_r = slack on floor r

N = the total number of floors

It can be seen from Equation 3.3 that given L , the total slack, A_t , the sum of all of the available floor space can be determined. However, because Fo_{ir} (the actual assignment of the free offices to the optimal floor) can not be determined prior to experimentation in Equation 3.4, there is no way to identify or control L_r , the slack on each of the floors. The question becomes, given that the slack is to be considered from a macroscopic viewpoint, how shall L , the total slack, be distributed to the individual floors.

The problem of distributing the total slack of the system to the M floors is dealt with by the following assumption: it is assumed that all of the floors have equal floor area, i.e., the area of each floor A_r is constant and is the total area divided by the number of floors. Note that for a multi-floor office building this might often be the case and in any event does not pose serious difficulties in physical interpretation. The constraint tightness then will be changed by increasing or decreasing the total space in which to locate all of the offices, the only variable component of which is the slack, and distributing this total space among the M floors such that the floor space on each

floor is equal. The slack L to be distributed among the M floors will be calculated or scaled as a function of the total space required by all of the free offices. Equation 3.5 illustrates this relationship.

$$L = A \sum_{i=1}^N S_i X_i \quad (3.5)$$

where L = the total system slack

A = the slack constant

N = the number of total offices

S_i = the size of office i

X_i = 1 if office i is free; 0, otherwise

The amount of slack to be incorporated into the problem then is viewed as some percent of the space needed by all of the free variables.

The Number of Free Variables. The third parameter treatment is the number of free variables for which an optimal solution is being sought. One recalls that Equation 2.5 relates the maximum number of partial solutions that would be investigated with the number of free variables. Given that no partial solution can be excluded from investigation,

$$N_p = \sum_{i=1}^n M^i$$

where N_p = the maximum number of partial solutions

M = the number of regions or floors

n = the number of free variables

From the form of Equation 2.5, it is easy to believe Geoffrion [12] when he reports that often it can be demonstrated that problem difficulty and therefore the solution time which is a function of the number of partial solutions investigated, grows exponentially with the number of free variables in a branch and bound solution process. As the number of free variables in a problem increases, so does the complexity of that problem. The number of free variables is being used as a factor in an attempt to determine if problem difficulty is drastically affected by the number of free variables and if the number of free variables is interactive with the choice of entering strategies.

Review of the Treatments. In the analysis of the comparative performance of the two specified free variable entering rules, the number of partial solutions before the optimal solution is identified and demonstrated shall be used as the measure of each of the two strategies' performances. The performance of the two free variable entering strategies will be compared for different levels for three characteristics which affect the structure of the problem.

The three treatments which will be manipulated throughout the experiment are (1) the correlation between the size and the fixed flow values of the free variables,

(2) the constraint tightness, and finally (3) the number of free variables.

3.3 Some Preliminary Statistical Comments

As previously stated, one factor which tremendously affects the power of the analysis of variance technique is the accuracy of the assumptions in the model underlying the procedure. For this discussion, assume that a simple one-way classification model can be used to describe each observation.

$$X_{ij} = M + T_i + E_{ij} \quad (3.6)$$

where M = the overall mean

T_i = the effect from treatment i

E_{ij} = residual

The necessary assumptions for an ANOVA to be performed are the following: (1) independence of residuals, (2) normality of the residuals, and (3) equality of residual variance.

Lin [18] points out as a result of these three assumptions that there exists two significant ramifications worthy of discussion at this point. The first ramification is that if indeed these three assumptions are to be satisfied, the test problem needs to be generated from a random process so that the solutions of the problem would themselves be random variables. The second ramification deals with the

problems encountered in the event that all of the analysis of variance assumptions are not satisfied. Of course, if a random generator is used, there is no reason to believe that the residuals will not be independent. Examples do exist, however, when the normality and equality of variance assumptions might not be satisfied. As discussed in the review of Lin's work, constant changes in the levels of some problem parameters often affect the problem difficulty in an exponential manner. If this does occur, obviously the residuals will not have equal variance across all treatments, e.g., the larger the number of free variables, the larger the variances. It also occurs in some of these problems that the residuals are not normally distributed. Both of these failures to satisfy the assumptions can be corrected by a transformation of the data. Much work has been done on developing the best transformator for a given data set; however, Lin [18] found that the Box and Cox power transform family handles this exponentially-increasing problem difficulty very well.

3.4 The Random Problem Generator

As indicated in previous sections, the selection of test problems in analysis of the comparative performances of solution procedures is crucial. It has been further stated by Lin [18] that of the three principal types of test problems (real world, designer developed, and randomly generated) the last or randomly generated problems are pre-

ferred. This is due to the generalizability of conclusions in that if an inference can be made about the superiority of one procedure from a random set of problems, then that inference can statistically be applied to all problems belonging to the set from which the test problems were drawn.

Little work has been done on developing parametric generating schemes. Parametric problem generators are generators which produce test problems having particular properties associated with aspects of the problem of interest to the investigator. Michaels and O'Neil [20] give several distinct advantages of parametric random problem generators.

- (1) The virtually limitless supply of test problems;
- (2) The ability to know and control problem characteristics;
- (3) The ability to conduct computational studies of the effects of parameter variation;
- (4) The option to regenerate a large test problem upon request as opposed to being required to save or store all of the data.

The advantages of parametric generators are apparent. The remainder of this section is devoted to a systematic development of the test problem generator used in this research. The parametric generator developed is one that controls the parameter treatments, which the literature suggest have a significant effect on the capacitated quadratic assignment problem's difficulty. These three param-

etric treatments, as previously discussed, are (1) the correlation between the free variables' flow to all fixed variables and the free variables' size, (2) the tightness of the size constraints, and (3) the number of free variables.

A Review of the Problem Generator Process

The problem generator randomly generates problems by developing the attributes of a capacitated quadratic assignment problem in two fashions. Some attributes are generated randomly; others are defined in a controlled manner.

Section 2.1 revealed that the capacitated quadratic office building location problem is generally described by the following three groups of attributes: (1) attributes dealing with the offices, (2) those dealing with the floors, and (3) those relating the various costs of movement between floors.

The random problem generator then needs to specify all the attributes in these groups in order to develop or generate a problem in its entirety. The manner in which these attributes are defined, however, differ. Several attributes of the problem, e.g., the number of free offices, the size of the floors, and the relationship between a given office's fixed flow and its size, are factors determining which levels will be changed throughout the design. It is advantageous to designate these levels as fixed quantitative values, therefore it is necessary to exert a cer-

tain amount of control over the random variables which will be drawn upon to define these attributes. It is important to realize that even though these attributes are being controlled and do therefore set some limitations on the scope of any inference, the problems generated via this controlled process are still random problems from the set of problems characterized by the controls.

The problem generator developed in this work, or the General Capacitated Quadratic Assignment Random Problem Generator, controls the following attributes:

I. Controlled Attributes

- (1) Total number of offices
- (2) The number of free offices
- (3) The number of floors
- (4) The size of the floors
- (5) The correlations between the flow among free offices and fixed offices and the size of the free offices
- (6) The density of nonzero flow values in the flow matrix.

The problem generator randomly defines the following attributes:

II. Randomized Attributes

- (1) The selection of the free offices
- (2) The location of the fixed offices
- (3) The size of the offices
- (4) The flow between offices
- (5) The cost of movement between floors

The Generating Procedure

Outlined below is the procedure for the General Capacitated Quadratic Assignment Random Problem Generator. Appendix B contains a computer code of this procedure.

Step 1. Given N , the number of total variables and how many of these are free, select which of the N total variables will have a fixed location and which will have a free location.

Step 2. Determine in a random manner on which of the M locations each of the fixed variables will be located.

Step 3. Generate the size and total fixed flow of each free variable using a bi-variable normal distribution so that the desired correlation exists between each free office's fixed flow and the size.

Step 4. Generate from a uniform distribution the size of each of the remaining fixed variables in order to complete the size attribute of the variables.

Step 5. Given a desired nonzero density of the flow matrix, generate the flow values from all free variables to all fixed variables from either a uniform or negative exponential distribution and then modify or adjust the nonzero components of this fixed flow for each free variable so that the sum of these components equals the total fixed flow for that free variable generated in Step 3.

Step 6. Generate the flow values that represent the flow between the fixed variables. Once again these values

can be generated from either a uniform or exponential distribution.

Step 7. Generate the flow values which represent the flow from free variables to other free variables once again by either a uniform or exponential distribution.

Step 8. Calculate the floor space or region constraint constant given that the location and size of the fixed variables, the size of the free variables, and the total system slack assuming that the constraint constant is the same for every set.

Step 9. If not specified, determine the cost of movement between floors randomly, each cost to be within a given range.

An Example Problem

To demonstrate the performance of this random problem generator, let us proceed step by step through a sample problem. Generate a problem with the following characteristics:

- (1) Ten total variables
- (2) Four free variables
- (3) Three regions
- (4) An .80 nonzero density for the flow matrix
- (5) A .9 correlation between the free variables fixed flow and size
- (6) A total slack value of one half of the total area required by the free variable

(7) All flow values are generated from a uniform distribution

(8) An equal cost of movement between all regions

The steps which follow all refer to the steps outlined in the previous section.

Step 1. Given the number of total and free variables, the procedure randomly places the appropriate number of variables into vectors FIXED and FREE.

FREE = [3, 4, 9, 10]

FIXED = [1, 2, 5, 6, 7, 8]

Step 2. Given the number of regions, the elements of the vector FIXED are randomly assigned to vectors associated with those regions.

REG 1 = [5, 8]

REG 2 = [2, 6]

REG 3 = [1, 7]

Step 3. Given that a certain correlation is desired, generate the size and total fixed flow of each free variable using a bi-variate normal distribution.

FXFLO = [544, 480, 369, 521]

SIZE = [534, 478, 403, 492]

Step 4. Using a uniform distribution, generate the size of the remaining fixed variables to complete the size vector.

SIZE = [244, 64, 534, 478, 187, 145, 384, 201, 403, 492]

Step 5. Given a desired nonzero density of the flow matrix A, generate the flow values from all free variables to all fixed variables such that the total fixed flow for every free variable is equal to the value generated in Step 3.

$$A = \begin{bmatrix} 111 & 0 & 0 & 127 & 120 & 186 \\ 122 & 81 & 0 & 145 & 0 & 132 \\ 0 & 0 & 149 & 0 & 110 & 110 \\ 74 & 95 & 119 & 162 & 0 & 71 \end{bmatrix}$$

Step 6. Generate from a uniform distribution the values of matrix B, where matrix B represents the flow between all of the fixed variables.

$$B = \begin{bmatrix} 0 & 10 & 49 & 29 & 8 & 23 \\ 10 & 0 & 1 & 76 & 46 & 76 \\ 49 & 1 & 0 & 78 & 75 & 0 \\ 29 & 76 & 78 & 0 & 20 & 78 \\ 8 & 46 & 75 & 20 & 0 & 6 \\ 23 & 76 & 0 & 78 & 6 & 0 \end{bmatrix}$$

Step 7. Once again from a uniform distribution, generate matrix C where matrix C represents the flow between all free variables.

$$C = \begin{bmatrix} 0 & 0 & 0 & 78 \\ 0 & 0 & 58 & 27 \\ 0 & 58 & 0 & 4 \\ 78 & 27 & 4 & 0 \end{bmatrix}$$

Step 8. Knowing that P, the slack constant as a percent of space required by the free variables, is equal to .5, calculate L, the total slack.

$$L = (P) \left(\sum_{i=1}^n S_i X_i \right)$$

where L = the total system slack

P = the percent of addition slack

S_i = the size of office i

X_i = 1 if office is free; 0 if office i is fixed

n = the number of free variables

$$L = (.5)(1907) = 953.$$

Knowing L, the total slack and the total space required by the free and fixed variables, calculate the total system space and determine A_r , the space per region given that each region has the same size.

$$A_r = \frac{A_t}{m} = \frac{(1225 + 1907 + 953)}{3} = 1362$$

Step 9. Given that a problem is desired with an equal cost of movement between floors,

$$C_{12} = C_{13} = C_{23} = 1$$

3.5 The Experimental Design

The second phase of the pre-experimentation methodology consists of the development of an efficient design. The principle of experimentation sets forth the importance of varying the factors of independent treatments in order to observe the effect such changes have on the dependent variables. Traditionally, factors tend to be varied individually, rather than in combination. Sequential experimentation as opposed to factorial experimentation, however, now is viewed as inadequate for the following reasons. Firstly, the sequential approach of varying treatment levels precludes the investigation of the interaction effect between factors. Secondly, the sequential approach may lead the investigator to attribute effects which influence the dependent variables to some of the treatments which effects are in reality not a result of these factors, but rather variations in some factors not included in the experiment. The factorial design eliminates these two problem areas.

In Section 3.2., the effects and relative importance of departure from the assumptions on the analysis of variance were discussed. Box [5] points out that designs with

equal frequencies in all design classification cells have two statistical advantages: the computations are extremely simplified and the effects of inequality of error variance are not serious. Therefore, the use of a factorial design with an equal number of observations in each design cell is of great benefit.

Recall from Chapter I that this student's intent is to investigate the relative superiority of two basic free variable entering strategies over varying problem conditions. In Section 3.1, the three problem characteristics of interest were introduced and developed. These three treatments are:

- (1) The correlation between the fixed flow and the size of the free variables
- (2) The constraint tightness
- (3) The number of free variables

Each of these three treatments will be investigated at three levels in an attempt to minimize the likelihood that any non-linear characteristics that might be exhibited will not be viewed as linear ones. The above prescribes a design of the form of Figure 3.1. Note that there are two levels for the entering rule effect and that these two levels represent the two basic free variable entering strategies. Further note that all other treatments have three levels. These three levels are at fixed, equal intervals and are quantitative values. The correlation levels will

be fixed at .9, 0.0, and -.9. The constraint tightness will be manipulated by adding to the space required by all of the offices an additional slack of 50, 125, or 200 percent of the total area required by the free variables. Problem difficulty will be controlled to some extent by the number of free variables. This design will include levels of 12, 13, and 14 free variables. These last levels are chosen because of the capacity of the algorithm to find and assure an optimum solution within a reasonable expenditure of resources.

The equal frequency, factorial design approach to such an experiment would randomly generate a series of R problems, characterized by the levels defining each cell, and then would solve each problem. For each of these problems, the number of branches explored would be recorded. An analysis of variance of these observations would then be performed to indicate whether or not the free variable entering strategies statistically differ in performance.

It is interesting to note that experiments arranged in this way use problems which can not be compared across different levels of the parameter treatments; that is to say that the variation among randomly generated problems can only be understood in light of the levels of the different parametric treatments. This occurrence in experimental design is known as "nesting".

A review of Figure 3.1 indicates that the problem

		N = 12						N = 13						N = 14							
		P = -0.9		P = 0.0		P = 0.9		P = -0.9		P = 0.0		P = 0.0		P = -0.9		P = 0.0		P = 0.9			
		T = 0.50	T = 1.25	T = 2.00	T = 0.50	T = 1.25	T = 2.00	T = 0.50	T = 1.25	T = 2.00	T = 0.50	T = 1.25	T = 2.00	T = 0.50	T = 1.25	T = 2.00	T = 0.50	T = 1.25	T = 2.00		
S ₁	PROBLEM SET 1																				
	PROBLEM SET 2																				
	S ₂	PROBLEM SET 1																			
		PROBLEM SET 2																			

FIGURE 3.1
THE EXPERIMENTAL DESIGN

effect, although it can not be compared across different parametric treatment levels, can be compared across the free variable entering strategies. This results because the series of problems solved in the cells where only the free variable entering strategy levels are changed are identical. Thus, the problem effect is nested, but only in the number of free variables, the correlation, and the constraint tightness effects.

The realizations of the experiment in fact involve five main effects besides error variance. Variance among the number of partial solutions explored can be viewed as being caused by:

- (1) the variation among the two free variable entering strategies
- (2) the variation among the levels of the correlation between free variables' fixed flow and size
- (3) the variation among the levels of constraint tightness
- (4) the variation among the levels of the number of free variables
- (5) the variation between problems generated within levels of the parametric treatments.

The first four effects are all fixed level effects and the fifth or problem effect is random.

A statistical model of the design can be seen in Equation 3.7.

$$\begin{aligned}
& M + S_i + N_j + SN_{ij} + C_k + SC_{ik} + NC_{jk} + SNC_{ijk} \\
Y_{ijklm} = & + T_l + ST_{il} + NT_{jl} + SNT_{ijl} + CT_{kl} + SCT_{ikl} + \\
& NCT_{jkl} + SNCT_{ijkl} + P_m^{(T)}(jkl) + SP_{im}^{(T)}(jkl) \\
& + E_m(ijkl) \tag{3.7}
\end{aligned}$$

- where Y_{ijklm} = the response on replicate M at the parametric treatment levels j, k, l for number of free variables, correlation, and constraint tightness respectively solved by free variable entering strategy i .
- M = the overall mean response
- S_i = the effect due to the i^{th} free variable entering strategy
- N_j = the effect due to the j^{th} number of free variables level
- SN_{ij} = the effect due to strategy versus number of free variables interaction for strategy i and number of free variables j .
- C_k = the effect due to the k^{th} level of the correlation treatment
- SC_{ik} = the effect due to the strategy i versus correlation level k interaction
- NC_{jk} = the effect due to the number of free variables level j versus the correlation level k interaction
- SNC_{ijk} = the effect due to the strategy i versus number of free variables level j versus correlation level k interaction
- T_l = the effect due to the constraint tightness level l
- ST_{il} = the effect due to the strategy i versus constraint tightness level l interaction

- NT_{jl} = the effect due to the j^{th} level of the number of free variables versus the l^{th} level of constraint tightness
- SNT_{ijl} = the effect due to the strategy i versus number of free variables level j versus the constraint tightness level l
- CT_{kl} = the effect due to the correlation at level k versus the constraint tightness at level l
- SCT_{ikl} = the effect due to the i strategy versus the correlation at level k versus the constraint tightness at level l
- NCT_{jkl} = the effect due to the number of free variables at level j versus the correlation level k versus the constraint tightness at level l
- $SNCT_{ijkl}$ = the effect due to strategy i versus the number of free variables at level j versus the correlational level k versus the constraint tightness at level l
- $P_m(jkl)$ = the effect due to problem or replicate M with the j^{th} , k^{th} , and l^{th} levels of number of free variables, correlation, and constraint tightness respectively
- $SP_{im}(jkl)$ = the effect due to strategy i versus problem M interaction

Note that one obvious difficulty does exist and that is the individual $SP_{im}(jkl)$ and $E_m(ijkl)$ are observed on exactly the same Y_{ijklm} . The result is that the variation due to strategy versus problem interaction can not be distinguished from residual error unless an independent estimate of error variance is available. However, this confounding does not eliminate the ability to construct tests for significance of either the main free variable entering strategy

effect (S), the number of free variables effect (N), the correlation between the free variables fixed flow and size effect (C), the constraint tightness effect (T), or the interaction effects between any of these. Table 3.1 presents the expected mean square table for the proposed design.

From Table 3.1, the proper test statistics are:

$$F_S = \frac{SS_S / (a-1)}{SS_{SP(N,C,T)} / bcd(a-1)(e-1)}$$

$$F_N = \frac{SS_N / (b-1)}{SS_P / bcd(e-1)}$$

$$F_C = \frac{SS_C / (c-1)}{SS_P / bcd(e-1)}$$

$$F_T = \frac{SS_T / (d-1)}{SS_P / bcd(e-1)}$$

$$F_{SN} = \frac{SS_{SN} / (a-1)(b-1)}{SS_{SP} / bcd(a-1)(e-1)}$$

$$F_{SC} = \frac{SS_{SC} / (a-1)(c-1)}{SS_{SP} / bcd(a-1)(e-1)}$$

$$F_{NC} = \frac{SS_{NC} / (b-1)(c-1)}{SS_P / bcd(e-1)}$$

$$F_{ST} = \frac{SS_{ST} / (a-1)(d-1)}{SS_{SP} / bcd(a-1)(e-1)}$$

$$F_{NT} = \frac{SS_{NT} / (b-1)(d-1)}{SS_P / bcd(e-1)}$$

$$F_{CT} = \frac{SS_{CT} / (c-1)(d-1)}{SS_P / bcd(e-1)}$$

$$F_{SNC} = \frac{SS_{SNC} / (a-1)(b-1)(c-1)}{SS_{SP} / bcd(a-1)(e-1)}$$

$$F_{SNT} = \frac{SS_{SNT} / (a-1)(b-1)(d-1)}{SS_{SP} / bcd(a-1)(e-1)}$$

$$F_{SCT} = \frac{SS_{SCT} / (a-1)(c-1)(d-1)}{SS_{SP} / bcd(a-1)(e-1)}$$

$$F_{NCT} = \frac{SS_{NCT} / (b-1)(c-1)(d-1)}{SS_P / bcd(e-1)}$$

$$F_{SNCT} = \frac{SS_{SNCT} / (a-1)(b-1)(c-1)(d-1)}{SS_{SP} / bcd(a-1)(e-1)}$$

Note: where SS_S means the Sum of Squares for effect S and where a, b, c, d, e are the number of levels for effects S, N, C, T, P respectively.

Chapter IV presents the results of an experiment run according to this design. Before discussing these results, a discussion of the parameters of the problem for which levels were fixed is appropriate. The problem characteristics which were given a fixed level through the experiment were as follows:

- i) The number of floors was set equal to three.
- ii) The total number of offices was set equal to 57.
- iii) The cost of movement between floors was held constant and equal for all movements.
- iv) The density of nonzero entries in the flow matrix was fixed so that 60 percent of all flow values would equal zero.

Table 3.1 Expected Mean Squares

<u>Effect</u>	<u>Expected Mean Square</u>	<u>d.f.</u>
S	$\sigma^2_{SP(N,C,T)} + bcde\sigma^2_S$	(a-1)
N	$a\sigma^2_{P(N,C,T)} + acde\sigma^2_N$	(b-1)
C	$a\sigma^2_{P(N,C,T)} + abde\sigma^2_C$	(c-1)
T	$a\sigma^2_{P(N,C,T)} + abce\sigma^2_T$	(d-1)
$P_{(N,C,T)}$	$a\sigma^2_{P(N,C,T)}$	bcd(e-1)
SN	$\sigma^2_{SP(N,C,T)} + cde\sigma^2_{SN}$	(a-1)(b-1)
SC	$\sigma^2_{SP(N,C,T)} + bde\sigma^2_{SC}$	(a-1)(c-1)
NC	$a\sigma^2_{P(N,C,T)} + ade\sigma^2_{NC}$	(b-1)(c-1)
ST	$\sigma^2_{SP(N,C,T)} + bce\sigma^2_{ST}$	(a-1)(d-1)
NT	$a\sigma^2_{P(N,C,T)} + ace\sigma^2_{NT}$	(b-1)(d-1)
CT	$a\sigma^2_{P(N,C,T)} + abe\sigma^2_{CT}$	(c-1)(d-1)
$SP_{(N,C,T)}$	$\sigma^2_{SP(N,C,T)}$	bcd(a-1)(e-1)
SNC	$\sigma^2_{SP(N,C,T)} + de\sigma^2_{SNC}$	(a-1)(b-1)(c-1)
SNT	$\sigma^2_{SP(N,C,T)} + ce\sigma^2_{SNT}$	(a-1)(b-1)(d-1)

Table 3.1 Expected Mean Squares (continued)

<u>Effect</u>	<u>Expected Mean Square</u>	<u>d.f.</u>
SCT	$\sigma^2_{SP(N,C,T)} + be\sigma^2_{SCT}$	$(a-1)(c-1)(d-1)$
NCT	$a\sigma^2_{P(N,C,T)} + ae\sigma^2_{NCT}$	$(b-1)(c-1)(d-1)$
SNCT	$\sigma^2_{SP(N,C,T)} + e\sigma^2_{SNCT}$	$(a-1)(b-1)(c-1)(d-1)$

where the levels of S, N, C, T, and P_(N,C,T) are a, b, c, d, and e respectively.

CHAPTER IV

ANALYSIS OF THE EXPERIMENTAL RESULTS

4.1 An Examination of Assumptions

The design as specified in Chapter III was run and the appropriate measures of performance were collected. The results of the experiment can be found in Appendix C. Before proceeding to perform an analysis of variance, it is necessary to investigate the experimental results to determine if the ANOVA assumptions are indeed satisfied for the proposed model. The assumptions of concern at this point are the following.

- (1) The residuals are to be normally distributed.
- (2) The residuals are to be identically distributed, e.g., the residuals are to have constant and equal variance.

Box [4] has pointed out that the analysis of variance method is very robust and actually is quite insensitive to departures from normality in the underlying populations being sampled. It has been shown that ANOVA methods are relatively powerful as long as the distribution being sampled is symmetric and is not bimodal. The equal variance assumption, however, appears to be highly significant and is perhaps the most critical ANOVA assumption.

Figures 4.1 and 4.2 show the scatter diagrams of cell means versus cell variance, and cell mean versus cell stan-

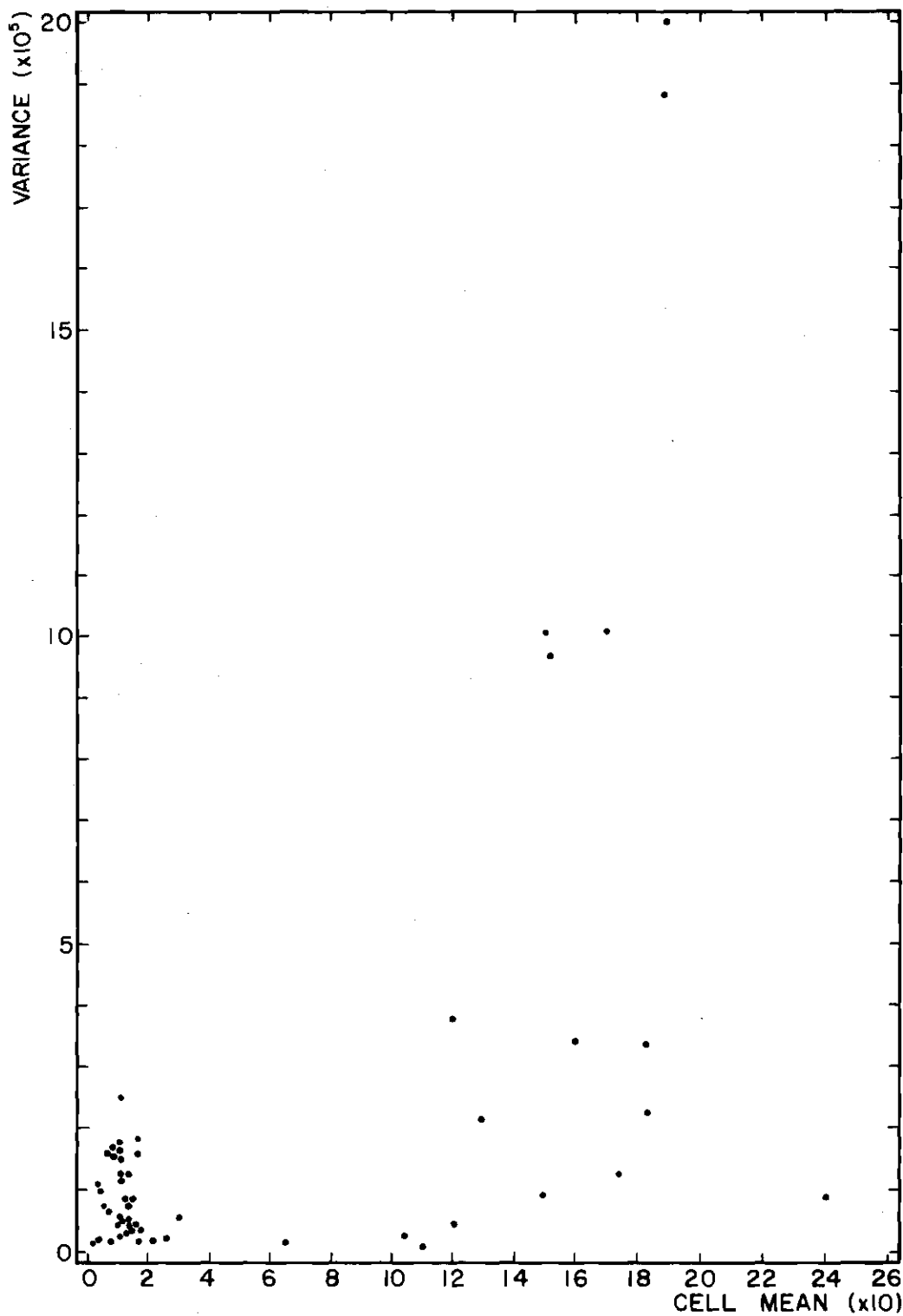


FIGURE 4.1
PLOT OF
RESIDUAL CELL VARIANCE vs. CELL MEAN

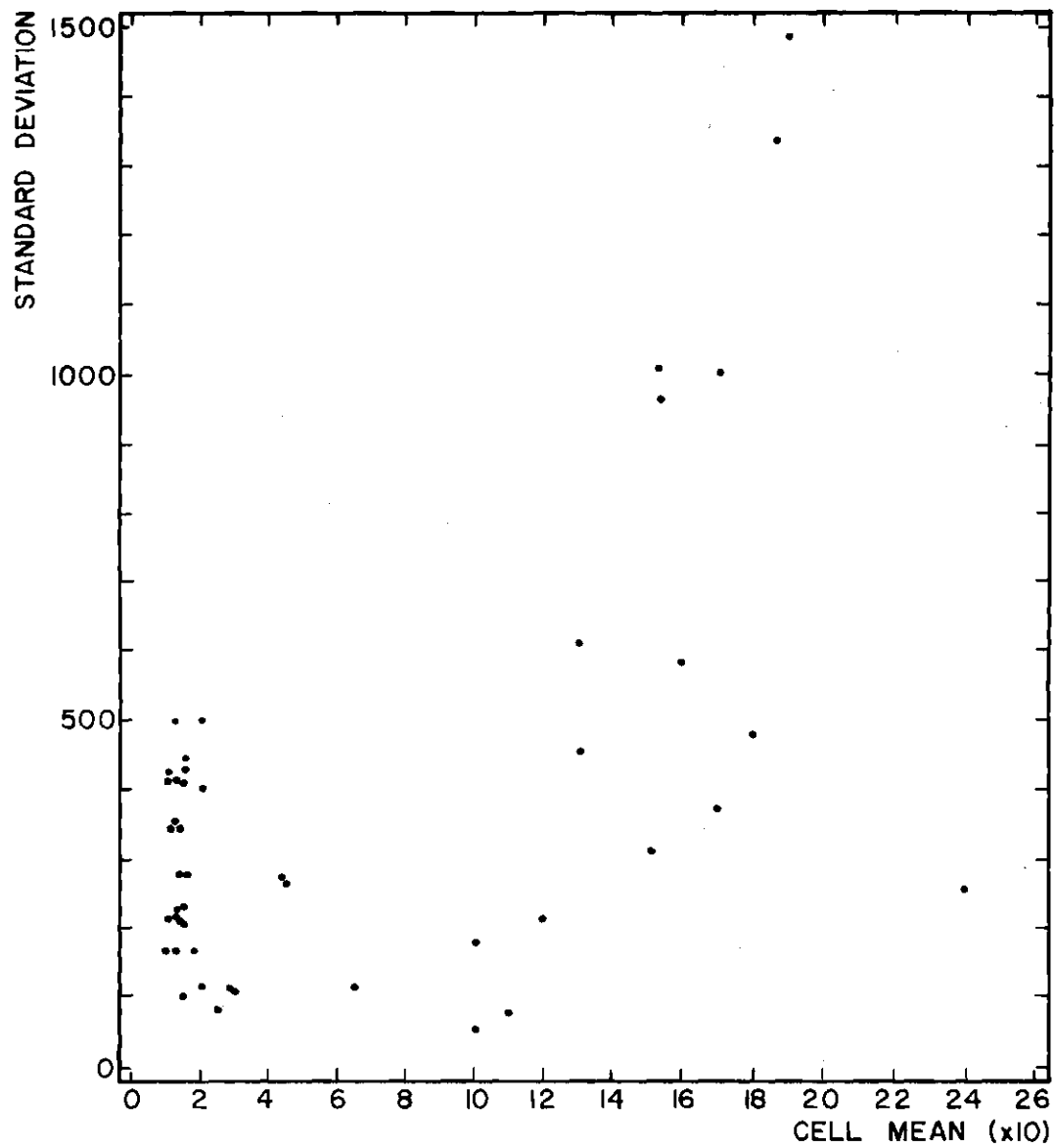


FIGURE 4.2
PLOT OF
RESIDUAL STANDARD DEVIATION vs. CELL MEAN

dard deviation respectively of the residuals observed when the appropriate ANOVA model was fitted to the results from the experiment. It is evident in these figures that there was a trend between cell mean and cell variance, i.e., the cell variance gets larger as the cell mean increases. Although this trend is not desirable, the possibility of such a trend was discussed in Chapter II. In Figure 4.1, it appears as though the variance is increasing in an exponential manner. Because the family of power transforms will be the only transform considered in this work, it appears that the logarithm base 10 might be a transform which would correct the exponentially-growing variance nicely. Lee [16] adds credibility to this, noting that the log transform is an excellent transformation for correcting trend between cell mean and cell variance. The suggested transform then is:

$$Y_i = \log_{10} x_i \quad (4.1)$$

At this point, the number of partial solutions examined needs to be transformed, collected, and re-examined with respect to the ANOVA assumptions. Appendix C shows these transformed responses.

Figure 4.3 shows the scatter diagram of cell mean versus cell variance of the residuals observed when the appropriate ANOVA model was fitted to the transformed observations. Since Figure 4.3 appears scattered around a flat

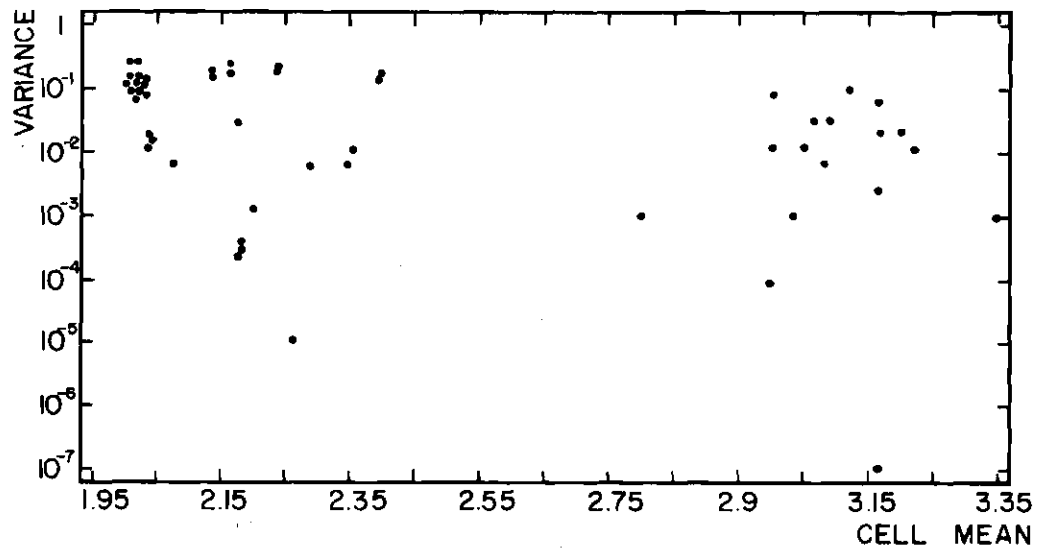


FIGURE 4.3
PLOT OF THE RESIDUAL VARIANCES
OF THE TRANSFORMED OBSERVATION

straight line, it suggests that cell variances are rather uniform. With only two replicates per cell, however, it is impossible to further justify the hypothesis of equality of variance by statistical tests. Figure 4.4 is a histogram of the residuals. Note that this histogram is definitely unimodal. The ANOVA assumptions appear to be at least loosely satisfied and at this point that is all that can be said. In order to obtain more information about the equality of cell variance under the transformation, data must be pooled.

4.2 An Analysis of Variance

Table 4.1 shows the sum of squares, degrees of freedom and mean square variation due to each of the main effects in the experiment. An examination of Table 4.1 reveals that the amount of variation attributed to N , the number of free variables, is relatively insignificant. This suggests that perhaps data can be pooled by eliminating the distinction between the levels of the number of free variables. This enables the original 2 by 3 and 3 by 3 design with two replicates to be collapsed into a 2 by 3 by 3 design with six replicates. Table 4.2 shows the ANOVA results of this design.

This new design has six replicates per cell and therefore contains sufficient degree of freedom remaining after estimation of cell mean and cell variance to examine statistically whether or not the variance is uniform through-

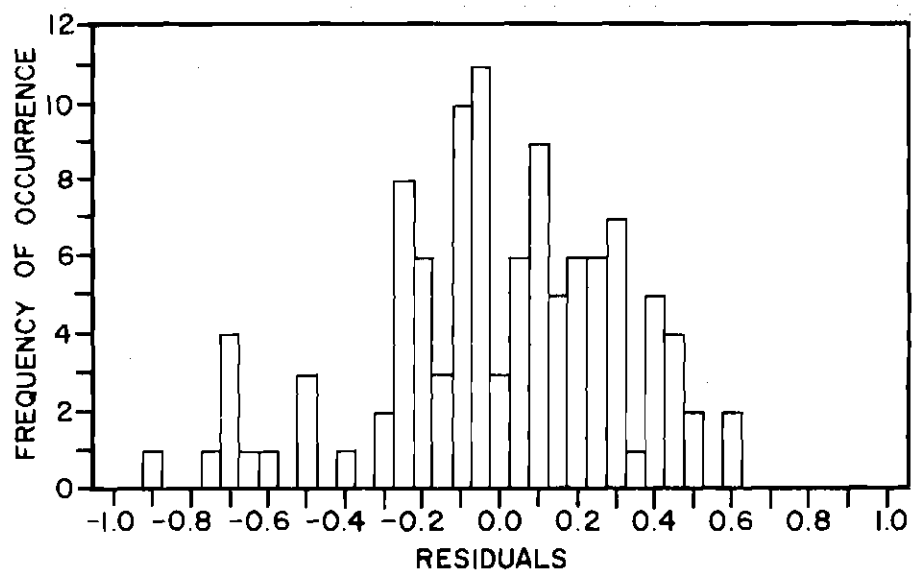


FIGURE 4.4
HISTOGRAM OF ERROR RESIDUALS
OF TRANSFORMED OBSERVATIONS

Table 4.1 An ANOVA of the Transformed Observations

<u>Effect</u>	<u>Name</u>	<u>S.S.</u>	<u>d.f.</u>	<u>M.S.</u>	<u>F</u>
S	Strategy	.421	1	.421	22.16 *
N	Number	.095	2	.047	.19
C	Correlation	.329	2	.165	.67
T	Tightness	24.247	2	12.123	49.36 *
SN	SXN interaction	.128	2	.064	3.37
SC	SXC interaction	.131	2	.066	3.47
NC	NXC interaction	.128	4	.032	.13
NT	NXT interaction	.181	4	.045	.18
ST	SXT interaction	.038	2	.019	1.0
TC	TXC interaction	.036	4	.009	.036
SNT	SXNXT interaction	.051	4	.013	.68
SNC	SXNXC interaction	.064	4	.016	.84
STC	SXTXC interaction	.023	4	.006	.32
NTC	NXTXC interaction	.011	8	.001	.004
SNTC	SXNXTXC interaction	.024	8	.003	.16
P	Problem	6.632	27	.245	
SP	SXP interaction	.514	27	.019	

*Statistically significant at .99 confidence level

Table 4.2 An ANOVA with the N Effect Pooled

<u>Effect</u>	<u>Name</u>	<u>S.S.</u>	<u>d.f.</u>	<u>M.S.</u>	<u>F</u>
S	Strategy	.421	1	.421	17.915*
C	Correlation	.329	2	.164	.732
T	Tightness	24.247	2	12.124	54.125*
P	Problem	2.015	9	.224	2.872*
SC	SXC interaction	.131	2	.066	2.808
ST	SXT interaction	.038	2	.019	.808
SP	SXP interaction	.212	9	.023	.301
CT	CXT interaction	.036	4	.009	.080
SCT	SXCXT interaction	.023	4	.006	.255
E	Error	5.602	72	.078	

*Statistically significant at .99 confidence level

out all of the design cells. Two basic statistical tests of homoscedasticity are the Hartley and Bartlett tests. The Hartley statistic equalled 19.7. This value failed to lead to a rejection of the hypothesis that the variances were equal with six degrees of freedom, 18 classifications, at 95 percent confidence level. The Bartlett statistic for equality of variance equalled 34.2 and failed to reject the equal variance hypothesis with 35 degrees of freedom only at a confidence level of .50 percent. These statistics, then, appear to indicate that equal variance may possibly exist among the cells.

While these tests of the validity of the equality of variance assumption of the analysis of variance procedure are not entirely satisfying, they do appear to justify the application of the procedure. Figure 4.6 indicates that the residuals of the transformed observations are apparently non-normal, but they are unimodal. However, studies have shown that the ANOVA is robust to the normality assumption as long as the residuals are unimodal. Also the crucial assumption of equality of variance appears to be satisfied for the transformed observations.

4.3 Final Analysis of Variance

As outlined in the previous section, the original experimental design was reduced to a 2 by 3 by 3 design with six replicates. A preliminary analysis of the proposed statistical model indicates that the logarithm function trans-

forms the original observations in such a manner that the ANOVA assumptions are at least loosely satisfied.

The model suggested by the collapsed design is as follows:

$$Y_{ijk} = M + S_i + C_j + SC_{ij} + T_k + ST_{ik} + CT_{jk} \\ + SCT_{ijk} + P_{L(j,k)} + SP_{iL(j,k)} \quad (4.2)$$

where each letter suggests the same effect as Equation 3.7.

Main effects and interactions on this new model whose F-ratios are greater than their corresponding critical F-value are listed in Table 4.3. Table 4.4 gives the estimate of the effect of moving from different levels of each main effect. In Table 4.4, a positive effect indicates that the mean number of partial solutions examined was larger at high levels, and in a likewise manner, a negative value indicates that the mean number of partial solutions examined was higher at the low level setting.

Before proceeding, it should be recalled that the data analysis and statistical tests proposed in this chapter are only approximate. It must be stressed that a value bordering on a level of significance of the F-distribution should be accepted as significant at that level only with reservation.

4.4 Examination of Main Effects and Interactions

From Table 4.3, it can be seen that the only signifi-

Table 4.3 The Statistically Significant Effects

S, Strategy
T, Constraint Tightness
P, Problem Effect

Table 4.4 Estimates of the Main Effects

<u>Treatment</u>	<u>Estimated Effect</u>
S	-.062
T	.511
C	-.067
P	.129

cant main effects or interactions are S, the free variable entering strategy; T, the constraint tightness; and F, the problem effect. C, the correlation between the fixed flow and size and all of the interactions, is insignificant.

Figure 4.6 is a graphic representation of all of the two-way interactions. It appears from these graphs that masking of the effects is not taking place, so it can be concluded that indeed these interactions are insignificant.

S, the free variable entering strategy effect, however, is significant at a .99 percent confidence level. The estimate of the free variable entering strategy effect is negative. This indicates that the mean number of partial solutions explored before an optimal solution is found and demonstrated is larger for strategy 1 than for strategy 2.

Effect T, or the constraint tightness, was also found to be significant at a .99 percent confidence level. The estimated effect of T was positive. This means that as the constraints are made tighter, the mean number of partial solutions and thus problem difficulty increases.

A Newman-Keuls Range Test indicated that there was a significant difference in the mean number of partial solutions between the first and second levels, but not between the second and third levels. This indicates that little simplification of the problem occurs when more than 1.25 percent of the space required by the fixed variable is allowed for the assignment of the free variables.

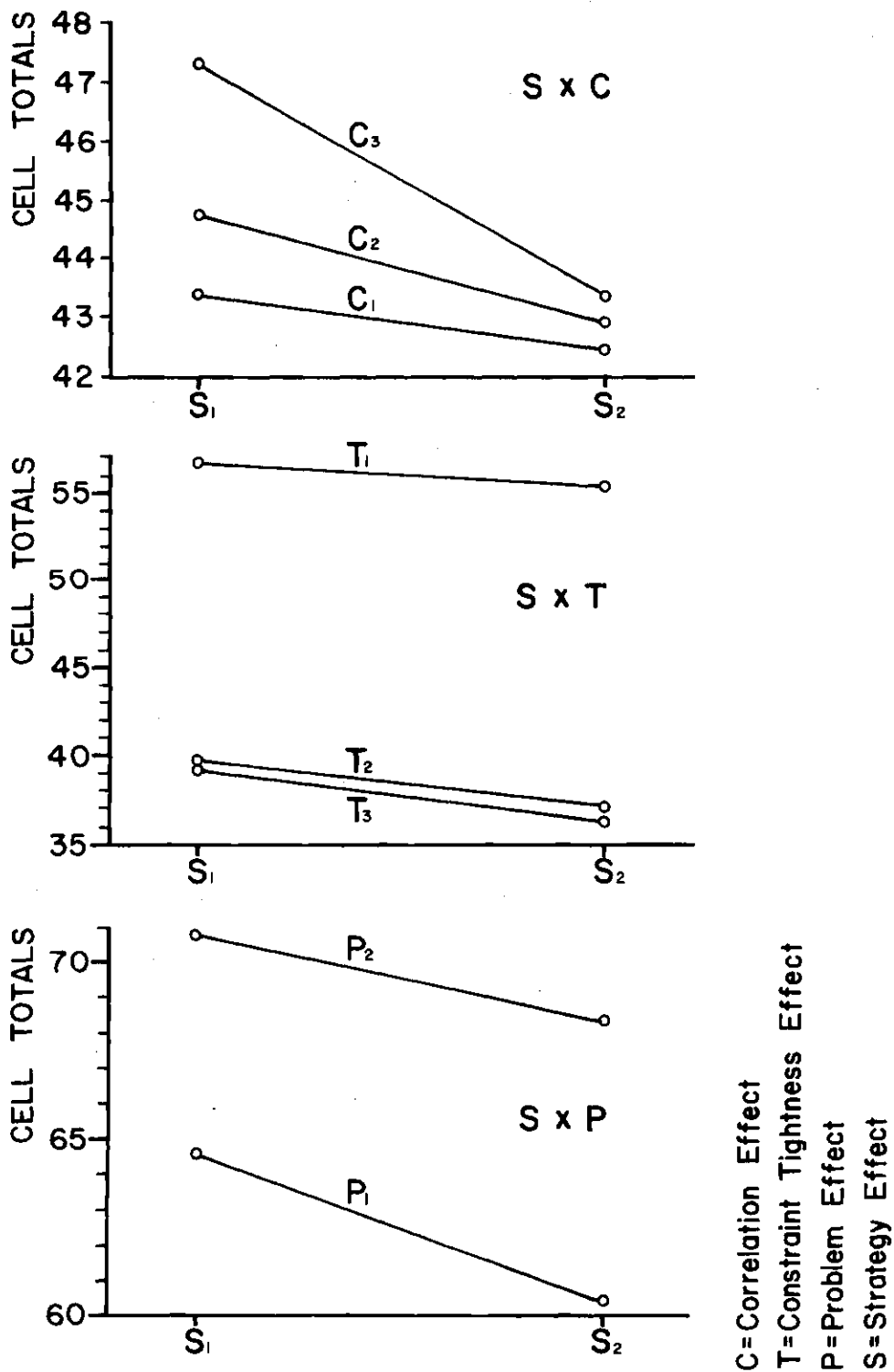


FIGURE 4.5
 GRAPHICAL REPRESENTATION OF
 SOME TWO WAY INTERACTIONS

Lastly, P, the problem effect, was found to be significant at a .99 percent confidence level. This confirms the suspicion of many that the implicit enumeration solution procedure performance is highly problem dependent.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

5.1 Summary of Results

The result of this study indicates that free variable entering strategy two, or ranking by a free variable's connectiveness with all fixed variables, is significantly superior to strategy one, or ranking free variables by their size requirement, for the class of capacitated quadratic assignment problems considered in this study. No significant interaction was found to exist between strategy selection and various problem parameters. The constraint tightness was found to be the most significant effect of the problem parameter examined. As was expected, problem difficulty increased sharply as the constraints became more and more active. The problem effect was also significant. This finding reconfirms the belief held by many that interpretation of the traditional form of analysis of the capacitated quadratic assignment problem is extremely hampered by the problematic effect.

The number of free variables and the correlation between the fixed flow and size of the free variables were not found to be statistically significant. These findings suggest areas for further work. Recall that the number of free variables were 12, 13, and 14 for levels 1, 2, and 3

respectively. It is quite possible that if the interval between these levels was increased that the magnitude of the problem might be changed sufficiently enough to be reflected in the chosen measure of performance.

The main results of the research effort can be summarized as follows:

- (1) The development of a capacitated quadratic assignment random problem generator.

The generator developed in this work provides randomly created complete capacitated quadratic assignment problems. The development of this generator facilitates the examination of these problems in future work.

- (2) A demonstration of the validity of the analysis of variance procedure as a means of comparing the performance of solution algorithms over wide ranges of problem conditions.
- (3) The examination and selection of the better of two basic heuristic free variable entering strategies for a least bound branching implicit enumeration solution procedure.
- (4) An examination of several general factors of the capacitated quadratic assignment problem in order to determine their influence on problem solution difficulty.

5.2 Recommendations for Further Research

At the conclusion of this research, certain areas stand out as worthy of mention as possible areas for further work. These areas are outlined and summarized in the following groups.

- (1) The reformation of the capacitated quadratic assignment problem in terms of other practical problems.

The problems dealt with in this work were characterized by having a large number of variables with a fixed as opposed to a free assignment. The structure and logic of other problems will likely suggest different heuristic free variable entering strategies.

- (2) The development of new free variable entering strategies.

The results of this work indicate that strategy two is superior to strategy one. This finding might suggest and in no way precludes the existence of a still better heuristic strategy. Some thought might be given to developing some rules which incorporate the components of both strategies one and two, i.e., components of the objective and constraints.

- (3) An examination of other problem parameters.

This study examined the effect three problem parameters had on the capacitated quadratic assignment problem. Two of these three parameters were determined to be not significantly related to problem solution difficulty at the levels investigated. Some additional work could look at these same parameters or entirely different ones in different ways.

APPENDIX A

The Solution Procedure Code


```

PROGRAM BRABD(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,
* TAPE1)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
* LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000)
INTEGER VAR,TVAR,SUM,DTSIZE,SIZE,F

C
WRITE(6,999)
999 FORMAT(1H1)

C
C READ IN VALUES FOR SIZE(I),THE INITIAL CONSTRAINTS ON
C CAMPUS SIZES.
C
READ(5,*) ICCHIP
READ(5,700)(SIZE(I),I=1,3)
700 FORMAT(3I10)
WRITE(6,955)
955 FORMAT(1X,32HTHE STARTING POINT IS AS FOLLOWS)
WRITE(6,956)
956 FORMAT(1X,5X,48HCAMPUS SIZE CONSTRAINTS FOR CAMPUSES
* 1,2,3 RESP.)
WRITE(6,700)(SIZE(I),I=1,3)

C
C READ IN VALUES FOR THE NUMBER OF CAMPUSES,NLC, THE
C NUMBER OF DEPARTMENTS,NBD, THE INITIAL ADDRESS FOR
C THE SOLUTION VECTOR,IK, AND THE NUMBER OF DEPTS.
C FIXED AT THE OUTSET,LO.
C
READ(5,705)NLC,NBD,IK,LC,NCT
705 FORMAT(5I10)
IPRED(IK) = -1
IKT=IK

C
C READ THE INITIAL VALUE FOR VAR(IK,NBD*NLC). THIS
C VECTOR
C IS A STRING OF 0,1 VARIABLES. THOSE WITHIN THE FIRST
C NBD PLACES WITH A VALUE OF ONE REPRESENT DEPTS.
C ON CAMPUS 1. SIMILARLY, THOSE IN THE NEXT NBD PLACES
C WITH VALUE OF ONE ARE THOSE DEPTS. PLACED ON CAMPUS
C WHERE A VALUE OF ONE OCCURS, THOSE VALUES IN PLACES
C REPRESENTED AS MULTIPLES OF NBD ADDED OR SUBTRACTED
C FROM THE ADDRESS WITH A ONE MUST BE ZERO - HENCE, A
C DEPT. CANNOT BE PLACED ON MORE THAN ONE CAMPUS.
C
NPROD = NBD*NLC

```

```

READ(5,706) (VAR(J),J=1,NBD)
LF1=NBD+1
LF2=2*NBD
READ(5,706) (VAR(J),J=LF1,LF2)
LF2=LF2+1
LF3=3*NBD
READ(5,706) (VAR(J),J=LF2,LF3)
706 FORMAT(57I1)
CALL PACK(IK)
WRITE(6,957)
957 FORMAT(1X,5X,41HTHE VECTOR STRINGS FOR THE STARTING
* POINT,
*36H ASSIGNMENTS TO CAMPUSES 1,2,3 RESP.)
WRITE(6,707) (VAR(J),J=1,NBD)
LF2=2*NBD
WRITE(6,707) (VAR(J),J=LF1,LF2)
LF2=LF2+1
WRITE(6,707) (VAR(J),J=LF2,LF3)
707 FORMAT(2X,57I1)
C
C READ IN VALUES FOR FLOW(I,J) - THE MATRIX DISPLAYING
C THE FLOWS CONNECTING DEPTS. MATRIX IS ARRANGED WITH
C NON-DIRECTIONAL FLOW TOTALS IN MIRROR IMAGE AROUND A
C ZERO-VALUED MAIN DIAGONAL.
C
DO 720 LN=1,57
READ(5,725) (FLOW(LN,J),J=1,57)
725 FORMAT(3X,12F6.1)
720 CONTINUE
C
C READ IN THE VALUES FOR THE DEPT. SIZES, DTSIZE(I).
C
READ(5,726) (DTSIZE(J),J=1,57)
726 FORMAT(4X,12I6)
C
SVARY(1,1)=SVARY(1,2)=SVARY(1,3)=0.0
DO 401 LN=1,NPROD
J=((LN-1)/NBD)+1
K=MOD((LN-1),NBD)+1
IF(VAR(LN).EQ.1) SVARY(1,J)=SVARY(1,J)+DTSIZE(K)
401 CONTINUE
WRITE(6,991)
991 FORMAT(1X,5X,31HTHE LOADS ON CAMPUSES AT OUTSET)
WRITE(6,728) (SVARY(1,K),K=1,3)
728 FORMAT(4X,3F15.0)
C
READ(5,728) CST12,CST13,CST23
WRITE(6,990)
990 FORMAT(1X,5X,32HTHE COSTS PER UNIT FLOW BETWEEN ,
*26HCAMPUSES 1-2,1-3,2-3 RESP.)

```

```

WRITE (6,728)CST12,CST13,CST23
IF(CST12.LE.CST13)GO TO 420
IF(CST13.LE.CST23)GO TO 422
COST1=CST23
COST2=CST13
GO TO 435
420 IF(CST12.LE.CST23)GO TO 425
COST1=CST23
COST2=CST12
GO TO 435
422 IF(CST12.LE.CST23)GO TO 430
COST1=CST13
COST2=CST23
GO TO 435
425 IF(CST13.LE.CST23)GO TO 429
COST1=CST12
COST2=CST23
GO TO 435
429 COST1=CST12
COST2=CST13
GO TO 435
430 COST1=CST13
COST2=CST12
435 CONTINUE
SENZ=0.0
CALL OPTRACE (SENZ,NLC,NBD,IKT,CST12,CST13,CST23,LO,I,
* UB)
SVARX(1)=UB
ZMIN(IK)=SVARX(1)
WRITE (6,989)SVARX(1)
989 FORMAT(1X,5X,33HAGGREGATE FLOW COSTS AT OUTSET = ,
* F15.2)
C
IF(IDCHIP.EQ.1) CALL SORT(NLC,NBD,IK)
IF(IDCHIP.EQ.2) CALL SORTFX(NLC,NBD,IK)
C
C
SENZ=1.0
WRITE (6,925)LO
CALL SECOND(TIME1)
CALL OPTRACE (SENZ,NLC,NBD,IK,CST12,CST13,CST23,LO,I,
* UB)
CALL SECOND(TIME2)
TIME3=TIME2-TIME1
IUB=UB
C
WRITE (6,731)
731 FORMAT(1X,42HA GOOD FEASIBLE SOLUTION IS PRINTED
* BELOW.)
925 FORMAT(1X,5X,43HTHE NUMBER OF DEPTS. FIXED AT THE
* OUTSET = ,I4)

```

```

WRITE (6,730) SVARX(I), (SVARY(I,K),K=1,3)
730 FORMAT(1X,5X,23HAGGREGATE FLOW COSTS = ,F10.2,/,
*5X,31HCAMPUS LOADS FOR 1,2,3 RESP. = ,3F10.2)
WRITE (6,951)
951 FORMAT(1X,5X,/,19HDEPTS. ON CAMPUS 1.)
WRITE (6,733) (LIST1(I),I=1,19)
WRITE (6,732) (LIST1(I),I=20,38)
WRITE (6,732) (LIST1(I),I=39,57)
WRITE (6,952)
952 FORMAT(1X,5X,/,19HDEPTS. ON CAMPUS 2.)
WRITE (6,733) (LIST2(I),I=1,19)
WRITE (6,732) (LIST2(I),I=20,38)
WRITE (6,732) (LIST2(I),I=39,57)
WRITE (6,953)
953 FORMAT(1X,5X,/,19HDEPTS. ON CAMPUS 3.)
WRITE (6,733) (LIST3(I),I=1,19)
WRITE (6,732) (LIST3(I),I=20,38)
WRITE (6,732) (LIST3(I),I=39,57)
732 FORMAT(5X,19I3)
733 FORMAT(/,5X,19I3)
WRITE (6,652)
652 FORMAT(1X,/,42HTHE ORDER IN WHICH THE DEPTS. ARE
* ASSIGNED,
*14H IS AS FOLLOWS)
WRITE (6,651) (JLDEPT(J),J=1,I)
651 FORMAT(1X,4I3,/,/)
WRITE (6,997) TIME3
997 FORMAT(1X,25HTIME TO RUN CPT-INTCHG = ,F8.4)
C WRITE (6,655)
655 FORMAT(1X,/,39HINTERMEDIATE SOLUTION STAGES NOW
* FOLLOW)
C
740 CALL BRANCH(NLC,NBD,NCT,IK,IL)
C
DO 780 II=1,IL
CALL TEST(NBD,NLC,II,F)
IF(F.EQ.0)GO TO 780
IKT=IKT+1
IPRED(IKT)=IK
DO 735 J=1,NPROD
VAR( J)=TVAR(II,J)
735 CONTINUE
CALL PACK(IKT)
C
C
DO 745 M=1,NCT
LST(M)=0
745 CONTINUE
CALL LBCALC(NLC,NBD,IKT,COST1,COST2,X)
C WRITE (6,750) IKT,X,(VAR(J),J=1,NBD)

```

```

750 FORMAT(5X,6HIKT = ,I5,2X,4HX = ,F5.0,2X,13HVAR(IKT,J)
* = ,57I1)
  LF1=NBD+1
C  WRITE(6,751) (VAR(J),J=LF1,LF3)
751 FORMAT(114I1)
  SENZ=0.0
  CALL OPTRACE(SENZ,NLC,NBD,IKT,CST12,CST13,CST23,LO,I,
* UB)
  IXCT=I
  IF(X.GE.0.0)GO TO 775
  ZMIN(IKT)=UB
  GO TO 780
775 ZMIN(IKT)=UB+X
C  WRITE(6,741)ZMIN(IKT)
780 CONTINUE
741 FORMAT(1X,10X,12HZMIN(IKT) = ,F10.2)
  IF(IKT.LT.NCT)GO TO 781
  CALL CLEAN(IUB,IKT,ICTR)
  IKT=ICTR
781 CONTINUE

C
C  THE BRANCHING DECISION IS TO PROCEED FROM THE
C  VECTOR THAT HAS THE LOWEST LOWER BOUND. IN THIS
C  FOR THAT VECTOR, THE ANTECEDENT NODES FOR EACH SUCH
C  VECTOR MUST BE EXCLUDED SINCE THEY WILL HAVE LOWER
C  BOUNDS. THE VECTOR IPRED(IK) STORES THE IMMEDIATE
C  PREDECESSOR TO THE SOLUTION VECTOR IN VAR(IK,NPROD).
C  THE COLLECTION OF PREDECESSOR NODES FOR A GIVEN
C  VECTOR
C  ARE IN LST(K).
C
  DO 782 I=2,IKT
  LST(I)=I
782 CONTINUE
  DO 790 II=1,IKT
  IJ=IKT+1-II
  LVAL=IPRED(IJ)
785 IF(LVAL.LT.0)GO TO 790
  LST(LVAL)=0
  MM=LVAL
  LVAL=IPRED(MM)
  GO TO 785
790 CONTINUE
  TOPS=500000.
  DO 795 II=1,IKT
  IF(LST(II).EQ.0)GO TO 795
  J=LST(II)
  IF(TOPS-ZMIN(J))795,795,794
794 TOPS=ZMIN(J)
  IK=J

```

```

795 CONTINUE
    LB=ZMIN(IK)
    CALL UNPACK(IK)
C
    DO 800 J=1,NCT
    LST(J)=0
800 CONTINUE
C    WRITE(6,801)LB,IK
801 FORMAT(2X,4HLB = ,I5,2X,4HIK = ,I5)
C
C    THE VALUE IK IS THE IDENTIFIER OF THE BEST NODE
C    FROM WHICH TO BRANCH AT THIS STAGE, ITS CORRES.
C    LOWER BOUND HAS THE VALUE LB.
C
C    IF THE LB = UB, THE OPTIMAL PLACEMENT VECTOR
C    IS IDENTIFIED BY THE VALUE VAR(IK,NPROD). IF
C    LB DOES NOT EQUAL UB, A NEW UPPER BOUND MAY BE
C    GENERATED IF THE OPTRACE RETURN VALUE IMPROVES
C    UPON THE BOUND.
C
    LO = 0
    DO 810 I=1,NPROD
    LO=LO+VAR(I)
810 CONTINUE
811 IF(LO.EQ.NBD.AND.LB.LE.(IUB+1))GO TO 900
C    WRITE(6,930)IUB
930 FORMAT(1X,14HUPPER BOUND = ,I10)
    GO TO 740
C
900 WRITE(6,850)
850 FORMAT(5X,37HTHE OPTIMAL SOLUTION HAS BEEN REACHED)
    WRITE(6,860)LB,(SVARY(IXCT,K),K=1,3)
860 FORMAT(5X,15HCOST OF FLOW = ,I10,/,
*31HCAMPUS LOADS FOR 1,2,3 RESP. = ,3F10.0)
C
    I=0
    DO 890 L=1,NBD
    IF(VAR( L).EQ.0)GO TO 890
    WRITE(6,870) L
870 FORMAT(1H ,10X,4HDEPT,I5,2X,15HIS ON CAMPUS 1.)
890 CONTINUE
    I=0
    LI1=NBD+1
    LI2=2*NBD
    DO 910 L=LI1,LI2
    IF(VAR( L).EQ.0)GO TO 910
    LNBD=L-NBD
    WRITE(6,875)LNBD
875 FORMAT(1H ,10X,4HDEPT,I5,2X,15HIS ON CAMPUS 2.)
910 CONTINUE

```

```

I=0
LI1=2*NBD+1
LI2=3*NBD
DO 920 L=LI1,LI2
IF(VAR(L).EQ.0)GO TO 920
LNBD=L-2*NBD
WRITE(6,880)LNBD
880 FORMAT(1H ,10X,4HDEPT,I5,2X,15HIS ON CAMPUS 3.)
920 CONTINUE
CALL EXIT
END

```

```

SUBROUTINE SORT(NLC,NBD,IK)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
* LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000)
DIMENSION ISTORE(57),ZSUM(57)
INTEGER VAR,TVAR,SUM,DTSIZE,SIZE,F

C
DO 500 J=1,NBC
JLDEPT(J)=0
500 CONTINUE
I=0
N1=NLC*NBD
N2=NLC-1
DO 570 L=1,NBC
I=I+1
DO 510 K=1,N2
K1=K*NBD+L
IF(VAR(L).EQ.VAR(K1))510,502
502 I=I-1
GO TO 570
510 CONTINUE
JLDEPT(I)=L
570 CONTINUE

C
C
ICANT=1
605 MAXV=-9999
JLOC=0

```

```

        ILOC=0
        DO 610 J=1,I
        L=JLDEPT(J)
        IF(L.EQ.0)GO TO 610
        IF(MAXV.GE.DTSIZE(L))GO TO 610
        MAXV=DTSIZE(L)
        JLOC=J
        ILOC=L
610  CONTINUE
C
        ISTORE(ICANT)=ILOC
        JLDEPT(JLOC)=0
        ICANT=ICANT+1
        IF(ICANT.LE.I)GO TO 605
C
        DO 615 J=1,I
        JLDEPT(J)=ISTORE(J)
615  CONTINUE
C
        RETURN
        END

SUBROUTINE CLEAN(IUB,IKT,ICTR)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
*  LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000)
INTEGER VAR
ICTR=1
C
C
        DO 80 J=2,IKT
        DO 40 K=2,IKT
        IF(J.EQ.IPRED(K))GO TO 80
40  CONTINUE
        UB=IUB
        IF((ZMIN(J)-UB).GT.2.0)GO TO 80
        ICTR=ICTR+1
        WRITE(1)(IBUF(J,K),K=1,3),ZMIN(J)
80  CONTINUE
        DO 90 I=2,IKT

```



```

      DO 85 K=1,3
      IBUF(I,K)=0
85  CONTINUE
      IPRED(I)=-1
      ZMIN(I)=0.0
90  CONTINUE
      REWIND1
      DO 100 J=2,ICTR
      READ(1)(IBUF(J,K),K=1,3),ZMIN(J)
100 CONTINUE
      REWIND1
      WRITE(6,95)ICTR
95  FORMAT(10X,31HNO. OF SOLUTIONS TRANSFERRED = ,I5)
      RETURN
      END

```

```

SUBROUTINE UNPACK(JK)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
* LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000)
INTEGER VAR
DATA MASK/18/

```

C

```

      J=0
      DO 4 I=1,171,57
      J=J+1
      IBUF(JK,J)=SHIFT(IBUF(JK,J),3)
      DO 3 K=1,57
      IBUF(JK,J)=SHIFT(IBUF(JK,J),1)
      VAR(I+K-1)=IBUF(JK,J).AND.MASK
3  CONTINUE
4  CONTINUE
      RETURN
      END

```

```

SUBROUTINE PACK(JK)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
* LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000)
INTEGER VAR

```

C

```

J=0
DO 2 I=1,171,57
J=J+1
IBUF(JK,J)=0
DO 1 K=1,57
IBUF(JK,J)=SHIFT(IBUF(JK,J),1)
IBUF(JK,J)=IBUF(JK,J).OR.VAR(I+K-1)
1 CONTINUE
2 CONTINUE
RETURN
END

```

```

SUBROUTINE BRANCH(NLC,NBD,NCT,IK,IZ)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
* LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000)
INTEGER VAR,TVAR,SUM,DTSIZE,SIZE,F

```

C

```
N1=NLC*NBD
```

C

```

DO 580 IL=1,NLC
DO 581 K2=1,N1
TVAR(IL,K2)=VAR(K2)
581 CONTINUE
580 CONTINUE

```

C

```

DO 585 J=1,NBC
IF(JLDEPT(J).EQ.0)GO TO 585
JM=JLDEPT(J)

```

```

      N2=NLC-1
      DO 583 K1=1,N2
      K1M=K1*NBD+JM
      IF((VAR(JM)+VAR(K1M)).EQ.1)GO TO 584
583  CONTINUE
      GO TO 590
584  CONTINUE
585  CONTINUE
590  JLOC=J
C
      DO 550 IX=1,NLC
      N2=IX-1
      I14=JLDEPT(JLOC) + N2*NBD
      TVAR(IX,I14)=1
550  CONTINUE
C
      IZ=NLC
      RETURN
      END

SUBROUTINE TEST(NBD,NLC,IKT,F)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
* LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000)
INTEGER VAR,TVAR,SUM,DTSIZE,SIZE,F
C
C
      DO 350 L=1,NLC
      KL = (L-1)*NBD + 1
      KJ = L*NBD
      SUM=0
      DO 310 I=KL,KJ
      IF(I.LE.NBD)GO TO 16
      J=I-(L-1)*NBD
      GO TO 17
16  J=I
17  SUM = SUM + TVAR(IKT,I)*DTSIZE(J)
310 CONTINUE
      IF(SUM.LE.SIZE(L))GO TO 348
      F=0

```

```

      GO TO 360
348 F=1
350 CONTINUE
360 CONTINUE
      RETURN
      END

```

```

SUBROUTINE LBCALC(NLC,NBD,IKT,COST1,COST2,Z)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
* LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000)
INTEGER VAR,TVAR,SUM,DTSIZE,SIZE,F

```

C
C

```

      IXCT=0
      Z=0.0
      DO 410 J=1,NBC
      IF(JLDEPT(J).EQ.0)GO TO 410
      JM=JLDEPT(J)
      N2=NLC-1
      DO 405 K1=1,N2
      K1M=K1*NBD+JM
      IF((VAR(JM)+VAR(K1M)).EQ.1)GO TO 408
405 CONTINUE
      GO TO 411
408 CONTINUE
410 CONTINUE
      J=NBD
411 LOC=J

```

C

```

      I=0
      DO 413 J=LOC,NBD
      IF(JLDEPT(J).EQ.0)GO TO 413
      I=I+1
413 CONTINUE

```

C

```

      IF(I.EQ.0)GO TO 575

```

C

C

```

      IF((I.AND.18).EQ.1)GO TO 414

```

```

      GO TO 416
414  IF((LOC.AND.18).EQ.1)GO TO 417
      GO TO 909
415  IXCT=1
909  N=LOC+1
      NN=I+LOC-2
      LL=JLDEPT(LOC)
      GO TO 525
419  N=LOC+2
      NN=I+LOC-2
      IXCT=1
      LL=JLDEPT(LOC)
      GO TO 525
416  IF((LOC.AND.18).EQ.0) GO TO 415
      IF(JLDEPT(LOC+1).EQ.0)GO TO 419
      N=LOC
      NN=I+LOC-1
      GO TO 418
417  IF(JLDEPT(LOC+1).EQ.0)GO TO 421
      N=LOC
      NN=I+LOC-2
      LL=JLDEPT(I+LOC-1)
      GO TO 525
421  N=LOC+2
      NN=I+LOC-1
      LL=JLDEPT(LOC)
      GO TO 525
525  SMIN=0.0
      DO 527 J=1,NBD
      SMIN=SMIN + VAR(J)*FLOW(J,LL)
527  CONTINUE
      RMIN=0.0
      J1=NBD+1
      J2=2*NBD
      DO 529 J=J1,J2
      RMIN=RMIN + VAR(J)*FLOW(J-NBD,LL)
529  CONTINUE
      TMIN=0.0
      J1=2*NBD+1
      J2=3*NBD
      DO 531 J=J1,J2
      TMIN=TMIN + VAR(J)*FLOW(J-2*NBD,LL)
531  CONTINUE
      IF(TMIN.LE.RMIN)GO TO 532
      IF(RMIN.LE.SMIN)GO TO 533
      X=COST2*SMIN+COST1*RMIN
      GO TO 550
C
532  IF(TMIN.LE.SMIN)GO TO 535
      X=COST2*SMIN+COST1*TMIN

```

```

      GO TO 550
C
533 IF(SMIN.LE.TMIN)GO TO 534
      X=COST2*RMIN+COST1*TMIN
      GO TO 550
C
534 X=COST2*RMIN+COST1*SMIN
      GO TO 550
C
535 IF(SMIN.LE.RMIN)GO TO 536
      X=COST2*TMIN+COST1*RMIN
      GO TO 550
C
536 X=COST2*TMIN+COST1*SMIN
550 Z=Z+X
      IF(IXCT.NE.1)GO TO 548
      IXCT=0
      LL=JLDEPT(I+LOC)
      GO TO 525
548 CONTINUE
C
418 CONTINUE
      DO 520 M=N,NN,2
      LL=JLDEPT(M)
      LL1=JLDEPT(M+1)
      SMIN=0.0
      SMIN1=0.0
      DO 420 J=1,NBC
      SMIN=SMIN + VAR(J)*FLOW(J,LL)
      SMIN1=SMIN1 + VAR(J)*FLOW(J,LL1)
420 CONTINUE
      RMIN=0.0
      RMIN1=0.0
      J1=NBD+1
      J2=2*NBD
      DO 422 J=J1,J2
      RMIN=RMIN + VAR(J)*FLOW(J-NBD,LL)
      RMIN1=RMIN1 + VAR(J)*FLOW(J-NBD,LL1)
422 CONTINUE
      TMIN=0.0
      TMIN1=0.0
      J1=2*NBD+1
      J2=3*NBD
      DO 424 J=J1,J2
      TMIN=TMIN + VAR(J)*FLOW(J-2*NBD,LL)
      TMIN1=TMIN1 + VAR(J)*FLOW(J-2*NBD,LL1)
424 CONTINUE
C
      IF(TMIN.LE.RMIN)GO TO 430
      IF(RMIN.LE.SMIN)GO TO 435

```

```
      X=COST2*SMIN+COST1*RMIN
      ICMP=3
      GO TO 460
C
430 IF(TMIN.LE.SMIN)GO TO 445
      X=COST2*SMIN+COST1*TMIN
      ICMP=2
      GO TO 460
C
435 IF(SMIN.LE.TMIN)GO TO 440
      X=COST2*RMIN+COST1*TMIN
      ICMP=1
      GO TO 460
C
440 X=COST2*RMIN+COST1*SMIN
      ICMP=3
      GO TO 460
C
445 IF(SMIN.LE.FMIN)GO TO 450
      X=COST2*TMIN+COST1*RMIN
      ICMP=1
      GO TO 460
C
450 X=COST2*TMIN+COST1*SMIN
      ICMP=2
      GO TO 460
C
460 IF(TMIN1.LE.RMIN1)GO TO 470
      IF(RMIN1.LE.SMIN1)GO TO 475
      Y=COST2*SMIN1+COST1*RMIN1
      JCMP=3
      GO TO 495
C
470 IF(TMIN1.LE.SMIN1)GO TO 485
      Y=COST2*SMIN1+ COST1*TMIN1
      JCMP=2
      GO TO 495
C
475 IF(SMIN1.LE.TMIN1)GO TO 480
      Y=COST2*RMIN1+COST1*TMIN1
      JCMP=1
      GO TO 495
C
480 Y=COST2*RMIN1+COST1*SMIN1
      JCMP=3
      GO TO 495
C
485 IF(SMIN1.LE.RMIN1)GO TO 490
      Y=COST2*TMIN1+COST1*RMIN1
      JCMP=1
```

```

GO TO 495
C
490 Y=COST2*TMIN1+COST1*SMIN1
    JCMP=2
C
495 IF(JCMP.EQ.ICMP)GO TO 500
C
C
    Z=Z+X+Y+COST1*FLOW(LL,LL1)
    A=COST2*(SMIN+SMIN1)+COST1*(RMIN+RMIN1)
    D=COST1*(SMIN+SMIN1)+COST2*(RMIN+RMIN1)
    B=COST2*(SMIN+SMIN1)+COST1*(TMIN+TMIN1)
    E=COST1*(SMIN+SMIN1)+COST2*(TMIN+TMIN1)
    C=COST2*(RMIN+RMIN1)+COST1*(TMIN+TMIN1)
    F1=COST1*(RMIN+RMIN1)+COST2*(TMIN+TMIN1)
    XZ=AMIN1(A,B,C,D,E,F1)
    IF(XZ.EQ.0)GO TO 520
    IF((X+Y+COST1*FLOW(LL,LL1)).LE.XZ)GO TO 520
    Z=Z-(X+Y+COST1*FLOW(LL,LL1))+XZ
    GO TO 520
500 Z=Z+X+Y
520 CONTINUE
    GO TO 580
C
C
C
575 Z=-1.0
580 CONTINUE
    RETURN
    END

SUBROUTINE OPTRACE(SENZ,NLC,NBD,IK,CST12,CST13,CST23,
* LO,I,UB)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
* LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000)
INTEGER VAR,TVAR,SUM,DTSIZE,SIZE,F
DO 3 I=1,NBD
SVARX(I) = 0.0
DO 4 J=1,NLC

```



```

      SVARY(I,J)=0.0
4  CONTINUE
      DO 3 J=1,NBD
      DVARU(I,J)=0.0
      DO 3 K=1,NLC
      DVARV(I,J,K)=0.0
3  CONTINUE
      DO 8 L=1,NBD
      LIST1(L)=LIST2(L)=LIST3(L)=0
8  CONTINUE

C
C      TO OBTAIN J1,J2,AND J3 AS WELL AS LIST1,
C      LIST2 AND LIST3 FROM VAR(IK,NBD*NLC)
C

      J1=J2=J3=0
      DO 9 L=1,NBD
      J1=J1 + VAR(    L)
      J2 = J2 + VAR(    NBD+L)
      J3 = J3 + VAR(    2*NBD+L)
9  CONTINUE

C

      I=0
      DO 10 L=1,NBD
      IF(VAR(    L).EQ.0.0)GO TO 10
      I=I+1
      LIST1(I)=L
10 CONTINUE
      I=0
      LI1=NBD+1
      LI2=2*NBD
      DO 11 L=LI1,LI2
      IF(VAR(    L).EQ.0.0) GO TO 11
      I=I+1
      LIST2(I) = L - NBD
11 CONTINUE
      I=0
      LI1=2*NBD+1
      LI2=3*NBD
      DO 12 L=LI1,LI2
      IF(VAR(    L).EQ.0.0)GO TO 12
      I=I+1
      LIST3(I)=L-2*NBD
12 CONTINUE
805 ZSUM=0.0
      ICOUNT = L0
      I=ICOUNT
      SVARY(I,1)=SVARY(I,2)=SVARY(I,3)=0.0
      DO 810 JJ=1,J1
      L1=LIST1(JJ)
      SVARY(I,1)=SVARY(I,1)+DTSIZE(L1)

```

```

      DO 810 JK=1,J2
      L2=LIST2(JK)
      ZSUM=ZSUM + CST12*FLOW(L1,L2)
810  CONTINUE
      DO 820 JJ=1,J1
      L1=LIST1(JJ)
      DO 820 JL=1,J3
      L3=LIST3(JL)
      ZSUM=ZSUM + CST13*FLOW(L1,L3)
820  CONTINUE
      DO 830 JK=1,J2
      L2=LIST2(JK)
      SVARY(I,2)=SVARY(I,2)+DTSIZE(L2)
      DO 830 JL=1,J3
      L3=LIST3(JL)
      ZSUM=ZSUM + CST23*FLOW(L2,L3)
830  CONTINUE
      DO 840 JL=1,J3
      L3=LIST3(JL)
      SVARY(I,3)=SVARY(I,3)+DTSIZE(L3)
840  CONTINUE
      SVARX(I) = ZSUM
      IF(SENZ.EQ.0.0.OR.I.EQ.NBD)GO TO 845

```

C
C
C
C
C
C

TO OBTAIN THE ELIGIBLE DEPT. LIST FROM
VAR(IK,NBD*NLC)

```

      DO 15 I=1,NBD
      ILDEPT(I)=0
15  CONTINUE
      I=0
      N2=NLC-1
13  DO 19 L=1,NBD
      I=I+1
      DO 18 K=1,N2
      K1=K*NBD+L
      IF(VAR( L).EQ.VAR( K1))18,7
7   I=I-1
      GO TO 19
18  CONTINUE
      ILDEPT(I)=L
      ICTR=I
19  CONTINUE
      ICOUNT = ICOUNT + 1
      N100=NBD+1
30  IF(ICOUNT.EQ.N100)GO TO 800

```

C

I = ICOUNT

```

DO 65 J=1,ICTR
IF(ILDEPT(J).EQ.0) GO TO 65
ID = ILDEPT(J)
C
C
31 K=1
C
C THIS SECTION TESTS FOR FEASIBLE PLACEMENT OF A
C DEPARTMENT ON A CAMPUS. THE -100 VALUE FOR THE
C DEPTS. CONTRIBUTION TO INTER-CAMPUS FLOW IS AN
C ARBITRARY INDICATOR OF INFEASIBILITY.
C
DVARU(I,ID) = DTSIZE(ID)
38 T = SIZE(K) - SVARY(I-1,K)
IF(DVARU(I,ID).LE.T)GO TO 40
DVARV(I,ID,K) = -100.0
K = K + 1
GO TO 62
C
C IF THIS IS A FEASIBLE PLACEMENT, THEN THE
C CONTRIBUTION TO FLOWS MUST BE CALCULATED.
C
40 IF(K.EQ.1)GO TO 45
IF(K.EQ.2)GO TO 50
IF(K.EQ.3)GO TO 55
GO TO 60
45 DVARV(I,ID,1) = 0.0
DO 46 JJ=1,J2
L2 = LIST2(JJ)
DVARV(I,ID,1) = DVARV(I,ID,1) + CST12*FLOW(ID,L2)
46 CONTINUE
DO 47 JJ=1,J3
L3 = LIST3(JJ)
DVARV(I,ID,1) = DVARV(I,ID,1) + CST13*FLOW(ID,L3)
47 CONTINUE
GO TO 60
C
50 DVARV(I,ID,2) = 0.0
DO 51 JJ=1,J1
L1 = LIST1(JJ)
DVARV(I,ID,2) = DVARV(I,ID,2) + CST12*FLOW(ID,L1)
51 CONTINUE
DO 52 JJ=1,J3
L3 = LIST3(JJ)
DVARV(I,ID,2) = DVARV(I,ID,2) + CST23*FLOW(ID,L3)
52 CONTINUE
GO TO 60
C
55 DVARV(I,ID,3) = 0.0
DO 56 JJ=1,J1

```

```

L1 = LIST1(JJ)
DVARV(I, ID, 3) = DVARV(I, ID, 3) + CST13*FLOW(ID, L1)
56 CONTINUE
DO 57 JJ=1, J2
L2 = LIST2(JJ)
DVARV(I, ID, 3) = DVARV(I, ID, 3) + CST23*FLOW(ID, L2)
57 CONTINUE
60 CONTINUE

```

C
C
C
C

IF THE POSSIBLE PLACEMENTS OF A DEPT. ON A CAMPUS
ARE NOT EXHAUSTED, THEN THE CYCLE MUST CONTINUE.

```

K = K + 1
62 IF(K.LE.3)GO TO 38
65 CONTINUE

```

C
C
C
C

THE MINIMUM CONTRIBUTION TO FLOW CORRESPONDING
TO A DEPT. PLACEMENT MUST NOW BE FOUND.

```

XMIN = 100000.0
DO 75 J=1, 57
DO 75 K=1, 3
ID = ILOEPT(J)
IF(ID.EQ.0)GO TO 75
IF(DVARV(I, ID, K).LT.0.0)GO TO 75
IF(XMIN.LT.0.0)GO TO 75
IF(XMIN - DVARV(I, ID, K))75, 75, 74
74 XMIN = DVARV(I, ID, K)
IDC = ID
JC = J
KC = K
75 CONTINUE

```

C
C
C
C

INTERNAL RECORD-KEEPING ON THE ACTUAL BEST
PLACEMENT AT THIS STAGE MUST BE DONE.

```

GO TO(77, 78, 79)KC
77 LIST1(J1+1) = IDC
J1 = J1 + 1
GO TO 80
78 LIST2(J2+1) = IDC
J2 = J2 + 1
GO TO 80
79 LIST3(J3+1) = IDC
J3 = J3 + 1
80 CONTINUE

```

C

```

SVARX(I)=SVARX(I-1)+XMIN
DO 90 KK=1, 3
IF(KK.EQ.KC)GO TO 89

```

```
      SVARY(I, KK) = SVARY(I-1, KK)
      GO TO 90
89  SVARY(I, KC) = SVARY(I-1, KC) + OVARU(I, IDC)
90  CONTINUE
C
      ILDEPT(JC) = 0
      ICOUNT = ICOUNT + 1
      GO TO 30
C
800 CONTINUE
      I=ICOUNT - 1
      UB = SVARX(I)
C
C
      DO 390 ITER=1,5
      NPROD=NBD*NLC
      DO 500 J=1, NPROD
      IVAR(J)=0
500 CONTINUE
C
      DO 128 J=1, J1
      L=LIST1(J)
      ILIST1(J)=LIST1(J)
      DO 125 K=1, NBC
      IF(L.EQ.JLDEPT(K))GO TO 128
125 CONTINUE
      LIST1(J)=0
128 CONTINUE
      DO 129 J=1, J2
      L=LIST2(J)
      ILIST2(J)=LIST2(J)
      DO 126 K=1, NBC
      IF(L.EQ.JLDEPT(K))GO TO 129
126 CONTINUE
      LIST2(J)=0
129 CONTINUE
      DO 130 J=1, J3
      L=LIST3(J)
      ILIST3(J)=LIST3(J)
      DO 127 K=1, NBC
      IF(L.EQ.JLDEPT(K))GO TO 130
127 CONTINUE
      LIST3(J)=0
130 CONTINUE
C
      DO 225 J=1, J1
      B=1.0
      C=1.0
      L=LIST1(J)
      IF(L.EQ.0)GO TO 225
```

```

ASUM=BSUM=CSUM=0.0
IF(DTSIZE(L).GT.(SIZE(2)-SVARY(I,2)))B=0.0
IF(DTSIZE(L).GT.(SIZE(3)-SVARY(I,3)))C=0.0
DO 200 J10=1,J1
LM=ILIST1(J10)
IF(LM.EQ.0)GO TO 200
IF(LM.EQ.L)GO TO 200
ASUM=ASUM+FLOW(LM,L)
200 CONTINUE
DO 210 K10=1,J2
LM=ILIST2(K10)
IF(LM.EQ.0)GO TO 210
BSUM=BSUM+FLOW(LM,L)
210 CONTINUE
DO 215 L10=1,J3
LM=ILIST3(L10)
IF(LM.EQ.0)GO TO 215
CSUM=CSUM+FLOW(LM,L)
215 CONTINUE
X1=CST12*BSUM+CST13*CSUM
IF(B.EQ.0.0)GO TO 211
Y=CST12*ASUM+CST23*CSUM
IF(C.EQ.0.0)GO TO 212
214 Z=CST13*ASUM+CST23*BSUM
GO TO 216
211 Y=100000.0
IF(C.NE.0.0)GO TO 214
212 Z=100000.0
216 X=AMIN1(X1,Y,Z)
IF(X.EQ.X1)GO TO 225
IF(X.EQ.Y)GO TO 220
ILIST1(J)=LIST1(J)=0
SVARY(I,1)=SVARY(I,1)-DTSIZE(L)
J3=J3+1
ILIST3(J3)=LIST3(J3)=L
SVARY(I,3)=SVARY(I,3)+DTSIZE(L)
GO TO 225
220 ILIST1(J)=LIST1(J)=0
SVARY(I,1)=SVARY(I,1)-DTSIZE(L)
J2=J2+1
ILIST2(J2)=LIST2(J2)=L
SVARY(I,2)=SVARY(I,2)+DTSIZE(L)
225 CONTINUE
C
DO 250 J=1,J2
A=1.0
C=1.0
L=LIST2(J)
IF(L.EQ.0)GO TO 250
ASUM=BSUM=CSUM=0.0

```

```

IF(DTSIZE(L).GT.(SIZE(1)-SVARY(I,1)))A=0.0
IF(DTSIZE(L).GT.(SIZE(3)-SVARY(I,3)))C=0.0
DO 230 J10=1,J1
LM=ILIST1(J10)
IF(LM.EQ.0)GO TO 230
ASUM=ASUM+FLOW(LM,L)
230 CONTINUE
DO 235 K10=1,J2
LM=ILIST2(K10)
IF(LM.EQ.0)GO TO 235
IF(LM.EQ.L)GO TO 235
BSUM=BSUM+FLOW(LM,L)
235 CONTINUE
DO 240 L10=1,J3
LM=ILIST3(L10)
IF(LM.EQ.0)GO TO 240
CSUM=CSUM+FLOW(LM,L)
240 CONTINUE
X=AMAX1(ASUM,BSUM,CSUM)
X1=CST12*ASUM+CST23*CSUM
IF(A.EQ.0.0)GO TO 231
Y=CST12*BSUM+CST13*CSUM
IF(C.EQ.0.0)GO TO 232
229 Z=CST13*ASUM+CST23*BSUM
GO TO 241
231 Y=100000.
IF(C.NE.0.0)GO TO 229
232 Z=100000.
241 X=AMIN1(X1,Y,Z)
IF(X.EQ.X1)GO TO 250
IF(X.EQ.Y)GO TO 245
ILIST2(J)=LIST2(J)=0
SVARY(I,2)=SVARY(I,2)-DTSIZE(L)
J3=J3+1
ILIST3(J3)=LIST3(J3)=L
SVARY(I,3)=SVARY(I,3)+DTSIZE(L)
GO TO 250
245 ILIST2(J)=LIST2(J)=0
SVARY(I,2)=SVARY(I,2)-DTSIZE(L)
J1=J1+1
ILIST1(J1)=LIST1(J1)=L
SVARY(I,1)=SVARY(I,1)+DTSIZE(L)
250 CONTINUE
C
DO 280 J=1,J3
A=B=1.0
L=LIST3(J)
IF(L.EQ.0)GO TO 280
ASUM=BSUM=CSUM=0.0
IF(DTSIZE(L).GT.(SIZE(1)-SVARY(I,1)))A=0.0

```

```

IF(DTSIZE(L).GT.(SIZE(2)-SVARY(I,2)))B=0.0
DO 255 J10=1,J1
LM=ILIST1(J10)
IF(LM.EQ.0)GO TO 255
ASUM=ASUM+FLOW(LM,L)
255 CONTINUE
DO 260 K10=1,J2
LM=ILIST2(K10)
IF(LM.EQ.0)GO TO 260
BSUM=BSUM+FLOW(LM,L)
260 CONTINUE
DO 265 L10=1,J3
LM=ILIST3(L10)
IF(LM.EQ.0)GO TO 265
IF(LM.EQ.L)GO TO 265
CSUM=CSUM+FLOW(LM,L)
265 CONTINUE
X1=CST13*ASUM+CST23*BSUM
IF(A.EQ.0.0)GO TO 256
Y=CST13*CSUM+CST12*BSUM
IF(B.EQ.0.0)GO TO 261
257 Z=CST12*ASUM+CST23*CSUM
GO TO 262
256 Y=100000.
IF(B.NE.0.0)GO TO 257
261 Z=100000.
262 X=AMIN1(X1,Y,Z)
IF(X.EQ.X1)GO TO 280
IF(X.EQ.Z)GO TO 270
ILIST3(J)=LIST3(J)=0
SVARY(I,3)=SVARY(I,3)-DTSIZE(L)
J1=J1+1
ILIST1(J1)=LIST1(J1)=L
SVARY(I,1)=SVARY(I,1)+DTSIZE(L)
GO TO 280
270 ILIST3(J)=LIST3(J)=0
SVARY(I,3)=SVARY(I,3)-DTSIZE(L)
J2=J2+1
ILIST2(J2)=LIST2(J2)=L
SVARY(I,2)=SVARY(I,2)+DTSIZE(L)
280 CONTINUE
C
DO 325 J=1,J1
JC=-1
L=LIST1(J)
IF(L.EQ.0)GO TO 325
ASUM=BSUM=CSUM=DSUM=ESUM=FSUM=0.0
N=1
305 DO 310 K=N,J2
LL=LIST2(K)

```



```

IF(LL.EQ.0)GO TO 310
IF(DTSIZE(L).GT.(SIZE(2)-SVARY(I,2)+DTSIZE(LL)))GO TO
* 310
IF(DTSIZE(LL).GT.(SIZE(1)-SVARY(I,1)+DTSIZE(L)))GO TO
* 310
JC=LL
KC=K
GO TO 311
310 CONTINUE
GO TO 325
311 CONTINUE
IF(JC.EQ.-1)GO TO 325
DO 315 JK=1,J1
JL=ILIST1(JK)
IF(JL.EQ.0.OR.JL.EQ.L)GO TO 315
ASUM=ASUM+FLOW(JL,L)
CSUM=CSUM+FLOW(JL,JC)
315 CONTINUE
DO 316 JK=1,J3
JL=ILIST3(JK)
IF(JL.EQ.0)GO TO 316
ESUM=ESUM + FLOW(JL,L)
FSUM=FSUM + FLOW(JL,JC)
316 CONTINUE
DO 320 K=1,J2
LM=ILIST2(K)
IF(LM.EQ.0.OR.LM.EQ.JC)GO TO 320
BSUM=BSUM+FLOW(LM,JC)
DSUM=DSUM+FLOW(LM,L)
320 CONTINUE
IF((CST13*ESUM+CST23*FSUM+CST12*(CSUM+DSUM)).GT.
*(CST13*FSUM+CST23*ESUM+CST12*(ASUM+BSUM)))GO TO 321
IF(KC.EQ.J2)GO TO 325
N=KC+1
ASUM=BSUM=CSUM=DSUM=ESUM=FSUM=0.0
JC=-1
IF(N.GT.J2)GO TO 325
GO TO 305
321 ILIST1(J)=LIST1(J)=0
J1=J1+1
ILIST1(J1)=LIST1(J1)=JC
SVARY(I,1)=SVARY(I,1)-DTSIZE(L)+DTSIZE(JC)
ILIST2(KC)=LIST2(KC)=0
J2=J2+1
ILIST2(J2)=LIST2(J2)=L
SVARY(I,2)=SVARY(I,2)-DTSIZE(JC)+DTSIZE(L)
325 CONTINUE
C
DO 350 J=1,J2
JC=-1

```

```

L=LIST2(J)
IF(L.EQ.0)GO TO 350
ASUM=BSUM=CSUM=DSUM=ESUM=FSUM=0.0
N=1
330 DO 335 K=N,J3
LL=LIST3(K)
IF(LL.EQ.0)GO TO 335
IF(DTSIZE(L).GT.(SIZE(3)-SVARY(I,3)+DTSIZE(LL)))GO TO
* 335
IF(DTSIZE(LL).GT.(SIZE(2)-SVARY(I,2)+DTSIZE(L)))GO TO
* 335
JC=LL
KC=K
GO TO 336
335 CONTINUE
GO TO 350
336 CONTINUE
IF(JC.EQ.-1)GO TO 350
DO 337 JK=1,J1
JL=ILIST1(JK)
IF(JL.EQ.0)GO TO 337
ESUM=ESUM + FLOW(JL,L)
FSUM=FSUM + FLOW(JL,JC)
337 CONTINUE
DO 340 JK=1,J2
JL=ILIST2(JK)
IF(JL.EQ.0.OR.JL.EQ.L)GO TO 340
ASUM=ASUM+FLOW(JL,L)
CSUM=CSUM+FLOW(JL,JC)
340 CONTINUE
DO 345 K=1,J3
LM=ILIST3(K)
IF(LM.EQ.0.OR.LM.EQ.JC)GO TO 345
BSUM=BSUM+FLOW(LM,JC)
DSUM=DSUM+FLOW(LM,L)
345 CONTINUE
IF((CST12*ESUM+CST13*FSUM+CST23*(CSUM+DSUM)).GT.
*(CST12*FSUM+CST13*ESUM+CST23*(ASUM+BSUM)))GO TO 346
IF(KC.EQ.J3)GO TO 350
N=KC+1
ASUM=BSUM=CSUM=DSUM=ESUM=FSUM=0.0
JC=-1
IF(N.GT.J3)GO TO 350
GO TO 330
346 ILIST2(J)=LIST2(J)=0
J2=J2+1
ILIST2(J2)=LIST2(J2)=JC
SVARY(I,2)=SVARY(I,2)-DTSIZE(L)+DTSIZE(JC)
ILIST3(KC)=LIST3(KC)=0
J3=J3+1

```

```

        ILIST3(J3)=LIST3(J3)=L
        SVARY(I,3)=SVARY(I,3)-DTSIZE(JC)+DTSIZE(L)
350  CONTINUE
C
        DO 380 J=1,J3
        JC=-1
        L=LIST3(J)
        IF(L.EQ.0)GO TO 380
        ASUM=BSUM=CSUM=DSUM=ESUM=FSUM=0.0
        N=1
355  DO 356 K=N,J1
        LL=LIST1(K)
        IF(LL.EQ.0)GO TO 356
        IF(DTSIZE(L).GT.(SIZE(1)-SVARY(I,1)+DTSIZE(LL)))GO TO
* 356
        IF(DTSIZE(LL).GT.(SIZE(3)-SVARY(I,3)+DTSIZE(L)))GO TO
* 356
        JC=LL
        KC=K
        GO TO 357
356  CONTINUE
        GO TO 380
357  CONTINUE
        IF(JC.EQ.-1)GO TO 380
        DO 360 JK=1,J3
        JL=ILIST3(JK)
        IF(JL.EQ.0.OR.JL.EQ.L)GO TO 360
        ASUM=ASUM+FLOW(JL,L)
        CSUM=CSUM+FLOW(JL,JC)
360  CONTINUE
        DO 361 K=1,J2
        JL=ILIST2(K)
        IF(JL.EQ.0)GO TO 361
        ESUM=ESUM + FLOW(JL,L)
        FSUM=FSUM + FLOW(JL,JC)
361  CONTINUE
        DO 365 K=1,J1
        LM=ILIST1(K)
        IF(LM.EQ.0.OR.LM.EQ.JC)GO TO 365
        BSUM=BSUM+FLOW(LM,JC)
        DSUM=DSUM+FLOW(LM,L)
365  CONTINUE
        IF((CST23*ESUM+CST12*FSUM+CST13*(CSUM+DSUM)).GT.
* (CST23*FSUM+CST12*ESUM+CST13*(ASUM+BSUM)))GO TO 370
        IF(KC.EQ.J1)GO TO 380
        N=KC+1
        ASUM=BSUM=CSUM=DSUM=ESUM=FSUM=0.0
        JC=-1
        IF(N.GT.J1)GO TO 380
        GO TO 355

```

```

370 ILIST3(J)=LIST3(J)=0
    J3=J3+1
    ILIST3(J3)=LIST3(J3)=JC
    SVARY(I,3)=SVARY(I,3)-OTSIZE(L)+OTSIZE(JC)
    ILIST1(KC)=LIST1(KC)=0
    J1=J1+1
    ILIST1(J1)=LIST1(J1)=L
    SVARY(I,1)=SVARY(I,1)-OTSIZE(JC)+OTSIZE(L)
380 CONTINUE
C
    DO 400 J=1,J1
    L=ILIST1(J)
    IF(L.EQ.0)GO TO 400
    IVAR(L)=1
400 CONTINUE
    DO 401 J=1,J2
    L=ILIST2(J)
    IF(L.EQ.0)GO TO 401
    IVAR(NBD+L)=1
401 CONTINUE
    DO 402 J=1,J3
    L=ILIST3(J)
    IF(L.EQ.0)GO TO 402
    IVAR(2*NBD+L)=1
402 CONTINUE
    DO 405 L=1,NBC
    LIST1(L)=LIST2(L)=LIST3(L)=0
    ILIST1(L)=ILIST2(L)=ILIST3(L)=0
405 CONTINUE
    J1=J2=J3=0
    DO 406 L=1,NBC
    J1=J1+IVAR(L)
    J2=J2+IVAR(NBD+L)
    J3=J3+IVAR(2*NBD+L)
406 CONTINUE
C
    J=0
    DO 410 L=1,NBC
    IF(IVAR(L).EQ.0)GO TO 410
    J=J+1
    ILIST1(J)=LIST1(J)=L
410 CONTINUE
    J=0
    LI1=NBD+1
    LI2=2*NBD
    DO 411 L=LI1,LI2
    IF(IVAR(L).EQ.0)GO TO 411
    J=J+1
    ILIST2(J)=LIST2(J)=L-NBD
411 CONTINUE

```

```

      J=0
      LI1=2*NBD+1
      LI2=3*NBD
      DO 412 L=LI1,LI2
      IF(IVAR(L).EQ.0)GO TO 412
      J=J+1
      ILIST3(J)=LIST3(J)=L-2*NBD
412 CONTINUE
C
390 CONTINUE
   SENZ=0.0
   LO=NBD
   GO TO 805
C
845 CONTINUE
   UB=ZSUM
850 CONTINUE
   RETURN
   END

```

```

SUBROUTINE SORTFX(NLC,NBD,IK)
COMMON ZMIN(2000),IPRED(2000),IBUF(2000,3)
COMMON TVAR(5,171),SVARY(57,5),JLDEPT(57)
COMMON SVARX(57),VAR(171),LIST1(57),LIST2(57),
* LIST3(57)
COMMON SIZE(3),DTSIZE(57),FLOW(57,57)
COMMON ILDEPT(57),DVARU(57,57),DVARV(57,57,3)
COMMON IVAR(171),ILIST1(57),ILIST2(57),ILIST3(57)
COMMON LST(2000),SUMFLO(57),ISTORE(57)
DIMENSION ISTORE(57),ZSUM(57)
INTEGER VAR,TVAR,SUM,DTSIZE,SIZE,F
C
   DIMENSION SUMFLO(57),ISTORE(57)
C *****
C *****
   DO 300 J=1,NBD
   JLDEPT(J)=0
   ISTORE(J)=0
300 CONTINUE
C *****
C THIS ISTORES AND ORDERS BY FLOW WRT FIXED VAR.
C *****
   I=0
   N1=NLC*NBD
   N2=NLC-1

```

```

DO 370 L=1,NBC
I=I+1
DO 310 K=1,N2
K1=K*NBD+L
IF(VAR(L).EQ.VAR(K1))310,302
302 I=I-1
GO TO 370
310 CONTINUE
JLDEPT(I)=L
IGRAB=I
370 CONTINUE
14 FORMAT(5X,1I3)
WRITE(6,14) IGRAB
C *****
JC=0
DO 800 I=1,57
IF(JLDEPT(I).EQ.0) GO TO 750
JC=JC+1
ISTORE(JC)=JLDEPT(I)
750 CONTINUE
800 CONTINUE
DO 820 I=1,IGRAB
815 FORMAT(5X,1I2)
WRITE(6,815) ISTORE(I)
820 CONTINUE
DO 450 J=1,IGRAB
SUMFLC(J)=0.
KI=ISTORE(J)
DO 410 IN=1,3
DO 400 K=1,57
SUM=VAR(K*IN)*FLOW(KI,K)
SUMFLC(J)=SUMFLO(J)+SUM
400 CONTINUE
410 CONTINUE
450 CONTINUE
DO 460 I=1,IGRAB
455 FORMAT(5X,1F10.2)
WRITE(6,455) SUMFLO(I)
460 CONTINUE
KA=0
DO 580 II=1,IGRAB
MAX=-5
DO 550 I=1,IGRAB
IF(KA.EQ.0) GO TO 530
DO 520 J=1,KA
560 IF(JLDEPT(J).EQ.ISTORE(I)) GO TO 550
520 CONTINUE
530 CONTINUE
IF(SUMFLO(I).GT.MAX) GO TO 540
CONTINUE

```

```
GO TO 550
540 MAX=SUMFLO(I)
    LI=I
550 CONTINUE
    KA=KA+1
    JLDEPT(KA)=ISTORE(LI)
580 CONTINUE
    DO 600 I=1,IGRAB
590 FORMAT(5X,1I2)
    WRITE(6,590) JLDEPT(I)
600 CONTINUE
    RETURN
    END
```

APPENDIX B

The Random Problem Generator Code


```

PROGRAM LAST(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
DIMENSION FV(60),FIXED(60),LOC1(60),LOC2(60),LOC3(60)
DIMENSION IFLCW(60,60),IF(60),S(60),SS(60)
DIMENSION IFXC(60,60)
DIMENSION IFR(60,60)
DIMENSION IFPR(60)
DIMENSION IFXFLO(60,60)
DIMENSION IP(60),I00D(60)
DIMENSION SIGMA(3),RVEC(13,2),WKVEC(14)
DIMENSION MIF(60),VAR(300),FLOW(60,60)
DIMENSION IFXF(60)
DIMENSION FVP(60)
INTEGER SS,S,FV,FIXED,ASS,FVP,VAR
INTEGER SIZEFX,SIZEF,AT,AR
INTEGER SLOC1,SLOC2,SLOC3,SORT
INTEGER Q
REAL LACKSA
DIMENSION ASS(60)
IPHC=1
NNCT=1950
READ(5,*)INPV
READ(5,*)INTV
READ(5,*) NLOC
READ(5,*)IRN
READ(5,*) IHRMUP
READ(5,*) SIGMA(1),SIGMA(2),SIGMA(3)
READ(5,*) PC
READ(5,*) LACKSA
READ(5,*)SORT
READ(5,*) CST12,CST13,CST23
READ(5,*) Q
READ(5,*) XM
3 FORMAT(2X,1I3)
5 CONTINUE
DO 15 I=1,IHRMUP
Z=RAND(IRN)
15 CONTINUE
K=0
XINTV=INTV
SEG=100./XINTV
2 FORMAT(2X,1F6.2)
C WRITE(6,2)SEG
DO 11 I=1,INTV
FV(I)=0
11 CONTINUE
10 IF(K.EQ.INPV) GO TO 50
R=RAND(IRN)
RN=R*100
C WRITE(6,2) RN
DO 35 J=1,INTV

```

```

        IF(RN.LT.J*SEG) GO TO 40
35 CONTINUE
40 JSTAR=J
        IF(K.EQ.0) GO TO 44
        DO 42 I=1,K
        IF(JSTAR.EQ.FV(I)) GO TO 10
42 CONTINUE
44 K=K+1
        FV(K)=JSTAR
        GO TO 10
50 CONTINUE
        1 FORMAT(2X,1I4)
        DO 55 I=1,INFV
C        WRITE(6,1) FV(I)
55 CONTINUE
C*****
        KPD=0
        DO 64 I=1,INFV
        MAX=99999
        DO 63 J=1,INFV
        IF(KPD.EQ.0) GO TO 58
        DO 62 JOL=1,KPD
        IF(FV(J).EQ.FVP(JOL)) GO TO 63
62 CONTINUE
58 CONTINUE
        IF(FV(J).LT.MAX) GO TO 61
        GO TO 63
61 JKDP=J
        MAX=FV(JKDP)
63 CONTINUE
        KPD=KPD+1
        FVP(KPD)=FV(JKDP)
64 CONTINUE
        DO 68 I=1,INFV
C        WRITE(6,1) FVP(I)
68 CONTINUE
        DO 69 I=1,INFV
        FV(I)=FVP(I)
69 CONTINUE
C*****
        INFX=INTV-INFV
        DO 60 I=1,INFX
        FIXED(I)=0
60 CONTINUE
        K=0
        DO 80 I=1,INTV
        DO 70 J=1,INFV
        IF(I.EQ.FV(J)) GO TO 80
70 CONTINUE
        K=K+1

```

```
      FIXED(K)=I
80  CONTINUE
      DO 85 I=1,INFX
C   WRITE(6,1) FIXED(I)
85  CONTINUE
      DO 100 I=1,INFX
      LOC1(I)=0
      LOC2(I)=0
      LOC3(I)=0
100 CONTINUE
      ID=0
      DO 912 I=1,NLCC
      ASS(I)=0
912 CONTINUE
      JOE=INFX-NLOC
1001 R=RAND(IRN)
      RN=R*100
1000 DO 1010 I=1,NLOC
      IF(RN.LT.(100*I/NLOC)) GO TO 1020
1010 CONTINUE
1020 KILL=I
1021 DO 1022 I=1,NLOC
      IF(FIXED(KILL).EQ.ASS(I)) GO TO 1001
1022 CONTINUE
      ID=ID+1
      ASS(ID)=FIXED(KILL)
      IF(ID.EQ.NLOC) GO TO 1030
      GO TO 1001
1030 CONTINUE
      L1=1
      L2=1
      L3=1
      LOC1(1)=ASS(1)
      LOC2(1)=ASS(2)
      LOC3(1)=ASS(3)
      DO 150 I=1,INFX
      DO 188 KO=1,IC
      IF(FIXED(I).EQ.ASS(KO)) GO TO 150
188 CONTINUE
      R=RAND(IRN)
      RN=R*100
      IF(RN.GT.0.AND.RN.LT.33.3) GO TO 110
      IF(RN.GT.33.3.AND.RN.LT.66.6) GO TO 120
      IF(RN.GT.66.6.AND.RN.LT.100) GO TO 130
110 L1=L1+1
      LOC1(L1)=FIXED(I)
      GO TO 150
120 L2=L2+1
      LOC2(L2)=FIXED(I)
      GO TO 150
```

```

130 L3=L3+1
    LOC3(L3)=FIXED(I)
    GO TO 150
150 CONTINUE
111 FORMAT(2X,"LOC1")
C   WRITE(6,111)
    IF(L1.EQ.0) GO TO 156
    DO 155 I=1,L1
C   WRITE(6,3) LOC1(I)
155 CONTINUE
156 CONTINUE
112 FORMAT(2X,"LOC2")
C   WRITE(6,112)
    IF(L2.EQ.0) GO TO 161
    DO 160 I=1,L2
C   WRITE(6,3) LOC2(I)
160 CONTINUE
161 CONTINUE
113 FORMAT(2X,"LOC3")
C   WRITE(6,113)
    IF(L3.EQ.0) GO TO 166
    DO 165 I=1,L3
C   WRITE(6,3) LOC3(I)
165 CONTINUE
166 CONTINUE
C*****
C GENERATE BIVARATE NORMAL SS,IFXF A TOTAL OF INFV
C *****
    ISEED=IRN
    N=INFV
    K=2
    IR=INFV
    CALL GGNRM(ISEED,N,K,SIGMA,IR,RVEC,WKVEC,IER)
    DO 170 I=1,INFV
    IFXF(I)=IFIX((((20*(INFV*(1-PC)))*RVEC(I,1))+
* ((120*INFV)*(1-PC))))
    IF(Q.NE.2) GO TO 167
    IFXF(I)=IFIX(((60*(INFV*(1-PC)))+(10*(1-PC)*RVEC(I,1))
* ))
167 CONTINUE
    SS(I)=IFIX((65*RVEC(I,2))+500)
170 CONTINUE
169 FORMAT(5X,"FXFLO",5X,"SIZE")
C   WRITE(6,169)
171 FORMAT(5X,2I12,5X,1F6.3,5X,1F6.3)
    DO 172 I=1,INFV
C   WRITE(6,171) IFXF(I),SS(I),RVEC(I,1),RVEC(I,2)
172 CONTINUE
C*****
C*****

```

```

      IQ=0
      DO 200 I=1,INTV
      DO 202 J=1,INTV
      IF(I.EQ.FIXED(J)) GO TO 205
202  CONTINUE
      IQ=IQ+1
      S(I)=SS(IQ)
      GO TO 200
205  CONTINUE
      S(I)=0
200  CONTINUE
C*****
      201  FORMAT(5X,1I10)
      DO 204 I=1,INTV
C      WRITE(6,201) S(I)
      204  CONTINUE
C*****
      DO 210 I=1,INTV
      IF(S(I).EQ.0) GO TO 211
      GO TO 210
211  CONTINUE
      R=RAND(IRN)
      RN=R*400
      S(I)=IFIX(RN)
210  CONTINUE
C*****
      DO 215 I=1,INTV
C      WRITE(6,201) S(I)
      215  CONTINUE
C*****
      DO 250 I=1,INFX
      DO 252 J=I,INFX
      R=RAND(IRN)
      IF(R.GT.PC) GO TO 251
      IFX0(I,J)=0
      IFX0(J,I)=0
      GO TO 252
251  R=RAND(IRN)
      RN=R*100
      IQQZ=IFIX(RN)
      IFX0(I,J)=IQQZ
      IFX0(J,I)=IQQZ
      IF(I.EQ.J) IFX0(I,J)=0
252  CONTINUE
250  CONTINUE
C*****
      NOO=INFX-1
      DO 260 I=1,INFX
261  CONTINUE
      DO 270 J=1,INFX

```

```

R=RAND(IRN)
IF(R.GT.PC) GO TO 262
IFXFLO(I,J)=0
GO TO 270
262 R=RAND(IRN)
IF(Q.EQ.2) RN=-XM*ALOG(R)
IF(Q.NE.2) RN=R*100.
IQQZ=IFIX(RN)
IFXFLO(I,J)=IQQZ
270 CONTINUE
IT=0
DO 280 JI=1,INFX
IT=IT+IFXFLO(I,JI)
280 CONTINUE
IMOD=IFXF(I)-IT
C WRITE(6,*)(IFXFLO(I,JO),JO=1,INFX)
C WRITE(6,*) IT,IMOD
IBOOM=0
DO 290 JJ=1,INFX
IF(IFXFLO(I,JJ).LE.0) IBOOM=IBOOM-1
IF(IFXFLO(I,JJ).GT.0) KLAST=JJ
IBOOM=IBOOM+1
290 CONTINUE
GGG=IMOD/IBOOM
IGGG=IFIX(GGG)
INGG=IBOOM*IGGG
IADD=IMOD-((IBOOM-1)*IGGG)
DO 295 JK=1,INFX
IF(IFXFLO(I,JK).LE.0) GO TO 295
IF(JK.EQ.KLAST) GO TO 296
IFXFLO(I,JK)=IFXFLO(I,JK)+IGGG
295 CONTINUE
296 IFXFLO(I,KLAST)=IFXFLO(I,KLAST)+IADD
IHOOK=0
DO 297 JL=1,INFX
IF(IFXFLO(I,JL).LT.0) IHOOK=1
297 CONTINUE
IF(IHOOK.EQ.1) GO TO 261
260 CONTINUE
C*****
IF(Q.EQ.1) GO TO 1501
IF(Q.EQ.2) GO TO 1540
ICUE=0
DO 1225 II=1,INFX
MAX=-99999
DO 1200 I=1,INFX
IF(ICUE.EQ.0) GO TO 1211
DO 1210 J=1,ICUE
IF(I.EQ.IFPR(J)) GO TO 1200
1210 CONTINUE

```

```

1211 CONTINUE
      IF(IFXF(I).GT.MAX) GO TO 1205
      GO TO 1200
1205 IPOLO=I
      MAX=IFXF(I)
1200 CONTINUE
      ICUE=ICUE+1
      IFPR(ICUE)=IPOLO
1225 CONTINUE
C*****
      IPSTAR=225
      DO 1500 I=1,INFV
      IPSTAR=IPSTAR-2
      ITIE=IFPR(I)
      LCOUNT=INFV+1-I
      DO 1510 J=1,INFV
      JTIE=IFPR(J)
      IF(J.GE.LCOUNT) GO TO 1520
      IF(ITIE.EQ.JTIE) GO TO 1520
      IPSTAR=IPSTAR-2
      IFR(ITIE,JTIE)=IPSTAR
      IFR(JTIE,ITIE)=IPSTAR
      GO TO 1510
1520 IFR(ITIE,JTIE)=0
      IFR(JTIE,ITIE)=0
1510 CONTINUE
1500 CONTINUE
      GO TO 1549
1501 CONTINUE
C*****
      DO 1530 I=1,INFV
      DO 1535 J=1,INFV
      IF(I.EQ.J) GO TO 1536
      R=RAND(IRN)
      IF(R.LT.PC) GO TO 1536
      R=RAND(IRN)
      RN=R*100.
      IQQZ=IFIX(RN)
      IFR(I,J)=IQQZ
      IFR(J,I)=IQQZ
      GO TO 1537
1536 CONTINUE
      IFR(I,J)=0
      IFR(J,I)=0
1537 CONTINUE
1535 CONTINUE
1530 CONTINUE
1531 CONTINUE
      GO TO 1549
C*****

```

```

1540 CONTINUE
      DO 1541 I=1,INPV
      DO 1542 J=1,INPV
      IF(I.EQ.J) GO TO 1543
      R=RAND(IRN)
      IF(R.GT.PC) GO TO 1544
      IFR(I,J)=0
      IFR(J,I)=0
      GO TO 1542
1544 R=RAND(IRN)
      RN=-XM*ALOG(R)
      IQQZ=IFIX(RN)
      IFR(I,J)=IQQZ
      IFR(J,I)=IQQZ
1543 CONTINUE
1542 CONTINUE
1541 CONTINUE
C*****
1549 CONTINUE
      DO 1600 I=1,INFX
C      WRITE(6,*)(IFXO(I,J),J=1,INFX)
1600 CONTINUE
      DO 1601 I=1,INPV
C      WRITE(6,*)(IFXFLO(I,J),J=1,INFX)
1601 CONTINUE
      DO 1602 I=1,INPV
C      WRITE(6,*)(IFR(I,J),J=1,INPV)
1602 CONTINUE
C*****
      DO 1700 I=1,INFX
      IX=FIXED(I)
      DO 1701 J=1,INFX
      JX=FIXED(J)
      IFLOW(IX,JX)=IFXO(I,J)
1701 CONTINUE
1700 CONTINUE
      DO 1702 I=1,INPV
      IX=FV(I)
      DO 1703 J=1,INFX
      JX=FIXED(J)
      IFLOW(IX,JX)=IFXFLO(I,J)
      IFLOW(JX,IX)=IFXFLO(I,J)
1703 CONTINUE
1702 CONTINUE
      DO 1704 I=1,INPV
      IX=FV(I)
      DO 1705 J=1,INPV
      JX=FV(J)
      IFLOW(IX,JX)=IFR(I,J)
1705 CONTINUE

```



```

1704 CONTINUE
C*****
      DO 475 I=1,INTV
C      WRITE(6,*)(JFLOW(I,J),J=1,INTV)
      475 CONTINUE
C*****
      SIZEFX=0
      DO 630 I=1,INTV
      DO 600 J=1,INFX
      IF(I.EQ.FIXED(J)) GO TO 610
600 CONTINUE
      GO TO 630
610 SIZEFX=SIZEFX+S(I)
630 CONTINUE
      SIZEF=0
      DO 650 I=1,INTV
      DO 660 J=1,INFX
      IF(I.EQ.FV(J)) GO TO 670
660 CONTINUE
      GO TO 650
670 SIZEF=SIZEF+S(I)
650 CONTINUE
      AT=SIZEFX+(SIZEF*LACKSA)
      AQ=AT/NLOC
      AR=IFIX(AQ)
C*****
651 FORMAT(5X,2I12)
C      WRITE(6,651) AT,AR
C*****
      SLOC1=0
      SLOC2=0
      SLOC3=0
      DO 810 I=1,INTV
      DO 800 J=1,L1
      IF(I.EQ.LOC1(J)) GO TO 805
800 CONTINUE
      VAR(I)=0
      GO TO 810
805 SLOC1=SLOC1+S(I)
      VAR(I)=1
810 CONTINUE
      DO 840 I=1,INTV
      DO 830 J=1,L2
      IF(I.EQ.LOC2(J)) GO TO 835
830 CONTINUE
      VAR(INTV+I)=0
      GO TO 840
835 SLOC2=SLOC2+S(I)
      VAR(INTV+I)=1
840 CONTINUE

```

```

      DO 880 I=1,INTV
      DO 870 J=1,L3
      IF(I.EQ.LOC3(J)) GO TO 875
870  CONTINUE
      VAR(2*INTV+I)=0
      GO TO 880
875  SLOC3=SLOC3+S(I)
      VAR(2*INTV+I)=1
880  CONTINUE
C*****
      IF(AR.LT.SLOC1) GO TO 900
      IF(AR.LT.SLOC2) GO TO 900
      IF(AR.LT.SLOC3) GO TO 900
      GO TO 910
901  FORMAT("INFEASIBLE LOCATION HAS OCCURED")
900  CONTINUE
      GO TO 5
910  CONTINUE
C*****
      DO 1300 I=1,INTV
      DO 1305 J=1,INTV
      FLOW(I,J)=IFLOW(I,J)
1305 CONTINUE
1300 CONTINUE
C*****
1101 FORMAT(3I10)
      WRITE (6,1101) AR,AR,AR
1102 FORMAT(5I10)
      WRITE (6,1102) NLOC,INTV,IPHC,INFX,NNCT
1103 FORMAT(57I1)
      WRITE (6,1103) (VAR(I),I=1,INTV)
      WRITE (6,1103) (VAR(INTV+I),I=1,INTV)
      INTV2=2*INTV
      WRITE (6,1103) (VAR(INTV2+I),I=1,INTV)
1104 FORMAT(3X,12F6.1)
      DO 953 I=1,INTV
      WRITE (6,1104) (FLOW(I,J),J=1,INTV)
953  CONTINUE
1105 FORMAT(4X,12I6)
      WRITE (6,1105) (S(I),I=1,INTV)
1106 FORMAT(4X,3F15.0)
      WRITE (6,1106) CST12,CST13,CST23
C*****
      STOP
      END

```

```

FUNCTION RAND(IRN)
  IRN=IRN*16777219
  IF(IRN.LT.0)IRN=-IRN
  RAND=IRN/(2814749767107.E2)
  RETURN
END

```

```

SUBROUTINE GGNOR (ISEED,N,R)

```

```

C
C-GGNOR-----S-----LIBRARY 3-----
C
C
C
C FUNCTION - GENERATES PSEUDO NORMAL RANDOM
C NUMBERS.
C USAGE - CALL GGNOR(ISEED,N,R)
C PARAMETERS ISEED - INPUT. AN INTEGER VALUE IN THE
C EXCLUSIVE RANGE (1,2147483647). ISEED IS
C REPLACED BY A NEW ISEED TO BE USED IN
C SUBSEQUENT CALLS.
C N - INPUT NUMBER OF DEVIATES TO BE
C GENERATED.
C R - OUTPUT VECTOR CONTAINING THE
C NORMAL PSEUDO RANDOM NUMBERS.
C PRECISION - SINGLE
C REQD. IMSL ROUTINES - GGUB, MERFI, UERTST
C LANGUAGE - FORTRAN
C-----
C
C LATEST REVISION - JANUARY 20,1975
C
C DIMENSION R(N)
C GET N RANDOM NUMBERS
C CALL GGUB(ISEED,N,R)
C TRANSFORMS EACH UNIFORM
C DEVIATE
C DO 5 I=1,N
C CALL MDNRIS(R(I),R(I),IER)
5 CONTINUE
RETURN
END

```

SUBROUTINE GGARM (ISEED,N,K,SIGMA,IR,RVEC,WKVEC,IER)

```

C
C-GGARM-----S-----LIBRARY 3-----
C
C-GGARM1
C
C  FUNCTION      GGARM  - MULTIVARIATE NORMAL DEVIATE
C  GENERATOR.
C
C  ENTRY GGARM SHOULD BE USED ON
C  THE FIRST
C  CALL TO FACTOR THE SIGMA MATRIX
C  AND
C  GENERATE DEVIATES.
C  GGARM1 - MULTIVARIATE NORMAL DEVIATE
C  GENERATOR.
C  ENTRY GGARM1 SHOULD BE USED ON
C  ALL BUT THE
C  FIRST CALL, IF MULTIPLE CALLS
C  ARE NECESSARY.
C  USAGE - CALL GGARM(ISEED,N,K,SIGMA,IR,
C  RVEC,WKVEC,IER)
C  CALL GGARM1(ISEED,N,K,SIGMA,IR,
C  RVEC,WKVEC,IER)
C  PARAMETERS ISEED - INPUT. AN INTEGER VALUE IN THE
C  EXCLUSIVE
C  RANGE (1,2147483647). ISEED IS
C  USED TO
C  INITIATE THE GENERATION, AND ON
C  EXIT, HAS
C  BEEN REPLACED BY A NEW ISEED
C  FOR SUBSEQUENT
C  USE.
C  N - INPUT. NUMBER OF K-DEVIATE
C  VECTORS TO
C  GENERATE.
C  K - INPUT. NUMBER OF RANDOM DEVIATES
C  PER VECTOR.
C  SIGMA - INPUT VECTOR OF LENGTH AT LEAST
C  K(K+1)/2
C  CONTAINING THE VARIANCE-
C  COVARIANCE VALUES.
C  SIGMA IS A POSITIVE DEFINITE
C  MATRIX STORED
C  IN SYMMETRIC MATRIX MODE AND IS
C  REPLACED
C  BY ITS FACTOR(SQUARE-ROOT) ON
C  EXIT.
C  IR - INPUT. ROW DIMENSION OF OUTPUT
C  MATRIX RVEC
C  IN THE CALLING PROGRAM. IR
C  MUST BE GREATER

```

```

C          THAN OR EQUAL TO N.
C          RVEC - OUTPUT. N X K MATRIX OF
C          MULTIVARIATE NORMAL
C          DEVIATES.
C          WKVEC - WORK AREA WHERE THE NORMAL
C          DEVIATES
C          ARE GENERATED. WKVEC MUST BE
C          DIMENSIONED
C          AT LEAST K IN LENGTH.
C          IER - ERROR PARAMETER
C          TERMINAL ERROR = 128+N
C          N = 1, INDICATES AN INPUT
C          ERROR TO
C          LUDECF(SIGMA).
C          PRECISION - SINGLE
C          REQD. IMSL ROUTINES - GGNOR,GGUE,LUDECF,MERFI,UERTST
C          LANGUAGE - FORTRAN
-----
C          -----
C          LATEST REVISION - FEBRUARY 18, 1975
C
C          DIMENSION SIGMA(1),RVEC(IR,K),WKVEC(K)
C          DECOMPOSE SIGMA MATRIX
C          CALL LUDECF(SIGMA,SIGMA,K,A,B,IER)
C          IF(IER .GT. 128) GO TO 9000
C          RECALCULATE DIAGONAL OF
C          SIGMA
C          L = 0
C          DO 5 I=1,K
C             L = L+I
C          5 SIGMA(L) = 1.0/SIGMA(L)
C          GO TO 10
C          ENTER GGNRM1 IF MORE K-
C          DEVIATE
C          RANDOM VECTORS
C          DISTRIBUTED WITH
C          THE SAME SIGMA ARE
C          REQUIRED.
C          ENTRY GGNRM1
C          GENERATE N X K NORMAL
C          RANDOM DEVIATES
C          10 DO 25 J=1,N
C          GENERATE K NORMAL
C          DEVIATES
C          CALL GGNOR(ISEED,K,WKVEC)
C          L = 1
C          CONVERT THE K UNIVARIATE
C          NORMAL

```

```

C                                     RANDOM DEVIATES TO
C      MULTIVARIATE                               NORMAL DEVIATES.
C
      DO 20 II=1,K
        RVEC(J,II) = 0.0
        DO 15 I=1,II
          RVEC(J,II) = RVEC(J,II)+DBLE(WKVEC(I))
          * *DBLE(SIGMA(L))
15      L = L+1
20      CONTINUE
25      CONTINUE
      GO TO 9005
9000      CONTINUE
      CALL UERTST (IER,6HGGNRM )
9005      RETURN
      END

```

```

      SUBROUTINE GGUB (ISEED,N,R)
C
C-GGUB-----S-----LIBRARY 3-----
C      -----
C
C      FUNCTION          - BASIC UNIFORM (0,1) PSEUDO-RANDOM
C      NUMBER
C
C      GENERATOR
C      USAGE           - CALL GGUB(ISEED,N,R)
C      PARAMETERS      ISEED - INPUT. AN INTEGER VALUE IN THE
C      EXCLUSIVE
C
C      RANGE (1,2147483647). ISEED IS
C      REPLACED BY
C
C      A NEW ISEED TO BE USED IN
C      SUBSEQUENT CALLS.
C
C      N               - INPUT. THE NUMBER OF DEVIATES TO
C      BE GENERATED
C
C      ON THIS CALL.
C      R(N)           - OUTPUT VECTOR OF LENGTH N,
C      CONTAINING THE
C
C      FLOATING POINT (0,1) DEVIATES.
C      PRECISION      - SINGLE
C      LANGUAGE        - FORTRAN
C-----
C
C      -----
C      LATEST REVISION - JANUARY 28,1974
C
C      DIMENSION      R(1)

```

```

C                                     I2P31M = (2**31) - 1
C                                     S2PN31 = 1 / (2**31)
DATA                                I2P31M/2147483647/,
*                                  S2PN31/16614000000000000000B/
DO 5 I=1,N
    ISEED = MOD(16807*ISEED,I2P31M)
5 R(I) = FLOAT(ISEED)*S2PN31
RETURN
END

```

```

SUBROUTINE LUDECP (A,UL,N,D1,D2,IER)

```

```

C
C-LUDECP-----S-----LIBRARY 3-----
C
C
C FUNCTION                - CHOLESKY DECOMPOSITION OF A
C   MATRIX -
C                               SYMMETRIC STORAGE MODE
C USAGE                    - CALL LUDECP (A,UL,N,D1,D2,IER)
C PARAMETERS A            - INPUT VECTOR OF LENGTH N(N+1)/2
C   CONTAINING
C                               THE N BY N POSITIVE DEFINITE
C   SYMMETRIC
C                               MATRIX STORED IN SYMMETRIC
C   STORAGE MODE.
C   UL                    - OUTPUT VECTOR OF LENGTH N(N+1)/2
C   CONTAINING
C                               THE DECOMPOSED MATRIX L SUCH
C   THAT A = L*
C                               L-TRANSPOSE. L IS STORED IN
C   SYMMETRIC
C                               STORAGE MODE. THE DIAGONAL OF L
C   CONTAINS THE
C                               RECIPROCAL OF THE ACTUAL
C   DIAGONAL ELEMENTS.
C   N                    - ORDER OF A. (INPUT)
C   D1,D2                - COMPONENTS OF THE DETERMINANT OF
C   A.
C                               DETERMINANT(A) = D1*2**D2.
C   (OUTPUT)
C   IER                  - ERROR PARAMETER.
C                               TERMINAL ERROR = 128+N.
C                               N = 1 INDICATES THAT MATRIX A
C                               ALGORITHMICALLY NOT POSITIVE
C   DEFINITE.

```

```

C                                     (SEE THE CHAPTER L PRELUDE).
C   PRECISION                         - SINGLE
C   REQD. IMSL ROUTINES                - UERTST
C   LANGUAGE                           - FORTRAN

```

```

C-----
C
C   LATEST REVISION                    - FEBRUARY 8,1974
C

```

```

      DIMENSION      A(1),UL(1)
      DATA          ZERO,ONE,FOUR,SIXTN,SIXTH/0.0,1.,
* 4.,16.,.0625/
      D1=ONE
      D2=ZERO
      RN = ONE/(N*SIXTN)
      IP = 1
      IER=0
      DO 45 I = 1,N
        IQ = IP
        IR = 1
        DO 40 J = 1,I
          X = A(IP)
          IF (J .EQ. 1) GO TO 10
          DO 5 K=IQ,IP1
            X = X-UL(K)*UL(IR)
            IR = IR+1
5          CONTINUE
10         IF (I.NE.J) GO TO 30
            D1 = D1*X
            IF (A(IP)+X*RN .LE. A(IP)) GO TO 50
15         IF (ABS(D1) .LE. ONE) GO TO 20
            D1 = D1 * SIXTH
            D2 = D2 + FOUR
            GO TO 15
20         IF (ABS(D1) .GE. SIXTH) GO TO 25
            D1 = D1 * SIXTN
            D2 = D2 - FOUR
            GO TO 20
25         UL(IP) = ONE/SQRT(X)
            GO TO 35
30         UL(IP) = X * UL(IR)
35         IP1 = IP
            IP = IP+1
            IR = IR+1
40        CONTINUE
45        CONTINUE
            GO TO 9005
50        IER = 129
9000       CONTINUE
            CALL UERTST(IER,6HLUDECP)
9005       RETURN

```


END

SUBROUTINE MERFI (P,Y,IER)

```

C
C-MERFI-----S-----LIBRARY 3-----
C
C-MERFCI
C-MONRIS
C
C FUNCTION      MERFI - COMPUTE THE INVERSE ERROR
C               FUNCTION.
C               MERFCI - COMPUTE THE INVERSE COMPLEMENTED
C                       ERROR FUNCTION.
C               MONRIS - COMPUTE THE INVERSE GAUSSIAN
C                       INTEGRAL.
C
C USAGE          - CALL MERFI(P,X,IER)
C                - CALL MERFCI(P,X,IER)
C                - CALL MONRIS(P,X,IER)
C
C PARAMETERS     P - INPUT VALUE
C                1.0 AND 1.0 FOR MERFI P MUST BE BETWEEN -
C                0.0 AND 2.0 FOR MERFCI P MUST BE BETWEEN
C                0.0 AND 1.0 FOR MONRIS P MUST BE BETWEEN
C                X - OUTPUT RESULT OF COMPUTATION.
C                IER - ERROR INDICATOR
C                   TERMINAL ERROR = 128+N
C                   N=1 INDICATES P LIES OUTSIDE
C                   THE LEGAL DOMAIN. PLUS OR MINUS INFINITY
C                   IS GIVEN AS THE RESULT (SIGN IS THE SIGN OF
C                   THE FUNCTION VALUE OF THE NEAREST LEGAL
C                   ARGUMENT).
C
C PRECISION     - SINGLE
C REQD. IMSL ROUTINES - UERTST
C LANGUAGE      - FORTRAN
C-----
C
C LATEST REVISION - JANUARY 20, 1976
C
C DIMENSION     A(65),C(17),D(25),E(23)

```



```

C      INVERSE ERROR FUNCTION ENTRY
      INT = 1
      X = P
      GO TO 5
C      INVERSE COMPLEMENTED ERROR FUNCTION ENTRY
      ENTRY MERFCI
      INT = 2
      X = 1. - P
      GO TO 5
C      INVERSE GAUSSIAN INTEGRAL ENTRY
      ENTRY MONRIS
      INT = 3
      X = 1.-P-P
5      IER = 0
      SIGMA = SIGN(1.,X)
      IF (.NOT.(X.GT.-1..AND.X.LT.1.)) GO TO 45
      Z = ABS(X)
      IF(Z.GT..8) GO TO 25
      W = Z*Z/.32-1.
      N = 22
      IPP = 1
      L = 1
10     LB2 = 1
      X3 = 1.
      X4 = W
      X6 = A(IPP)
15     X6 = X6 + A(IPP+LB2) * X4
      X5 = X4 * W * 2.-X3
      X3 = X4
      X4 = X5
      LB2 = LB2 + 1
      IF (LB2 .LE. N) GO TO 15
      GO TO (20,35),L
20     Y = Z * X6 * SIGMA
      IF (INT .NE. 3) GO TO 9005
      GO TO 40
25     B = SQRT(-ALOG(1.-Z*Z))
      IF (Z .GT. .9975) GO TO 30
      W = H1*B+H2
      IPP = 24
      L = 2
      N = 16
      GO TO 10
30     W = H3 * B + H4
      IPP = 41
      N = 24
      L = 2
      GO TO 10
35     Y = B * X6 * SIGMA
      IF (INT .NE. 3) GO TO 9005

```

```

40 Y = R2*Y
   GO TO 9005
45 Y = SIGMA*XINF
   IER = 129
9000 CONTINUE
     CALL UERTST(IER,6HMERFI )
9005 RETURN
     END

```

SUBROUTINE UERTST (IER,NAME)

```

C
C-UERTST-----LIBRARY 3-----
C
C
C
C   FUNCTION           - ERROR MESSAGE GENERATION
C   USAGE              - CALL UERTST(IER,NAME)
C   PARAMETERS        IER - ERROR PARAMETER. TYPE + N WHERE
C                       TYPE= 128 IMPLIES TERMINAL
C                       64 IMPLIES WARNING WITH
C                       32 IMPLIES WARNING
C                       N   = ERROR CODE RELEVANT TO
C   CALLING ROUTINE   NAME - INPUT SCALAR CONTAINING THE NAME
C   OF THE            CALLING ROUTINE AS A 6-
C   CHARACTER LITERAL STRING.
C   LANGUAGE          - FORTRAN
C-----
C
C   LATEST REVISION   - AUGUST 1, 1973
C
C   DIMENSION         ITYP(2,4),IBIT(4)
C   INTEGER           WARN,WARF,TERM,PRINTR
C   EQUIVALENCE       (IBIT(1),WARN),(IBIT(2),WARF),
C * (IBIT(3),TERM)
C   DATA            ITYP /10HWARNING ,10H ,
C *                  10HWARNING(WI,10HTH FIX) ,
C *                  10HTERMINAL ,10H ,
C *                  10HNON-DEFINE,10HD /,
C *                  IBIT / 32,64,128,0/
C   DATA            PRINTR/6LOUTPUT/
C   IER2=IER
C   IF (IER2 .GE. WARN) GO TO 5

```

```
C          NON-DEFINED
    IER1=4
    GO TO 20
  5  IF (IER2 .LT. TERM) GO TO 10
C          TERMINAL
    IER1=3
    GO TO 20
 10  IF (IER2 .LT. WARF) GO TO 15
C          WARNING(WITH FIX)
    IER1=2
    GO TO 20
C          WARNING
 15  IER1=1
C          EXTRACT *N*
 20  IER2=IER2-IBIT(IER1)
C          PRINT ERROR MESSAGE
    WRITE (PRINTR,25) (ITYP(I,IER1),I=1,2),NAME,IER2,IER
 25  FORMAT(26H *** I M S L(UERTST) *** ,2A10,4X,A6,4X,I2,
*      8H (IER = ,I3,1H))
    RETURN
    END
```

APPENDIX C

The Experimental Results

OBS ---	PROBLEM LEVEL -----	STRATEGY LEVEL -----	NUMBER LEVEL -----	CORRELATION LEVEL -----	CONSTRAINT TIGHTNESS LEVEL -----
1	-1	-1	1	1	1
2	-1	-1	1	1	0
3	-1	-1	1	1	-1
4	-1	-1	1	0	1
5	-1	-1	1	0	0
6	-1	-1	1	0	-1
7	-1	-1	1	-1	1
8	-1	-1	1	-1	0
9	-1	-1	1	-1	-1
10	-1	-1	0	1	1
11	-1	-1	0	1	0
12	-1	-1	0	1	-1
13	-1	-1	0	0	1
14	-1	-1	0	0	0

OBS ---	PROBLEM LEVEL -----	STRATEGY LEVEL -----	NUMBER LEVEL -----	CORRELATION LEVEL -----	CONSTRAINT TIGHTNESS LEVEL -----
15	-1	-1	0	0	-1
16	-1	-1	0	-1	1
17	-1	-1	0	-1	0
18	-1	-1	0	-1	-1
19	-1	-1	-1	1	1
20	-1	-1	-1	1	0
21	-1	-1	-1	1	-1
22	-1	-1	-1	0	1
23	-1	-1	-1	0	0
24	-1	-1	-1	0	-1
25	-1	-1	-1	-1	1
26	-1	-1	-1	-1	0
27	-1	-1	-1	-1	-1
28	-1	1	1	1	1

OBS ---	PROBLEM LEVEL -----	STRATEGY LEVEL -----	NUMBER LEVEL -----	CORRELATION LEVEL -----	CONSTRAINT TIGHTNESS LEVEL -----
29	-1	1	1	1	0
30	-1	1	1	1	-1
31	-1	1	1	0	1
32	-1	1	1	0	0
33	-1	1	1	0	-1
34	-1	1	1	-1	1
35	-1	1	1	-1	0
36	-1	1	1	-1	-1
37	-1	1	0	1	1
38	-1	1	0	1	0
39	-1	1	0	1	-1
40	-1	1	0	0	1
41	-1	1	0	0	0
42	-1	1	0	0	-1

OBS ---	PROBLEM LEVEL -----	STRATEGY LEVEL -----	NUMBER LEVEL -----	CORRELATION LEVEL -----	CONSTRAINT TIGHTNESS LEVEL -----
43	-1	1	0	-1	1
44	-1	1	0	-1	0
45	-1	1	0	-1	-1
46	-1	1	-1	1	1
47	-1	1	-1	1	0
48	-1	1	-1	1	-1
49	-1	1	-1	0	1
50	-1	1	-1	0	0
51	-1	1	-1	0	-1
52	-1	1	-1	-1	1
53	-1	1	-1	-1	0
54	-1	1	-1	-1	-1
55	1	-1	1	1	1
56	1	-1	1	1	0

<u>OBS</u>	<u>PROBLEM</u> <u>LEVEL</u>	<u>STRATEGY</u> <u>LEVEL</u>	<u>NUMBER</u> <u>LEVEL</u>	<u>CORRELATION</u> <u>LEVEL</u>	<u>CONSTRAINT</u> <u>TIGHTNESS</u> <u>LEVEL</u>
57	1	-1	1	1	-1
58	1	-1	1	0	1
59	1	-1	1	0	0
60	1	-1	1	0	-1
61	1	-1	1	-1	1
62	1	-1	1	-1	0
63	1	-1	1	-1	-1
64	1	-1	0	1	1
65	1	-1	0	1	0
66	1	-1	0	1	-1
67	1	-1	0	0	1
68	1	-1	0	0	0
69	1	-1	0	0	-1
70	1	-1	0	-1	1

OBS ---	PROBLEM LEVEL -----	STRATEGY LEVEL -----	NUMBER LEVEL -----	CORRELATION LEVEL -----	CONSTRAINT TIGHTNESS LEVEL -----
71	1	-1	0	-1	0
72	1	-1	0	-1	-1
73	1	-1	-1	1	1
74	1	-1	-1	1	0
75	1	-1	-1	1	-1
76	1	-1	-1	0	1
77	1	-1	-1	0	0
78	1	-1	-1	0	-1
79	1	-1	-1	-1	1
80	1	-1	-1	-1	0
81	1	-1	-1	-1	-1
82	1	1	1	1	1
83	1	1	1	1	0
84	1	1	1	1	-1

OBS ---	PROBLEM LEVEL -----	STRATEGY LEVEL -----	NUMBER LEVEL -----	CORRELATION LEVEL -----	CONSTRAINT TIGHTNESS LEVEL -----
85	1	1	1	0	1
86	1	1	1	0	0
87	1	1	1	0	-1
88	1	1	1	-1	1
89	1	1	1	-1	0
90	1	1	1	-1	-1
91	1	1	0	1	1
92	1	1	0	1	0
93	1	1	0	1	-1
94	1	1	0	0	1
95	1	1	0	0	0
96	1	1	0	0	-1
97	1	1	0	-1	1
98	1	1	0	-1	0

OBS ---	PROBLEM LEVEL -----	STRATEGY LEVEL -----	NUMBER LEVEL -----	CORRELATION LEVEL -----	CONSTRAINT TIGHTNESS LEVEL -----
99	1	1	0	-1	-1
100	1	1	-1	1	1
101	1	1	-1	1	0
102	1	1	-1	1	-1
103	1	1	-1	0	1
104	1	1	-1	0	0
105	1	1	-1	0	-1
106	1	1	-1	-1	1
107	1	1	-1	-1	0
108	1	1	-1	-1	-1

OBS	RESPONSE	RESIDUAL	TRANSFORMED RESPONSE	TRANSFORMED RESIDUAL
---	-----	-----	-----	-----
1	1270	537.4	3.10380	.37110
2	67	-189.7	1.82600	-.39740
3	67	286.1	1.86200	.11160
4	1075	-234.0	3.03140	.11840
5	46	-532.0	1.66270	-.74360
6	46	198.9	1.66270	-.23710
7	2113	227.6	3.32480	.23170
8	79	-820.3	1.89760	-.69160
9	79	165.8	1.89760	-.18770
10	902	94.4	2.95520	.18390
11	160	-138.1	2.20410	-.08080
12	160	371.4	2.20410	.40550
13	1132	-34.4	3.05380	.17130
14	226	-261.4	2.35410	-.03940

<u>OBS</u>	<u>RESPONSE</u>	<u>RESIDUAL</u>	<u>TRANSFORMED RESPONSE</u>	<u>TRANSFORMED RESIDUAL</u>
15	226	417.5	2.35410	.44960
16	1660	334.7	3.26950	.27570
17	322	-354.8	2.50780	.00570
18	322	493.7	2.50780	.49740
19	685	-197.6	2.83560	.02590
20	130	-209.5	2.11390	-.23230
21	130	333.6	2.11390	.23110
22	790	-233.8	2.89760	.04550
23	79	-317.9	1.89760	-.48290
24	79	309.1	1.89760	-.01150
25	832	-333.1	2.92010	.02570
26	142	-312.2	2.15220	-.26260
27	142	398.6	2.15220	.21680
28	475	-84.3	2.67660	.09820

OBS	RESPONSE	RESIDUAL	TRANSFORMED RESPONSE	TRANSFORMED RESIDUAL
---	-----	-----	-----	-----
29	43	-104.4	1.63340	-.50280
30	43	307.5	1.63340	-.06070
31	692	2.4	2.84010	.17140
32	43	-156.4	1.63340	-.53760
33	43	333.8	1.63340	-.04010
34	1091	271.2	3.03780	.27890
35	42	-209.4	1.62320	-.58260
36	42	359.1	1.62320	-.02970
37	583	-326.5	2.76560	.05800
38	136	-147.4	2.13350	-.07980
39	136	478.7	2.13350	.41450
40	674	-311.8	2.82860	.05690
41	136	-117.9	2.13350	-.10480
42	133	491.0	2.12380	.41870

OBS	RESPONSE	RESIDUAL	TRANSFORMED RESPONSE	TRANSFORMED RESIDUAL
---	-----	-----	-----	-----
43	934	-128.1	2.97030	.13450
44	136	-208.4	2.13350	-.13000
45	136	509.4	2.13350	.44220
46	565	-694.7	2.75200	-.08470
47	106	-313.4	2.02530	-.26490
48	106	526.8	2.02530	.28150
49	604	-678.0	2.78100	-.09380
50	106	-322.4	2.02530	-.28040
51	106	531.3	2.02530	.28860
52	1152	-152.4	3.06140	.14860
53	106	-331.4	2.02530	-.29580
54	106	535.7	2.02530	.29570
55	806	-237.4	2.98630	-.05550
56	169	-434.6	2.22780	-.22480

OBS	RESPONSE	RESIDUAL	TRANSFORMED RESPONSE	TRANSFORMED RESIDUAL
---	-----	-----	-----	-----
57	169	41.2	2.22780	.28420
58	2571	915.1	3.41010	.26790
59	550	-374.9	2.74030	.10470
60	550	356.1	2.74030	.61120
61	2801	568.7	3.44730	.12490
62	826	-420.2	2.91690	.09850
63	826	565.9	2.91690	.60240
64	1354	199.6	3.13160	.13110
65	70	-575.0	1.84500	-.66900
66	70	-65.5	1.84500	-.18260
67	1917	403.7	3.28260	.17080
68	82	-752.3	1.91380	-.70880
69	82	-73.3	1.91380	-.21980
70	1682	-190.1	3.22580	.00280

OBS	RESPONSE	RESIDUAL	TRANSFORMED RESPONSE	TRANSFORMED RESIDUAL
---	-----	-----	-----	-----
71	67	-956.6	1.82600	-.90520
72	94	-81.2	1.97310	-.26640
73	1902	672.6	3.27920	.24010
74	289	-397.3	2.46080	-.11460
75	190	46.8	2.27870	.16670
76	1785	414.3	3.25160	.17030
77	277	-466.7	2.44240	-.16730
78	178	61.3	2.25040	.11210
79	2614	1102.1	3.41730	.29360
80	367	-434.1	2.56460	-.07950
81	238	147.8	2.37650	.21190
82	822	-255.1	2.91480	.04680
83	244	-421.2	2.38730	-.03850
84	244	-9.3	2.38730	.40350

OBS	RESPONSE	RESIDUAL	TRANSFORMED RESPONSE	TRANSFORMED RESIDUAL
----	-----	-----	-----	-----
85	1294	86.7	3.11190	.15360
86	244	-473.2	2.38730	-.07340
87	244	17.0	2.38730	.42410
88	1300	-37.6	3.11390	.06540
89	244	-525.1	2.38730	-.10810
90	244	43.2	2.38730	.44470
91	3204	1776.7	3.50560	.50840
92	67	-734.2	1.82600	-.67680
93	67	-108.1	1.82600	-.18240
94	3128	1624.4	3.49520	.43390
95	67	-764.7	1.82600	-.20190
96	67	-92.8	1.82600	-.16860
97	2282	702.1	3.35830	.23280
98	67	-795.1	1.82600	-.72700

<u>OBS</u>	<u>RESPONSE</u>	<u>RESIDUAL</u>	<u>TRANSFORMED</u> <u>RESPONSE</u>	<u>TRANSFORMED</u> <u>RESIDUAL</u>
99	67	-77.4	1.82600	-.15480
100	2512	734.5	3.40000	.27350
101	226	-711.2	2.35410	-.22570
102	118	21.1	2.07180	.03850
103	2514	714.2	3.40030	.23590
104	226	-720.2	2.35410	-.24120
105	118	25.5	2.07180	.04560
106	2501	678.8	3.39810	.19570
107	226	-729.1	2.35410	-.25660
108	118	29.9	2.07180	.05270

BIBLIOGRAPHY

1. Ahrens, J. H., and G. Finke, "Merging and Sorting Applied to the Zero-One Knapsack Problem," Operations Research, Vol. 23, No. 6, 1975, 1099-1109.
2. Bartlett, M.S., "The Use of Transformations," Biometrics, 3, 1947, 39-52.
3. Bartlett, M.S., "Properties of Sufficiency and Statistical Tests," Proc. Roy Soc., A., 160, 1937, 268-282.
4. Box, G. E. P., "Non-normality and Tests on Variances," Biometrika, 40, 1953, 318-335.
5. Box, G. E. P., "Some Theorems on Quadratic Forms Applied in the Study of ANOVA Problems: I. Effect of Inequality of Variance in the One-Way Classification," Annual Mathematical Statistics, 25, 1954, 290-302.
6. Box, G. E. P., and D. R. Cox, "An Analysis of Transformations," Journal of Royal Statistical Society, 13, 26, 1964, 211-252.
7. Cochran, W. G., "Some Consequences When the Assumptions for the Analysis of Variance Are Not Satisfied," Biometrics, 3, 1947, 22-38.
8. Dolby, J. L., "A Quick Method of Choosing a Transformation," Technometrics, 5, 1963, 317-325.
9. Draper, N. R. and W. G. Hunter, "Transformations: Some Examples Revisited," Technometrics, 11, 1969, 23-40.
10. Eisenhart, C., "The Assumptions Underlying the Analysis of Variance," Biometrics, 3, 1947, 1-21.
11. Ellwein, L. B., "A Flexible Enumeration Scheme for Zero-One Programming," paper submitted for publication to Operations Research.
12. Geoffrion, A.M., "Recent Computational Experience with Three Classes of Integer Linear Programs," Working Paper P-3699, The RAND Corporation, Santa Monica, Calif., 1967.

BIBLIOGRAPHY (Continued)

13. Graves, Robert J., Implicit Enumeration and Branch and Bound Approach to a Sub-Class of Location-Allocation Problems, Ph.D. dissertation, School of Industrial and Systems Engineering, University of Buffalo, 1974.
14. Hartley, H. O., "Testing the Homogeneity of a Set of Variances," Biometrika, vol. 31, 1940.
15. Hicks, Charles R., Fundamental Concepts in the Design of Experiments. New York: Holt, Rinehart and Winston, 1964.
16. Lee, Wayne, Experimental Design and Analysis. San Francisco: W. H. Freeman, 1975.
17. Levene, H., "Robust Test for Equality of Variance" in I. Olkin (Ed.) Contributions to Probability and Statistics. Stanford University Press, 1960.
18. Lin, Benjamin, Development of Controlled Computational Experiments on Integer Programming Procedure, Ph.D. dissertation, School of Industrial and Systems Engineering, Georgia Institute of Technology, 1975.
19. Lin, B., and R. L. Rardin, "Development of a Parametric Generating Procedure for Integer Programming Test Problems," A Georgia Institute of Technology Working Paper, 1975.
20. Michaels, W. M., and R. P. O'Neill, "A Mathematical Program Generator," Presented at the ORSA/TIMS Joint National Meeting, Chicago, Illinois, April 30, 1975.
21. Pierce, J. F., "Application of Combinatorial Programming to a Class of All-Zero-One Integer Programming Problems," Management Science, 15, 1968, 191-209.
22. Rardin, R. L., Group Theoretic and Related Approaches to Fixed Charge Problem, Ph.D. dissertation, School of Industrial and Systems Engineering, Georgia Institute of Technology, 1974.
23. Rardin, R. L., and V. E. Unger, "Solving Fixed Charge Network Problems with Group Theory-Based Penalties," A Georgia Institute of Technology Working Paper, 1975.

BIBLIOGRAPHY (Concluded)

24. Sampford, M. R., and J. Taylor, "Censored Observations in Randomized Block Experiments," Journal of Royal Statistical Society, B, 21, 1959, 214-237.
25. Taylor, J., "The Analysis of Designed Experiments with Censored Observations," Biometrics, 29, 1973, 35-43.
26. Trauth, C. A., and R. E. Woolsey, "Integer Linear Programming: A Study in Computational Efficiency," Management Science, 15, 1969, 481-493.
27. Zions, Stanley, Linear and Integer Programming. Englewood Cliffs: Prentice-Hall, Inc., 1974.