OPTIMIZATION OF SINGLE RESOURCE ALLOCATION PROBLEMS WITH NONSEPARABLE OBJECTIVE FUNCTIONS

By

UMIT YUCEER

Bachelor of Science
Middle East Technical University
Ankara, Turkey
1972

Master of Science
Middle East Technical University
Ankara, Turkey
1974

Master of Science in Engineering
Johns Hopkins University
Baltimore, Maryland
1977

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
DOCTOR OF PHILOSOPHY
December, 1980

OPTIMIZATION OF SINGLE RESOURCE ALLOCATION PROBLEMS WITH NONSEPARABLE OBJECTIVE FUNCTIONS

Thesis Approved:

Philip M. Wolfe
Thesis Adviser

M. Palmer Terrell

Wayne Thomas

Joe H. Mize

Donald W. Grace

Norman N. Durham
Dean of the Graduate College

PREFACE

Solving nonlinear integer programming problems usually causes difficulties. One way to solve this type of problem is to devise clever approximate methods (heuristics). This research investigates the mathematical properties of the single resource allocation problems with nonseparable objective functions. Consequently some heuristic algorithms are proposed for solving this particular class of allocation problems.

This work is related to two RAND Publications. Alternative measures of supply performance is discussed by R. B. S. Brooks, C. A. Gillen, and J. Y. Lu in [3]. They also provided some mathematical arguments for optimization of these measures. A portion of this research is concentrated on improving their methods. In the other publication by B. L. Miller [12], the concept of discrete convexity was introduced. This concept proved to be very useful in developing a primal search algorithm.

Another aspect of this research is the computational experience with these algorithms. Several problems of various sizes were solved to compare the performance of the algorithms with respect to the time and computational feasibility and the accuracy of the solutions.

I wish to express my appreciation to Dr. Philip M. Wolfe for his encouragement and guidance throughout this research as my research adviser. I also extend my sincere thanks to Dr. Marvin P. Terrell,

the chairman of my Ph. D. committee, Dr. Joe H. Mize, Dr. Donald W. Grace, Dr. A. Goicoechea, and Dr. W. C. Turner who served as members of my committee. It has been my pleasure to work under these outstanding faculty members during my education in Oklahoma State University.

I would like to dedicate this work to my late father who was a source of inspiration and encouragement to me. I am also grateful to my mother, sisters and brother for their continuing support in all the years of my education.

I thank Velda Davis and TOP Services Unlimited for their excellent typing of my dissertation.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Allocation problems deal with allocating the available resources to a set of activities, or sequences of jobs in an economically effective manner. The economical effectiveness is measured by the return which the activity yields. This research considers only the class of allocation problems with a single constraint and a nonseparable nonlinear return function over the nonnegative integers.

The objective of this research is to develop a primal search algorithm to solve single resource allocation problems with nonseparable nonlinear objective functions. In addition, the Solution Method of Brooks et al. [3] has been successfully modified for the same purpose. These two algorithms have been implemented to solve a real life problem, namely the War Readiness Spares Kit Problem. Several problems of various sizes were solved for comparison of the performances of the algorithms. The computational experience with them reveals that they can generate good solutions within specified time limits. The modified algorithm of Brooks et al. proved to be superior to the Primal Search Algorithm in terms of time and computational feasibility and the quality of the solutions. Moreover, a simulation model was developed to test the accuracy of the solutions obtained by analytical methods. The simulation study confirms the analytical results.

Allocation problems are discussed in Chapter II. An important

aspect of the allocation problems is the definition of the return function. After defining the foundations of the return function, a mathematical model is developed to represent the single resource allocation problems with nonseparable return functions. The background of this study is presented in the remainder of this chapter.

Chapter III describes three methods to solve this particular class of allocation problems. Mathematical properties of this class of problems are investigated thoroughly, then a primal search algorithm is presented. This algorithm generates a better feasible solution by moving along a favorable direction until no more progress can be made. A second approach requires that the objective function be approximated by a separable function, then marginal analysis can be applied to find the optima. This approach is the modification of the method proposed by Brooks et al. in [3]. The third method is a saddle point search algorithm. At each iteration, a Lagrange multiplier is first determined, then the optimal solution to the primal problem is sought by an algorithm proposed by B. L. Miller [12]. If two consecutive Lagrange multipliers are sufficiently close together, the iterations are terminated, the final solution is declared as the saddle point solution to the problem.

A real life problem, namely the War Readiness Spares Kit problem, is introduced in Chapter IV. A simple problem is used to demonstrate how the algorithms proceed in finding a (near) optimal solution. Several problems of various sizes were solved, then some computational features of the algorithms are displayed in tables and figures. Finally, a simulation model is utilized to observe the response of the spare kits obtained by these algorithms. The FORTRAN listings of the computer programs of the algorithms and the simulation model are given in the

appendices. A summary and conclusions of this research are given in Chapter V. In addition, some potential research areas are mentioned briefly.

CHAPTER II

ALLOCATION PROBLEMS

An Orientation

Allocation problems are frequently encountered in industry, government and military. Several organizations are involved in performing many activities such as manufacturing products, providing services, transportation, inventory control and purchasing, etc. Usually each organization performs a set of related and/or similar activities. For instance, a manufacturing plant produces several types of products, hence manufacturing each type product is considered an activity for this company. Naturally some resources such as men, money, machine, time, raw materials, so forth, are required to perform these activities. Unfortunately there exists physical, temporal or economical restrictions on the resources in real life. This means that the activities can not be performed in the best possible fashion. Some of the reasons for the limited resources are the capacity of the facilities, the amount of raw materials available at any time, lack of capital or more men power when needed, etc. Consequently the conflict arises as how to allocate the available resources into a set of activities in an effective manner.

An allocation of the resources to a set of activities yields a certain return which can be used to determine what "the effective manner" should be. This return is in general expressed in terms of a unit; total profit (in terms of dollars) from the production of some goods or the

total cost (in terms of dollars) of transporting some goods from plants or warehouses to the retail stores or customers. Obviously, the magnitude of the return depends upon the magnitude of the resources allocated and upon the levels of the activities. The return infact defines a measure of performance of the activity levels. This performance may be interpreted as the effectiveness or ineffectiveness of the activity. The allocation problem is then to allocate the available resources to a set of activities in such a way as to maximize some measure of effectiveness or to minimize some measure of ineffectiveness.

The problems with a single resource available for allocations to activities are classified as single-resource allocation problems. Infact, knapsack problems belong to this class, but they have separable return functions. This research is focused on single resource allocation problems with nonlinear, nonseparable return functions. The activities are nonnegative integers, the resource is money, and the objective function represents the return on the resource in terms of an ineffectiveness measure.

Properties of the Return Function

An important aspect of the allocation problems is the definition of the return function of the activities. Usually there are several activities to be considered simultaneously in allocating a resource. An activity vector will then represent the magnitudes (with respect to some unit) of the activities of concern as its components. As an example, the quantities of n-different type of products produced by a manufacturing company forms an activity vector. A resource can indeed be used in numerous ways by manipulating the magnitudes of the components of the activity vector.

The basic assumptions of the return function will then be stated as follows:

i) The return function is a real-valued function defined on an activity space.

ii) It is monotonic; if the level of one activity is increased while others remain fixed, the return on the activity vector is increased, in case of effectiveness; it is decreased for ineffectiveness.

iii) The return function is concave if it is an effectiveness measure. It is convex if the return function is an ineffectiveness measure. If the activity space is discrete, concavity or convexity is replaced by discrete convexity or concavity.

A real-valued function f defined on a discrete rectangle D is a discretely convex if $x,y \in D$, and $0 < \alpha < 1$,

$$\min_{z \in N(\alpha x + (1-\alpha)y)} f(z) \leq \alpha L(x) + (1-\alpha)f(y)$$

where $N(\alpha x + (1-\alpha)y) = \{w, \text{integer}: \; ||w - (\alpha x + (1-\alpha)y)|| < 1\}$ is the neighborhood of the point $\alpha x + (1-\alpha)y$ with a radius less than one. The restriction of a convex function to a discrete rectangle is not unfortunately a discretely convex function [12].

Mathematical Formulation

A mathematical model for this class of problems will be constructed as follows:

Let x_i denote the quantity (integer) for the i^{th} activity $i = 1, \ldots, n$, and $f(x_1, \ldots, x_n)$ denote a measure of ineffectiveness of the activity vector $(x_1, x_2, \ldots, x_n)$. Further $f(x_1, \ldots, x_n)$ possesses the properties i), ii), iii) of the previous section.

The only available limited resource is money and denoted by M. Application of each unit of i^{th} activity costs $\$c_i$, and $c_i > 0$, $i = 1, \ldots, n$. Hence the total cost of the activity vector $(x_1, x_2, \ldots, x_n)$ is

$$\sum_{i=1}^{n} c_i x_i$$

The requirement is that the total cost of the activity vector not exceed the available resource M.

The problem is then in mathematical form;

$$\min f(x_1, \ldots, x_n) \tag{2.1}$$

subject to

$$\sum_{i=1}^{n} c_i x_i \leq M \tag{2.2}$$

$$x_i \geq 0, \text{ integer } i = 1, \ldots, n.$$

Background of the Study

An important aspect of allocation problems is the development and the construction of a measure of effectiveness or ineffectiveness. Several researchers have different observations and assumptions. According to R. E. Bellman [1], the return from each activity is independent of the returns from the other activities, and hence the total return

is the sum of the individual returns. This establishes the separability assumption of the (in)effectiveness measure. However, in a study done by R. B. S. Brooks, C. A. Gillen, and J. Y. Lu [3], alternative measures of supply performance for the U.S. Air Force Operations were developed. They are the fill rate, backorders, operational rate, and NORS. They are briefly defined as follows: fill rate is the ratio of the number units issued over a fixed time period to the number demanded over the same period. Backorders consist of the number of stock-outs from base supply at any point in time. Operational rate is the probability of no due-outs at any point in time. Finally, the number of NORS (not operationally ready supply) aircraft is the number grounded for lack of spare parts at any point in time. Brooks et al. observed that the NORS measure was a better representation of the performance for Air Force base operations. Fill rate, backorders and operational rate yield separable (in)effectiveness measures. On the contrary, the NORS measure does not yield a separable ineffectiveness measure. This fact reveals that the separability assumption of the (in)effectiveness measure is not adequate for some real life problems. Therefore, nonseparable (in)effectiveness measures should also be taken into consideration.

Finding numerical solutions to a problem is as important as the recognition of it. Deriving numerical solutions to the problems is not always an easy task. Methods do not exist to solve all kinds of problems. Sometimes modification of some method will suffice for the purpose. New methods are always sought to solve some problems better and faster than the existing methods. If there is no method available to solve a certain class of problems, then the discovery and the development of a new method becomes necessary.

If the activities in allocation problems assume fractional values, then either linear programming or nonlinear programming methods can be used to solve them. In particular, feasible direction methods are commonly used in solving nonlinear programming problems. The procedure entails developing a sequence of solutions with improving objective function values; a new solution is obtained from the previous one by determining a search direction in which progress can be made and determining the step size in that direction. Methods of feasible directions have been described in many books and articles, e.g., in [11, 17, 16, 15].

In some allocation problems, the activities can assume only integer values. Solving integer programming problems usually causes difficulties. Even though some of the solution techniques are theoretically appealing, they are not computationally satisfactory. (See Garfinkel and Nemhauser [6].) Even Branch-and-Bound methods are computationally inefficient for large scale problems.

A few methods have been developed to solve single resource allocation problems with separable objective and constraint functions. R. E. Bellman suggests using dynamic programming to solve this type of problem. A good treatment of the method together with flow charts for computational experimentation is given in [1]. He also proved that dynamic programming would obtain the global optimum. Another method for solving the same class of problems is marginal analysis [5]. This method generates a sequence of undominated activities by considering the incremental return per additional dollar spent for each activity. Convergence of the marginal analysis to the global solution can easily be proved by means of Lagrangian analysis [5]. The marginal analysis is easy and efficient to

implement when solving single resource allocation problems with convex and separable objective and constraint functions.

There have been a few attempts to solve one dimensional allocation problems with a nonseparable objective function. One attempt was by R. B. S. Brooks, C. A. Gillen, and J. Y. Lu [3]. They approximated the NORS function by a linear combination of separable functions, then set up the Lagrangian function. This allowed them to optimize a separable function. Unfortunately, the parameters in the linear combination for the NORS function were difficult to obtain. However, they adapted the following procedure to solve the problem: fix the parameters and repeatedly change the Lagrange multiplier until an activity that meets the budget constraint is obtained. Then recalculate the parameters and repeat the process. These steps are repeated until the parameters stabilize. The adjustment of the Lagrange multiplier was done by linear programming, as suggested in [2]. The authors claimed that the method was successful in producing a good solution. They nevertheless pointed out that the algorithm was not guaranteed to always work. It is rather a cumbersome approach.

Another study was done by B. L. Miller [12] of the RAND Corporation. His approach uses the NORS function as it is, contrary to the approach of [3]. He developed the concept of discrete convexity and proved that the Lagrangian function was discretely convex for a fixed multiplier. Then he proposed a heuristic search procedure to find the optimal solution to the primal problem which was the minimization of Lagrangian over the nonnegative integers. His algorithm consists of three phases. In the first phase an initial solution is obtained. The second phase is to generate a better feasible solution by changing only one component at a

time while others remain fixed, and repeating this process for each component. In the third phase, the neighborhood of this point is searched using heuristic rules to obtain improvement. If an improvement is obtained, then the second phase and consequently the third phase is repeated. Otherwise, that point is called an unverified optimal solution. He claimed his method found food solutions and was computationally feasible.

Many real life integer programming problems are too large to be solved by exact algorithms. Consequently, clever approximate methods (heuristics) are the only alternatives to solving these problems [9, 5]. A few of the heuristic algorithms which are pertinent to this research will be discussed briefly. An heuristic procedure to solve integer linear programming problems was developed by F. S. Hillier [7, 8], Efficient Heuristic Procedures for Integer Linear Programming Problems with an Interior and a Bound-and-Scan Algorithm for Pure Integer Linear Programming with General Variables. His method has three phases. In the first phase, a region within which to explore for better solutions is identified. The second phase consists of moving along a direction while searching nearby for a feasible integer solution. The purpose of the third phase is to improve upon the feasible solution obtained in the second phase. His research reveals that the optimum can be obtained in a reasonable amount of time and effort. On the other hand, S. Reiter and D. B. Rice [14] developed an approximate method to solve quadratic integer programming problems. Their method consists of choosing a random starting point, then locating a feasible point and applying a modified gradient maximizing procedure in search of a better point.

They pointed out that their method could be extended to more general integer programming problems.

In this research a primal algorithm is developed to solve this particular class of simple resource allocation problem (described above) in an efficient manner. This approach makes use of the convexity and the monotonicity of the objective function and the linearity of the constraint. The search procedure is based on the concept of feasible directions. In addition the Algorithm of Brooks et al. [3, p. 24] has been modified to solve the same class of problems. The method of marginal analysis has been implemented in Step 1 of their algorithm in order to optimize a separable programming problem.

CHAPTER III

SOLUTION METHODS

General

The mathematical problem to be solved is of the form

$$\begin{aligned} &\min f(X) \\ \text{subject to} \quad &CX \leq M \\ &X \geq 0, \text{ integer} \end{aligned} \tag{3.1}$$

where f is discretely convex on the set $\Pi^n = \{X = (x_1, \ldots, x_n)$: x_i is a nonnegative integer for all $i = 1, \ldots, n\}$. The function f is monotonically decreasing in each component. Also, f has the properties i), ii), iii) of the section on Properties of the Return Function in Chapter II. Further $C = (c_1, c_2, \ldots, c_n)$ with each $c_i > 0$; $i = 1, \ldots, n$, and $M > 0$.

All the solution methods described in this chapter are based on the following theorem.

Theorem 1: If X^* is optimal to this problem, then $X^* + e_k$ is infeasible for all $k = 1, \ldots, n$ where e_k is the k^{th} unit vector.

Proof: If $X^* + e_k$ for some $k = 1, \ldots, n$ were feasible, then $f(X^* + e_k) \leq f(X^*)$ by monotone decreasing property, and $X^* + e_k$ would be optimal. This contradicts the hypothesis of the theorem.

The Lagrangian function of this problem is given by

$$L(X; \lambda) = f(X) + \lambda(CX - M) \quad \text{for } X \in \Pi^n,\ \lambda > 0. \tag{3.2}$$

The primal problem is stated as

$$\min_{X \in \Pi^n} L(X; \lambda) \tag{3.3}$$

and the dual problem is then stated as

$$\max_{\lambda > 0} L(X; \lambda) \tag{3.4}$$

A point $\bar{X} \in \Pi^n$ is called a stationary point of a discretely convex function $g: \Pi^n \to R$ if $g(\bar{X}) \leq g(\bar{X} + S)$ where $S = \sum_{k=1}^{n} s_k e_k$, $s_k = -1, 0, 1$ for $k = 1, \ldots, n$.

If $L(X; \lambda)$ is a discretely convex function on Π^n for $\lambda \gg 0$, and if X^* solves the primal problem, then $L(X^*; \lambda) \leq L(X^* \pm e_k; \lambda)$ for $k = 1, \ldots, n$ holds. This condition is not sufficient unfortunately for non-separable functions. The sufficiency of this condition for separable functions can be proven easily.

Theorem 2: Let $g: \Pi^n \to R$ be a separable function, then for X^* to be stationary it is sufficient to have $g(X^*) \leq g(X^* \pm e_k)$ for $k = 1, \ldots, n$.

Proof: Since g is separable, $g(X) = \sum_{i=1}^{n} g_i(x_i)$. Let $S = \pm e_k$ for some $k = 1, \ldots, n$. Then $g(X^*) \leq g(X^* \pm e_k)$, and $\Sigma g_i(x_i^*) \leq \sum_{i \neq k} g_i(x_i^*) + g_k(x_k^* \pm 1)$, further

$g_k(x_k^*) \leq g_k(x_k^* \pm 1)$, or by replacing ± 1 by s_k

$$g_k(x_k^*) \leq g_k(x_k^* + s_k) \qquad \text{where } s_k = -1,\ 0,\ +1.$$

This argument is obviously true for all k = 1, ..., n, hence

$$\sum_{k=1}^{n} g_k(x_k^*) \leq \sum_{k=1}^{n} g_k(x_k^* + s_k). \tag{3.5}$$

The left hand side of the inequality is $g(X^*)$, and the right hand side of the inequality is $g(X^* + S)$. Consequently, $g(X^*) \leq g(X^* + S)$. In other words, the condition is derived from $g(X^*) \leq g(X^* \pm e_k)$ for all k = 1, ..., n. In addition, if g is unimodal in each component, then X^* is the global optimum. This result implies that it suffices to optimize one component at a time.

These results will be implemented to develop algorithms to solve the problem (3.1). An example of a nonseparable discretely convex function is the following;

$$G(X;\lambda) = \sum_{j=0}^{m} \left(1 - \prod_{i=1}^{n} F_i(x_i + j)\right) + \sum_{i=1}^{n} \sum_{j=1} \left(1 - F_i(x_i + j)\right) + \lambda \sum_{i=1}^{n} c_i x_i \tag{3.6}$$

where $F_i(\cdot)$ is the cumulative distribution function of a Poisson random variable with mean λ_i, i=1, ..., n. The construction of this function is discussed in Chapter IV, and the discrete convexity of $G(X; \lambda)$ is proved by B. L. Miller in [12].

A Primal Search Algorithm

This algorithm makes use of the monotonicity and the discrete convexity of the objective function. A better feasible solution is obtained from the current solution by determining a search direction and heuristically selecting a step size. This algorithm is developed during this research.

Step 0. Initialization, pick a λ and p where $0 < p < 1$, and set $k = 0$. Find X^o such that $L(X^o; \lambda) \leq L(X^o \pm e_k; \lambda)$ for $i = 1, \ldots, n$. In particular, the first differences for the function (3.6) are given by

$$\Delta_i L(X; \lambda) = L(X + e_i; \lambda) - L(X; \lambda) \text{ for } i = 1, \ldots, n \qquad (3.7)$$

and

$$\Delta_i L(X; \lambda) = -\sum_{j=0}^{m} ((F_i(x_i + j + 1) - F_i(x_i + j)) \prod_{\substack{k=1 \\ k \neq i}}^{n} F_k(x_k + j) - 1 +$$

$$F_i(x_i + m + 1) + \lambda c_i \qquad (3.8)$$

Since $0 \leq \prod_{k \neq i} F_i(x_i + j) \leq 1$ for $0 \leq j \leq m$, then

$$-\sum_{j=1}^{m} F_i(x_i + j) - F_i(x_i + m + 1) + F_i(x_i) + \sum_{j=1}^{m} F_i(x_i + j) - 1 +$$

$$F_i(x_i + m + 1) + \lambda c_i \leq \Delta_i L(X; \lambda) \qquad (3.9)$$

and

$$F_i(x_i) - 1 + \lambda c_i \leq \Delta_i L(X; \lambda) \leq F_i(x_i + m + 1) - 1 + \lambda c_i \qquad (3.10)$$

Therefore, it suffices to find an $X^o = (x_1^o, \ldots, x_n^o)$ where each x_i^o satisfies $-1 + F_i(x_i^o - 1) + \lambda C_i \leq 0$ and

$$-1 + F_i(x_i^o) + \lambda c_i > 0 \quad i = 1, \ldots, n. \qquad (3.11)$$

Step 1. Search for a better feasible solution; set $k = k + 1$. Construct a favorable direction

$$d_k = -\nabla f(X^k) - p \cdot p_k \cdot C \tag{3.12}$$

where p_k is obtained by solving $(-\nabla f(X^k) - p_k C).C = 0$ and p is left as an input parameter by the user. d_k is normalized such the $\max\{|d_{ki}|; i = 1, \ldots, n\} = 1$. Calculate an α by $\alpha = \left[\frac{M - C.X^k}{\sqrt{n}\ C.d_k}\right]$ where $[\cdot\cdot]$ denotes the greatest integer function. Then a point y_α is obtained as $y_\alpha = X^k + \alpha d_k$. The nearest integer point V to y_α is obtained by $v_i = [y_{\alpha i} + .5]$ for $i = 1, \ldots, n$.

i) If V is feasible and $f(V) < f(X^k)$, set $X^{k+1} = V$, go to Step 1.

ii) If V is feasible, but $f(V) \geq f(X^k)$, then check the points $V + e_i$ for $i = 1, \ldots, n$, Figure 1. If there exists an i^* such that $V + e_{i^*}$ is feasible and $f(V + e_{i^*}) < f(X^k)$, set $X^{k+1} = V + e_{i^*}$, and go to Step 1. Otherwise, go to Step 2.

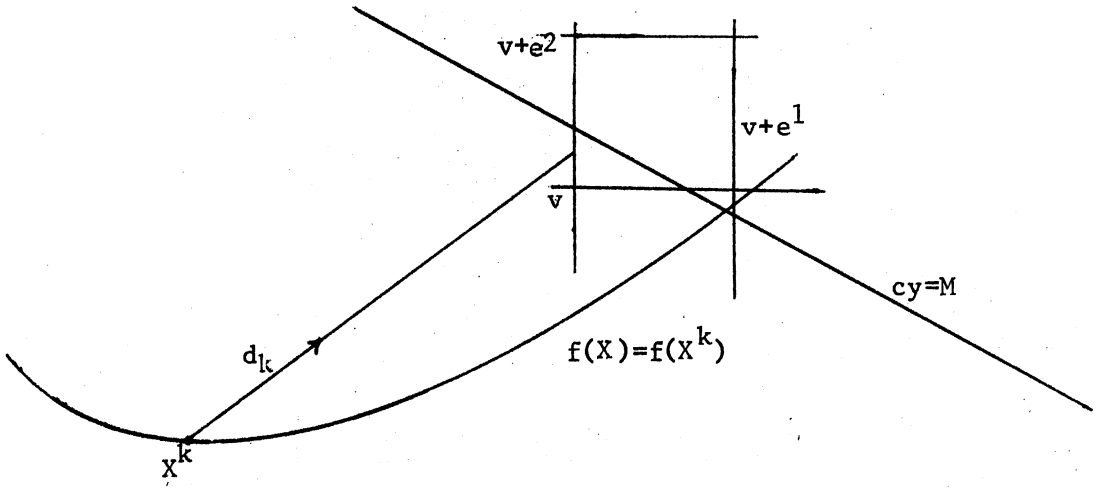

Figure 1. Search in the Positive Quadrant

iii) If V is not feasible, then search the points $V - e_i$ for $i = 1,$

..., n. If there exists an i^* such that $V - e_{i^*}$ is feasible and $f(V - e_{i^*}) < f(X^k)$, set $X^{k+1} = V - e_i$ and go to Step 1. Otherwise go to Step 2.

Step 2. Test of optimality via Theorem 1. Test the points $X^k + e_i$, $i = 1, \ldots, n$. If there exists an i^* such that $X^k + e_{i^*}$ is feasible and $f(X^k + e_{i^*}) < f(X^k)$, set $X^{k+1} = X^k + e_{i^*}$ and go to Step 1, otherwise terminate.

The Modified Algorithm of Brooks et al.

This approach requires the objective function f(X) be approximated by a separable function of the form

$$f(X) \simeq \sum_{i=1}^{n} \sum_{j=0}^{m} b_j F_i(x_i;\ j)$$

where

$$b_j = \beta_j(x_1', x_2', \ldots, x_n') \geq 0 \tag{3.13}$$

for some $X' = (x_1', x_2', \ldots, x_n')$, and for some function $\beta_j(\ .\ )$.

Let $$g_i(x_i) = \sum_{j=0}^{m} b_j F_i(x_i;\ j)$$

then

$$f(X) = \sum_{i=1}^{n} g_i(x_i) \tag{3.14}$$

An example of such an approximation is given in Appendix A. In particular, if $g_i(x_i)$ is strictly decreasing, then the implementation of the marginal analysis is justified.

Step 0. Initialization, pick b_0^o, b_1^o, ..., b_m^o, set k = 1.

Step 1. Solve the problem.

$$\min \sum_{i=1}^{n} g_i(x_i) \tag{3.15}$$

Subject to

$$\sum_{i=1}^{m} c_i x_i \leq M$$

$$x_i \geq 0 \text{ integer}$$

by the following algorithm which is known as marginal analysis [5], also a consequence of Theorem 2.

Step 1 i) Initialization: $X^o = (0, 0, \ldots, 0)$, set $\ell = 0$.

Step 1 ii) $X^{\ell+1} = X^{\ell} + e_j$, if feasible, where j is any index for which the ratio

$$r_i = \frac{g_i(x_i + 1) - g_i(x_i)}{C_i} \tag{3.16}$$

is maximum.

Step 1iii) Stop if $CX^{\ell+1} \geq M$, and set $X^k = X^{\ell}$, then go to Step 2. Otherwise go to 1ii.

Step 2 Calculate $b_j^k = \beta_j(X^k)$ for j = 0, 1, ..., m.

Step 3 If b_j^k is very close to b_j^{k-1} for all j = 0, 1, ..., m, then $X^* = X^k$, and stop. Otherwise, set k = k + 1, and go to 1.

This is a modified version of the algorithm proposed by R. B. S. Brooks, C. A. Gillen, and J. Y. Lu on page 24 of [3]. The modification is also accomplished during this research. The Step 1 of their approach

was quite different from the one described above. They attempted to find the saddle point solution to the problem

$$\max_{\lambda \geq 0} \min_{X \geq 0} L(X: \lambda) = \Sigma g_i(x_i) + \lambda(\Sigma c_i x_i - M).$$

Their Step 1 was as follows:

Step 1) Find $x_1^k, x_2^k, \ldots, x_n^k$ at k^{th} iteration which maximizes

$$\sum_{j=0}^{m} b_j \sum_{i=1}^{n} \log F_i(x_i + j) - \lambda \sum_{i=1}^{n} c_i x_i$$

Since this function is separable, the optimization is carried out for each component separately. At the end of the third step, the Lagrange Multiplier is rearranged as suggested in [2], and the algorithm repeated again.

This algorithm may not converge in general. The convergence needs to be proved for specific functions. In fact, the convergence of this algorithm for a separable function as an approximation of Equation (3.6) was given by a theorem in [3].

A Saddle Point Search Algorithm

This algorithm solves the problem

$$\max_{\lambda > 0} \min_{X \in \Pi^n} L(X; \lambda).$$

The algorithm in fact searches the saddle point (X^*, λ^*) of the Lagrangian function. This is, however, more complicated than the previous algorithms, and requires subalgorithms to solve the primal and dual subproblems. This was originally suggested by H. Everett [2].

Step 0. Initialization. Pick X^o and λ_o. Set $k = 0$.

Step 1. Solve the primal subproblem.

$$\min_{X \in \Pi^n} L(X; \lambda_k)$$

to obtain X^{k+1}.

Step 2. Solve the dual subproblem

$$\max_{\lambda > 0} L(X^{k+1}; \lambda)$$

to obtain λ_{k+1}.

Step 3. Stop if λ_{k+1} is very close to λ_k. Otherwise set $k = k + 1$ and go to Step 1.

B. L. Miller developed an algorithm to solve the primal subproblem for discretely convex functions. Since a detailed description of this algorithm is presented in 12 , it is not going to be discussed here. This is a very complicated algorithm. The author claims it produces good solutions.

At the k^{th} iteration ($k \geq 1$), a piecewise convex approximation of $L(X; \lambda)$ can be given in the space generated by the points, X^0, X^1, ..., X^{k+1} as follows:

$$L(X; \lambda) = \sum_{t=0}^{k+1} \mu_t L(X^t; \lambda) \tag{3.17}$$

where $\sum_{t=0}^{k+1} \mu_t = 1$, $\mu_t \geq 0$ for all $t = 0, 1, \ldots, k+1$.

Then the dual subproblem is restated as

$$\max \sum_{t=0}^{k+1} \mu_t L(X^t; \lambda) \tag{3.18}$$

Subject to $\sum_{t=0}^{k+1} \mu_t = 1$

$$\lambda>0,\ \mu_t \geq 0 \text{ for } t = 0, 1, \ldots, k+1$$

Since $L(X; \lambda) = f(X) + \lambda CX - \lambda M$, then the problem

$$\max \sum_{t=0}^{k+1} \mu_t f(X^t) + \lambda \sum_{t=0}^{k+1} \mu_t CX^t - \lambda M \tag{3.19}$$

Subject to $\sum_{t=0}^{k+1} \mu_t = 1$

$$\lambda>0,\ \mu_t \geq 0 \text{ for } t = 0, 1, \ldots, k+1$$

can be considered as the Lagrangian of the linear program

$$\min \sum_{t=0}^{k+1} \mu_t f(X^t) \tag{3.20}$$

Subject to $\sum_{t=0}^{k+1} \mu_t CX^t \leq M$

$$\sum_{t=0}^{k+1} \mu_t = 1$$

$$\mu_t \geq 0 \text{ for } t = 0, 1, \ldots, k+1$$

or as the Lagrangian of the dual linear program

$$\min (\lambda M + \nu) \tag{3.21}$$

Subject to $\lambda CX^t + \nu \geq - f(X^t)$ for $t = 0, 1, \ldots, k+1$

$\lambda > 0$

ν unrestricted.

Therefore, the dual linear program will be solved instead of the dual subproblem in Step 2. One of the findings of this research is solving this linear program without using the simplex method.

The primal linear program has an optimal solution $\mu_0^*, \mu_1^*, \ldots, \mu_{k+1}^*$, since the feasible set is bounded. Then the constraint of the Equation (3.20) implies that there exists $\bar{X} = \sum_{t=0}^{k+1} \mu_t^* X^t$ not necessarily integer such that $C\bar{X} = M$ and $f(\bar{X}) = \sum_{t=0}^{k+1} \mu_t^* f(X^t)$. On the other hand, the dual linear program yields a solution (λ^*, ν^*) such that $\lambda^* M + \nu^* = -f(\bar{X})$. Let I_s be the index set of all t's such that $CX^t \leq M$ and I_b the index set of all t's such that $CX^t \geq M$ for $t = 0, 1, \ldots, k+1$. Since f represents a measure of ineffectiveness of the activity vector X, then Property (iii) of II-b applies and $-f(X^i) < -f(\bar{X}) < -f(X^j)$ for all $i \in I_s$ and $j \in I_b$. Consequently the line $\lambda M + \nu = -f(\bar{X})$ lies in the region bounded by the lines $\lambda = 0$, $\lambda CX^i + \nu = -f(X^i)$ and $\lambda CX^j + \nu = -f(X^j)$ for all $i \in I_s$ and $j \in I_b$. Hence, this line lies in the intersection of all these regions. The intersection is the region bounded by $\lambda = 0$, $\lambda CX^{i*} + \nu = -f(X^{i*})$ and $\lambda CX^{j*} + \nu = -f(X^{j*})$ where $CX^{i*} = \max \{CX^i;\ i \in I_s\}$ and $CX^{j*} = \min \{CX^j;\ j \in I_s\}$. Finally

$$\lambda^* = \frac{f(X^{i*}) - f(X^{j*})}{CX^{j*} - CX^{i*}}. \tag{3.22}$$

The significance of this result lies in the fact that the simplex method is not required to solve a linear programming of the type

$$\min du + y$$

$$\text{Subject to} \qquad a_i u + y \geq b_i \quad i = 1, \ldots, m \tag{3.23}$$

$$u \geq 0$$

$$y \text{ unrestricted}$$

with the property that $a_1 < a_2 < \ldots < a_m$ and $b_1 > b_2 > \ldots > b_m$.

CHAPTER IV

APPLICATION AND COMPUTIONAL RESULTS

The War Readiness Spares Kit Problem

This problem has arisen from the operations of U.S. Air Force. It is a typical example of single resource allocation problems. A brief description of the problem is presented in the following paragraph.

A squadron of airplanes is on certain occasions sent out to duty at remote bases for a period of a month or so. Usually these bases lack the necessary repair facilities; consequently, an airplane in such a base can only be repaired by replacing the defective item by a good one. Thus the squadron needs to stock some essential spare parts; these are the parts which form the war readiness spares kit. During the flying period, if an airplane breaks down due to failure of some parts, and if these parts are available in the kit, it is repaired. If at least one of these required parts is in short supply and if there are some other nonoperationally-ready planes awaiting repairs, then at least one of them is cannibalized to consolidate the required parts in order to repair the others. Consequently the number of cannibalized planes defines an ineffectiveness measure for the performance of the spare kit. The problem is then to determine the number of each spare part in the kit which will yield the least ineffectiveness under the condition that the total cost of the spare kit is not to exceed a certain budget level.

With regard to the mathematical model of Chapter II, Mathematical Formulation, the only available resource is the budget. Let the activity vector $X = (X_1, X_2, \ldots, X_n)$ represent the spare kit. Then x_i denotes the amount of spare part i purchased at the beginning of the flying period at a cost of $\$c_i$, and the total cost of the spare kit is given by the sum $\sum_{i=1}^{n} c_i x_i$ or in vectoral form CX. The measure of ineffectiveness is the expected number of cannibalized planes or nonoperationally ready supply airplane, and is denoted by E(NORS/X). The demand of the spare part i is assumed to be a Poisson Process with a rate μ_i and independent of the others. Then the probability of $j \geq 0$ or fewer backorder is $F_i(x_i + j)$ where $F_i(\,.\,)$ is the cumulative poisson distribution with mean μ_i. The repairs are restricted by the spare part with the most number of backorders, consequently this particular spare part causes the cannibalization of some nonoperational airplanes. The probability that the item in worst shape is backordered j or less is equivalent to the probability that all n items are backordered j or less. Furthermore the latter probability is equal to the probability of cannibalizing at most j airplanes. Hence P(number of cannibalized airplanes $\leq$ j) = $\prod_{i=1}^{n} F_i(x_i + j)$ since the demands of spare parts are stochastically independent. Finally E(number of cannibalized airplanes) = $\sum_{j=0}^{\infty}$ (1-P(number of cannibalized airplanes $\leq$ j)). Under the assumption of cannibalization, the number of cannibalized airplanes is the same as the number of nonoperationally ready supply airplanes because of a lack of some spare parts.

Therefore $$E(NORS/X) = \sum_{j=0}^{\infty} \left\{1 - \prod_{i=1}^{n} F_i(x_i + j)\right\}. \qquad (4.1)$$

This function is approximated for computational purposes by the following function [12]

$$E(NORS/X) = \sum_{j=0}^{m} \left(1 - \prod_{i=1}^{n} F_i(x_i + j)\right) + \sum_{i=1}^{n} \sum_{j=m+1}^{\infty} \left(1 - F_i(x_i + j)\right) \quad (4.2)$$

where m is chosen such that $\prod_{i=1}^{n} F_i(x_i + j)$ is very close to 1 for $j > m$.

Finally the war readiness spares kit problem in mathematical form is

$$\min f(X) = E(NORS/X) \quad (4.3)$$

Subject to

$$CX \leq M \quad (4.4)$$

$$X \geq 0 \text{ and integer.}$$

In addition, the first differences of E(NORS/X) is given by

$$\Delta_i f(X) = f(X + e_i) - f(X) \quad \text{for } i=1, \ldots, n. \quad (4.5)$$

$$\Delta_i f(X) = - \sum_{j=0}^{m} \left(F_i(x_i + j + 1) - F_i(x_i + j)\right) \prod_{\substack{k=1 \\ k \neq 1}}^{n} F_k(x_k + j) - 1 +$$

$$F_i(x_i + m + 1) \quad (4.6)$$

An Example

A war readiness spares kit problem with five items is used to demonstrate how the algorithms of Chapter III work. The steady-state Poisson demand rates and cost of each item is given in Table I.

The budget is assumed to be M = \$25000. In addition the value for m of Equation (4.2) is chosen to be 5.

i) Solution by Means of the Primal Search Method

A listing of the computer program is given in Appendix D.

Iteration 0. Initialization: Parameter p is chosen to be .85 and the Lagrange multiplier is specified as $\lambda = .0002$. The effect of different parameters will be discussed later. This iteration seeks an X^o which satisfies the conditions of (3.11).

TABLE I

COSTS AND DEMAND RATES FOR THE SAMPLE WRSK PROBLEM

Item No.	Cost/Unit	Demand Rate
1	2980.	2.10
2	1751.	1.50
3	462.	1.20
4	1500.	5.00
5	345.	3.50

Since $-1 + F_1(1) + (.0002)\ (2980) = -1 + .37962 + .596 = -.02438$, and $-1 + F_1(2) + (.0002)\ (2980) = -1 + .64963 + .596 = .24563$, $x_1^o = 2$. Repeating the same argument for each component, yields $X^o = (2, 2, 3, 6, 6)$.

Iteration 1. Seeking a better feasible solution: The first differences at the point $X^o = (2, 2, 3, 6, 6)$ are computed from (3.8) and $\Delta f\ (2, 2, 3, 6, 6) = (-.240, -.106, -.015, -.163, -.035)$

$$p_1 = \frac{-\Delta f(2, 2, 3, 6, 6)}{C.C}\ .C = .00008$$

Since $p = .85$, then after normalization, the direction is

$$d_1 = (.610, -.225, -.268, 1., .190)$$

and α is evaluated to be 1. Thus a point $U = (3, 2, 3, 7, 6)$ is obtained by finding the nearest integer point to $Y = (2.610, 1.775, 2.732, 7., 6.190)$. However the point U is not feasible, cost

(U) = 26398, hence a search in the negative direction is required. For that purpose $\Delta_1 f(2, 2, 3, 7, 6) = -.262$, $\Delta_2 f(3, 1, 3, 7, 6) = -.333$, $\Delta_3 f(3, 2, 2, 7, 6) = -.076$, $\Delta_4 f(3, 2, 3, 6, 6) = -.185$, and $\Delta_5 f(3, 2, 3, 7, 5) = -.100$ are calculated. The minimum of these values is for i = 3, but (3, 2, 2, 7, 6) is not feasible. The second minimum value is for i = 5, again (3, 2, 3, 7, 5) is not feasible. The third minimum is for i = 4, fortunately (3, 2, 3, 6, 6) is feasible and f(3, 2, 3, 6, 6) = .98571. Therefore X^1 = (3, 2, 3, 6, 6) with CX = 24898.

Iteration 2. As in Iteration 1, a search direction d_2 = (-.356, .362, -.037, 1., .216) issuing from (3, 2, 3, 6, 6) is determined. A point U = (3, 2, 3, 7, 6) is obtained similarly. Since U is infeasible, and a search in the negative direction does not yield a distinct and better feasible point than X^1 = (3, 2, 3, 6, 6), a search in the positive direction at X^1 is employed. All the points of the form $X^1 + e_i$ for i = 1, 2, ..., 5 are infeasible. Hence X^1 = (3, 2, 3, 6, 6) is optimal.

ii) Solution by the Modified Solution Method of Brooks et al.

A listing of the computer program is given in Appendix E.

Iteration 1. Choose $b_0 = b_1 = \ldots = b_4 = 0$ and $b_5 = 1$. Set $x_i^o = 0$ for i = 1, 2, ..., 5, then $g_i(x_i) = b_5 \log F_i(x_i + 5)$, and calculating the ratio $r_i = \dfrac{g_i(x_i + 1) - g_i(x_i)}{C_i}$ for each i = 1, ..., 5 yields: $r_1 = 0.00000496$, $r_2 = 0.00000202$, $r_3 = 0.00000270$, $r_4 = 0.00014200$, $r_5 = 0.00024952$. For instance,

$$r_1 = \frac{b_5}{c_1} \log \frac{F_1(6)}{F_1(5)} = \frac{1}{2980} \log \frac{.99413}{.97955} = 4.96 \times 10^{-6}$$

r_5 is the maximum, hence $x_5^1 = 1$. Updating r_5 yields: $r_1 = 0.00000496$, $r_2 = 0.00000202$, $r_3 = 0.00000270$, $r_4 = 0.00014200$, $r_5 = 0.00011714$. The maximum is r_4, then $x_4^1 = 1$. Continuing in this fashion produces $X^1 = (2, 2, 2, 8, 7)$ with $\text{cost}(X^1) = 24801$, and calculating b_j by $b_j^1 = \prod_{i=1}^{n} F_i(x_i^1 + j)$ gives $b_0^1 = .419$, $b_1^1 = .726$, $b_2^1 = .898$, $b_3^1 = .967$, $b_4^1 = .991$, and $b_5^1 = .998$.

Iteration 2. Repeating the same steps with b_j^1 $j = 0, 1, \ldots, 5$ yields $X^2 = (2, 2, 4, 7, 9)$ with $\text{cost}(X^2) = 24915$, and $b_0^2 = .450$, $b_1^2 = .728$, $b_2^2 = .891$, $b_3^2 = .962$, $b_4^2 = .988$, and $b_5^2 = .996$. Since b_j^1's are not within .001 of b_j^2's, go to Iteration 3.

Iteration 3. Repeating the same steps again with b_j^2 for $j = 0, 1, 2, \ldots, 5$ yields $X^3 = X^2$ and $b_j^3 = b_j^2$, hence X^2 is optimal with $f(2, 2, 4, 7, 9) = .98619$.

This solution however is a near optimal solution. The primal search procedure has produced a better solution than this one. On the other hand this method is faster, the execution time to solve this small problem with this algorithm is .15 seconds and The Primal Search Method requires .24 seconds.

In this algorithm b_j's are treated as parameters. They have practical implications for real life problems. β_j is given by the product $\prod_{i=1}^{n} F_i(\bar{X} + j)$ for some $\bar{X}$ and $j = 0, 1, \ldots, m$. This product is in fact equal to the probability of cannibalizing at most j airplanes with a kit $\bar{X}$. In other words b_j gives the operational rate of j airplanes with a kit $\bar{X}$. Selecting $b_0 = b_1 = \ldots = b_{m-1} = 0$ and $b_m = 1$ implies that all the planes will be cannibalized and corresponds to the most pessimistic view.

iii) Solution by the Saddle Point Search Algorithm.

A listing of the computer program is given in Appendix F.

Iteration 1. A starting solution of $X^0 = (10, 10, 10, 10, 10)$ with $f(X^0) = .02486$ and $cost(X^0) = 70380$ and a Lagrange multiplier of $\lambda_0 = .0001$ is chosen. In the first step the subproblem $\min_{X \in \Pi^n} L(X;\lambda_0) = f(X) + \lambda_0(CX - M)$ is solved by the algorithm proposed by B. L. Miller in [12]. This algorithm produces the solution $X^1 = (2, 2, 2, 6, 6)$ with $f(X^1) = 1.28241$ and $cost(X^1) = 21456$. As an example the generation of this solution is given in Appendix B. In the second step the linear program

$$\min .02486\ \mu_1 + 1.28241\ \mu_2$$

Subject to $$70380\ \mu_1 + 21456\ \mu_2 \leq 25000$$

$$\mu_1 + \mu_2 = 1$$

$$\mu_1, \mu_2 \geq 0$$

is solved to obtain $\lambda_1 = \frac{1.28241 - .02486}{70380 - 21456} = .0000257$ in the same way as discussed in Chapter III, The Saddle Point Search Algorithm. Since λ_1 is not within .000005 of λ_0 another iteration is performed.

Iteration 2. The subproblem $\min_{X \in \Pi^n} L(X;\lambda_1) = f(X) + \lambda_1(CX - M)$ is solved again in the first step by the same method to obtain $X^2 = (4, 4, 4, 9, 9)$ with $f(X^2) = .17727$ and $cost(X^2) = 37377$. In the second step the linear program $\min .02486\ \mu_1 + 1.28261\ \mu_2 + .17727\ \mu_3$,

Subject to $$70380\ \mu_1 + 21456\ \mu_2 + 37377\ \mu_3 \leq 25000$$

$$\mu_1 + \mu_2 + \mu_3 = 1$$

$$\mu_i \geq 0 \quad i = 1, 2, 3$$

is solved to obtain $\lambda_2 = \frac{1.28261 - .17727}{37377 - 21456} = .0000694$. Since λ_2 is not within .000005 of λ_1, another iteration is required. Continuing in this fashion produces the following results:

Iteration 3. $X^3 = (3, 3, 3, 8, 7)$ with $f(X^3) = .49955$ and cost $(X^3) = 29994$. Further $\lambda_3 = .0000917$.

Iteration 4. $X^4 = (2, 2, 3, 7, 7)$ with $f(X^4) = 1.02459$ and cost $(X^4) = 23763$. Also $\lambda_4 = .0000848$.

Iteration 5. $X^5 = (3, 2, 3, 7, 7)$ with $f(X^5) = .75556$ and cost $(X^5) = 26743$. Also $\lambda_5 = .0000903$.

Iteration 6. $X^6 = (2, 2, 3, 7, 6)$ with $f(X^6) = 1.06282$ and cost $(X^6) = 23418$ and $\lambda_6 = .0000903$.

This algorithm produced a solution with a better objective value than the previous ones, but it is an infeasible solution. In fact, this algorithm suffers from the duality gap. In the first step of this algorithm, the subproblem $\min_{X \in \Pi^n} L(X;\lambda)$ is solved. Since $L(X;\lambda)$ is discrete in $X \in \Pi^n$, there is a range for λ values which produce the same answer. From the results of previous algorithms, it was discovered that the optimal solution was $X^* = (3, 2, 3, 6, 6)$. Evidently its corresponding Lagrange multiplier should lie between $\lambda_4 = .0000848$ and $\lambda_5 = .0000903$. Unfortunately the algorithm won't generate this value, and if λ is close to λ_4, the solution to subproblem is X^5 and if λ is close to λ_5, then the solution is X^6, see Figure 2.

Computational Results

An important aspect of any algorithm is the computational efficiency. More specifically, the amount of time required to generate a (near)

optimal solution to the problem is of importance. Therefore several problems of various sizes need to be solved by these methods for comparison. In order to obtain several test problems, the data in Table II was obtained from the data given in [3] by randomly selecting 100 items with some modifications. The purpose was to avoid duplication in demand rates and/or costs of the items, and to have a variety of items with different demands and costs. In Table II, the first column shows the item number, second column is for the unit cost of the item, and third column shows the demand rate of the items for a period of 30 days.

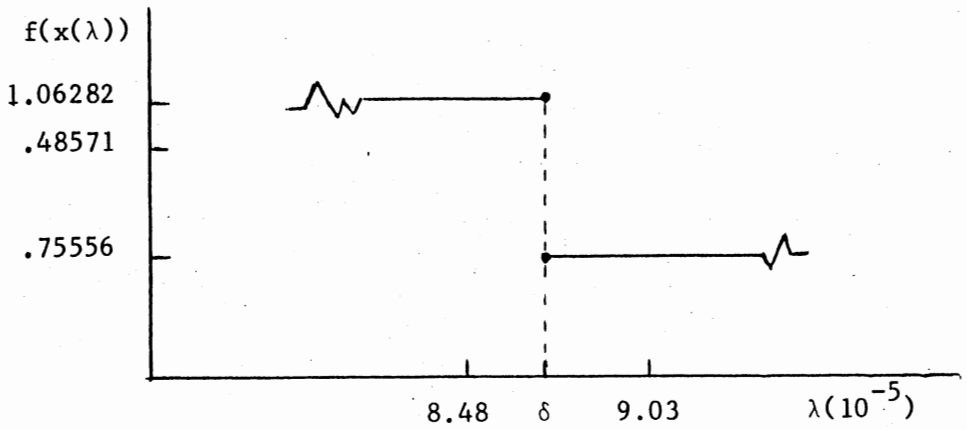

Figure 2. Duality Gap

Several test problems of various sizes are obtained by a random selection of the items without replacement from the list in Table II. A computer program was prepared to obtain the test problems of sizes 10, 20, 40, and 80. The computer program generates probability for a uniformly distributed random variable. Each item in the list has an

TABLE II

THE DATA

ITEM	COST/UNIT	DEMAND RATE	ITEM	COST/UNIT	DEMAND RATE
# 1	1613.00	0.50	# 51	139.00	3.20
# 2	1338.00	0.50	# 52	1409.00	3.20
# 3	1219.00	0.60	# 53	1844.00	3.25
# 4	214.00	0.70	# 54	25.00	3.30
# 5	628.00	0.80	# 55	163.00	3.30
# 6	350.00	0.90	# 56	1572.00	3.30
# 7	1261.00	0.90	# 57	225.00	3.40
# 8	79.00	1.00	# 58	2083.00	3.40
# 9	2223.00	1.00	# 59	345.00	3.50
# 10	800.00	1.05	# 60	725.00	3.50
# 11	157.00	1.10	# 61	900.00	3.50
# 12	171.00	1.10	# 62	593.00	3.70
# 13	196.00	1.10	# 63	229.00	4.00
# 14	1080.00	1.15	# 64	352.00	4.00
# 15	136.00	1.20	# 65	849.00	4.00
# 16	366.00	1.20	# 66	414.00	4.30
# 17	462.00	1.20	# 67	4378.00	4.30
# 18	2056.00	1.20	# 68	537.00	4.70
# 19	1703.00	1.25	# 69	71.00	5.00
# 20	388.00	1.30	# 70	1500.00	5.00
# 21	863.00	1.30	# 71	784.00	5.10
# 22	950.00	1.30	# 72	569.00	5.30
# 23	344.00	1.40	# 73	257.00	5.50
# 24	608.00	1.50	# 74	1240.00	6.00
# 25	631.00	1.50	# 75	411.00	6.50
# 26	1751.00	1.50	# 76	863.00	6.70
# 27	215.00	1.70	# 77	3007.00	6.80
# 28	2051.00	1.70	# 78	8500.00	6.90
# 29	326.00	1.80	# 79	379.00	7.00
# 30	1289.00	1.80	# 80	1404.00	7.00
# 31	552.00	1.90	# 81	376.00	7.10
# 32	1393.00	1.90	# 82	1279.00	7.20
# 33	784.00	2.00	# 83	1090.00	8.00
# 34	855.00	2.00	# 84	659.00	8.30
# 35	3388.00	2.00	# 85	2370.00	8.50
# 36	1208.00	2.10	# 86	1246.00	8.70
# 37	2484.00	2.10	# 87	2066.00	9.00
# 38	2580.00	2.10	# 88	3898.00	9.30
# 39	172.00	2.25	# 89	1121.00	9.80
# 40	750.00	2.30	# 90	36.00	10.00
# 41	555.00	2.40	# 91	796.00	10.00
# 42	449.00	2.50	# 92	475.00	10.40
# 43	814.00	2.60	# 93	2053.00	11.00
# 44	928.00	2.60	# 94	124.00	11.30
# 45	1150.00	2.80	# 95	2775.00	11.50
# 46	2632.00	2.90	# 96	1830.00	11.80
# 47	704.00	3.00	# 97	675.00	12.00
# 48	896.00	3.00	# 98	3430.00	12.00
# 49	1449.00	3.00	# 99	1560.00	12.50
# 50	326.00	3.10	#100	196.00	13.00

equal probability to be selected. When an item is assigned to a sample problem, then the number of the items in the list is decreased by one, and the probability of selecting any item is recalculated. This process is repeated until a sample problem of predetermined size is obtained. Its listing is given in Appendix C. The five test problems in Tables III and V were solved by the Primal Search Algorithm and the Modified Algorithm of Brooks et al. The Saddle Point Search Algorithm is not used, because it is very time consuming and suffers from the duality gap as explained in the previous section. Both of these algorithms are coded in WATFIV for the IBM 370/168 at the Oklahoma State University Computer Center. The computational results are displayed in Tables IV and VI. In addition, the solutions of test problems by both algorithms are given in Table V.

A careful investigation of Table IV reveals that the Modified Algorithm of Brooks et al. is computationally much faster than the Primal Search Algorithm in producing a (near) optimal solution, in other words it takes much less time to converge to a solution. Three iterations were required to solve any problem of any size, and the value of the objective function was evaluated at the end only. The number of iterations which the Primal Search Algorithm requires is dependent on the size of the problem, and increases as the problem size increases. At the end of each iteration, the Primal Search Algorithm evaluates the objective function. This is perhaps one of the reasons why it takes so long to generate a solution. The number of steps within each iteration remains the same for the Primal Search Method for any problem of any size, but varies for the Modified Algorithm of Brooks et al. with respect to the size of the problem.

TABLE III

SAMPLE PROBLEMS

PROBLEMS OF SIZE 10

	1		2		3		4		5	
	c	μ	c	μ	c	μ	c	μ	c	μ
# 1	215.00	1.70	1338.00	0.50	414.00	4.30	1208.00	2.10	215.00	1.70
# 2	2056.00	1.20	608.00	1.50	1409.00	3.20	414.00	4.30	1751.00	1.50
# 3	171.00	1.10	225.00	3.40	1261.00	0.90	1530.00	5.00	1572.00	3.30
# 4	326.00	3.10	725.00	3.50	345.00	3.50	449.00	2.50	3388.00	2.00
# 5	449.00	2.50	593.00	3.70	462.00	1.20	1449.00	3.00	2083.00	3.40
# 6	863.00	1.30	704.00	3.00	1572.00	3.30	1409.00	3.20	1393.00	1.90
# 7	2484.00	2.10	388.00	1.30	139.00	3.20	800.00	1.05	631.00	1.50
# 8	855.00	2.00	2580.00	2.10	1703.00	1.25	157.00	1.10	136.00	1.20
# 9	1080.00	1.15	139.00	3.20	163.00	3.30	1150.00	2.30	1613.00	0.50
# 10	71.00	5.00	1289.00	1.80	552.00	1.90	2484.00	2.10	537.00	4.70

PROBLEMS OF SIZE 20

	1		2		3		4		5	
	c	μ	c	μ	c	μ	c	μ	c	μ
# 1	215.00	1.70	414.00	4.30	215.00	1.70	388.00	1.30	2580.00	2.10
# 2	2056.00	1.20	1409.00	3.20	1751.00	1.50	1080.00	1.15	863.00	1.30
# 3	171.00	1.10	1261.00	0.90	1572.00	3.30	896.00	3.00	4378.00	4.30
# 4	326.00	3.10	345.00	3.50	3388.00	2.00	171.00	1.10	1613.00	0.50
# 5	449.00	2.50	462.00	1.20	2083.00	3.40	1219.00	0.60	784.00	2.00
# 6	863.00	1.30	1572.00	3.30	1393.00	1.90	25.00	3.30	326.00	3.10
# 7	2484.00	2.10	139.00	3.20	631.00	1.50	2223.00	1.00	214.00	0.70
# 8	855.00	2.00	1703.00	1.25	136.00	1.20	537.00	4.70	1500.00	5.00
# 9	1080.00	1.15	163.00	3.30	1613.00	0.50	704.00	3.00	814.00	2.60
# 10	71.00	5.00	552.00	1.90	537.00	4.70	552.00	1.90	704.00	3.00
# 11	1338.00	0.50	3388.00	2.00	704.00	3.00	3388.00	2.00	172.00	2.25
# 12	631.00	1.50	849.00	4.00	552.00	1.90	196.00	1.10	631.00	1.50
# 13	2083.00	3.40	1500.00	5.00	449.00	2.50	139.00	3.20	171.00	1.10
# 14	725.00	3.50	750.00	2.30	814.00	2.60	1449.00	3.00	1150.00	2.80
# 15	593.00	3.70	2632.00	2.90	1289.00	1.80	593.00	3.70	215.00	1.70
# 16	896.00	3.00	896.00	3.00	1703.00	1.25	214.00	0.70	449.00	2.50
# 17	388.00	1.30	2223.00	1.00	725.00	3.50	2484.00	2.10	800.00	1.05
# 18	750.00	2.30	157.00	1.10	157.00	1.10	928.00	2.60	2484.00	2.10
# 19	1844.00	3.25	814.00	2.60	1844.00	3.25	1261.00	0.90	537.00	4.70
# 20	1289.00	1.80	855.00	2.00	900.00	3.50	229.00	4.00	849.00	4.00

TABLE III (Continued)

PROBLEMS OF SIZE 40

	1		2		3		4		5	
	c	μ	c	μ	c	μ	c	μ	c	μ
# 1	215.00	1.70	215.00	1.70	2580.00	2.10	462.00	1.20	2083.00	3.40
# 2	2056.00	1.20	1751.00	1.50	863.00	1.30	1080.00	1.15	629.00	0.80
# 3	171.00	1.10	1572.00	3.30	4378.00	4.30	1338.00	0.50	1613.00	0.50
# 4	326.00	3.10	3388.00	2.00	1613.00	0.50	215.00	1.70	2484.00	2.10
# 5	449.00	2.50	2083.00	3.40	784.00	2.00	593.00	3.70	214.00	0.70
# 6	863.00	1.30	1393.00	1.90	326.00	3.10	1261.00	0.90	555.00	2.40
# 7	2484.00	2.10	631.00	1.50	214.00	0.70	704.00	3.00	2223.00	1.00
# 8	855.00	2.00	136.00	1.20	1500.00	5.00	136.00	1.20	855.00	2.00
# 9	1080.00	1.15	1613.00	0.50	814.00	2.60	344.00	1.40	71.00	5.00
# 10	71.00	5.00	537.00	4.70	704.00	3.00	4378.00	4.30	895.00	3.00
# 11	1338.00	0.50	704.00	3.00	172.00	2.25	171.00	1.10	1080.00	1.15
# 12	631.00	1.50	552.00	1.90	631.00	1.50	172.00	2.25	414.00	4.30
# 13	2083.00	3.40	449.00	2.50	171.00	1.10	2632.00	2.90	2056.00	1.20
# 14	725.00	3.50	814.00	2.60	1150.00	2.80	366.00	1.20	593.00	3.70
# 15	593.00	3.70	1289.00	1.80	215.00	1.70	784.00	2.00	325.00	1.80
# 16	896.00	3.00	1703.00	1.25	449.00	2.50	225.00	3.40	1393.00	1.90
# 17	388.00	1.30	725.00	3.50	800.00	1.05	71.00	5.00	537.00	4.70
# 18	750.00	2.30	157.00	1.10	2484.00	2.10	25.00	3.33	153.00	3.30
# 19	1844.00	3.25	1844.00	3.25	537.00	4.70	345.00	3.50	366.00	1.20
# 20	1289.00	1.80	900.00	3.50	849.00	4.00	157.00	1.10	725.00	3.50
# 21	414.00	4.30	2056.00	1.20	350.00	0.90	552.00	1.90	1208.00	2.10
# 22	1409.00	3.20	171.00	1.10	225.00	3.40	229.00	4.00	157.00	1.10
# 23	350.00	0.90	896.00	3.00	136.00	1.20	326.00	3.10	1703.00	1.25
# 24	345.00	3.50	2223.00	1.00	71.00	5.00	2051.00	1.70	950.00	1.30
# 25	136.00	1.20	1219.00	0.60	2223.00	1.00	1150.00	2.80	750.00	2.30
# 26	1572.00	3.30	25.00	3.30	1572.00	3.30	814.00	2.50	1572.00	3.30
# 27	139.00	3.20	79.00	1.00	725.00	3.50	849.00	4.00	1261.00	0.90
# 28	462.00	1.20	4378.00	4.30	229.00	4.00	2083.00	3.40	462.00	1.20
# 29	163.00	3.30	2632.00	2.90	2632.00	2.90	139.00	3.20	3388.00	2.00
# 30	552.00	1.90	326.00	1.80	555.00	2.40	537.00	4.70	1500.00	5.00
# 31	3388.00	2.00	2484.00	2.10	25.00	3.30	414.00	4.30	1409.00	3.20
# 32	4378.00	4.30	196.00	1.10	628.00	0.80	214.00	0.70	449.00	2.50
# 33	1500.00	5.00	139.00	3.20	366.00	1.20	896.00	3.00	2580.00	2.10
# 34	172.00	2.25	326.00	3.10	1393.00	1.90	163.00	3.30	800.00	1.05
# 35	1150.00	2.80	352.00	4.00	344.00	1.40	950.00	1.30	136.00	1.20
# 36	704.00	3.00	214.00	0.70	608.00	1.50	631.00	1.50	4378.00	4.30
# 37	1261.00	0.90	2580.00	2.10	1080.00	1.15	2056.00	1.20	631.00	1.50
# 38	2223.00	1.00	928.00	2.60	2083.00	3.40	1613.00	0.50	1219.00	0.60
# 39	555.00	2.40	350.00	0.90	388.00	1.30	1572.00	3.30	1150.00	2.80
# 40	1393.00	1.90	849.00	4.00	157.00	1.10	350.00	0.90	79.00	1.00

TABLE IV

COMPUTATIONAL FEATURES OF (1) THE PRIMAL SEARCH ALGORITHM AND (2) THE MODIFIED ALGORITHM OF BROOKS ET AL.

	Number of Variables					
	n=10 M=35000	m=7	n=20 M=65000	m=7	n=40 M=125000	m=7
	(1)	(2)	(1)	(2)	(1)	(2)
Average Execution Time (seconds)	1.40	.35	4.25	.82	18.36	2.22
Range	1.12 - 1.84	.29 - .40	2.88 - 7.21	.67 - 1.07	12.27 - 24.51	2.02 - 2.49
Standard Deviation	.28	.05	1.60	.16	5.02	.18
Average Number of Iterations	6	3	7	3	8	3
Range	5-7	3-3	5-11	3-3	6-11	3-3
Average Number of Steps Within Each Iteration	2	48	2	79	2	148
Range	1-3	34-61	1-3	60-103	1-3	121-185
Average Number of Objective Value Estimations	7	1	8	1	9	1

TABLE IV (Continued)

	Number of Variables					
	n=10 M=35000	m=7	n=20 M=65000	m=7	n=40 M=125000	m=7
	(1)	(2)	(1)	(2)	(1)	(2)
Relative Performance Average Ratio*	1.03	.97	1.01	.99	1.10	.92

*Average of ratios of solutions obtained by (1) to by (2) or by (2) to by (1) respectively. See also Table VI.

TABLE V

SOLUTIONS TO PROBLEMS OF SIZE 80 BY (1) THE PRIMAL SEARCH ALGORITHM AND (2) THE MODIFIED ALGORITHM OF BROOKS ET AL.

PROBLEM NUMBER: 1

ITEM	COST/UNIT	DEMAND	(1)	(2)
# 1	2980.00	2.10	3	3
# 2	1751.00	1.50	3	2
# 3	462.00	1.20	3	2
# 4	1902.00	5.00	8	8
# 5	345.00	3.50	7	6
# 6	1289.00	1.80	3	3
# 7	1844.00	3.25	5	5
# 8	896.00	3.00	6	5
# 9	388.00	1.30	3	3
# 10	3430.00	12.00	16	17
# 11	1338.00	0.50	1	1
# 12	855.00	2.00	4	4
# 13	376.00	7.10	13	12
# 14	2370.00	8.50	12	13
# 15	3898.00	9.30	13	13
# 16	71.00	5.00	11	9
# 17	326.00	1.80	4	4
# 18	225.00	3.40	8	6
# 19	411.00	6.50	12	11
# 20	814.00	2.60	5	5
# 21	124.00	11.30	21	18
# 22	1240.00	6.00	10	10
# 23	2223.00	1.00	2	1
# 24	659.00	8.30	14	13
# 25	950.00	1.30	3	2
# 26	1404.00	7.00	11	11
# 27	257.00	5.50	11	10
# 28	631.00	1.50	3	3
# 29	379.00	7.00	13	12
# 30	1150.00	2.80	5	5
# 31	326.00	3.10	7	6
# 32	2775.00	11.50	16	17
# 33	196.00	13.00	22	20
# 34	2083.00	3.40	5	6
# 35	414.00	4.30	9	8
# 36	784.00	5.10	9	9
# 37	171.00	1.10	3	3
# 38	196.00	1.10	3	3
# 39	593.00	3.70	7	7
# 40	139.00	3.20	8	6
# 41	3388.00	2.00	3	3
# 42	784.00	2.00	4	4
# 43	8500.00	6.90	8	9
# 44	2632.00	2.90	4	5
# 45	1090.00	8.00	13	13
# 46	750.00	2.30	5	4
# 47	1393.00	1.90	4	3
# 48	2056.00	1.20	2	2
# 49	1613.00	0.50	1	1
# 50	1830.00	11.80	17	17
# 51	352.00	4.00	8	7
# 52	172.00	2.25	6	5
# 53	163.00	3.30	8	6
# 54	1572.00	3.30	6	6
# 55	555.00	2.40	5	4
# 56	344.00	1.40	4	3
# 57	1244.00	8.70	14	14
# 58	157.00	1.10	3	3
# 59	863.00	6.70	11	11
# 60	1121.00	9.80	15	15
# 61	863.00	1.30	3	2
# 62	1080.00	1.15	2	2
# 63	849.00	4.00	7	7
# 64	800.00	1.05	2	2
# 65	214.00	3.70	2	2
# 66	3007.00	6.80	10	10
# 67	1261.00	0.90	2	1
# 68	675.00	12.00	19	18
# 69	229.00	4.00	9	7
# 70	2484.00	2.10	3	3
# 71	704.00	3.00	6	5
# 72	136.00	1.20	4	3
# 73	937.00	4.70	9	8
# 74	4378.00	4.30	6	6
# 75	796.00	10.00	16	16
# 76	628.00	0.80	2	1
# 77	1449.00	3.00	5	5
# 78	725.00	3.50	7	6
# 79	350.00	0.90	3	2
# 80	475.00	10.40	17	16

ERRORS(X)=	2.85581	2.74865
COST(X)=	699942.00	699939.00
EXECUTION TIME: (IN SECONDS)	98.19	16.53

TABLE V (Continued)

PROBLEM NUMBER= 2

ITEM	COST/UNIT	DEMAND	(1)	(2)
# 1	25.00	3.30	9	9
# 2	1289.00	1.80	3	3
# 3	1830.00	11.80	17	17
# 4	1613.00	0.50	0	1
# 5	704.00	3.00	6	5
# 6	1500.00	5.60	8	8
# 7	628.00	0.80	2	1
# 8	196.00	13.00	22	20
# 9	900.00	3.50	6	6
# 10	4378.00	4.30	6	6
# 11	1572.00	3.30	5	5
# 12	1208.00	2.10	4	3
# 13	366.00	1.20	3	2
# 14	352.00	4.00	8	7
# 15	2580.00	2.10	3	3
# 16	725.00	3.50	7	6
# 17	1080.00	1.15	2	2
# 18	1844.00	3.25	5	5
# 19	2775.30	11.50	16	17
# 20	124.60	11.30	21	18
# 21	79.00	1.00	3	3
# 22	376.00	7.10	13	12
# 23	863.00	1.30	3	2
# 24	3430.00	12.00	16	19
# 25	171.00	1.10	3	3
# 26	1279.00	7.20	11	11
# 27	2566.00	9.00	13	13
# 28	36.00	10.00	20	18
# 29	414.00	4.30	8	7
# 30	593.00	3.73	7	6
# 31	8500.00	6.90	8	9
# 32	2223.00	1.00	1	1
# 33	950.00	1.30	2	2
# 34	896.00	3.00	5	5
# 35	784.00	2.00	4	3
# 36	2484.00	2.10	3	3
# 37	388.00	1.30	3	2
# 38	659.00	8.30	14	13
# 39	2051.00	1.70	3	2
# 40	136.00	1.20	4	3
# 41	1751.00	1.50	2	2
# 42	344.00	1.40	3	3
# 43	1219.00	0.60	1	1
# 44	750.00	2.30	4	4
# 45	1246.00	8.70	13	13
# 46	800.00	1.05	2	2
# 47	71.00	5.00	11	9
# 48	631.00	1.50	3	2
# 49	855.00	2.00	4	3
# 50	475.00	10.40	17	16
# 51	1703.00	1.25	2	2
# 52	163.00	3.30	8	6
# 53	537.00	4.70	9	8
# 54	638.00	1.50	3	3
# 55	2632.00	2.90	4	4
# 56	379.00	7.00	13	11
# 57	675.00	12.30	19	18
# 58	411.00	6.50	12	11
# 59	1090.00	8.03	13	12
# 60	462.00	1.20	3	2
# 61	928.00	2.60	5	4
# 62	3898.00	9.30	13	13
# 63	569.00	5.30	10	9
# 64	555.00	2.40	5	4
# 65	229.00	4.00	8	7
# 66	345.00	3.50	7	6
# 67	796.00	10.00	16	15
# 68	3007.00	6.80	10	10
# 69	257.00	5.50	11	9
# 70	1560.00	12.50	18	18
# 71	1121.00	9.80	15	15
# 72	350.00	0.90	2	2
# 73	849.00	4.60	7	7
# 74	1240.00	6.00	10	9
# 75	552.00	1.93	4	3
# 76	3388.00	2.00	3	3
# 77	2056.00	1.20	2	2
# 78	1338.00	0.50	1	1
# 79	863.00	6.70	11	11
# 80	1261.00	0.90	1	1

ERRORS(X)=	3.07850	2.90754
COST(X)=	699995.00	599990.00
EXECUTION TIME (IN SECONDS)	113.79	16.99

TABLE V (Continued)

PROBLEM NUMBER: 3

ITEM	COST/UNIT	DEMAND	(1)	(2)
# 1	608.00	1.50	3	3
# 2	229.00	4.00	9	7
# 3	906.00	3.50	6	6
# 4	628.00	3.80	2	1
# 5	1404.00	7.00	11	11
# 6	1208.00	2.10	4	4
# 7	1613.00	0.50	1	1
# 8	475.00	10.40	17	16
# 9	2580.00	2.10	3	3
# 10	1261.00	0.93	2	1
# 11	350.00	0.90	2	2
# 12	863.00	6.70	11	11
# 13	366.00	1.20	3	2
# 14	1080.00	1.15	2	2
# 15	1449.00	3.00	5	5
# 16	257.00	5.50	11	9
# 17	388.00	1.30	3	2
# 18	345.00	3.50	7	6
# 19	1219.00	0.60	1	1
# 20	2775.00	11.50	16	16
# 21	896.00	3.00	6	5
# 22	449.00	2.50	5	5
# 23	631.00	1.50	3	3
# 24	3007.00	6.80	10	10
# 25	414.00	4.30	9	8
# 26	215.00	1.70	5	3
# 27	71.00	5.00	12	10
# 28	3430.00	12.00	16	17
# 29	2066.00	9.00	13	13
# 30	1409.00	3.20	5	5
# 31	1289.00	1.80	3	3
# 32	171.00	1.10	3	2
# 33	950.00	1.30	3	2
# 34	1751.00	1.50	3	2
# 35	593.00	3.70	7	6
# 36	1240.00	6.00	10	10
# 37	2053.00	11.00	16	16
# 38	2056.00	1.20	2	2
# 39	569.00	5.30	10	9
# 40	796.00	10.00	16	16
# 41	157.00	1.10	3	2
# 42	753.00	2.30	5	4
# 43	855.00	2.00	4	4
# 44	225.00	3.40	8	6
# 45	1246.00	8.70	14	13
# 46	214.00	0.70	2	1
# 47	4378.00	4.30	6	6
# 48	163.00	3.30	8	6
# 49	139.00	3.20	8	6
# 50	2223.00	1.00	1	1
# 51	124.00	11.30	20	18
# 52	196.00	1.10	3	2
# 53	2484.00	2.10	3	3
# 54	462.00	1.20	3	2
# 55	36.00	10.00	20	17
# 56	1844.00	3.25	5	5
# 57	459.00	8.30	14	13
# 58	411.00	6.50	12	11
# 59	8500.00	6.90	8	9
# 60	79.00	1.00	4	3
# 61	725.00	3.50	7	6
# 62	552.00	1.90	4	3
# 63	863.00	1.30	3	2
# 64	1500.00	5.00	8	8
# 65	1560.00	12.50	18	21
# 66	379.00	7.00	13	12
# 67	1830.00	11.80	17	18
# 68	675.00	12.00	19	18
# 69	255.00	2.40	5	4
# 70	1150.00	2.80	5	5
# 71	172.00	2.25	6	4
# 72	784.00	2.00	4	4
# 73	2632.00	2.90	4	5
# 74	196.00	13.60	22	20
# 75	1338.00	0.50	1	1
# 76	814.00	2.60	5	5
# 77	3898.00	9.30	13	13
# 78	2083.00	3.40	5	5
# 79	326.00	1.80	4	3
# 80	344.00	1.40	4	3

E(N)RS/X(I)=	2.95023	2.78927
COST(X)=	699974.00	699992.00
EXECUTION TIME: (IN SECONDS)	118.14	16.41

TABLE V (Continued)

PROBLEM NUMBER: 4

ITEM	COST/UNIT	DEMAND	(1)	(2)
# 1	4378.00	4.30	6	7
# 2	345.00	3.50	8	6
# 3	163.00	3.30	8	6
# 4	1565.00	12.50	19	25
# 5	796.00	10.00	16	16
# 6	593.00	3.70	7	7
# 7	1449.00	3.00	5	5
# 8	704.00	3.00	6	5
# 9	855.00	2.00	4	4
# 10	537.00	4.70	9	8
# 11	1240.00	6.00	10	10
# 12	552.00	1.90	4	4
# 13	1844.00	3.25	6	5
# 14	366.00	1.20	3	2
# 15	196.00	1.10	3	2
# 16	2051.00	1.70	3	3
# 17	928.00	2.60	5	5
# 18	1338.00	0.50	1	1
# 19	1219.00	0.60	1	1
# 20	2053.00	11.00	16	16
# 21	2484.00	2.10	3	3
# 22	3898.00	9.30	13	14
# 23	1613.00	0.50	1	1
# 24	1080.00	1.15	2	2
# 25	414.00	4.33	9	8
# 26	3007.00	6.80	10	10
# 27	1500.00	5.00	8	8
# 28	555.00	2.40	5	4
# 29	2632.00	2.90	5	5
# 30	214.00	0.70	2	1
# 31	124.00	11.30	21	18
# 32	449.00	2.50	6	5
# 33	1393.00	1.90	4	3
# 34	800.00	1.05	3	2
# 35	1409.00	3.20	6	5
# 36	215.00	1.70	5	3
# 37	950.00	1.30	3	2
# 38	411.00	6.50	12	11
# 39	326.00	1.80	5	3
# 40	326.00	3.10	7	6
# 41	1703.00	1.25	2	2
# 42	3430.00	12.00	17	17
# 43	1751.00	1.50	3	2
# 44	2370.00	8.50	13	13
# 45	225.00	3.40	8	6
# 46	71.00	5.00	12	10
# 47	388.00	1.30	4	2
# 48	139.00	3.20	8	6
# 49	353.00	0.90	3	2
# 50	1830.00	11.80	17	23
# 51	229.00	4.00	9	7
# 52	376.00	7.10	13	12
# 53	2083.00	3.40	6	5
# 54	1289.00	1.80	4	3
# 55	79.00	1.00	4	3
# 56	569.00	5.30	10	9
# 57	172.00	2.25	6	4
# 58	844.00	4.00	8	7
# 59	171.00	1.10	4	2
# 60	3388.00	2.00	3	3
# 61	36.00	10.00	22	17
# 62	784.00	5.10	9	9
# 63	863.00	6.70	12	11
# 64	2580.00	2.10	3	3
# 65	896.00	3.00	6	5
# 66	25.00	3.30	10	7
# 67	379.00	7.00	13	12
# 68	1090.00	8.00	13	13
# 69	1150.00	2.80	5	5
# 70	462.00	1.20	3	2
# 71	1246.00	8.70	14	13
# 72	157.00	1.10	4	2
# 73	257.00	5.90	11	10
# 74	8500.00	6.90	8	9
# 75	1208.00	2.10	4	4
# 76	1572.00	3.30	6	5
# 77	352.00	4.00	8	7
# 78	196.00	13.00	23	20
# 79	784.00	2.00	4	4
# 80	1279.00	7.20	12	11

E(NDRS/X)=	2.80987	2.52640
COST(X)=	699987.00	699998.00
EXECUTION TIME (IN SECONDS)	139.97	17.05

TABLE VI

SOLUTIONS OF SAMPLE PROBLEMS IN TABLE III
BY (1) PRIMAL SEARCH ALGORITHM AND
BY (2) MODIFIED ALGORITHM OF
BROOKS ET AL.

E(NORS/X)		Problem #1	Problem #2	Problem #3	Problem #4	Problem #5
n = 10	(1)	.23888	.58207	.62086	2.32702	1.79822
	(2)	.23888	.53143	.58712	2.33009	1.79822
n = 20	(1)	1.68634	2.56705	2.37961	1.23491	2.81074
	(2)	1.66472	2.56511	2.37962	1.22914	2.71815
n = 40	(1)	3.11343	3.15778	2.98825	2.27049	3.75845
	(2)	3.04320	3.08386	2.69192	2.13480	2.98737

Average ratio is the average of the ratios of the solutions found by one method to the solutions found by the other. If the ratio is less than one, then the algorithm is producing better results than the other one. Comparison of the ratios in Table IV (also see Table VI) indicates that the Modified Algorithm of Brooks et al. produced better solutions on the average. Since the ratios are close to each other, the algorithms generate good solutions.

Execution time of the algorithms are also affected by the magnitude of the resource, the algorithms take longer to find a solution to the problem as the magnitude increases. The relation between the magnitude of the resource and the execution time seems linear though. In Figure 3, the affect of the magnitude of the resource on the execution time of the Modified Algorithm of Brooks et al. is depicted for the Problem #1 of Table VI. For this particular problem, the regression equation is

$$T = (.000028467)\ (\text{resource}) - .365589. \qquad (4.7)$$

For every problem and the algorithm the regression analysis needs to be worked out individually, since the coefficients of the regression line are heavily dependent on the problem under consideration and the algorithm used. The coefficients in equation (4.7) cannot be applied in general.

An advantage of the Modified Algorithm of Brooks et al. is that it does not require the user to determine some parameters as in the Primal Search Algorithm. Evidently the selection of such parameters effect the execution time. Thus the Modified Algorithm of Brooks et al. is not subject to variational affects of parameter selection by the user.

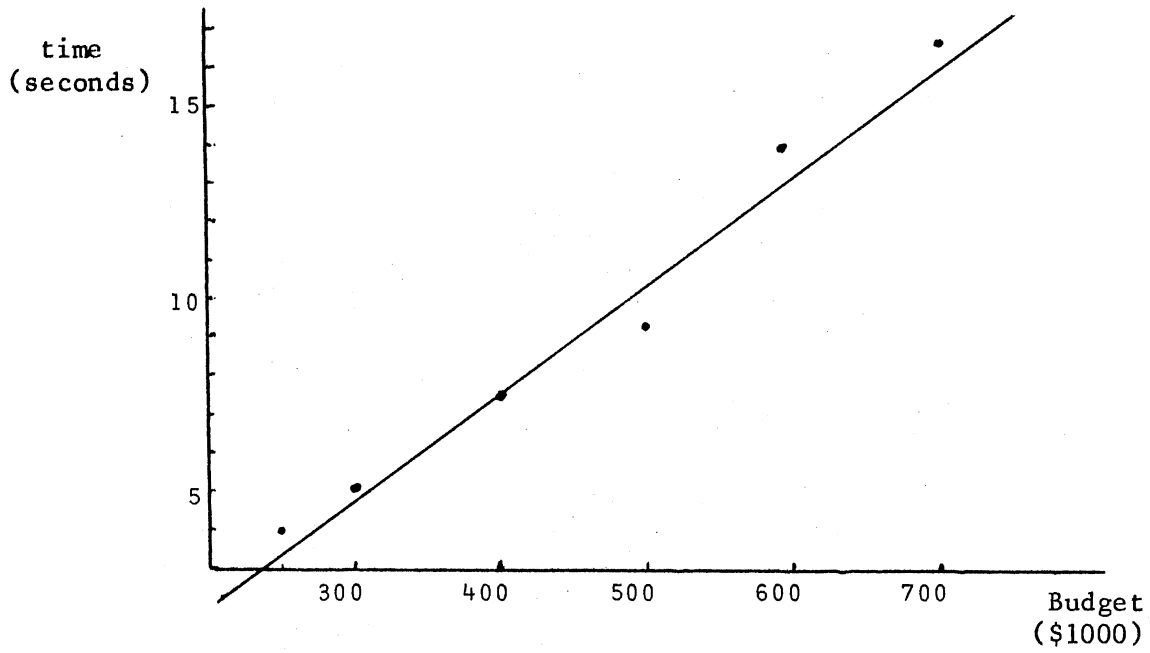

Figure 3. The Effect of the Size of Resource on the Execution Time (The Modified) Algorithm of Brooks et al. is Used)

The Primal Search Method has two parameters p and the Lagrange multiplier λ. In order to measure their effect on the execution time the example problem of Chapter V, Remarks on Computational Experience, was run with different parameter values. The results are summarized in the Figures 4 and 5.

Both of the algorithms produce near optimal solutions. In fact the solutions which are generated by both are very close to each other. The values of objective functions of the sample problems in Table III are given in Table VI. In some instances the Primal Search Algorithm has produced better solutions than the Modifed Algorithm of Brooks et al., and sometimes the latter has produced better solutions than the former.

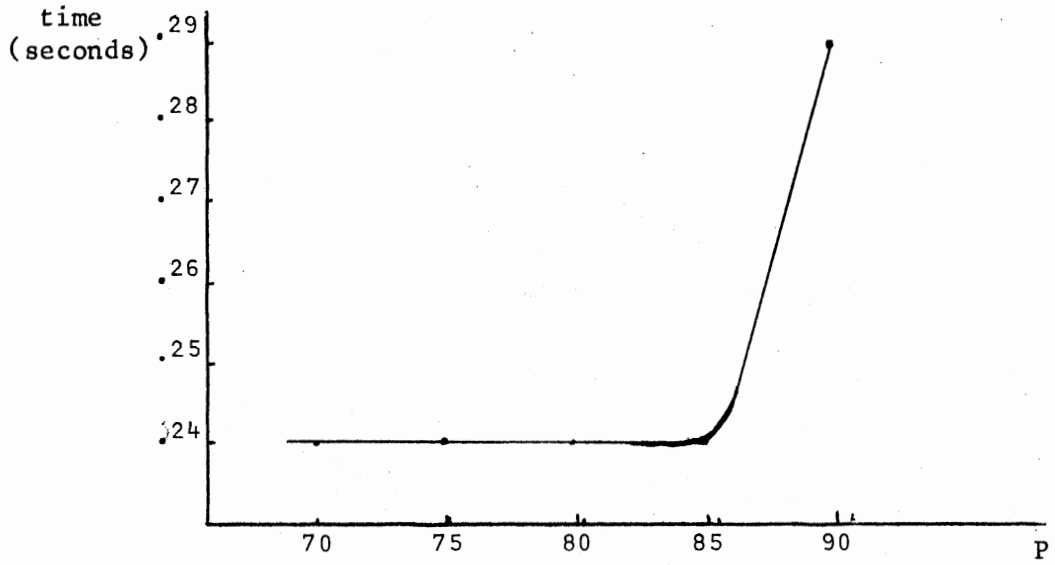

Figure 4. The Effect of p on the Execution Time, λ = .0002 for the Primal Search Algorithm

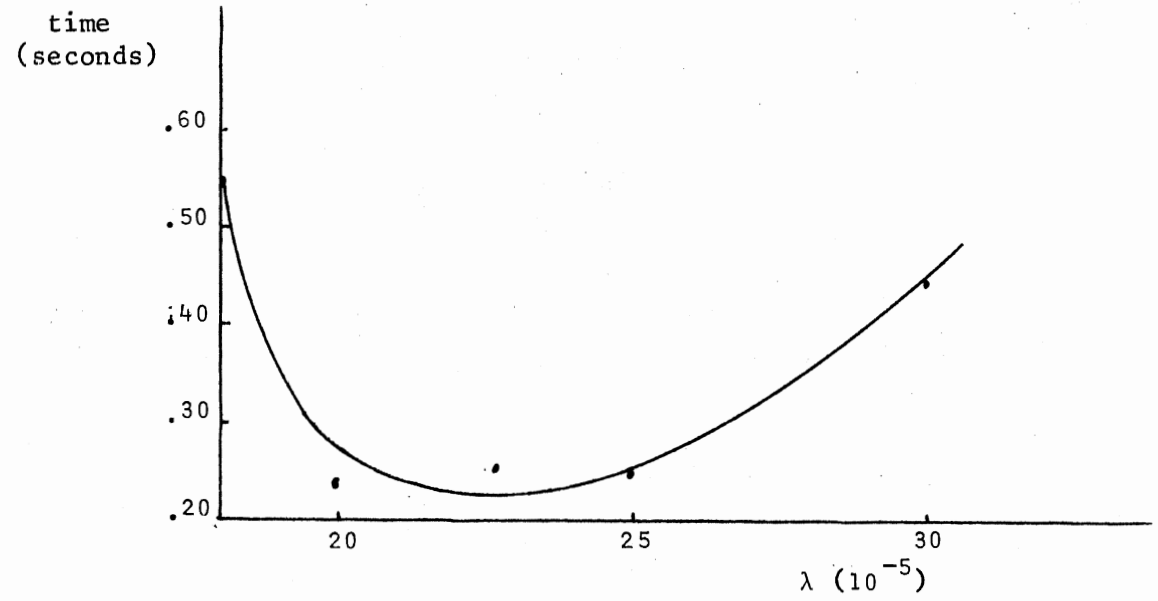

Figure 5. The Effect of the Lagrange Multiplier With p = .85 for the Primal Search Algorithm

The Primal Search Algorithm contains some parameters to be determined by the user heuristically, then it can miss the optimal, but can generate a near optimal solution. On the other hand, the other method requires the NORS function be transformed into a separable function. Obviously the transformation slightly alters the shape of the NORS function, consequently this algorithm may not be able to find the optimal, but produces a near optimal solutions. Another observation is that this algorithm produces better solutions than the other one as the size of the problems increases (see Tables VI and VII). The reason would be that the number of points in the region to be explored by the Primal Search Algorithm increases as the problem size increases. The method searches only the points along the favorable direction, and cannot reach some others. On the other hand, the algorithm lacks the flexibility to readjust the search direction in a most effective way as in the continuous problems. Consequently, it produces only near optimal solutions.

Testing the Solutions by Simulation

All the efforts up to now are focused on finding a solution for the War Readiness Spares Kit Problem. The solutions obtained by either method are near optimal solutions. In fact the optimality of the solutions is not guaranteed due to discrete characteristics of the problem. Consequently the question that comes to mind is how good are the solutions. One way of answering this question is to simulate the operation of a squadron of airplanes with a spares kit obtained by one of the previous methods. More specifically the objective function is evaluated by simulation instead of the mathematical expression (4.2).

A simulation model was developed by GASP IV. The system status of

this model is the Number of Not Operationally Ready Supply airplanes due to lack of spare parts. The number of NORS is a time persistent variable, hence TIMST is used to evaluate $E(NORS/X) \sum_{k=0}^{n} kp_k(X)$ where n is the number of airplanes in the squadron and $p_k(X)$ is the percentage of time when k = 0, 1, ..., n planes are NORS with a spares kit X. Simulation model collects statistical data on the number of NORS airplanes and $p_k(X)$, and produces the mean and the time dependent standard deviation for E(NORS/X).

The entities of the simulation model are the spare parts in the kit and the airplanes demanding the spare parts. Mean time between failures of an item on any airplane is assumed to be exponentially distributed. An airplane fails if any part in it fails, and only one part fails at a time. If the failed part is available in the kit, that plane is repaired. If the needed part is not available, but there are also one other plane awaiting repairs, one of them is cannibalized at the end of the flying day of two flying hours to repair the other by consolidating the necessary parts. Otherwise the airplane is put into the repair queue. Consequently, the events of the simulation model are the breakdown of an airplane, the cannibalization of an airplane, and reinitialization of the kit and the system status at the end of every flying period of 60 flying hours.

Two files were used in the simulation model. The first file was the event file and the second file was the repair queue.

File (1) event file

attribute (1) time of the incident

attribute (2) event code - 1. Breakdown

2. Cannibalization

3. Reinitialization

attribute (3) number of the grounded plane

attribute (4) number of the failed part in attribute (3)

File (2) repair queue

attribute (3) as above

attribute (4) as above

The FORTRAN Listings of the Simulation Model is provided in the Appendix G.

There is a slight but important difference between the problem definition used in the simulation analysis and the War Readiness Spares Kit Problem of Chapter IV. This difference lies in the assumption of the cannibalization of the airplanes. In the simulation model, when there are only two planes awaiting repairs, then one is cannibalized to repair the other one. This is not the case in general. For instance, if the cannibalization policy requires at least three planes be grounded before the cannibalization, then the second cannibalization will take place when a second set or three planes are grounded due to lack of spare parts. On the other hand, in the simulation model the second cannibalization will occur when a second set of two planes are grounded due to lack of spare parts. Consequently the cannibalization policy has an impact on the result. It was not known what kind of policy was used by U.S. Air Force, when this policy was adapted. Furthermore, simulation results turn out to be biased estimates of E(NORS/X).

Simulation produces statistical estimates for E(NORS/X), namely a mean value $\tilde{E}$(NORS/X).

Four problems with 80 parts (see Table V) for a squadron of six planes were studied by analytical methods and simulation. The operation of the squadron was simulated by using spares kits obtained by analytical methods for a time horizon of 20 flying periods, or as a total of 1200 flying hours. The analytical and simulation results are summarized in Table VII. The maximum number of NORS airplanes at any period of sixty flying hours was five (see Appendix G).

From the simulation and analytical results, the average bias

$$\bar{b} = (3.65 - 2.86 + 3.77 - 3.08 + 3.77 - 2.95 + 3.71 - 2.81 + 3.73 - 2.75 + 3.79 - 2.91 + 3.71 - 2.79 + 3.67 - 2.53)$$

$$\bar{b} = .89$$

and standard deviation $\bar{b} = 1.34$.

Table VII indicates that the analytical results lie within one standard deviation of $\tilde{E}$(NORS/X). In other words, the simulation results, after the adjustment for the bias resulting from the cannibalization policy confirm the analytical results. Therefore the conclusion is that the analytical methods can indeed produce good solutions to the problems.

TABLE VII

SIMULATION OF THE KITS OF TABLE V

	The Primal Search Method	Simulation of the Solutions by the Primal Search Method		Modified Solution Method of Brooks et al.	Simulation of the Solutions by the Modified Algorithm of Brooks et al.	
	$E(NORS/X)$	$\tilde{E}(NORS/X)$	σ	$E(NORS/X)$	$\tilde{E}(NORS/X)$	σ
Problem #1		3.65			3.73	
	2.86	2.76*	.40	2.75	2.84*	.42
Problem #2		3.77			3.79	
	3.08	2.88*	.40	2.91	2.90*	.42
Problem #3		3.77			3.71	
	2.95	2.88*	.41	2.79	2.82*	.42
Problem #4		3.72			3.67	
	2.81	2.83*	.42	2.53	2.78*	.42

*$\bar{E}(NORS/X)$ values after adjustment of biasedness; $E(NORS/X) = \tilde{E}(NORS/X) - \bar{b}$

σ is equal to $1/\sqrt{20}$ of the standard deviation obtained in the simulation.

CHAPTER V

CONCLUSIONS

General

The main objective of this research was to develop an efficient algorithm and/or modify the existing methods to solve single resource allocation problems with a nonseparable objective function. A primal search algorithm was proposed after a thorough investigation of the mathematical properties of this class of problems. It makes use of monotonicity and discrete convexity of the objective function, and contains heuristic rules. On the other hand, the solution method which was proposed by Brooks et al. [3] was modified successfully. This approach requires that the objective function be transformed into a separable function. The third approach was to express the single resource allocation problem as a saddle point problem and to find the saddle point solution.

Remarks On Computational Experience

Computational experience revealed that the modified solution method of Brooks et al. was computationally superior to the other two and produced good solutions. The primal search algorithm produced good solutions too, but was slower and less accurate in solving large size problems. The third approach suffers from the duality gap, and it is very slow comparatively to the other two. Hence it was not used for comparison. An

immediate consequence of this study is that the modified solution method of Brooks et al. can be used to solve large size War Readiness Spares Kit Problems efficiently and accurately.

The simulation was proposed only to test the accuracy of the analytical results. This was an attempt to validate these results by simulating the operations of a squadron. However, some of the assumptions of the simulation were not exactly in compliance with the actual operations.

The cannibalization policy was such an assumption, hence it introduced some biasedness in the simulation results. Since a thorough investigation of this policy was beyond the scope of this research, the biasedness was resolved heuristically by calculating the average of differences between the simulation and analytical results. Fortunately, the simulation results are in confirmation with the analytical results. In other words, the analytical results can indeed produce good solutions to the War Readiness Spares Kit Problem.

Further Research

There are several potential areas of research as a consequence of this study. One of the assumptions of the War Readiness Spares Kit Problem is that the demands of the spare parts are independently distributed. The validity of this assumption should be questioned. In fact, the demand rates and their dependency should be investigated so that joint distribution function can be developed. It would be gratifying to develop the NORS function with dependently distributed demands and solution methods to solve the War Readiness Spares Kit Problem. Another research area is the effects of the different cannibilization policies by simulation. A thorough understanding and knowledge of operations of

squadrons in remote bases is essential for this purpose. The Primal Search Algorithm takes to many iterations to converge to a solution. There is a possibility that it might be improved in such a way as to reduce the number of iterations resulting in increased computational efficiency.

REFERENCES

1. Bellman, R. E. and S. E. Dreyfus. Applied Dynamic Programming. Princeton, New Jersey: Princeton University Press, 1962.

2. Brooks, R. B. S. and A. Geoffrion. "Finding Everett's Lagrange Multipliers by Linear Programming." Operations Research, Vol. 14 (1966), pp. 1149-1152.

3. Brooks, R. B. S., C. A. Gillen, and J. Y. Lu. Alternative Measures of Supply Performance: Fills, Backorders, Operational Rate, and NORS. Santa Monica, California: The RAND Corporation, RM-6094-PR, August 1969.

4. Churchman, C. W., R. L. Ackoff, and E. L. Arnoff. Introduction to Operations Research. New York, New York: John Wiley and Sons, Inc., 1957.

5. Fox, B. "Discrete Optimization Via Marginal Analysis." Management Science, Vol. 13 (1966), pp. 210-216.

6. Garfinkel, R. S. and G. L. Nemhauser. Integer Programming. New York, New York: John Wiley and Sons, Inc., 1972.

7. Hillier, R. S. "Efficient Heuristic Procedures for Integer Linear Programming With an Interior." Operations Research, Vol. 17 (1969), pp. 600-638.

8. _____. "A Bound-and-Scan Algorithm for Pure Integer Linear Programming With General Variables." Operations Research, Vol. 17 (1969), pp. 639-680.

9. Hu, T. C. and S. M. Robinson (Eds.). Mathematical Programming. New York, New York: Academic Press, Inc., 1973.

10. Lasdon, L. S. Optimization Theory for Large Systems. New York, New York: MacMillan, 1970.

11. Luenberger, D. G. Introduction to Linear and Nonlinear Programming. Reading, Massachusetts: Addison-Wesley Publishing Co., Inc., 1973.

12. Miller, B. L. Unconstrained Optimization in the Integers. Santa Monica, California: The RAND Corporation, RM-6165-PR, January 1970.

13. Pritsker, A. A. B. The GASP IV Simulation Language. New York, New York: John Wiley and Sons, Inc., 1974.

14. Reiter, S. and D. B. Rice. "Discrete Optimizing Solution Procedures for Integer Linear and Nonlinear Programming Problems." Management Science, Vol. 12 (1966), pp. 829-850.

15. Rosen, J. B., O. L. Managasarian, and K. Ritter (Eds.). Nonlinear Programming. New York, New York: Academic Press, Inc., 1970.

16. Zangwill, W. I. Nonlinear Programming: A Unified Approach. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1969.

17. Zoutendijk, G. Method of Feasible Directions. Amsterdam, New York: Elsevier, 1960.

APPENDIXES

APPENDIX A

A TRANSFORMATION OF THE E(NORS/X) FUNCTION INTO A SEPARABLE FUNCTION

$$E(NORS/X) = \sum_{j=0}^{m} \left(1 - \prod_{i=1}^{n} F_i\,(x_i + j)\right) + \sum_{j=m+1}^{\infty} \sum_{i=1}^{n} \left(1 - F_i\,(x_i + j)\right) \quad (A.1)$$

$$= m+1 - \sum_{j=0}^{m} \prod_{i=1}^{n} F_i\,(x_i + j) + \sum_{j=m+1}^{\infty} \sum_{i+1}^{n} \left(1 - F_i\,(x_i + j)\right) \quad (A.2)$$

Let $b_j = \prod_{i=1}^{n} F_i\,(x'_i + j)$ for $j = 0, 1, \ldots, m$ and some

$$X' = (x'_1, x'_2, \ldots, x'_n).$$

Then $$\sum_{j=0}^{m} \prod_{i=1}^{n} F_i\,(x_i + j) \simeq \sum_{j=0}^{m} b_j \sum_{i=1}^{n} \log F_i\,(x_i + j) \quad (A.3)$$

Since $(m + 1)$ is constant and the term $\sum_{j=m+1}^{\infty} \sum_{i=1}^{n} \left(1 - F_i\,(x_i + j)\right)$

is negligible if m is sufficiently large,

therefore, $E(NORS/X) \simeq - \sum_{i=1}^{n} \sum_{j=0}^{m} b_j \log F_i\,(x_i + j)$ which is a (A.4)

separable function. The problem 3.1 will then be restated as

$$\max \sum_{i=1}^{n} \sum_{j=0}^{m} b_j \log F_i\,(x_i + j) \quad (A.5)$$

subject to $$\sum_{i=1}^{n} c_i x_i \leq M \quad (A.6)$$

$$x_i \geq 0 \text{ integer for } i = 1, 2, \ldots, n.$$

It suffices to prove that (A.5) is strictly increasing to justify the implementation of marginal analysis by Theorem 4 of [5]. Obviously a separable function is strictly increasing if it is so in each component individually. Thus, it is required to show that

$$\sum_{j=0}^{m} b_j \log F_i(x_i + j)$$

is strictly increasing for each i=1, ..., n.

Let $x_i < y_i$ for some i=1, ..., n, then for any j=0, 1, ..., m $F_i(x_i + j) \leq F_i(y_i + j)$ since $F_i(\cdot)$ is the cumulative sums of a Poisson Distribution with mean μ_i. Consequently

$$\frac{F_i(y_i + j)}{F_i(x_i + j)} \geq 1, \text{ and } \log \frac{F_i(y_i + j)}{F_i(x_i + j)} \geq 0 \tag{A.7}$$

On the other hand;

$$\Delta_i = \sum_{j=0}^{m} b_j \log F_i(y_i + j) - \sum_{j=0}^{m} b_j \log F_i(x_i + j) \tag{A.8}$$

is equal to $\Delta_i = \sum_{j=0}^{m} b_j \log \frac{F_i(y_i + j)}{F_i(x_i + j)}$. Therefore $\Delta_i \geq 0$

for all i=1, ..., n, because $\log \frac{F_i(y_i + j)}{F_i(x_i + j)} \geq 0$ for all i=1, ..., n,

and j=0, ..., m and $b_j \geq$ for all j=0, ..., m.

The conclusion is then $\sum_{j=0}^{m} b_j \log F_i(x_i + j)$ is strictly increasing and further (A.5) is strictly increasing.

The convergence of the algorithm of Chapter III under The Modified Solution Method of Brooks et al., was proved in Theorem 2 of [3].

APPENDIX B

AN EXAMPLE PROBLEM SOLUTION BY

B. L. MILLER'S ALGORITHM

The Example of Section 4b will be solved by B. L. Miller's Algorithm with the Lagrange Multiplier $\lambda = .0001$. The Lagrangian $L(X;\lambda) = f(X) + \lambda(CX - M)$ is used in this discussion. Let $g(X)$ represent the function $L(X;\lambda)$ for a fixed λ.

Step 1. Initialization: An initial point is obtained by solving $-1 + F_i(\omega_i - 1) + \lambda C_i \leq 0$ and $-1 + F_i(\omega_i) + \lambda C_i > 0$ and setting $X_i = \max(0, \omega_i - k)$ for all $i=1, \ldots, n$ where k is an input parameter and chosen as 1 in this particular example. Since $-1 + F_1(2) + (.0001)(2980) = -1 + .6496 + .2980 = -.0524$ and $-1 + F_1(3) + (.0001)(2980) = -1 + .8386 + .2980 = .1366$, then $\omega_1 = 3$, consequently $x_1^o = 2$. Repeating the same argument for each component yields that $x_2^o = 2$, $x_3^o = 2$, $x_4^o = 6$, and $x_5^o = 6$.

Step 2. Finding A Stationary Vector: Starting from $X^o = (2, 2, 2, 6, 6)$, a point X is sought such that $\Delta_i g(X - e_i) \leq 0$ and $\Delta_i g(X) > 0$ for all $i=1, \ldots, 5$. For instance for item 5, $\Delta_5 g(2, 2, 2, 6, 6)$ is calculated as follows; $\Delta_5 g(2, 2, 2, 6, 6) = \lambda C_5 + F_5(6 + 5 + 1) - 1.$

$$- \sum_{j=0}^{5} \left(F_5(6 + j + 1) - F_5(6 + j)\right) \prod_{i=1}^{4} F_i(X_i + j)$$

j	$F_1(2+j)$	$F_2(2+j)$	$F_3(2+j)$	$F_4(6+j)$	$F_5(7+j) - F_5(6+j)$	Product
0	.64963	.80885	.87949	.76218	.03855	.01358
1	.83864	.93436	.96623	.86663	.01687	.01107
2	.93787	.98142	.99225	.93191	.00656	.00558
3	.97955	.99554	.99850	.96817	.00230	.00217
4	.99414	.99907	.99975	.98630	.00073	.00072
5	.99851	.99983	.99996	.99455	.00021	.00020
					SUM =	.03332

Then Δ_5 g(2, 2, 2, 6, 6) = .0345 + .99992 - 1. -.03332 = .00110. Since it is positive, x_5 = 6. This argument is repeated for each item to obtain a stationary vector X = (2, 2, 2, 6, 6) with f(X) = 1.28241, cost (X) = 21456, and g(X) = 3.42801.

Step 3. A Heuristic Approach to obtain improvement in "Up-phase": Let $S_i = \sum_{k=i}^{n} e_k - e_i$, then the ratios Δ_i g(X + S_i)/Δ_i g(X) are ranked in ascending order, and the vectors of the form X + U_i will be tested for improvement where U_i is the vector with +1 for variables ranked from 1 to i and 0 for the ones ranked i + 1 to n. Calculation of the ratios gives Δ_1 g(X + S_1)/Δ_1 g(X) = -5.61, Δ_2 g(X + S_2)/ Δ_2 g(X) = .15, Δ_3 g(X + S_3)/Δ_3 g(X) = 1.47, Δ_4 g(X + S_4)/Δ_4 g(X) = 1.44 and Δ_5 g(X + S_5)/Δ_5 g(X) = 1.64. The ordering according to the magnitudes yields 1, 2, 4, 3, 5 respectively. For instance, for i = 4 U_4 = (1, 1, 0, 1, 0) and X + U_4 = (3, 3, 2, 7, 6), but this point does not yield a better solution, since g(X + U_4) = 3.48107 is greater than g(X) = 3.42801. Repeating the same argument for each component results that the current solution is stationary.

The same argument is repeated for the "down-phase" too. This does not yield a better solution either.

Step 4. A Second Initialization and Reducing the Set of Potential Improving Points.

First the up-phase is implemented. For instance for i = 1, the vector (0, 1, 1, 1, 1) will be checked for improvement by looking at Δ_1 g(2, 3, 3, 7, 7). Since Δ_1 g(2, 3, 3, 7, 7) = -.028 is negative, the test fails. For i = 2, the vector (1, 0, 1, 1, 1) is tested and Δ_2 g(3, 2, 3, 7, 7) = .005, then the conclusion is that x_2^* can not be 3. Next the vector (1, 0, 0, 1, 1) is used for the third component and

Δ_3 g(3, 2, 2, 7, 7) = -.056. Similarly Δ_4 g(3, 2, 3, 6, 7) = -.069 and Δ_5 g(3, 2, 3, 7, 6) = -.022. The test fails for i = 1, 3, 4, and 5. Consequently Part 1 is used. Calculation of the ratios Δ_1 g(2, 3, 3, 7, 7)/Δ_1 g(2, 2, 2, 6, 6) = -5.61, Δ_3 g(3, 2, 2, 7, 7)/Δ_3 g(2, 2, 2, 6, 6) = 1.37, Δ_4 g(3, 2, 3, 6, 7)/Δ_4 g(2, 2, 2, 6, 6) = 1.33 and finally Δ_5 g(3, 2, 3, 7, 6)/Δ_5 g(2, 2, 2, 6, 6) = 1.52 gives that the maximum ratio is for i = 5. By contradiction it will be proved that $s_5 = 1$ implies $s_3 = 1$ where X + S is the point to be tested. Since $b_0 = -\lambda C_3 - \Delta_3$ g(2, 2, 3, 6, 7) = .042 > 0, S_3 can not be 1. Repeating this argument for i = 1 and i = 4 does not yield a better solution. Consequently the down phase is implemented. Down-phase does not generate a better feasible solution. Finally the conclusion is that X = (2, 2, 2, 6, 6) is a stationary point.

APPENDIX C

THE FORTRAN LISTING OF THE SAMPLE PROGRAM GENERATOR

```
      C
      C ************************************
      C *                                  *
      C *                                  *
      C *   SAMPLE PROBLEM GENERATOR       *
      C *                                  *
      C ************************************
      C
    1       DIMENSION M(100),A(100),B(100),AX(100),BX(100),AUX(100),BUX(100),
     *X(100)
      C
      C   INPUT THE PROBLEM SIZE
      C
    2       READ(5,100) NSIZE
    3   100 FORMAT(I4)
      C
      C   INPUT THE DATA
      C
    4       READ(5,110) (M(I),I=1,100)
    5       READ(5,120) (X(I),I=1,100)
    6   110 FORMAT(20I4)
    7   120 FORMAT(20F4.0)
    8       DO 9 I=1,100
    9       A(I)=FLOAT(M(I))
   10     9 CONTINUE
      C
      C
      C   GENERATE FIVE SAMPLE PROBLEMS
      C
   11       WRITE(6,250)
   12   250 FORMAT(1H1,'SAMPLE PROBLEMS')
      C
   13       IX=12345
   14       DO 15 ISAY=1,5
   15       ITOT=100
   16       DO 10 I=1,100
   17       A(I)=FLOAT(M(I))
   18    10 B(I)=X(I)
      C
      C     GENERATE A PROBABILITY FROM A UNIFORM DISTRIBUTION
      C
   19       DO 20 J=1,NSIZE
   20       CALL RANDU(IX,IY,P)
   21       IX=IY
   22       DO 25 JJ=1,100
      C
      C   FIND THE ITEM  SELECTED
      C
   23       PJ=FLOAT(JJ)/FLOAT(ITOT)
   24       IF(P.GT.PJ) GO TO 25
   25       ISEL=JJ
   26       GO TO 30
   27    25 CONTINUE
   28    30 AX(J)=A(ISEL)
   29       BX(J)=B(ISEL)
   30       DO 40 I=1,ITOT
   31       IF(I-ISEL) 45,40,50
      C
      C   DISCARD THE SELECTED ITEM FROM THE LIST
      C
   32    45 AUX(I)=A(I)
   33       BUX(I)=B(I)
   34       GO TO 40
   35    50 II=I-1
   36       AUX(II)=A(I)
   37       BUX(II)=B(I)
   38    40 CONTINUE
      C
      C   DECREASE THE NUMBER IN THE LIST BY ONE
      C
   39       ITOT=ITOT-1
   40       DO 55 I=1,ITOT
   41       B(I)=BUX(I)
   42    55 A(I)=AUX(I)
   43    20 CONTINUE
      C
      C   WRITE OUT THE SAMPLE PROBLEMS
      C
   44       WRITE(6,200) ISAY
   45       WRITE(6,210) (AX(J),J=1,NSIZE)
   46       WRITE(6,210) (BX(J),J=1,NSIZE)
   47    15 CONTINUE
   48   200 FORMAT(I4)
   49   210 FORMAT((10F8.2))
   50       STOP
   51       END
```

APPENDIX D

COMPUTER CODE FOR THE

PRIMAL SEARCH

ALGORITHM

A List of the Variables in the Computer Program

ALFA	Stepsize
BASLA	The Subroutine that finds an initial point
BETA	Product of the parameters p (user determined) and p_k
BUTCE	Budget
C(100)	Cost of item I
CBETA	The parameter p_k
CDOTD	The dot product of the cost and direction vectors
CDOTX	The dot product of the cost vector and the current solution, also the cost of the kit
CKARE	The dot product the cost vector with itself
COST	Cost of the current solution
DENOM	The dot product of the Cost vector the vector of first differences at a solution
DIF	The first difference
DIREC	The direction vector
ENORS	E(NORS/X)
EPS	Parameter for testing the sufficiently closeness
IOX	Integer Optimal Solution
ITETA	Indicator of the search direction. If ITETA = 1, the search is in the positive direction, if ITETA = -1, it is in the negative direction.
IX(100)	Current integer solution
IY(100)	Initial solution
KODE	Indicator of the result of the search, "1" means success, "0" means failure
N	Total number of items in the problem
NEWP	New integer point
NITER	Number of iterations already performed

NNORS	Number of NORS function evaluations
NSTEP (NITER)	Number of steps within iteration NITER
M	Upper bound
PAR	Parameter p (user determined)
PDFOP	Function routine which calculates the individual poisson terms
POISON (40,100)	The matrix which contains the cumulative Poisson sums
SIRALA (X,IR)	The subroutine which ranks the array X in descending order. IR(k) gives the rank of X(k).
TABLE (POISON)	The subroutine which fills in the matrix POISON
UCOS	Cost of NEWP
UNORS	E(NORS/NEWP)
VNORS	Function Routine to calculate E(NORS/X) funtion
XLAG	Lagrange parameter (user determined)
XLAM(I)	Poisson demand rate of item I.

FORTRAN Listing of the Computer Program

```
$JOB TIME=05
C  ************************************
C  *                                  *
C  *                                  *
C  *     A PRIMAL SEARCH ALGORITHM    *
C  *                                  *
C  *                                  *
C  ************************************
C
C    MAIN PROGRAM: TO READ IN THE DATA
C                  TO WRITE OUT THE SOLUTIONS
C
1          DIMENSION IX(100),IOX(100),IY(100),DELTA(100)
2          COMMON N,M/BLOCK1/C(100)/BLOCK2/XLAM(100)/BLOCK3/BUTCE,PAR/BLOCK4
          *NITER,NSTEP(500)/BLOCK5/POISON(40,100)/COUNT/NNORS
3          EPS=.00005
C     INPUT THE NECESSARY INFORMATION
C
4          NNORS=0
5          READ(5,90) NOPR
6          READ(5,100) N,M,BUTCE
7          READ(5,110) (C(I),I=1,N)
8          READ(5,110) (XLAM(I),I=1,N)
9          READ(5,120) PAR,XLAG
C
C     CALCULATE THE PROBABILITIES
C
10         CALL TABLE(POISON)
C
C     FIND AN INITIAL SOLUTION
11         CALL BASLA(XLAG,IY)
12         DO 10 I=1,N
13        10 IX(I)=IY(I)
C
C     START THE OPTIMIZATION
C
14         CALL OPTIM(IX,IOX,ENORS,COST)
C
C     PRINT THE RESULTS
C
15         WRITE(6,400)
16         WRITE(6,405) NOPR
17         WRITE(6,209) N,BUTCE
18         WRITE(6,200)
19         DO 20 I=1,N
20        20 WRITE(6,210) I,C(I),XLAM(I),IY(I),IOX(I)
21         WRITE(6,201) ENORS,COST
22         WRITE(6,205) NNORS
23         WRITE(6,220) NITER,(NSTEP(I),I=1,NITER)
24         WRITE(6,250) PAR,XLAG
25        90 FORMAT(I4)
26       100 FORMAT(2I4,F10.0)
27       110 FORMAT(10F8.0)
28       120 FORMAT(2F10.0)
29       200 FORMAT(///,38X,'INITIAL',5X,'OPTIMAL',/,5X,'ITEM',3X,'COST/UNIT',6
            1X,'DEMAND',4X,'SOLUTION',4X,'SOLUTION',//)
30       201 FORMAT(///,5X,10HE(NORS/X)=,F10.5,5X,9HCOST(X)=$,F10.2)
31       205 FORMAT(//,5X,'# OF NORS EVALUATIONS=',I4)
32       209 FORMAT(///,5X,11H# OF ITEMS=,I4,3X,8HBUDGET=$,F10.2)
33       210 FORMAT(5X,'#',I3,2(2X,F10.2),2(8X,I4))
```

```
34      220 FORMAT(//,5X,'# OF ITERATIONS=',I4,/,5X,'# OF STEPS IN EACH ITERA
           *ION:',(5X,20I4))
35      250 FORMAT(//,5X,'LAMBDA=',F10.2,5X,'LAGRANGE MULTIPLIER=',F10.8,/)
36      400 FORMAT(1H1,//,10X,'A PRIMAL SEARCH METHOD')
37      405 FORMAT(///,5X,'PROBLEM NUMBER',I4)
38          STOP
39          END

40          SUBROUTINE OPTIM(IX,IOX,ENORS,COST)
        C *                                  *
        C *    OPTIMIZATION ROUTINE          *
        C *                                  *
41          DIMENSION IX(100),IOX(100),DIREC(100),DELTA(100),NEWP(100),
           1XNEW(100),NEIGH(100),IY(100),IRANK(100),XDEL(100)
42          COMMON N,M/BLOCK1/C(100)/BLOCK2/XLAM(100)/BLOCK3/BUTCE,PAR/BLOCK4
           *NITER,NSTEP(500)
        C
43          NITER=1
44          ENORS=VNORS(IX)
45          COST=PROD(IX)
46          CKARE=DOTPR(C)
        C
        C   EVALUATE THE FIRST DIFFERENCES
47      200 CALL TJREV(IX,DELTA)
48          NSTEP(NITER)=0
49          DENOM=DOTPR(DELTA)
50          CBETA=DENOM/CKARE
        C
        C   EVALUATE THE DIRECTION PARAMETER PK
51          BETA=PAR*CBETA
        C
        C   DETERMINE THE SEARCH DIRECTION
52          DO 10 I=1,N
53          DIREC(I)=DELTA(I)-BETA*C(I)
54          WSA=ABS(DIREC(I))
55          IF(I.EQ.1) GO TO 15
56          IF(WSA.LE.XSI) GO TO 10
57       15 XSI=WSA
58       10 CONTINUE
        C
        C   NORMALIZE THE DIRECTION
59          DO 11 I=1,N
60       11 DIREC(I)=DIREC(I)/XSI
        C   CALCULATE THE STEP SIZE ALPHA
61          CDOTD=DOTPR(DIREC)
62          CDOTX=PROD(IX)
63          XN=SQRT(FLOAT(N))
64          AMAX=(BUTCE-CDOTX)/(CDOTD*XN)
65          IF(AMAX.LT.1.0) AMAX=1.0
66          MMAX=IFIX(AMAX+.5)
67          AMAX=FLOAT(MMAX)
68          ALFA=AMAX
        C
        C   OBTAIN A NEW POINT
69      100 DO 20 I=1,N
70          WN=FLOAT(IX(I))+ALFA*DIREC(I)+.5
71          IF(WN.GE.1.) GO TO 20
72          WN=1.0
73       20 NEWP(I)=IFIX(WN)
74          NSTEP(NITER)=NSTEP(NITER)+1
```

```
      C   CALCULATE ITS COST $ E(NORS/X)
   75         UCOS=PROD(NEWP)
   76         UNORS=VNORS(NEWP)
      C   IF INFEASIBLE , PERFORM A SEARCH IN THE NEGATIVE DIRECTION
   77         IF(UCOS.GT.BUTCE) GO TO 30
      C   IF A BETTER FEASIBLE POINT, REPEAT THE STEPS
   78         IF(UNORS.LT.ENORS) GO TO 500
   79         ITETA=1
   80         GO TO 40
   81      30 ITETA=-1
   82      40 CALL SEARCH(ITETA,UCOS,UNORS,ENORS,NEWP,KODE)
   83         IF(KODE.EQ.1) GO TO 500
   84         IF(ALFA.EQ.1.) GO TO 900
   85         ALFA=ALFA-1.
   86         GO TO 100
   87     900 UCOS=COST
   88         UNORS=ENORS
   89         CALL SEARCH(1,UCOS,UNORS,ENORS,IX,KODE)
   90         IF(KODE) 600,1000,600
   91     500 DO 45 I=1,N
   92      45 IX(I)=NEWP(I)
   93     600 COST=UCOS
   94         ENORS=UNORS
   95         NITER=NITER+1
   96         GO TO 200
   97    1000 DO 50 I=1,N
   98      50 IOX(I)=IX(I)
   99         RETURN
  100         END

  101         SUBROUTINE SEARCH(ITETA,UCOS,UNORS,ENORS,NEWP,KODE)
      C
      C   THIS SUBROUTINE PERFORMS SEARCHES EITHER IN POSITIVE OR NEGATIVE DI
      C
  102         DIMENSION NEWP(100),NEIGH(100),IRANK(100),DELTA(100),INSET(100),D
             *L(100)
  103         COMMON N,M/BLOCK1/C(100)/BLOCK2/XLAM(100)/BLOCK3/BUTCE,PAR/BLOCK4
             *NITER,NSTEP(500)
      C
  104         NSTEP(NITER)=NSTEP(NITER)+1
  105         IF(ITETA.EQ.1) GO TO 9
      C
      C   CALCULATE THE FIRST DIFFERENCES FOR NEGATIVE DIRECTION
  106         CALL TURND(NEWP,DELTA)
  107         GO TO 11
      C   CALCULATE THE FIRST DIFFERENCES
  108       9 CALL TUREV(NEWP,DELTA)
  109      11 DO 10 I=1,N
  110         DELTA(I)=FLOAT(ITETA)*DELTA(I)
  111      10 DEL(I)=DELTA(I)
      C
      C   RANK THE FIRST DIFFERENCES
  112         CALL SIRALA(DEL,IRANK)
  113         DO 20 I=1,N
  114         DO 15 J=1,N
  115         JV=J
  116         IF(IRANK(J).EQ.I) GO TO 17
  117      15 CONTINUE
  118      17 INSET(I)=JV
  119      20 CONTINUE
```

```
      C
      C    CHECK WHETHER THERE IS A BETTER FEASIBLE POINT
      C
120           DO 25 I=1,N
121           JVAL=INSET(I)
122           COSJ=UCOS+FLOAT(ITETA)*C(JVAL)
123           IF(COSJ.GT.BUTCE) GO TO 25
124           DO 35 JJ=1,N
125        35 NEIGH(JJ)=NEWP(JJ)
126           IF(NEWP(JVAL).EQ.0.AND.ITETA.EQ.-1) GO TO 25
127           NEIGH(JVAL)=NEWP(JVAL)+ITETA
128           XNORS=UNORS-DELTA(JVAL)
129           IF(XNORS-ENORS) 100,120,110
130        25 CONTINUE
131           GO TO 120
132       110 DO 40 I=1,N
133        40 NEWP(I)=NEIGH(I)
134           UNORS=XNORS
135           UCOS=COSJ
      C     IF NO BETTER FEASIBLE POINT, FAILURE $ KODE=0
136       120 KODE=0
137           RETURN
      C
      C     IF A BETTER FEASIBLE POINT, SUCCESS $ KODE=1
138       100 KODE=1
139           DO 70 I=1,N
140        70 NEWP(I)=NEIGH(I)
141           UNORS=XNORS
142           UCOS=COSJ
143           RETURN
144           END

145           FUNCTION PROD(IX)
      C
      C     TO CALCULATE THE DOT PRODUCT C.IX
      C
146           DIMENSION IX(100)
      C
147           COMMON N,M/BLOCK1/C(100)
148           SUM=0.0
149           DO 10 I=1,N
150           IXI=IX(I)
151        10 SUM=SUM+FLOAT(IXI)*C(I)
152           PROD=SUM
153           RETURN
154           END

155           FUNCTION DOTPR(X)
      C
      C     TO CALCULATE THE DOT PRODUCT C.X
      C
156           DIMENSION X(100)
157           COMMON N,M/BLOCK1/C(100)
158           SUM=0.0
159           DO 10 J=1,N
160           SUM=SUM+C(J)*X(J)
161        10 CONTINUE
162           DOTPR=SUM
163           RETURN
164           END
```

```
165           FUNCTION VNORS(IX)
      C
      C    TO EVALUATE THE E(NORS/X)
      C
166           DIMENSION IX(100),PM(100,20),PRO(20)
167           COMMON N,M/BLOCK5/POISON(40,100)/COUNT/NNORS
168           DOUBLE PRECISION SUM,PRO,TOPL,PM,XW
      C
169           EPS=.00001
170           EPSQ=.00000001
171           NNORS=NNORS+1
172           M1=M+1
173           SUM=0.D+00
174           DO 20 JJ=1,M1
175           PRO(JJ)=1.D+00
176           DO 26 II=1,N
177           IXJ=IX(II)+JJ-1
178           IF(IXJ.GT.0) GO TO 24
179           XX=PDFOP(IXJ,II)
180           GO TO 25
181        24 IF(IXJ.GT.40) IXJ=40
182           XX=POISON(IXJ,II)
183        25 IF(XX.LT.EPSQ) XX=0.0
184           IF(PRO(JJ).LT.EPSQ) GO TO 20
185           PRO(JJ)=PRO(JJ)*XX
186        26 CONTINUE
187        20 SUM=SUM+1.-PRO(JJ)
188           TOPL=0.D+00
189           DO 30 I=1,N
190           K=0
191        40 MXK=IX(I)+M+2+K
192           WX=PDFOP(MXK,I)
193           WW=FLOAT(K+1)*WX
194           IF(WX.LT.EPS) GO TO 30
195           TOPL=TOPL+WW
196           K=K+1
197           GO TO 40
198        30 CONTINUE
199           VNORS=SUM+TOPL
200           RETURN
201           END

202           SUBROUTINE TUREV(IX,DELTA)
      C
      C    TO EVALUATE THE FIRST DIFFERENCES
      C
203           DIMENSION IX(100),DELTA(100)
204           COMMON N,M
      C
205           DO 10 K=1,N
206           DELTA(K)=DIF(IX,K)
207        10 CONTINUE
208           RETURN
209           END

210           SUBROUTINE TURND(NEWP,DELTA)
      C
      C    TO EVALUATE THE FIRST DIFFERENCES   IN NEGATIVE DIRECTION
      C
211           DIMENSION NEWP(100),NEIGH(100),DELTA(100)
```

```
212         COMMON N,M
      C
213         DO 10 I=1,N
214         DO 20 J=1,N
215         IF(J.EQ.I) GO TO 25
216         NEIGH(J)=NEWP(J)
217         GO TO 20
218      25 NEIGH(J)=NEWP(J)-1
219         IF(NEIGH(J).LT.0) NEIGH(J)=0
220      20 CONTINUE
221         DELTA(I)=DIF(NEIGH,I)
222      10 CONTINUE
223         RETURN
224         END

225         FUNCTION DIF(IX,K)
      C
      C   TO COMPUTE THE FIRST DIFFERENCE OF ITEM K
      C
226         DIMENSION IX(100)
227         COMMON N,M/BLOCK5/POISON(40,100)
228         DOUBLE PRECISION PRO,SUM,EPSQ
      C
229         EPSQ=1.D-08
230         M1=M+1
231         SUM=0.D+00
232         DO 20 J1=1,M1
233         PRO=1.D+00
234         J=J1-1
235         DO 30 I=1,N
236         IXJ=IX(I)+J
237         IF(I.EQ.K) GO TO 25
238         IF(IXJ.EQ.0) GO TO 23
239         IF(IXJ.GT.39) IXJ=39
240         PRO=PRO*POISON(IXJ,I)
241         GO TO 30
242      23 IXJ1=IXJ
243         GO TO 26
244      25 IXJ1=IXJ+1
245      26 IF(PRO.LT.EPSQ) GO TO 20
246         PRO=PRO*POISON(IXJ1,I)
247      30 CONTINUE
248      20 SUM=SUM+PRO
249         IXM=IX(K)+M+1
250         IF(IXM.GT.40) IXM=40
251         DIF=SUM+1.-POISON(IXM,K)
252         RETURN
253         END

254         SUBROUTINE TABLE(POISON)
      C
      C   TO FILL IN THE ARRAY FOR CUMULATIVE POISSON SUMS
      C
255         DIMENSION POISON(40,100)
256         COMMON N,M/BLOCK2/XLAM(100)
257         DOUBLE PRECISION EPS,TERM,SUMX
      C
258         EPS=1.D-06
259         DO 10 J=1,N
260         TERM=1.D+00
```

```
      SUMX=1.D+00
      DO 20 I=1,40
      TERM=TERM*XLAM(J)/FLOAT(I)
      IX=I
      IF(TERM.LT.EPS) GO TO 25
      SUMX=SUMX+TERM
      POISON(I,J)=EXP(-XLAM(J))*SUMX
      IF(POISON(I,J).GT.1.0) POISON(I,J)=1.0
   20 CONTINUE
   25 IF(IX.EQ.40) GO TO 10
      DO 30 I=IX,40
   30 POISON(I,J)=1.0
   10 CONTINUE
      RETURN
      END

      FUNCTION PDFOP(K,I)
C
C   TO CALCULATE INDIVIDUAL TERMS OF THE POISSON DISTRIBUTION
C
      COMMON /BLOCK2/XLAM(100)
      DOUBLE PRECISION SUM,WW
C
      SUM=0.D+00
      IF(K.LT.2) GO TO 10
      DO 15 J=2,K
      SUM=SUM+ALOG(FLOAT(J))
   15 CONTINUE
   10 WW=-XLAM(I)+K*ALOG(XLAM(I))-SUM
      IF(WW.LT.-15.0) GO TO 20
      PDFOP=DEXP(WW)
      GO TO 25
   20 PDFOP=0.0
   25 RETURN
      END

      SUBROUTINE SIRALA(X, IR)
C
C   THIS SUBROUTINE RANKS AN ARRAY OF ELEMENTS IN DESCENDING ORDER
C   IR(K) IS THE RANK OF ITEM K IN THIS ORDERING
C
      COMMON N,M
      DIMENSION X(100),IR(100),ISEQ(100),Y(100)
C
      DO 9 I=1,N
    9 ISEQ(I)=I
      DO 10 I=1,N
      XMIN=X(I)
      IMINJ=I
      IMIN=ISEQ(I)
      N1=N-I+1
      DO 20 J=1,N1
      IF(X(J).GT.XMIN) GO TO 21
      GO TO 20
   21 XMIN=X(J)
      IMIN=ISEQ(J)
      IMINJ=J
   20 CONTINUE
      IR(IMIN)=I
      IF(N1.EQ.1) GO TO 10
```

```
310           N2=N1-1
311           DO 30 J=1,N2
312           IF(J.LT.IMINJ) GO TO 33
313           J1=J+1
314           ISEQ(J)=ISEQ(J1)
315           Y(J)=X(J1)
316           GO TO 29
317        33 Y(J)=X(J)
318        29 X(J)=Y(J)
319        30 CONTINUE
320        10 CONTINUE
321           RETURN
322           END

323           SUBROUTINE BASLA(XLAG,IX)
      C
      C    THIS SUBROUTIN FINDS AN INITIAL POINT WITH A PREDETERMINED
      C    LAGRANGE PARAMETER
      C
324           DIMENSION IX(100)
325           COMMON N,M/BLOCK1/C(100)/BLOCK5/POISON(40,100)
      C
326           DO 10 I=1,N
327           IW=0
328           A=XLAG*C(I)+PDFOP(IW,I)-1.
329           IF(A.GE.0.0) GO TO 10
330        15 IW=IW+1
331           B=A+PDFOP(IW,I)
332           IF(A.LE.0.0.AND.B.GT.0.0) GO TO 10
333           A=B
334           GO TO 15
335        10 IX(I)=IW
336           RETURN
337           END

      $ENTRY
```

A PRIMAL SEARCH METHOD

PROBLEM NUMBER 1

OF ITEMS= 5 BUDGET=$ 25000.00

ITEM	COST/UNIT	DEMAND	INITIAL SOLUTION	OPTIMAL SOLUTION
# 1	2980.00	2.10	2	3
# 2	1751.00	1.50	2	2
# 3	462.00	1.20	3	3
# 4	1500.00	5.00	6	6
# 5	345.00	3.50	6	6

E(NORS/X)= 0.98571 COST(X)=$ 24898.00

OF NORS EVALUATIONS= 3

OF ITERATIONS= 2
OF STEPS IN EACH ITERATION: 2 3

LAMBDA= 0.85 LAGRANGE MULTIPLIER=0.00020000

STATEMENTS EXECUTED= 13876

CORE USAGE OBJECT CODE= 15256 BYTES, ARRAY AREA= 43028 BYTES, TOTAL AREA AVAILABLE= 145408 BYTES

DIAGNOSTICS NUMBER OF ERRORS= 0, NUMBER OF WARNINGS= 0, NUMBER OF EXTENSIONS= 0

COMPILE TIME= 0.27 SEC, EXECUTION TIME= 0.24 SEC, 12.35.52 MONDAY 4 AUG 80 WATFIV - JUN 1977

APPENDIX E

COMPUTER CODE FOR THE MODIFIED ALGORITHM OF BROOKS ET AL.

A List of the Variables in the Computer Program

BO	Coefficients $b_0, b_1, \ldots, b_m$
C(I)	Cost of item I
COST	Cost of the current solution X
ENORS	E(NORS/X)
EPS	Parameter for testing sufficiently closeness
F(100,50)	Matrix for the cumulative poisson sums
ICOUN	Counter for the number of iterations
IMAX	The index I for which RATIO(I) is maximum
IX(100)	Current solution
M	An upper bound
N	The total number of items under consideration
NOPR	Problem number
NSTEP (ICOUN)	Number of steps in iteration ICOUN
OPTIM	The optimization routine
PDFOP	Function routine which calculates the individual poisson terms
RATIO (100)	Ratio of return per dollar investment
RMAX	Ratio (IMAX)
TABLE	The subroutine to fill in the matrix F(100,50)
XLAM (100)	Poisson demand rates
XINVT	Budget
VNORS	Function routine to calculate E(NORS/X)

The FORTRAN Listing of the Computer Program

```
      C
      C ***********************************************
      C
      C   MODIFIED SOLUTION METHOD OF BROOKS ET AL
      C
      C ***********************************************
      C
 1          DIMENSION IX(100),BO(100)
 2          COMMON N,M/BLOCK1/C(100),XINVT/BLOCK2/XLAM(100)/BLOCK4/F(100,50)/
           *LOCK5/ICOUN,NSTEP(50)
      C
 3          EPS=.001
 4          EPSQ=EPS**2
      C
      C   INPUT THE DATA
      C
 5          READ(5,90) NOPR
 6          READ(5,100) N,M,XINVT
 7          READ(5,110) (C(I),I=1,N)
 8          READ(5,110) (XLAM(I),I=1,N)
 9          WRITE(6,400)
10          WRITE(6,405) NOPR
11          WRITE(6,105) N,XINVT
12          M1=M+1
      C
      C   INITIALIZE THE PARAMETERS B
      C
13          DO 10 J=1,M
14       10 BO(J)=0.0
15          BO(M1)=1.0
      C
      C   CALCULATE THE PROBABILITIES
16          CALL TABLE(F)
17          ICOUN=1
      C
      C   PERFORM THE OPTIMIZATION
      C
18    1000 CALL OPTIM(BO,IX,COST)
      C
      C   CHECK IF B-NEW IS SUFFICIENTLY CLOSE TO B-OLD
      C
19          SUMSQ=0.0
20          DO 20 J1=1,M1
21          PRO=1.0
22          DO 40 I5=1,N
23          IXJ=IX(I5)+J1
24          IF(IXJ.GT.50) GO TO 40
25          PRO=PRO*F(I5,IXJ)
26       40 CONTINUE
27          IF(PRO.LT.EPSQ) PRO=0.0
28          SUMSQ=SUMSQ+((BO(J1)-PRO)**2)
29          BO(J1)=PRO
30       20 CONTINUE
      C
      C   IF SO, TERMINATE AND PRINT THE RESULTS
      C
31          IF(SUMSQ.LT.EPS) GO TO 500
      C   OTHERWISE REPEAT THE STEPS
32          ICOUN=ICOUN+1
```

```
33          GO TO 1000
34        500 WRITE(6,240)
35          DO 30 I=1,N
36         30 WRITE(6,250) I,C(I),XLAM(I),IX(I)
37          ENORS=VNORS(IX)
38          WRITE(6,201) ENORS,COST
39          WRITE(6,300) (BO(J),J=1,M1)
40          WRITE(6,301) ICOUN,(NSTEP(I),I=1,ICOUN)
41         90 FORMAT(I4)
42        100 FORMAT(2I4,F10.0)
43        105 FORMAT(//,5X,' N=',I4,5X,' BUDGET=$',F10.2)
44        110 FORMAT(10F8.0)
45        201 FORMAT(//,5X,10HE(NORS/X)=,F10.5,2X,9HCOST(X)=$,F10.2)
46        240 FORMAT(///,38X,'OPTIMAL',/,5X,'ITEM',3X,'COST/UNIT',5X,'DEMAND',4X
           1,'SOLUTION',//)
47        250 FORMAT(5X,'#',I3,2(2X,F10.2),8X,I4)
48        300 FORMAT(//,5X,'B:',10F10.3)
49        301 FORMAT(///,5X,16H# OF ITERATIONS=,I4,//,(5X,'# OF STEPS IN EACH IT
           *ERATION:',2014))
50        400 FORMAT(1H1,//,5X,'THE MODIFIED SOLUTION METHOD OF BROOKS ET AL',//
           *)
51        405 FORMAT(///,5X,'PROBLEM NUMBER',I4)
52          STOP
53          END

54          SUBROUTINE OPTIM(BO,IX,COST)
      C
      C   OPTIMIZATION ROUTINE PERFORMS THE MARGINAL ANALYSIS
      C
55          DIMENSION IX(100),BO(100),RATIO(100),MR(100)
56          COMMON N,M/BLOCK1/C(100),XINVT/BLOCK2/F(100,50)/BLOCK5/ICOUN,NSTEP
           *(50)
      C
57          M1=M+1
58          COST=0.0
59          DO 10 I=1,N
60          IX(I)=0
      C
      C   CALCULATE THE RATIO OF RETURN PER EACH DOLLAR INVESTED FOR EACH ITE
61          SUM=0.0
62          DO 20 J=1,M1
63          JJ=J+1
64          WW=F(I,JJ)/F(I,J)
65         20 SUM=SUM+BO(J)*ALOG(WW)
66         10 RATIO(I)=SUM/C(I)
67          NSTEP(ICOUN)=1
68        200 NOR=0
69          NSTEP(ICOUN)=NSTEP(ICOUN)+1
70          DO 21 I=1,N
71         21 MR(I)=-1
72        100 IMR=0
      C
      C   FIND THE MAXIMUM RATIO & ADD ONE UNIT OF THIS ITEM TO THE KIT
      C
73          DO 20 I=1,N
74          IF(MR(I).EQ.1) GO TO 25
75          IMR=IMR+1
76          IF(IMR.EQ.1) GO TO 26
77          IF(RATIO(I).LE.RMAX) GO TO 25
78         26 RMAX=RATIO(I)
```

```
79          IMAX=I
80       25 CONTINUE
81          MR(IMAX)=1
      C
      C    CALCULATE THE COST OF THE NEW KIT
82          XCOST=COST+C(IMAX)
      C
      C    IF COST EXCEEDS THE BUDGET, FIND THE ITEM WHICH WILL YIELD MAXIMUM
      C    WITHOUT EXCEEDING THE BUDGET, THEN RETURN THE MAIN PROG.
      C
83          IF(XCOST.GT.XINVT) GO TO 30
84          COST=XCOST
85          IX(IMAX)=IX(IMAX)+1
86          SUM=0.0
87          DO 35 J=1,M1
88          IXJ=IX(IMAX)+J
89          IXJ1=IXJ+1
90          IF(IXJ.GE.50) GO TO 35
91          WW=F(IMAX,IXJ1)/F(IMAX,IXJ)
92          SUM=SUM+BC(J)*ALOG(WW)
93       35 CONTINUE
      C    OTHERWISE UPDATE THE RATIO(IMAX), REPEAT THE STEPS
94          RATIO(IMAX)=SUM/C(IMAX)
95          GO TO 200
96       30 NOR=NOR+1
97          IF(NOR.LT.N) GO TO 100
98          RETURN
99          END

100         SUBROUTINE TABLE(F)
      C
      C    CALCULATING CUMULATIVE POISSON SUMS
      C
101         DIMENSION F(100,50)
102         COMMON N,M/BLOCK1/C(100),XINVT/BLOCK2/XLAM(100)
103         DOUBLE PRECISION EPS,TERM,SUM
      C
104         EPS=1.D-06
105         DO 10 I=1,N
106         F(I,1)=PDFOP(0,I)
107         TERM=1.D+00
108         SUM=1.D+00
109         DO 30 J=2,50
110         TERM=TERM*XLAM(I)/FLOAT(J-1)
111         JX=J
112         IF(TERM.LT.EPS) GO TO 25
113         SUM=SUM+TERM
114         F(I,J)=EXP(-XLAM(I))*SUM
115         IF(F(I,J).GT.1.0) F(I,J)=1.0
116      30 CONTINUE
117      25 IF(JX.EQ.50) GO TO 10
118         DO 35 J=JX,50
119      35 F(I,J)=1.0
120      10 CONTINUE
121         RETURN
122         END

123         FUNCTION PDFOP(K,I)
      C
      C    CALCULATING THE INDIVIDUAL POISSON TERMS
```

```
      C
124           COMMON /BLOCK2/XLAM(100)
125           DOUBLE PRECISION SUM,WW
      C
126           SUM=0.D+00
127           IF(K.LT.2) GO TO 10
128           DO 15 J=2,K
129           SUM=SUM+ALOG(FLOAT(J))
130        15 CONTINUE
131        10 WW=-XLAM(I)+K*ALOG(XLAM(I))-SUM
132           IF(WW.LT.-15.0) GO TO 20
133           PDFOP=DEXP(WW)
134           GO TO 25
135        20 PDFOP=0.0
136        25 RETURN
137           END

138           FUNCTION VNORS(IX)
      C
      C    CALCULATING THE E(NORS/X) FUNCTION
      C
139           DIMENSION IX(100),PRO(20)
140           COMMON N,M/BLOCK4/F(100,50)
141           DOUBLE PRECISION SUM,PRO,TOPL
      C
142           EPS=.00001
143           M1=M+1
144           SUM=0.D+00
145           DO 10 J=1,M1
146           PRO(J)=1.D+00
147           DO 20 I=1,N
148           IXJ=IX(I)+J
149        20 PRO(J)=PRO(J)*F(I,IXJ)
150        10 SUM=SUM+1.-PRO(J)
151           TOPL=0.D+00
152           DO 30 I=1,N
153           K=0
154        40 MXK=IX(I)+M+2+K
155           WX=PDFOP(MXK,I)
156           WW=FLOAT(K+1)*WX
157           IF(WX.LT.EPS) GO TO 30
158           TOPL=TOPL+WW
159           K=K+1
160           GO TO 40
161        30 CONTINUE
162           VNORS=SUM+TOPL
163           RETURN
164           END

      $ENTRY
```

```
THE MODIFIED SOLUTION METHOD OF BROOKS ET AL


PROBLEM NUMBER    1

N=    5      BUDGET=$   25000.00


ITEM    COST/UNIT        DEMAND     OPTIMAL SOLUTION

#   1      2980.00          2.10              2
#   2      1751.00          1.50              2
#   3       462.00          1.20              4
#   4      1500.00          5.00              7
#   5       345.00          3.50              9

E(NCRS/X)=    0.98619   COST(X)= $  24915.00

B:       0.450      0.728      0.891      0.962      0.988      0.996


    # OF ITERATIONS=    3
    # OF STEPS IN EACH ITERATION:  23  26  26
STATEMENTS EXECUTED=     8963
CORE USAGE        OBJECT CODE=     7320 BYTES, ARRAY AREA=    22800 BYTES, TOTAL AREA AVAILABLE=   145408  BYTES
DIAGNOSTICS         NUMBER OF ERRORS=        0, NUMBER OF WARNINGS=        0, NUMBER OF EXTENSIONS=        0
COMPILE TIME=      0.13 SEC, EXECUTION TIME=      0.14 SEC,      12.11.57    MONDAY        4 AUG 80    WATFIV - JUN 1977
```

APPENDIX F

COMPUTER CODE FOR THE SADDLE POINT SEARCH ALGORITHM

A List of the Variables in the Computer Program

A(50)	The cost of the current solution for subroutine DUALPR
AXM	Minimum of A(I) which exceed BUTCE; A(IXM)
AXP	Maximum of A(I) which are less than BUTCE; A(IXP)
BASLA	Subroutine which finds an initial point
BUTCE	Budget
C(100)	Cost of the item
COST	Total cost of the current solution
CX(50)	E(NORS/X) (used only in DUALPR)
CXM	CX(IXM)
CXP	CX(IXP)
DIF	Function routine for calculating first differences
DOTPR	The dot product of two vectors
DUALPR	Subroutine which solves the dual problem to obtain the Lagrange multiplier
ENORS	E(NORS/X)
INPUT	The subroutine which reads in the necessary data and initializes the arrays
ISTAR	Starting solution X^o
IX(100)	Current solution
IX0(100)	Solution obtained in the Subroutine BASLA
IX1(100)	Solution obtained in the Subroutine SEARCH
IX2(100)	Solution obtained in the Subroutine TARAMA
1X0(100)	Final and/or optimal solution
IY(100)	Initial solution
KPAR	Parameter for finding an initial point in BASLA
M	An upper bound

N	Total number of items under consideration
NARA	Number of times Subroutine ARAMA called
NBAS	Number of time Subroutine BASLA called
NITER	Number of iterations
NOPG	Number of points generated by the Saddle Point Search Algorithm
NOPR	Problem number
NPA1	Number of times Part 1 called
NPA2	Number of times Part 2 called
NPA3	Number of times Part 3 called
NSEA	Number of times SEARCH called
NVNOR	Number of E(NORS/X) function evaluations
OPTIM	Optimization Routine
PART1	The subroutine which performs a test by calculating an upper bound to the linear program in determining whether a component can be increased by one
PART2	The subroutine which calculates an upper bound for the sum of first differences
PART3	The subroutine which performs the same test as in PART1 but sets up the linear program explicitly
PDFOP	Function routine for calculating individual poisson terms
PLAG	Current Lagrange multiplier
POISON	Matrix for the cumulative poisson terms
POLD	Previous Lagrange multiplier
SEARCH	The subroutine that makes a one dimensional search for each component separately
SIRALA (X,IR)	The subroutine that ranks the array X in descending order
TABLE	The subroutine which fills in the matrix POISON

TARAMA	The subroutine which makes an attempt for improvement heuristically
VNORS	The function routine to calculate E(NORS/X) + λ CX
VXO	E(NORS/IXO)
VX1	E(NORS/IX1)
VX2	E(NORS/IX2)
XLAM(100)	Poisson demand rates
XLPRO	The Linear Programming Routine which solves problems of the type max CX subject to $AX \leq b \quad X \geq 0$.

FORTRAN Listing of the Computer Program

```
      $JOB TIME=05
      C ************************************
      C *                                  *
      C *                                  *
      C *  A  SADDLE POINT SEARCH ALGORITHM *
      C *                                  *
      C *                                  *
      C ************************************
      C
      C   MAIN PROGRAM
      C
    1       DIMENSION IX(100),IY(100)
      C
      C   INPUT THE DATA
    2       CALL INPUT(NOPR,KPAR,PLAG,IY)
      C
      C   PERFORM THE OPTIMIZATION
      C
    3       CALL OPTIM(PLAG,KPAR,IY,ENORS,IX,COST)
      C
      C   PRINT THE RESULTS
      C
    4       CALL OTPUT(PLAG,NOPR,IY,ENORS,IX,COST)
      C
    5       STOP
    6       END

    7       SUBROUTINE INPUT(NOPR,KPAR,PLAG,IY)
      C
      C   TO READ IN THE DATA
      C
    8       DIMENSION IY(100)
    9       COMMON N,M/BLOCK1/C(100)/BLOCK2/XLAM(100)/BLOCK3/BUTCE
   10       COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50),
           *NPA2(50),NPA3(50),NVNOR/TAB/POISON(50,100)
      C
   11       READ(5,90) NOPR
   12       READ(5,100) N,M,BUTCE
   13       READ(5,110) (C(I),I=1,N)
   14       READ(5,110) (XLAM(I),I=1,N)
   15       READ(5,150) KPAR,ISTAR,PLAG
      C
      C   INITIALIZE THE COUNTERS
   16       NITER=1
   17       NVNOR=0
   18       DO 19 I=1,50
   19       NBAS(I)=0
   20       NSEA(I)=0
   21       NTAR(I)=0
   22       NARA(I)=0
   23       NPA1(I)=0
   24       NPA2(I)=0
   25       NPA3(I)=0
   26    19 CONTINUE
   27       DO 10 I=1,N
   28    10 IY(I)=ISTAR
      C
      C   CALCULATE THE CUMULATIVE POISSON SUMS
      C
   29       CALL TABLE(POISON)
```

```
30        90 FORMAT(I4)
31       100 FORMAT(2I4,F10.0)
32       110 FORMAT(10F8.0)
33       150 FORMAT(2I4,F10.0)
34           RETURN
35           END

36           SUBROUTINE OTPUT(PLAG,NOPR,IY,ENORS,IX,COST)
     C
     C    WRITING OUT THE RESULTS
     C
37           DIMENSION IY(100),IX(100)
38           COMMON N,M/BLOCK1/C(100)/BLOCK2/XLAM(100)/BLOCK3/BUTCE
39           COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50),
            *NPA2(50),NPA3(50),NVNOR/TAB/POISON(5),100
     C
40           WRITE(6,300)
41           WRITE(6,405) NOPR
42           WRITE(6,205) N,BUTCE
43           WRITE(6,220)
44           DO 20 I=1,N
45        20 WRITE(6,225) I,C(I),XLAM(I),IY(I),IX(I)
46           WRITE(6,210) ENORS,COST
47           WRITE(6,230) PLAG
48           WRITE(6,240) NITER,NVNOR
49           WRITE(6,245)
50           DO 30 I=1,NITER
51        30 WRITE(6,250) I,NBAS(I),NSEA(I),NTAR(I),NARA(I),NPA1(I),NPA2(I),NP
            13(I)
52       205 FORMAT(5X,'NUMBER OF ITEMS=',I4,2X,'BUDGET=$',F10.2)
53       220 FORMAT(///,38X,'INITIAL',5X,'OPTIMAL',/,5X,'ITEM',3X,'COST/UNIT',6
            1X,'DEMAND',4X,'SOLUTION',4X,'SOLUTION',//)
54       225 FORMAT(5X,'#',I3,2(2X,F10.2),2(8X,I4))
55       210 FORMAT(//,5X,'E(NORS/X)=',F10.5,5X,'COST(X)=$',F10.2)
56       230 FORMAT(/,5X,'LAGRANGE MULTIPLIER=',F10.6)
57       240 FORMAT(//,5X,'# OF ITERATIONS=',I4,//,5X,'# OF NORS EVALUATIONS=',
            1I4)
58       245 FORMAT(//,5X,'# OF  STEPS IN EACH ITERATION',//,5X,'ITERATION #',4
            *X,'STEP1  STEP2  STEP3  STEP4  PART1  PART2  PART3',/)
59       250 FORMAT(12X,I4,2X,7(3X,I4))
60       300 FORMAT(1H1,//,5X,'THE SADDLE POINT SEARCH ALGORITHM',//)
61       405 FORMAT(///,5X,'PROBLEM NUMBER',I4,/)
62           RETURN
63           END

64           SUBROUTINE OPTIM(PLAG,KPAR,IYO,VXA,IXO,COST)
     C
     C    OPTIMIZATION ROUTINE
     C
65           DIMENSION IXO(100),IX1(100),IXO(100),IYO(100),CX(50),A(50),IX2(100
            *)
66           COMMON N,M/BLOCK3/BUTCE/BLOCK4/NOPG/BLOCK1/C(100)
67           COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50),
            *NPA2(50),NPA3(50),NVNOR
     C
68           EPS=.000002
69           NOPG=1
70           ZZ=0.0
71           CX(NOPG)=VNORS(ZZ,IYO)
72           A(NOPG)=DOTPR(IYO)
```

```
      C
      C   STEP1: FIND AN INITIAL POINT
      C
   73   500 CALL BASLA(PLAG,KPAR,IX0)
   74       IF(NITER.GT.1) GO TO 20
   75       DO 25 I=1,N
   76    25 IY0(I)=IX0(I)
      C
      C   STEP2: FIND A STATIONARY POINT
      C
   77    20 CALL SEARCH(PLAG,IX0,IX1)
   78       VX1=VNORS(PLAG,IX1)
      C
      C   STEP3: A HEURISTIC IMPROVEMENT TEST
      C
   79       CALL TARAMA(PLAG,IX1,VX1,IX2,VX2)
      C
      C   COMPARE THE NEW POINT WITH THE INCUMBENT POINT
   80       IF(VX2-VX1) 14,12,12
   81    14 DO 13 I=1,N
   82    13 IX0(I)=IX2(I)
   83       GO TO 20
      C
      C   STEP4: REDUCTION OF POTENTIAL CANDIDATES BY CONTRADICTION
      C
   84    12 CALL ARAMA(PLAG,IX2,VX2,IX0,VX0)
   85       IF(VX0-VX1) 10,15,15
   86    10 DO 11 I=1,N
   87    11 IX0(I)=IX0(I)
   88       GO TO 20
   89    15 NOPG=NOPG+1
      C
      C   READJUST THE COEFFICEINT OF THE LINEAR PROGRAM
      C
   90       A(NOPG)=DOTPR(IX0)
   91       COST=A(NOPG)
   92       VXA=VNORS(ZZ,IX0)
   93       CX(NOPG)=VXA
   94       POLD=PLAG
      C
      C   SOLVE THE DUAL LP TO OBTAIN THE LAGRANGE MULTIPLIER
      C
   95       CALL DUALPR(CX,A,PLAG)
      C
      C   IF THE NEW MULTIPLIER IS SUFFICIENTLT CLOSE TO OLD ONE, TERMINATE
      C   OTHERWISE REPEAT THE ITERATIONS
      C
   96       FARK=ABS(POLD-PLAG)
   97       IF(FARK.LT.EPS) GO TO 1000
   98       NITER=NITER+1
   99       GO TO 500
  100 1000 RETURN
  101       END

  102       SUBROUTINE DUALPR(CX,A,X)
      C
      C   SOLVE THE DUAL TO OBTAIN A LAGRANGE MULTIPLIER
      C
  103       DIMENSION CX(50),A(50)
  104       COMMON /BLOCK3/BUTCE/BLOCK4/NOPG
```

```
105           IX=0
106           IY=0
107           DO 10 I=1,NOPG
108           IF(A(I)-BUTCE) 20,20,30
109        20 IY=IY+1
110           IF(IY.EQ.1) GO TO 25
111           IF(A(I)-AXP) 10,10,25
112        25 AXP=A(I)
113           CXP=CX(I)
114           GO TO 10
115        30 IX=IX+1
116           IF(IX.EQ.1) GO TO 35
117           IF(A(I)-AXM) 35,10,10
118        35 AXM=A(I)
119           CXM=CX(I)
120        10 CONTINUE
121           X=(CXP-CXM)/(AXM-AXP)
122           RETURN
123           END

124           SUBROUTINE BASLA(XLAG,KPAR,IX)
      C
      C    FINDS AN INITIAL POINT
      C
125           DIMENSION IX(100)
126           COMMON N,M/BLOCK1/C(100)
127           COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50)
             *NPA2(50),NPA3(50),NVNOR
128           NBAS(NITER)=NBAS(NITER)+1
129           DO 10 I=1,N
130           IW=0
131           A=XLAG*C(I)+PDFOP(IW,I)-1.
132           IF(A.GE.0.0) GO TO 11
133        15 IW=IW+1
134           B=A+PDFOP(IW,I)
135           IF(A.LE.0.0.AND.B.GT.0.0) GO TO 11
136           A=B
137           GO TO 15
138        11 IW=IW-KPAR
139           IF(IW.LT.0) IW=0
140        10 IX(I)=IW
141           RETURN
142           END

143           SUBROUTINE SEARCH(XLAG,IX,IXO)
      C
      C    PERFORMS ONE DIMENSIONAL SEARCH FOR EACH COMPONENT SEPARATELY
      C
144           DIMENSION IX(100),IXO(100),IY(100)
145           COMMON N,M
146           COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50)
             *NPA2(50),NPA3(50),NVNOR
      C
147           NSEA(NITER)=NSEA(NITER)+1
148           DO 10 I=1,N
      C
      C    CALCULATE THE FIRST DIFFERENCE
149           X1=DIF(XLAG,IX,I)
      C
```

```
      C    IF IT IS POSITIVE , THE POINT X(I) IS STATIONARY, OTHERWISE
      C    FIND X(I) SUCH THAT ITS FIRST DIFFERENCE IS POSITIVE
      C
  150         IF(X1) 20,25,25
  151      20 DO 11 J=1,N
  152         IF(I.EQ.J) GO TO 12
  153         IY(J)=IX(J)
  154         GO TO 11
  155      12 IY(J)=IX(J)+1
  156      11 CONTINUE
  157         Y1=DIF(XLAG,IY,I)
  158         IF(Y1) 30,36,36
  159      30 DO 13 J=1,N
  160         IF(I-J) 13,14,13
  161      14 IX(J)=IX(J)+1
  162      13 CONTINUE
  163         X1=Y1
  164         GO TO 20
  165      25 DO 15 J=1,N
  166         IF(I.EQ.J) GO TO 16
  167         IY(J)=IX(J)
  168         GO TO 15
  169      16 IY(J)=IX(J)-1
  170         IF(IY(J).LT.0) GO TO 36
  171      15 CONTINUE
  172         Y2=DIF(XLAG,IY,I)
  173         IF(Y2) 36,36,40
  174      40 DO 60 J=1,N
  175      60 IX(J)=IY(J)
  176         X1=Y2
  177         GO TO 25
  178      36 DO 65 J=1,N
  179      65 IXO(J)=IX(J)
  180      10 CONTINUE
  181         RETURN
  182         END

  183         SUBROUTINE TARAMA(XLAG,IX,VX,IXO,VXO)
      C
      C    OBTAINS IMPROVEMENT  HEURISTICALLY
      C
  184         DIMENSION IX(100),IY(100),RATIO(100),IRANK(100),IXO(100)
  185         COMMON N,M
  186         COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50),
             *NPA2(50),NPA3(50),NVNOR
      C
  187         NTAR(NITER)=NTAR(NITER)+1
  188         ITETA=1
  189     100 DO 10 I=1,N
  190         DO 20 J=1,N
  191         IF(J.EQ.I) GO TO 25
  192         IF(ITETA.EQ.-1.AND.IX(J).EQ.0) GO TO 25
  193         IY(J)=IX(J)+ITETA
  194         GO TO 20
  195      25 IY(J)=IX(J)
  196      20 CONTINUE
  197         ALFB=DIF(XLAG,IY,I)
  198         ALFA=DIF(XLAG,IX,I)
  199         RATIO(I)=ALFB/ALFA
  200      10 CONTINUE
```

```
      C
      C   RANK THE RATIO OF FIRST DIFFERENCES OF EACH COMPONENT
      C
201           CALL SIRALA(RATIO, IRANK)
202           DO 30 I=1,N
203           DO 35 J=1,N
204           IF(IRANK(J).GT.I) GO TO 38
205           IF(ITETA.EQ.-1.AND.IX(J).EQ.0) GO TO 38
206           IY(J)=IX(J)+ITETA
207           GO TO 35
208        38 IY(J)=IX(J)
209        35 CONTINUE
      C
      C   CALCULATE ITS E(NORS/X)
210           VY=VNORS(XLAG, IY)
      C
      C   COMPARE WITH THE INCUMBENT SOLUTION
211           IF(VY.LT.VX) GO TO 50
212        30 CONTINUE
213           IF(ITETA.EQ.-1) GO TO 90
214           ITETA=-1
215           GO TO 100
216        50 DO 60 I=1,N
217        60 IXO(I)=IY(I)
218           VXO=VY
219           GO TO 110
220        90 DO 70 I=1,N
221        70 IXO(I)=IX(I)
222           VXO=VX
223       110 RETURN
224           END

225           SUBROUTINE ARAMA(XLAG,IX,VX,IXO,VXO)
      C
      C   REDUCES THE SET OF POTENTIAL CANDIDATES BY CONTRADICTION
      C
226           DIMENSION IX(100),IXO(100),IY(100),IZ(100),ISIR(100),JARR(100),MO
             *C(100)
227           COMMON N,M/BLOCK5/ITETA/BLOCK8/NOZE,JARR/BLOCK6/WLAM
228           COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50),
             *NPA2(50),NPA3(50),NVNOR
229           NARA(NITER)=NARA(NITER)+1
      C
230           WLAM=XLAG
231           ITETA=1
232       500 ICOUN=0
233           NOZE=0
234           IKODE=0
235       400 DO 10 I=1,N
236           IF(NOZE.EQ.0) GO TO 19
237           DO 15 II=1,NOZE
238           IF(JARR(II).EQ.I) GO TO 10
239        15 CONTINUE
240        19 DO 20 J=1,N
241           IF(J.EQ.I) GO TO 25
242           IF(ITETA.EQ.-1.AND.IX(J).EQ.0) GO TO 25
243           IF(NOZE.EQ.0) GO TO 29
244           DO 28 JJ=1,NOZE
245           IF(JARR(JJ).EQ.J) GO TO 25
246        28 CONTINUE
```

```
247        29 IY(J)=IX(J)+ITETA
248           GO TO 20
249        25 IY(J)=IX(J)
250        20 CONTINUE
251           ALFA2=DIF(XLAG,IY,I)
252           IF(ALFA2) 11,12,12
253        11 ICOUN=ICOUN+1
254           ALFA1=DIF(XLAG,IX,I)
255           RATIO=ALFA2/ALFA1
256           ISIR(ICOUN)=I
257           IF(ICOUN.EQ.1) GO TO 41
258           IF(RATIO.LT.RMAX) GO TO 10
259        41 RMAX=RATIO
260           JJ=I
261           GO TO 10
262        12 NOZE=NOZE+1
263           JARR(NOZE)=I
264        10 CONTINUE
265           IF(IKODE.EQ.1) GO TO 110
266           IF(ICOUN.EQ.0) GO TO 110
267           DO 30 I=1,N
268           IF(I.EQ.JJ) GO TO 33
269           GO TO 35
270        33 IF(ITETA.EQ.-1.AND.IX(I).EQ.0) GO TO 35
271           IY(I)=IX(I)+ITETA
272           GO TO 30
273        35 IY(I)=IX(I)
274        30 CONTINUE
275           IF(ICOUN.EQ.1) GO TO 60
276           NUC=0
277           DO 50 J=1,ICOUN
278           IF(ISIR(J).EQ.JJ) GO TO 50
279           JL=ISIR(J)
      C
      C    PROVE BY CONTRADICTION THAT JL-COMPONENT CAN'T BE 1
280           CALL PART1(IY,JJ,JL,IZ,ICODE)
281           IF(ICODE.EQ.0) GO TO 55
282           DO 59 J1=1,N
283        59 IY(J1)=IZ(J1)
284           GO TO 50
285        55 NUC=NUC+1
286           MOUC(NUC)=J
287        50 CONTINUE
288           IF(NUC.EQ.0) GO TO 60
289           IF(NUC.GT.1) GO TO 40
290           JUC=MOUC(NUC)
291           IF(ITETA.EQ.-1.AND.IY(JUC).EQ.0) GO TO 60
292           IY(JUC)=IY(JUC)+ITETA
293           GO TO 60
294        40 CALL PART2(NUC,MOUC,IY,XLB)
295           IF(XLB.GE.VX) GO TO 100
296           CALL PART3(IY,VX,JJ,NUC,MOUC,IZ)
297           DO 61 J1=1,N
298        61 IY(J1)=IZ(J1)
299        60 VY=VNORS(XLAG,IY)
300           IF(VY.GE.VX) GO TO 100
301           VXO=VY
302           DO 65 I=1,N
303        65 IXO(I)=IY(I)
304           GO TO 300
```

```
305      100 ICOUN=ICOUN-1
306          IF(ICOUN.EQ.0) GO TO 110
307          NOZE=NOZE+1
308          JARR(NOZE)=JJ
309          IKODE=1
310          GO TO 400
311      110 IF(ITETA.EQ.-1) GO TO 200
312          ITETA=-1
313          GO TO 500
314      200 DO 70 I=1,N
315       70 IXO(I)=IX(I)
316          VXO=VX
317      300 RETURN
318          END

319          SUBROUTINE PART1(IX,JJ,JL,IY,ICODE)
         C
         C   PERFORMS A TEST BY SIMPLY EVALUATING AN UPPER BOUND FOR THE LINEAR
         C   IF ICODE=1 , TEST SUCCEEDED
         C   IF ICODE=0 TEST FAILED
         C
320          DIMENSION IX(100),IY(100),IZ(100),XC(50),XA(50),JARR(100)
321          COMMON N,M/BLOCK5/ITETA/BLOCK6/WLAM/BLOCK8/NOZE,JARR
322          COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50),
            *NPA2(50),NPA3(50),NVNOR/TAB/POISON(5),100)
         C
323          NPA1(NITER)=NPA1(NITER)+1
324          M1=M+1
325          XLAG=WLAM
326          BO=-DIF(XLAG,IX,JL)
327          IF(BO.GE.0.0) GO TO 50
328          DO 10 I=1,N
329          IF(I.EQ.JJ) GO TO 11
330          IZ(I)=IX(I)
331          GO TO 10
332       11 IZ(I) IX(I)-ITETA
333       10 CONTINUE
334          BI=-DIF(XLAG,IZ,JJ)
335          DO 20 I=1,M1
336          IXS=IX(JL)+I
337          T1=PDFOP(IXS,JL)
338          T2=POISON(IXS,JL)
339          XC(I)=-T1/T2
340          IXP=IX(JJ)+I+1
341          P1=PDFOP(IXP,JJ)
342          P2=POISON(IXP,JJ)
343          XA(I)=-P1/P2
344       20 CONTINUE
345          TER1=1.
346          TER2=1.
347          IF(BI.LT.0.0) GO TO 22
348          W1=0.0
349          GO TO 23
350       22 W1=(1.1)*BO/BI
351       23 TERM=W1*BI
352          DO 30 I=1,M1
353          WI=XC(I)-W1*XA(I)
354          IF(WI) 30,30,35
355       35 DO 40 K=1,N
356          DO 37 KK=1,NOZE
```

```
      IF(JARR(KK).EQ.K) GO TO 36
   37 CONTINUE
      IXK=IX(K)+I+1
      TER1=TER1*POISON(IXK,K)
      GO TO 38
   36 IXK=IX(K)+I
      TER1=TER1*POISON(IXK,K)
   38 IF(K.EQ.JJ) GO TO 39
      IYK=IX(K)+I-1
      GO TO 41
   39 IYK=IX(K)+I+1
   41 TER2=TER2*POISON(IYK,K)
   40 CONTINUE
      UI=TER1-TER2
      TERM=TERM+UI*WI
   30 CONTINUE
      WW=TERM-BO
      ICODE=0
      IF(WW) 50,55,55
   50 ICODE=1
      IF(ITETA.EQ.-1.AND.IX(JL).EQ.0) GO TO 55
      IX(JL)=IX(JL)+ITETA
   55 DO 60 I=1,N
   60 IY(I)=IX(I)
      RETURN
      END

      SUBROUTINE PART2(NUC,MOUC,IY,XLB)
C
C   IF PART1 FAILS, THIS FINDS AN UPPER BOUND FOR THE SUM OF FIRST
C   DIFFERENCES OF THE UNDETERMINED COEFFICIENTS
C
      DIMENSION IY(100),MOUC(100),IZ(100)
      COMMON N,M/BLOCK5/ITETA/BLOCK6/WLAM
      COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50),
     *NPA2(50),NPA3(50),NVNOR
C
      NPA2(NITER)=NPA2(NITER)+1
      SUM=0.0
      DO 10 I=1,NUC
      IXK=MOUC(I)
      IXK1=IXK-1
      IF(IXK1.EQ.0) GO TO 15
      DO 20 J=1,IXK1
      IF(ITETA.EQ.-1.AND.IY(J).EQ.0) GO TO 25
      IZ(J)=IY(J)+ITETA
      GO TO 20
   25 IZ(J)=IY(J)
   20 CONTINUE
      NXK=IXK
      GO TO 35
   15 NXK=1
   35 DO 30 J=NXK,N
   30 IZ(J)=IY(J)
      TERM=DIF(WLAM,IZ,IXK)
      IF(TERM.LT.0.0) GO TO 40
      TERM=0.0
   40 SUM=SUM+TERM
   10 CONTINUE
      XLB=SUM+VNORS(WLAM,IY)
```

```
410           RETURN
411           END

412           SUBROUTINE PART3(IX,VX,JJ,NUC,MOUC,IY)
      C
      C   PERFORMS THE SAME TEST AS PART1, BUT SETS UP THE LINEAR PROGRAM EX
      C
413           DIMENSION IX(100),IY(100),IZ(100),JARR(100),MOUC(100),KOUC(100),
             *10,10)
414           COMMON N,M/BLOCK5/ITETA/BLOCK8/NOZE,JARR/BLOCK9/NROW,NCOL/BLOCK6
             *LAM
415           COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50)
             *NPA2(50),NPA3(50),NVNOR/TAB/POISON(50,100)
416           NPA3(NITER)=NPA3(NITER)+1
      C
417           NUCB=NUC
418           XLAG=WLAM
419           NROW=M+2
420           NCOL=M+3
421           NCOL1=NCOL-1
422           DO 9 I=1,N
423           IF(I.EQ.JJ) GO TO 8
424           IZ(I)=IX(I)
425           GO TO 9
426         8 IZ(I)=IX(I)-ITETA
427         9 CONTINUE
428           DO 10 ISAY=1,NUCB
429           JL=MOUC(ISAY)
430           Y(1,1)=0.0
431           DO 11 J=2,NCOL1
432           TER1=1.0
433           TER2=1.0
434           DO 40 K=1,N
435           DO 37 KK=1,NOZE
436           IF(JARR(KK).EQ.K) GO TO 36
437        37 CONTINUE
438           IXK=IZ(K)+J
439           GO TO 38
440        36 IXK=IZ(K)+J-1
441        38 TER1=TER1*POISON(IXK,K)
442           IF(K.EQ.JJ) GO TO 39
443           IYK=IZ(K)+J-1
444           GO TO 41
445        39 IYK=IZ(K)+J
446        41 TER2=TER2*POISON(IYK,K)
447        40 CONTINUE
448        11 Y(1,J)=TER1-TER2
449           Y(1,NCOL)=-DIF(XLAG,IZ,JJ)
450           DO 15 I=2,NROW
451           IXS=IX(JL)+I-1
452           Y(I,1)=PDFOP(IXS,JL)/POISON(IXS,JL)
453           IXP=IX(JJ)+I-2
454           IXR=IX(JJ)+I-1
455        15 Y(I,NCOL)=PDFOP(IXP,JJ)/POISON(IXP,JJ)
456           DO 25 I=2,NROW
457           DO 25 J=2,NCOL1
458           IF(J.EQ.I) GO TO 27
459           Y(I,J)=0.0
460           GO TO 25
461        27 Y(I,J)=-1.0
```

```
462       25 CONTINUE
463          BO=-DIF(XLAG,IX,JL)
464          DO 210 I=1,NROW
465      210 CONTINUE
466          CALL XLPRO(Y,VOBJ)
467          FARK=VOBJ-BO
468          IF(FARK.GT.0.0) GO TO 10
469          IF(IX(JL).EQ.0.AND.ITETA.EQ.-1) GO TO 50
470          IX(JL)=IX(JL)+ITETA
471       50 KUC=NUC-1
472          IF(KUC.EQ.0) GO TO 10
473          NKUC=0
474          DO 55 J=1,NUC
475          IF(MOUC(J).EQ.JL) GO TO 56
476          IF(NKUC.EQ.1) GO TO 57
477          KOUC(J)=MOUC(J)
478          GO TO 55
479       56 NKUC=1
480          GO TO 55
481       57 J1=J-1
482          KOUC(J1)=MOUC(J)
483       55 CONTINUE
484          CALL PART2(KUC,KOUC,IX,XLB)
485          IF(XLB.GE.VX) GO TO 500
486          NUC=KUC
487          DO 60 I=1,NUC
488       60 MOUC(I)=KOUC(I)
489       10 CONTINUE
490      500 DO 70 I=1,N
491       70 IY(I)=IX(I)
492          RETURN
493          END

494          FUNCTION DIF(XLAG,IX,K)
         C
         C    CALCULATES THE FIRST DIFERENCES
         C
495          DIMENSION IX(100)
496          COMMON N,M/BLOCK1/C(100)/TAB/POISON(50,100)
497          DOUBLE PRECISION PRO,SUM
498          M1=M+1
499          SUM=0.D+00
500          DO 20 J1=1,M1
501          PRO=1.D+00
502          DO 30 I=1,N
503          IXJ=IX(I)+J1
504          IF(I.EQ.K) GO TO 25
505          PRO=PRO*POISON(IXJ,I)
506          GO TO 30
507       25 PRO=PRO*PDFOP(IXJ,I)
508       30 CONTINUE
509       20 SUM=SUM+PRO
510          IXM=IX(K)+M+2
511          DIF=-SUM-1.+POISON(IXM,K)+XLAG*C(K)
512          RETURN
513          END

514          FUNCTION VNORS(XLAG,IX)
         C
         C    EVALUATES THE FUNCTION E(NORS/X)+L.C.X
```

```
      C
515         DIMENSION IX(100),PM(100,20),PRO(20)
516         COMMON N,M
517         COMMON /COUNT/NITER,NBAS(50),NSEA(50),NTAR(50),NARA(50),NPA1(50)
           *NPA2(50),NPA3(50),NVNOR/TAB/POISON(50,100)
518         DOUBLE PRECISION SUM,PRO,TOPL,PM,XW
519         NVNOR=NVNOR+1
520         EPS=.00001
521         M1=M+1
522         SUM=0.D+00
523         DO 20 JJ=1,M1
524         PRO(JJ)=1.D+00
525         DO 25 II=1,N
526         IXJ=IX(II)+JJ-1
527         IF(IXJ.GT.0) GO TO 24
528         XX=PDFOP(IXJ,II)
529         GO TO 25
530      24 XX=POISON(IXJ,II)
531      25 PRO(JJ)=PRO(JJ)*XX
532      20 SUM=SUM+1.-PRO(JJ)
533         TOPL=0.D+00
534         DO 30 I=1,N
535         K=0
536      40 MXK=IX(I)+M+2+K
537         WX=PDFOP(MXK,I)
538         WW=FLOAT(K+1)*WX
539         IF(WX.LT.EPS) GO TO 30
540         TOPL=TOPL+WW
541         K=K+1
542         GO TO 40
543      30 CONTINUE
544         VNORS=SUM+TOPL+XLAG*DOTPR(IX)
545         RETURN
546         END

547         SUBROUTINE TABLE(POISON)
      C
      C   CALCULATES THE CUMULATIVE POISSON SUMS
      C
548         DIMENSION POISON(50,100)
549         COMMON N,M/BLOCK2/XLAM(100)
550         DOUBLE PRECISION EPS,TERM,SUMX
551         EPS=1.D-06
552         DO 10 J=1,N
553         TERM=1.D+00
554         SUMX=1.D+00
555         DO 20 I=1,50
556         TERM=TERM*XLAM(J)/FLOAT(I)
557         IX=I
558         IF(TERM.LT.EPS) GO TO 25
559         SUMX=SUMX+TERM
560         POISON(I,J)=EXP(-XLAM(J))*SUMX
561         IF(POISON(I,J).GT.1.0) POISON(I,J)=1.0
562      20 CONTINUE
563      25 IF(IX.EQ.50) GO TO 10
564         DO 30 I=IX,50
565      30 POISON(I,J)=1.0
566      10 CONTINUE
567         RETURN
568         END
```

```
569           FUNCTION PDFOP(K,I)
      C
      C    CALCULATES THE INDIVIDUAL POISSON TERMS
      C
570           COMMON /BLOCK2/XLAM(100)
571           DOUBLE PRECISION SUM,WW
572           SUM=0.D+00
573           IF(K.LT.2) GO TO 10
574           DO 15 J=2,K
575           SUM=SUM+ALOG(FLOAT(J))
576        15 CONTINUE
577        10 WW=-XLAM(I)+K*ALOG(XLAM(I))-SUM
578           IF(WW.LT.-15.0) GO TO 20
579           PDFOP=DEXP(WW)
580           GO TO 25
581        20 PDFOP=0.0
582        25 RETURN
583           END

584           FUNCTION DOTPR(IX)
      C
      C    CALCULATES THE DOT PRODUCT C.X
      C
585           DIMENSION IX(100)
586           COMMON N,M/BLOCK1/C(100)
587           SUM=0.0
588           DO 10 I=1,N
589           XX=FLOAT(IX(I))
590        10 SUM=SUM+XX*C(I)
591           DOTPR=SUM
592           RETURN
593           END

594           SUBROUTINE SIRALA(X,IR)
      C
      C    THIS ARRANGES AN ARRAY OF ELEMENTS IN ASCENDING ORDER
      C    IR(K) IS THE RANK OF ITEM K INTHE LIST
      C
595           DIMENSION X(100), IR(100), ISEQ(100), Y(100)
596           COMMON N,M
597           DO 9 I=1,N
598         9 ISEQ(I)=I
599           DO 10 I=1,N
600           XMIN=X(I)
601           IMINJ=I
602           IMIN=ISEQ(I)
603           N1=N-I+1
604           DO 20 J=1,N1
605           IF(X(J).LT.XMIN) GO TO 21
606           GO TO 20
607        21 XMIN=X(J)
608           IMIN=ISEQ(J)
609           IMINJ=J
610        20 CONTINUE
611           IR(IMIN)=I
612           IF(N1.EQ.1) GO TO 10
613           N2=N1-1
614           DO 30 J=1,N2
615           IF(J.LT.IMINJ) GO TO 33
616           J1=J+1
```

```
      ISEQ(J)=ISEQ(J1)
      Y(J)=X(J1)
      GO TO 29
   33 Y(J)=X(J)
   29 X(J)=Y(J)
   30 CONTINUE
   10 CONTINUE
      RETURN
      END

      SUBROUTINE XLPRO(Y,VOBJ)
C
C   SOLVES A LINEAR PROGRAM OF THE TYPE
C   MAX CX ST AX<B,X>0
C
      DIMENSION Y(10,20),X(10,20),ISNB(10),ISBV(10),MROW(10)
      COMMON /BLOCK9/NROW,NCOL
C
      ISBV(1)=0
      ISNB(1)=NCOL+1
      DO 10 I=2,NROW
   10 ISBV(I)=I-2+NCOL
      DO 11 J=2,NCOL
   11 ISNB(J)=J-1
      NROW1=NROW-1
      NCOL1=NCOL-1
      NCOL2=NCOL+1
      NCOL3=NCOL+NROW1
      DO 12 I=1,NROW
      DO 12 J=NCOL2,NCOL3
      IM=I-1
      JM=J-NCOL
      IF(JM.EQ.IM) GO TO 13
      Y(I,J)=0.0
      GO TO 12
   13 Y(I,J)=1.0
   12 CONTINUE
  100 NNC=0
      NPV=0
      NPC=0
      NNV=0
      DO 15 J=2,NCOL
      IF(Y(1,J).LT.0.0) GO TO 16
      NNC=NNC+1
      IF(NNC.EQ.NCOL1) GO TO 500
      GO TO 15
   16 NPC=NPC+1
      IF(NPC.EQ.1) GO TO 17
      IF(Y(1,J).GT.YMIN) GO TO 15
   17 YMIN=Y(1,J)
      JMIN=J
   15 CONTINUE
      IEV=ISNB(JMIN)
      NNVA=1
      DO 20 I=2,NROW
      IF(Y(I,JMIN).LE.0.0) GO TO 25
      NNV=NNV+1
      TETA=Y(I,1)/Y(I,JMIN)
      IF(NNV.EQ.1) GO TO 23
      IF(TETA-TMIN) 23,20,20
```

```
671        23 TMIN=TETA
672           IMIN=I
673           MROW(NMVA)=IMIN
674           GO TO 25
675        25 NPV=NPV+1
676           IF(NPV.EQ.NROW1) GO TO 450
677        20 CONTINUE
678           DO 26 I=2,NROW
679           IF(Y(I,JMIN).LE.0.0) GO TO 26
680           TETA=Y(I,1)/Y(I,JMIN)
681           IF(TETA.GT.TMIN) GO TO 26
682           IF(I.EQ.IMIN) GO TO 26
683           NMVA=NMVA+1
684           MROW(NMVA)=I
685        26 CONTINUE
686           IF(NMVA.EQ.1) GO TO 200
687           DO 80 I=1,NMVA
688           IK=MROW(I)
689           JPOS=0
690           DO 80 J=NCOL2,NCOL3
691           X(IK,J)=Y(IK,J)/Y(IK,JMIN)
692           IF(X(IK,J).GT.0.0) GO TO 85
693           GO TO 80
694        85 JPOS=JPOS+1
695           IF(JPOS.GT.1) GO TO 80
696           IF(IK.GT.MROW(1)) GO TO 87
697           GO TO 93
698        87 IF(J-JKMIN) 80,90,93
699        90 IF(X(IMIN,JKMIN).LE.X(IK,J)) GO TO 80
700        93 IMIN=IK
701           JKMIN=J
702        80 CONTINUE
703       200 IDV=ISBV(IMIN)
704           ISNB(JMIN)=IDV
705           ISBV(IMIN)=IEV
706           DO 40 I=1,NROW
707           DO 40 J=1,NCOL3
708           IF(I.EQ.IMIN) GO TO 55
709           IF(J.EQ.JMIN) GO TO 60
710           X(I,J)=Y(I,J)-(Y(I,JMIN)*Y(IMIN,J)/Y(IMIN,JMIN))
711           GO TO 40
712        55 IF(J.EQ.JMIN) GO TO 57
713           X(I,J)=Y(I,J)/Y(IMIN,JMIN)
714           GO TO 40
715        57 X(I,J)=1./Y(I,J)
716           GO TO 40
717        60 X(I,J)=-Y(I,J)/Y(IMIN,JMIN)
718        40 CONTINUE
719           DO 70 I=1,NROW
720           DO 70 J=1,NCOL3
721        70 Y(I,J)=X(I,J)
722           GO TO 100
723       450 WRITE(6,250)
724       250 FORMAT(5X,'UNBOUNDED SOLUTION')
725       500 RETURN
726           END

      $ENTRY
```

```
THE SADDLE POINT SEARCH ALGORITHM


PROBLEM NUMBER   1

NUMBER OF ITEMS=    5  BUDGET=$  25000.00


ITEM    COST/UNIT        DEMAND     INITIAL SOLUTION     OPTIMAL SOLUTION

#  1      2980.00          2.10              2                   2
#  2      1751.00          1.50              2                   2
#  3       462.00          1.20              2                   3
#  4      1500.00          5.00              6                   7
#  5       345.00          3.50              6                   6

E(NORS/X)=    1.06282      COST(X)=$  23418.00

LAGRANGE MULTIPLIER=  0.000090

# OF ITERATIONS=   6

# OF NORS EVALUATIONS= 141

# OF  STEPS IN EACH ITERATION

 ITERATION #     STEP1  STEP2  STEP3  STEP4  PART1  PART2  PART3
           1         1      1      1      1      5      0      0
           2         1      5      5      2      0      0      0
           3         1      4      4      2      3      0      0
           4         1      2      2      1      1      0      0
           5         1      2      2      1      0      0      0
           6         1      2      2      1      2      0      0

STATEMENTS EXECUTED=  273427

CORE USAGE       OBJECT CODE=   31504 BYTES, ARRAY AREA=   48572 BYTES, TOTAL

DIAGNOSTICS        NUMBER OF ERRORS=       0, NUMBER OF WARNINGS=       0, N

COMPILE TIME=     0.52 SEC, EXECUTION TIME=     5.02 SEC,     12.27.33     TUE

        C$STOP
```

APPENDIX G

COMPUTER CODE FOR THE SIMULATION MODEL

A List of Non-GASP Variables in the

Computer Program

BOZUL	The subroutine which handles the breakdowns of airplanes
CANZE	The subroutine which performs the cannibalization
IKIT(100)	The kit obtained by an analytical solution method
KIT(100)	Current kit during the simulation
MODE	Repair code; 0 implies "no repair", and 1 implies "repair"
N	The total number of items in the kit
NNORS	Number of NORS planes
NOAPL	Number of airplanes flying at any time
NPIS	Number of airplanes in a squadron initially
RSTAR	The subroutine for reinitializing the kit and NNORS at the beginning of each flying period
XLAM(100)	Mean time before failure

FORTRAN Listing of the Simulation Model

```
FORTRAN IV G1  RELEASE 2.0            MAIN              DATE = 80217      23/44/02          PAGE 0001

                C ************************************
                C *                                  *
                C *     SIMULATION MODEL             *
                C *                                  *
                C************************************
                C
                C   MAIN PROGRAM
                C
0001                  DIMENSION NSET(1000)
0002                  COMMON QSET(1000)
0003                  COMMON /GCOM1/ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,NN
                     1APO,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TTBEG,
                     2TTCLR,TTFIN,TTRIB(25),TTSET
0004                  COMMON/JCOM1/ N,KIT(100),XLAM(100)/JCOM2/ARRAY(10,100),ITSET(10,10
                     10), IKIT(100)/UCOM3/WDAY,NNORS,NPIS,NOAPL
0005                  EQUIVALENCE(NSET(1),QSET(1))
                C
                C   INPUT THE DATA
                C
0006                  NCRDR=5
0007                  NPRNT=6
0008                  READ(NCRDR,100) N
0009                  READ(NCRDR,100) NPIS
0010                  READ(NCRDR,101) (IKIT(I),I=1,N)
0011                  READ(NCRDR,102) (XLAM(I),I=1,N)
0012                  DO 10 I=1,N
0013               10 XLAM(I)=60./XLAM(I)
                C
                C   WRITE OUT THE DATA
                C
0014                  WRITE(NPRNT,200) NPIS
0015                  WRITE(NPRNT,201) (IKIT(I),I=1,N)
0016                  WRITE(NPRNT,202) (XLAM(I),I=1,N)
                C
                C   START THE SIMULATION
                C
0017                  CALL GASP
                C
```

```
0018              100 FORMAT(I4)
0019              101 FORMAT(40I2)
0020              102 FORMAT(10F8.0)
0021              200 FORMAT(/,5X,'NUMBER OF PLANES IN THE SQUADRON=',I4)
0022              201 FORMAT(/,5X,'KIT:',10I4)
0023              202 FORMAT(/,5X,'MTBF:',10F8.2)
0024                  STOP
0025                  END
```

```
FORTRAN IV G1  RELEASE 2.0            INTLC          DATE = 80217      23/44/02          PAGE 0001

0001                  SUBROUTINE INTLC
                C
0002                  COMMON /GCOM1/ATRIB(25),JEVNT,MFA,MFE(100),M_E(100),MSTOP,NCRDR,NN
                     1APO,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TTBEG,
                     2TTCLR,TTFIN,TTRIB(25),TTSET
0003                  COMMON /GCOM6/ EENQ(100),IINN(100),KKRNK(100),MMAXQ(100),QQTIM(100
                     1),SSOBV(25,5),SSTPV(25,6),VVNQ(100)
0004                  COMMON/UCOM1/ N,KIT(100),XLAM(100)/UCOM2/ARRAY(10,100),ITSET(10,10
                     10),IKIT(100)/UCOM3/WDAY,NNORS,NPIS,NOAPL
                C
0005                  NOAPL=NPIS
0006                  WDAY=2.
0007                  NNORS=0
0008                  A=FLOAT(NNORS)
                C   COLLECT STATISTICS
0009                  CALL TIMST(A,TNOW,1)
                C
                C   INITIALIZE THE KIT
                C
0010                  DO 15 I=1,N
0011               15 KIT(I)=IKIT(I)
```

```
C
C   CAUSE THE FIRST FAILURES
C
0012          DO 10 K=1,NOAPL
0013          DO 20 I=1,N
0014          ARRAY(K,I)=-XLAM(I)*ALOG(DRAND(K))+.10536
0015          IF(I.EQ.1) GO TO 25
0016          IF(ARRAY(K,I).GE.BMIN) GO TO 20
0017       25 BMIN=ARRAY(K,I)
0018          NUMB=I
0019       20 CONTINUE
0020          ATRIB(1)=BMIN
0021          ATRIB(2)=1.0
0022          ATRIB(3)=K
0023          ATRIB(4)=NUMB
0024          CALL FILEM (1)
0025       10 CONTINUE
C
C   SCHEDULE MAINTENANCE
0026          ATRIB(1)=WDAY
0027          ATRIB(2)=2.
0028          CALL FILEM (1)
C
C   SCHEDULE THE NEXT FLYING PERIOD
0029          ATRIB(1)=60.
0030          ATRIB(2)=3.
0031          CALL FILEM (1)
0032          RETURN
0033          END
```

```
0001                  SUBROUTINE RSTAR
                C
0002                  COMMON /SCOML/ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,NN
                     1APO,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TTBEG,
                     2TTCLR,TTFIN,TTRIB(25),TTSET
0003                  COMMON /SCOM6/ EENQ(100),IINN(100),KKRNK(100),MMAXQ(100),QQTIM(100
                     1),SSOBV(25,5),SSTPV(25,6),VVN2(100)
0004                  COMMON/UCOM1/ N,KIT(100),XLAM(100)/JCOM2/ARRAY(10,100),ITSET(10,10
                     10),IKIT(100)/UCOM3/NDAY,NNORS,NPIS,NOAPL
                C
0005                  ATRIB(1)=TNOW+50.
0006                  ATRIB(2)=3.
0007                  CALL FILEM (1)
                C
                C   REINITIALIZE THE KIT
                C
0008                  DO 10 I=1,N
0009                10 KIT(I)=IKIT(I)
                C
                C   REINITIALIZE NUMBER OF NORS PLANES
0010                  NNORS=0
                C
                C   COLLECT STATISTICS
0011                  A=FLOAT(NNORS)
0012                  CALL TIMST(A,TNOW,1)
0013                  NOAPL=NPIS
                C
                C   SCHEDULE THE BREAKDOWNS
                C
0014                  DO 20 K=1,NOAPL
0015                  DO 25 I=1,N
0016                  ARRAY(K,I)=TNOW-XLAM(I)*ALOG(DRAND(K))+.10536
0017                  IF(I.EQ.1) GO TO 27
0018                  IF(ARRAY(K,I).GT.BRTIM) GO TO 25
0019                27 BRTIM=ARRAY(K,I)
0020                  NUMB=I
0021                25 CONTINUE
0022                  ATRIB(1)=BRTIM
0023                  ATRIB(2)=1.0
0024                  ATRIB(3)=K
0025                  ATRIB(4)=NUMB
0026                  CALL FILEM (1)
0027                20 CONTINUE
```

```
0028                    IF(NNQ(2).EQ.0) RETURN
0029                    NXX=NNQ(2)
0030                    DO 15 J=1,NXX
0031                    CALL RMOVE(MFE(2),2)
0032                 15 CONTINUE
0033                    RETURN
0034                    END
```

```
FORTRAN IV G1  RELEASE 2.0              EVNTS              DATE = 80217          23/44/32          PAGE 0001

0001                    SUBROUTINE EVNTS(IX)
0002                    GO TO (1,2,3), IX
0003                  1 CALL BOZUL
0004                    GO TO 10
0005                  2 CALL CANZE
0006                    GO TO 10
0007                  3 CALL RSTAR
0008                 10 RETURN
0009                    END
```

```
FORTRAN IV G1  RELEASE 2.0              TAMIR              DATE = 80217          23/44/32          PAGE 0001

0001                    SUBROUTINE TAMIR(K,KTOT,MODE)
                  C
0002                    COMMON/UCOM1/ N,KIT(100),XLAM(100)/UCOM2/ARRAY(10,100),ITSET(10,10
                       10),IKIT(100)/UCOM3/WDAY,NNORS,NPIS,NOAPL
                  C
                  C
                  C   REPAIR THE PLANE IF NEEDED ITEM IS AVAILABLE
                  C
0003                    IF(KIT(KTOT).EQ.0) GO TO 50
0004                    MODE=1
                  C
                  C   UPDATE THE KIT
                  C
0005                    KIT(KTOT)=KIT(KTOT)-1
0006                    RETURN
0007                 50 MODE=0
0008                    RETURN
0009                    END
```

```
FORTRAN IV G1  RELEASE 2.0          BOZUL          DATE = 80217        23/44/02          PAGE 0001

0001                  SUBROUTINE BOZUL
                C
0002                  COMMON /GCOM1/ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,NN
                     1APO,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NPRNT,PPARM(5),4),TNOW,TTBEG,
                     2TTCLR,TTFIN,TTRIB(25),TTSET
0003                  COMMON/UCOM1/ N,XIT(100),XLAM(100)/UCOM2/ARRAY(10,100),ITSET(10,10
                     10),IKIT(100)/UCOM3/WDAY,NNORS,NPIS,NOAPL
                C
0004                  KPLAN=IFIX(ATRIB(3))
0005                  NUMB=IFIX(ATRIB(4))
                C
                C   REPAIR THE PLANE KPLAN WITH SPARE PART NUMB
                C
0006                  CALL TAMIR(KPLAN,NUMB,MODE)
0007                  IF(MODE.EQ.0) GO TO 100
                C
                C   IF REPAIRED , SCHEDULE ITS BREAKDOWN
                C
0008                  IPART=NUMB
0009                  ARRAY(KPLAN,IPART)=TNOW-XLAM(IPART)*ALOG(DRAND(KPLAN))+.10536
0010                  DO 15 I=1,N
0011                  IF(I.EQ.1) GO TO 18
0012                  IF(ARRAY(KPLAN,I).GE.BTIME) GO TO 15
0013               18 BTIME=ARRAY(KPLAN,I)
0014                  NUMB=I
0015               15 CONTINUE
0016                  ATRIB(1)=BTIME
0017                  ATRIB(2)=1.0
0018                  ATRIB(3)=KPLAN
0019                  ATRIB(4)=NUMB
0020                  CALL FILEM (1)
0021                  RETURN
0022              100 ATRIB(3)=KPLAN
0023                  ATRIB(4)=NUMB
0024                  CALL FILEM (2)
                C
                C   IF NOT REPAIRED DUE TO PARTS SHORTAGES, PUT IN THE REPAIR QUEUE
                C
0025                  NOAPL=NOAPL-1
0026                  RETURN
0027                  END
```

```
0001                  SUBROUTINE CANZE
                C
0002                  COMMON /GCOM1/ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,NN
                     1APQ,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NPRNT,PPARM(50,5),TNOW,TTBEG,
                     2TTCLR,TTFIN,TTRIB(25),TTSET
0003                  COMMON /GCOM6/ EENQ(100),IINN(100),KKRNK(100),MMAXQ(100),QQTIM(100
                     1),SSQBV(25,5),SSTPV(25,6),VVNQ(100)
0004                  COMMON/UCOM1/ N,KIT(100),XLAM(100)/UCOM2/ARRAY(10,100),ITSET(10,100
                     10),IKIT(100)/UCOM3/WDAY,NNORS,NPIS,NOAPL
                C
0005                  ATRIB(1)=TNOW+WDAY
0006                  ATRIB(2)=2.0
0007                  CALL FILEM (1)
                C
                C   IF THERE IS ONLY ONE PLANE IN THE REPAIR QUEUE, NO CANNIBILIZATION
0008                  IF(NNQ(2).LE.1) RETURN
                C
                C   CANNIBILIZE ONE TO REPAIR THE OTHERS
                C
0009                50 NNORS=NNORS+1
0010                  A=FLOAT(NNORS)
                C
                C   COLLECT STATISTICS ON NUMBER OF CANNIBILIZED PLANES
                C
0011                  CALL TIMST(A,TNOW,1)
0012                  IF(NNORS.EQ.NPIS) RETURN
0013                  CALL RMOVE(MFE(2),2)
0014                  NNIK=IFIX(ATRIB(4))
0015                  KPLAN=ATRIB(3)
0016                  DO 10 I=1,N
0017                  IF(I.EQ.NNIK) GO TO 10
                C
                C   UPDATE THE KIT
0018                  KIT(I)=KIT(I)+1
0019               10 CONTINUE
0020                  NPIQ=NNQ(2)
```

```
C
C    CHECK THE REPAIR QUEUE IF ANY PLANE IS AWAITING REPAIR
0021        DO 20 I=1,NPIQ
0022        CALL RMOVE(MFE(2),2)
0023        DIF=TNOW-ATRIB(1)
0024        KP=ATRIB(3)
0025        KTOT=ATRIB(4)
0026        DO 21 J=1,N
0027     21 ARRAY(KP,J)=ARRAY(KP,J)+DIF
C
C    REPAIR THEM & UPDATE THE KIT
C
0028        CALL TAMIR(KP,KTOT,MODE)
0029        IF(MODE.EQ.0) GO TO 25
0030        NOAPL=NOAPL+1
C
C    SCHEDULE THE NEXT BREAKDOWN
C
0031        ARRAY(KP,KTOT)=TNOW-XLAM(KTOT)*ALOG(DRAND(KP))+.10536
0032        DO 35 J=1,N
0033        IF(J.EQ.1) GO TO 37
0034        IF(ARRAY(KP,J).GE.BRTIM) GO TO 35
0035     37 BRTIM=ARRAY(KP,J)
0036        NUMB=J
0037     35 CONTINUE
C
C    RECORD THE BREAKDOWN TIME, PLANE NO, SPARE PART
C
0038        ATRIB(1)=BRTIM
0039        ATRIB(2)=1.0
0040        ATRIB(3)=KP
0041        ATRIB(4)=NUMB
0042        CALL FILEM (1)
0043        GO TO 20
C
C    IF NOT REPAIRED KEEP IN THE REPAIR QUEUE
C
0044     25 ATRIB(3)=KP
0045        ATRIB(4)=KTOT
0046        CALL FILEM (2)
0047     20 CONTINUE
0048        IF(NNQ(2).GT.1) GO TO 50
0049        RETURN
0050        END
```

NUMBER OF PLANES IN THE SQUADRON= 6

KIT:	3	3	3	8	7	3	5	6	3	16
KIT:	1	4	13	12	13	11	4	8	12	5
KIT:	21	10	2	14	3	11	11	3	13	5
KIT:	7	16	22	5	9	9	3	3	7	8
KIT:	3	4	8	4	13	5	4	2	1	17
KIT:	8	6	8	6	5	4	14	3	11	15
KIT:	3	2	7	2	2	10	2	19	9	3
KIT:	6	4	9	6	16	2	5	7	3	17

MTBF:	28.57	40.00	50.00	12.00	17.14	33.33	13.46	20.00	46.15	5.00
MTBF:	120.00	30.00	8.45	7.06	6.45	12.00	33.33	17.65	9.23	23.08
MTBF:	5.31	10.00	60.00	7.23	46.15	8.57	10.91	40.00	8.57	21.43
MTBF:	19.35	5.22	4.62	17.65	13.95	11.76	54.55	54.55	15.22	18.75
MTBF:	30.00	30.00	8.70	20.69	7.50	26.09	31.58	50.00	120.00	5.08
MTBF:	15.00	26.67	18.18	18.18	25.00	42.86	6.90	54.55	8.96	6.12
MTBF:	46.15	52.17	15.00	57.14	85.71	8.82	66.67	5.00	15.00	28.57
MTBF:	20.00	50.00	12.77	13.95	6.00	75.00	20.00	17.14	66.67	5.77

```
                    SIMULATION PROJECT NUMBER   1  BY       UNIT

                    DATE  5/ 19/   80        RUN NUMBER    1 OF    1
                    LLSUP=0000000000000000    GASP IV VERSION 25JAN75

NNCLT=      0    NNSTA=      1    NNHIS=      0    NNPRM=      0    NNPLT=      0    NNSTR=      6    NNTRY=    100
NMATR=      4    NNFIL=      2    NNSET=   1000    NNEQD=      0    NNEQS=      0    NFLAG=      0

TIMST NO.   1    LLABT=E(NORS)    I.C. =      0.0

KKRNK= (    1)      4

IIAN = (    1)      2

MSTOP=      1    JJCLR=      1    JJBEG=      1    IICRD=      0    TTBEG=      0.0          TTFIN=      0.1200E+04
JJFIL=      1
IISED= 12345          34567          55789          54321          22345          31277
```

```
**GASP FILE STORAGE AREA DUMP AT TIME  0.0        **

MAXIMUM NUMBER OF ENTRIES IN FILE STORAGE AREA =  8

                PRINTOUT OF FILE NUMBER   1
                    TNOW =  0.0
                    QQTIM=  0.0

                          FILE CONTENTS
ENTRY   1  =  0.1088E+00  0.1000E+01  0.6000E+01  0.2700E+02
ENTRY   2  =  0.1234E+00  0.1000E+01  0.5000E+01  0.9000E+01
ENTRY   3  =  0.1377E+00  0.1000E+01  0.1000E+01  0.3200E+02
ENTRY   4  =  0.2378E+00  0.1000E+01  0.3000E+01  0.2400E+02
ENTRY   5  =  0.2827E+00  0.1000E+01  0.4000E+01  0.7800E+02
ENTRY   6  =  0.9802E+00  0.1000E+01  0.2000E+01  0.5700E+02
ENTRY   7  =  0.2000E+01  0.2000E+01  0.6000E+01  0.2700E+02
ENTRY   8  =  0.6000E+02  0.3000E+01  0.6000E+01  0.2700E+02

                PRINTOUT OF FILE NUMBER   2
                    TNOW =  0.0
                    QQTIM=  0.0

                THE FILE IS EMPTY
```

GASP FILE STORAGE AREA DUMP AT TIME 0.1200E+04

MAXIMUM NUMBER OF ENTRIES IN FILE STORAGE AREA = 9

PRINTOUT OF FILE NUMBER 1
TNOW = 0.1200E+04
QQTIM= 0.1200E+04

TIME PERIOD FOR STATISTICS 0.1200E+04
AVERAGE NUMBER IN FILE 3.2840
STANDARD DEVIATION 2.0534
MAXIMUM NUMBER IN FILE 8

FILE CONTENTS

ENTRY	1	=	0.1200E+04	0.2000E+01	0.6000E+01	0.2700E+02
ENTRY	2	=	0.1200E+04	0.1000E+01	0.3000E+01	0.7400E+02
ENTRY	3	=	0.1200E+04	0.1000E+01	0.6000E+01	0.7300E+02
ENTRY	4	=	0.1200E+04	0.1000E+01	0.2000E+01	0.3200E+02
ENTRY	5	=	0.1200E+04	0.1000E+01	0.5000E+01	0.2100E+02
ENTRY	6	=	0.1200E+04	0.1000E+01	0.4000E+01	0.3100E+02
ENTRY	7	=	0.1201E+04	0.1000E+01	0.1000E+01	0.7000E+01
ENTRY	8	=	0.1260E+04	0.3000E+01	0.5000E+01	0.2700E+02

PRINTOUT OF FILE NUMBER 2
TNOW = 0.1200E+04
QQTIM= 0.1200E+04

TIME PERIOD FOR STATISTICS 0.1200E+04
AVERAGE NUMBER IN FILE 0.9359
STANDARD DEVIATION 0.5307
MAXIMUM NUMBER IN FILE 5

THE FILE IS EMPTY

GASP SUMMARY REPORT

SIMULATION PROJECT NUMBER 1 BY UMIT

DATE 5/ 19/ 80 RUN NUMBER 1 OF 1

CURRENT TIME = 0.1200E+04

STATISTICS FOR TIME-PERSISTENT VARIABLES

	MEAN	STD DEV	MINIMUM	MAXIMUM	TIME INTERVAL	CUR. VALUE
E(NORS)	0.3730E+01	0.1875E+01	0.0	0.5000E+01	0.1200E+04	0.0

VITA

Umit Yuceer

Candidate for the Degree of

Doctor of Philosophy

Thesis: OPTIMIZATION OF SINGLE RESOURCE ALLOCATION PROBLEMS WITH NONSEPARABLE OBJECTIVE FUNCTIONS

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Solhan, Bingol, Turkey, May 18, 1950, the son of Mr. and Mrs. Hikmet Yuceer.

Education: Graduated from Science Lycée, Ankara, Turkey, in June, 1967; received Bachelor of Science in June, 1972, and Master of Science in July, 1974, in Mathematics from the Middle East Technical University, Ankara, Turkey; received Master of Science in Engineering from the Johns Hopkins University, Baltimore, Maryland; completed requirements for the Degree of Doctor of Philosophy at Oklahoma State University in December, 1980.

Professional Experience: Research Engineer in the Petroleum Office of Turkey, Ankara, Turkey, August, 1973 - June, 1974; Teaching Assistant in the Department of Mathematical Sciences in Johns Hopkins University, September, 1975 - December, 1976; Graduate Research Assistant in the School of Industrial Engineering and Management in Oklahoma State University, September, 1977 to present.

Professional Organizations: American Institute of Industrial Engineers, Alpha Pi Mu.