INTERVAL CONVEX PROGRAMMING, ORTHOGONAL LINEAR PROGRAMMING, AND PROGRAM GENERATION PROCEDURES

by

John Heard Ristroph

Dissertation submitted to the Graduate Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
in
Industrial Engineering and Operations Research

APPROVED:

G. K. Bennett, Co-chairman

J. W. Schmidt, Jr., Co-chairman

M. H. Agee

D. M. Miller

J. Washenberger

December, 1975

Blacksburg, Virginia

ACKNOWLEDGEMENTS

The author wishes to thank the members of his committee, Drs. Agee, Miller and Washenberger, and his co-chairmen, Drs. Bennett and Schmidt. Special thanks is due to Dr. Bennett who served as the sole chairman during the developmental stages of the research. Also it was Dr. Bennett who conceived the idea of formulating the linear programming problem in the interval number system. The research effort spanned several years, and without the continued interest and assistance of these men it would not have been completed. Also the continuing support and encouragement of the author's family, particularly his wife, Margaret M. Ristroph, is gratefully acknowledged.

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

RESEARCH OBJECTIVES

This research develops certain topics in the subject area known as mathematical programming. The general form of the mathematical programming problem (MPP) is:

$$\text{maximize} \quad f(x), \tag{1.1}$$

subject to:

$$g_i(x) \leq b_i, \; i \in I_1, \tag{1.2}$$

$$g_i(x) \geq b_i, \; i \in I_2, \tag{1.3}$$

$$g_i(x) = b_i, \; i \in I_3, \tag{1.4}$$

where $f(x)$ and the $g_i(x)$ are real valued functions of the n dimensional real vector x, and I_1, I_2, and I_3 are index sets. The b_i are also real numbers. Relation 1.1 is referred to as the objective function, and relations 1.2 through 1.4 are the constraint set. A point x is feasible if it is such that the constraints are not violated at x; otherwise x is called an infeasible point. The set of all feasible points is called the feasible region. An optimum, x^*, is a feasible point such that the value of the objective function at any other feasible point is not greater than $f(x^*)$. The MPP is to determine x^*.

A particular type of MPP is the convex programming problem (CPP). In the CPP the objective function is concave, and the feasible region is convex. A function, h, is said to be concave [45] if and only if:

$$\theta h(x) + (1-\theta)h(y) \leq h[\theta x + (1-\theta)y]$$

where

$$0 \leq \theta \leq 1.$$

A function is convex if its negative, -h, is concave. A set, S, is convex if and only if when x and y are any two points of S, then z is also a point of S, where

$$z = \theta x + (1-\theta)y$$

and

$$0 \leq \theta \leq 1.$$

This research is concerned with the development of a method for determining the range of optimal points of a convex programming problem when the parameters of the problem (for example, the b_i) are not real numbers, but rather entire closed intervals of real numbers. Other related topics which are developed include methods to facilitate the solution of, and to solve, linear programming problems (LPPs). A LPP is a special type of CPP in which all functions are linear. Also methods for generating MPPs with parameters chosen randomly, but with known solutions, are presented. Illustrative examples of these techniques are provided.

ORGANIZATION

In the remaining sections of chapter one, first a review of convex programming is given; then the problem of parametric variation in linear programs and the utility of an interval formulation are discussed. Next a review of some topics of interval arithmetic is presented, followed by a chapter summary.

The development of the solution technique for CPPs with interval parameters is the topic of chapter two. Chapter three contains a description of three heuristic algorithms designed to determine good starting points for use in the solution of LPPs, and a new algorithm which yields exact solutions to the LPP is also presented. The methods for obtaining a MPP with randomly chosen coefficients, but known solution, is presented in chapter four. The final chapter contains the summary and suggestions for further research.

CONVEX PROGRAMMING

General

In the section on research objectives, the MPP, CPP, and LPP were presented without comment. Various aspects of these problems pertinent to the research will now be reviewed in greater detail.

It is interesting to note that although the CPP is but a particular type of MPP, the techniques which solve

the CPP are often used to solve the more general MPP. The MPP is treated as though it were a CPP, and a x^* is found. However, instead of being the optimum of the entire feasible region, this x^* may only be a local optimum. Thus, generally when CPP methods are used to solve the MPP, one attempts to determine all local optima by choosing different starting points for the CPP algorithm. The best of these local optima is assumed to be the global optimum. However, if all local optima are not found, then there is no guarantee that the true global optimum has been found.

The origin of CPP solution techniques may be traced most directly to the work of George B. Dantzig [9]. In 1947 he devised the Simplex solution procedure to the LPP. Then in 1951 Kuhn and Tucker developed the necessary and sufficient conditions for a point to be an optimum to the CPP [14]. The Kuhn-Tucker conditions state that a point x is an optimal solution to the CPP if and only if there exist variables $u_1,\ldots,u_m$ such that

$$\frac{\partial}{\partial x_j}\{f(x)-\sum_{i=1}^{m} u_i[g_i(x)-b_i]\} = 0,\ j=1,\ldots,n, \qquad 1.5$$

$$u_i[g_i(x)-b_i] = 0,\ i=1,\ldots,m, \qquad 1.6$$

$$u_i \geq 0,\ i\in I_1, \qquad 1.7$$

$$u_i \leq 0,\ i\in I_2, \qquad 1.8$$

$$u_i \text{ unrestricted},\ i\in I_3, \qquad 1.9$$

$$g_i(x) \leq b_i, \quad i\varepsilon I_1, \tag{1.10}$$

$$g_i(x) \geq b_i, \quad i\varepsilon I_2, \tag{1.11}$$

$$g_i(x) = b_i, \quad i\varepsilon I_3, \tag{1.12}$$

These conditions gave rise to what is known as the dual CPP:

$$\text{minimize} \quad \{f(x) - \sum_{i=1}^{m} u_i[g_i(x)-b_i]\}, \tag{1.13}$$

subject to:

$$\frac{\partial}{\partial x_j}\{f(x) - \sum_{i=1}^{m} u_i[g_i(x)-b_i]\} = 0, \quad j=1,\ldots,n, \tag{1.14}$$

$$u_i \geq 0, \quad i\varepsilon I_1, \tag{1.15}$$

$$u_i \leq 0, \quad i\varepsilon I_2, \tag{1.16}$$

$$u_i \text{ unrestricted}, \quad i\varepsilon I_3. \tag{1.17}$$

The MPP given by relations 1.1 through 1.4 is referred to as the primal MPP. An important result of duality which will be used in chapter two is given below [45].

Theorem 1.1: If the following are true:

a) the point x' is feasible for the primal problem;

b) the point (x',u') is feasible for the dual problem;

c) the value of the primal objective function is equal to the value of the dual objective function;

then x' is an optimal solution to the primal problem.

Following the publication of the Kuhn-Tucker conditions, various algorithms were developed which solve the CPP. Most algorithms were for a particular type of CPP, such as the quadratic programming problem (QPP) in which the objective

function is the sum of a linear and a quadratic function and the constraint set is linear.

All algorithms for the CPP are of an iterative nature in that they choose a sequence of points which hopefully converge to x^*. Algorithms may be distinguished [4] by the different methods used to determine the following (given a current point x):

(1) in which direction, s, the new point lies;

(2) at what distance, d, from the current point the new point will be.

Summaries of solution techniques available for the CPP may be found in Beveridge and Schechter [4], Kunzi, Krelle, and Oetelli [27], Wilde and Beightler [42], or Zangwill [45]. It is appropriate to briefly discuss some of the different algorithms. One may note that several of the methods employ similar concepts (e.g., motion along a boundary), but use different mathematical techniques to achieve their ends.

Simplex method

The Simplex method of linear programming (1947) uses motions along the constraint surfaces, never leaving the feasible region [9]. The constraint surface is a convex polyhedron or simplex. If the dimension of the space is n, then the intersection of at least n hyperplanes defined by the constraint set forms a vertex. Vertices are connected by edges of dimension one which are formed by the inter-

section of at least n-1 hyperplanes. Hyperplanes are said to intersect when for some set of points, P, the constraints corresponding to these intersecting hyperplanes are binding, e.g.:

$$g_i(x) = b_i, \; x \epsilon P.$$

The Simplex method iteratively moves from one vertex to another along an edge such that not only is feasibility maintained, but also the value of the objective function never becomes less optimal. It may be shown that x* is some vertex. Thus, when the method reaches the vertex corresponding to x*, the procedure terminates.

Three basic methods are available for improving the efficiency of the Simplex method. The first is the use of more efficient computational techniques [9, 26, 34, 35, 43] such as the product form of the inverse. The second method is to exploit special problem structures [6, 18, 39, 41, 43, 44] as is done with the transportation model. Another example is the use of the decomposition principle in which several smaller problems are solved instead of one large one. The third method is to use some technique which is initially more efficient than the Simplex method. Kunzi [28, 29] does this by examining the inner products of the gradients of the $g_i(x)$ and $f(x)$. Carrol [7] proposed using Penalty Function methods (discussed later) to obtain a starting point. Most computer codes for the LPP incorporate

the first and second methods. For example, the Mathematical Programming System supplied by IBM for its 360 and 370 model computers uses the revised simplex method with the product form of the inverse. It also permits the use of the decomposition algorithm [30].

Multiplex method

Frisch's Multiplex method (1957) for linear and quadratic programming [27] is such that the general form for the direction, s, in which the new point lies is given by

$$S = q_o g_i(x) + \sum_{i \varepsilon I} q_i a_i^T$$

where q_o and the q_i, $i \varepsilon I$, are scalars and I is an index set determined by the algorithm. The vectors s, a_i, and x are n dimensional. The vector x is the current point, and

$$a_i = \begin{pmatrix} a_{i1} \\ \cdot \\ \cdot \\ \cdot \\ a_{in} \end{pmatrix}^T$$

is a vector used in forming one of the constraints

$$\sum_{j=1}^{n} a_{ij} x_j \sim b_i$$

where $\sim$ indicates $\leq$, $\geq$, or =. The vector g(x) is the current gradient of the objective function. The scalar coefficients are generally determined in part by the solution of a system of linear equations which has as its constant coefficients

the various inner products of the a_i amongst themselves and with $g(x)$. Also a matrix multiplication is sometimes necessary. When no constraints are tight, however, it is possible to take a pure gradient move. Generally, though, s is perpindicular to some boundary or along it. The step size, d, is chosen so that either $f(x + ds)$ is maximized or some constraint not previously tight becomes tight, whichever occurs first.

Gradient Projection method

The Gradient Projection method of Rosen (1960) for linear and quadratic programming [36] is similar in some respects to the Multiplex method. When no constraints are tight, s is equal to the gradient. When some constraints are tight, a projection matrix is formed which when multiplied by the gradient yields $\tilde{g}(x)$, the projected gradient, which does not leave the feasible region. Determination of the projection matrix involves matrix inversion, multiplications, and subtractions. The step size, d, is determined as in the Multiplex method. The Gradient Projection Method has been generalized so that it is applicable to any CPP [37].

Ricocheting Gradient method

Another algorithm for solving the CPP is Greenstadt's Ricocheting Gradient method (1966). In this method [13] if

no constraints are tight (e.g., x is an interior point), s is the gradient direction, and d is determined so as to approximately maximize f(x + ds). If the new point is either on a boundary or infeasible, several manipulations are made to move into the interior away from the boundary, after which another gradient move is taken. These manipulations include first finding a nearby interior point by a modificiation of Newton's method. Next another feasible point is found roughly equidistant from all constraints which are within a certain tolerance of being tight. This involves the solution of simultaneous equations. Then a quadratic programming problem is solved to determine the direction which will be taken into the interior of the feasible region. After an interpolation procedure, the new point is obtained from which another gradient move is taken. Thus, the method is called Ricocheting Gradient: One follows the gradient until a boundary is met, then ricochets back into the interior, and begins anew.

Penalty Function method

A completely different approach to the CPP was developed during the 1960's, the Penalty Function method. This method was first proposed by Courant in 1943, used by Carroll in 1960, proven by Fiacco and McCormick in 1964, and later extended by them, then generalized by Zangwill in 1967 [7, 8, 15, 16, 46]. The method does not explicitly recognize

constraints as do the methods previously cited, but rather incorporates them into the objective function, yielding an unconstrained function (the penalty function) to be maximized by any of the available techniques. A succession of these unconstrained problems is solved, generating a sequence of points which converge to x*. The unconstrained maximization techniques which have apparently been the most successful are second order gradient techniques [6, 17].

It is noted that this brief summary of methods is by no means complete, but has been included due to the similarity of these methods to research which will be presented later.

Parametric variation in linear programs

The most widely used form of convex programming is linear programming (LP). Beale[1] reports that LP "still covers most of the practical applications of mathematical programming, and that almost all practical applications of solving more general mathematical programming problems are based on the Simplex method of linear programming." The LPP may be stated mathematically as follows:

$$\text{maximize} \quad \sum_{j=1}^{n} c_j x_j \tag{1.18}$$

subject to:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i \in I_1, \tag{1.19}$$

---

[1] Beale, Mathematical Programming in Practice (New York, 1968), p. 1.

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i, \quad i \in I_2, \tag{1.20}$$

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i \in I_3. \tag{1.21}$$

The symbols a_{ij}, c_j, and b_i represent real numbers referred to as the problem parameters. For a given set of these parameters an x* may be determined, if the feasible region is not the null set. Having obtained x*, one may then perform a sensitivity analysis by repeatedly changing some set of the parameters from their current values to some set of new values. Beale [2] notes that many users of LP think that sensitivity analysis is far more important than the mere determination of x*. Dantzig [9] advances some reasons for this. They are:

(1) At the optimum a small parametric change may cause a big shift in x*. In the face of "certain inherent variabilities in the processes and materials not taken account of in the model,"[2] one may wish to move to some neighboring point and trade off optimality for stability.

(2) Parametric values may to some extent be controllable, in which case one would want to know what effect would result from this variation.

(3) Parametric values may only be estimates, "making

---

[2]Dantzig, Linear Programming and Extensions (Princeton, 1968), pp. 216-217.

it important to know for what ranges of values the solution is still optimum. If it turns out that the optimum solution is extremely sensitive to their values, it may be necessary to obtain better estimates."[3]

In addition to sensitivity analysis, there is another technique, parametric programming, which is used to examine the effects of a certain type of change in parametric values from their current values [2, 9, 24, 39, 41]. With this method one studies the manner in which the optimum varies for a one parameter family of alternative right hand sides (b_i's) or objective functions (c_j's). All changes are linear functions of the form

$$b_i = b_{io} + \theta b_{io}$$

or

$$c_j = c_{jo} + \theta c_{jo}$$

where θ is the parameter which is varied. Also methods exist [10, 32, 40] in which the set $\{a_{ij}\}_{i=1}^{i=m}$ may be systematically varied for some given "j" as a function of θ. For problems which may be accurately modeled by these techniques, such a method is of great utility.

Another method available for taking into account the variation of parameters is that of stochastic programming [1, 2, 19, 41]. Such techniques often require knowledge of

---

[3] Ibid.

the distributional form of parameters. Generally, these techniques give rise to equivalent nonlinear programs and sometimes non-convex problems. Various assumptions, however, may permit an equivalent linear programming formulation, oftentimes of vast size.

The solution of the equivalent problem will yield x*. For example, consider the case in which all functions are linear, but the c_j are stochastic. Here one would seek to maximize the expected value of $\sum_{j=1}^{n} c_j x_j$, $E\{\sum_{j=1}^{n} c_j x_j\}$, where E is the expected value operator. Thus one would maximize $\sum_{j=1}^{n} E(c_j)x_j$, subject to the constraint set. This is but an ordinary LPP. After an initial x* is obtained, one would then systematically vary a current set values for the parameter $E(c_j)$ and use sensitivity analysis to discover the range over which x* may vary in event of different realizations of the c_j. However, to exhaustively examine the effects of a change from any number of given c vectors may be rather tedious, not to mention the cases in which the b_i and a_{ij} are also stochastic. Nonetheless, it may be more important to know the range rather than some specific x*. Such may be the case for the planning phase of some activity. For example, suppose the x_j are production levels for different commodities. Knowledge of the optimal ranges of production levels for the various commodities would

enable one to determine an approximate plant size necessary to realize economic returns to scale.

An alternative approach to the problem of variation of parametric values has been suggested. Reasoning that one would often be interested in the range of possible values of x^* as a function of a given range of possible values of parameters, he proposed the formulation of the LPP in the interval number system. In this number system, an interval number is actually a closed interval of real numbers. If it is possible to formulate and solve interval linear programs, then the resultant x_j^* will themselves be interval numbers. Thus, the solution of an interval linear program would include a built-in sensitivity analysis.

The concept of formulating a LPP in the interval number system may best be illustrated by means of an example. Consider the following linear program:

maximize $\quad c_1x_1 + c_2x_2,$

subject to:

$$a_{11}x_1 + a_{12}x_2 \leq b_i,$$

$$a_{21}x_1 + a_{22}x_2 \leq b_2,$$

$$x_1, x_2 \geq 0 .$$

Suppose that it is only possible to specify the intervals

within which the various parameters lie, e.g.:

$$1 \leq c_1 \leq 2,$$

$$1 \leq c_2 \leq 2,$$

$$2 \leq a_{11} \leq 3,$$

$$1 \leq a_{12} \leq 3,$$

$$3 \leq b_i \leq 4,$$

$$1 \leq a_{21} \leq 2,$$

$$3 \leq a_{22} \leq 4,$$

$$4 \leq b_2 \leq 9.$$

Each parameter is free to assume any value within its specified interval. Further, suppose that one is interested in a range of values over which x* could vary corresponding to different parametric values.

A conventional approach to this problem is to choose some specific parametric values, determine x*, and then use sensitivity analysis to determine how x* varies. To this end consider Figure 1 on the following page. This figure depicts the linear programs corresponding to several possible realizations of various parametric values. For any of these realizations, as well as any others which may occur, standard linear programming techniques may be used to determine an optimum. The interval optimum would then consist of a vector of closed intervals of real numbers which contains all possible x*.

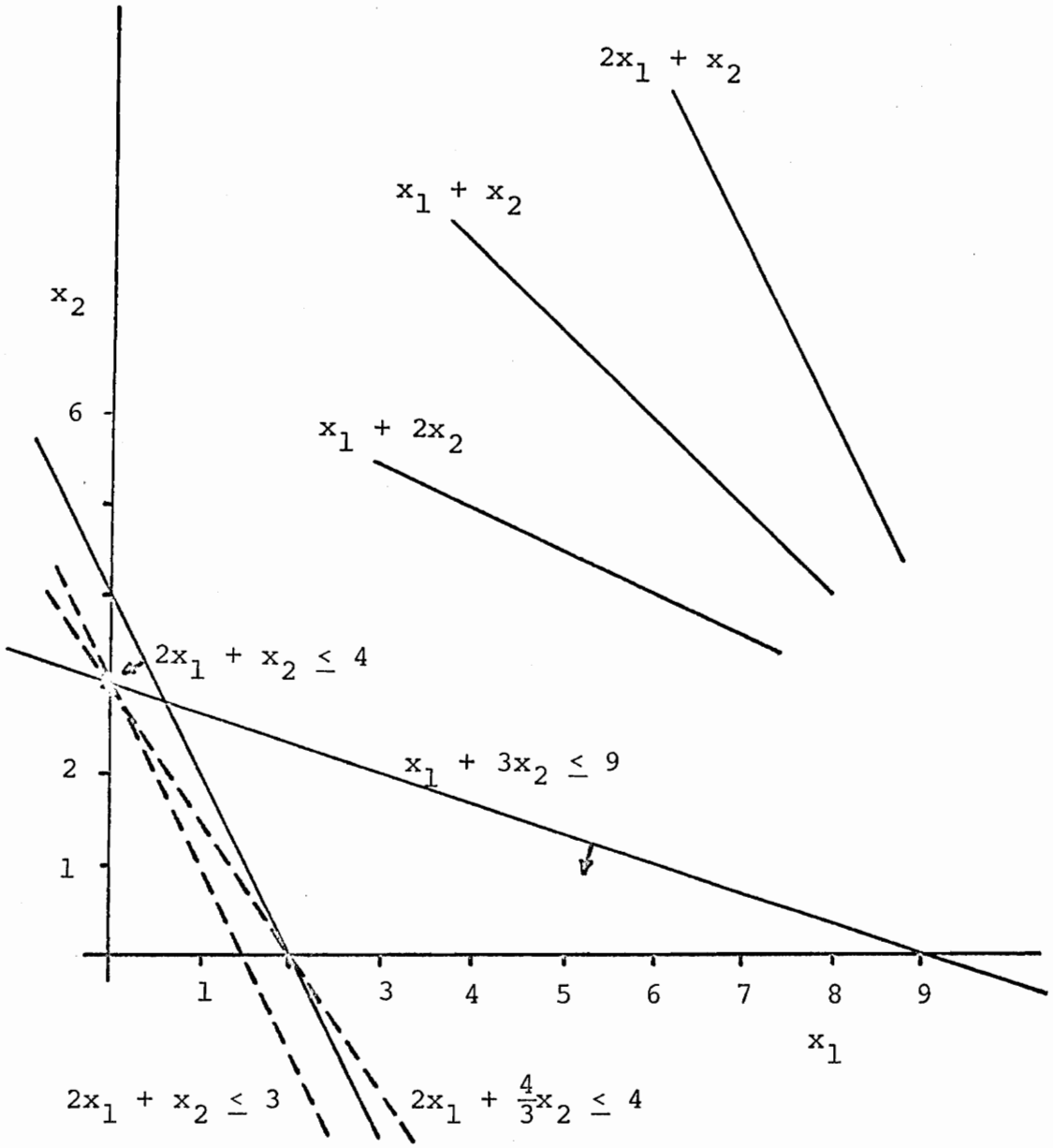

Figure 1. Sample feasible regions and objective functions.

By referring to this figure, one may see on a rather intuitive basis that the interval vector X*, which contains any x*, can be obtained from the solution of the following linear programs. (Other LPPs may also be used.) The minimal value of x_1^* and the maximal value of x_2^* is given by the solution of either:

maximize $x_1 + 2x_2$,

subject to:

$$2x_1 + x_2 \leq 3,$$

$$x_1 + 3x_2 \leq 9,$$

$$x_1, x_2 \geq 0;$$

or

maximize $x_1 + 2x_2$,

subject to:

$$2x_1 + (4/3)x_2 \leq 4,$$

$$x_1 + 3x_2 \leq 9,$$

$$x_1, x_2 \geq 0.$$

For either of these programs, the optimum is

$$\begin{pmatrix} x_1^* \\ x_2^* \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$$

The minimal value of x_2^* and the maximal value of x_1^* is given by the solution of

maximize $2x_1 + x_2$,

subject to:

$$2x_1 + x_2 \leq 4,$$

$$x_1 + 3x_2 \leq 9,$$

$$x_1, x_2 \geq 0.$$

The optimum of this program is

$$\begin{pmatrix} x_1^* \\ x_2^* \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.$$

Thus the interval vector which contains any x* is

$$X^* = \begin{pmatrix} [0,2] \\ [0,3] \end{pmatrix}.$$

For a LPP with hundreds of interval parameters, intuition would be of little or no help in guiding the choice of parametric changes for sensitivity analysis. Further, no decision rule is known which could be used to guide a sensitivity analysis in seeking out maximal and minimal x_j^*. Thus, it appears that sensitivity analysis may not be too practical when dealing with larger problems.

Problems of this type may, however, be directly and easily formulated in the interval number system. For example, the previous problem may be formulated as

maximize $[1,2]X_1 + [1,2]X_2$

subject to:

$$[2,3]X_1 + [1,3]X_2 \leq [3,4],$$

$$[1,2]X_1 + [3,4]X_2 \leq [4,9],$$

$$X_1, X_2 \geq [0,0].$$

Given such a formulation and using properties of the interval number system, a solution to the above interval problem may be obtained by solving four specific real valued LPPs. (The method by which this is done will be demonstrated in the next chapter.) It is noted that although the problem is formulated in the interval number system, the solution procedure for determining the interval optimum X* is performed in the real number system, thus permitting the use of existing techniques.

The next section contains a discussion of the basic fundamentals of interval arithmetic as they apply to the interval convex programming problem (ICPP), the interval number system analogue of the CPP. After that is the chapter summary.

INTERVAL ARITHMETIC

Organization

This section is devoted to a summary of various interval arithmetic topics. First a brief history of interval arithmetic is given. Next some general comments are made. Then the basic definitions and arithmetic operations are presented. Following this, some different properties and laws of interval numbers are explained. Then topics concerning interval matrices and vectors are discussed. After

this, interval functions are explained, and then interval equations, solution sets, and interval inequalities. The purpose of this summary is to facilitate the reading of the following chapter.

History

Interval arithmetic is a fairly recent development. The concept appears to have been first formulated in 1951 by Paul Dwyer [11, 38]. The original treatment was rather heuristic, and it was not until 1959 that formal proofs were presented by Moore [31, 38]. The development of interval arithmetic closely parallels that of the digital computer, as does that of convex programming. The tremendous advance in computing speed afforded by computers permitted the use of lengthy algorithms, but also gave rise to a concomitant propagation of error. Interval arithmetic was developed to cope with this error propagation by providing a built-in error analysis. For example, consider the problems associated with obtaining the solution, x, to a set of linear simultaneous equations. Due to the truncation and round-off error of a computer created by its finite (single and double) precision arithmetic, the $\hat{x}$ provided by the machine is usually but an estimate of the true solution. In order to determine how accurate an estimate this $\hat{x}$ is, one must perform an error analysis. When one uses interval arith-

metic, the solution, X[5], is such that each element of X is a closed interval

$$X = \begin{pmatrix} [x_{1L}, x_{1R}] \\ \vdots \\ [x_{nL}, x_{nR}] \end{pmatrix}$$

Further, X is such that for any element x_j of the true solution x, the following is true:

$$x_j \in [x_{jL}, x_{jR}]$$

Interval arithmetic was thus developed to cope with the error propagation of electronic computers. At the same time, it was the speed of these computers which encouraged the development of and practical implementation of convex programming algorithms. In spite of their relatively concurrent development, interval arithmetic and convex programming were not explicitly used together until 1965 when Oetelli [33] used linear programming to solve certain linear interval simultaneous equations.

General

Most generally, the interval number system forms an abelian semi-group under addition and under multiplication. An abelian semi-group is a set of elements which is closed under a binary operation and for which the associative law

---

[5]Hereafter capital letters will be used to denote interval numbers unless otherwise specified.

holds. The interval number system contains the field of real numbers, and it is from the properties of real numbers that the properties of interval numbers are derived. This follows from the fact that the basic definitions and operations are defined using real numbers, as will be seen in the following review taken largely from the works of Bennett, Hansen, Moore, and Shayer [3, 20, 31, 38].

Definitions and operations

Definition 1: Interval numbers: An interval number is an ordered pair of real numbers, $[c_L, c_R]$, with $c_L \leq c_R$, which defines the set, C, of real numbers:

$$C = \{ x : \ c_L \leq x \leq c_R \} .$$

C is referred to as an interval number.

Operations on interval numbers are defined in terms of their elements. Let the symbol * denote one of the operations of addition, subtraction, multiplication, or division; and let C and H be interval numbers where

$$C = [c_L, c_R]$$

and

$$H = [h_L, h_R] .$$

Then the following definition of the arithmetic operations may be given.

Definition 2: Interval arithmetic operations:

$$C*H = \{\ x*y :\ c_L \le x \le c_R\ ,\ \ h_L \le y \le h_R\ \}\ ,$$

except that C/H is not defined if zero is an element of H. This basic definition gives rise to the following results:

$$C+H = [c_L+h_L,\ c_R+h_R],$$

$$C\cdot H = [\min(c_Lh_L,\ c_Lh_R,\ c_Rh_L,\ c_Rh_R),$$

$$\max(c_Lh_L,\ c_Lh_R,\ c_Rh_L,\ c_Rh_R)]$$

$$C-H = [c_L,c_R] + [-h_R,\ -h_L]$$

Also

$$C/H = [c_L,c_R] \cdot [1/h_L,\ 1/h_R]$$

provided $0 \notin H$.

The symbols C·H and CH are equivalent.

Associated with an interval number is its width.

Definition 3: Width: The width of the interval number C, W(C), is

$$W(C) = c_R-c_L.$$

If W(C) is equal to 0, then c_R equals c_L and the interval number is said to be degenerate. The set of all degenerate interval numbers is isomorphic to the real number system.

Properties and rules

Shayer [38] shows that interval arithmetic is closed, associative, and commutative under both operations of addition and multiplication. However, the distributive law does not always hold. Additive and multiplicative

identities exist, and are [0,0] and [1,1] respectively. Except for degenerate interval numbers, multiplicative and additive inverses do not exist.

Using the above information, the reflexive, symmetric, and transitive laws can be shown to hold in the interval number system. Let C, E, and H be interval numbers. Then these laws are given below as:

Reflexive law:

$$C = C \ .$$

Symmetric law:

$$\text{If } C = E, \text{ then } E = C \ .$$

Transitive law:

$$\text{If } C = E, \text{ and } E = H, \text{ then } C = H \ .$$

These laws imply the cancellation law of addition:

Cancellation law of addition:

$$\text{If } C + E = C + F, \text{ then } E = F \ .$$

However, the cancellation law for multiplication does not hold.

Matrices and vectors

In the real number system, there are arrays of real numbers called vectors or matrices. Such arrays exist in the interval number system, only the elements are interval numbers rather than real numbers. Interval matrix addition and subtraction are defined as in the real number system: elements in corresponding positions are added or subtracted.

Analogously to the real number system, interval matrix multiplication is defined by the row into column operation. The inverse of an interval matrix is itself an interval matrix. The interval matrix inverse is defined in a manner that insures that it contains the inverse of any real matrix contained in the matrix to be inverted.

Interval functions

The concept of a function in the interval number system is analogous to that of the real number system: a unique mapping from a domain space into a range space wherein the spaces consist of interval numbers. The following definition of an interval function may be given.

Definition 4: Interval functions and real restrictions: Let $f(p,x)$ be a real function of the real vector x and the real parameter set p. Then the corresponding interval function $F(P,X)$ of the interval vector X and the interval parameter set P is defined [20] as:

$$F(P,X) = \{f(p,x): x \varepsilon X,\ p \varepsilon P\}$$

The function $f(p,x)$ is the real restriction of $F(P,X)$. For example, a linear interval function is of the form

$$G(P,X) = \sum_{j=1}^{n} P_j X_j.$$

The corresponding real restriction would then be

$$g(p,x) = \sum_{j=1}^{n} p_j x_j,$$

where $p \varepsilon P$ and $x \varepsilon X$.

Interval derivative

In light of the definition of an interval function, the interval partial derivative of $F(P,X)$ with respect to X_j, $\frac{\partial F(P,X)}{\partial X_j}$, is defined as

$$\frac{\partial F(P,X)}{\partial X_j} = \{ \frac{\partial f(p,x)}{\partial x_j} : x \varepsilon X, \ p \varepsilon P \}$$

Interval equations

An interval equation is of the form

$$G(P,X) = B ,$$

where P is the interval valued parameter set, X is an n dimensional vector variable, B is an interval number or expression, and $G(P,X)$ is an interval function. The solution set, S, of an interval equation or set of interval equations

$$G_i(P_i,X) = B_i , \ i=1,\ldots,m,$$

is that set of x's such that

$$g_i(p_i,x) = b_i , \ i=1,\ldots,m,$$

where $g_i(p_i,x)$ is the real restriction of $G_i(P_i,X)$ and p_i,x, and b_i are real numbers such that $p_i \varepsilon P_i$, $b_i \varepsilon B_i$, and $x \varepsilon X$. Thus S is a set of real numbers. The solution

$$X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} [x_{1L}, x_{1R}] \\ \vdots \\ [x_{nL}, x_{nR}] \end{pmatrix}$$

is such that

$$X_j = [\min x_j \varepsilon S, \max x_j \varepsilon S], \quad j=1,\ldots,n.$$

If the $G_i(P_i,X)$ are linear, the resultant equation set may be written as

$$AX = B$$

where A is an n by n interval matrix. If A is such that any real matrix R contained in A is not singular, then the equation set has the solution X which is contained in the interval vector $A^{-1}B$:

$$X \subset A^{-1}B.$$

However, matrix inversion is not the only technique by which linear interval simultaneous equations may be solved. Otelli [33] has devised a procedure by which it is possible to specify the solution set for each separate equation by two real valued inequalities. Under the condition that the interval widths of the A_{ij} and B_i be sufficiently small, the techniques of linear programming may be used on this inequality set to find the max x_j and the min x_j, for all j. The resulting solutions to the series of linear programs yield the interval solution X.

Interval inequalities

Shayer and Moore [31, 38] define the partial ordering of two interval numbers, C and E, as

$$C < E$$

if and only if every element of C is less than every element of E. This implies

$$c_R < e_L.$$

Using this definition, the relation

$$G(P,X) < B$$

would then imply that the solution set S includes only those x such that

$$g(p,x) < b_L,$$

for all $p \varepsilon P$.

SUMMARY

This chapter has provided a brief statement of the research objectives. This led to a review of the CPP, and then the problem of parametric variation was introduced. This problem of variation gave rise to an interval formulation of the CPP, and then a brief review of interval arithmetic was given. Now that the subject has been briefly reviewed, this section will elaborate a bit more on the research objectives and their interrelationships.

The general research objective is to formulate the CPP in the interval number system and to develop solution

procedures for these interval convex programming problems (ICPPs). The purpose of the research is to provide users of convex programming with a method for handling the problem of uncertainty in parametric values. The proposed method is substantially different from the standard techniques of stochastic programming and sensitivity analysis, in that the optimal answer is an interval vector rather than a point or series of points in Euclidean n-space. In essense, given the ranges (intervals) over which input parameters may vary, the solution to the ICPP is itself an interval vector.

Research on the ICPP led to the development of two solution algorithms. One method provides minimal width solutions, whereas the solutions of the other algorithm are not of minimal width. Both methods involve the solutions of 2n (where n is the dimension of the space) real valued programs, each program containing roughly double the number of variables and constraints as the original problem. The approximate technique is designed for a special category of ICPPs, known as interval separable convex programming problems (ISCPPs). Interval linear and quadratic programs are special types of ISCPPs. The approximate solution procedure available for ISCPPs is more efficient than the minimal width technique because the series of 2n programs which must be solved are not as non-linear as those which must be solved for the exact technique. In fact, when the approx-

imate technique is used to solve interval linear or quadratic programs, the resulting series of programs are linear or quadratic, respectively. This is important for it is anticipated that interval linear programs will be the most commonly occurring type of ICPP, just as linear programs are the most commonly occurring type of CPP.

Thus, any ICPP may be solved using existent CPP techniques. However, the resulting 2n CPPs are each larger than the original interval problem. This is the price which is paid for the additional information and flexibility of the interval formulation. In light of the increased problem size and the fact that interval linear programs can be solved using a series of real valued linear programs, it was decided to investigate LPPs to see if they may be more efficiently solved. As noted by Beale, the Simplex method and its extensions are the most practical way of solving the LPP, a problem of central importance in convex programming. Extensive research [4, 9, 26, 34, 35, 43] has been devoted to efficient computational techniques for use in the Simplex algorithm. Thus, it was decided to attempt to improve the efficiency with which LPPs may be solved by using heuristic algorithms designed to swiftly indicate a good starting point for the Simplex procedure. These methods are presented in Chapter Three. A fourth algorithm which is a new procedure for the solution of the LPP is also developed. These methods are conceptually similar to CPP solution

techniques described earlier in that moves are made either:

(1) along the gradient;

(2) along the constraint boundaries in such a manner that the value of the objective function does not decrease;

(3) away from constraints boundaries such that the value of the objective function does not decrease;

(4) on the basis of a penalty formulation.

An effort was made to avoid some of the computational intricacies of the methods described earlier in order to reduce computational time.

In order to test the heuristics, it was desirable to have several solved problems. One method for doing this was to randomly generate problems, and then use some known solution technique. However, instead of actually having to go through the computational expense of repeatedly using a known solution procedure, it became apparent that it would be useful to devise a method for generating problems with randomly chosen parameters, but with known solutions. Further, it was hoped that these generation techniques would bring about some additional insight into the CPP, and perhaps lead to more efficient solution techniques.

The research objectives may now be stated in rather specific terms. They are given below.

(1) Formulate the ICPP, making all definitions in a user oriented manner so that the technique may be readily applied.

(2) Develop solution procedures for the ICPP.

(3) Design and test the three above mentioned heuristic algorithms. Prove that the fourth algorithm is an exact solution procedure for the LPP.

(4) Develop methods of generating programs which have randomly chosen parameters, but known solutions.

CHAPTER TWO

INTERVAL CONVEX PROGRAMMING

ORGANIZATION

The ICPP is concerned with specifying the range of possible solutions of a CPP for which it is possible to give only interval estimates of the problem parameters (such as cost coefficients, stipulations, availabilities, and so forth). This chapter explicitly defines the ICPP and presents techniques for its solution. The first section contains several definitions, including the interval convex programming problem statement. The second section presents some possible approaches to solving the ICPP and introduces the general approach which is used. Then some preliminary theorems and definitions which are necessary to the solution processes are presented in the third section. The fourth section contains solution procedure details, and the last section is a brief summary.

DEFINITIONS AND PROBLEM STATEMENT

The original use of interval arithmetic as an error propagation analysis tool did not make it necessary to recognize the concept of identical interval numbers. For

example, it is true in interval arithmetic that

$$[-2,2] - [-2,2] = [-4,4].$$

Now suppose A is an interval number equal to [-2,2]. Then applying the definition of interval arithmetic operations (definition 2), it is seen that

$$A-A = \{x-y : a_L \leq x \leq a_R,\ a_L \leq y \leq a_R\}$$

and hence

$$A-A = [-4,4]$$

This consequence of the use of definition 2 is desirable when interval arithmetic is used for error propagation analysis. However, it is inappropriate for certain other areas of analysis, including interval convex programming. Hence, a new definition incorporating the work of Bennett [3] will now be introduced.

Definition 4: Identical and equal interval vectors: Two interval vectors, A and B, are said to be identical if and only if

a) $a_{jL} = b_{jL}$, $\forall j$,

b) $a_{jR} = b_{jR}$, $\forall j$,

c) $G(A,B,W) = \{g(a,b,w):\ a_j=b_j,\ a_{jL} \leq a_j \leq a_{jR},\ \forall j;\ w \in W\}$

where G(A,B,W) is an interval function, g(a,b,w) is its real extension, and W is an arbitrary interval expression. If only conditions a) and b) are met, then the vectors are said to be equal.

This definition may be illustrated by considering two

identical interval numbers, A and B, and two equal interval numbers, C and D. It may be seen that

$$A-B = \{a-b : a=b,\ a_L \leq a \leq a_R\}$$

or

$$A-B = [0,0].$$

However,

$$C-D = \{c-d : c_L \leq c \leq c_R,\ d_L \leq d \leq d_R\}$$

or

$$C-D = [c_L-d_R,\ c_R-d_L].$$

Throughout this text, if the same alphabetic symbol is used for two interval vectors, then they are presumed identical. To denote that A and B are identical the notation

$$A \equiv B$$

will be used. When alphabetic symbols are not used, it is never assumed that two interval numbers are identical. As an example suppose

$$A = [-2,2].$$

Then the notation used herein implies that

$$A-A = [0,0],$$

but

$$[-2,2] - [-2,2] = [-4,4].$$

The stipulation that two interval numbers are equal in no way implies that they are identical.

The use of identical interval numbers, although inappropriate for error computations, is quite necessary for the

analysis of ICPPs. The existence of identical interval numbers makes possible the expansion of certain previously existing properties of the interval number system. These new results will be noted in the section which contains various preliminary theorems and definitions.

The presentation of the remaining definitions of this section is facilitated by recalling the example presented on pages 12 through 15 in Chapter One. In this example it was desired to determine the range of optimal values for a linear program having only range or interval estimates of its parameters. The interval formulation of the problem was given as

$$\begin{aligned} \text{maximize} \quad & [1,2]X_1 + [1,2]X_2, \\ \text{Subject to:} \quad & [2,3]X_1 + [1,3]X_2 \leq [3,4], \\ & [1,2]X_1 + [3,4]X_2 \leq [4,9], \\ & X_1, X_2 \geq [0,0]. \end{aligned}$$

The last constraint merely insures non-negativity. This example involves the use of interval inequalities. Recall that the previously given definition of an interval inequality implies that an relation

$$G(P,X) < B$$

has a solution set S which includes only those values of x such that

$$g(p,x) < b_L$$

for all $p \varepsilon P$. Thus, over the feasible domain

$$X_1, X_2 \geq [0,0]$$

the relation

$$[2,3]X_1 + [1,3]X_2 < [3,4]$$

has the solution set defined by

$$\{x: \ 3x_1 + 3x_2 < 3, \ x \geq 0\}$$

This solution set is obtained intuitively by using the most restrictive parametric realizations possible.

At this point is is appropriate to recall the context within which it is intended that ICPPs will be used. It is assumed that an individual can specify the functional form of his problem, but can obtain only range or interval estimates of the problem parameter set. It is thus assumed that by the solution set of a relation such as

$$[2,3]X_1 + [1,3]X_2 < [3,4]$$

is meant the union of all solution sets which may result from different realizations of the parameter set. Examples of two possible realizations are shown below:

$$2.6x_1 + 1.7x_2 < 3.8,$$

$$2.1x_1 + 2.7x_2 < 3.4.$$

The following new definition may now be given.

Definition 5: Solution set of an interval inequality: The solution set, S, of the interval inequality

$$G(P,X) < H(P,X)$$

is defined to be the set of all x which satisfy

$$g(p,x) < h(p,x)$$

for any $p \varepsilon P$, where $g(p,x)$ and $h(p,x)$ are the real restrictions of $G(P,X)$ and $H(P,X)$, respectively. The solution set of the relation

$$G(P,X) \leq H(P,X)$$

is the union of the solution sets of

$$G(P,X) < H(P,X)$$

$$G(P,X) = H(P,X).$$

The relations "greater than" (>) and "greater than or equal to" ($\geq$) are similarly defined. In fact, the above definition may be used with the symbol ">" substituted for the symbol "<". The following definition is a direct extension of the previous one, and of the preceeding statement concerning greater than (or equal to) inequalities.

Definition 6: Solution set of simultaneous interval relations: Let the symbol "$\sim_i$" designate any of the symbols "=", "<", "$\leq$", ">", or "$\geq$". The solution set, S, of the simultaneous interval relations

$$G_i(P,X) \sim_i H_i(P,X) \quad , \ i=1,\ldots,m,$$

is

$$S = \{x : g_i(p,x) \sim_i h_i(p,x), \ i=1,\ldots,m; \ p \varepsilon P\}$$

where $g_i(p,x)$ and $h_i(p,x)$ are the real restrictions of $G_i(P,X)$ and $H_i(P,X)$. Thus S is a set of real vectors. The minimal width solution

$$X = \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} [x_{1L}, x_{1R}] \\ \\ [x_{nL}, x_{nR}] \end{pmatrix}$$

is such that

$$[x_{jL}, x_{jR}] = [\inf x_j \varepsilon S, \sup x_j \varepsilon S].$$

A solution which contains the minimal width solution is simply referred to as a solution.

The definitions of interval convex functions, interval optima and the ICPP statement may now be given.

Definition 7: Interval convex function: The interval function F(P,X) is interval convex (concave) over some region, D, if and only if all real restrictions of F(P,X) are convex (concave) for any arbitrary, but fixed, $p \varepsilon P$ over the region D.

As an example of the preceding definition note that the interval function

$$[2,3]X_1 + [1,3]X_2$$

defined over the region, D,

$$X \geq [0,0]$$

is interval convex over D since any realization of the parameter set would result in a function of linear form which is convex. As before, two possible realizations are shown below:

$$2.6x_1 + 1.7x_2,$$

$$2.1x_1 + 2.7x_2.$$

Definition 8: Interval optimal set: The interval optimal set, Q, of an interval function F(P,X) constrained by the relations

$$G_i(P,X) \leq H_i(P,X) \quad , i\varepsilon I_1,$$

$$G_i(P,X) \geq H_i(P,X) \quad , i\varepsilon I_2,$$

$$G_i(P,X) = H_i(P,X) \quad , i\varepsilon I_3,$$

is defined to be the set of all real vectors x which optimize the MPPs

maximize $f(p,x)$

subject to:

$$g_i(p,x) \leq h_i(p,x) \quad , i\varepsilon I_1,$$

$$g_i(p,x) \geq h_i(p,x) \quad , i\varepsilon I_2,$$

$$g_i(p,x) = h_i(p,x) \quad , i\varepsilon I_3,$$

for any $p\varepsilon P$ where $f(p,x)$, $g_i(p,x)$, and $h_i(p,x)$ are real restrictions of $F(P,X)$, $G_i(P,X)$, and $H_i(P,X)$, respectively. The minimal width interval optimum

$$X^* = \begin{pmatrix} X_1^* \\ \cdot \\ \cdot \\ \cdot \\ X_n^* \end{pmatrix} = \begin{pmatrix} [x_{1L}^*, x_{1R}^*] \\ \\ \\ [x_{nL}^*, x_{nR}^*] \end{pmatrix}$$

is such that

$$X_j^* = [\inf x_j \varepsilon Q, \sup x_j \varepsilon Q]$$

Any interval vector which contains the minimal width interval optimum is simply referred to as an interval optimum.

Interval convex programming problem statement: The ICPP is to determine the interval optimum of an interval concave function subject to an interval constraint set. This interval constraint set is such that the feasible region corresponding to any parametric realization is convex. This implies that for the following to be an ICPP

maximize $F(P,X)$,

subject to:

$$G_i(P,X) \leq H_i(P,X) \quad , i\varepsilon I_1,$$

$$G_i(P,X) \geq H_i(P,X) \quad , i\varepsilon I_2,$$

$$G_i(P,X) = H_i(P,X) \quad , i\varepsilon I_3,$$

then the following functions are interval convex

i) $G_i(P,X)$, $i\varepsilon I_1$,

ii) $H_i(P,X)$, $i\varepsilon I_2$,

and the following functions are interval concave

i) $G_i(P,X)$, $i\varepsilon I_2$,

ii) $H_i(P,X)$, $i\varepsilon I_1$,

iii) $F(P,H)$,

and the following functions are interval linear

i) $G_i(P,X)$, $H_i(P,X)$, $i\varepsilon I_3$.

Thus, the ICPP has been defined. Before examining possible approaches to the ICPP, it is appropriate to give an example showing how interval convex programming might be

used in a practical context. Consider an individual engaged in planning activities, for example, planning future production facilities. There exist various linear programming models concerned with production planning. However, it may be quite difficult or impossible to specify demand or various other parameters exactly. In such situations one may use interval linear programming to determine the approximate size of the production facility, approximate storage space necessary, etc. The exact policy would be determined by management decisions in which considerations other than the results of a mathematical program are relevant. Thus, ICPP may be more useful as a planning or decision aid, rather than as an explicit decision model.

POSSIBLE APPROACHES

The Simplex logic

A mathematical statement of the ICPP has been given, so now some possible approaches to solving the problem are examined. The most natural approach is to attempt to directly extend the CPP solution techniques of the real number system. This approach may best be illustrated by using a simple linear programming example. Consider the following problem:

Maximize $[-1,1]X_1 + [-1,1]X_2$,

subject to:

$$[1,1]X_1 \leq [1,2],$$

$$[1,1]X_2 \leq [1,2],$$

$$X_1, X_2 \geq [0,0].$$

The optimal region of this program may be determined intuitively and is shown in Figure 2 on the following page. The shaded region and the heavy dark lines indicate the optimal region.

Now suppose the logic of the Simplex solution procedure is used. The standard form of the problem is

maximize $C_1X_1 + C_2X_2$

subject to:

$$A_{11}X_1 + Y_1 = B_1$$

$$A_{21}X_2 + Y_2 = B_2$$

where all variables are restricted to be non-negative. The various tableaux are shown in Table 1 on the next page. Pivot elements are underlined, and basis vectors are indicated by the use of the degenerate interval number "1". Consider the first tableau. Since there exist $c_j \varepsilon C_j$ such that all realizations of $c_j - z_j$ could be negative (-1, for example) for non-basis variables, then this solution is a possible optimum. Thus it is known that

$$x = \begin{pmatrix} x_1 \\ \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ \\ 0 \end{pmatrix}$$

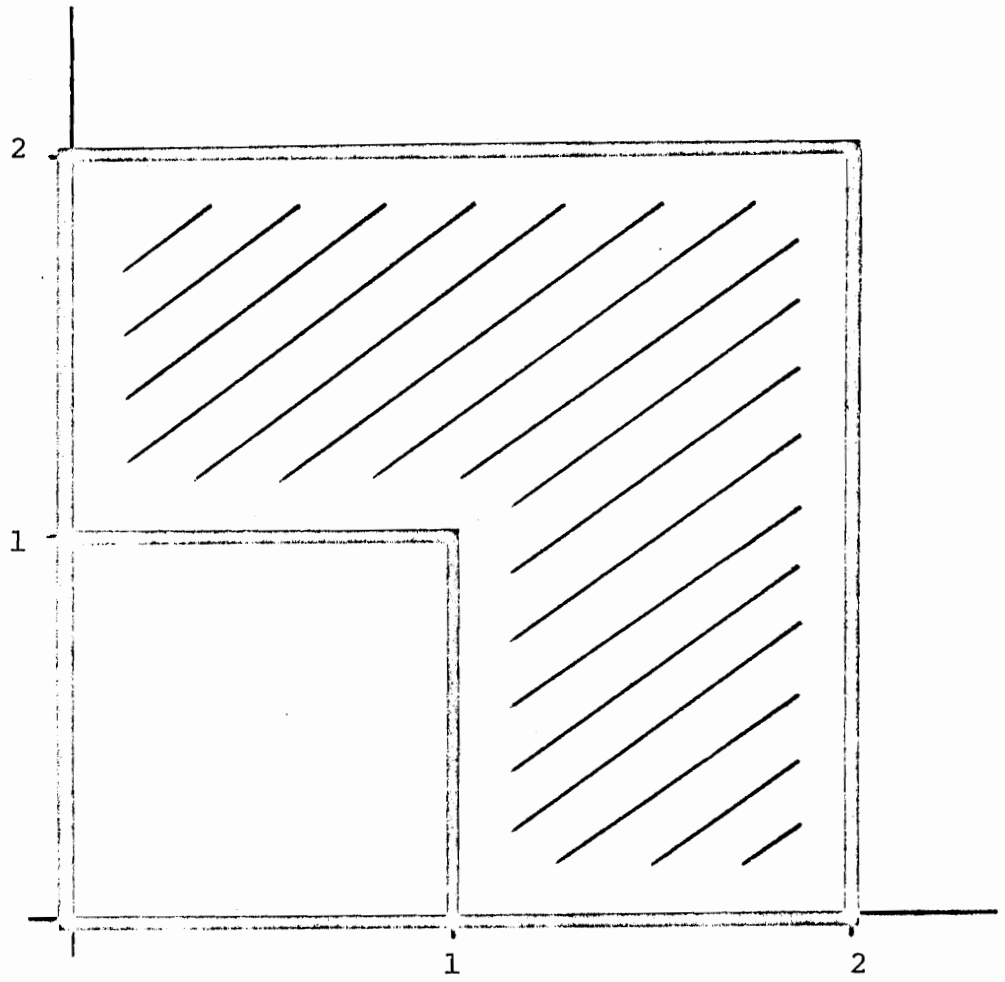

Figure 2. Example optimal region.

Table 1. Simplex tableaux

	X_1	X_2	Y_1	Y_2	B	C_B	θ
C_j	[-1,1]	[-1,1]	[0,0]	[0,0]			
I	[1,1]		1		[1,2]	[0,0]	[1,2]
		[1,1]		1	[1,2]	[0,0]	-
Z_j	[0,0]	[0,0]	-	-			
C_j-Z_j	[-1,1]	[-1,1]	-	-			
II	1		[1,1]		[1,2]	[-1,1]	-
		[1,1]		1	[1,2]	[0,0]	[1,2]
Z_j	-	[0,0]	[-1,1]	-			
C_j-Z_j	-	[-1,1]	[-1,1]	-			
III	1		[1,1]		[1,2]	[-1,1]	[1,2]
		1		[1,1]	[1,2]	[-1,1]	-
Z_j	-	-	[-1,1]	[-1,1]			
C_j-Z_j	-	-	[-1,1]	[-1,1]			
IV	[1,1]		1		[1,2]	[0,0]	-
		1		[1,1]	[1,2]	[-1,1]	[1,2]
Z_j	[0,0]	-	-	[-1,1]			
C_j-Z_j	[-1,1]	-	-	[-1,1]			

is a point in the optimal region. However, note that there exist $c_j \varepsilon C_j$ such that $c_j - z_j$ may be positive for a nonbasis variable; for example, it may occur that

$$c_1 - z_1 = 1.$$

Hence, there could be other optima, corresponding to different realizations of the c_j. Thus, let X_1 enter the basis and Y_1 leave. Due to the structure of the problem, the basis inverse is the identity matrix. Thus, the second tableau is identical to the first one, as are all subsequent tableau. The second tableau indicates that there exist $c_j \varepsilon C_j$ such that realizations of the $c_j - z_j$ could be negative (-2 or -1, for example) for non-basis variables. Thus, this solution is a possible optimum and the points specified by

$$1 \leq x_1 \leq 2 \quad ,$$

$$0 \leq x_2 \leq 0$$

are elements of the optimal region. However, as before, there exist realizations such that the value of the objective function may be increased. Let X_2 enter the basis and Y_2 leave. The third tableau shows that the points specified by

$$1 \leq x_1 \leq 2 \quad ,$$

$$1 \leq x_2 \leq 2$$

are elements of the optimal region. Again there exist realizations for which this solution may not be optimal,

so let Y_1 enter the basis and X_1 leave. The fourth tableau reveals that points such that

$$0 \leq x_1 \leq 0 \quad ,$$

$$1 \leq x_2 \leq 2$$

are elements of the optimal region. At this point all possible new bases have been examined. The four tableaux may be used to trace out the optimal region in Figure 1 (realizing that degeneracy was possible) by proceding in a counter-clockwise manner.

Now the same problem, phrased differently, will be considered. However, due to the method of matrix inversion to be used, the answers will differ from those previously established. Let the problem statement be

maximize $\quad [-1,1]X_1 + [-1,1]X_2 \ ,$

subject to:

$$[1/2,1]X_1 \leq [1,1]$$

$$[1/2,1]X_2 \leq [1,1]$$

$$X_1, \ X_2 \geq [0,0]$$

The solutions corresponding to the four bases examined earlier will be shown again, in the same order, next to the new solutions. These results are shown in Table 2 on the following page.

Table 2. Simplex solutions

Tableau	New solution	Previous solution
I	$1 \leq y_1 \leq 2,\ 1 \leq y_2 \leq 2$	$1 \leq y_1 \leq 2,\ 1 \leq y_2 \leq 2$
II	$1 \leq x_1 \leq 2,\ 1/2 \leq y_2 \leq 2$	$1 \leq x_1 \leq 2,\ 1 \leq y_2 \leq 2$
III	$1/2 \leq x_1 \leq 4,\ 1/2 \leq x_2 \leq 4$	$1 \leq x_1 \leq 2,\ 1 \leq x_2 \leq 2$
IV	$1/2 \leq y_1 \leq 2,\ 1 \leq x_2 \leq 2$	$1 \leq y_1 \leq 2,\ 1 \leq x_2 \leq 2$

The inversion method used is illustrated by

$$\begin{pmatrix} [1/2,1] & [0,0] \\ [0,0] & [1/2,1] \end{pmatrix}^{-1} = \frac{[1,1]}{[1/2,1][1/2,1] - [0,0][0,0]} \begin{pmatrix} [1/2,1] & -[0,0] \\ -[0,0] & [1/2,1] \end{pmatrix}$$

The computations were performed without treating identical interval numbers as such, for this is what must usually be done when a computer is used. Thus even if all possible optima could be determined using the Simplex logic, the answers would not, in general, be of minimal width. Further, each iteration may involve considerable computational effort. Also the danger exists of the occurrence of an interval basis for which no inverse exists, such as

$$\begin{pmatrix} [1,2] & [1,2] \\ [1,2] & [1,2] \end{pmatrix}$$

at which point the method fails. An interval basis such as this could occur if the following realizations (both of which have inverses) are possible:

$$\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, \qquad \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix},$$

Additionally, since interval inverses are not usually of minimal width, the solution value of the entering variable determined by the feasibility criterion and the solution value actually obtained may not be the same. This could lead to problems in maintaining feasibility. In light of these difficulties, it was decided to examine the use of existing techniques other than the Simplex method.

Other techniques

The difficulty in obtaining interval inverses and in determining step size also ruled out methods such as those of Rosen and Frisch. Also, penalty function techniques appear unpromising since these methods contain terms such as

$$[1,1]/[G_i(P_1X) - B_i] \ .$$

Whenever a B_i is non-degenerate, there are values of X which are elements of an interval constraint set such that the interval valued denominator contains the real number zero, thus resulting in solutions of infinite width. Considerations of this type indicated that instead of modifying some existing technique, it would be more appropriate to develop a completely different approach.

One possible approach is to specify a real valued constraint set whose parameters are elements of the original interval parameter set. This real valued constraint set would be such that its solution set contains all solution sets corresponding to possible realizations of the interval parameter set. It might be suspected that solutions obtained using this set may define the maximal or minimal optimal values. The fallacy of this approach may be illustrated by the following example:

maximize $[3,3]X_1 + [1,1]X_2$,

subject to:

$$[1,1]X_1 + [1,1]X_2 \leq [4,4],$$

$$[1,1]X_2 \leq [1,3.5],$$

$$[1,1]X_1 - [1,1]X_2 \geq [0,0],$$

$$X_1, X_2 \leq [0,0].$$

The maximal and minimal feasible regions are shown in Figure 3 on the following page. The shaded maximal region includes the minimal region which has single shading. The interval optimal region is shown by the heavy dark lines, and the minimal width optimal solution is

$$X^* = \begin{pmatrix} X_1^* \\ X_2^* \end{pmatrix} = \begin{pmatrix} [1,3.5] \\ [.5,2] \end{pmatrix} .$$

Note that the maximal value of x_2^* does not occur at any of the extreme realizations of the parameter set.

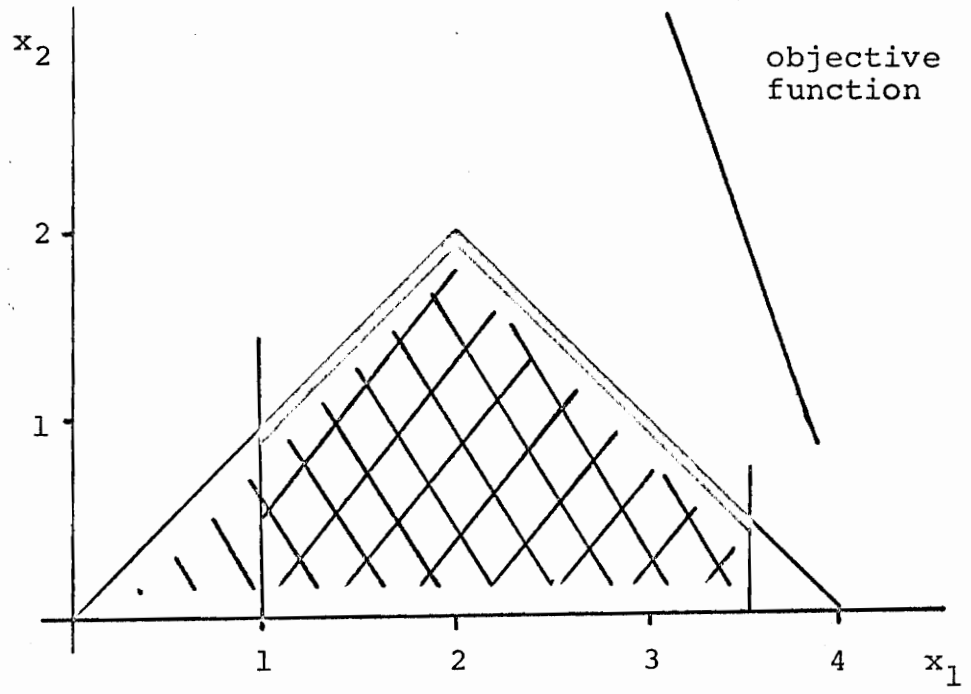

Figure 3. Maximal and minimal feasible regions.

The approach used

Although the above idea was untenable, it suggested a workable method. In this method the solution set is specified by real valued equations and inequalities. The maximal and minimal optimal solutions are determined using duality theory. The optimal region may be specified by mathematical relations, and then the maximal and minimal points of this region are determined using conventional techniques.

The next section develops methods for specifying interval solution sets by real valued relations. Also presented are some new topics in interval arithmetic which will be subsequently used. The development of these various results greatly facilitates the presentation of the solution algorithms.

PRELIMINARY DEFINITIONS AND THEOREMS

This section contains material necessary to the development of the solution procedures given in the following section. The first topic examined in this section are certain consequences of the recognition of identical interval numbers. This development consists of four theorems and one definition. Then theorem 2.5 provides a method for specifying a real valued constraint set which has the same

solution set as the corresponding interval constraint set. Next a special type of interval function, the interval separable function, is defined and an example is given. Then the interval separable convex programming problem (ISCPP) is defined. Following this are two theorems concerning interval separable constraint sets and their corresponding real valued constraint sets. Two examples are then given to illustrate the two theorems. Finally, four theorems are presented which further explore the properties of interval separable functions. All of the results of this section will be used in the presentation of the ICPP solution techniques in the following section. Some consequences of the definition of identical interval numbers will now be examined.

Identical interval numbers and functions

The recognition of identical interval numbers does not require the modification of any previously given definition; that is to say that it is not inconsistent, but only introduces a completely new term. The only previously given properties of the interval number system affected are the prior lack of additive and multiplicative inverses and of a distributive law. The following three theorems deal with these properties.

Theorem 2.1: Let A, B, and C be interval numbers. Then

$$A - A \equiv [0,0].$$

Further if

$$A \equiv C$$

and

$$B \equiv C \quad ,$$

then

$$A \equiv B \quad .$$

Proof: The definition of identical interval numbers may be used to obtain

$$A - A = \{x-y : x=y,\ a_L \leq x \leq a_R\}$$

and hence

$$A - A = [0,0].$$

Also let $G[(A-A),\ [0,0],\ W]$ be an arbitrary interval function. Then clearly

$$G\{(A-A),[0,0],W\} = \{g[(a-a),0,w]:\ a-a=0,\ 0\leq 0\leq 0,\ w\varepsilon W\},$$

thus fulfilling the third condition of definition 4, so that

$$A-A \equiv [0,0].$$

Now transitivity of equal interval numbers has already been established [31]. To prove that the third condition of definition 4 is satisfied note that

$$G(C,A,W) = \{g(c,a,w):\ c=a,\ c_L\leq c\leq c_R,\ w\varepsilon W\}$$

and

$$G(B,C,W) = \{g(b,c,w):\ b=c,\ b_L\leq b\leq b_R,\ w\varepsilon W\}.$$

If the two above equations are true, then it follows that

$$G(B,A,W) = \{g(b,a,w):\ b=a,\ b_L\leq b\leq b_R,\ w\varepsilon W\},$$

thus proving the theorem.

Theorem 2.2: Let A, B, and C be interval numbers.

If

$$A \equiv B$$

then

$$C+A \equiv C+B$$

and

$$C \cdot B \equiv C \cdot B$$

Proof: Since A and B are identical, then it is true that

$$a_L = b_L,$$

$$a_R = b_R,$$

$$G(A,B,Z) = \{g(a,b,z): \ a=b, \ a_L \leq a \leq a_R, \ z \epsilon Z\}.$$

To show equality, note that

$$C+A = [c_L+a_L, \ c_R+a_R]$$

and

$$C+B = [c_L+b_L, \ c_R+b_R],$$

so C+A and C+B are such that their end points are equal. Similarly it follows that the end points of C·A and C·B are equal. Now let

$$G(A+C,B+C,Z) = \{g(a+c,b+c,z):$$
$$a=b, (a+c) \epsilon (A+C), (b+c) \epsilon (B+C), \ z \epsilon Z\}$$

so that

$$G(A+C,B+C,Z) + \{g(a+c,b+c,z): \ a=b,$$
$$a_L+c_L \leq a+c \leq a_R+c_R, \ b_L+c_L \leq b+c \leq b_R+c_R, \ z \epsilon Z\}$$

where g is the real extension of the arbitrary interval function G. This may be rewritten as follows

$$G(A+C,B+C,Z) = \{g(a+c,b+c,z):$$
$$a+c=b+c,\ a_L+c_L \leq a+c \leq a_R+c_R,\ z\varepsilon Z\},$$

so that (A+C) and (B+C) are identical interval numbers. It follows in a similar manner that (A·C) and (B·C) are identical; however, it is notationally more cumbersome to do so due to the method of specifying end points for interval multiplication.

Theorem 2.3: Let A, B, and C be interval numbers. Then

$$(B+C)A \equiv BA+CA.$$

Proof: The definition of interval arithmetic implies that

$$(B+C)A = \{(x+y)w:\ a_L \leq w \leq a_R,\ b_L \leq x \leq b_R,\ c_L \leq y \leq c_R\}.$$

Using the definition of identical interval operations,

$$BA+CA = \{xw+yz:\ w=z,\ a_L \leq w \leq a_R,\ b_L \leq x \leq b_R,\ c_L \leq y \leq c_R\}.$$

Thus

$$BA+CA = \{xw+yw:\ a_L \leq w \leq a_R,\ b_L \leq x \leq b_R,\ c_L \leq y \leq c_R\}$$

or

$$BA+CA = \{(x+y)w:\ a_L \leq w \leq a_R,\ b_L \leq x \leq b_R,\ c_L \leq y \leq c_R\}.$$

Let

$$(B+C)A = [r_L, r_R]$$

and

$$BA+CA = [s_L, s_R].$$

Then since both r_L and s_L are equal to

$$\inf \{(x+y)w: \ a_L \leq w \leq a_R, \ b_L \leq x \leq b_R, \ c_L \leq y \leq c_R\}$$

and similarly r_R and s_R are equal, then conditions a) and b) of definition 4 are met, so that

$$(B+C)A = BA+CA.$$

Now let $G[(B+C)A,BA+CA,Z]$ be an interval function. Then

$$G[(B+C)A,BA+CA,Z] = \{g[(b+c)a,ba+ca,Z]:$$

$$r_L \leq (b+c)a \leq r_R, \ s_L \leq ba+bc \leq s_R, z \in Z\}.$$

And recalling the equality of the end points, it follows that

$$G[(B+C)A,BA+BC,Z] = \{g[(b+c)a,ba+bc,z]:$$

$$(b+c)a = ba+ca, \ r_L \leq (b+c)a \leq r_R, z \in Z\}.$$

This satisfies condition c) of definition 4, thus proving the theorem.

The above three theorems dealt with identical interval numbers. The next definition and theorem deal with a natural extension of the above topic, namely identical interval functions.

Definition 9: Identical interval functions: Two interval functions, $G(P,X)$ and $H(P,X)$ are identical over some region Q if and only if

$$g(p,x) = h(p,x)$$

for all $p \in P$ and $x \in Q$ where $g(p,x)$ and $h(p,x)$ are the respective real extensions. That two interval functions are identical will be indicated, as before, either by the use of the same alphabetic symbols or by the use of the symbol "≡".

Note that if the values of $F(P,X)$ and $G(P,X)$ can be shown to be identical interval numbers for any particular P and X, then the stipulation of condition three in definition 4 which requires that

$$g(p,x) = h(p,x)$$

indicates that $G(P,X)$ and $H(P,X)$ are identical interval functions. This provides an alternative means of identifying identical interval functions.

A property of identical interval functions which will be used in the presentation of solution techniques is given below.

Theorem 2.4: Let S be the solution set of the following interval relations

$$W_i[G_1(P,X),\ldots,G_q(P,X)] \sim_i V_i[H_1(P,X),\ldots,H_\mu(P,X)] \ , \ i\varepsilon I,$$

where $\sim_i$ indicates either an equality or an inequality. Let $R_j(P,X)$ and $S_K(P,X)$ be interval functions which are identical to the $G_j(P,X)$ and $H_K(P,X)$, over the region S. Let D be the solution set of

$$W_i[R_1(P,X),\ldots,R_q(P,X)] \sim_i V_i[S_1(P,X),\ldots,S_\mu(P,X)], \ i\varepsilon I,$$

then

$$S = D.$$

Proof: By the definition of the solution set of interval relations it is true that

$$S = \{x: \; w_i[g_1(p,x),\ldots,g_q(p,x)] \sim_i$$

$$v_i[h_1(p,x),\ldots,h_\mu(p,x], \; p\varepsilon P, \; i\varepsilon I\}$$

where w_i, v_i, g_j, and h_k are the appropriate real extensions. And since for all $x\varepsilon S$

$$g_j(p,x) = r_j(p,x),$$

$$h_k(p,x) = s_k(p,x),$$

where the r_j and s_k are the real extensions of the R_j and S_k, by the definition of identical interval functions, then it follows that

$$S = \{x: \; w_i[r_1(p,x),\ldots,r_q(p,x)] \sim_i$$

$$v_i[s_1(p,x),\ldots,s_k(p,x)], \; p\varepsilon P, \; i\varepsilon I\}.$$

However, this set may be seen to be D, thus proving the theorem.

The preceding development of identical interval numbers and functions extends the interval number system in a manner which is consistent with earlier results and which is necessary for the analysis of the interval convex programming problem. Two numerical examples contrasting the use of identical and equal interval numbers are given following theorem 2.7. These examples involve the relationship between an interval constraint set and its corresponding real constraint set. The first step in the development of these corresponding constraint sets is given by theorem 2.5 below.

A corresponding real constraint set

Theorem 2.5: Let S be the solution set of the relations

$$F_i(P,X) \leq H_i(P,X) \quad , i\varepsilon I_1, \qquad 2.1$$

$$F_i(P,X) \geq H_i(P,X) \quad , i\varepsilon I_2, \qquad 2.2$$

$$F_i(P,X) = F_i(P,X) \quad , i\varepsilon I_3, \qquad 2.3$$

Let D be the solution set of the relations

$$f_i(p,x) \leq h_i(p,x) \quad , i\varepsilon I_1, \qquad 2.4$$

$$f_i(p,x) \geq h_i(p,x) \quad , i\varepsilon I_2, \qquad 2.5$$

$$f_i(p,x) = h_i(p,x) \quad , i\varepsilon I_3, \qquad 2.6$$

$$p_{kL} \leq p_k \leq p_{kR} \quad , \forall k. \qquad 2.7$$

Then it follows that D and S are equal sets.

Proof: Restriction 2.7 merely specifies that p be an element of P. Thus the theorem is but a restatement of definition 6.

Theorem 2.5 permits the replacement of an interval constraint set by a real valued constraint set in which the parameters are treated as bounded variables. Both constraint sets specify the same solution sets. However, the functions of the real valued constraint set are of a higher degree than those of the interval valued constraint set. For example, if the interval constraint set consists of interval linear functions with some parameters having width greater than zero, then the corresponding

real valued constraint set will contain quadratic terms such as $p_{ij}x_j$. But if the interval constraint set consists of interval separable functions (defined below), then it is possible to specify real valued constraint sets which are of the same degree.

Interval separable functions

Definition 10: Interval separable function: The interval function F(P,X) is interval separable over some set of real vectors, D, if and only if there exist real valued parameter sets p' and p" such that

$$f(p',x) \leq f(p,x) \leq f(p'',x)$$

for all $x \varepsilon D$ and for all $p \varepsilon P$, where f(p,x) is the real restriction of F(P,X). Further, the function f(p,x) must be continuous when either x or p are permitted to vary for all $x \varepsilon D$ and for all $p \varepsilon P$. The vectors p' and p" are referred to as the lower and upper parameters, respectively.

As an example of this definition, it may be observed that if the interval function F(P,X) is such that

$$F(P,X) = F(P_1,P_2,X_1,X_2),$$

$$F(P,X) = P_1X_1 - P_2X_2,$$

$$F(P,X) = [3,4]X_1 - [3,4]X_2,$$

then F(P,X) is interval separable over the set D where

$$D = \{x: x \geq 0\}.$$

For if

$$p' = \begin{pmatrix} 3 \\ \\ 4 \end{pmatrix}$$

and

$$p'' = \begin{pmatrix} 4 \\ \\ 3 \end{pmatrix}$$

then

$$3x_1 - 4x_2 \leq p_1x_1 - p_2x_2 \leq 4x_1 - 3x_2$$

where

$$3 \leq p_1 \leq 4$$

and

$$3 \leq p_2 \leq 4$$

for

$$x \geq 0.$$

Further, the function

$$f(p,x) = p_1x_1 - p_2x_2$$

is continuous with respect to variables p_1, p_2, x_1, and x_2 over the previously specified ranges.

Now suppose that F(P,X) has been defined slightly differently. Let

$$F(P,X) = F(P_1,X_1,X_2)$$

$$F(P,X) = P_1X_1 - P_1X_2$$

$$F(P,X) = [3,4]X_1 - [3,4]X_2$$

Then it may be observed that F(P,X) is not interval separable over the region D. For example, if

$$p' = 3$$

and

$$p'' = 4,$$

then it is not always true that

$$3x_1-3x_2 \leq p_1x_1-p_1x_2 \leq 4x_1-4x_2.$$

To illustrate this, let

$$p = 3.5$$

and

$$x = \begin{pmatrix} 0 \\ \\ 1 \end{pmatrix},$$

then

$$3.5(0) - 3.5(1) \not\leq 4(0) - 4(1).$$

However, F(P,X) would be interval separable over the region

$$E = \{x_1,x_2: \ x_1-x_2 \geq 0\}$$

It is easily verified for this region that

$$p' = 3$$

and

$$p'' = 4.$$

Now that interval separable functions have been defined, it is appropriate to define the ISCPP.

Definition 11: Interval separable convex programming problem (ISCPP): An ISCPP is an ICPP in which each interval function

and its first partials with respect to the X_j are interval separable over a set D which contains the solution set of the constraint relations.

In the following section a solution procedure for the ISCPP is given. This special procedure does not necessarily provide minimal width solutions. However, it is generally computationally more efficient than the minimal width solution method, particularly for an important class of ICPP, interval linear programs. The remainder of this section consists of theorems concerning interval separable functions.

Interval separable corresponding constraint sets

Theorems 2.6 and 2.7 will now be presented. The first theorem establishes the conditions under which a real valued constraint set contains a corresponding interval constraint set, and the second theorem specifies the conditions under which equality of solution sets occurs.

Theorem 2.6: Let the following be true:

a) The set of real vectors S is the solution set of

$$G_i(P,X) \leq H_i(P,X) \quad , i\varepsilon I_1,$$

$$G_i(P,X) \geq H_i(P,X) \quad , i\varepsilon I_2,$$

$$G_i(P,X) = H_i(P,X) \quad , i\varepsilon I_3.$$

b) Each $G_i(P,X)$ and $H_i(P,X)$ is interval separable for all $x\varepsilon D$ with lower and upper parameters p_i' and p_i'',

respectively.

c) $A = S \cap D$.

d) The set of real vectors B is the solution set of

$$g_i(p_i',x) \leq h_i(p_i'',x) \quad , i\varepsilon I_1,$$

$$g_i(p_i'',x) \geq h_i(p_i',x) \quad , i\varepsilon I_2,$$

$$g_i(p_i',x) \geq h_i(p_i'',x) \quad , i\varepsilon I_3,$$

$$g_i(p_i'',x) \leq h_i(p_i',x) \quad , i\varepsilon I_3,$$

$$x\varepsilon D.$$

Then the following statement is true: if $x\varepsilon A$, then $x\varepsilon B$.

Proof: Let $x\varepsilon A$, then $x\varepsilon S$ and $x\varepsilon D$ by definition of set intersection. Since $x\varepsilon S$, then it follows from theorem 2.1 that there exists some $p\varepsilon P$ such that

$$g_i(p,x) \leq h_i(p,x) \quad , i\varepsilon I_1,$$

$$g_i(p,x) \geq h_i(p,x) \quad , i\varepsilon I_2,$$

$$g_i(p,x) = h_i(p,x) \quad , i\varepsilon I_3.$$

Interval separability may be used to obtain

$$g_i(p_i',x) \leq g_i(p,x) \leq h_i(p,x) \leq h_i(p_i'',x) \quad , i\varepsilon I_1,$$

$$g_i(p_i'',x) \geq g_i(p,x) \geq h_i(p,x) \geq h_i(p_i',x) \quad , i\varepsilon I_2,$$

$$g_i(p_i',x) \leq g_i(p,x) = h_i(p,x) \leq h_i(p_i'',x) \quad , i\varepsilon I_3,$$

$$g_i(p_i'',x) \geq g_i(p,x) = h_i(p,x) \geq h_i(p_i',x) \quad , i\varepsilon I_3.$$

Rewriting the above relations and recalling that $x\varepsilon D$ results in

$$g_i(p_i',x) \leq h_i(p_i'',x) \quad , i\varepsilon I_1,$$

$$g_i(p_i'',x) \geq h_i(p_i',x) \quad , \; i\varepsilon I_2,$$

$$g_i(p_i',x) \leq h_i(p_i'',x) \quad , \; i\varepsilon I_3,$$

$$g_i(p_i'',x) \geq h_i(p_i',x) \quad , \; i\varepsilon I_3.$$

Thus, it follows that $x\varepsilon B$.

Theorem 2.6 has proved containment; now theorem 2.7 will specify the conditions under which equality of the solution sets occurs. The proof is tedious to follow due to the notation involved. However, conceptually it is rather straightforward. First it is assumed that the upper and lower parameters of all functions may be elements of the interval parameter set P and that no two functions use the same parameters. Then convex combinations of upper and lower parameters are formed in such a manner that the theorem is proved.

Theorem 2.7: Let the four conditions of theorem 2.6 be true, and also let the following two conditions be true.

e) Let there exist $p_1^*\varepsilon P$ and $p_2^*\varepsilon P$ such that for all i:

$$g_i(p_1^*,x) = g_i(p_i',x) \quad ,$$

$$h_i(p_1^*,x) = h_i(p_i'',x) \quad ,$$

$$g_i(p_2^*,x) = g_i(p_i'',x) \quad ,$$

$$h_i(p_2^*,x) = h_i(p_i',x) \quad .$$

f) No two functions of the interval constraint set use an identical parameter.

It follows that if $x\varepsilon B$, then $x\varepsilon A$.

Proof: Let $x \varepsilon B$, then conditions e) and f) imply

$$g_i(p_1^*,x) \leq h_i(p_1^*,x) \quad , i\varepsilon I_1, \qquad 2.8$$

$$g_i(p_1^*,x) \geq h_i(p_2^*,x) \quad , i\varepsilon I_2, \qquad 2.9$$

$$g_i(p_2^*,x) \leq h_i(p_1^*,x) \quad , i\varepsilon I_3, \qquad 2.10$$

$$g_i(p_2^*x) \geq h_i(p_2^*,x) \quad , i\varepsilon I_3. \qquad 2.11$$

Now in light of condition f), partition the interval parameter set P into disjoint subsets P_i so that

$$P = \begin{pmatrix} P_1 \\ P_2 \\ \cdot \\ \cdot \\ \cdot \\ P_{2m} \end{pmatrix} ,$$

where the partitioning is done so that the interval constraint set given in condition a) may be written as

$$G_i(P_i,X) \leq H_i(P_{m+i},X) \quad , i\varepsilon I_1,$$

$$G_i(P_i,X) \geq H_i(P_{m+i},X) \quad , i\varepsilon I_2,$$

$$G_i(P_i,X) = H_i(P_{m+i},X) \quad , i\varepsilon I_3.$$

Similarly, partition the parameter set p so that the following is true

$$g_i(p_i,x) = g_i(p,x) \quad ,$$

$$h_i(p_{m+i},x) = h_i(p,x) \quad ,$$

for all i and for all $p\varepsilon P$. Also, denote the corresponding partitions of p_1^* by $p_{1,i}^*$ and $p_{1,m+i}^*$ and the partitions of p_2^* by $p_{2,i}^*$ and $p_{2,m+i}^*$. Further let

$$f_i(p,x) = g_i(p,x) - h_i(p,x)$$

for all i. Thus it is true that

$$f_i(p^*_{1,i},p^*_{1,m+i},x) = g_i(p^*_{1,i},x) - h_i(p^*_{1,m+i},x) \quad , i\varepsilon I_3, \qquad 2.12$$

$$f_i(p^*_{2,i},p^*_{2,m+i},x) = g_i(p^*_{2,i},x) - h_i(p^*_{2,m+i},x) \quad , i\varepsilon I_3. \qquad 2.13$$

Now since the $G_i(P_i,X)$ and $H_i(P_{m+i},X)$ are interval separable, then by definition the $g_i(p_i,x)$ and $h_i(p_{m+i},x)$ are continuous. Note that in relations 2.12 and 2.13 the values of $f_i(p_i,p_{m+i},x)$ vary from non-positive to non-negative. Thus, by continuity it follows for some particular

$$(p^o_i,p^o_{m+i}) = \alpha_i(p^*_{1,i},p^*_{1,m+i}) + (1-\alpha_i)(p^*_{2,i},p^*_{2,m+i})$$

where

$$0 \leq \alpha \leq 1$$

it is true that

$$f_i(p^o_i,p^o_{m+i},x) = 0 = g_i(p^o_i,x) - h_i(p^o_{m+i},x)$$

Further since p^*_1 and p^*_2 are elements of p and since (p^o_i,p^o_{m+i}) is a convex combination, then

$$p^o_i \varepsilon P_i \quad , i\varepsilon I_3,$$

$$p^o_{m+i} \varepsilon P_{m+i} \quad , i\varepsilon I_3.$$

Recalling that the partitioning was performed so that

$$g_i(p^*_{1,i},x) \leq h_i(p^*_{1,m+i},x) \quad , i\varepsilon I_1,$$

$$g_i(p^*_{2,i},x) \geq h_i(p^*_{2,m+i},x) \quad , \; i\varepsilon I_2,$$

then letting

$$p^o_i = p^*_{1,i} \quad , \; i\varepsilon I_1,$$

$$p^o_{m+i} = p^*_{1,m+i} \quad , \; i\varepsilon I_1,$$

$$p^o_i = p^*_{2,i} \quad , \; i\varepsilon I_2,$$

$$p^o_{m+i} = p^*_{2,m+i} \quad , \; i\varepsilon I_2,$$

it follows that

$$p^o_k \varepsilon P_k \quad , \; k=1,\ldots,2m,$$

or that $p^o \varepsilon P$. Thus

$$g_i(p^o,x) \leq h_i(p^o,x), \; i\varepsilon I_1,$$

$$g_i(p^o,x) \geq h_i(p^o,x), \; i\varepsilon I_2,$$

$$g_i(p^o,x) = h_i(p^o,x), \; i\varepsilon I_3;$$

and by definition of a solution set, it follows that $x\varepsilon A$, proving the theorem.

Numerical examples

The use of the preceding two theorems is illustrated by the following two numerical examples, which also contrast the use of identical and equal interval numbers. The first example illustrates containment of solution sets (theorem 2.6), and the second demonstrates equality of solution sets (theorem 2.7).

First, consider the interval equations

$$G_1(P,X) = P_1X_1+P_1X_2 = 2P_2 = H_1(P,X) \quad ,$$

$$G_2(P,X) = P_1X_1 \qquad = P_2 = H_2(P,X)$$

where

$$P_1 = [1,2]$$

and

$$P_2 = [1,2]$$

It may be observed that all functions are interval separable over

$$D = \{x_1,x_2: \ x_1 \geq 0, \ x_2 \geq 0\}$$

by letting

$$p' = \begin{pmatrix} 1 \\ \\ 1 \end{pmatrix}$$

and

$$p'' = \begin{pmatrix} 2 \\ \\ 2 \end{pmatrix} .$$

This is true because for all $p \varepsilon P$, all $x \varepsilon D$; and for all i, it may be seen that

$$g_i(p',x) \leq g_i(p,x) \leq g_i(p'',x),$$

$$h_i(p',x) \leq h_i(p,x) \leq h_i(p'',x).$$

Now let S_1 be the solution set of the two interval equations, then $S_1 \cap D$ may be determined. Using the definition of solution sets, it is seen that $S_1 \cap D$ consists of those x such that

$$p_1x_1+p_1x_2=2p_2,$$
$$p_1x_1 \quad =p_2 ,$$
$$1 \leq p_1,p_2 \leq 2 ,$$
$$x_1,x_2 \geq 0 \quad .$$

Thus

$$x_1=p_2/p_1$$

and

$$p_1(p_2/p_1)+p_1x_2=2p_2$$

or

$$x_2=p_2/p_1$$

Since the term p_2/p_1 may vary between 1/2 and 2, then

$$S_1 \cap D = \{x_1,x_2: \ 1/2 \leq x_1 \leq 2, \ x_2=x_1\}.$$

Now theorem 2.6 may be used to obtain the set B which contains $S_1 \cap D$. Let B include all x satisfying the relations given in condition d) of theorem 2.4. Thus B is specified by

$$x_1+x_2 \leq 4 ,$$
$$2x_1+2x_2 \geq 2 ,$$
$$x_1 \quad \leq 2 ,$$
$$2x_1 \quad \geq 1 ,$$
$$x_1,x_2 \geq 0 .$$

That B contains $S_1 \cap D$ is readily seen by referring to Figure 4 on the next page. The set $S_1 \cap D$ is represented by the dark line and the set B by the shaded region.

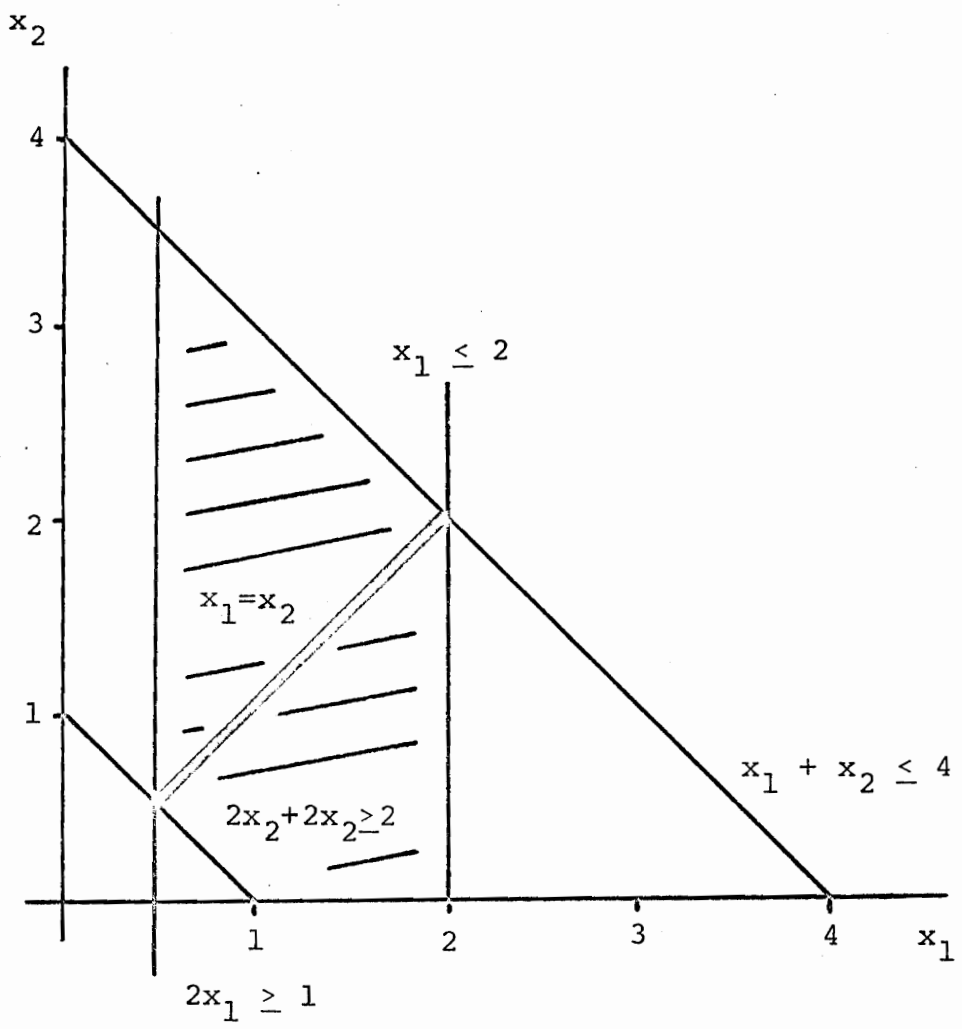

Figure 4. Solution sets.

Now consider a second example which is structured such that theorem 2.7 may be used and the solution sets are equal. Let S_2 be the solution set of

$$P_1X_1+P_1X_2 = 2P_2$$

$$P_3X_1 = P_4$$

where for all k

$$P_k = [1,2].$$

The equations are similar to those of the preceding example, except that no two functions use the same parameter. Further, the sets B and D may be defined as before since if

$$p' = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

and

$$p'' = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \end{pmatrix} ,$$

then it may seem that the equations are interval separable over D and that $S_2 \cap D$ is contained in B by theorem 2.6. The set $S_2 \cap D$ is specified by the following relations:

$$p_1x_1+p_1x_2 = p_2$$

$$p_3x_1 = p_4$$

$$1 \leq p_1,p_2,p_3,p_4 \leq 2$$

$$x_1,x_2 \geq 0 \quad .$$

Thus

$$x_1 = p_4/p_3$$

and

$$x_2 = 2p_2/p_1 - x_1 \quad .$$

And it follows that

$$S_2 \cap D = \{x_1, x_2: \ 1-x_1 \leq x_2 \leq 4-x_1; \ 1/2 \leq x_1 \leq 2; \ x_1, x_2 \geq 0\}.$$

Now let

$$p_1^* = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \end{pmatrix}$$

and

$$p_2^* = \begin{pmatrix} 2 \\ 1 \\ 2 \\ 1 \end{pmatrix} \quad .$$

It may be seen that this choice of p_1^* and p_2^* fulfills condition e) of theorem 2.7; also condition f) is satisfied since no two functions use the same parameter. Thus the additional requirements of theorem 2.7 have been met, and $S_2 \cap D$ is not only contained by B, but it is also equal to it. This equality may be readily seen by referring to Figure 4 and realizing that the relations

$$2x_1 + 2x_2 = 2$$

and

$$x_1 + x_2 = 1$$

are equivalent. This concludes the numerical examples of the use of theorems 2.6 and 2.7.

Properties of interval separable functions

The above two theorems are used in the development of the second of the two solution techniques given in the following section. This second technique provides solutions to interval separable programming problems (ISCPPs). Some properties of interval separable functions which are used in the next section are presented in the group of four theorems given below.

Theorem 2.8: Let $F(P,X)$ be interval separable over D and let Z be a non-negative interval variable. Then $Z \cdot F(P,X)$ is interval separable over D $\{z: z \geq 0\}$.

Proof: Since $F(P,X)$ is interval separable, then there exist lower and upper parameters, p' and p", such that

$$f(p',x) \leq f(p,x) \leq f(p'',x)$$

for all $p \in P$ and $x \in D$. Hence, if z is non-negative, then

$$zf(p',x) \leq zf(p,x) \leq zf(p'',x)$$

for all $p \in P$ and $x \in D$. Thus, $Z \cdot F(P,X)$ is interval separable over D $\{z: z \geq 0\}$.

Theorem 2.9: Let $F(P,X)$ be interval separable over D, and let Z be a non-positive interval variable. Then $Z \cdot F(P,X)$ is interval separable over D $\{z: z \leq 0\}$.

Proof: The proof of this theorem proceeds as that of theorem 2.4, except that now $Z \cdot F(P,X)$ is seen to be interval

separable with lower and upper parameters p" and p', respectively.

Theorem 2.10: Let there be a set of interval functions, $F_i(P_i,X)$, all of which are interval separable over D. Let M and N be arbitrary index sets. Then the function

$$\sum_{i\in M} F_i(P_i,X) - \sum_{i\in N} F_i(P_i,x)$$

is interval separable over D.

Proof: Let the lower and upper parameters of $F_i(P,X)$ be p_i' and p_i'' respectively. Then

$$f_i(p_i'',x) \geq f_i(p,x) \geq f_i(p_i',x) \quad , \ i\in M,$$

$$-f_i(p_i',x) \geq -f_i(p,x) \geq -f_i(p_i'',x) \quad , \ i\in N,$$

for all $x\in D$ and $p_i\in P_i$. It follows that

$$\sum_{i\in M} f(p_i'',x) - \sum_{i\in N} f(p_i',x) \geq \sum_{i\in M} f_i(p,x) -$$

$$\sum_{i\in N} f_i(p,x) \geq \sum_{i\in M} f_i(p_i',x) - \sum_{i\in N} f_i(p_i'',x)$$

for all $x\in D$ and $p_i\in P_i$. Then by the definition of interval separability the function

$$\sum_{i\in M} F_i(P_i,X) - \sum_{i\in N} F_i(P_i,X)$$

is interval separable over D with lower parameters p_i', $i\in M$, and p_i'', $i\in N$, and upper parameters p_i'', $i\in M$, and p_i', $i\in N$.

Theorem 2.11: Let F(P,X) be interval separable over D, and let Z be an interval variable. Then

$$Z\cdot F(P,X) \equiv W\cdot F(P,X) - V\cdot F(P,X)$$

where W and V are non-negative interval variables which may

be chosen so that

$$W-V \equiv Z.$$

Proof: First it will be shown that there exist non-negative W and V such that (W-V) and Z are identical. Consider

$$Z=W-V \qquad 2.12$$

where W and V are non-negative. Equation 2.12 may be written as

$$[z_L-z_R] = [w_L,w_R] - [v_L,v_R],$$

implying

$$[z_L,z_R] = [w_L-v_R,w_R-v_L].$$

That Z may be expressed this way may be shown by examining the three possible ways in which the values of z_L and z_R can occur. If both are non-negative then Z may be represented by letting V equal [0,0]. Similarly, if Z were non-positive, then let W be [0,0]. Finally, if z_L were non-positive and z_R non-negative, then let

$$w_L = v_L = 0,$$

so that w_L-v_R is the desired non-positive number and w_R-v_L the desired non-negative number. Thus, Z and (W-V) are equal. Now let G[Z,(W-V),Y] be any interval function as in definition 4 (identical and equal interval vectors). Then

$$G[Z,(W-V),Y] = \{g[z,(w-y),y]:$$

$$z_L \leq z \leq z_R,\ w_L-v_R \leq w-v \leq w_R-v_L, y \varepsilon Y\}.$$

It is permissable to require that

$$z = w-v$$

so that

$$G[Z,(W-V),Y] = \{g[z,(w-v),y]:$$
$$w-v=z, z_L \leq z \leq z_R, w_L-v_R \leq w-v \leq w_R-v_L, \ y\epsilon Y\}.$$

It has already been shown that it is possible to stipulate that

$$Z = W-V$$

where W and V are non-negative. Merging these requirements results in

$$G[Z,(W-V),Y] = \{g[z,(w-v),y]: \ w-v=z, \ z_L \leq z \leq z_R, \ y\epsilon Y\}.$$

which implies that

$$Z \equiv W-V.$$

Thus, it may always be assumed that there exist non-negative values W and V such that their difference is identical to Z. Now by theorem 2.1 it is seen that

$$Z-(W-V) \equiv [0,0].$$

Theorem 2.2 may be used to multiply both sides of this identity by any arbitrary value of F(P,X) so that

$$[Z-(W-V)]\cdot F(P,X) \equiv [0,0]\cdot F(P,X).$$

Obviously

$$[0,0]\cdot F(P,X) \equiv [0,0]$$

and so transitivity and then distributivity may be used to obtain

$$Z\cdot F(P,X)-(W-V)\cdot F(P,X) \equiv [0,0].$$

Adding $(W-V)\cdot F(P,X)$ to both sides as in theorem 2.2 and using the additive identity implies

$$Z\cdot F(P,X) \equiv (W-V)\cdot F(P,X).$$

Then theorem 2.3 may be used to obtain

$$Z\cdot F(P,X) \equiv W\cdot F(P,X)-V\cdot F(P,X)$$

where

$$Z \equiv W-V,$$

and W and V have non-negative elements so that

$$W,V \geq [0,0]$$

thus proving the theorem.

SOLUTION ALGORITHMS

In this section methods for solving ICPPs are presented. The first method is appropriate for obtaining the minimal width solution of any ICPP. The second method is applicable only for ISCPPs and does not necessarily provide minimal width solutions. After the presentation of these methods, two commonly occurring special cases are examined: the interval linear programming problem (ILPP) and the interval quadratic programming problem (IQPP).

Minimal width solution procedure

The minimal width solution procedure is given by theorem 2.12 below.

Theorem 2.12: Let the minimal width solution of the following ICPP

$$\text{maximize} \qquad F(P,X), \tag{2.13}$$

subject to:

$$G_i(P,X) \leq H_i(P,X) \quad , i\varepsilon I_1, \tag{2.14}$$

$$G_i(P,X) \geq H_i(P,X) \quad , i\varepsilon I_2, \tag{2.15}$$

$$G_i(P,X) = H_i(P,X) \quad , i\varepsilon I_3, \tag{2.16}$$

be

$$X^* = \begin{pmatrix} [x^*_{1L}, x^*_{1R}] \\ \vdots \\ [x^*_{nL}, x^*_{nR}] \end{pmatrix}. \tag{2.17}$$

Let y^*_{jL} and y^*_{jR} be the global optima of the following 2n programs

$$\begin{matrix}\text{minimize} \\ \text{maximize}\end{matrix} \quad y_j$$

subject to:

$$g_i(p,y) \leq h_i(p,y) \quad , i\varepsilon I_1, \tag{2.19}$$

$$\mu_i \geq 0 \quad , i\varepsilon I_1, \tag{2.20}$$

$$g_i(p,y) \geq h_i(p,y) \quad , i\varepsilon I_2, \tag{2.21}$$

$$\mu_i \leq 0 \quad , i\varepsilon I_2, \tag{2.22}$$

$$g_i(p,y) = h_i(p,y) \quad , i\varepsilon I_3, \tag{2.23}$$

$$\mu_i \text{ unrestricted} \quad , i\varepsilon I_3, \tag{2.24}$$

$$\frac{\partial}{\partial x_j}\{f(p,y) - \Sigma_i \mu_i [g_i(p,y) - h_i(p,y)]\} = 0, \quad \forall j, \tag{2.25}$$

$$\sum_i \mu_i [g_i(p,y) - h_i(p,y)] = 0 \qquad 2.26$$

$$p_{kL} \leq p \leq p_{kR} \quad , \forall k, \qquad 2.27$$

and let

$$y^* = \begin{pmatrix} [y^*_{1L}, y^*_{1R}] \\ \vdots \\ [y^*_{nL}, y^*_{nR}] \end{pmatrix}.$$

Then it is true that

$$Y^* = X^*.$$

Proof: Let Q be the interval optimal set of the ICPP specified by relations 2.13 through 2.16. Now using the definition of an interval optimal set, if y is an element of Q, then y is an optimal point for the CPP

$$\text{maximize} \qquad f(p^1, z) \qquad 2.28$$

subject to:

$$g_i(p^1, z) \leq h_i(p^1, z) \quad , i \varepsilon I_1, \qquad 2.29$$

$$g_i(p^1, z) \geq h_i(p^1, z) \quad , i \varepsilon I_2, \qquad 2.30$$

$$g_i(p^1, z) = h_i(p^1, z) \quad , i \varepsilon I_3, \qquad 2.31$$

for some $p^1 \varepsilon P$. Further theorem 1.1 indicates that y is an otpimal point of the above program if and only if

$$g_i(p^1, y) \leq h_i(p^1, y) \quad , i \varepsilon I_1, \qquad 2.32$$

$$\mu_i \geq 0 \quad , i \varepsilon I_1, \qquad 2.33$$

$$g_i(p^1, y) \geq h_i(p^1, y) \quad , i \varepsilon I_2, \qquad 2.34$$

$$\mu_i \leq 0 \quad , i \varepsilon I_2, \qquad 2.35$$

$$g_i(p^1,y) = h_i(p^1,y) \quad , i\varepsilon I_3, \tag{2.36}$$

$$\mu_i \text{ unrestricted} \quad , i\varepsilon I_3, \tag{2.37}$$

$$\frac{\partial}{\partial y_j}\{f(p^1,y) - \Sigma_i \mu_i[g_i(p^1,y)-h_i(p^1,y)]\} = 0 \ , \ \forall j, \tag{2.38}$$

$$\Sigma_i \mu_i[g_i(p^1,y)-h_i(p^1,y)] = 0. \tag{2.39}$$

Thus restrictions 2.19 through 2.27 define Q in that they consist of all possible p^1 such that relations 2.32 through 2.39 are true. Further the y^*_{jL} and y^*_{jR} are the minimal and maximal members of Q by the theorem statement that these points are global optima. Hence Y* is the minimal width solution of the ICPP and

$$Y^* = X^*,$$

proving the theorem.

In the process of solving each of these 2n programs, the minimal and maximal values of the various y_j thus far encountered may continually be recorded. These points may later be used as starting points when the actual minimal and maximal values of the remaining y_j are to be determined. This may result in a considerable computational savings, particularly when one considers that the series of 2n programs would nearly always involve non-convex programs. However, even then, the computational burden imposed by the minimal width solution procedure may be overwhelming for larger problems. To reduce this burden a method was devised to solve a special class of ICPPs, the ISCPPs.

ISCPP algorithm

This method will usually be more efficient than the general ICPP solution procedure, but it will not necessarily provide minimal width solutions. As a consequence of theorems 2.8 through 2.11 it may be seen that interval linear and quadratic programs, two programs commonly occurring in the real number system, are ISCPPs. Moreover, the ISCPP procedure may be further simplified for these two program types due to their special structures. Even then the resulting programs are computationally expensive, and so in following chapter methods for further efficiencies are examined. The ISCPP solution procedure is given by theorem 2.13 below.

Theorem 2.13: Let S be the interval optimal set of the following ISCPP

$$\text{maximize} \quad F(P,X) \ , \tag{2.40}$$

subject to:

$$G_i(P,X) \leq H_i(P,X) \quad , \ i\varepsilon I_1, \tag{2.41}$$

$$G_i(P,X) \geq H_i(P,X) \quad , \ i\varepsilon I_2, \tag{2.42}$$

$$G_i(P,X) = H_i(P,X) \quad , \ i\varepsilon I_3, \tag{2.43}$$

$$S \subset T \tag{2.44}$$

where all functions are interval separable over T. Let the set D be defined by relations 2.45 through 2.56 given below:

$$g_i(p_i',x) \leq h_i(p_{m+i}'',x) \quad , \ i\varepsilon I_1, \tag{2.45}$$

$$\mu_i \geq 0 \quad , i\epsilon I_1, \tag{2.46}$$

$$g_i(p_i'',x) \geq h_i(p'_{m+i},x) \quad , i\epsilon I_2, \tag{2.47}$$

$$\mu_i \leq 0 \quad , i\epsilon I_2, \tag{2.48}$$

$$g_i(p_i',x) \leq h_i(p''_{m+i},x) \quad , i\epsilon I_3, \tag{2.49}$$

$$g_i(p_i'',x) \geq h_i(p'_{m+i},x) \quad , i\epsilon I_3, \tag{2.50}$$

$$w_i, v_i \geq 0 \quad , i\epsilon I_3, \tag{2.51}$$

$$\begin{aligned}\frac{\partial}{\partial x_j}\{f(p'_{4m+j},x) &- \sum_{i\epsilon I_1}\mu_i[g_i(p''_{2m+i,j},x)-h_i(p'_{3m+i,j},x)] \\ &- \sum_{i\epsilon I_2}\mu_i[g_i(p'_{2m+i,j},x)-h_i(p''_{3m+i,j},x)] \\ &- \sum_{i\epsilon I_3}w_i[g_i(p''_{2m+i,j},x)-h_i(p'_{3m+i,j},x)] \\ &+ \sum_{i\epsilon I_3}v_i[g_i(p'_{2m+i,j},x)-h_i(p''_{3m+i,j},x)]\} \leq 0, \quad \forall j,\end{aligned} \tag{2.52}$$

$$\begin{aligned}\frac{\partial}{\partial x_j}\{f(p''_{4m+j},x) &- \sum_{i\epsilon I_1}\mu_i[g_i(p'_{2m+i,j},x)-h_i(p''_{3m+i,j},x)] \\ &- \sum_{i\epsilon I_2}\mu_i[g_i(p''_{2m+i,j},x)-h_i(p'_{3m+i,j},x)] \\ &- \sum_{i\epsilon I_3}w_i[g_i(p'_{2m+i,j},x)-h_i(p''_{3m+i,j},x)] \\ &+ \sum_{i\epsilon I_3}v_i[g_i(p''_{2m+i,j},x)-h_i(p'_{3m+i,j},x)]\} \geq 0, \quad \forall j,\end{aligned} \tag{2.53}$$

$$\begin{aligned}&\sum_{i\epsilon I_1}\mu_i[g_i(p_i',x)-h_i(p''_{m+i},x)] \\ &+ \sum_{i\epsilon I_2}\mu_i[g_i(p_i'',x)-h_i(p'_{m+i},x)] \\ &+ \sum_{i\epsilon I_3}w_i[g_i(p_i',x)-h_i(p''_{m+i},x)] \\ &- \sum_{i\epsilon I_3}v_i[g_i(p_i'',x)-h_i(p'_{m+i},x)] \leq 0, \quad \forall j,\end{aligned} \tag{2.54}$$

$$\sum_{i\varepsilon I_1} \mu_i [g_i(p_i'',x) - h_i(p_{m+i}',x)]$$

$$+ \sum_{i\varepsilon I_2} \mu_i [g_i(p_i',x) - h_i(p_{m+i}'',x)] \qquad 2.55$$

$$- \sum_{i\varepsilon I_3} w_i [g_i(p_i'',x) - h_i(p_{m+i}',x)]$$

$$- \sum_{i\varepsilon I_3} v_i [g_i(p_i',x) - h_i(p_{m+i}'',x)] \geq 0, \quad \forall j,$$

$$x\varepsilon T. \qquad 2.56$$

Then it is true that

$$S \subset D.$$

The symbol (') refers to the lower parameter of the interval function corresponding to the appropriate real extension. For example, p_i' in the term $g_i(p_i',x)$ of relation 2.44 is the lower parameter of $G_i(P,X)$; and $p'_{2m+i,j}$ in term $g_i(p'_{2m+i,j},x)$ of relation 2.51 is the lower parameter of $\frac{\partial}{\partial x_j} G_i(P,X)$. Similarly the symbol (") refers to upper parameters.

Proof: Theorem 2.12 may be used to obtain the interval optimal set, S, of the ISCPP given by relation 2.40 through 2.44. Thus S is defined by the relations below:

$$g_i(p,x) \leq h_i(p,x) \quad , i\varepsilon I_1, \qquad 2.57$$

$$\mu_i \geq 0 \quad , i\varepsilon I_1, \qquad 2.58$$

$$g_i(p,X) \geq h_i(p,x) \quad , i\varepsilon I_2, \qquad 2.59$$

$$\mu_i \leq 0 \quad , i\varepsilon I_2, \qquad 2.60$$

$$g_i(p,x) = h_i(p,x) \quad , i\varepsilon I_3, \qquad 2.61$$

$$\mu_i \text{ unrestricted} \quad , i\varepsilon I_3, \tag{2.62}$$

$$\frac{\partial}{\partial x_j}\{f(p,x)-\Sigma_i \mu_i[g_i(p,x)-b_i]\} = 0, \quad \forall j, \tag{2.63}$$

$$\Sigma_i \mu_i[g_i(p,x)-b_i] = 0, \tag{2.64}$$

$$p_{kL} \leq p_k \leq p_{kR} \quad , \forall k, \tag{2.65}$$

$$x\varepsilon T. \tag{2.70}$$

Theorem 2.5 indicates that the set S is alternatively specified by the following interval relations

$$G_i(P,X) \leq H_i(P,X), \tag{2.71}$$

$$U_i \geq [0,0] \quad , \tag{2.72}$$

$$G_i(P,X) \geq H_i(P,X), \tag{2.73}$$

$$U_i \leq [0,0] \quad , \tag{2.74}$$

$$G_i(P,X) = H_i(P,X), \tag{2.75}$$

$$U_i \text{ unrestricted}, \tag{2.76}$$

$$\frac{\partial}{\partial X_j}\{F(P,X)-\Sigma_i U_i[G_i(P,X)-H_i(P,X)]\} = [0,0] \quad , \forall j, \tag{2.77}$$

$$\Sigma_i U_i[G_i(P,X)-H_i(P,X)] = [0,0], \tag{2.78}$$

in conjunction with the restriction

$$x\varepsilon T. \tag{2.79}$$

Now let E be the solution set of the following interval relations

$$G_i(P_i,X) \leq H_i(P_{m+i},X) \quad , i\varepsilon I_1, \tag{2.80}$$

$$U_i \geq [0,0] \quad , i\varepsilon I_1, \tag{2.81}$$

$$G_i(P_i,X) \geq H_i(P_{m+i},X) \quad , i\varepsilon I_2, \tag{2.82}$$

$$U_i \leq [0,0] \quad , i\varepsilon I_2, \tag{2.83}$$

$$G_i(P_i,X) = H_i(P_{m+i},X) \quad , i\varepsilon I_3, \tag{2.84}$$

$$U_i \text{ unrestricted} \quad , i\varepsilon I_3, \tag{2.85}$$

$$\frac{\partial}{\partial x_j}\{F(P_{6m+j} - \sum_i U_i[G_i(P_{2m+i,j},X) - H_i(P_{3m+i,j},X)]\} = [0,0] \quad , \forall j, \tag{2.86}$$

$$\sum_i U_i[G_i(P_{4m+i},X) - H_i(P_{5m+i},X)] = [0,0] \tag{2.87}$$

$$P_k = P \quad , \forall k, \tag{2.88}$$

in conjunction with the restriction

$$x\varepsilon T. \tag{2.89}$$

Note that the p_k are not required to be identical, but merely equal, a less severe restriction. Thus, it follows that

$$S \subset E \tag{2.90}$$

Now let

$$W_r - V_r \equiv U_r \tag{2.91}$$

for any r so that by theorem 2.11

$$U_i[G_i(P_r,X) - H_i(P_{m+r},X)] \equiv (W_i - V_i) \cdot [G_i(P_r,X) - H_i(P_{m+r},X)]. \tag{2.92}$$

Then theorem 2.4 and the distributive law for identical numbers may be used to rewrite the interval relations specifying the set E as

$$G_i(P_i,X) \leq H_i(P_{m+i},X) \quad , i\varepsilon I_1, \tag{2.93}$$

$$U_i \geq [0,0] \quad , i\varepsilon I_1, \tag{2.94}$$

$$G_i(P_i,X) \geq H_i(P_{m+i},X) \quad , i\varepsilon I_2, \tag{2.95}$$

$$U_i \leq [0,0] \quad , i\varepsilon I_2, \tag{2.96}$$

$$G_i(P_i,X) = H_i(P_{m+i},X) \quad , i\varepsilon I_3, \tag{2.97}$$

$$(W_i - V_i) \text{ unrestricted} \quad , i\varepsilon I_3, \tag{2.98}$$

$$W_i, V_i \geq [0,0] \quad , i\varepsilon I_3, \tag{2.99}$$

$$\frac{\partial}{\partial x_j}\{F(P_{6m+j},X) - \sum_{i\varepsilon I_1, I_2} U_i[G_i(P_{2m+i,j},X) - H_i(P_{3m+i,j},X)] - \sum_{i\varepsilon I_3} W_i[G_i(P_{2m+i,j},X) - H_i(P_{3m+i,},X)] + \sum_{i\varepsilon I_3} V_i[G_i(P_{2m+i,j},X) - H_i(P_{3m+i,j},X)]\}$$

$$= [0,0] \quad , \forall j, \tag{2.100}$$

$$\sum_{i\varepsilon I_1, I_2} U_i[P_{4m+i},X) - H_i(P_{5m+i},X)] + \sum_{i\varepsilon I_3} W_i[G_i(P_{4m+i},X) - H_i(P_{5m+i},X)] - \sum_{i\varepsilon I_3} V_i[G_i(P_{4m+i},X) - H_i(P_{5m+i},X) = [0,0] \tag{2.101}$$

$$P_k = P \quad , \forall j, \tag{2.102}$$

$$x\varepsilon T \tag{2.103}$$

By the theorem statement G_i, H_i, and the various partials are all interval separable. Thus, theorems 2.8, 2.9, and 2.10 may be used to show that all functions in relations 2.93 through 2.101 are interval separable. Further, since

no function involves the same parameter theorem 2.7 may be used to specify a real valued constraint set which in conjunction with relations 2.102 and 2.103 is equal to E. This set is specified by relations 2.104 through 2.115 below:

$$g_i(p'_i,x) \leq h_i(p''_{m+i},x) \quad , i\varepsilon I_1 \qquad 2.104$$

$$\mu_i \geq 0 \quad , i\varepsilon I_1 \qquad 2.105$$

$$g_i(p''_i,x) \geq h_i(p'_{m+i},x) \quad , i\varepsilon I_2 \qquad 2.106$$

$$\mu_i \leq 0 \quad , i\varepsilon I_2 \qquad 2.107$$

$$g_i(p'_i,x) \leq h_i(p''_{m+i},x) \quad , i\varepsilon I_3, \qquad 2.108$$

$$g_i(p''_i,x) \geq h_i(p'_{m+i},x) \quad , i\varepsilon I_3, \qquad 2.109$$

$$w_i, v_i \geq 0 \quad , i\varepsilon I_3, \qquad 2.110$$

$$\frac{\partial}{\partial x_j}\{f(p'_{6m+j},x) - \sum_{i\varepsilon I_1}\mu_i[g_i(p''_{2m+i,j},x)-h_i(p'_{3m+i,j},x)] - \sum_{i\varepsilon I_2}\mu_i[g_i(p'_{2m+i,j},x)-h_i(p''_{3m+i,j},x)] - \sum_{i\varepsilon I_3}w_i[g_i(p''_{2m+i,j},x)-h_i(p'_{3m+i,j},x)] + \sum_{i\varepsilon I_3}v_i[g_i(p'_{2m+i,j},x)-h_i(p''_{3m+i,j},x)]\} \leq 0, \quad \forall j, \qquad 2.111$$

$$\frac{\partial}{\partial x_j}\{f(p''_{ym+j},x) - \sum_{i\varepsilon I_1}\mu_i[g_i(p'_{2m+i,j},x)-h_i(p''_{3m+i,j},x)] - \sum_{i\varepsilon I_2}\mu_i[g_i(p''_{2m+i,j},x)-h_i(p'_{3m+i,j},x)] - \sum_{i\varepsilon I_3}w_i[g_i(p'_{2m+i,j},x)-h_i(p''_{3m+i,j},x)] + \sum_{i\varepsilon I_3}v_i[g_i(p''_{2m+i,j},x)-h_i(p'_{3m+i,j},x)]\} \geq 0, \quad \forall j, \qquad 2.112$$

$$\sum_{i\in I_1} \mu_i [g_i(p'_{4m+i},x) - h_i(p''_{5m+i},x)]$$

$$+ \sum_{i\in I_2} \mu_i [g_i(p''_{4m+i},x) - h_i(p'_{5m+i},x)]$$

$$+ \sum_{i\in I_3} w_i [g_i(p'_{4m+i},x) - h_i(p''_{5m+i},x)] \qquad 2.113$$

$$- \sum_{i\in I_3} v_i [g_i(p''_{4m+i},x) - h_i(p'_{5m+i},x)] \leq 0 \quad ,$$

$$\sum_{i\in I_1} \mu_i [g_i(p''_{4m+i},x) - h_i(p'_{5m+i},x)]$$

$$+ \sum_{i\in I_2} \mu_i [g_i(p'_{4m+i},x) - h_i(p''_{5m+i},x)]$$

$$+ \sum_{i\in I_3} w_i [g_i(p''_{4m+i},x) - h_i(p'_{5m+i},x)] \qquad 2.114$$

$$- \sum_{i\in I_3} v_i [g_i(p'_{4m+i},x) - h_i(p''_{5m+i},x)] \geq 0 \quad ,$$

$$x\in T. \qquad 2.115$$

Now since it may be seen that

$$p'_{4m+i} = p'_i,$$

$$p''_{4m+i} = p''_i,$$

$$p'_{5m+i} = p'_i,$$

and

$$p''_{5m+i} = p''_i,$$

then relations 2.104 through 2.115 may be relabeled as necessary to coincide with relations 2.45 through 2.56. Hence

$$E = D$$

and thus

$$S \subset D,$$

proving the theorem.

The set D is specified by the real valued relations 2.45 through 2.56. The following series of 2n programs may be solved to determine a solution of the ISCPP:

$$\begin{Bmatrix}\text{maximize}\\ \text{minimize}\end{Bmatrix} \quad x_j \quad , \; j=1,\ldots,n \; ,$$

subject to:

$$x \varepsilon D.$$

The constraint set used to solve ISCPPs seems to contain more restrictions than does that of the minimal width method. However, it must be noted that not only are the functions less non-linear because the p_r' and p_r'' are constants, but also all of the restrictions on the p_k are removed. Thus, in general the ISCPP algorithm will be more efficient than the minimal width algorithm. Also, the ISCPP routine may be used to efficiently provide starting points for the minimal width routine. One possible disadvantage of the ISCPP is the restriction that $x \varepsilon T$ where T is such that all functions are interval separable over T. Unless it is possible to determine in advance that the optimal set S is contained in T so that this restraint may be dropped due to redundance, then it simply may not be possible to use the ISCPP technique. Fortunately, interval linear and quadratic programming problems (ILPPs and IQPPs) are such that this may be determined in advance. Non-negative and non-positive variables give rise to separable linear functions. If a variable is not restricted, this may be han-

dled in the same manner as was done for U_i, $i\varepsilon I_3$, in the preceding theorem. The non-restricted interval variable may be defined to be identical to the difference of two non-negative interval variables.

Interval linear programming

The results of the ISCPP procedure may be applied directly to ILPPs and IQPPs. It is possible, however, to further exploit their special structure and simplify the real valued constraint set of the ISCPP. Theorems 2.14 and 2.15 deal with the ILPP and the IQPP. Non-negativity restrictions have been added simply to insure separability. The identities

$$X \equiv W-V$$

or

$$X \equiv -Y$$

may always be used to obtain this standard form.

Theorem 2.14: Let S be the interval optimal set of the following ILPP:

$$\text{Max} \sum_j C_j X_j \qquad 2.116$$

subject to:

$$\sum_j A_{ij} X_j \leq B_i \quad , i\varepsilon I_1, \qquad 2.117$$

$$\sum_j A_{ij} X_j \geq B_i \quad , i\varepsilon I_2, \qquad 2.118$$

$$\sum_j A_{ij} X_j = B_i \quad , i\varepsilon I_3, \qquad 2.119$$

$$X_j \geq [0,0] \quad , \forall j. \tag{2.120}$$

Let D be the solution set of the following relations

$$\sum_j c_{jL}x_j \leq i\sum_{\varepsilon I_1} b_{iR}\mu_i + i\sum_{\varepsilon I_2} b_{iL}\mu_i + i\sum_{\varepsilon I_3} b_{iR}w_i - i\sum_{\varepsilon I_3} b_{iL}v_i \tag{2.121}$$

$$\sum_j c_{jR}x_j \geq i\sum_{\varepsilon I_1} b_{iL}\mu_i + i\sum_{\varepsilon I_2} b_{iR}\mu_i + i\sum_{\varepsilon I_3} b_{iL}w_i - i\sum_{\varepsilon I_3} b_{iR}v_i \tag{2.122}$$

$$\sum_j a_{ijL}x_j \leq b_{iR} \quad , i\varepsilon I_1, \tag{2.123}$$

$$\mu_i \geq 0 \quad , i\varepsilon I_1, \tag{2.124}$$

$$\sum_j a_{ijR}x_j \geq b_{iL} \quad , i\varepsilon I_2, \tag{2.125}$$

$$\mu_i \leq 0 \quad , i\varepsilon I_2, \tag{2.126}$$

$$\sum_j a_{ijL}x_j \leq b_{iR} \quad , i\varepsilon I_3, \tag{2.127}$$

$$\sum_j a_{ijR}x_j \geq b_{iL} \quad , i\varepsilon I_3, \tag{2.128}$$

$$v_i, w_i \geq 0 \quad , i\varepsilon I_3, \tag{2.129}$$

$$i\sum_{\varepsilon I_1} a_{ijL}\mu_i = i\sum_{\varepsilon I_2} a_{ijR}\mu_i + i\sum_{\varepsilon I_3} a_{ijL}w_i - i\sum_{\varepsilon I_3} a_{ijR}v_i \leq c_{jR}, \quad \forall j, \tag{2.130}$$

$$i\sum_{\varepsilon I_1} a_{ijR}\mu_i + i\sum_{\varepsilon I_2} a_{ijL}\mu_i + i\sum_{\varepsilon I_3} a_{ijR}w_i - i\sum_{\varepsilon I_3} a_{ijL}v_i \geq c_{jL}, \quad \forall j, \tag{2.131}$$

$$x_j \geq 0 \quad , \forall j. \tag{2.132}$$

Then it is true that D contains S.

Proof: It follows from the proof of theorem 2.12 that S is equal to the solution set of

$$\sum_j a_{ij}x_j \leq b_i \quad , i\varepsilon I_1, \tag{2.133}$$

$$\mu_i \geq 0 \quad , i\varepsilon I_1, \tag{2.134}$$

$$\sum_j a_{ij}x_j \geq b_i \quad , i\varepsilon I_2, \tag{2.135}$$

$$\mu_i \leq 0 \quad , i\varepsilon I_2, \tag{2.136}$$

$$\sum_j a_{ij}x_j = b_i \quad , i\varepsilon I_3, \tag{2.137}$$

$$\mu_i \text{ unrestricted} \quad , i\ I_3, \tag{2.138}$$

$$\frac{\partial}{\partial x_j}\{\sum_j c_j x_j - \sum_i \mu_i [\sum_j (a_{ij}x_j - b_i)]\} = 0 \quad , \forall j, \tag{2.139}$$

$$\sum_i \mu_i [\sum_j (a_{ij}x_j - b_i)]\} = 0 \quad , \tag{2.140}$$

$$a_{ijL} \leq a_{ij} \leq a_{ijR} \quad , \forall ij, \tag{2.141}$$

$$b_{iL} \leq b_i \leq b_{iR} \quad , \forall i, \tag{2.142}$$

$$c_{jL} \leq c_j \leq c_{jR} \quad , \forall j, \tag{2.143}$$

$$x_j \geq 0 \quad , \forall j. \tag{2.144}$$

Equation 2.139 may be written

$$c_j - \sum_i a_{ij}\mu_i = 0 \quad , \forall j, \tag{2.145}$$

Also, equation 2.140 may be rearranged to

$$\sum_j x_j \sum_i a_{ij}\mu_i - \sum_i \mu_i b_i = 0, \tag{2.146}$$

and using equation 2.145 this becomes

$$\sum_j c_j x_j - \sum_i \mu_i b_i = 0. \qquad 2.147$$

Let equations 2.146 and 2.147 replace equations 2.139 and 2.140, respectively, in the constraint set. Then definition 8 (solution set of simultaneous interval relations) indicates that the solution set, S, specified by the altered constraint set is also the solution set of

$$\sum_j A_{ij} X_j \leq B_i \quad , i \varepsilon I_1, \qquad 2.148$$

$$U_i \geq [0,0] \quad , i \varepsilon I_1, \qquad 2.149$$

$$\sum_j A_{ij} X_j \geq B_i \quad , i \varepsilon I_2, \qquad 2.150$$

$$U_i \leq [0,0] \quad , i \varepsilon I_2, \qquad 2.151$$

$$\sum_j A_{ij} X_j = B_i \quad , i \varepsilon I_3, \qquad 2.152$$

$$U_i \text{ unrestricted} \quad , i \varepsilon I_3, \qquad 2.153$$

$$\sum_i A_{ij} U_i = C_j \quad , \forall j, \qquad 2.154$$

$$\sum_j C_j X_j = \sum_i B_i U_i \quad , \qquad 2.155$$

$$X_j \geq [0,0] \quad , \forall j, \qquad 2.156$$

All of the above relations are interval separable except for 2.154 and 2.155, which are not unless I_3 is the null set. To surmount this problem let

$$W_i - V_i \equiv U_i \quad , i \varepsilon I_3, \qquad 2.157$$

and

$$W_i, V_i \geq [0,0] \quad , i\varepsilon I_3. \tag{2.158}$$

Then $(W_i - V_i)$ may be substituted for U_i, $i\varepsilon I_3$, in equations 2.154 and 2.155. Theorem 2.4 implies that this does not change the solution set. Thus, equations 2.154 and 2.155 become

$$\sum_{i\varepsilon I_1} A_{ij}U_i + \sum_{i\varepsilon I_2} A_{ij}U_i + \sum_{i\varepsilon I_3} A_{ij}(W_i - V_i) = C_j \quad , \forall j, \tag{2.159}$$

$$\sum_j C_j X_j = \sum_{i\varepsilon I_1} B_i U_i + \sum_{i\varepsilon I_2} B_i U_i + \sum_{i\varepsilon I_3} B_i (W_i - V_i) \quad . \tag{2.160}$$

The distributive law for identical numbers (the B_i and A_{ij}, $i\varepsilon I_3$) and theorem 2.4 imply that the equations

$$\sum_{i\varepsilon I_1} A_{ij}U_i + \sum_{i\varepsilon I_2} A_{ij}U_i + \sum_{i\varepsilon I_3} A_{ij}W_i - \sum_{i\varepsilon I_3} A_{ij}V_i = C_j \quad , \forall j, \tag{2.161}$$

$$\sum_j C_j X_j = \sum_{i\varepsilon I_1} B_i U_i + \sum_{i\varepsilon I_2} B_i U_i + \sum_{i\varepsilon I_3} B_i W_i - \sum_{i\varepsilon I_3} B_i V_i \tag{2.162}$$

may be substituted for equations 2.159 and 2.160, and 2.155 in the constraint set without altering the solution set. The altered constraint set is such that all functions are interval separable by theorems 2.8 through 2.11, and since some functions use the same parameter then theorem 2.7 may be used, to specify the set D, given by relations 2.122 to 2.132. proving the theorem.

Thus, a solution to the ILPP may be determined by successively the series of programs

$$\begin{Bmatrix}\text{maximize}\\ \text{minimize}\end{Bmatrix} \quad x_j \quad , \quad j=1,\ldots,n \ ,$$

subject to: $x \varepsilon D$

where D is specified by the linear functions 2.121 through 2.132. The series of programs are linear programs, the computationally easiest form of CPP to solve.

Interval quadratic programming

The IQPP may be approached in much the same manner as the ILPP. Theorem 2.15 below specifies a set of real vectors containing the interval optimal set of an IQPP. As before this set may be searched for its maximal and minimal values by a series of 2n programs, thus determining a solution to the IQPP.

Theorem 2.15: Let S be the interval optimal set of the following IQPP:

$$\text{Maximize} \quad \sum_j C_j X_j - \sum_j \sum_k Q_{jk} X_j X_k \qquad 2.163$$

subject to:

$$\sum_j A_{ij} X_j \leq B_i \quad , \ i \varepsilon I_1, \qquad 2.164$$

$$\sum_j A_{ij} X_j \geq B_i \quad , \ i \varepsilon I_2, \qquad 2.165$$

$$\sum_j A_{ij} X_j = B_i \quad , \ i \varepsilon I_3 \qquad 2.166$$

$$X_j \geq [0,0] \quad , \ \forall j, \qquad 2.167$$

where it is understood that

$$q_{kj} = q_{jk} \ , \ \forall jk.$$

Let D be the solution set of the following relations:

$$\sum_j a_{ijL}x_j \leq b_{iR} \quad , i\varepsilon I_1, \tag{2.168}$$

$$\mu_i \geq 0 \quad , i\varepsilon I_1, \tag{2.169}$$

$$\sum_j a_{ijR}x_j \geq b_{iL} \quad , i\varepsilon I_2, \tag{2.170}$$

$$\mu_i \leq 0 \quad , i\varepsilon I_2, \tag{2.171}$$

$$\sum_j a_{ijL}x_j \leq b_{iR} \quad , i\varepsilon I_3, \tag{2.172}$$

$$\sum_j a_{ijR}x_j \geq b_{iL} \quad , i\varepsilon I_3, \tag{2.173}$$

$$w_i, v_i \geq 0 \quad , i\varepsilon I_3, \tag{2.174}$$

$$c_{jL} + 2\sum_k q_{kjL}x_k - i\varepsilon\sum_{I_1} a_{ijR}\mu_i - i\varepsilon\sum_{I_2} a_{ijL}\mu_i - i\varepsilon\sum_{I_3} a_{ijR}w_i + i\varepsilon\sum_{I_3} a_{ijL}v_i \leq 0 \quad , \forall j, \tag{2.175}$$

$$c_{jR} + 2\sum_k q_{kjR}x_k - i\varepsilon\sum_{I_1} a_{ijL}\mu_i - i\varepsilon\sum_{I_2} a_{ijR}\mu_i - i\varepsilon\sum_{I_3} a_{ijL}w_L + i\varepsilon\sum_{I_3} a_{ijR}v_i \geq 0 \quad , \forall j, \tag{2.176}$$

$$i\varepsilon\sum_{I_1}\mu_i[\sum_u a_{ijL}x_j - b_{iR}] + i\varepsilon\sum_{I_2}\mu_i[\sum_j a_{ijR}x_j - b_{iL}] + i\varepsilon\sum_{I_3} w_i[\sum_j a_{ijL}x_j - b_{iR}] - i\varepsilon\sum_{I_3} v_i[\sum_j a_{ijR}x_j - b_{iL}] \leq 0, \tag{2.177}$$

$$i\varepsilon\sum_{I_1}\mu_i[\sum_j a_{ijR}x_j - b_{iL}] + i\varepsilon\sum_{I_2}\mu_i[\sum_j a_{ijL}x_j - b_{iR}] + i\varepsilon\sum_{I_3} w_i[\sum_j a_{ijR}x_j - b_{iL}] - i\varepsilon\sum_{I_3} v_i[\sum_j a_{ijL}x_j - b_{iR}] \geq 0, \tag{2.178}$$

$$x_j \geq 0 \quad , \forall j. \tag{2.179}$$

Proof: It follows from theorem 2.12 that S is equal to the solution set of

$$\sum_j a_{ij}x_j \leq b_i \quad , i\varepsilon I_1, \tag{2.180}$$

$$\mu_i \geq 0 \quad , i\varepsilon I_1, \tag{2.181}$$

$$\sum_j a_{ij}x_j \geq b_i \quad , i\varepsilon I_2, \tag{2.182}$$

$$\mu_i \leq 0 \quad , i\varepsilon I_2, \tag{2.183}$$

$$\sum_j a_{ij}x_j = b_i \quad , i\varepsilon I_3, \tag{2.184}$$

$$\mu_i \text{ unrestricted} \quad , i\varepsilon I_3, \tag{2.185}$$

$$\frac{\partial}{\partial x_j}\{\sum_j c_j x_j + \sum_{jk}\sum q_{ik}x_j x_k - \sum_i \mu_i[\sum_j a_{ij}x_j - b]\} = 0, \quad \forall j, \tag{2.186}$$

$$\sum_i u_i[\sum_j a_{ij}x_j - b_i] = 0, \tag{2.187}$$

$$a_{ijL} \leq a_{ij} \leq a_{ijR} \quad , \forall ij, \tag{2.188}$$

$$q_{jkL} \leq q_{jk} \leq q_{jkR} \quad , \forall jk, \tag{2.189}$$

$$c_{jL} \leq c_j \leq c_{jR} \quad , \forall j, \tag{2.190}$$

$$x_j \geq 0 \quad , \forall j \ . \tag{2.191}$$

Equation 2.186 may be written

$$c_j + 2\sum_k q_{jk}x_k - \sum_i a_{ij}\mu_i = 0, \quad \forall j, \tag{2.192}$$

and as before let

$$W_i - V_i \equiv Z_i \quad , i\varepsilon I_3, \tag{2.193}$$

$$W_i - V_i \geq [0,0] \quad , i\varepsilon I_3. \tag{2.194}$$

Then definition 6 (solution set of simultaneous interval relations), theorem 2.3 and theorem 2.4 may be used to obtain the following interval constraint set describing S.

$$\sum_j A_{ij}X_j \leq B_i \quad , i\varepsilon I_1, \tag{2.195}$$

$$U_i \geq [0,0] \quad , i\varepsilon I_1, \tag{2.196}$$

$$\sum_j A_{ij}X_j \geq B_i \quad , i\varepsilon I_2, \tag{2.197}$$

$$U_i \leq [0,0] \quad , i\varepsilon I_3, \tag{2.198}$$

$$\sum_j A_{ij}X_j = B_i \quad , i\varepsilon I_3, \tag{2.199}$$

$$W_i, V_i \geq [0,0] \quad , i\varepsilon I_3, \tag{2.200}$$

$$C_j + 2\sum_k Q_{kj}X_k - \sum_{i\varepsilon I_1} A_{ij}\mu_i - \sum_{i\varepsilon I_2} A_{ij}U_i - \sum_{i\varepsilon I_3} A_{ij}W_i + \sum_{i\varepsilon I_3} A_{ij}V_i = [0,0], \quad \forall j, \tag{2.201}$$

$$\sum_{i\varepsilon I_1} U_i[\sum_j A_{ij}X_j - B_i] + \sum_{i\varepsilon I_2} U_i[\sum_j A_{ij}X_j - B_i] + \sum_{i\varepsilon I_3} W_i[\sum_j A_{ij}X_j - B_i] - \sum_{i\varepsilon I_3} V_i[\sum_j A_{ij}X_j - B_i] = [0,0] \tag{2.202}$$

$$X_j \geq [0,0] \quad , \forall j. \tag{2.203}$$

All of the above relations may be seen to be interval separable, and as in theorem 2.14 a set D specified by relations 2.168 through 2.179 may be obtained. This set will, as before, contain S, thus proving the theorem.

It may be noted that relation 2.177 is redundant and hence may be omitted. However, unlike quadratic programs formulated in the real number system, there appears to be no convenient method, such as restricted entry, to handle constraint 2.178. If relation 2.178 were not present, the series of programs would be the computationally attractive

linear programs. Hence, one possible solution procedure would be to drop restriction 2.178 from the constraint set, thus increasing the size of the feasible region and probably the width of the interval solution. If the resulting solution is of unsuitable width for decision making purposes, then it may be used as a starting point for a series of programs using either relations 2.18 through 2.27 (minimal width solution) or relations 2.168 through 2.179 (the IQPP procedure given above).

Other ICPP topics

Thus far the minimal width and ISCPP algorithms have been developed, as well as special techniques for the ILPP and IQPP. Some remaining aspects of interval convex programming may now be presented. First, it may be noted that maximization problems may be converted to minimization problems merely by changing the sign of the objective function, as is done in the real number system. This follows directly from the definition of an interval optimal set, for this definition is in terms of real valued convex programs. Also, dual interval solutions may be obtained by determining minimal and maximal μ_i (for $i \varepsilon I_1$, I_2) or $w_i - v_i$ (for $i \varepsilon I_3$) instead of x_j. Another relevant observation is that since the Kuhn-Tucker conditions are necessary conditions for the more general mathematical program, it may be seen that the logic of interval convex programming

algorithms may be used for interval mathematical programming. However, for this more general problem it may be practically impossible to obtain minimal width solutions due to the difficulty in testing sufficiency conditions; solutions may be obtained by the algorithms of this section.

Finally, suppose the algorithms do not yield solutions of width sufficiently small enough for decision making purposes. Then either the decision must be made without using this aid, or the widths of parametric values must be decreased. If the second alternative is chosen then sensitivity analyses must be performed on the series of 2n real valued programs to discover crucial parameters. Such analyses may prove to be time consuming and costly. This is the price which is paid for the additional information. In fact, the same comment may be made of interval convex programming.

SUMMARY

This chapter contains a complete development of the ICPP. After a brief introduction, various terms are defined so that a problem statement may be given. Then some possible approaches are presented. These approaches not only provide some familiarization with the ICPP, but may hopefully prove helpful to anyone doing subsequent research. The third section develops topics of interval arithmetic which are necessary to the solution techniques. Particularly,

interval separability is introduced and developed; for this is the basis of relatively more efficient ISCPP and the special ILPP and IQPP algorithms. Then the preceding section used the preliminary definitions and theorems in the development of various solution algorithms.

An attempt was made to structure interval convex programming in an user oriented manner. An analyst may be skilled enough to determine the functional form of a problem, but may be unable to determine parametric values. For example, rarely would one know exact sales prices, availabilities, production rates, and so forth. However, it may well be that expert opinion or statistical estimators can provide interval estimates. Then interval convex programming may be used, and the interval estimates directly incorporated into the problem formulation. Following this an interval convex programming algorithm may be used instead of some type of exhaustive sensitivity analysis. The result of using interval convex programming is an interval solution which may be used, along with other considerations, as a decision aid.

A price, however, must be paid for the information which interval convex programming provides in that an entire series of programs must be solved. This price may be reduced somewhat by continually storing possible starting points for subsequent programs. Moreover separability may be exploited, particularly in the case of ILPPs, to reduce

the computational burden. The following chapter is concerned with methods designed to increase the efficiency of solving linear programs, and hence of interval linear programs.

CHAPTER THREE

ALGORITHMS FOR LINEAR PROGRAMMING

INTRODUCTION

In chapter one it was noted that the most frequently occurring type of mathematical programming problem is the linear programming problem. Thus, it is reasonable to assume that the most frequently occurring type of interval convex programming problem will be the interval linear programming problem which was presented in chapter two. The recommended solution technique for this problem is the algorithm designed specifically to exploit linearity. This procedure involves the solution of a series of 2n linear programs, each of which has more constraints and variables than does the original problem. The topic of this chapter is the development of techniques designed to increase the efficiency of this procedure for solving the interval linear programming problem (ILPP).

One method for reducing the computational burdens of ILPPs is described in the preceding chapter. This technique consists of continually storing points corresponding to the maximal and minimal values of the x_j for which linear programs must still be solved. These points are then used as the starting points of subsequent linear programs. Thus,

the quality of the starting points produced by this technique is roughly proportional to the percentage of the 2n programs which have been solved. Hence, this method is not expected to provide good starting points for the first few programs. For these initial programs it is necessary to develop more general techniques which do not depend upon a knowledge of previous test points.

Three heuristics which were designed to efficiently determine a good starting point for the linear programming problem are presented in this chapter. These methods, however, may all be classified as failures, and hence are but briefly described. Fortunately two interesting results did evolve from these early failures. The first result is a method which may be used to generate linear programs, and some more general programs, which have randomly generated parameters but known solutions. These programs were used in testing the heuristics so that it would not be necessary to determine optimal solutions by the simplex method. The procedure for generating such programs is the topic of the next chapter. The second result is a new algorithm for solving the linear programming problem. The development of this algorithm, the Orthogonal method of linear programming, is the primary topic of this chapter. In this method the direction of a move is computed using the Gram Schmidt (GS) orthogonalization procedure in such a manner that a move may be made not only across constraint surfaces, but also through

the interior of the feasible region.

ORGANIZATION

The three heuristics are briefly presented in the next section, and then the remainder of the chapter is devoted to an explanation of the orthogonal method of linear programming. The presentation of this new method has the following organization. First, the Gram Schmidt (GS) process is reviewed, and next some theorems necessary for the development of the new method are proved. Then following the presentation of the algorithm, its convergence properties are further explained and proved. Finally, a numerical example is given and the chapter is concluded with a chapter summary.

HEURISTICS

General

The Simplex algorithm of George Dantzig and its various revisions and extensions are the most commonly used linear programming algorithms. These methods have been examined quite thoroughly and computationally optimized to a large degree [34, 35]. Fundamentally different approaches, such as that of Fritsch [27] and Rosen [36] which are applicable to more general problems generally have not been able to

exploit linearity as successfully as the simplex method. However, Rosen [36] reports that frequently the projected gradient method converges in fewer iterations than Simplex method does; but due to longer computational time per iteration, net computer run time is longer. Few iterations are necessary since Rosen's method is not restricted to moving along the edges of the simplex; higher computational times are due to frequent matrix inversions. Fritsch's method appears in the above respects to be quite similar to that of Rosen. It seems that one of the basic reasons for the relative computational efficiency of the Simplex method is the fact that it is specifically designed for linear functions.

In light of these experiences an attempt was made to develop heuristic procedures capable of exploiting linearity and of swiftly indicating a good starting point for the simplex algorithm. A commonly chosen starting point for the simplex procedure is the origin. Then by a series of successive matrix inversions one adjacent extreme point or simplex vertex after another is tested for optimality in a manner such that the value of the objective function never decreases. It seems plausible that if a starting point closer to the optimum than the origin is chosen, then generally less iterations or inversions would be required. The heuristics were intended to find such points. The performance of the heuristics was examined by extensive computer aided testing on linear programs with completely random structures but with

solutions determined on an a priori basis. All of the heuristics may be classified as failures. A very brief description of each is given below to fully document the research activities and to prevent the duplication of such fruitless efforts.

Penalty formulation with pattern search

In one attempt to quickly determine a good starting point, a penalty function formulation of the linear programming problem was developed. Then a pattern search was used on the transformed problem in the hope that it would rapidly follow the ridges induced by the simplex. Computational strategies using the norms of the gradients of the linear functions were devised to avoid frequent constraint function evaluations for constraints which were in no danger of being violated. Nonetheless it was found that if the penalty for constraint violation were made large, then the surface of the simplex was closely followed and the frequent changes in the directions of the ridges forced a large number of function evaluations. Thus, a large penalty induced computational ineffectiveness. On the other hand reducing the penalty required less function evaluations, but resulted in very poor starting points. No satisfactory weighting method could be determined, and so the heuristic proved unsuccessful.

Ricocheting directions

Next a method suggested by the ricochet gradient technique was examined. In this technique moves were made along the normal to the objective function until a constraint surface was reached. Then a linear combination of the normals to the objective function and to the constraint was formed. The problem formulation was

maximize (c,x),

subject to:

$$(a_i,x) \geq b_i \quad , \forall i,$$

where

$$|c| = |a_i| = 1 \ , \forall i.$$

Now suppose that a move along c has caused the following equality.

$$(a_k,x) = b_k$$

Realizing that (a_k,c) must be negative for the constraint surface to be encountered, then the direction

$$d = c - \frac{(a_k,c)a_k}{2}$$

is such that

$$(c,d) > 0$$

and

$$(a_k,d) < 0.$$

So a move may be made in direction d to a point centrally located among the surrounding constraints, and then another

gradient move may be made. Unfortunately situations in which oscillatory behavior of the test points occurred were frequently observed. Although this method was unsatisfactory, attempts to modify it led to the eventual development of the new method for linear programming.

Regression analyses

The third heuristic involved regression analyses on hundreds of randomly generated linear programs with known solutions. It was desired to determine an equation to be used for predicting, on the basis of various inner products, quotients, etc., the probability that an inequality constraint would be an equality at the optimum. The highest R^2 values observed were generally in the vicinity of 0.15, and attempts to use the resulting equations generally produced results not significantly different at the .05 level from a purely random choice. Extensive experimentation using a wide range of fitting functions revealed no consistent pattern. Thus, this approach was eventually classified as unsatisfactory, along with the others.

Although the heuristics were classified as failures, one did ultimately evolve into a new method for solving the linear programming problem. In an attempt to remedy the oscillatory nature of the second heuristic, the Gram Schmidt (GS) orthogonalization procedure was examined as a method of determining new search directions. Subsequent investigation

gradually resulted in the new linear programming algorithm. A review of the GS process is given in the following section, after which the algorithm will be developed.

GRAM SCHMIDT PROCEDURE

In this method a series of orthogonal vectors are produced by forming successive linear combinations on a set of vectors [4]. Let

$$V = \{a_i : i=1,\ldots,m+n\}$$

be a set of vectors where n is the dimension of the space. Let a'_k indicate the k'th vector chosen from V. For example, if the first vector chosen is a_4, then

$$a'_1 = a_4.$$

Let T be the current set of the vectors chosen from V. Thus, if currently only a_8 and a_5 have been chosen then

$$T = \{a_8, a_5\}.$$

Let the i'th orthogonal vector determined by the process be denoted as z_i. Then

$$z_1 = a_1$$

and

$$z_i = a_1' - \sum_{k=1}^{i-1} \frac{(z_k, a_i')}{(z_k, z_k)} z_k \quad , \quad 1<i\leq n.$$

Some properties of the process which will be used later are given below.

Property 1: The vectors are orthogonal so that

$$(z_i, z_k) = 0 \quad , \; i \neq k.$$

Property 2: The orthogonality of the z_i imply that if z_i is not null, then

$$(z_i, z_i) = (z_i, a_1').$$

Property 3: Since

$$a_i' = z_i + \sum_{k=1}^{i-1} \frac{(z_k, a_k')}{(z_k, z_k)} z_k \quad , \; 1<i\leq n$$

then for q greater than i

$$(z_q, a_i') = 0 \quad , \; q>i.$$

Property 4: The vector z_k is null if and only if a_k' is linearly dependent on the k-1 vectors in T.

Property 5: If T contains n vectors which are linearly independent, then the z_i span n space.

Property 6: If the a_k' are linearly independent, then if a solution to the equations

$$(a_k', x) = b_k' \quad , \; k=1,\ldots,q,$$

exists where

$$q \leq n,$$

it may be obtained by letting

$$p_1 = \frac{b_1'}{(z_1, z_1)}$$

and

$$p_k = \frac{b_k' - \sum_{j=1}^{k-1} (a_k', z_j) p_j}{(z_k, z_k)}, \quad k=2,\ldots,q.$$

This solution is

$$x = \sum_{i=1}^{q} p_i z_i.$$

This may be seen to follow from property 3.

Property 7: If the a_k' are linearly independent, and if a solution to the equation

$$\sum_{k=1}^{q} w_k a_k' = c$$

exists where

$$q \leq n$$

then it is given by

$$w_q = \frac{(z_q, c)}{(z_q, a_q')}$$

and

$$w_{q-j} = \frac{(z_{q-j}, c) - \sum_{k=0}^{j-1} (z_{q-j}, a_{q-k}') w_{q-k}}{(z_{q-j}, a_{q-j}')}, \quad j=1,\ldots,q-1$$

These various properties will be useful in the following section.

PRELIMINARY THEOREMS

This section presents all of the theorems and results necessary for an understanding of how the new linear programming method works. After this section, a statement of the algorithm is given. Then once the procedure is defined, its convergence is established and a numerical example is given.

The following form of the linear programming problem is assumed:

$$\text{maximize } (c,x), \tag{3.1}$$

subject to:

$$(a_i,x) \geq b_i \quad , \; i=1,\ldots,m+n. \tag{3.2}$$

It is noted that any linear program may be reduced to this form, so there is no loss of generality in using this formulation. Moreover this concise format facilitates the presentation of the method. The surplus variable y_i is defined to be

$$y_i = (a_i,x) - b_i \quad , \; i=1,\ldots,m+n, \tag{3.3}$$

so that all y_i are non-negative if and only if x is feasible. The first theorem may be used to determine the optimality of a test point.

Theorem 3.1: Let there exist a feasible point x. Then x is optimal if and only if there exist scalars w_i such that

$$\sum_{i=1}^{m+n} w_i a_i = c \quad , \tag{3.4}$$

where all w_i are non-positive and all $w_i y_i$ equal zero.
Proof: This is but a restatement of the Kuhn-Tucker conditions for this particular type of problem formulation. It is noted that the w_i may be seen to be dual variables corresponding to the primal formulation given by relations 3.1 and 3.2.

A brief description of the algorithm is necessary before proceeding further. Basically, the way the algorithm works is initially a test point is examined for optimality. If the point is optimal no further moves are made. If not, then a series or cycle of q moves, $\bar{z}_0$ through $\bar{z}_{q-1}$, are made if it is feasible to do so; and then the optimality test is repeated. If it fails, another cycle of moves are made. The attributes of the directions of the moves are presented in the following theorems.

Theorem 3.2 will subsequently be used in proving that the sequence of search directions is such that, under certain assumptions, at least one move may be made each cycle.
Theorem 3.2: Let there exist a feasible point x which is such that no more than n values of y_i are zero. Let a_i be an element of the set T if and only if y_i is zero, and assume that the elements of T are linearly independent. Also, let

$$\sum_{i=1}^{m+n} w_i a_i = c \quad , \tag{3.5}$$

$$w_i y_i = 0 \quad , \; i = 1,\ldots,m+n, \tag{3.6}$$

and

$$w_f > 0. \qquad 3.7$$

Then it is possible to move in a direction $\bar{z}'_q$ where

$$(c,\bar{z}'_q) > 0.$$

Proof: Let T have q elements where

$$q \leq n ,$$

and let the elements be denoted $a'_1,\ldots,a'_q$, where

$$a'_q = a_f.$$

The elements of T are linearly independent, so that by property 4 of the Gram Schmidt process the following q non-null vectors may be formed

$$z_k = a'_k - \sum_{i=0}^{k-1} \frac{(z_i,a'_k)}{(z_i,z_i)} z_i \quad , k=1,\ldots,q. \qquad 3.8$$

Where for convenience z_o is the null vector. It will now be shown that the vector z_q may be chosen to be $\bar{z}'_q$. First note that by property 3 of the GS process it is true that

$$(a'_k,z_q) = 0 \quad , k=1,\ldots,q-1.$$

Also, property 2 indicates that (z_q,a_q) equals (z_q,z_q) and is hence positive. Therefore, a move in direction z_q is feasible because all y'_k remain zero except for y'_q which increases. Note that if more than q values of y_i were zero, then z_q might not be feasible. Now it will be shown that a move in direction z_q increases the value of the objective

function. Equations 3.3 and 3.5 imply that

$$\sum_{i=1}^{m+n} w_i(a_i,x) = \sum_{i=1}^{m+n} w_i b_i + \sum_{i=1}^{m+n} w_i y_i = (c,x) \qquad 3.9$$

Now all $w_i y_i$ are zero. Also, it is known that

$$y_i = 0 \quad , a_i \varepsilon T, \qquad 3.10$$

and since

$$y_i > 0 \quad , a_i \not\varepsilon T, \qquad 3.11$$

then

$$w_i = 0 \quad , a_i \not\varepsilon T. \qquad 3.12$$

Let

$$v = x + d\bar{z}_q$$

where d is a positive scalar. Then at v, relations 3.10 and 3.12 are true with the exception that y_f has increased. The value of w_f is positive; and for sufficiently small d, v is feasible so relation 3.11 is true. Thus, letting

$$z'_q = z_q \qquad 3.13$$

it is seen that at least one feasible direction exists such that the value of the objective function increases by amount $w_f y_f$, proving the theorem.

The next theorem defines a set of moves which increase the value of the objective function, and it also provides a computationally efficient method for calculating them. Then theorems 3.2 and 3.3 will both be used in the proof of theorem 3.4 which establishes under certain assumptions,

that the value of the objective function will be increased every cycle.

Theorem 3.3: Let z_0 be the null vector and let $\bar{z}_0$ equal c. Also let

$$\bar{z}_k = c - \sum_{i=0}^{k} \frac{(z_i,c)}{(z_i,z_i)} z_i \quad , k=1,\ldots,q, \tag{3.14}$$

where

$$q < n, \tag{3.15}$$

and

$$z_i = a_i' - \sum_{j=0}^{i-1} \frac{(z_j,a_i')}{(z_j,z_j)} z_j \quad , i=1,\ldots,q. \tag{3.16}$$

Let the vectors a_i' be the only elements of the set T and assume that they are linearly independent. Further, let c be linearly independent of the elements of T. Then it is true that

$$(c,\bar{z}_k) > 0 \quad , k=1,\ldots,q, \tag{3.17}$$

and

$$\bar{z}_k = \bar{z}_{k-1} - \frac{(c,z_k)}{(z_k,z_k)} z_k \quad , k=1,\ldots,q. \tag{3.18}$$

Proof: The linear independence assumptions imply the $\bar{z}_k$ are not null by property 4 of the GS process, and so property 2 indicates that

$$(\bar{z}_k,c) = (\bar{z}_k,\bar{z}_k)$$

and is hence positive. To prove equation 3.18 merely form the difference

$$\bar{z}_k - \bar{z}_{k-1} \quad , \; k=1,\ldots,q,$$

and express the quantities using equation 3.14, so that

$$\bar{z}_k - \bar{z}_{k-1} = \frac{(z_k,c)}{(z_k,z_k)} z_k,$$

proving the theorem.

It may now be proved that the moves $\bar{z}_0$, through $\bar{z}_{q-1}$ are such that the value of the objective function is increases every cycle. The proof requires that certain assumptions be made regarding the test points. The purpose of these assumptions is to state the conditions under which a move increasing the value of the objective function will definitely be made. The absence of these conditions in no way implies that such a move may not be made. These conditions will be assumed throughout a large part of the development of the algorithm. Ultimately it will be shown that these conditions are always satisfied by the algorithm, using perturbation if necessary.

Theorem 3.4: Let there exist a feasible point x such that there are exactly q values of y_i', y_1' through y_q' which are zero, and a_1' through a_q' which are linearly independent and such that

$$\sum_{k=1}^{q} a_k' w_k' = c$$

where the subscripts have been arranged so that

$$w_q' > 0 \; .$$

Let a_1' through a_q' be the only elements of the set T, and moreover let c be linearly independent of any q-1 elements of T. Let $\bar{z}_0$ equal c and z_0 be the null vector. Then one of the directions

$$\bar{z}_k = c - \sum_{i=0}^{k} \frac{(z_i,c)}{(z_i,z_i)} z_i \quad , k=1,\ldots,q-1$$

is feasible where

$$z_i = a_i' - \sum_{j=0}^{i-1} \frac{(z_j,a_i')}{(z_j,z_j)} z_j \quad , i=1,\ldots,q-1.$$

Proof: Suppose that $\bar{z}_0$ through $\bar{z}_{q-2}$ are not feasible; for if one were, there, then the theorem would be proved. Now from theorem 3.2, it is known that

$$z_q = a_q' - \sum_{i=0}^{q-1} \frac{(z_i,a_k')}{(z_i,z_i)} z_i \qquad 3.19$$

is a feasible direction such that

$$(c,z_q) > 0 \qquad 3.20$$

Also, it is known from theorem 3.3 and from the assumption that c is linearly independent of any q-1 elements of T that

$$\bar{z}_{q-1} = c - \sum_{i=1}^{q-1} \frac{(z_i,c)}{(z_i,z_i)} z_i \qquad 3.21$$

is such that

$$(c,\bar{z}_{q-1}) > 0. \qquad 3.22$$

Multiply equation 3.21 by z_q to obtain

$$(z_q,\bar{z}_{q-1}) = (z_q,c) \qquad 3.23$$

so that

$$(z_q, \bar{z}_{q-1}) > 0.$$

Then multiply through equation 3.19 by $\bar{z}_{q-1}$ to obtain

$$(\bar{z}_{q-1}, z_q) = (\bar{z}_{q-1}, a_k'). \qquad 3.24$$

Thus, relations 3.23 and 3.24 imply that

$$(\bar{z}_{q-1}, a_k') > 0$$

Now since $(c, \bar{z}_{q-1})$ is positive and property 2 of the GS process indicates that

$$(\bar{z}_{q-1}, a_k') = 0 \quad , k < q-1,$$

then $\bar{z}_{q-1}$ is a feasible direction such that the value of the objective function increases, proving the theorem.

Thus, when a non-optimal test point is examined for optimality, it is only necessary to choose any a_f such that w_f is positive. Then by not permitting a_f to enter the GS process until a move is made, it is assured that one of the directions $\bar{z}_0$ through $\bar{z}_{q-1}$ will be feasible, given the assumptions of theorem 3.4.

Some additional consequences of performing moves in the directions $\bar{z}_k$ are proved by the following corollaries.

Corollary 3.1: Let a_1' through a_q' be linearly independent. Let $\bar{z}_0$ equal c and z_0 equal the null vector so that

$$z_i = a_i' - \sum_{j=0}^{i-1} \frac{(z_j, a_i')}{(z_j, z_j)} z_j \quad , i=1,\ldots,q,$$

and

$$\bar{z}_k = c - \sum_{i=0}^{k} \frac{(z_i,c)}{(z_i,z_i)} z_i \quad , k=1,\ldots,q.$$

If and only if

$$c = \sum_{i=1}^{k} a_i' w_i, \tag{3.25}$$

then $\bar{z}_k$ is the null vector.

Proof: Equation 3.25 implies that there is a linear dependence between c and the a_i'. Thus, property 4 of the GS process may be used to prove the corollary. Note that by property 5 of the GS process, if q equals n, then $\bar{z}_n$ must be null. Thus, the maximal set of possible moves for any one cycle is $\bar{z}_0$ through $\bar{z}_{n-1}$.

The following two corollaries may be used in determining the feasibility of a move in direction $\bar{z}_k$.

Corollary 3.2: It is true that

$$(a_i,\bar{z}_k) = (a_i,\bar{z}_{k-1}) - \frac{(c,z_k)}{(z_k,z_k)} (a_i,z_k) \quad , \forall i,k. \tag{3.26}$$

where

$$q \leq n.$$

Proof: Equation 3.26 follows directly from equation 3.18.

Corollary 3.3: A value of $(a_r,\bar{z}_k)$ is zero if a_r is linearly dependent on the vectors a_1' through a_k' which are used in the calculation of $\bar{z}_k$.

Proof: Suppose that a_r is linearly dependent on a_1' through

a_k'. Then property 4 of the GS process may be used to obtain

$$0 = a_r - \sum_{i=1}^{k} \frac{(z_i, a_r)}{(z_i, z_i)} z_i. \qquad 3.26$$

The vector $\bar{z}_k$ is orthogonal to z_1 through z_k so that multiplying both sides of equation 3.21 by $\bar{z}_k$ yields

$$0 = (\bar{z}_k, a_r) + 0,$$

which proves the corollary.

Corollary 3.2 provides a computationally efficient method for determining the inner products $(a_i, \bar{z}_k)$ which are used in determining d such that a new point $x+d\bar{z}_k$ is feasible. Corollary 3.3 indicates that it is not necessary to compute $(a_i, \bar{z}_k)$ if a_i has been included in T, for of necessity it is linearly dependent on itself so that the inner product is zero. More importantly, however, it will be seen shortly that corollary 3.3 implies that the set T used in the computation of the $\bar{z}_k$ will always include linearly independent vectors, an assumption made in several theorems.

Thus, the search directions are given by the $\bar{z}_k$. To complete the specification of the algorithm it is necessary to determine the distances associated with these directions. The following theorem may be used for this purpose.

Theorem 3.5: Let the current test point x be feasible, and let there exist at least one negative $(a_i, \bar{z}_k)$. Then the maximal multiplier, d_s, of $\bar{z}_k$ such that the point

$$v = x + d_s \bar{z}_k \qquad 3.27$$

is feasible is given by

$$d_s = \min_i \{d_i = - \frac{y_i}{(a_i, \bar{z}_k)} : (a_i, \bar{z}_k) < 0\}. \qquad 3.28$$

Proof: The point v is feasible if

$$(a_i, v) \geq b_i \quad , \forall i.$$

Now the point x is feasible so that

$$y_i = (a_i, x) - b_i \geq 0 \quad , \forall i.$$

Thus, v is feasible if d is such that

$$(a_i, x + d\bar{z}_k) \geq b_i \quad , \forall i,$$

or

$$y_i + d(a_i, \bar{z}_k) \geq 0 \quad , \forall i. \qquad 3.29$$

Hence d is bounded only by those $(a_i, \bar{z}_k)$ which are negative, and this bound is

$$- \frac{y_i}{(a_i, \bar{z}_k)} \geq d.$$

Thus, relation 3.28 merely chooses the least upper bound on d, proving the theorem.

Theorems 3.4 and 3.5 may be used to determine which vector, a*, is to be entered into the GS process so that the next z_k may be computed. The manner in which this is done is explained by the following corollary.

Corollary 3.4: Let the current test point x be such that d_s exists where

$$d_s = \min_i \{d_i = - \frac{y_i}{(a_i, \bar{z}_k)} : (a_i, \bar{z}_k) < 0\}.$$

The next vector, a*, to enter the GS process may be determined by considering the following cases.

Case 1: The current value of d_s is positive. If this is the case, let the new test point be

$$v = x + d_s \bar{z}_k,$$

and let

$$a^* = a_s.$$

This will force y_s to zero, but by corollary 3.2 subsequent $\bar{z}_{k+j}$ will not decrease y_s further.

Case 2: A value of d_s previously computed during the current cycle was positive. If this has occurred then at least one move has already been made during the current cycle, so for the current value of s let

$$a^* = a_s,$$

and make a move if it is feasible to do so.

Case 3: No value of d_s thus far computed for the current cycle has been positive, including the present. If the current cycle is the first cycle, that is, if no test for optimality have been made yet, then let

$$a^* = a_s$$

and continue with the GS process. Otherwise it may be assumed that the test point at the end of the previous cycle was not optimal and that

$$w_f > 0.$$

If theorem 3.4 is to be used in order to insure that, under suitable assumptions, a move may be made during the current cycle, then care must be exercised regarding the selection of a*. If $(a_f,\bar{z}_k)$ is not negative, then it is permissable to let

$$a^* = a_s.$$

However, if $(a_f,\bar{z}_k)$ is negative, then d_f is zero. Nonetheless, if any other value of y_i is zero and is such that $(a_i,\bar{z}_k)$ is not equal to zero, then a* should not be set equal to a_f. If $(a_i,\bar{z}_k)$ is zero, then a_i is linearly dependent on a_1' through a_k'. Hence it will also be linearly dependent on a_1' through a_k' and a_{k+1}', and so $(a_i,\bar{z}_{k+1})$ and subsequent $(a_i,\bar{z}_{k+j})$ will also be zero. Thus, not permitting a_i to be a* if $(a_i,\bar{z}_k)$ is zero will never prevent a move from being made. It seems reasonable to determine

$$(a_v,\bar{z}_k) = \min_i \{(a_i,\bar{z}_k) : y_i = 0, \quad a_i \neq a_f, \quad (a_i,\bar{z}_k) \neq 0\},$$

and let

$$a^* = a_v,$$

although it is merely necessary that y_i be equal to zero and that $(a_i,\bar{z}_k)$ is not zero. If $(a_v,\bar{z}_k)$ does not exist or if only y_f is zero, then let

$$a^* = a_f.$$

Regardless of the choice of a*, since no move may be made in direction $\bar{z}_k$, then d_s remains zero.

Proof: This corollary merely explains how theorems 3.4 and 3.5 may be used to determine the vector to be entered into the GS process. Case 3 prevents a_f from entering prematurely. Thus, if the test point at the beginning of the cycle is such that the conditions of 3.4 are met, then it is assured that a move may be made during the cycle. It is noted that theorem 3.4 does not require that all w_i be calculated for the test point at the end of the last cycle; any one positive w_f is sufficient to both ascertain that the point is not optimal and to appropriately restrict the entry of a_f so that a move may be made.

Now suppose a situation exists in which it is not possible to use theorem 3.5 to compute the value of d_s, so that corollary 3.4 may not be used. If there is no negative $(a_i,\bar{z}_k)$ then theorem 3.5 is not applicable, but the following corollary may be used.

Corollary 3.5: If there exists a non-null direction $\bar{z}_k$ such that no $(a_i,\bar{z}_k)$ is negative, then the optimal feasible value of the objective function is unbounded.

Proof: The directions $\bar{z}_k$ are such that $(c,\bar{z}_k)$ is positive for non-null $\bar{z}_k$. The value of d is merely restricted by equation 3.29 so that

$$y_i + d(a_i,\bar{z}_k) \geq b_i \quad , \forall i.$$

Thus, if no $(a_i,\bar{z}_k)$ is negative, then d may be increased without limit so that $x+d\bar{z}_k$ is feasible and $(c,x+d\bar{z}_k)$ is

unbounded, proving the corollary.

Corollary 3.5 shows that if corollary 3.4 may not be used to determine a*, then the solution is unbounded. It will now if shown that if a* is determined using corollary 3.4 then the elements of T, the vectors currently in the GS process, must be linearly independent.

Corollary 3.6: If a vector a* to be used to compute z_{k+1} is chosen as in corollary 3.4, then the vectors a_1' through a_k' and a* are linearly independent.

Proof: This follows from corollary 3.4, for if a* were linearly dependent, then $(a^*,\bar{z}_k)$ would be zero by corollary 3.3. Hence either a* would not be considered in the evaluation of d_s in equation 3.28 in which only negative $(a_i,\bar{z}_k)$ need to be examined, or it would be prevented from entering in case 3.

Corollary 3.6 implies that a move direction does not exist only when $\bar{z}_k$ is null, a condition which the algorithm is designed to handle. Also, move directions and distances have been specified. Thus, it is not necessary to prove any more theorems prior to the presentation of the algorithm. There are, however, two more topics which should be briefly introduced now. After the presentation of the algorithm they will be examined more fully. The first topic concerns the case in which it is possible to write

$$\sum_{k=1}^{q \leq n} w_k' a_k' = c \quad ,$$

but there are more than q values of y_i which are equal to zero. The development of this section often assumed that such a situation does not exist. This case corresponds to the situation in which there is a tie for the leaving variable in the simplex method. If it is assumed that such a situation can exist neither the Simplex method nor the Orthogonal method can be proved to converge. Thus, in the development of both methods, it is initially assumed that this situation will never exist; then later special techniques are devised to prevent any looping which might occur as a result of such degenerate points. In the Orthogonal method, the technique of perturbation is used to prevent looping in the algorithm, due to its computational simplicity. The second topic to be introduced is the method to be used for obtaining an initial feasible starting point. The algorithm itself may be used to obtain such a point by repeatedly maximizing the value of any violated constraint. This commonly used technique will be stated more explicitly following the presentation of the algorithm and the proofs of its properties.

THE ORTHOGONAL METHOD OF LINEAR PROGRAMMING

General description

The method operates in cycles. For each cycle, search directions $\bar{z}_0$ through $\bar{z}_q$ are iteratively evaluated, and an attempt is made to move in those directions. If $(c,\bar{z}_k)$ is zero, then c is linearly dependent on the a_1' through a_k' which are in the GS process for the current cycle, and the test point is checked for optimality. If it is not optimal, there is a positive w_f; and so a_f will not be permitted to enter the GS process during the next cycle until a move has been made or can be made. At the end of each cycle, if a condition exists such that a move might not be made during the next cycle, then perturbation is performed.

The logic for each cycle is identical, except that for the initial cycle there is no positive w_f. Thus, to specify the algorithm it is only necessary to describe one cycle, and to specify the initial conditions. Let the set F either contain the vector a_f or be the null set. Initially it is the null set. Let r be the number of the current cycle, and initialize it to zero. The set T_r consists of those vectors currently in the GS process during cycle r. The vector $\bar{z}_0$ is equal to c, and z_0 is the null vector. All other symbols which will be used have previously been identified and will again be defined in the course of the presentation of

the algorithm. The logic flowchart of the method is given in Figure 5 on the following page, and the details of the algorithm are given below.

The description of the algorithm will refer to the corollaries and theorems of the preceeding section. This procedure serves two purposes. Not only does it validate the steps of the algorithm; but also it indicates the specific use and purpose of each theorem and corollary, thus in a sense completing their presentation. The proofs of algorithmic properties, such as convergence, is the topic of the next section.

Step 1: Begin cycle r.

Increment r, set k equal to zero, and let T_r be the null set. Then go to step 4 and attempt to move in the gradient direction, $\bar{z}_0$.

Step 2: Begin iteration k of cycle r.

Increment k and let a* be equal to a'_k, the new element of T_r. Then compute

$$z_k = a'_k - \sum_{i=1}^{k-1} \frac{(z_i, a'_k)}{(z_i, z_i)} z_i.$$

By corollary 3.6 this vector will never be null. Evaluate

$$(z_k, z_k) = \sum_{j=1}^{n} z_{k_j}^2.$$

If k is equal to n, then compute (c, z_n) and go to step 6 to see if an optimal solution has been reached; otherwise

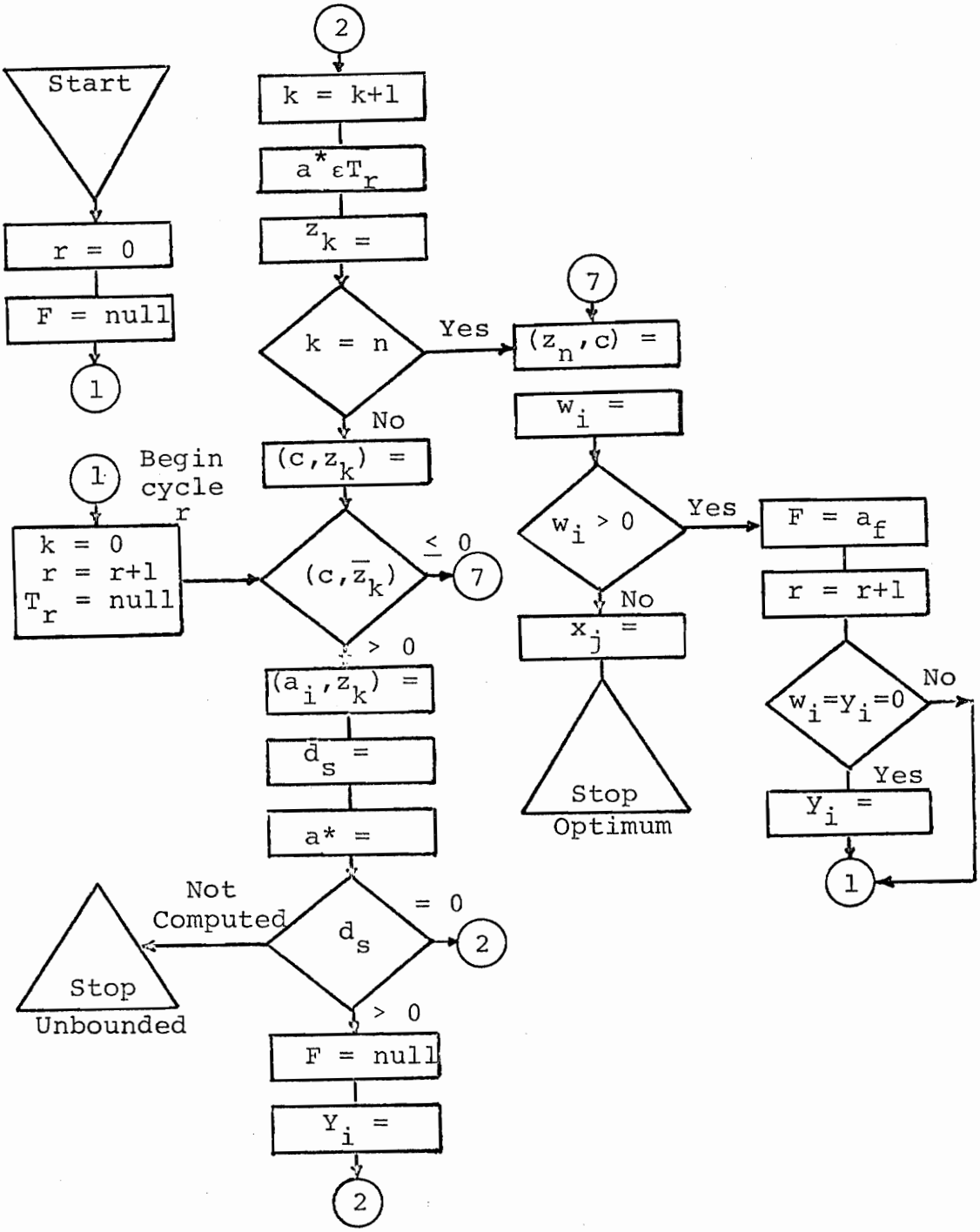

Figure 5. Orthogonal method logic flowchart.

continue.

Step 3: Check the linear independence of T_r and c.

Compute (c,z_k) using the formula

$$(c,z_k) = (c,a_k') - \sum_{i=1}^{k-1} \frac{(z_i,a_k')}{(z_i,z_i)} (z_i,c),$$

and use corollary 3.2 to evaluate

$$(c,\bar{z}_k) = (c,\bar{z}_{k-1}) - \frac{(c,z_k)^2}{(z_k,z_k)} .$$

It may be seen from theorem 3.3 and corollary 3.1 that $(c,\bar{z}_k)$ is non-negative, and is zero only if c is linearly dependent on the elements of T_r. If $(c,\bar{z}_k)$ is positive go to step 4; if it is zero, then go to step 6 to see if the test point is optimal.

Step 4: Determine the next vector to enter T_r.

Corollaries 3.2 and 3.3 may be used to compute the $(a_i,\bar{z}_k)$:

$$(a_i,\bar{z}_k) = (a_i,\bar{z}_{k-1}) - \frac{(c,z_k)}{(z_k,z_k)} (z_k,a_i) \quad , \; i \notin T_r,$$

and

$$(a_i,\bar{z}_k) = 0 \quad , \; i \in T_r.$$

where

$$(a_i,z_k) = (a_i,a_k') - \sum_{j=0}^{k} \frac{(z_j,a_k')}{(z_j,z_j)} (z_j,a_i) \quad , \; k>0.$$

Then determine

$$d_s = \min_i \{d_i = - \frac{y_i}{(a_i, \bar{z}_k)} : \quad (a_i, \bar{z}_k) < 0\}$$

where by theorem 3.5 $d_s \bar{z}_k$ is the maximal feasible move which may be made along $\bar{z}_k$. If no (a_i, z) is negative, then it is known from corollary 3.5 that the solution is unbounded. Corollary 3.4 implies that a^*, the next element of T_r, may be determined by considering the following two cases.

Case 1: Either d_s is positive or F is null. If this occurs then a^* is equal to a_s. If d_s is positive then go to step 5; if it is zero then go to step 2.

Case 2: The set F contains the vector a_f and d_s is zero. If only y_f is equal to zero, then a^* is a_f. Otherwise, let a^* equal a_v where

$$(a_v, \bar{z}_k) = \min_i \{(a_i, \bar{z}_k) : y_i = 0, \quad a_i \neq a_f, \quad (a_i, \bar{z}_k) \neq 0\}.$$

If a_v does not exist, then let a^* equal a_f. No move is made, so go to step 2 and begin another iteration.

Step 5: Update F and the y_i.

This step is entered when it is possible to make a move, so let F be the null set. The $y_{i,new}$ corresponding to a^* and the elements of T_r are zero. The other values may be updated by the formula

$$y_{i,new} = y_{i,old} + d_s(a_i, \bar{z}_k).$$

Now go to step 2 and begin another cycle.

Step 6: See if the optimum has been reached.

Corollary 3.1 implies that this step is entered as soon as c may be expressed as a linear combination of the a'_k. Use property 7 of the GS process to evaluate the coefficients of the a'_k in the expression

$$\sum_{k=1}^{q} w'_k a'_k = c.$$

They are

$$w'_q = \frac{(z_q, c)}{(z_q, z_q)}$$

and

$$w'_{q-j} = (z_{q-j}, c) - \sum_{k=0}^{j-1} \frac{(z_{q-j}, a'_{q-k}) w_{q-k}}{(z_{q-j}, z_{q-j})} \quad , \; j=1,\ldots,q-1.$$

All other w_i may be assigned the value of zero. If the coefficient of a_f is positive then let F equal a_f, and go to step 7 and see if perturbation is necessary. If all w'_k are non-positive, then the test point is optimal, and the optimal dual solution is the set of w_i computed above. The optimal primal solution x* may be determined from the surplus variables if all x_j are sign restricted. Otherwise, property 6 of the GS process may be used to compute

$$p_1 = \frac{b'_1}{(z_1, z_1)}$$

and

$$p_k = \frac{b'_k - \sum_{j=1}^{k-1} (a'_k, z_j) p_j}{(z_k, z_k)} \quad ,k=2\ldots,q.$$

so that

$$x^* = \sum_{i=1}^{q} p_i z_i .$$

Step 7: Perturb the y_i, if necessary.

To insure that a move may be made during the next cycle, let g_r be some very small positive number. Then let $y_{i,new}$ equal g_r if both the current values of y_i and w_i are zero. It is sufficient that g_r is less than the increase in the objective function during the cycle just completed divided by n times the absolute value of the most negative w_i observed thus far. However, it should be chosen as small as is computationally possible. Now go to step 1 and begin another cycle.

ALGORITHMIC PROPERTIES AND PROOFS

In the preceding three sections the Gram Schmidt process was reviewed, various preliminary theorems and corollaries were proved, and then these results were used in the statement of the algorithm. The computational aspects of the new method were thus developed and proved in the two sections which preceded the explicit statement of the algorithm. It is of particular interest that the algorithm will always produce at least one move each cycle if the conditions of theorem 3.4 are met, and that this move will increase the value of the objective function. This observation naturally leads to the question of

convergence. The convergence theorems of this section prove that the algorithm either converges to an optimum in a finite number of cycles or detects an unbounded solution, assuming that it is possible to make a move each cycle. Then the perturbation theorems show that a move may be made every cycle, and ultimately prove that the algorithm consists of a finite number of cycles after which either an unbounded condition or an optimum is detected.

Convergence proofs

The first theorem shows that unbounded solutions can always be detected by the algorithm. The second theorem proves that if the optimal solution is bounded, then the algorithm will converge to the optimum in a finite number of steps, assuming that a move may be made each cycle.

Theorem 3.6: Assume that at least one $\bar{z}_k$ is feasible every cycle unless an optimum has been reached. Let x_r be a non-optimal test point at the end of cycle r. Then the point x_{r+1} may not be determined by the algorithm only if the solution is unbounded.

Proof: The only time a move is not made, although a feasible direction exists, is when d_s cannot be computed. This occurs only when all $(a_i, \bar{z}_k)$ are non-negative for all $a_i \notin T_r$. Since $(a_i, \bar{z}_k)$ is zero for all $a_i \in T_r$, then it is true that $(a_i, \bar{z}_k)$ is non-negative for all a_i. Thus a feasible direction exists and an unbounded move may be made, and it

has previously been shown that a move in the direction $\bar{z}_k$ will increase the value of the objective function. Accordingly, if the algorithm fails to produce a move, then an unbounded solution exists; a condition detected by the algorithm.

The finite nature of the algorithm under the appropriate assumptions may now be established.

Theorem 3.7: Assume that it is possible to make at least one move every cycle, unless the optimum has been reached or an unbounded solution exists. If a bounded solution exists, the algorithm will converge to it in a finite number of steps. If the solution is unbounded, the algorithm will detect this condition.

Proof: It will first be shown that there are a finite number of end of cycle test points. Assume that the optimum has not been reached, and consider two non-optimal end of cycle test points, x_r and x_{r+j}. The algorithm is such that all moves increase the value of the objective function; thus

$$(c,x_{r+j}) > (c,x_r).$$

Now corresponding to these points are the sets T_r and T_{r+j}. It is necessary to prove that these sets are not identical. The algorithm is such that

$$(a'_k,x_r) = b'_k \quad , \quad a'_k \varepsilon T_r, \tag{3.30}$$

and

$$(a'_j,x_{r+j}) = b'_j \quad , \quad a'_j \varepsilon T_{r+j}. \tag{3.31}$$

Let equations

$$\sum_k w'_k a'_k = c \qquad 3.32$$

and

$$\sum_j w''_j a'_j = c, \qquad 3.33$$

correspond to the equations 3.30 and 3.31, respectively. Then

$$\sum_k w'_k (a'_k, x_r) = (c, x_r) = \sum_k w'_k b'_k$$

and

$$\sum_j w''_j (a'_j, x_{r+j}) = (c, x_{r+j}) = \sum_j w''_j b'_j .$$

If T_r and T_{r+j} were not distinct then w'_k and w''_j may be chosen to be identical so that

$$\sum_k w'_k b'_k = \sum_j w''_j b'_j$$

would be true, implying that

$$(c, x_r) = (c, x_{r+j}) .$$

However, this is known to be false, so it must be true that T_r and T_{r+j} are distinct. Since these arbitrary two sets may not be equal, then each end of cycle set T_s must be unique. Since there are m+n different values of a_i, then the number of distinct subsets with n or less elements of this collection is finite. Thus, the number of end of cycle test points is finite. Now a move may be made every cycle unless the optimum is reached. Therefore, if a test point is not optimal and move cannot be made, theorem 3.6 shows

that the algorithm will detect the unbounded condition. Otherwise, the point must be optimal. This proves that under the assumption of the theorem statement the algorithm will converge to the optimum in a finite number of steps or will detect an unbounded solution.

Perturbation and finiteness

It will be proved in the following theorems that the technique of perturbation may be used not only to force the conditions of 3.4 to occur so that it is always possible to make a move, but also to insure that an optimal point will be detected as such. It is also proved that at most a finite number of perturbations may be performed. Then it is shown that these proofs imply that the algorithm consists of a finite number of cycles.

The perturbation technique used is stated in step 7 of the algorithm statement. At the end of each cycle the values of the y_i are examined. If a value of y_i is zero and the value of the corresponding w_i is also zero, then perturbation is performed; otherwise it is not. Perturbation consists of subtracting some very small positive quantity from the b_i for which both w_i and y_i are zero. This causes the new value of the perturbed y_i to the very small positive quantity.

The following theorem proves that perturbation forces the conditions of theorem 3.4 to be true.

Theorem 3.8: Let there exist a feasible end of cycle test point x such that at least one w_i is positive but the conditions of theorem 3.4 are not met. Let a small positive quantity g be added to all y_i which are equal to zero and such that w_i is also equal to zero. Then the conditions of theorem 3.4 will be true for the perturbed problem.

Proof: A test point is an end of cycle point only if c may be expressed as a linear combination of the a_k'. This may occur after either n iterations or if a $\bar{z}_k$ is null. Suppose that after perturbation exactly q values of y_k' are zero. Then the w_k' corresponding to these y_k' are by construction such that

$$\sum_{k=1}^{q} w_k' a_k' = c \qquad 3.34$$

where no w_k' is zero. Further, since the w_k' are not zero, the a_k' were included in the Gram Schmidt process during the cycle and hence must be linearly independent by corollary 3.6. Now if c were linearly dependent on any group of q-1 values of a_k', then it would be possible to write

$$\sum_{k=1}^{q} v_k' a_k' = c, \qquad 3.35$$

where at least one v_k' and w_k' is not equal. For example, the value of v_r' can be set equal to zero. Then subtracting equation 3.35 from equation 3.34 results in

$$\sum_{\substack{k=1 \\ k\neq r}}^{q} (w_k'-v_k)a_k' + w_r'a_r' = 0. \qquad 3.36$$

If equation 3.36 were true, then the a_k' would not be linearly independent. Thus, c is linearly independent of any group of q-1 values of a_k', and the conditions of theorem 3.4 are all satisfied. Note also that the above argument indicates that when the conditions of theorem 3.4 are met, then the resultant w_k' are unique. This concludes the proof.

Thus, it is seen that perturbation causes the conditions of theorem 3.4 to be met for non-optimal end of the cycle points implying that at least one move may be made every cycle. Corollary 3.7 will establish that the algorithm is capable of identifying optimal points.

Corollary 3.7: Let there exist a bounded solution. If it is not possible to move from an end of cycle test point, then the point is an optimum and the algorithm will detect it as such.

Proof: If there is a positive w_i, then perturbation may be used to force the conditions of theorem 3.4 to occur, so that a move may be made from the current point. Now if no move may be made then the conditions of theorem 3.4 cannot be true. However, it may be seen from theorem 3.8 that perturbation always forces all of the conditions of theorem 3.4 to be true with but one exception. Perturbation does not change any value of the w_i. Hence, if the conditions of

theorem 3.4 are not met then it is because no w_i is positive. Thus, the point is an optimum and the algorithm will detect it as such before beginning another cycle. It may be alternatively seen that perturbation is capable of forcing unique w'_k to be associated with any perturbed point, thus allowing theorem 3.1 to be used.

It will now be shown that the number of perturbations which may be performed are finite, and this will lead to a final theorem which establishes that the number of cycles performed by the algorithm is finite.

Theorem 3.9: There exist values of the positive numbers g_t sufficiently small so that the number of perturbations which may be performed by the algorithm is finite.

Proof: Consider an end of cycle point x_r before perturbation, if necessary, is performed at the end of cycle r+j. The algorithm is such that for the end of cycle point x_r it is true that

$$\sum_i w_{ri} a_i = c, \qquad 3.37$$

$$y_{ri} = 0 \quad , \quad a_i \epsilon T_r, \qquad 3.38$$

$$w_{ri} = 0 \quad , \quad a_i \not\epsilon T_r, \qquad 3.39$$

where the additional subscript r indicates the cycle number; also

$$(a_i, x_r) = y_{ri} + b_i \quad , \quad \forall i. \qquad 3.40$$

Then it is follows from equations 3.38 through 3.40 that

$$\sum_i w_{ri}(a_i,x_r) = \sum_i w_{ri}b_i$$

and from equation 3.37 that

$$\sum_i w_{ri}(a_i,x_r) = (c,x_r)$$

and hence it is true that

$$(c,x_r) = \sum_i w_{ri}b_i.$$

Let t equal r+j, and suppose that perturbation is necessary at the end of cycle t. Let p_{ti} equal one if the value of y_i is increased by g_t and equal zero otherwise. Then perturbation changes the constraint set to

$$(a_i,x) + y_i = b_i - p_{ti}g_t \quad , \forall i. \qquad 3.43$$

Now suppose that for the altered problem, there still exists a feasible solution x_r' such that

$$y'_{ri} = 0 \quad , a_i \varepsilon T_r,$$

and

$$(a_i x'_r) = y'_{ri} + b_i - p_{ti}g_t \quad , \forall i.$$

Then as before it is true that

$$(c,x'_r) = \sum_i w_{ri}(b_i - p_{ti}g_t)$$

or

$$(c,x'_r) = \sum_i w_{ri}b_i - \sum_i w_{ri}p_{ti}g_t. \qquad 3.44$$

Now it is true that since moves have been performed

$$(c,x_t) > (c,x_r).$$

Thus, for g_t sufficiently close to zero, it is true that

$$(c,x_t) > (c,x_r'). \qquad 3.45$$

Such a condition will be seen to be desirable, for it implies finiteness. For equation 3.45 to be true, it is seen from equation 3.44 that

$$g_t < \frac{(c,x_t) - (c,x_r)}{\sum_i w_{ri} p_{ti}} . \qquad 3.46$$

Let w* be the most negative w_i observed during any cycle. Hence, it follows that it is sufficient that

$$g_t < \frac{(c,x_t) - (c,x_{t-1})}{n|w^*|}$$

so that relation 3.45 will always be true. Thus, it may be seen as in theorem 3.7 that the algorithm will not again select the set T_r even if it corresponds to a feasible end of cycle point. Also, the algorithm will not select T_r if such a selection corresponds to an infeasible point. Hence, due to the finiteness of the set of all T_s, it follows that only a finite number of perturbations may be performed, proving the theorem.

It is now possible to prove that the algorithm requires only a finite number of cycles to be performed.

Corollary 3.8: After a finite number of cycles the algorithm will either converge to an optimum and detect it such or will detect an unbounded solution.

Proof: It has been proved that the number of perturbations

is finite, so assume that the last perturbation has occurred. Further, since the number of cycles is finite, then assume that the last cycle has occurred. If the solution is bounded, then corollary 3.7 indicates that the algorithm will detect an optimal point as such. And if the solution is unbounded, theorem 3.6 indicates that the algorithm will detect the condition. This proves the corollary and establishes the finiteness of the algorithm.

It is appropriate to note that although the technique of perturbation is necessary to establish some desirable properties of the orthogonal method, it is not anticipated that perturbation will be frequently used; for situations requiring perturbation are the exception rather than the rule. It is also noted that when the b_i are perturbed, it is not the original problem which is solved but rather the perturbed problem. However, for perturbation quantities g_t very close to zero, the change in the problem is negligible when compared to changes induced in the problem by the finite arithmetic of computing machines.

Starting points

The only property of the algorithm not yet discussed is how x_0, the initial feasible point implicitly assumed to exist in most of the theorems, may be determined. The algorithm itself may be used to determine a feasible point just as is done in many other algorithms. One method

for doing this is described below. Let N be the set of constraints which are not violated and V be the set of constraints which are violated. Then iteratively suppose that $a_p \varepsilon V$ and

$$\text{maximize } (a_p, x),$$

subject to:

$$(a_i, x) \geq b_i \quad , \; a_i \varepsilon N.$$

As soon as

$$(a_p, x) \geq b_p$$

then remove a_p from V and add it to N. Continue this process until V is empty, then procede with the maximization of (c,x). If it is impossible to make V null, then no feasible solution exists.

NUMERICAL EXAMPLE AND COMPUTATIONAL ASPECTS

The orthogonal method of linear programming and its properties have been presented and validated in the preceding sections. This section will present a short numerical example. This example will serve to illustrate certain computational aspects of the method.

Numerical example

Consider the problem

$$\text{maximize } -4x_1 + 2x_2 - 13x_3,$$

subject to:

$$3x_1 - 2x_2 + 4x_3 \geq -1,$$

$$-2x_1 + x_2 + 5x_3 \geq -2,$$

$$x_1 \geq 0,$$

$$x_2 \geq 0,$$

$$x_3 \geq 0,$$

Let c be alternatively designated as a_6, and let

$$A = \begin{pmatrix} a_1 \\ \cdot \\ \cdot \\ \cdot \\ a_6 \end{pmatrix} .$$

The various inner products (a_i,a_j) are frequently needed during the algorithm, so compute the matrix AA^T which has (a_i,a_j) as the element on row i, column j. Then

$$AA^T = \begin{pmatrix} 29 & 12 & 3 & -2 & 4 & -68 \\ 12 & 30 & -2 & 1 & 5 & -55 \\ 3 & -2 & 1 & 0 & 0 & -4 \\ -2 & 1 & 0 & 1 & 0 & 2 \\ 4 & 5 & 0 & 0 & 1 & -13 \\ -68 & -55 & -4 & 2 & -13 & 189 \end{pmatrix}$$

Since the origin is feasible, choose it as x_0.

To initialize the algorithm let r be zero and let F be null.

Step 1: Let k be zero, and let T_1 be the null set.

Step 4: First compute the $(a_i,\overline{z}_0)$ or (a_i,c) and note the initial values of y_i:

$$\begin{array}{rccccc} i = & 1 & 2 & 3 & 4 & 5 \\ (a_i,\bar{z}_0) = & -68 & -55 & -4 & 2 & -13 \\ y_i = & 1 & 2 & 0 & 0 & 0 \end{array} .$$

Now compute d_s:

$$d_s = \min_i \{d_i = - \frac{y_i}{(a_i,\bar{z}_0)} : (a_i,\bar{z}_0) < 0\}.$$

Both d_3 and d_5 are zero, and F is null. So arbitrarily chose a_3 to be a* and go to step 2.

Step 2: Let k equal one, and let a_1' equal a_3 so that

$$z_1 = a_3 = (1 \quad 0 \quad 0)$$

and

$$(z_1,z_1) = 1.$$

Since k is not equal to n, then go to step 3.

Step 3: Compute

$$(c,z_1) = (c,a_3) - 0 = -4$$

and

$$(c,\bar{z}_1) = (c,c) - \frac{(c_1,z_1)^2}{(z_1,z_1)} = 173 > 0.$$

Step 4: Use the formulae

$$(a_i,z_1) = (a_i,a_1') - \sum_{j=0}^{k-1} \frac{(z_j,a_1')}{(z_j,z_j)} (z_j,a_i)$$

and

$$(a_i,\bar{z}_1) = (a_i,\bar{z}_1) - \frac{(c,z_1)}{(z_1,z_1)} (z_1,a_i) \quad , a_i \notin T_1,$$

$$(a_i,\bar{z}_1) = 0 \quad , a_i \in T_1,$$

to obtain

$$\begin{aligned} i &= \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \\ (a_i,z_1) &= \quad 3 \quad -2 \quad - \quad 0 \quad 0 \\ (a_i,\bar{z}_1) &= -56 \quad -63 \quad 0 \quad 2 \quad -13 \\ y_i &= \quad 1 \quad 2 \quad 0 \quad 0 \quad 0 \ . \end{aligned}$$

Thus,

$$d_5 = - \frac{0}{-13} = 0$$

is the minimal d_i, and a* is equal to a_5. Since d_s is zero, go to step 2 and begin another iteration.

Step 2: Let k equal two and let a_2' equal a_5 so that

$$z_2 = a_5 - \frac{(z_1,a_5)}{(z_1,z_1)} z_1 = (0 \quad 0 \quad 1)$$

and

$$(z_2,z_2) = 1.$$

Since k is not equal to n, go to step 3.

Step 3: Compute

$$(c,z_2) = (c_1,a_2') - \frac{(z_1,a_2')}{(z_1,z_1)} (z_1,c) = -13$$

and

$$(c,\bar{z}_2) = (c,\bar{z}_1) - \frac{(c,z_2)^2}{(z_2,z_2)} = 4 > 0 \ .$$

Step 4: Use the formulae

$$(a_i,z_2) = (a_i,a_2') - \frac{(z_1,a_2')}{(z_1,z_1)} (z_1,a_i)$$

and

$$(a_i,\bar{z}_2) = (a_i,\bar{z}_1) - \frac{(c_1,z_2)}{(z_2,z_2)}(z_2,a_i) \quad , a_i \notin T_1,$$

$$(a_i,\bar{z}_2) = 0 \quad , a_i \in T_1,$$

to obtain

$i =$	1	2	3	4	5
$(a_i,z_2) =$	4	5	-	0	-
$(a_i,\bar{z}_2) =$	-4	2	0	2	0
$y_i =$	1	2	0	0	0

Thus,

$$d_1 = -\frac{1}{-4} = \frac{1}{4}$$

is the minimal d_i and a* is equal to a_1. Since a move may be made it is necessary to update the values of the y_i in step 5.

Step 5: Update the y_i by the formula

$$y_{i,new} = y_{i,old} + d_s(a_i,\bar{z}_2)$$

for those i such that $a_i \notin T_i$, and let $y_{1,new}$ equal zero, so that

$$y_{1,new} = 0,$$

$$y_{2,new} = 2+(1/4)(2) = 5/2$$

$$y_{4,new} = 0+(1/4)(2) = 1/2,$$

Now another iteration may be performed.

Step 2: Let k equal three and let a_3' equal a_1 so that

$$z_3 = a_1 - \frac{(z_1,a_1)}{(z_1,z_1)} z_1 - \frac{(z_2,a_1)}{(z_1,z_1)} z_2 = (0 \quad -2 \quad 0)$$

and

$$(z_3,z_3) = 4.$$

Now k is equal to n, so compute (c,z_3) and go to step 6.

$$(c,z_3) = (c,a_3') - \frac{(z_1,a_3')}{(z_1,z_1)} (z_1,c) - \frac{(z_2,a_3')}{(z_2,z_2)} (z_2,c) = -4$$

Step 7: An end of cycle test point has been found. Thus, determine the w_i and check for optimality:

$$w_3' = w_1 = \frac{(z_3,c)}{(z_3,z_3)} = -1,$$

$$w_2' = w_5 = (z_2,c) - \frac{(z_2,a_3')(w_3)}{(z_2,z_2)} = -9,$$

$$w_1' = w_3 = (z_3,c) - \frac{(z_1,a_3')(w_3)+(z_1,a_2')w_2}{(z_1,z_1)} = -1.$$

All w_k' are negative, and so the current point is optimal. The optimal dual solution is given above with the understanding that all w_i not explicitly calculated are zero. The optimal primal values may be directly determined by current values of the y_i corresponding to the sign restraints, thus

$$x^* = \begin{pmatrix} 0 \\ 1/2 \\ 0 \end{pmatrix}.$$

Computational aspects

The Orthogonal method of linear programming is not well suited to hand calculations, for it lacks a convenient tableau format. This has the effect of making the algorithm slightly more difficult to present, but has nothing to do with its performance on a computer. It is interesting to compare the computational requirements of the new method with that of a more familiar algorithm, the Simplex method.

Each new test point generated by the Simplex method requires on the order of nm multiplications and additions. Whereas iteration k of any cycle of the orthogonal method requires on the order of k(m+n) multiplications and subtractions, implying that each cycle takes on the order of $\frac{n(n+1)(m+n)}{2}$ multiplications and additions. If each cycle were to produce a result identical to n iterations of the Simplex method, then roughly (n)(nm) Simplex multiplications and additions would be required. Of course, this comparison is to some degree meaningless since the two methods cannot be expected to produce the assumed identical results, nor is the Simplex method required to operate in cycles. The contrast of the two methods only indicates that the Orthogonal method may have some promise, nothing more. The primary factor affecting the relative efficiency of the two methods is how slowly or swiftly the Orthogonal method will actually converge to the new optimum. The new method

has the capability of moves through the interior (the first move of each cycle) or across constraint surfaces; however, once an a_i has been added to T_r, it cannot be removed until the next cycle.

Another very important computational consideration is the computer memory required by the Orthogonal method. First, suppose that most work will be performed in core. Then the primary consideration here is the array sizes which are necessary for efficient performance. Here it is useful to take advantage of the special form of a_i which correspond to sign restrictions. Thus, let the problem formulation be

$$\text{Max } (c,x),$$

subject to:

$$Ax \geq b ,$$

$$x \geq 0 ,$$

and let the c vector be augmented to the A matrix. The matrix A and either the lower or upper half of AA^T would then be stored in core. Of course, some singly dimensioned work arrays would also be necessary, but they will require much less space than the above arrays. When a vector a_i enters T_r the (a_i,a_k) and a_{ij} associated with it are no longer used in calculations for the remainder of cycle r, but the (z_i,a_k) and z_{ij} are needed. Hence, the A and AA^T matrices may be partially destroyed by each iteration of a cycle to store the (z_i,a_k) and z_{ij}. Then at the

beginning of each cycle the matrices may be reinitialized from disk storage; with adequate blocking factors this should be a fairly efficient procedure. Thus, the array storage required by the orthogonal method is on the order of (mn + mm/2) words of memory plus the amount required by blocking. A similar usage of core memory in the simplex method would require roughly mn words, realizing that it is not necessary to store the identity matrix present in every tableau. Thus, the simplex method would require less core storage.

Now suppose that the problem were rather large. Then both methods would generally use peripheral storage and bring into core memory only a few column vectors at a time. Hence, neither method would require a large amount of core storage locations for arrays. It may be noted that computational efficiency is primarily of concern when the linear program is large, and for this case core memory requirements are about equal. In conclusion, it seems that the most important factor concerning the relative computational efficiency of the orthogonal method will be the speed with which it converges.

SUMMARY

The computational aspects of the solution procedure recommended for the interval linear programming problem gives

rise to consideration of procedures designed to swiftly provide good starting points for the solution of linear programs. Three heuristics designed to determine such points were unsuccessful. However, a new method of linear programming evolved from one of the heuristics. This method consists of cycles in which the Gram Schmidt process is iteratively used to compute the directions of potential moves. The presentation of the algorithm consists of a review of the Gram Schmidt process, preliminary theorems, the statement of the algorithm, proofs concerning the finiteness of the algorithm, and the numerical example and discussion of computational aspects.

The testing of the heuristics led to the development of techniques of generating convex programs with randomly chosen parameters but known solutions. These techniques may be used to ascertain the quality of a heuristic without actually having to solve the programs. The program generation techniques are the topic of the next chapter.

CHAPTER FOUR

PROGRAM GENERATION

INTRODUCTION AND ORGANIZATION

Methods for generating mathematical programs with arbitrarily chosen parameters but known solutions are the topic of this chapter. These techniques permit computationally efficient testing of the quality of starting point heuristics, for it is not necessary to perform a search to determine the optimum. In fact, it was the testing of the heuristics discussed in the preceding chapter which prompted the development of a generation procedure for linear programs. This procedure subsequently evolved into three more general methods. The first method uses the constraint functions in formation of the objective function, whereas the second method uses the gradients of the constraint functions. The techniques are referred to as the direct and the gradient generation procedures, respectively. The third method uses either of the first two techniques to generate integer programs. This chapter consists of the presentation of these methods and a brief summary. The following chapter contains conclusions and recommendations for further research.

DIRECT GENERATION PROCEDURE

As in Chapter Three, the following general problem formulation will be assumed.

$$\text{maximize} \quad f(p_0,x), \tag{4.1}$$

subject to:

$$g_i(p_i,x) \geq b_i \quad , i\varepsilon I, \tag{4.2}$$

$$g_j(p_j,x) \geq b_j \quad , j\varepsilon J, \tag{4.3}$$

where I and J are index sets. The forms of the constraint functions may be chosen in a completely arbitrary manner. For example, some may be linear, some exponential, etc. Let

$$y_k = g_k(p_k,x) - b_k \quad , \forall k, \tag{4.4}$$

and let $r_k(y_k)$ be any function such that

$$r_k(0) = \inf \{r_k(y_k) : y_k \geq 0\}, \forall k. \tag{4.5}$$

Some functions for which relation 4.5 is true are

$$r_k(y_k) = y_k, \tag{4.6}$$

$$r_k(y_k) = y_k^2, \tag{4.7}$$

$$r_k(y_k) = e^{y_k}, \tag{4.8}$$

$$r_k(y_k) = \sin(y_k). \tag{4.9}$$

Now the steps of the procedure may be given.

1. Choose functional forms for the $g_k(p_k,x)$ and $r_k(y_k)$.
2. Choose values for the parameter sets p_k, except for

p_0, and the optimum x*, randomly generating them if desired.

3. Compute

$$b_i = g_i(p_i,x^*) \quad , i\varepsilon I, \tag{4.10}$$

and

$$b_j = g_j(p_j,x^*) - e_j, \quad j\varepsilon J, \tag{4.11}$$

where the e_j are arbitrary non-negative numbers so

$$y_i^* = 0 \quad , i\varepsilon I, \tag{4.12}$$

and

$$y_j^* = e_j \quad , j\varepsilon J, \tag{4.13}$$

4. Randomly generate the negative scalars μ_i and form the objective function

$$f(p_0,x) = \sum_{i\varepsilon I} \mu_i r_i [g_i(p_i,x)-b_i]. \tag{4.14}$$

It may be seen by construction that a primal optimum of the generated program is x*, for this point is feasible and such that $f(p_0,x^*)$ is maximal. The value of $f(p_0,x)$ is maximal because by construction at x* the functions

$$r_i[g_i(p_i,x^*) - b_i] = r_i(0) \quad , i\varepsilon I,$$

are minimal for non-negative y_i. Then since the y_i must be non-negative and since the μ_i are negative, it follows that

$$f(p_0,x^*) = \sum_{i\varepsilon I} \mu_i r_i [g_i(p_i,x^*) - b_i]$$

is a global optimum. Additionally, the partials of the

objective function with respect to the b_k evaluated at x* or the optimal dual variables, v_k^*, may be computed from

$$v_i^* = \frac{\partial}{\partial b_i} \{\mu_i r_i [g_i(p_i,x)-b_i]\}\Big|_{x=x^*} \quad , \; i\varepsilon I, \qquad 4.15$$

and

$$v_j^* = 0 \quad , \; j\varepsilon J. \qquad 4.16$$

This procedure is illustrated by the following two examples, one a linear program and the other a quadratic program.

Generation of a linear program

Consider the following linear program,

maximize (c,x) ,

subject to:

$(a_i,x) \geq b_i$, $i=1,2$,

$(a_j,x) \geq b_j$, $j=3,4$.

The steps of the algorithm are given below.

1. All $g_i(p_i,x)$ are linear, and let $r_k(y_k)$ be the identity function as given by equation 4.6.
2. Arbitrarily choose

$$a_1 = (2 \quad -1) \; ,$$

$$a_2 = (0 \quad 1) \; ,$$

$$a_3 = (1 \quad 3) \; ,$$

$$a_4 = (1 \quad 0) \; ,$$

and

$$x^* = \begin{pmatrix} 3 \\ \\ 1 \end{pmatrix}.$$

3. Let only the first two constraints be equalities at the optimum so that

$$b_1 = 2(3) - 1(1) = 5 \ ,$$

$$b_2 = \quad 1(1) = 1 \ ,$$

$$b_3 = 1(3) + 3(1) - 2 = 4,$$

$$b_4 = 1(3) - 3 = 0$$

where

$$e_3 = 2$$

and

$$e_4 = 3.$$

4. Let

$$\mu_1 = -1$$

and

$$\mu_4 = -2$$

so that

$$(c,x) = (-2)(2x_1 - 1x_2 - 5) + (-1)(x_2 - 1)$$

or

$$(c,x) = -4x_1 + x_2 + 11.$$

Now realizing that the constant term "11" may be dropped from the objective function, the generated linear program is

maximize $-4x_1 + x_2$,

subject to:

$$2x_1 - x_2 \geq 5,$$

$$x_2 \geq 1,$$

$$x_1 + 3x_2 \geq 4,$$

$$x_1 \geq 0.$$

For this program the optimum is

$$x^* = \begin{pmatrix} 3 \\ \\ 1 \end{pmatrix},$$

with

$$(c,x^*) = 11.$$

Now it may be seen that for any linear program generated by the procedure that

$$\mu_i = \frac{\partial}{\partial b_i} \{ \sum_{i\varepsilon I} \mu_i [(a_i,x)-b_i] \} \Big|_{x=x^*} \quad , \; i\varepsilon I;$$

so for this program

$$v^* = \begin{pmatrix} -2 \\ -1 \\ 0 \\ 0 \end{pmatrix}.$$

Linear programs which have either no feasible

solution or an unbounded solution may also be generated. In general, programs which have no feasible solution may be obtained by forming the incompatible constraint

$$\sum_k \mu_i g_i(p_i,x) \leq \sum_k \mu_i b_i - d$$

where the μ_i and d are positive scalars. If the program is linear, then the fact that the dual of a primal with no feasible solution is unbounded may be used to obtain the unbounded (dual) program.

Generation of a quadratic program

Consider the preceding linear problem. It may be changed to a quadratic program merely by altering step 4. Choose

$$r_k(y_k) = y_k^2$$

instead of using the identity function. The objective function becomes

$$f(p_0,x) = (-2)(2x_1-1x_2-5)^2 + (-1)(x_1-1)^2$$

so that, dropping the constant term,

$$f(p_0,x) = 42x_1 - 20x_2 - 9x_1^2 + 8x_1x_2 - 20x_2 - 2x_2^2 .$$

The constraint set and optimum remain unchanged. For a quadratic program generated in this fashion

$$f(p_0,x^*) = 0;$$

also

$$v_k^* = 0 \quad , \forall k,$$

since

$$v_i^* = 2\mu_i[(a_i,x)-b_i](-1)\Big|_{x=x^*} \quad , \; i\varepsilon I,$$

or

$$v_i^* = 0 \quad , \; i\varepsilon I,$$

and

$$v_j^* = 0 \quad , \; j\varepsilon J.$$

If this condition is not desirable, then the second program generation procedure may be used.

GRADIENT GENERATION PROCEDURE

This method does not require that the objective function contain terms involving the constraint functions, and is more general than the first method in that regard. However, the optimum indicated by this technique may not be global if the program is not convex; also the former method may be computationally more efficient. The difference in the two methods is the manner in which the objective function is formed; the first three steps are identical. The remaining steps of the gradient generation procedure are given below.

4. Choose the form of $f(p_0,x)$ and write the symbolic expressions for the gradient of $f(p_0,x)$ evaluated at x^*; the elements of p_0 are as yet undetermined. So let

$$\nabla f(p_0,x^*) = \nabla f(p_0,x)\Big|_{x=x^*} \quad . \tag{4.17}$$

5. Determine the values of the gradients of the $g_i(p_i,x)$, $i\varepsilon I$, evaluate at x^*, letting

$$\nabla g_i(p_i,x^*) = \nabla g_i(p_i,x)\Big|_{x=x^*}.$$

The numerical values of the p_i are used so that the elements of each $\nabla g_i(p_i,x^*)$ are known scalars.

6. Generate negative scalars μ_i and let

$$\nabla f(p_0,x^*) = \sum_{i\varepsilon I} \mu_i \nabla g_i(p_i,x^*). \tag{4.18}$$

7. Solve equation 4.18 for the values of the elements of p_0.

This procedure may be seen to be such that the Kuhn-Tucker conditions are by construction true at x^*. For a convex programming problem, x^* is a global optimum; otherwise it may be only a local optimum. Also by construction the μ_i, $i\varepsilon I$, are the optimal dual variables v_i^*; all v_j^*, $j\varepsilon J$, are zero. If $f(p_0,x)$ is a linear function, then the elements of p_0 may be computed by a direct addition of the elements of the right hand side of equation 4.18. Thus, if the $g_i(p_i,x)$ are linear, then this procedure is equivalent to the direct generation procedure. If the form of $f(p_0,x)$ is quadratic, then the difference between the two methods is more evident. An example of this is given below.

Generation of a quadratic program

Consider the same constraint set used in the preceding examples,

$$2x_1 - x_2 \geq 5 ,$$

$$x_2 \geq 1 ,$$

$$x_1 + 3x_2 \geq 1 ,$$

$$x_1 \geq 0 .$$

and the same optimum

$$x^* = \begin{pmatrix} 3 \\ 1 \end{pmatrix} .$$

4. The form of $f(p_0,x)$ may be given by

$$f(p_0,x) = c_1x_1 + c_2x_2 + \tfrac{1}{2}q_{11}x_1^2 + q_{12}x_1x_2 + \tfrac{1}{2}q_{22}x_2^2,$$

so that

$$\frac{\partial f(p_0,x)}{\partial x_1} = c_1 + q_{11}x_1 + q_{12}x_2$$

and

$$\frac{\partial f(p_0,x)}{\partial x_2} = c_2 + q_{12}x_1 + q_{22}x_2.$$

Thus

$$\nabla f(p_0,x^*) = \begin{pmatrix} c_1+3q_{11}+3q_{22} \\ c_2+3q_{12}+q_{22} \end{pmatrix} . \qquad 4.19$$

5. As before, the first two constraints have been structured so that they will be equalities at the optimum. Thus compute

$$\nabla g_i(p_1,x^*) = \begin{pmatrix} 2 \\ \\ -1 \end{pmatrix}$$

and

$$\nabla g_2(p_2,x^*) = \begin{pmatrix} 0 \\ \\ 1 \end{pmatrix} .$$

6. Choose

$$\mu_1 = -2$$

and

$$\mu_2 = -1 \quad ,$$

so that

$$\nabla f(p_0,x^*) = \sum_{i\varepsilon I} \mu_i \nabla g_i(p_i,x^*) .$$

Evaluate the right hand side and use equation 4.19 to obtain

$$c_1 + 3q_{11} + 3q_{12} = -4 \qquad 4.20$$

$$c_2 + 3q_{12} + q_{22} = 1. \qquad 4.21$$

7. To solve these equations for p_0, arbitrarily let

$$q_{11} = 2,$$

$$q_{12} = 1,$$

and

$$q_{22} = 4,$$

so that equations 4.20 and 4.21 result in

$$c_1 = -13$$

and

$$c_2 = -6.$$

Thus the generated quadratic program is

$$\text{maximize} \quad -13x_1 - 6x_2 + x_1^2 + x_1x_2 + 2x_2^2 .$$

subject to:

$$2x_1 - x_2 \geq 5,$$

$$x_2 \geq 1,$$

$$x_1 + 3x_2 \geq 1,$$

$$x_1 \geq 0,$$

with

$$x^* = \begin{pmatrix} 3 \\ \\ 1 \end{pmatrix}$$

and

$$v^* = \begin{pmatrix} -2 \\ 0 \\ 0 \\ -1 \end{pmatrix}.$$

It may be seen that for any quadratic program generated by this procedure the majority of the parameters may be arbitrarily chosen. The form of the objective function

$$f(p_0,x) = \sum_{k=1}^{n} c_k x_k + \frac{1}{2} \sum_{k=1}^{n} \sum_{r=1}^{n} q_{kr} x_k x_r$$

indicates that the equation

$$\nabla f(p_0,x^*) = \sum_{i\varepsilon I} \mu_i \nabla g_i(p_i,x^*)$$

may be written

$$c_k + \sum_{r=1}^{n} q_{kr} x_r^* = \sum_{i\varepsilon I} \mu_i g_i(p_i,x^*), \quad k=1,\ldots,n. \qquad 4.22$$

There are n simultaneous linear equations in n(n+1) unknowns. Thus, if it is desired, all q_{kr} may be randomly chosen and the c_k computed from

$$c_k = \sum_{i\varepsilon I} \mu_i \nabla g_i(p_i,x^*) - \sum_{r=1} q_{kr} x_r^* \quad , k=1,\ldots,n. \qquad 4.23$$

Thus for linear and quadratic program generation both methods appear to be equally efficient. However, for more general programs the gradient technique may require the solution of equations which are not solved as easily as in the quadratic case.

INTEGER PROGRAM GENERATION

The direct and gradient program generation procedures may also be used to generate mathematical programs in which some or all variables are required to be integers. There is currently a technique for the generation of zero-one linear programs which was used in an analysis of covering

problem heuristics [26]. The method of this section bears no resemblence to this technique, and it may be used for non-linear integer programs. The procedure will be presented after a preliminary theorem, then it will be illustrated by the generation of an integer linear program.

Theorem 4.1 demonstrates the equality of two solution sets. The objective function is obtained by using one solution set, but the problem formulation stipulates the other solution set.

Theorem 4.1: Let D be the solution set of

$$g_{1i}(p_{1i},x) + g_{2i}(p_{2i},x) \geq b_i \quad , i\varepsilon I, \tag{4.24}$$

and

$$g_j(p_j,x) \geq b_j \quad , j\varepsilon J, \tag{4.25}$$

where by problem construction it is true that

$$g_{2i}(p_{2i},x) < 1 \quad , i\varepsilon I, \tag{4.26}$$

and the b_i are integers for $i\varepsilon I$. Additionally require that the $g_{1i}(p_{1i},x)$ be integers for $i\varepsilon I$.

Let E be the solution set of

$$g_{1i}(p_{1i},x) + g_{2i}(p_{2i},x) \geq b_i \quad , i\varepsilon I, \tag{4.27}$$

$$g_{1i}(p_{1i},x) \geq b_i \quad , i\varepsilon I, \tag{4.28}$$

and

$$g_j(p_j,x) \geq b_j \quad , j\varepsilon J. \tag{4.29}$$

Then it is true that

$$D = E.$$

Proof: Let x be an element of D and assume that

$$g_{1i}(p_{1i},x) < b_i \quad , i\varepsilon I.$$

Then since all quantities are integers it is true that

$$g_{1i}(p_{1i},x) \leq b_i - 1 \quad , i\varepsilon I;$$

and since

$$g_{2i}(p_{2i},x) < 1,$$

then

$$g_{1i}(p_{1i},x) < b_i - g_{2i}(p_{2i},x) \quad , i\varepsilon I,$$

or

$$g_{1i}(p_{1i},x) + g_{2i}(p_{2i},x) < b_i \quad , i\varepsilon I.$$

This contradicts the assumption that x is an element of D. Thus it follows that

$$g_{1i}(p_{1i},x) \geq b_i$$

and x is an element of E. Also if x is an element of E, then it is immediately true that x is an element of D, for the set E is specified by the same restrictions as set D, plus an additional one. Thus it may be seen that the two sets are equal, proving the theorem.

Now the procedure may be stated.

1. Choose functional forms for the $g_{1i}(p_{1i},x)$, $g_{2i}(p_{2i},x)$, and $g_j(p_j,x)$. Care must be exercised so that the $g_{1i}(p_{1i},x)$ will, with suitable restrictions on the variables if necessary, be integer valued.

2. Assign values to x* and to the parameter sets p_{1i}, p_{2i}, and p_j. Parametric values must be chosen such that

$$g_{2i}(p_{2i},x) < 1 \quad , i\varepsilon I,$$

will be true so theorem 4.1 may be used. Also let

$$0 \leq g_{2i}(p_{2i},x^*).$$

It may be seen in step 4 that this is necessary for feasibility. One convenient method for satisfying these requirements is to specify that the $g_{2i}(p_{2i},x)$ are constant functions with values within the allowable zero to one range.

3. Compute

$$b_j = g_j(p_j,x^*) - e_j \quad , j\varepsilon J,$$

where the e_j are arbitrary non-negative numbers so that

$$y_j^* = e_j \quad , j\varepsilon J.$$

4. Let

$$b_i = g_{1i}(p_{1i},x^*) \quad , i\varepsilon I,$$

So that

$$y_i^* = g_{2i}(p_{2i},x^*) \quad , i\varepsilon I.$$

5. Since theorem 4.1 indicates that the constraint set is implicitly

$$g_{1i}(p_{1i},x) + g_{2i}(p_{2i},x) \geq b_i \quad , i\varepsilon I,$$

$$g_{1i}(p_{1i},x) \geq b_i \quad , i\varepsilon I,$$

$$g_j(p_j,x) \geq b_j \quad , j\varepsilon J,$$

then the constraints

$$g_{1i}(p_{1i},x) \geq b_i \quad , i\varepsilon I,$$

which are by construction equalities at x* may be used to form $f(p_0,x)$ by either the direct or the gradient generation procedure.

6. The generated problem is

maximize $f(p_0,x)$

subject to:

$$g_{1i}(p_{1i},x) + g_{2i}(p_{2i},x) \geq b_i \quad , i\varepsilon I,$$

$$g_j(p_j,x) \geq b_i \quad , j\varepsilon J.$$

The point x* is by construction an optimum over the solution space E, represented by equations 4.27 through 4.29, as in the previous methods. Theorem 4.1 then implies that it is also an optimum over the solution space D, designated by equations 4.24 and 4.25. Thus if the direct method is used, then x* is a global optimum over E and hence over D; but if the gradient procedure is used, then x* may in general be either a local or a global optimum over D. However, for the gradient procedure if the problem with feasible region E is a convex programming problem, then x* must be a global optimum over E, and hence over D.

The integer procedure is illustrated by the following example.

Generation of an integer linear program

Consider the linear programming problem illustrated previously.

$$\text{maximize} \quad -4x_1 + x_2 ,$$

subject to:

$$2x_1 - x_2 \geq 5 ,$$

$$x_2 \geq 1 ,$$

$$x_1 + 3x_2 \geq 0 ,$$

$$x_1 \geq 0 ,$$

for which the optimal primal and dual variables are

$$x^* = \begin{pmatrix} 3 \\ \\ 1 \end{pmatrix}$$

and

$$v^* = \begin{pmatrix} -2 \\ -1 \\ 0 \\ 0 \end{pmatrix} .$$

This program may be converted to an integer linear program by following the steps of the integer procedure.

1. The functions are all to be linear.
2. Let

$$g_{11}(p_{11},x) = 2x_1 - x_2$$

and

$$g_{12}(p_{12},x) = x_2 \quad ,$$

where the x_j are restricted to be integers so that $g_{11}(p_{11},x)$ and $g_{12}(p_{12},x)$ are integers. Also let

$$g_3(p_3,x) = x_1 + 3x_2$$

and

$$g_4(p_4,x) = x_1.$$

Now choose

$$g_{21}(p_{21},x) = 1/2$$

and

$$g_{22}(p_{22},x) = 3/4.$$

3. As before, let

$$y_3^* = e_3 = 2$$

and

$$y_4^* = e_4 = 3$$

so that

$$b_3 = 4$$

and

$$b_4 = 0.$$

4. Now

$$b_1=2(3) - (1)=5$$

and

$$b_2= \qquad (1)=1$$

so that

$$y_1^* = g_{21}(p_{21}, x^*) = 1/2$$

and

$$y_2^* = g_{22}(p_{22}, x^*) = 3/4.$$

5. Form the objective function as before, letting μ_1 and μ_2 be -2 and -1, respectively, so that

$$(c,x) = (-2)(2x_1 - x_2 - 5) + (-1)(x_2 - 1).$$

6. Then the final form of the problem simplifies to

maximize $\quad -4x_1 + x_2$,

subject to:

$$2x_1 - x_2 \geq 4\tfrac{1}{2},$$

$$x_1 \geq \tfrac{1}{4},$$

$$x_1 + 3x_2 \geq 4,$$

$$x_1 \geq 0,$$

x_1, x_2 integers,

for which

$$x^* = \begin{pmatrix} 3 \\ \\ 1 \end{pmatrix}.$$

This is illustrated by figure 6 on the following page.

It is noted that although the example is linear, this condition is not necessary. For example, polynomial constraints may be structured so that the $g_{1i}(p_{1i}, x)$ must be integers. Additionally, it is not necessary that all x_j

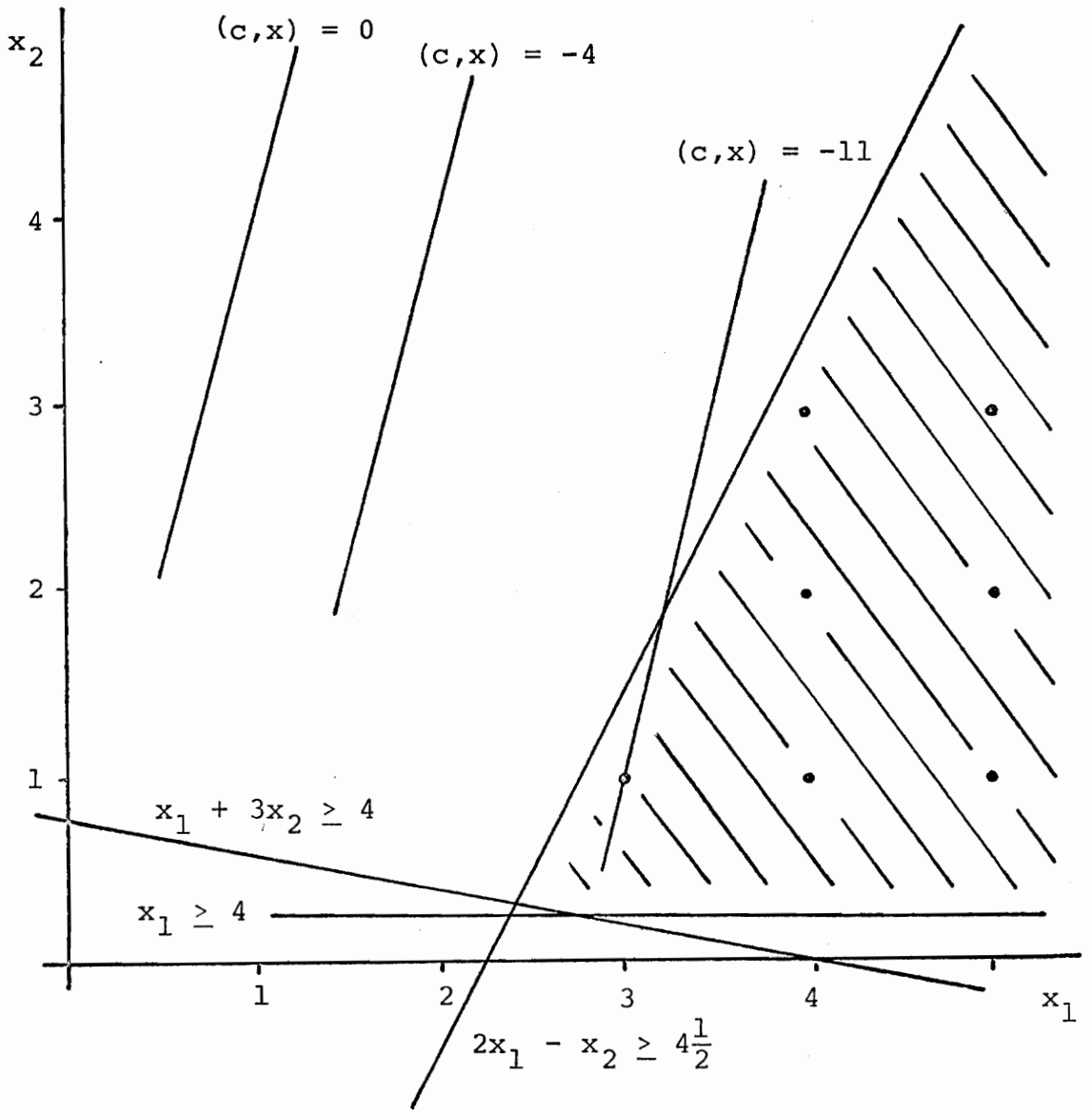

Figure 6. Integer programming example.

be integers. In order to use this generation technique it it merely necessary to have the $g_{1i}(p_{1i},x)$ integral and the $g_{2i}(p_{2i},x)$ between zero and one. Either interior or surface optima may be generated depending upon the choice of the $g_{2i}(p_{2i},x)$.

SUMMARY

The direct and the gradient generation procedures are appropriate for generating mathematical programs for continuous variables with known primal and dual solutions. The direct procedure is such that x* must always be a global optimum, regardless of convexity considerations. Also it may, for complex problems, be computationally more efficient than the gradient procedure. However, the gradient method permits more flexibility in the choice of the objective function. The optima obtained by this method are global optima for a convex programming problem, but may be only a local optimum for a non-convex problem. The integer program generation procedure forms a feasible region implicitly restricted such that either of the continuous variable techniques may be used to form the objective function in a manner forcing x* to be optimal.

The purpose of these methods is to provide a method for the efficient testing of starting point heuristics,

such as those discussed in Chapter Three. Additionally they may be used to investigate procedures for the removal of constraints which are not binding at the optimum and which thus merely serve to increase the complexity of the program since the algorithms are such that the constraints

$$g_j(p_j,x) \geq b_j \quad , \; j \in J,$$

are not binding at the optimum. In general, the generation procedures may be used whenever it is desired to test heuristics or programming techniques, but it is not desirable to determine optima by search methods.

This chapter concludes the presentation of research results. The following chapter contains a summary and recommendations for further research.

CHAPTER FIVE

SUMMARY AND RECOMMENDATIONS FOR FURTHER RESEARCH

SUMMARY

The research develops three distinct, but logically related topics. Interval convex programming provides a method for specifying the range of possible optimal solutions for convex programming problems for which only interval estimates of parametric values are available. The technique is intended to be a decision aid for the class of problems in which costs, availabilities, production capabilities, or other parameters are not known with certainty. The solution procedure for interval convex programs requires the solution of a series of 2n real valued programming problems. The computational expense of the method is the price which must be paid for the additional information which it provides: an inherent sensitivity analysis over the range of all possible parametric realizations.

Interval linear programs are anticipated to be the most frequently occurring type of interval convex programs, for linear programs are the most frequently occurring type of convex programs. Thus, it appears worthwhile to

attempt to reduce incremental information costs for interval linear programs. One method for reducing this cost is to choose starting points for linear programs which are closer to an optimum than a commonly chosen initial point, the origin. The series of 2n linear programs necessary for the solution of the interval problem is such that the initial programs may be used to provide starting points for later programs; however, other techniques must be used for the first programs. This consideration prompted an unsuccessful investigation of starting point heuristics which led to the development of a new method for solving the linear programming problem in which move directions are computed using the Gram Schmidt orthogonalization procedure. This method operates in cycles, each of which consists of no more than n iterations. Within each cycle moves are made across the surfaces of constraints which have been entered into the orthogonalization process.

The testing of the starting point heuristics induced an investigation of methods for generating programs with known solutions but arbitrarily chosen parameters. Such techniques permit efficient testing of heuristics, for it is not necessary to perform what may be a time-consuming search for the optimum. Two methods are available for the generation of programs with continuous variables. Also, there is a method for generating programs with integer

restrictions; this method uses either of the two continuous variable methods to form the objective function.

RECOMMENDATIONS FOR FURTHER RESEARCH

This research is addressed to problems of a practical nature: parametric uncertainty in convex programming problems, efficiency in linear programming, and the effective testing of starting point heuristics for mathematical programs. The research is of a basic nature in that its purpose is to provide algorithms and methods for the solution of these practical problems, rather than examine the solution of a certain numerical problem or of a particular class of problems, such as production, facility location, or scheduling problems. Nonetheless it is felt that a large component of the worth of this research lies in its actual implementation and use. This then opens up several areas of research, some of which are the explicit formulation of production, facility location, and scheduling problems as interval convex programming problems. However, even the formulation effort is of a rather basic nature, and it will be necessary to actually solve some existing real world problems to see how effectively interval convex programming works as a decision aid.

Another practical problem posed by the research is to

evaluate the computational efficiency of the Orthogonal method of linear programming as compared to that of the most commonly used method, the Simplex method. This is a problem of a rather large scope, for relative efficiency must be examined and statistically documented over a wide range of problem sizes and types before any general statements can be made.

The program generation techniques are perhaps of the most immediate applicability. Problems which may be formulated as mathematical programs are sometimes solved by some approximate technique in order to achieve computational efficiency. The generation techniques may encourage the growth of such heuristics, thus extending the domain of applicability of decision models.

BIBLIOGRAPHY

1. Abadie, J., editor, Integer and Non-linear Programming, North Holland Pub. Co., Amsterdam, 1970.

2. Beale, E. M. L., Mathematical Programming In Practice, John Wiley and Sons Pub. Co., New York, 1968.

3. Bennett, G. K., Jr., "A Method for Locating the Zeros of a Polynomial Using Interval Arithmetic," Master's thesis, San Jose State University, Ca., 1968.

4. Beveridge, S. G. G. and Schechter, R. S., Optimization: Theory and Practice, McGraw Hill, Inc., New York, 1970.

5. Boot, J. C. G., Quadratic Programming, Rand McNally and Company, Chicago, 1964.

6. Box, M. J., "A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems," Computer Journal, Vol. 9, No. 1, May, 1966, pp. 67 - 77.

7. Carroll, C. W., "The Created Response Surface Technique for Optimizing Nonlinear, Restrained Systems," Operations Research, Vol. 9, No. 2, March-April, 1961, pp. 169 - 84.

8. Courant, R., "Variational Methods for the Solution of Problems of Equilibrium and Vibrations," Bull. Am. Math. Soc., Vol. 49, No. 1, 1943, pp. 1 - 23.

9. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, Princeton, N. J., 1968.

10. Dumitru, V. and Ionescu, V., "Numerical Solution of a Class of Problems of Nonlinear Programming," Studii si Cercetari Matematice, Vol. 18, No. 6, 1966, pp. 993 - 942, as abstracted by Bereanu, B., in Operations Research / Management Science International Literature Digest, edited by Rosenthal, A. J., 1967, p. 409.

11. Dwyer, P. S., Linear Computations, Wiley Pub. Co., New York, 1951.

12. Geoffrion, A. M., "Elements of Large-Scale Mathematical Programming," Management Science, Vol. 16, No. 11, July, 1970, pp. 652 - 91.

13. Greenstadt, J. L., "A Ricocheting Gradient Method for Nonlinear Optimization," Journal of SIAM Applied Mathematics, Vol. 14, No. 13, May, 1966, pp. 429 - 45.

14. Gue, R. L. and Thomas, M. E., Mathematical Methods in Operations Research, Macmillan Company, London, 1968.

15. Fiacco, A. V. and McCormick, G. P., "Extensions of SUMT for Nonlinear Programming: Equality Constraints and Extrapolation," Management Science, Vol. 12, No. 11, July, 1966, pp. 816 - 28.

16. Fiacco, A. V. and McCormick, G. P., "The Sequential Unconstrained Minimization Technique for Nonlinear Programming, a Primal-Dual Method," Management Science, Vol. 10, No. 2, January, 1964, pp. 360 - 66.

17. Fiacco, A. V. and McCormick, G. P., "Computational Experience for the Sequential Unconstrained Minimization Technique for Nonlinear Programming," Management Science, Vol. 10, No. 4, July, 1964, pp. 601 - 17.

18. Hadley, G., Linear Programming, Addison-Wesley Pub. Co., Inc., Reading, Mass., 1963.

19. Hadley, G., Nonlinear and Dynamic Programming, Addison-Wesley Pub. Co., Inc., Reading, Mass., 1964.

20. Hansen, E., "Interval Arithmetic In Matrix Computations," J. Soc. Indus. Appl. Math., Series B, Numerical Analysis, Vol. 2, No. 2, 1965, pp. 308 - 20.

21. Hansen, E., "On Solving Systems of Equations Using Interval Arithmetic," Mathematics of Computation, Vol. 22, No. 102, April, 1968, pp. 374 - 84.

22. Hansen, E., "On the Solution of Linear Algebraic Equations with Interval Coefficients," Linear Algebra and Its Applications, Vol. 2, No. 2, April, 1969, pp. 153 - 165.

23. Hansen, E. and Smith, R., "Interval Arithmetic in Matrix Computations, Part II," J. Soc. Indus. Appl. Math., Series B, Numerical Analysis, Vol. 4, No. 1, March, 1967, pp. 1 - 9.

24. Hillier, F. S. and Lieberman, G. J., Introduction to Operations Research, Holden-Day, Inc., San Francisco, 1967.

25. Hoffman, A., Mannos, M., Sokolowsky, D., and Wiegmann, N., "Computational Experience in Solving Linear Programs," J. Soc. Indus. Appl. Math., Vol. 1, No. 1, September, 1953, pp. 17 - 33.

26. Kuhn, H. W. and Quant, R. E., "An Experimental Study of the Simplex Method," in Decomposition Principles for Solving Large Structured Linear Programs, notes for a course presented by Mathematica at Berkeley, Cal., April 24 - 26, 1963.

27. Kunzi, H. P., Krelle, W., Oetelli, W., translated by Levin, F., Nonlinear Programming, Blaisdell Pub. Co., London, 1966.

28. Kunzi, H. P., "Mathematical Optimization of Large Systems," Ablauf-und Planungsforschung, Vol. 8, No. 4, 1967, pp. 395 - 407, as abstracted by Uebe, G., in Operations Research / Management Science International Literature Review, edited by Rosenthal, A. J., 1968, p. 467.

29. Kunzi, H. P., "The Triplex Algorithm," Unternehmensforschung, Vol. 12, No. 3, 1968, pp. 145 - 54, as abstracted by Uebe, G., in Operations Research / Management Science International Literature Digest, edited by Rosenthal, A. J., 1969, p. 155.

30. Mathematical Programming System / 360 (360A-CO-14x) Linear and Separable Programming - User's Manual, International Business Machine Corp., White Plains, New York, 1969.

31. Moore, R. E., Interval Analysis, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1966.

32. Mueller-Merbach, H. and Darmstadt, F. H., "Parametric Linear Programming," Ablauf-und Planungsforschung, Vol. 8, No. 3, 1967, pp. 341 - 54, as abstracted by Uebe, G., in Operations Research / Management Science International Literature Digest, edited by Rosenthal, A. J., 1968, pp. 397 - 99.

33. Oetelli, W., "On the Solution Set of a Linear System with Inaccurate Coefficients," J. Soc. Indus. Appl. Math., Series B, Numerical Analysis, Vol. 2, No. 1, 1965, pp. 115 - 18.

34. Orchard-Hays, W., Advanced Linear Programming Computing Techniques, McGraw-Hill, New York, 1968.

35. Paranjape, S. R., "The Simplex Method: Two Basic Variables Replacement," Management Science, Vol. 12, No. 1, September, 1965, pp. 135 - 41.

36. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part I - Linear Constraints," J. Soc. Indus. Appl. Math., Vol. 8, No. 1, March, 1960, pp. 181 - 217.

37. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part II - Nonlinear Consraints," J. Soc. Indus. Appl. Math., Vol. 9, No. 4, December, 1961, pp. 514 - 52.

38. Shayer, Sidney, "Interval Arithmetic with Some Applications for Digital Computers," Master's thesis, San Jose State University, Cal., 1965.

39. Taha, H. A., Operations Research, An Introduction, MacMillan Company, New York, 1971.

40. van de Panne, C., "Parameterizing an Activity Vector in Linear Programming," Operations Research, Vol. 21, No. 1, January-February, 1973, p. 389.

41. Wagner, H. M., Principles of Operations Research with Applications to Managerial Decisions, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1969.

42. Wilde, D. J. and Beightler, C. S., Foundations of Optimization, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1967.

43. Wismer, D. A., editor, Optimization Methods for Large Scale Systems ... with Applications, McGraw-Hill, Inc., New York, 1971.

44. Wolfe, P., "Computer Routines for Linear Programming and Decomposition," in Decomposition Principles for Solving Large Structure Linear Programs, notes for a course presented by Mathematica, at Berkeley, Cal., April 24 - 26, 1963.

45. Zangwill, W. I., Nonlinear Programming: A Unified Approach, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1969.

46. Zangwill, W. I., "Non-linear Programming Via Penalty Functions," Management Science, Vol. 13, No. 5, January, 1967, pp. 344 - 58.

VITA

John Heard Ristroph was born in New Orleans, Louisiana on November 12, 1946. He received his primary and secondary education there, graduating from Jesuit High School in 1964. He entered Louisiana State University in Baton Rouge in September, 1964, and received his B.S. degree from LSU in Industrial Engineering in January, 1969, then his M.S. degree in Industrial Engineering from LSU in August, 1970. He was married to Margaret G. Morrison in August, 1969.

In September, 1970, he entered the Graduate School of Virginia Polytechnic Institute and State University to begin work on a Doctor of Philosophy in the Department of Industrial Engineering and Operations Research. Requirements for the Ph.D. in Industrial Engineering and Operations Research were completed in December, 1975.

INTERVAL CONVEX PROGRAMMING, ORTHOGONAL LINEAR PROGRAMMING, AND PROGRAM GENERATION PROCEDURES

by

John Heard Ristroph

(ABSTRACT)

Three topics are developed: interval convex programming, and program generation techniques. The interval convex programming problem is similar to the convex programming problem of the real number system except that all parameters are specified as intervals of real numbers rather than as real scalars. The interval programming solution procedure involves the solution of a series of 2n real valued convex programs where n is the dimension of the space. The solution of an interval programming problem is an interval vector which contains all possible solutions to any real valued convex program which may be realized.

Attempts to improve the efficiency of the interval convex programming problem lead to the eventual development of a new solution procedure for the real valued linear programming problem, Orthogonal linear programming. This new algorithm evolved from some heuristic procedures which were initially examined in the attempt to improve solution efficiency. In the course of testing these

heuristics, which were unsuccessful, procedures were developed whereby it is possible to generate discrete and continuous mathematical programs with randomly chosen parameters, but known solutions.