

NOT TO BE TAKEN FROM THIS ROOM

MULTIOBJECTIVE LINEAR PROGRAMMING
AND
MULTIOBJECTIVE ZERO-ONE PROGRAMMING

by

Gülseren Kızıltan

B.S. in E.E., Boğaziçi University, 1975

M.S. in I.E., Boğaziçi University, 1977

Bogazici University Library



14

39001100316424

Submitted to the Faculty of the School of Engineering
in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy

in

Industrial Engineering

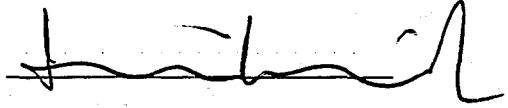
BOĞAZIÇI UNIVERSITY

1981

We hereby recommend that the thesis entitled "Multi-objective Linear Programming and Multiobjective Zero-One Linear Programming" submitted by GÜLSEREN KIZILTAN be accepted in partial fulfillment of the requirements for the Doctorate Degree in Industrial Engineering, School of Engineering, Boğaziçi University.

Examining Committee

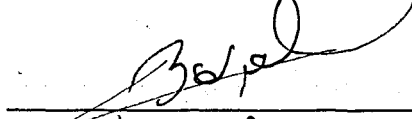
Prof.Dr. İbrahim KAVRAKOĞLU



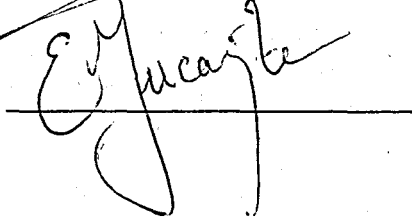
Prof.Dr. Murat SERTEL



Doç.Dr. Birsen KARPAK



Dr. Erkut YUCAOĞLU



Date: Jan. 22, 1981



ACKNOWLEDGEMENTS

This study was conducted under the supervision of Prof.Dr.İ.Kavrakoğlu and Dr.Erkut Yucaoğlu to both of whom I wish to express my sincere gratitude for their valuable guidance and help throughout the course of the research. I would like to thank Dr.Erkut Yucaoğlu once more for providing stimulating hours of discussion on so many occasions.

My heartfelt thanks go to all my friends who have helped me in innumerable ways.

My special thanks are due to Mrs. Nihal Yener for the meticulous and patient typing of this dissertation.

I would also like to thank my husband for his understanding and encouragement throughout my studies.

A B S T R A C T

This study deals with multiobjective linear and integer linear programming.

An algorithm for generating all efficient extreme points of multiobjective linear programs, or a subset of them corresponding to a decision maker specified space of objective weights, is given. The algorithm utilizes, with some modifications, earlier results given by various authors.

An original algorithm for bicriterion linear programs which requires only a series of divisions and comparisons for determination of adjacent efficient extreme points is also presented.

The developed algorithms are used in an application in power systems planning where efficient decision alternatives are generated to be presented to the decision makers.

A branch and bound algorithm which is based on extension of implicit enumeration techniques to multiobjective zero-one linear programming and which appears to be computationally quite efficient is also developed.

ÖZET

Bu çalışma çok amaçlı doğrusal ve tamsayılı doğrusal programlama konuları ile ilgilidir.

İlk olarak, çok amaçlı doğrusal programlama problemleri için bütün baskın uç noktaları, ya da karar verici tarafından belirlenen bir amaç işlevi ağırlıkları uzayına karşı gelen baskın uç noktalar altkümesini saptayan bir algoritma verilmiştir. Bu algoritma çeşitli araştırmacılar tarafından evvelce verilmiş sonuçların sentezine ve bazı yeni gözlemlere dayanmaktadır.

Bu çalışmada ayrıca iki amaçlı doğrusal programlama problemlerine yönelik ve komşu baskın uç noktaların bulunması için yalnızca bir dizi bölme ve karşılaştırma gerektiren yeni bir algoritma geliştirilmiştir.

Geliştirilen algoritmalar, enerji sistemleri planlaması alanında yapılan bir uygulamada, karar vericilere yardımcı olacak şekilde etkin karar seçenekleri oluşturmak amacıyla kullanılmıştır.

Çok amaçlı 0-1 doğrusal programlama problemleri için ise, gereken hesaplama süresi açısından oldukça etkin görünen bir dallandırıp-sınırlama algoritması geliştirilmiştir.

LIST OF TABLES

		<u>Page</u>
TABLE	I.1. Sample Problem Results	50
	I.2. Sample Problem Results	58
TABLE	II.1. Ranking of Power Plants	66
	II.2. Solution Summary Statistics	70
	II.3. Summary Statistics for Scenario B Solutions	72
	II.4. Summary Statistics for Scenario C Solutions	73
	II.5. Summary Statistics for Scenario D Solutions	74
	II.6. Efficient Solutions and Associated Representative Weights λ_i for Scenario C	78
	II.7. Computation Times for Model Application	79
	II.8. Efficient Solutions For Bicriterion Problem (Scenario D)	80
TABLE	III.1. Approaches to MOILP	84
	III.2. Comparison of MDM and MI Criteria for Branching	111
	III.3. Comparison of MDM, MI and Mixed Branching Criteria with Application of Tests II, III and IV	113
	III.4. Results for MI Criterion with Application of Tests II and III	113
	III.5. Computational Results	118

LIST OF FIGURES

		<u>Page</u>
FIGURE	I.1. The Cutting Hyperplane	31
	I.2. The Mechanics of the Algorithm	37
	I.3. Program Structure	41
	I.4. i) Flow Diagram of the MAIN Program	46
	ii) Flow Diagram of Subroutine EDGE and Subroutines SCAN and DROP.	49
FIGURE	II.1. The Overall Solution Procedure	64
	II.2. Solution Steps	69
	II.3. Power Expansion Alternatives for Scenario B	75
	II.4. Power Expansion Alternatives for Scenario C	76
	II.5. Power Expansion Alternatives for Scenario D	77
	II.6. Trade-off Function Between Economic Cost and Environmental Impact	80
FIGURE	III.1. Parametric Scalarization May Not Generate All Efficient Integer Solutions	83
	III.2. Rudimentary Multiobjective Implicit Enumeration Algorithm	103
	III.3. Flow Diagram of Multiobjective Implicit Enumeration Algorithm	117

TABLE OF CONTENTS

	<u>Page</u>
ACKNOWLEDGEMENTS	i
ABSTRACT	ii
ÖZET	iii
LIST OF TABLES	iv
LIST OF FIGURES	v
INTRODUCTION	1
PART I. MULTIOBJECTIVE LINEAR PROGRAMMING	10
I.1. LITERATURE SURVEY ON MULTIOBJECTIVE LINEAR PROGRAMMING	10
I.1.1. Generating Approaches	11
I.1.2. Interactive Approaches	27
I.1.3. Approaches to Bicriterion Linear Programming . .	30
I.2. AN ALGORITHM FOR GENERATING ALL OR A RELEVANT SUBSET OF EFFICIENT EXTREME POINTS	32
I.2.1. Observations on the Efficiency Check	33
I.2.2. Development of the Algorithm	36
I.2.3. The Computer Program and Computational Results .	41
I.3. AN ALGORITHM FOR BICRITERION LINEAR PROGRAMMING	44
I.3.1. Observations on the Efficiency Check	51
I.3.2. Development of the Algorithm	54
I.3.3. The Computer Program and Computational Results .	56
PART II. AN APPLICATION IN POWER SYSTEMS PLANNING	59
II.1. MODELLING THE POWER SYSTEM EXPANSION PROBLEM	60
II.2. SOLUTION PROCEDURE	63
II.2.1. Application of the Model with Three Objectives .	65
II.2.2. Application of the Model with Two Objectives . .	79
II.3. EVALUATION OF RESULTS	81

PART III. MULTIOBJECTIVE INTEGER LINEAR PROGRAMMING	82
III.1. MULTIOBJECTIVE INTEGER LINEAR PROGRAMMING METHODS . . .	84
III.2. AN ALGORITHM FOR MULTIOBJECTIVE ZERO-ONE LINEAR PROGRAMMING	96
III.2.1. Review of Implicit Enumeration	96
III.2.2. Development of a Multiobjective Implicit Enumeration Algorithm	100
III.2.2.1. Domination Tests and Alternative Branching Criteria	104
III.2.2.2. Experimentation on Branching Criteria and Use of Tests	110
III.2.3. The Final Algorithm and Computational Results	114
III.2.4. Remarks and Extensions on the Algorithm . . .	119
CONCLUSION	122
REFERENCES	125
APPENDIX I. COMPUTER PROGRAM FOR MULTIOBJECTIVE LINEAR PROGRAMMING	132
APPENDIX II. COMPUTER PROGRAM FOR BICRITERION LINEAR PROGRAMMING .	156
APPENDIX III. DATA FOR THE POWER SYSTEMS EXPANSION MODEL AS APPLIED TO TURKEY	168
APPENDIX IV. COMPUTER PROGRAM FOR MULTIOBJECTIVE ZERO-ONE LINEAR PROGRAMMING	170

INTRODUCTION

In the decision making environment, tradition requires the adoption of a single objective function, although the multiplicity of objectives in many situations can readily be recognized. In some cases this is justifiable because the various objectives can be subsumed under one general objective. However, in most real-life systems the several conflicting criteria entering into the evaluation of system performance cannot be combined satisfactorily to give a single performance measure. In complex industrial and economic problems, a general agreement on the constituents of good criteria for evaluating the performance of the system under consideration is difficult to attain. For such problems multiobjective analysis, which makes the possible trade-offs between the objectives explicit, could be of great help in arriving at the "preferred" solution.

The term "preferred" is used here because, in multiobjective optimization problems, generally, the set of all feasible solutions is only partially ordered with respect to the objectives and hence a single optimal solution cannot be found. Rather a set of efficient (non-inferior, Pareto-optimal) solutions, that is, a set of solutions for which no other solution exists which is better than the members of this set with respect to all the objectives, are generated. Only through the introduction of value judgements of the decision maker a complete ordering which is characteristic of single objective optimization problems can be achieved and a best solution can be identified.

Consideration of several different criteria in an optimization process has its origin in economics, particularly in the development of utility theory. The problem of the formation of a single optimality criterion which takes into account all the conflicting elementary criteria was first considered by Pareto [44] in his definition of maximum opheimity or maximum utility:

"We will say that the members of a collectivity enjoy maximum ophelimity in a certain position when it is impossible to find a way of moving from that position very slightly in such a manner that the ophelimity enjoyed by each of the individuals of that collectivity increases or decreases. That is to say, any small displacement in departing from that position necessarily has the effect of increasing the ophelimity which certain individuals enjoy and decreasing that which others enjoy, of being agreeable to some and disagreeable to others".

However Pareto himself had some difficulties in stating his definition of maximum ophelimity. One criticism has been that if the attained utility benefits cannot be decreased, then one could just as well speak of a minimum ophelimity. Also it should be noted that Pareto gave only what now could be the definition of local Pareto-optimality.

The concept of Pareto-optimality found its way into operations research through the work of Koopmans [39] on the activity analysis of production and allocation. Let y_n , $n=1, \dots, N$ denote the total net output of the n th commodity; let x_k , $k=1, \dots, K$ denote the selected amount or level of the k th activity where $x_k \geq 0$; and let a_{nk} denote the rate of flow per unit time of commodity n involved in the unit amount of activity k . One then has:

$$y = Ax, \quad \text{with} \quad A = \begin{bmatrix} a_{N1} & \cdot & \cdot & \cdot & \cdot & a_{1k} \\ \vdots & & & & & \vdots \\ a_{N1} & \cdot & \cdot & \cdot & \cdot & a_{Nk} \end{bmatrix},$$

where A is called the technology matrix. Among the commodities y_n , some are primary factors, some final commodities, and some are intermediate commodities. A point y in the commodity space is "possible" in a technology A if there exists a point $x \geq 0$ in the activity space with $y=Ax$.

Koopmans then states:

"A possible point in the commodity space is called efficient whenever an increase in one of its coordinates (the net output of one good) can be achieved only at the cost of a decrease in some other coordinate".

Let (A) denote the possible point set, a convex polyhedral cone spanned by the technology matrix A . Then, Koopmans states the following theorem:

THEOREM 1. A necessary and sufficient condition that a possible point $\hat{y} \in (A)$ be efficient is that \hat{y} possess a positive normal p to (A) defined by

$$p^T A \leq 0, \quad p^T \hat{y} = 0, \quad p > 0. \quad 1$$

Koopmans interprets the vector p as a vector of prices of the commodities in the point \hat{y} , and then states that a necessary and sufficient condition for \hat{y} to be efficient is that there exists a vector p of prices such that no activity in the technology permits a positive profit and such that the profit on all activities carried out at a positive level be zero.

A discussion of Koopmans' paper along with a numerical algorithm for the computation of efficient activities based on parameterization over the prices was presented by Charnes and Cooper [9].

A more general approach, the vector maximization problem of mathematical programming, was given by Kuhn and Tucker [40]. They defined the vector maximum problem as:

DEFINITION 1. The vector maximum problem is to find an x_0 that maximizes the vector function $G(x)$ constrained by $F(x) \geq 0, x \geq 0$; that is, to find an x_0 satisfying the constraints and such that $G(x) \geq G(x_0)$ for no x satisfying the constraints.

¹ Here, and throughout the text the notation used is as follows:

For $x, y \in R^n$

$$\begin{aligned} x \geq y & \text{ iff } x_i \geq y_i, & \forall i \in I = \{1, \dots, n\}; \\ x \geq y & \text{ iff } x \geq y \text{ and } x \neq y; \\ x = y & \text{ iff } x_i = y_i, & \forall i \in I \\ x > y & \text{ iff } x_i > y_i, & \forall i \in I \end{aligned}$$

The authors then restricted themselves to the so-called proper solutions based on the usual Kuhn-Tucker constraint qualifications, and gave the following theorems.

THEOREM 2. In order that x_0 be a proper solution of the vector maximum problem, it is necessary that there be some $v_0 > 0$ such that x_0 and some u_0 satisfy conditions (a) and (b) for

$$L(x,u) = v_0^T G(x) + u^T F(x)$$

where conditions (a) and (b) are the necessary conditions for $L(x,u)$ to have a saddle point at (x_0, u_0) .

$$(a) \quad L_x^0 \leq 0 \quad L_x^{0T} x_0 = 0 \quad x_0 \geq 0,$$

$$(b) \quad L_u^0 \geq 0 \quad L_u^{0T} u_0 = 0 \quad u_0 \geq 0.$$

Here L_x^0 and L_u^0 denote the partial derivatives evaluated at the particular point (x_0, u_0) .

THEOREM 3. In order that x_0 be a proper solution of the vector maximum problem, it is sufficient that there be some $v_0 > 0$ such that x_0 and some u_0 satisfy the above conditions (a) and (b) and condition (c) for

$$L(x,u) = v_0^T G(x) + u^T F(x)$$

where condition (c) is given by

$$(c) \quad L(x, u_0) \leq L(x_0, u_0) + L_x^{0T} (x - x_0).$$

THEOREM 4. Let $F(\cdot)$ and $G(\cdot)$ be concave as well as differentiable for $x \geq 0$. Then, x_0 is a proper solution of the vector maximum problem if and only if there is some $v_0 > 0$ such that x_0 and some u_0 give a solution of the saddle-point problem for

$$L(x,u) = v_0^T G(x) + u^T F(x)$$

Although Kuhn and Tucker introduced the vector maximum problem as early as 1951, multiobjective analysis did not gain immediate popularity. A different approach to multiobjective problems, goal programming, was described by Charnes and Cooper [10] about a decade later. The method is based on minimization of weighted absolute deviations from targets for each objective specified by the decision maker.

Along with increased proficiency in optimization techniques the vector maximum problem was again approached by various researchers from the general viewpoint of Kuhn and Tucker. Zadeh [60] suggested that if the objective function space is convex, the set of all efficient solutions can be found by solving a parametric scalar optimization problem, as implied by the Kuhn-Tucker conditions:

$$\max_{x \in X} v^T G(x) \quad v \geq 0$$

where X is the feasible region.

Equivalence of the vector optimization problem to a parametric scalar optimization problem was also investigated by Da Cunha and Polak [15] and Geoffrion [28]. An algorithm for maximizing two objective functions via parametric linear programming was given by Geoffrion [27].

Another implication of Kuhn-Tucker conditions is that one of the objectives can be taken as the primary one, and the others can be treated as constraints. In the constraint method developed by Haimes [32] the following problem is solved:

$$\begin{aligned} \max_{x \in X} & \quad g_r(x) \\ \text{s.t.} & \quad g_k(x) \geq L_k \quad \forall k, k \neq r \end{aligned}$$

where L_k is a parametric lower bound on objective k . Haimes states that any solution to the above problem where the multipliers related to the objectives appearing as constraints are all nonzero is an efficient solution.

An advantage of the above constraint method over weighting objectives or

The parametric scalarization method is that, even in the existence of a quality gap, all efficient solutions can be generated, whereas in such cases parametric scalarization fails to generate all efficient solutions.

In the last 15 years, the field of multiobjective optimization has been continually developing. Several different approaches which have been considered are categorized as follows by Cohon and Marks [13].

- i) A priori articulation of preferences and generating a single relevant solution.
- ii) Generating efficient solutions and then selecting the preferred solution from among these, by subjective evaluation.
- iii) Progressive articulation of preferences and arriving at the preferred solution in an interactive manner.

Each of these approaches has its advantages and disadvantages. The first approach requires the least computational effort in that it generates the preferred solution directly. But it places considerable burden on the decision maker by forcing him to articulate his preferences in an information void. The formation of the decision maker's preferences and his valuation system depends on the realizable levels of achievement of the objectives and can neither be expected to be complete nor definite at the start of the decision process. In the second approach, which is at the other extreme, the decision maker is presented with complete information on the possible levels of achievement and the available trade-offs between the objectives. However, this information may prove to be "too much". Usually a large number of efficient solutions are generated which are difficult to display and which could inhibit the ability of the decision maker to perform the final selection process. Also the computational effort associated with generating a large number of efficient solutions may be prohibitive. The last approach tries to make a compromise by embedding the decision maker within the solution process so that his implicit valuation system directs the process. But then the solutions depend on the accuracy of the local preference that the decision maker can indicate and on his complete consistency. Generally, there is no guarantee that the preferred solution can be obtained within a finite number

of interactive cycles. Also a continuing cooperation between the analyst and the decision maker is required. This may be difficult to achieve since several decision makers may be involved and they may not be accessible at all times.

These considerations indicate that in any application, problem specific as well as decision maker specific aspects will determine the structure of the best suited approach.

Numerous examples of each of these approaches can be found in the literature. Investigation of the functional relationship between the efficient set and the weighting vector was carried out by Reid and Vemuri [49], and Reid and Citron [50]. Beeson and Meisel [4] presented a computational algorithm for obtaining a characteristic set of efficient solutions for nonlinear problems through an adaptive search procedure. Lin [42] suggested the method of proper equality constraints, where all but one of the objectives are converted into equality constraints and conditions for identifying the proper equality constraints are developed. Payne et al [46] gave an algorithm for bicriteria optimization based on treating one of the objectives as a constraint and indicated an extension to the case of three objectives. The goal attainment method proposed by Gembicki and Haimes [25] is quite similar to goal programming. Specialized techniques for multiobjective linear programming based on the simplex method have been developed by Yu and Zeleny [59], Evans and Steuer [18], Isermann [34], and Ecker and Kouada [16].

Various interactive techniques utilizing preference information of the decision maker have been proposed. The Electre method developed by Roy [52] is mostly concerned with building outranking (preference) relationships from value judgements supplied by the decision maker. In the surrogate worth trade-off method presented by Haimes and Hall [33], trade-off functions between a primary objective and each of the other objectives are developed and through the use of surrogate worth functions the preferred solution within the indifference region of the decision maker is chosen. A variety of interactive techniques which are iterative in nature have been developed. In these approaches an efficient solution is generated, the decision maker's reactions to this solution are tested, the problem is modified accordingly, and the process is repeated until the decision maker is satisfied or some

other termination criterion is effective. The step method or STEM proposed by Benayoun et al [6], the algorithm of Belenson and Kapur [5], and the method of Zions and Wallenius [62] are examples of such approaches.

In recent years research in multiobjective integer programming problems has also been carried out. Shapiro [53] suggests use of integer programming duality theory. Bowman [8] has proposed a generalized Tchebycheff norm to parametrically generate all efficient solutions. Zions [63] has worked on extending the Zions-Wallenius method to integer problems. Bitran [7] has proposed a method for multiobjective zero-one integer programming problems based on determining the efficient solutions for the unconstrained problem and the directions of preference along which all objectives can be increased. Klein and Hannan [38] have worked on an implicit enumeration based algorithm for multiobjective zero-one problems consisting of solving a sequence of continually more constrained single objective function problems.

Apart from theoretical results and algorithms, several applications have also been reported in the literature. Those include a study by Cohon and Marks [12] on a water resource development and allocation problem, another study by Geoffrion et al [29] in applying an interactive approach to the operation of an academic department, an application to forest management by Steuer and Schuler [56], a study on formulating macroeconomic policy decisions in Finland by Wallenius et al [58], and a study in energy planning by Zions and Deshpande [64]. Macroeconomic policy problems and public investment problems such as design of water resource systems and urban transportation systems which are inherently multiobjective in nature are potential areas for other applications.

Still, there is continuing need and effort for increasing the computational efficiency associated with multiobjective optimization problems. Increased computational efficiency will increase the applicability of multiobjective analysis.

The growing interest in multiobjective optimization and the significant advances in this relatively new field, discussed above have provided the motivation for this dissertation. After a detailed consideration of the

state of the art, certain specific problems have been focused on. Through modifications and development of syntheses, some computationally efficient algorithms have been devised.

The text of this study is presented in three parts. In part one multiobjective linear programming and the special case of bicriterion linear programming are considered and new algorithms for generating either all or a relevant subset of efficient extreme points are given. In part two applications in power systems planning are presented. Part three is on multiobjective integer linear programming and an algorithm for multiobjective zero-one linear programming problems is developed.

PART ONE: MULTIOBJECTIVE LINEAR PROGRAMMING

The multiobjective linear programming (MOLP) problem is formulated as

$$\begin{aligned} \max_{x \in X} \quad & G_x \\ X = \{x \mid & Ax \leq b, \quad x \geq 0\} \end{aligned}$$

where G is a $p \times n$ matrix whose rows G_i , $i=1, \dots, p$ represent the different objective functions, A is an $m \times n$ matrix and x and b are n and m vectors respectively as in a standard linear programming problem; and maximization refers to determination of efficient solutions. This problem is alternatively termed the linear vector maximization problem in the literature.

I.1. LITERATURE SURVEY ON MULTIOBJECTIVE LINEAR PROGRAMMING

Several approaches to the MOLP problem have been given by various authors. Some concentrate on generating all efficient extreme points [16], [18], [19]; others also consider generating all efficient solutions which consist of the faces of the convex polyhedron X [17], [22], [34], [61]. Some others present only a subset of efficient extreme points to the decision maker [43], [54], [55], [57]. Several interactive approaches have been designed aiming at generating directly the efficient point which is the decision maker's preferred solution [5], [6], [62]. Some authors have considered the special case of bicriterion linear programming and given specialized algorithms [2], [14], [27]. Here, previous results in these areas will be reviewed.

First, let us introduce some notation common for linear problems. Suppose x^0 is a basic feasible solution with associated basis B . By renumbering variables as necessary, and partitioning A and G we have

$$x_B = B^{-1}b - B^{-1}N x_N \quad C = G_B B^{-1}N - G_N \quad Y = B^{-1}N$$

where the subscript B denotes basic and subscript N denotes nonbasic, C is the current reduced cost matrix and Y is the current constraint coefficient matrix. Furthermore, let $C_i(Y_i)$ denote the i th row and $C^k(Y^k)$ denote the k th column of C (Y). Also, e will denote a vector of suitable length with each component equal to one and the transpose of a matrix or vector is denoted by the superscript T. In the following, it is assumed that the convex polyhedron X forming the feasible region is bounded.

I.1.1. GENERATING APPROACHES

Generating approaches to the MOLP problem are based on several considerations from the computational theory of linear programming. The pivot selection rules in linear programming are designed to produce sequences of basic feasible solutions. Every vertex of X may be described in terms of at least one basic feasible solution so that enumerating basic feasible solutions allows one to enumerate vertices of X. Detection of those vertices of X which are efficient is accomplished by various efficiency checks which take the form of linear programming subproblems. Through further refinements and computations detection of efficient faces of X can be accomplished. Here a comparative and critical review of the results given by various authors will be presented.

Philip [48] considered the linear vector maximization problem and gave the following result for checking the efficiency of an arbitrary feasible point.

THEOREM I.1. A point x^0 is efficient if and only if there exists $\mu_i \geq 0$, $i \in I$ and $\lambda_j \geq 0$ such that

$$\sum_{i \in I} \mu_i A_i = \sum_j \lambda_j G_j \quad .$$

Here I is the index set of active constraints at x^0 , including the nonnegativity constraints. This theorem corresponds to the Kuhn-Tucker theorem for the general case where linearity is not assumed. Based on this theorem, the following subproblem is to be used for determining whether a given point x^0 is efficient:

$$\begin{aligned} \min \quad & z = e^T s + e^T t \\ \text{s.t.} \quad & Gv - A_I u + s - t = -Ge \\ & v \geq 0 \quad u \geq 0 \quad s \geq 0 \quad t \geq 0 \end{aligned}$$

Here A_I is the matrix formed by the set of active constraints at x^0 . And choosing $\alpha_j = 1 \quad \forall j$, the transformation $\lambda = v + e$ has been made. Then, the point x^0 is efficient if and only if $z_{\min} = 0$ in this subproblem.

Alternatively, Philip stated that Theorem I.1. implies that x^0 is an efficient extreme point if and only if there exists $\lambda > 0$ such that $\lambda^T C \geq 0$ where C is the reduced cost matrix as defined previously. This condition is equivalent to requiring $z_{\min} = 0$ in the following subproblem, where z gives the sum of artificial variables.

$$\begin{aligned} \min \quad & z = e^T t \\ \text{s.t.} \quad & C^T v - s + t = -C^T e \\ & v \geq 0, \quad t \geq 0, \quad s \geq 0 \end{aligned}$$

Philip mentioned that once at an efficient point x^0 , the artificial variables t_k are equal to zero and the variables s_k are the reduced costs for the objective function $(v + e)^T G x$. Consequently he stated that if some $s_k = 0$, the corresponding x_k could be made basic in the main problem and another efficient point could be obtained. He suggested solving this subproblem:

$$\begin{aligned} \min \quad & s_k \\ \text{s.t.} \quad & -C^T v + s = C^T e \\ & u \geq 0, \quad s \geq 0 \end{aligned}$$

If the minimum value of s_k is zero, then another efficient point is obtained by making x_k basic.

Thus, some basic results concerning the efficiency of a given point x^0 and indications on how to obtain other efficient extreme points once an efficient extreme point is found were given by Philip.

Zeleny [61] considered the MOLP problem in detail and gave two different approaches. In the first, he adapted the decomposition of parametric space approach originally developed by Gal and Nedoma [23] for multiparametric linear programming to a multiobjective linear programming context. Considering the multiparametric linear program

$$\max_{x \in X} \lambda^T Gx, \quad \lambda \in \Lambda \quad (P_\lambda)$$

the decomposition of the parametric space Λ means identification of a finite number of polyhedra $\Lambda(x^k)$ to be associated with the different extreme points x^k of the problem in question so that $\bigcup_k [\Lambda(x^k) \cap \Lambda] = \Lambda$. Here $\Lambda(x^k)$ is given by $\Lambda(x^k) = \{\lambda \mid \lambda^T C(k) \geq 0\}$ where $C(k)$ is the reduced cost matrix corresponding to the solution x^k . Defining $\Lambda = \{\lambda \mid \lambda_i \geq 0, \sum_i \lambda_i = 1\}$ Zeleny [61] states that the set of efficient points can be found by solving P_λ for all $\lambda \in \Lambda$. He gives the following results, considering a nondegenerate problem.

THEOREM I.2. Maximization of P_λ for all $\lambda \in \Lambda$ produces a finite covering of Λ .

THEOREM I.3. The set of efficient extreme points is a "connected" set. i.e. it is possible to reach any efficient extreme point from any other efficient extreme point by passing through only efficient extreme points.

These two theorems insure that the decomposition is finite and connected. Starting at an efficient point x^0 , the goal is to construct all polyhedra adjacent to $\Lambda(x^0)$ which do not have an empty intersection with Λ . Such polyhedra $\Lambda(x^i)$ correspond to nondominated or efficient solutions. When $\bigcup_i [\Lambda(x^i) \cap \Lambda] = \Lambda$ decomposition is complete. The following theorem checks adjacent polyhedra.

THEOREM I.4. Let x^0 be an efficient basic feasible solution. Let $H_k = \{\lambda \mid \lambda^T C^k = 0\}$. If $H_k \cap \Lambda(x^0) \cap \text{Int } \Lambda \neq \emptyset$ then introducing x_k into the basis leads to an efficient basic feasible solution.

That is, introducing nonbasic variables x_k , corresponding to nonredundant constraints of $\Lambda(x^0)$, into the basis leads to efficient solutions. A constraint is called nonredundant if and only if there exists a vector λ

such that the constraint holds as an equality. However, this is stated only as a sufficient but not necessary condition.

Zeleny enumerates a number of difficulties associated with this approach. Introducing into the basis nonbasic variables corresponding to redundant constraints does not necessarily lead to dominated solutions. In case of degeneracy, the one-to-one correspondence between an extreme point x^j and $\Lambda(x^j)$ can be destroyed, which implies that although all efficient extreme points have already been discovered Λ may not be fully decomposed. Also, in case of alternative solutions for which $\Lambda(x^i) = \Lambda(x^j)$ although Λ has been fully decomposed, all efficient extreme points may not have been enumerated. Also, Zeleny stated that efficient procedures for identifying nonredundant constraints were not available. Because of these difficulties, the decomposition approach was considered inefficient and disfavored by Zeleny. Instead, he developed an alternative approach, which he called the multicriteria simplex method. This second approach is based on the following theorem.

THEOREM I.5. \bar{x} is an efficient solution if and only if $z_{\max} = 0$ in the following LP problem

$$\begin{aligned} \max_{x \in \tilde{X}} \quad & z = e^T s \\ \tilde{X} = \quad & \{(x, s) \mid x \in X \quad Gx - s \geq G\bar{x}, \quad s \geq 0 \} \end{aligned}$$

This theorem actually follows from the definition of the efficient solution. Assuming we are at an efficient extreme solution, to check the efficiency of an adjacent extreme point x^j the above LP problem is solved with $\bar{x} = x^j$. This is repeated for all adjacent extreme points and those which are efficient are identified. Proceeding in this fashion all efficient extreme points will be identified since they form a connected set.

All convex combinations of efficient extreme points are not necessarily efficient. Thus, the set of all efficient points is not simply defined by the set of efficient extreme points. However, the set of all efficient points can be given as the union of efficient faces of the polyhedron X forming the feasible region.

Yu and Zeleny [59] consider a face of X defined as $F(I) = \{x \in X \mid A_i x = b_i \quad i \in I\}$ where $I \subseteq \{1, \dots, m, \dots, m+n\}$ and the nonnegativity constraints have been incorporated into the matrix A . They focus on those faces $F(I)$ for which I is maximal and systematically consider faces of decreasing dimension. They remark that the verification procedure to check all possible faces may be a prohibitive job and they develop some results that reduce the number of faces to be considered. To initiate their procedure, they construct the incidence matrix between the set of all efficient extreme points and the set of all $(n-1)$ dimensional faces. Using information from this incidence matrix and from the solutions of certain linear systems, the set of maximal efficient faces of dimension $n-1$, if any, is determined. The procedure continues by considering faces of dimension $n-2$ and lower at each step.

At any step, the face having the greatest number of incident efficient vertices is processed first. The system of linear equations

$$\sum_{i=1}^P \lambda_i G_i = \sum_{i \in I} u_i A_i$$
 where I denotes the index set of active constraints, i.e. examined for a nontrivial solution where $\lambda_i > 0, i=1, \dots, P$. If there is such a solution, the face considered is efficient, i.e. all points on the convex hull of its incident efficient vertices are efficient. Then, the column corresponding to that face is eliminated from the incidence matrix. Furthermore, any face whose incident efficient vertices is a subset of the incident efficient vertices of the newly found face is disregarded as it is not maximal, and the corresponding column is also eliminated. At some step, the incidence matrix will become vacuous, i.e. no more columns will remain and the procedure stops, with all efficient faces being determined.

Evans and Steuer [18] gave a revised simplex algorithm for the enumeration of the set of all efficient extreme points. They gave the following conditions for checking the efficiency of given solutions.

THEOREM I.6. A point $x^0 \in X$ is efficient if and only if there is a $\lambda > 0$ such that x^0 is optimal for P_λ

$$\max_{x \in X} \lambda^T Gx \quad (P_\lambda)$$

THEOREM I.7. A basic feasible solution x^0 is efficient if and only if there doesn't exist $u \geq 0$ such that

$$C u \leq 0 \quad Y_i u \leq 0 \quad i \in Q$$

where $Q = \{i | x_{B_i}^0 = 0\}$.

Intuitively, if $Cu \leq 0$ for some $u \leq 0$, then by raising some combination of nonbasic variables to a positive level, we can increase the values of all objectives. The condition $Y_i u \leq 0$, $i \in Q$ is introduced to be able to detect the efficiency of a degenerate basic feasible solution. Even if $Cu \leq 0$ for some $u \geq 0$, if $Y_i u \leq 0$ is not satisfied, the basic feasible solution is still efficient.

In checking the efficiency of an extreme point adjacent to a given efficient extreme point the following result is to be used for nondegenerate problems.

THEOREM I.8. Let x^0 be an efficient extreme point and let x_j be a nonbasic variable in the basic feasible solution associated with x^0 . Then, the adjacent extreme point with x_j a basic variable is efficient if and only if the following problem is consistent and bounded.

$$\begin{aligned} \max \quad & e^T s \\ \text{s.t.} \quad & Cu - C^j w + s = 0 \\ & u \geq 0, \quad s \geq 0, \quad w \text{ scalar} \end{aligned}$$

where C^j is the column of C corresponding to x_j .

However as later shown by Ecker and Kouada [16], the condition given by the theorem is not necessary for the efficiency of an adjacent vertex, though it is necessary for the efficiency of the edge connecting the two adjacent vertices. In general, two extreme points of X may be efficient and yet the edge connecting these two extreme points may not be efficient, as illustrated by the example in [16].

Evans and Steuer stated that degeneracy presents special problems because then the number of extreme points adjacent to a given extreme point exceeds the number of nonbasic variables. In these circumstances, they suggested

employing special procedures to insure examination of each adjacent extreme point. In another paper [19], they described two methods that explicitly deal with degeneracy. The first method employs an algorithm of Chernikova [11] for generating all edges emanating from a given efficient extreme point x^0 which is degenerate. Thus, it is insured that all adjacent extreme points are checked. Each of the edges generated by the Chernikova procedure yields a direction \bar{u} to be tested for efficiency by the condition given in the following theorem.

THEOREM I.9. Let x^0 be an efficient extreme point and let \bar{u} be a feasible direction at x^0 . Then \bar{u} is an efficient direction at x^0 if and only if there doesn't exist a feasible direction u such that $Cu \leq C\bar{u}$.

The second method is called the adjacent efficient basis procedure. Here a basis is defined to be an efficient basis if and only if $z_{\max} = 0$ in the LP

$$\begin{aligned} \max \quad & z = e^T s \\ \text{s.t.} \quad & Cu + s = 0 \\ & u \geq 0 \quad s \geq 0 \dots \end{aligned}$$

Here $z_{\max} = 0$ if and only if there doesn't exist u such that $Cu \leq u$. The authors then state that the set of efficient bases is connected and give a slightly different version of the subproblem of Theorem I.8. for identifying adjacent efficient bases.

THEOREM I.10. Assume we are at an efficient basis. Then an adjacent basis obtained by making nonbasic variable x_j basic is an efficient basis if and only if $z_{\max} = 0$ in the LP

$$\begin{aligned} \max \quad & z = e^T s \\ \text{s.t.} \quad & Cu - C^j w + s = 0 \\ & u \geq 0, \quad s \geq 0, \quad w \geq 0, \quad w \text{ scalar} \end{aligned}$$

Another result is that each efficient extreme point has at least one efficient basis associated with it. Thus, the procedure terminates when all efficient bases are located.

The "efficient basis" concept of Evans and Steuer is also used by

Isermann [34], however with a different terminology. A basic solution is said to be "dual feasible" if and only if there doesn't exist $u \geq 0$ such that $Cu \leq 0$. Then, by Gale's theorem of the alternative, the system $\lambda^T C \geq 0$, $\lambda > 0$ has a solution λ ; and hence the term dual feasible. Theorem I.6. implies that in case of degeneracy at least one of the degenerate efficient basic solutions, which represent an extreme point of X in common, is dual feasible. Thus determination of all dual feasible basic solutions is adequate for determining all efficient extreme points.

Let $N = \{j | C^j \geq 0\}$ denote the index set of potential pivot columns at a given dual feasible solution. Those $j \notin N$ are not considered since their introduction into basis leads to basic solutions which are not dual feasible. Isermann defined two dual feasible solutions x^1 and x^2 as adjacent if and only if they are obtained from one another by a single pivot and each $\bar{x} = \alpha x^1 + (1-\alpha)x^2$, $0 \leq \alpha \leq 1$ is efficient. Then he gave the following theorem for identification of adjacent dual feasible basic solutions.

THEOREM I.11. Let x^0 be a dual feasible basic solution and let P be a nonempty subset of N . Consider the LP

$$\begin{aligned} \max \quad & e^T s \\ \text{s.t.} \quad & Cs + s = e \\ & s \geq 0 \\ & u_i \geq 0 \quad \forall i \in N - P. \end{aligned}$$

Then

- i) If the LP has an optimal solution, then to each $r \in P$ there corresponds a dual feasible basic solution which is adjacent to x^0 .
- ii) Let x^1 be an adjacent dual feasible basic solution obtained by introducing x_r into basis. Then there exists some $P \subseteq N$ with $r \in P$ such that the LP has an optimal solution.

Next, the following theorem is given.

THEOREM I.12. Let $E = \{x^i | x^i \text{ is a dual feasible basic solution}\}$, $L = \{(x^i, x^j) | x^i \text{ and } x^j \text{ are adjacent dual feasible basic solutions}\}$. The undirected graph $G = (E, L)$ is finite and connected.

Thus, this connectedness property and the LP subproblem given in Theorem I.11 form the basis for the solution procedure consisting of enumeration of all dual feasible basic solutions. Further work is needed if the set of all efficient points is to be generated. Referring to Theorem I.11., all maximal index sets P have to be identified at each dual feasible basis, by successively adjusting the sign restrictions on the variables u_i and checking all possible combinations of indices. However, when a maximal index set P^i has been found, all index combinations forming a subset of P^i can be deleted from further consideration, thus reducing the number of combinations to be checked. At the end of the procedure for enumerating all dual feasible bases, j , all associated maximal index sets P^{jk} will then be available. Next the index sets $Q^{jk} = P^{jk} \cup D^j$, where D^j is the index set of basic variables for dual feasible basis j , are constructed. In this procedure the same index set Q^{jk} may be constructed several times and subsets of some index set Q^{jk} may have been constructed as well. All such subsets and duplications are eliminated so that finally one has r index sets U^j . Now, to each index set U^j there corresponds a face of X . Specifically, let $I^j = \{i | D^i \subseteq U^j\}$, that is let I^j denote the set of efficient extreme points whose basic variable index sets are subsets of U^j , and let S^j be given by

$$S^j = \{x | x = \sum_{i \in I^j} \alpha_i x^i, \quad \alpha_i \geq 0 \quad \forall i \in I^j, \quad \sum_{i \in I^j} \alpha_i = 1\}.$$

Then all points $x \in S^j$ are efficient and the set of all efficient solutions has been decomposed into r convex subsets.

Gal [22] gave a method for determining the set of all efficient solutions based on his earlier works [21], [23] on multiparametric linear programming. A set of vectors $\lambda^i \geq 0$ are associated with each efficient vertex and later by inspection of these sets higher dimensional efficient faces are determined.

The nondegenerate case is considered first. Similar to that done by Isermann, two efficient vertices x^i and x^j are defined to be efficient neighbors if and only if they are adjacent and all $\bar{x} = \alpha x^i + (1-\alpha)x^j$, $0 \leq \alpha \leq 1$, are efficient, i.e. the edge connecting the two vertices is efficient. Then Gal states that the linear vector maximum problem generates a connected

undirected graph $G(T, \Gamma)$ where the node set T consists of efficient basic solutions and an arc exists between two nodes if and only if the corresponding basic solutions are efficient neighbors. Next he considers the system of linear equations L_i

$$\begin{aligned} -\lambda^T C(i) + s &= 0 \\ \lambda^T e &= 1 \\ \lambda &\geq 0, \quad s \geq 0 \end{aligned} \quad (L_i)$$

where $C(i)$ denotes the reduced cost matrix corresponding to vertex x^i , and states the following theorem.

THEOREM I.12. The vertices x^i and x^j are efficient neighbors if and only if there exists nondegenerate basic solutions to L_i and L_j such that

$$\lambda^* = \begin{bmatrix} \lambda_B^i \\ \lambda_N^i \end{bmatrix} = \begin{bmatrix} \lambda_B^j \\ \lambda_N^j \end{bmatrix} \geq 0$$

with $\lambda_B^i = \lambda_B^j > 0$ and $\lambda_N^i = \lambda_N^j = 0$ such that both x^i and x^j are solutions of P_λ for $\lambda = \lambda^* \geq 0$.

Then Gal gives the following linear program for determining whether by introducing x_k into basis an efficient neighbor to x^i is found.

$$\begin{aligned} \min \quad & s_k \\ \text{s.t.} \quad & -\lambda^T C(i) + s = 0 \\ & \lambda^T e = 1 \\ & \lambda \geq 0 \quad s \geq 0 \end{aligned}$$

Gal states that if and only if the minimum value of s_k is zero and s_k is a nonbasic variable, then by introducing x_k into basis an efficient neighbor is reached. However, this is only true if the corresponding basic solution of the subproblem is nondegenerate.

While finding all efficient neighbors to a given vertex, all corresponding λ 's are also determined and assigned to both the present vertex and the

efficient neighbor. Finally, each efficient vertex x^i will be assigned a finite set of vectors $Y^i = \{\lambda^{i1}, \lambda^{i2}, \dots, \lambda^{ik}\}$ for which it is optimal. Gal states that efficient vertices $x^j, x^{j+1}, \dots, x^{j+r}$ are the extreme points of an efficient face if and only if there exists λ^* which is common to all Y^i , $i=j, \dots, j+r$. Therefore, by collecting together those vertices which maximize P_λ for the same λ^* , all efficient faces can be identified.

Gal also discusses briefly the implications of degeneracy in the MOLP problem. The implication of degeneracy is the existence of more than one basis corresponding to the same vertex. Then there would exist subgraphs of graph G with each basis being a node of exactly one of these subgraphs. Gal then states that it should be sufficient to generate only one basis for each degenerate vertex, and the difficulties enumerated by Zeleny should not arise.

Another approach for enumerating all efficient extreme points is given by Ecker and Kouada [16], which is based on checking the efficiency of edges incident to an efficient extreme point. They define F^j to be the edge of X , the convex polyhedron forming the feasible region, incident to an efficient extreme point x^0 , obtained by increasing the single nonbasic variable x_j and adjusting the nonbasic variables to maintain feasibility. Then; they give the following theorem.

THEOREM I.13. Let x^0 be a nondegenerate efficient extreme point. Then F^j is efficient if and only if $z_{\max} = 0$ in the following linear program:

$$\begin{aligned} \max \quad & z = e^T s \\ \text{s.t.} \quad & Cu + s = C^j \\ & u \geq 0, \quad s \geq 0 \end{aligned}$$

This linear program is similar to that given by Evans and Steuer in Theorem I.10. Here the parameter w is taken as one and the efficiency check concerns the incident edge and not the adjacent extreme point. Then, considering the dual of the linear program above, the authors give another result.

THEOREM I.14. Let x^0 be a nondegenerate efficient extreme point. Then F^j is efficient if and only if s_j is nonredundant in the set S given by

$$S = \{(v, s) \mid -C^T v + s = C^T e, \quad v \geq 0, \quad s \geq 0\}$$

where s_j is said to be nonredundant in S if and only if there exists a point $(\bar{v}, \bar{s}) \in S$ such that $\bar{s}_j = 0$. The dual program has minimization of s_j over the set S as its objective.

Again, similarity with results of Zeleny and Gal can be observed. Requiring s_j to be nonredundant in S is in effect the same as requiring the corresponding j th constraint of $\Lambda(x^0)$ to be nonredundant. Zeleny was able to give constraint nonredundancy only as a sufficient condition, because he considered the adjacent extreme point. However, here consideration of the efficiency of the incident edge allows the condition to be stated also as a necessary condition. Requiring s_j to be nonredundant in S is also equivalent to requiring the minimum value of s_j to be zero in the linear subprogram given by Gal which is modified by requiring $\lambda > 0$, instead of $\lambda \geq 0$. As the system of inequalities defining the feasible region of the subprogram is homogenous, one can require $\lambda \geq e$ and through the transformation $\lambda = v + e$, $v \geq 0$, the feasible region may be transformed to correspond to the set S . Then minimum value of s_j is zero if and only if s_j is nonredundant in S .

Considering degeneracy, Ecker and Kouada give the following result:

THEOREM I.15. Let x^0 be a degenerate basic feasible solution. Then F^j , $j \in N^{ND}$, is efficient if and only if s_j is nonredundant in the set S_D given by

$$S_D = \{(v, y, s) \mid -v^T C^{ND} + s = e^T C^{ND}, \quad -v^T C - y^T Y_D \leq e^T C, \quad v, y, s \geq 0\}$$

where $N^{ND} = \{j \mid Y_D^j \leq 0\}$ and Y_D is the matrix consisting of the rows, i , of Y for which $x_{B_i}^0 = 0$; C^{ND} is the matrix of columns C^j for $j \in N^{ND}$ and s is indexed as the x_j^1 s for $j \in N^{ND}$.

Again, this result is derived based on a duality relationship. The linear program of Theorem I.13 is augmented by the constraints $Y_D u \leq 0$ and some observations on its dual indicate the above result. One remark due here is that the condition $-v^T C - y^T Y_D \leq e^T C$ can be considered as a requirement that an associated dual feasible basis, using the terminology of Isermann, exists.

The procedure given for identifying nonredundant variables s_j is again based on minimizing s_k over the set S or S_D . However, Ecker and Kouada note that a minimization problem for each s_j is not usually needed. Any s_j which is nonbasic, or basic with value zero in any basic feasible solution of S or S_D has a value zero and is obviously nonredundant. Also, if one can observe that any s_j can be made nonbasic through a single pivot, one can conclude nonredundancy of that s_j . A simple condition for labeling a variable s_j as redundant also exists. Assume a basic feasible solution of S or S_D is at hand. Also assume that s_k is a basic variable (with a positive value) in a given equation and that all the coefficients of the nonbasic variables in that row are nonpositive. Then, s_k is obviously redundant because it cannot be decreased below its current value in any feasible solution of S or S_D . Consequently, through the use of these checks, nonredundancy or redundancy of some of the s_j 's can be determined without solving the associated minimization problems.

Recently, Ecker, Hegner and Kouada [17] gave an algorithm for describing the set of all efficient points as the union of maximal efficient faces. The efficient edges and extreme points are found in the manner described by Ecker and Kouada. The procedure for finding higher dimensional faces has features similar to both Isermann's and Gal's approach. The authors use two characterizations of maximal efficient faces simultaneously. Both a maximal index set α and a vector λ are associated with each maximal efficient face.

With each maximal efficient face incident to a given efficient extreme point represented by a simplex tableau T , a maximal index set α is associated such that the given face is defined by

$$f(T, \alpha) = \{x \in X \mid x_j = 0 \text{ if } j \in N_T - \alpha\}$$

where N_T is the set of nonbasic variables associated with the tableau T . It should be noted that the characterization given by $f(T, \alpha)$ is similar to the characterization given by Q^i in Isermann's approach. In addition, a vector λ^l is associated with each maximal efficient face such that $f(T, \alpha)$ is the set of optimal solutions for P_λ with $\lambda = \lambda^l$. Again, it can be observed

that this feature is similar to Gal's approach. Furthermore, a set I^k whose elements correspond to the subscripts of the extreme points found to be incident to that face is also constructed. Finally, the entire efficient set is described as the union of maximal efficient faces each of which is given as the convex hull of its efficient extreme points, stored by the sets I^k .

The main difference from Isermann's or Gal's approach is that while in those approaches determination of maximal efficient faces forms a separate phase of the algorithm, here efficient extreme points and maximal efficient faces are generated simultaneously. The incomplete knowledge of efficient faces is used at each iteration so as to avoid regeneration of maximal efficient faces found previously.

In giving their results, the authors assume nondegeneracy and to handle degeneracy, they adopt the lexicographic pivot rule. The efficiency of faces is checked by means of the following theorem:

THEOREM I.16. The face $f(T,F)$ is efficient if and only if $G(F) \neq \emptyset$ where

$$G(F) = \{ (v,s) \mid -C^T v + s = C^T e; \quad v, s \geq 0, \quad s_j = 0 \quad \text{if } j \in F \}.$$

Furthermore, the face $f(T,F)$ is maximal efficient if and only if F is a maximal index set.

This condition is related to the dual of the linear programming subproblem given by Isermann in Theorem I.11. Maximal efficient faces are constructed by finding maximal sets F for which $G(F) \neq \emptyset$. And with each such face a vector $\lambda = v + e$ is associated. These associated vectors λ allow one to check whether or not any of the previously encountered maximal efficient faces is incident to the current efficient extreme point. Such faces are recognized by using each previously stored vector λ in performing the matrix multiplications $\lambda^T C$, checking if $\lambda^T C \geq 0$, and if so identifying those sets F for which $\lambda^T C^k = 0, \quad \forall k \in F$.

When a pivot is made from an efficient extreme point to another adjacent

one along an efficient edge, those maximal efficient faces incident to the new efficient extreme point which contain that efficient edge are easily identified. If x_j is the entering variable and x_k is the leaving variable associated with that pivot, in any maximal index set containing j , j is simply replaced by k .

At any efficient extreme point, the index set J_T of nonbasic variables that lead to adjacent efficient extreme points along an efficient edge, and such efficient extreme points are identified first. Then, a subset of J_T , K^0 , is formed by eliminating those indices corresponding to edges that are incident to previously encountered efficient extreme points. Because if such edges belong to a maximal efficient face, that face would have already been constructed. Then, only maximal index sets which form a subset of K^0 are searched. This would lead to computational advantages in determining all maximal index sets.

Thus, by incorporating the determination of maximal efficient faces into the main body of the algorithm and through several considerations, computational savings are achieved in comparison to other procedures for determining the set of all efficient points. But still, the amount of work needed for determination of the entire efficient set over that needed for determining simply the set of efficient extreme points is enormous. Even if fewer combinations of indices are checked in determining maximal index sets, and duplication of effort is avoided, still a lot of computations are needed. The determination of maximal index sets is itself quite cumbersome. Furthermore, considerable calculations are required to reduce the combinations to be checked.

As an overview, the various efficiency checks proposed by several authors can be basically divided into two groups: i) Those which are based on multiparametric LP (Zeleny [61], Gal [22], Ecker and Kouada [16]) ii) Those which use an approach dual to that employed in multiparametric programming (Yu and Zeleny [59], Evans and Steuer [18], [19], Isermann [34].)

Considering the generation of efficient extreme points, the procedures in the second group require solving a linear programming subproblem with a different feasible region for each adjacent vertex. In the approaches of the first group, linear programming subproblems defined on the same feasible region are to be solved for each adjacent vertex. Moreover, it may

not be necessary to solve all of the subproblems if the checks given by Ecker and Kouada are carried out.

Another difference between the various approaches is in the way they handle degeneracy, which removes the one-to-one correspondance between bases and extreme points. In some additional conditions are incorporated (Evans and Steuer, [18], Ecker and Kouada [16]); in some others, for each efficient extreme point only certain bases which fulfill a specific criterion are enumerated (Evans and Steuer [19], Isermann [34], Gal [22]).

A different generating approach, which aims at generation of a relevant subset of efficient extreme points is given by Steuer [54], [55]. In his "interval criterion weights" method [54] Steuer examines the following problem .

$$\begin{aligned} \max_{x \in X} \quad & w^T Gx \\ w \in W = \quad & \{w \mid \ell_i \leq w_i \leq u_i, \quad \sum_{i=1}^p w_i = 1\} . \end{aligned}$$

With the specification of intervals on parametric objective weights w_i , the original gradient cone, whose extreme rays are defined by rows of G , is reduced to a subset of itself. Consequently, only a subset of efficient extreme points will be generated. Steuer states that the problem is insolvable in its present form and uses the q extreme rays of the reduced gradient cone to define new objectives and considers solution of the resulting multiobjective problem. However, he does not propose a method for computing q and the new objectives other than an exhaustive enumeration of all endpoint possibilities allowed for the weights. All possible endpoint combinations are enumerated and each combination is used to weight the objectives to obtain one new objective. Steuer also observes that $q > p$, where p is the number of original objectives, and rapidly becomes larger than p as p increases.

Steuer also offers a filtering technique to further reduce the list of candidate solutions before presentation to the decision maker [55]. The technique operates on the principle of discarding those generated solutions not sufficiently "dissimilar" from those already retained. In this way, a relatively evenly dispersed collection of a reduced set of efficient

extreme points can be presented to the decision maker, facilitating his final selection process.

In the next section, examples of interactive approaches to the multiobjective linear programming problem, which generate a single "preferred" solution, are discussed.

I.1.2. INTERACTIVE APPROACHES

These approaches rely on the progressive definition of the decision maker's preferences along with the exploration of the objective space. They assume that the decision maker is able to give preference information on a local level with respect to a particular solution. Some methods require explicit information regarding the trade-offs between the attainment levels of objectives at each step; others require implicit trade-off information by asking the decision maker to indicate the acceptability of the current achievement level. The Zionts-Wallenius method [62] is an example of the first category, while the STEM method [6] illustrates the second. Here, these two methods will be reviewed briefly.

Zionts-Wallenius Method:

Step 1. Choose a set of positive weights λ_i and solve the corresponding scalarized linear programming problem.

Step 2. From the set of nonbasic variables, N , select the set of efficient variables. That is for each nonbasic variable solve the following problem:

$$\begin{array}{ll} \min & \lambda^T C^k \\ \text{s. t.} & \lambda^T C^j \geq 0 \quad \forall j \in N, \quad j \neq k \\ & e^T \lambda = 1 \\ & \lambda \geq 0 \end{array}$$

If the minimum value of $\lambda^T C^k$ is negative, the variable x^k is said to be efficient, otherwise it is not efficient.

Step 3. For each efficient variable x_k , a set of trade-offs are defined by the reduced costs C^k , implying increases in some objectives and reductions in others. Present these trade-offs to the decision maker and request him to state whether the trade-offs are desirable, undesirable or neither. If all trade-offs are undesirable stop. The current solution is the preferred solution. Otherwise, for each desirable response construct an inequality of the form

$$\lambda^T C^k \leq -\delta$$

where δ is a sufficiently small positive number. For each undesirable response, construct an inequality of the form

$$\lambda^T C^k \geq \delta$$

and for each response of indifference construct an equality of the form

$$\lambda^T C^k = 0$$

Step 4. Find a feasible solution to all previously constructed constraints plus the following constraints

$$\begin{aligned} e^T \lambda &= 1 \\ \lambda_i &\geq \delta \quad i=1, \dots, \bar{p}. \end{aligned}$$

The resulting set of λ_i are the new weights. Go to step 1.

The method is convergent since at each interactive cycle the possible choice of weights λ_i is restricted and each trade-off which is attractive to the decision maker increases his implicit utility function value. However, the method requires the decision maker to be consistent. Otherwise, a feasible set of weights may not be found in step 4. The decision maker's implicit utility function may not be precise initially and earlier responses might unnecessarily constrain the outcome of subsequent iterations.

The STEM Method:

Step 1. Solve the problem with respect to each objective function separately. Construct a payoff matrix Z where Z_{ij} gives the value of the i th objective function when the j th objective function is at its maximum.

Step 2. Let \bar{f} denote the ideal solution where $\bar{f}_j = \max_{x \in X} C_j x$. At the m th interactive cycle, find a solution which is nearest in the minimax sense to the ideal solution \bar{f} , i.e. solve

$$\begin{aligned} \min \quad & y \\ \text{s.t.} \quad & y \geq (\bar{f}_j - C_j x) w_j, \quad j=1, \dots, p. \\ & x \in X^m \\ & y \geq 0 \end{aligned}$$

where X^m includes $Ax \leq b$, $x \geq 0$ plus any constraints added in the previous cycles. Here w_j give the relative importance of the distances from the optima and are chosen such that

$$w_i = \alpha_j / \sum_i \alpha_i$$

where

$$\alpha_j = \begin{cases} \frac{\bar{f}_j - f_j^{\min}}{\bar{f}_j} \left(\frac{1}{\sqrt{\sum_i (C_{ji})^2}} \right) & \text{if } \bar{f}_j > 0 \\ \frac{f_j^{\min} - \bar{f}_j}{f_j^{\min}} \left(\frac{1}{\sqrt{\sum_i (C_{ij})^2}} \right) & \text{if } \bar{f}_j \leq 0 \end{cases}$$

where f_j^{\min} is the minimum value in the j th column of the payoff table.

Step 3. The solution x^m is presented to the decision maker. If all objectives are satisfactory, the compromise solution has been found, stop. If none of the objectives are deemed satisfactory, stop. The model does not provide a solution for the decision maker. If some objectives are satisfactory and others are not, the decision maker must relax a satisfactory objective f_j to allow an improvement of the unsatisfactory objectives in the next iterative cycle. He is to define Δf_j as the amount of acceptable relaxation. Then for the next cycle, the feasible region X^m is modified as:

$$x^{m+1} = \begin{cases} x^m \\ C_j x \geq C_j x^m - \Delta f_j \\ C_i x \geq C_i x^m \end{cases} \quad i=1, \dots, p, \quad i \neq j$$

The weight w_j is set to zero, and a return to step 2 follows.

In this method, the decision maker may feel more comfortable in that he is only required to indicate satisfactory levels of objectives rather than indicating preferred trade-offs. However, convergence is not guaranteed and there is the possibility that the procedure terminates without providing a satisfactory solution to the decision maker.

In addition to the generating and interactive approaches, some other approaches to the multiobjective linear programming problem which are based on prior specification of preferences have been used. One basic approach is asking the decision maker to specify weights on each objective and solve the problem with the resulting weighted objective function. Another approach is goal programming, where the decision maker is asked to specify target levels for each objective and the sum of weighted absolute deviations from these target levels is minimized. For successful application of goal programming the decision maker should have a clear notion of targets and priorities and if targets are not specified properly, inferior solutions could result.

Upto here, an overview of available approaches to general multiobjective linear programming problem have been provided. In the next section the special case of bicriterion or two-objective linear programming problem is considered.

1.1.3. APPROACHES TO BICRITERION LINEAR PROGRAMMING

Bicriterion linear programming problems have a simpler structure in comparison to multiobjective problems with more than two objectives. Specialized algorithms exploiting this simplicity of structure have been developed for these problems.

Geoffrion [27] suggested use of parametric linear programming. Since the degree of parameterization is one, this approach works quite well for bicriterion problems. Here the two objective functions z_1 and z_2 are combined parametrically to give a single objective function $\lambda z_1 + (1-\lambda)z_2$. Initially, the linear program is solved with $\lambda=0$. If there is a unique solution, it is efficient. If there are alternative optima, these are checked to find the initial efficient solution. Then, through sensitivity analysis, the maximum value of λ for which the present solution still remains optimal is found. The nonbasic variable with current objective row entry equal to zero at this value of λ is then selected as the entering variable and thus a new efficient extreme point is found. This procedure is repeated until $\lambda=1$ is reached.

Zeleny [61, pp.149-158] observed that efficient extreme points could be selected out of the set of feasible extreme points by identifying a cutting hyperplane in the objective space. Let $u=C_1x$ and $v=C_2x$ for any $x \in X$, and let x^1 and x^2 be the efficient maximum solutions of C_1x and C_2x respectively. Also define $u_1=C_1x^1$, $v_1=C_2x^1$ and $u_2=C_1x^2$, $v_2=C_2x^2$. Then the cutting hyperplane L , is defined as:

$$L = \left\{ (u,v) \mid v - \left(\frac{v_2 - v_1}{u_2 - u_1} \right) u = v_1 - \left(\frac{v_2 - v_1}{u_2 - u_1} \right) u_1 \right\}.$$

The cutting hyperplane is depicted graphically in Figure I.1., where the shaded region represents the objective space.

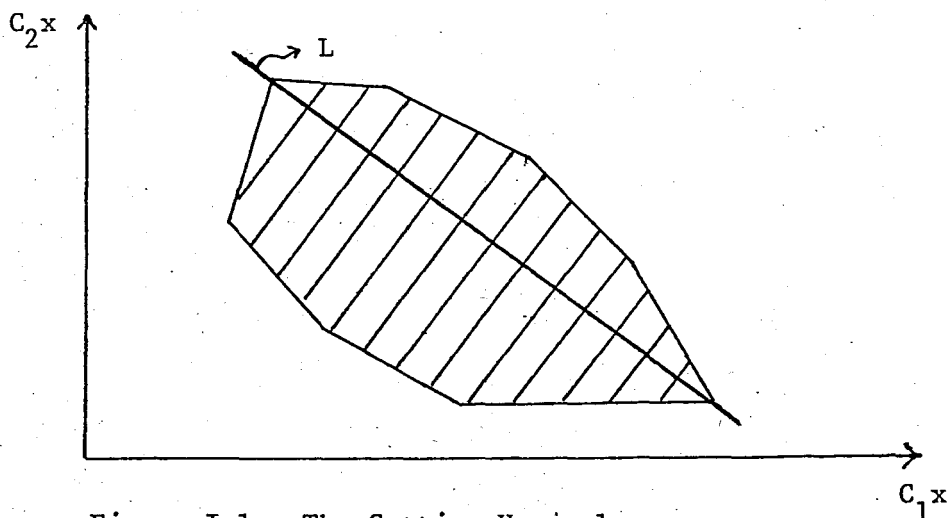


Figure I.1. The Cutting Hyperplane

Zeleny didnot pursue an algorithm based on the cutting hyperplane idea because of his reservations on its extension to problems with more than two objectives. However, this idea forms the conceptual basis for two algorithms operating in the objective space given by Aneja and Nair [2] and Cohon, Church and Sheer [14]. The two algorithms are essentially the same and employ a weighted objective function, the weights of which are changed at each iteration. The first two efficient points are found by maximizing the two objectives individually. New solutions are found iteratively by moving in a direction normal to the line segment connecting two previously found efficient points in the objective space. Two efficient points which give consecutive values, recorded up to the current iteration, of one of the objectives are chosen. Then, the weights of the objective function are changed so as to correspond to the slope of the line segment connecting these two points. Given two such points r and s , the new objective weights λ_1 and λ_2 are calculated as $\lambda_1 = |z_2(s) - z_2(r)|$ and $\lambda_2 = |z_1(s) - z_1(r)|$ where $z_i(s)$ denotes the value of i th objective at point s . Then, using these weights λ_1 and λ_2 a new objective function is formed and a new linear programming problem is solved in the next iteration. Aneja and Nair state that their algorithm needs exactly $2k-3$ such iterations, after the first two efficient points are found, if there are k ($k > 2$) efficient extreme points.

I.2. AN ALGORITHM FOR GENERATING ALL OR A RELEVANT SUBSET OF EFFICIENT EXTREME POINTS

Multiobjective linear programming has considerable computational requirements. Even generating only the efficient extreme points requires a lot of computational effort. As discussed above, going a step further and generating all efficient faces involves much more additional computational burden. Furthermore, the difficulty of meaningful presentation of the results to the decision maker is augmented. For practical, real-life problems, the second approach has quite limited applicability. Consequently, in this study, the main concern was generation of efficient extreme points or a relevant subset of them corresponding to a decision maker specified preference region.

The algorithm presented here is based on a synthesis of the parametric approach, some concepts of Isermann and the interval criterion weights concept of Steuer. Dual feasible bases are enumerated by the algorithm; however, the subproblem for checking dual feasibility of adjacent basic solutions utilizes the parametric approach. The simple tests described by Ecker and Kouada [16] are used in the subproblem to increase computational efficiency. To avoid duplication of effort, and hence to increase computational efficiency further, the efficiency checks are carried out monotonically with respect to one of the objectives; based on the observation of a "monotone connectedness" property. That is, after finding the initial efficient point by maximizing that objective, say objective k , at each dual feasible basis only those adjacent bases with nonincreasing values of objective k are checked for dual feasibility.

1.2.1. OBSERVATIONS ON THE EFFICIENCY CHECK

Reviewing briefly, the MOLP problem is formulated as

$$\begin{aligned} \max_{x \in X} \quad & Gx \\ X = \{x \mid & Ax \leq b, \quad x \geq 0 \} \end{aligned}$$

where the rows of the $p \times n$ matrix G represent the different objective functions. For any basic feasible solution x^0 , there is an associated basis B , a reduced cost matrix C and constraint coefficient matrix Y , as defined previously.

Adopting Isermann's terminology, a basis will be called dual feasible if and only if there exists $\lambda > 0$ such that $\lambda^T C \geq 0$. Furthermore, two adjacent dual feasible bases will be termed λ -adjacent if and only if they are alternative solutions of P_λ for some $\lambda > 0$. Then, the result given below, although not stated explicitly in the way chosen here, is implied by Isermann [34].

THEOREM I.17. Let x^i be a dual feasible basic solution and $C_{(i)}$ be the corresponding reduced cost matrix. Let x^j be the basic solution obtained by introducing nonbasic variable x_k into the basis and $C_{(j)}$ be its reduced cost matrix. Then, x^j is a λ -adjacent dual feasible basic solution if and

only if (there exists) $\exists \lambda > 0$ such that

$$\lambda^T C_{(i)} \geq 0 \quad \text{and} \quad \lambda^T C_{(i)}^k = 0.$$

where again $C_{(i)}^k$ is the column of $C_{(i)}$ corresponding to x_k .

Proof: Assume $\bar{\lambda} > 0$ satisfies the above condition. Then the solution represented by x^j is an alternative solution to P_λ for $\lambda = \bar{\lambda}$. Also, we have $\bar{\lambda}^T C_{(j)} = \bar{\lambda}^T C_{(i)} \geq 0$. Thus, x^j is a λ -adjacent dual feasible basic solution.

Let x^j be a λ -adjacent dual feasible basic solution. Then, $\exists \bar{\lambda} > 0$ such that x^i and x^j are alternative solutions of P_λ for $\lambda = \bar{\lambda}$. Let $y_{rk} > 0$ be the pivot element when x_k is the entering variable and x_ℓ is the leaving variable.

For the basic solution x^i , the inequality $\bar{\lambda}^T C_{(i)}^k \geq 0$ is satisfied. After pivoting, the column $C_{(j)}^k$ will be given by $-C_{(i)}^k / y_{rk}$ and for this new solution x^j , the inequality $-\bar{\lambda}^T C_{(i)}^k / y_{rk} \geq 0$ is satisfied. The two inequalities together imply $\bar{\lambda}^T C_{(i)}^k = 0$.

Next, we give a new result as a consequence of which the check for dual feasibility can be limited to certain adjacent bases.

THEOREM I.18. Each dual feasible basis, except the one where objective k attains its maximum, has at least one λ -adjacent dual feasible basis obtained by introducing into basis a nonbasic variable x_s with $C_{ks} < 0$.

Proof. If the given basis is not optimal with respect to objective k , there is at least one j such that $C_{kj} < 0$. And if the basis is dual feasible $\exists \lambda > 0$ such that

$$\lambda^T C^j = \delta_j \geq 0 \quad \forall j \in N$$

where N is the index set of nonbasic variables.

Let $\frac{\delta_s}{C_{ks}} = \max_{j, C_{kj} < 0} \frac{\delta_j}{C_{kj}}$

and let λ' be given by

$$\lambda'_k = \lambda_k - \frac{\delta_s}{C_{ks}} \quad \text{and} \quad \lambda'_j = \lambda_j > 0 \quad \forall j \neq k$$

obviously $\lambda' > 0$.

Then

i) If $C_{kj} \geq 0$ then $\lambda'^T C^j = \lambda^T C^j - \frac{\delta_s}{C_{ks}} C_{kj} \geq C^j \geq 0$

ii) If $C_{kj} < 0$ then $\lambda'^T C^j = \lambda^T C^j - \frac{\delta_s}{C_{ks}} C_{kj} = \delta_j - \frac{\delta_s}{C_{ks}} C_{kj} \geq 0$

because as $\frac{\delta_s}{C_{ks}} \geq \frac{\delta_j}{C_{kj}}$, we have $\delta_j - \frac{\delta_s}{C_{ks}} C_{kj} \geq 0$

Also $\lambda'^T C^s = \delta_s - \frac{\delta_s}{C_{ks}} C_{ks} = 0$,

and by introducing x_s into the basis a λ -adjacent dual feasible basic solution is obtained.

Thus, for each dual feasible basis, there exists at least one λ -adjacent dual feasible basis where the value of objective k is higher (if x_s enters the basis at a positive level) or remains the same (if x_s enters the basis at zero level). The theorem implies that one can go from any dual feasible basis to the dual feasible basis which is optimal with respect to objective k , by moving through a sequence of λ -adjacent dual feasible bases which have nondecreasing values of objective k . That is, there is a "monotone connectedness" property associated with the set of dual feasible solutions. Here, we define monotone connectedness as the existence of a path, consisting of efficient edges, along which the value of objective k is nonincreasing, and which connects the dual feasible solution which maximizes objective k to any dual feasible solution. Then, starting from a dual feasible basic solution

maximizing objective k , at each dual feasible basis, one need consider only those λ -adjacent dual feasible bases which have nonincreasing values of objective k . In other words, one needs to check only the nonredundancy of the constraints j with $C_{kj} < 0$.

Actually, a sense of direction is being incorporated. Each efficient edge is identified only once, along the direction of decrease of a chosen objective. In comparison to the previous approaches where the edges (or the efficient extreme points along the edges) are identified in both increasing and decreasing directions, the proposed method which avoids duplication of effort will lead to considerable computational efficiency.

I.2.2. DEVELOPMENT OF THE ALGORITHM

All dual feasible bases and hence all efficient extreme points are enumerated by the algorithm. The mechanics of the algorithm is given by the flow diagram in Figure I.2.

The first step of the algorithm is finding the initial dual feasible basis which is an optimal basis for a chosen objective, say objective k . To find this basis, first objective k is maximized. If there is a unique optimal solution, that solution is efficient and dual feasible. Dual feasibility follows from the fact that $C_{kj} > 0$, $\forall j$, and by taking λ_k arbitrarily large $\lambda^T C \geq 0$ can be achieved. If there are alternative optimal solutions, some of these may be dominated. First, an efficient basic solution is found by checking all alternative optima. If the basic solution is nondegenerate, it is dual feasible as implied by Theorem I.6. In case of degeneracy, we cannot be sure of the dual feasibility of the basis at hand. Then, all bases representing the efficient extreme point at hand are checked until a dual feasible basis is determined.

Next, all nonbasic variables that lead to λ -adjacent dual feasible bases with nondecreasing values of objective k are to be identified. That is, nonredundant constraints j of the system $C^T \lambda \geq 0$, $\lambda > 0$, which have positive entries in the k th column of this system ($C_{kj} > 0$) will be identified. Because working with the condition $\lambda > 0$ is tedious, we make the transformation

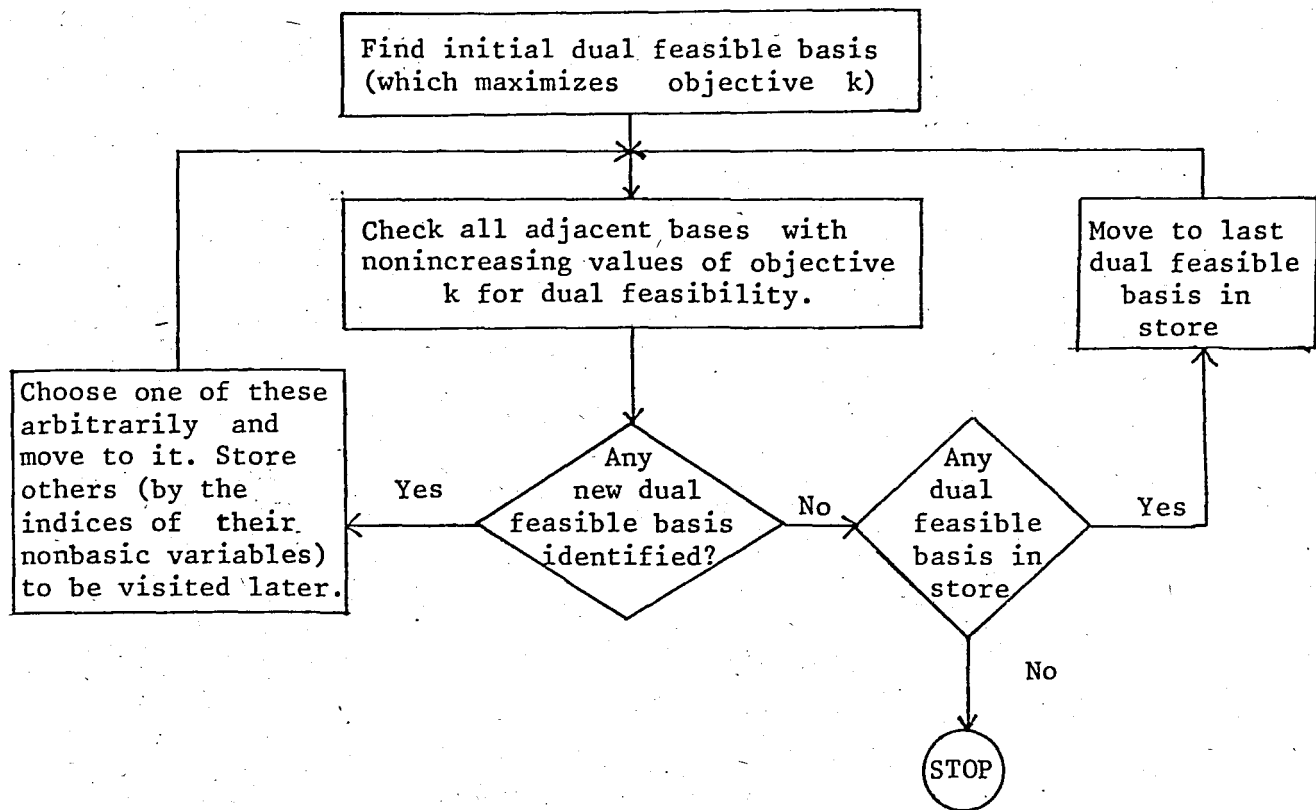


Figure 1.2. The Mechanics of the Algorithm

$\lambda = v + e$, $v \geq 0$, which is valid due to the homogeneity of the system of inequalities. The subproblem used for identifying nonredundant constraints is similar to that used by Ecker and Kouada [16]. Specifically, the following linear programming problem for each $q \in L$ where $L = \{j \in P \mid C_{kj} > 0\}$ and $P = \{j \mid C_j \neq 0\}$ is solved:

$$\begin{aligned}
 \min \quad & s_q \\
 \text{s.t.} \quad & -C_j^T v + s = C_j^T e, \quad \forall j \in P \\
 & v \geq 0 \quad s \geq 0
 \end{aligned}$$

In the subproblem those constraints j for which $C^j \geq 0$ are not included since they are obviously redundant, because $\lambda^T C^j > 0$ for any $\lambda > 0$.

The tests described by Ecker and Kouada [16] are used to eliminate the need for solving each subproblem. The procedure followed in determining JT, the set of nonbasic variables that lead to λ -adjacent dual feasible bases with nondecreasing values of objective k , can be described by these steps:

1. Form the sets P and L. Set $JT = \emptyset$.
2. Find a feasible solution to the subproblem.
3. For each $q \in L$ such that s_q is nonbasic, or basic with value zero set $JT = JT \cup \{q\}$, $L = L - \{q\}$. If $L = \emptyset$ stop.
4. For each $q \in L$ such that s_q is basic, but can be made nonbasic by a single pivot, set $JT = JT \cup \{q\}$ and $L = L - \{q\}$.
5. For each $q \in L$ such that s_q is basic with a positive value, and all nonbasic entries in the row in which it is basic are nonpositive, set $L = L - \{q\}$, $P = P - \{q\}$. (Because the value of s_q cannot be decreased below its current positive value). Delete the corresponding constraint.
6. If $L = \emptyset$ stop. Otherwise select $q \in L$, set $L = L - \{q\}$ and adjoin the objective s_q to be minimized.
7. If s_q has a minimum value of zero, set $JT = JT \cup \{q\}$. Otherwise delete the row in which s_q is basic and go to Step 6.

Another feature of the algorithm is the option of specifying intervals on parametric objective weights, w_i , as considered first by Steuer [54] in his interval criterion weights method. As discussed in Section I.1.2. Steuer considered the problem

$$\max_{x \in X} w^T Gx \quad w \in W = \{w \mid \ell_i \leq w_i \leq u_i, \sum_{i=1}^p w_i = 1\}$$

He stated that the problem is insolvable in this form, and derived an equivalent MOLP problem by defining new objectives. The problem is in fact solvable in its original form, by simply modifying the feasible region of the subproblem through the addition of the constraints implied by the intervals on objective weights w_i . Normalizing, we have

$$w_i = \lambda_i / \sum_j \lambda_j \quad i=1, \dots, p.$$

Also, we have $\lambda_i = v_i + 1$ and $\sum_j \lambda_j = \sum_j v_j + 1$. Therefore the constraints $\ell_i \leq w_i \leq u_i$ imply that

$$\ell_i \leq \frac{v_i + 1}{\sum_j v_j + 1} \leq u_i.$$

After some manipulation we have

$$(1 - u_i)v_i - \sum_{j \neq i}^p u_i v_j \leq p u_i - 1$$

and

$$(1 - \ell_i)v_i - \sum_{j \neq i}^p \ell_i v_j \geq p \ell_i - 1$$

Consequently, the subproblem is to be augmented by these constraints. Furthermore, any other information given by the decision maker concerning the objective weights and which can be expressed as a linear inequality can also be incorporated. For example if the decision maker states that objective i is at least twice as important as objective r , then the constraint $w_i - 2w_r \geq 0$ implies

$$\frac{v_i + 1}{\sum_j v_j + 1} - 2 \frac{v_r + 1}{\sum_j v_j + 1} \geq 0$$

or

$$v_i - 2v_r \geq 1.$$

To complete the algorithm a scheme of the order in which dual feasible bases will be generated; or the bookkeeping of the algorithm must be specified. The scheme used is similar to that proposed by Ecker and Kouada [16]. Throughout the algorithm, two different sets are formed and stored to keep track of dual feasible bases. These are:

- V_1 : The set of dual feasible bases identified, but not yet generated
- V_2 : The set of dual feasible bases already generated.

Bases are stored by the indices of their nonbasic variables. For each basis, the nonbasic variable index set is sorted in ascending order to facilitate the various basis comparisons taking place throughout the algorithm. Given the initial dual feasible basis identified by its nonbasic variable set N , the algorithm proceeds as follows:

1. Set $V_1 = \phi$, $V_2 = \{N\}$
2. Determine the set, R , of λ -adjacent dual feasible bases with nonincreasing values of objective k .
3. Set $R_1 = R - V_2$. If $R_1 = \phi$ go to Step 7.
4. Set $R_2 = R_1 - V_1$. If $R_2 = \phi$ go to Step 6.
5. Take last element of R_2 , denoted by N and pivot to the associated basis. Set $V_1 = V_1 \cup R_2 - \{N\}$ and $V_2 = V_2 \cup \{N\}$. Go to Step 2.
6. Take last element of R_1 , denoted by N and pivot to the associated basis. Set $V_1 = V_1 - \{N\}$ and $V_2 = V_2 \cup \{N\}$. Go to Step 2.
7. If $V_1 = \phi$ stop. All dual feasible bases have been enumerated. Otherwise take last element of V_1 , denoted by N and pivot to the associated basis. Set $V_1 = V_1 - \{N\}$ and $V_2 = V_2 \cup \{N\}$. Go to Step 2.

Here, some general comments regarding computational efficiency can be given. In this algorithm, the subproblems to be solved for testing dual feasibility are defined on the same feasible region in contrast to the approaches of Yu

and Zeleny, Evans and Steuer and Isermann, where the feasible region changes from one subproblem to the other. Thus, the changeover from one subproblem to the other requires much less computational effort here, and by using the simple tests of Ecker and Kouada, it may not be even necessary to solve explicitly each subproblem. In addition, using the monotone connectedness property introduces computational advantages in identifying those nonbasic variables that lead to λ -adjacent dual feasible bases (one needs to consider only those x_j with $C_{kj} > 0$); in identifying the corresponding bases, i.e. in forming the set R , and in identifying and eliminating from R those bases which are in V_1 and V_2 .

I.2.3. THE COMPUTER PROGRAM AND COMPUTATIONAL RESULTS

The algorithm has been coded in FORTRAN IV. The computer program enumerates all efficient extreme points or only those efficient extreme points satisfying the interval limits on the objective weights, if these are specified, and outputs a set of representative weights for each efficient extreme point. It is composed of a main program and 10 subroutines. The relationships between the main program and the various subroutines are as diagrammed in Figure I.3.

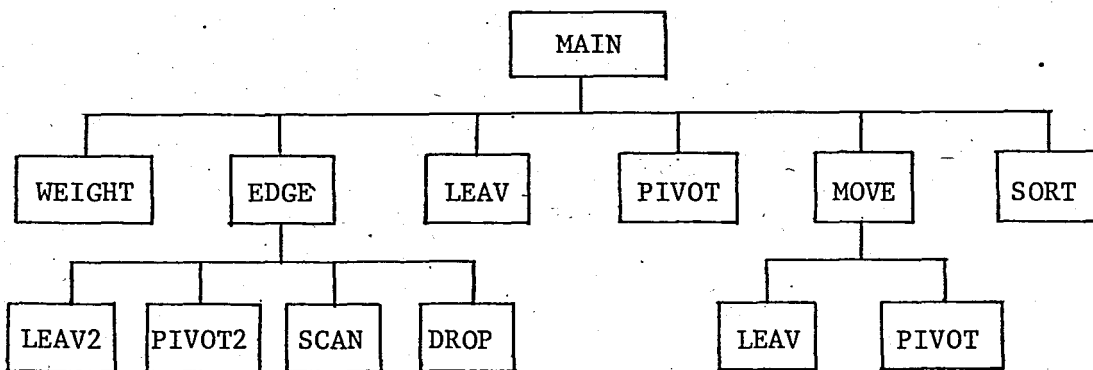


Figure I.3. Program Structure

The initial dual feasible basis is found in the MAIN program by maximizing the last objective, objective p, if no intervals on objective weights are specified. Otherwise a composite objective which is formed by using weights satisfying the interval limits is maximized. Here the tableau form of the simplex algorithm, where only the nonbasic entries are stored, is used. For each column of the current nonbasic cost and coefficient matrices, C and Y the associated nonbasic variable index, and for each row of Y the associated basic variable index are stored to keep track of operations on the simplex tableau. All input and bookkeeping of the algorithm is also done in the MAIN program.

If intervals on objective weights are specified, subroutine WEIGHT is called once at the start of the MAIN program to form the inequalities to be added to the subproblem for testing dual feasibility.

Once a new dual feasible basis is selected in the MAIN program, a move to that basis is accomplished by the subroutine MOVE. If the selected basis is an adjacent one a single pivot is carried out. Otherwise, the number of pivots needed is equal to the number of nonbasic variables differing between the current basis and the basis to which a move is desired. In this process, first the current nonbasic variable set, IXN, is compared against the nonbasic variable set of the selected basis, INT, to identify a variable which is included in IXN but not in INT. Such a variable becomes the entering variable. Next, a variable which is included in INT but not in IXN, is identified. If the implied pivot element is nonzero, this variable becomes the leaving variable. Otherwise, another variable which is included in INT but not in IXN, and for which the implied pivot element is nonzero is searched. When such a variable is found, the implied pivot is carried out. This procedure is repeated until IXN and INT are identical, meaning a move to the selected basis has been accomplished. It is to be noted that the pivots carried out are not necessarily feasible, but in the end, the selected basic which is feasible, is obtained.

The pivoting operations required in the MAIN program and subroutine MOVE are carried out by subroutine PIVOT. Subroutine LEAV is used to identify the variable to leave the basis. For this purpose, the minimum ratio rule

is used. If column k is the pivot column, i.e. the associated nonbasic variable of column k is the entering variable, the row, r , for which

$$b_r / y_{rk} = \min_i b_i / y_{ik} ; \quad y_{ik} > 0$$

is determined. Then y_{rk} gives the pivot element, and the variable which is stored as the basic variable of row r is the leaving variable. Given the pivot column and pivot row the pivoting operation is performed by subroutine PIVOT as follows. The pivot row, r , including the right hand side b_r , is divided by y_{rk} and the entering variable is eliminated from all other rows including the rows of C . The pivot column is updated to contain the entries associated with the leaving variable. That is, after pivoting column k is given by

$$\bar{C}^k = -C^k / y_{rk} ; \quad \bar{Y}^k = -Y^k / y_{rk} ; \quad \bar{y}_{rk} = 1 / y_{rk}$$

where \bar{C} and \bar{Y} are the new nonbasic cost and coefficient matrices. Now, the leaving variable becomes the nonbasic variable associated with column k and the entering variable becomes the basic variable associated with row r .

Subroutines LEAV2 and PIVOT2 are used by subroutine EDGE to perform similar operations in the subproblem for checking dual feasibility.

Subroutine EDGE is used to form the set JT , the set of nonbasic variables that lead to λ -adjacent dual feasible bases with nondecreasing values of objective p , along the steps given in section I.2.2. In carrying out steps 3 and 4 given there, subroutine SCAN is used to update the relevant sets. Step 5, i.e., identifying redundant constraints and deleting them, is carried out by subroutine DROP. Also, when an initial basic feasible solution to the subproblem is found at the start of subroutine EDGE, the corresponding objective weights are outputted to provide a representative weighting vector for the current dual feasible basis of the main problem.

The initial efficient solution is outputted by the MAIN program. The remaining efficient solutions are outputted in subroutine MOVE, after the move to them has been accomplished.

Double precision arithmetic is used in the computer program to avoid inaccuracies due to roundoffs. Only core memory is used. Currently the maximum problem size is limited to 50 variables, 50 constraints and 5 objectives.

The flow diagram of the MAIN program and subroutine EDGE are given in Figure I.4.

Computational results for a sample of randomly generated problems are given in Table I.1. The elements of matrices G and A were generated randomly in the interval $[0,20]$, with a 20 % density of zero elements for the matrix A. The elements of the right hand side vector, b, were generated randomly in the interval $[0, 10n]$ for each problem where n is the number of variables.

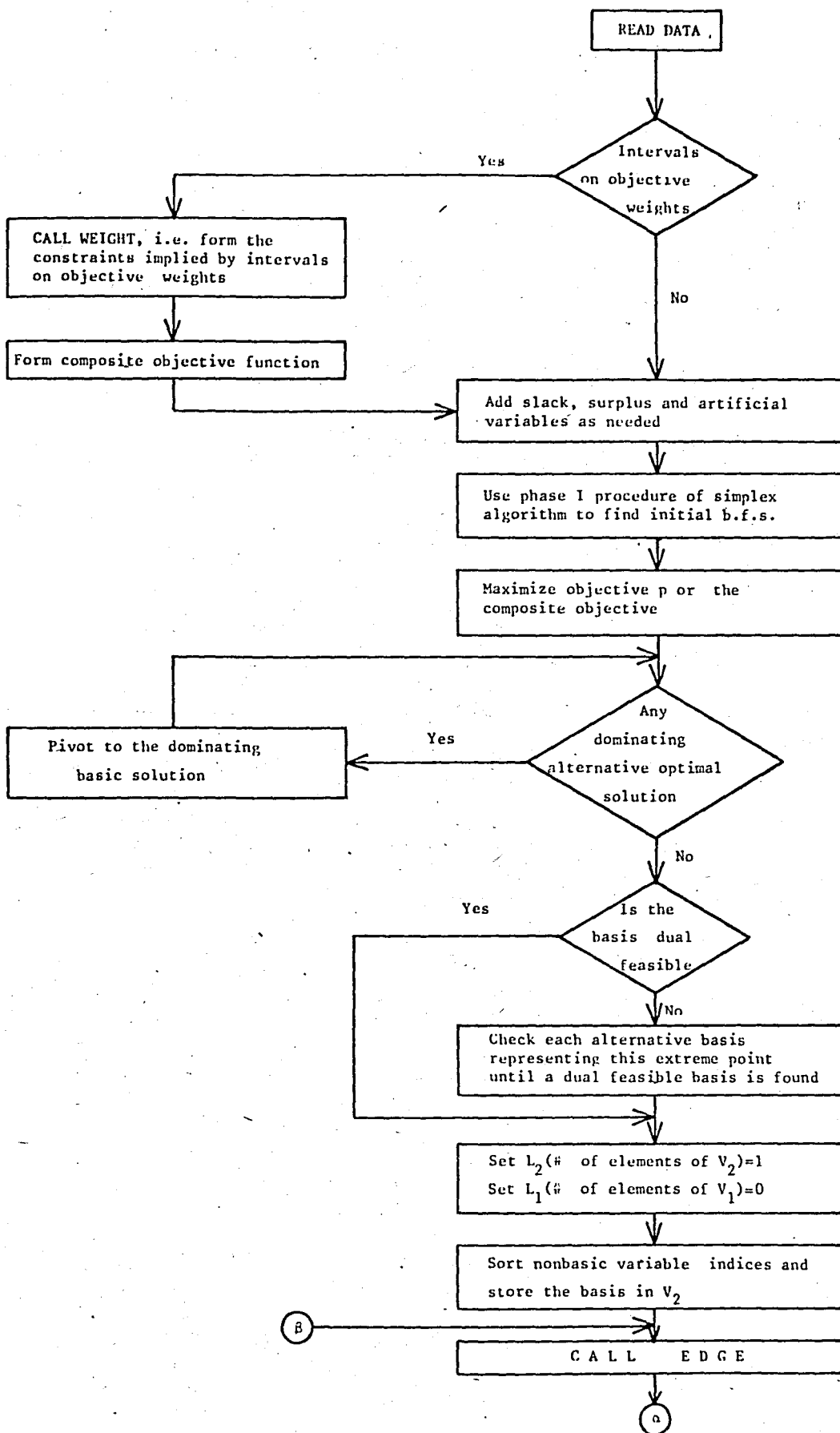
Inspection of results indicate that

- i) Computation time is strongly influenced by the number of efficient extreme points. (Compare problems 6 and 10)
- ii) Computation time also increases with number of variables and constraints. (Compare problems 6 and 8)
- iii) Computation time increases with number of objectives, but not so rapidly as in i) and ii). (Compare problems 1 and 11; and problems 2, 12 and 20.)

Listing of the computer program, definitions of the variables used, data input instructions and a sample output are given in Appendix I.

I.3. AN ALGORITHM FOR BICRITERION LINEAR PROGRAMMING

A lot of real life problems can be modelled quite accurately with two objective functions. A specialized and computationally more efficient algorithm for such problems will be a useful tool of multiobjective decision making. Thus, another direction of research was investigating the possibility of exploiting the special structure of bicriterion linear programming problems.



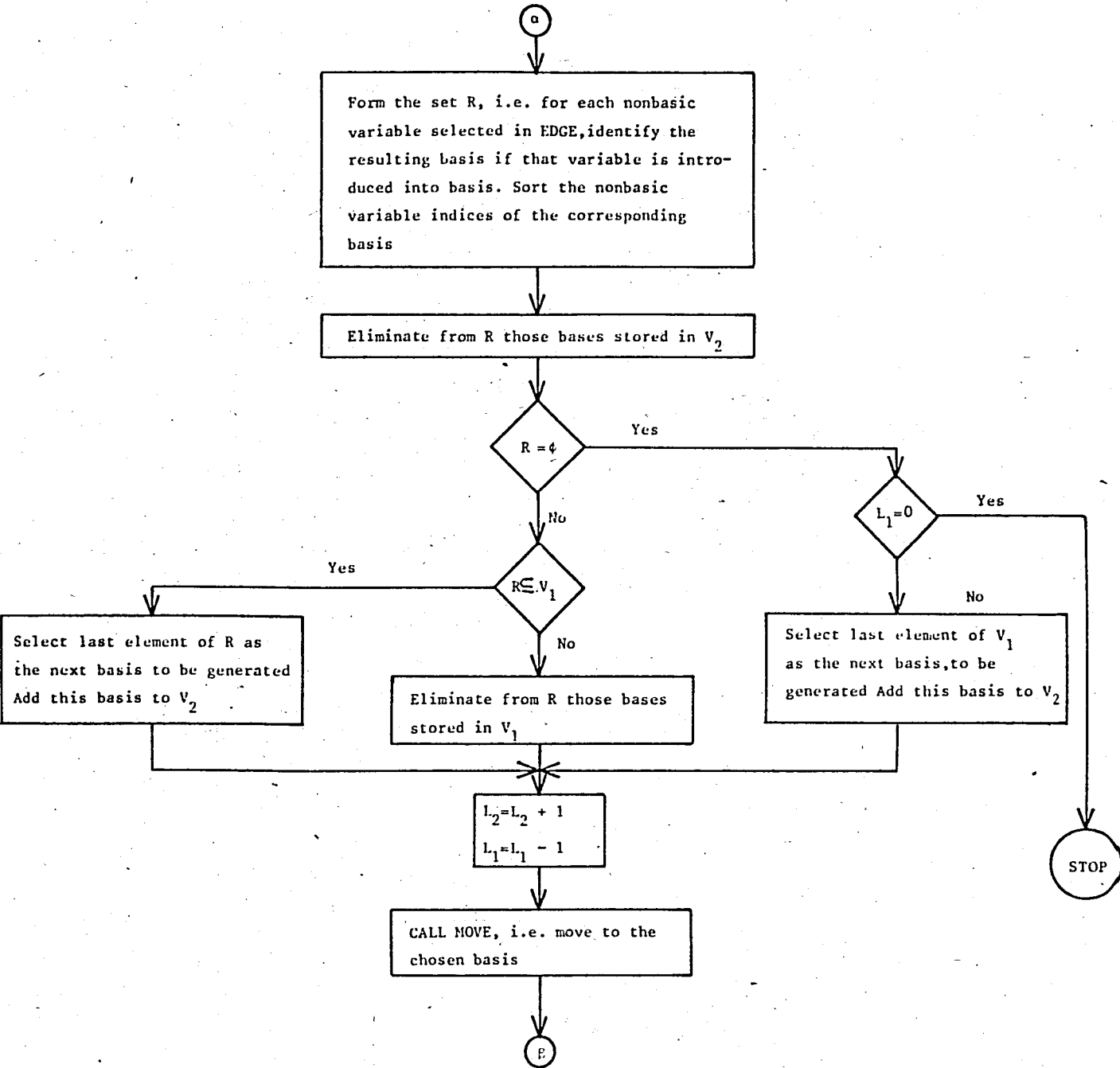
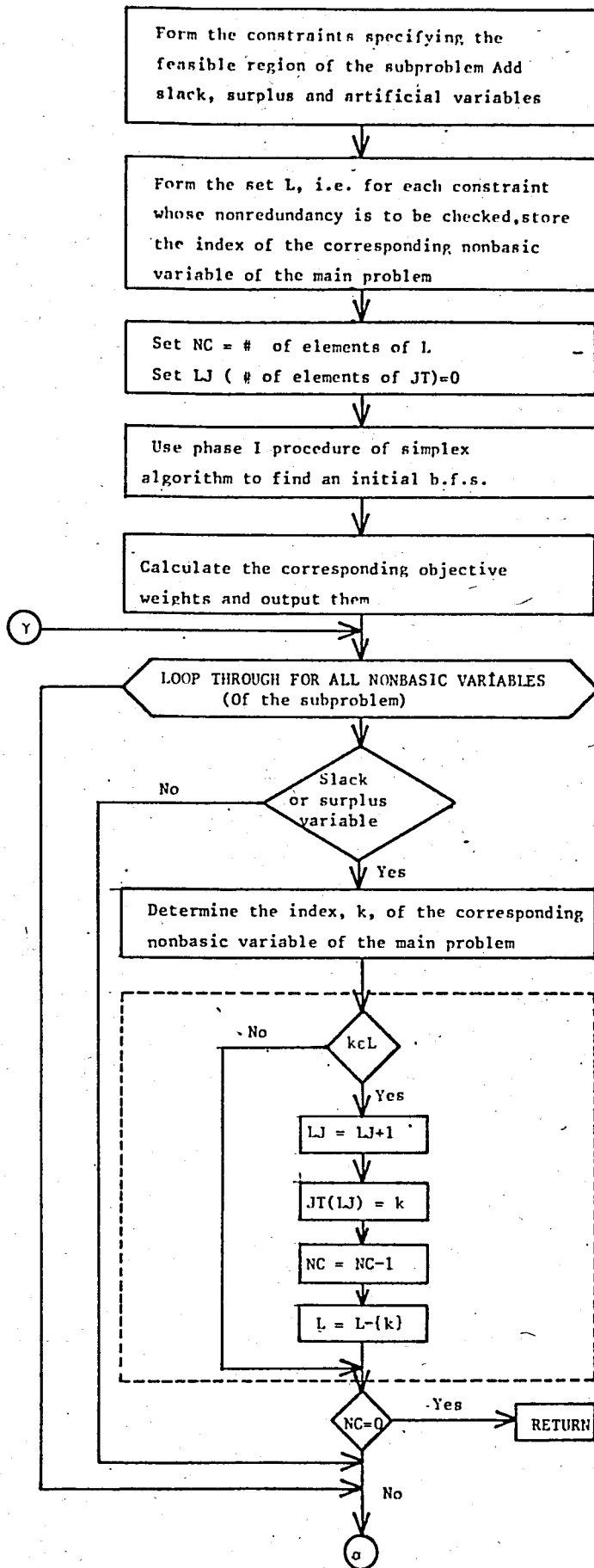
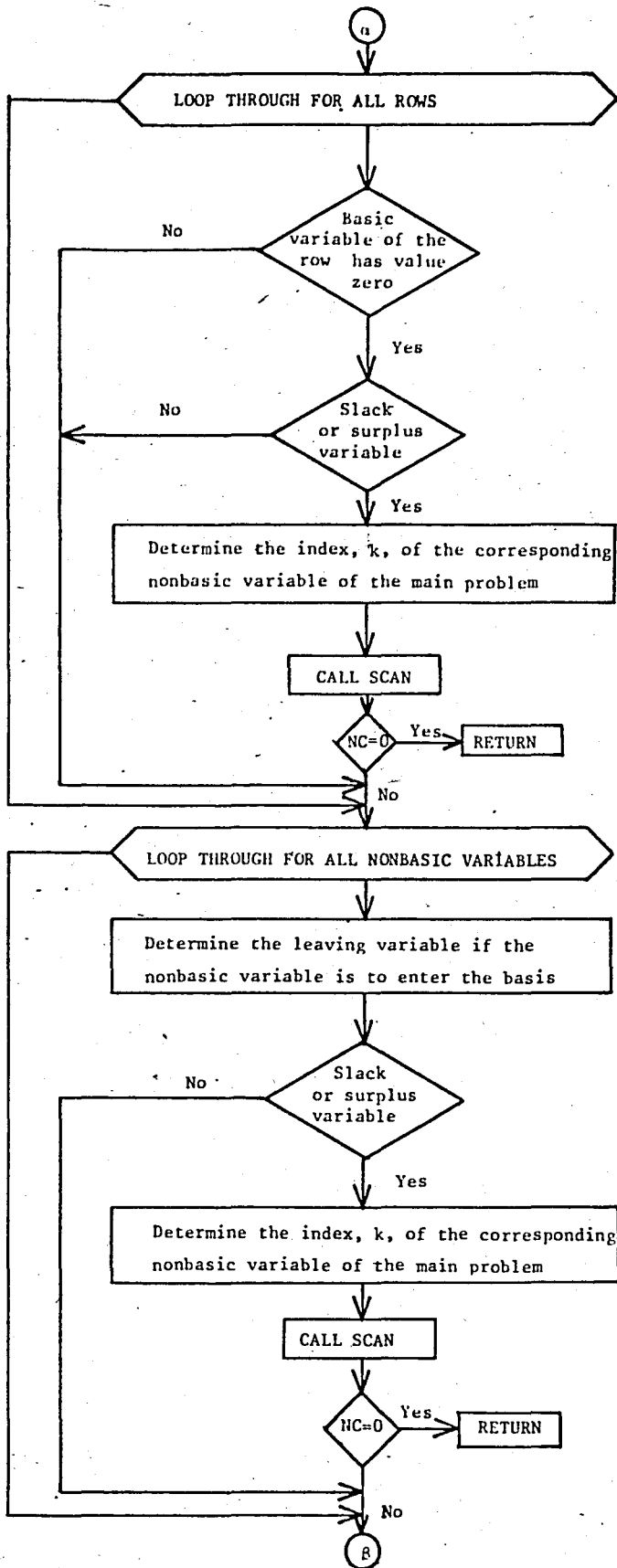


Figure 1.4. i) Flow Diagram of the MAIN Program



SUBROUTINE
SCAN



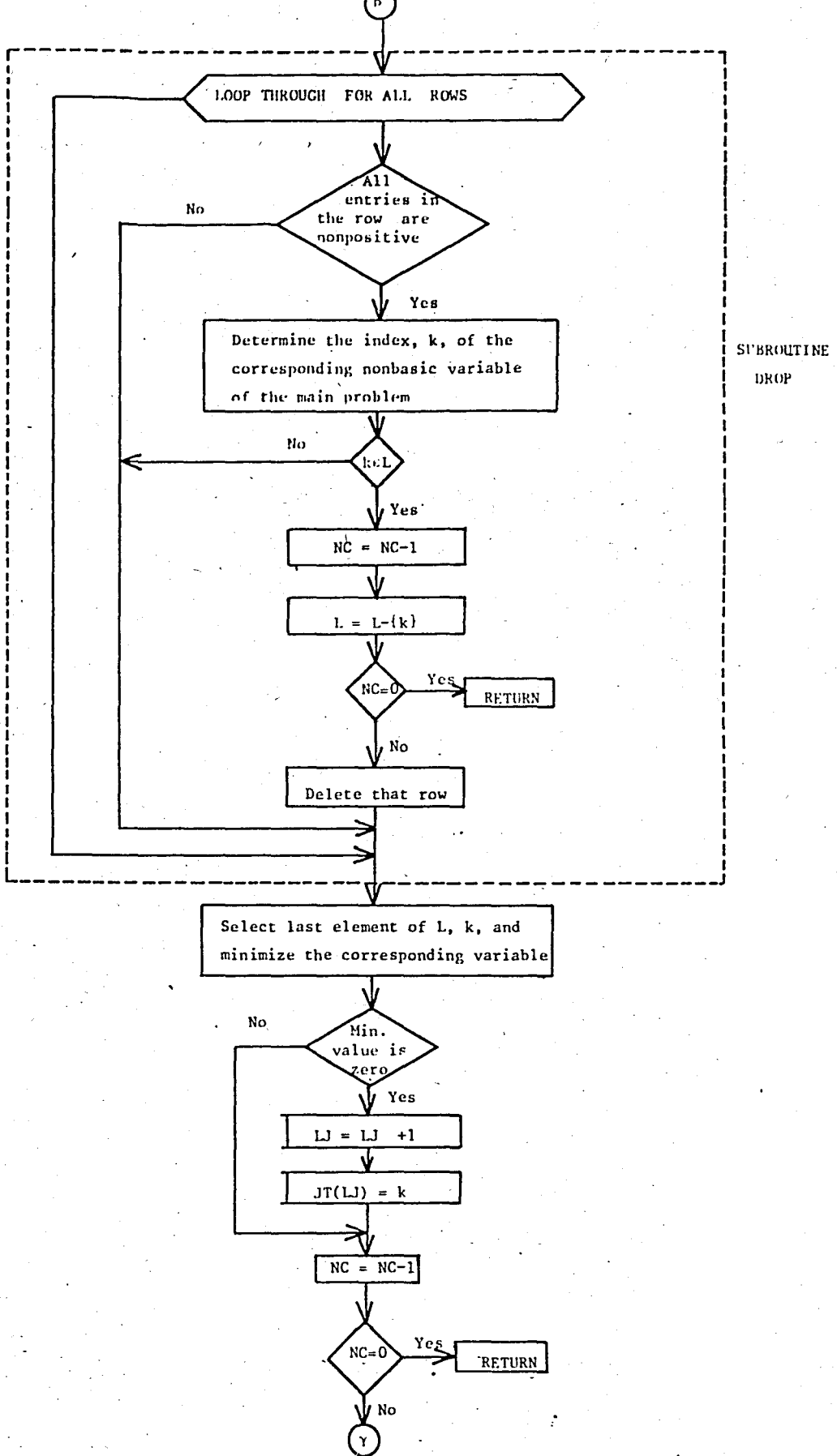


Figure I.4. ii) Flow Diagram of Subroutine EDGE and subroutines SCAN and DROP (called by EDGE.)

Table I.1. Sample Problem Results

Problem No	Problem p	Size m	n	No.of Efficient Extreme Pts.	CPU (msec)
1	2	4	6	1	257
2	2	5	8	3	701
3	2	6	10	2	572
4	2	7	12	2	679
5	2	8	14	3	1020
6	2	9	16	9	2804
7	2	10	18	7	2534
8	2	11	20	9	3441
9	2	12	22	5	2359
10	2	13	24	2	1520
11	3	4	6	1	259
12	3	5	8	3	736
13	3	6	10	1	357
14	3	7	12	2	757
15	3	8	14	9	3681
16	3	9	16	7	2523
17	4	4	6	5	1177
18	4	5	8	11	2719
19	5	4	6	2	502
20	5	5	8	3	825

3.1. OBSERVATIONS ON THE EFFICIENCY CHECK

The implications of dual feasibility for bicriterion problems and a simple check of dual feasibility of adjacent bases which requires only a series of divisions and comparisons are presented below. First, it is observed that the nonbasic variable set of any dual feasible basis can be partitioned into component subsets as follows:

Lemma I.1. Assume a dual feasible basic solution with corresponding reduced cost matrix C is given. Let N be the index set of nonbasic variables. Then $N=Q \cup R \cup S \cup T$ where

$$Q = \{j \in N \mid C_{1j} < 0, \quad C_{2j} > 0\}$$

$$R = \{j \in N \mid C_{1j} > 0, \quad C_{2j} < 0\}$$

$$S = \{j \in N \mid C_{1j} \geq 0, \quad C_{2j} \geq 0; \quad C_{1j} \text{ and } C_{2j} \text{ not both zero}\}$$

$$T = \{j \in N \mid C_{1j} = C_{2j} = 0\}$$

Proof: Given $j \in N$, observe that j either belongs to one of the sets Q, R, S or T or $j \in D$ where $D = \{j \in N \mid C_{1j} \leq 0, \quad C_{2j} \leq 0; \quad C_{1j} \text{ and } C_{2j} \text{ not both zero}\}$. We need to show $D = \emptyset$.

Assume $\exists k \in D$. Then we have

$$\lambda^T C^k = \lambda_1 C_{1k} + \lambda_2 C_{2k} < 0 \quad \forall \lambda > 0$$

which contradicts the dual feasibility of the given basis. Therefore $D = \emptyset$.

Next, we give a necessary and sufficient condition for a given basis to be dual feasible.

THEOREM I.19.

- i) A given basic solution with $Q \neq \emptyset$ and $R \neq \emptyset$ is dual feasible if and only if $D = \emptyset$ and $C_{1j}/C_{2j} \geq C_{1k}/C_{2k} \quad \forall j \in Q$ and $\forall k \in R$.
- ii) A given basic solution with either $Q = \emptyset$ or $R = \emptyset$ is dual feasible if and only if $D = \emptyset$.

Proof.

- i) Observe from proof of Lemma I.1. That if $D \neq \emptyset$ the given basis is not dual feasible. Therefore assume $D = \emptyset$.

For $j \in Q$:

$$\lambda^T C^j = \lambda_1 C_{1j} + \lambda_2 C_{2j} \geq 0 \quad \text{if and only if} \quad -\lambda_2 / \lambda_1 \leq C_{1j} / C_{2j} .$$

For $k \in R$:

$$\lambda^T C^k = \lambda_1 C_{1k} + \lambda_2 C_{2k} \geq 0 \quad \text{if and only if} \quad -\lambda_2 / \lambda_1 \geq C_{1k} / C_{2k}$$

For $s \in S$:

$$\lambda^T C^s > 0 \quad \forall \lambda > 0 .$$

For $t \in T$:

$$\lambda^T C^t = 0 \quad \forall \lambda > 0 .$$

Therefore $\lambda^T C \geq 0$. (condition for dual feasibility) if and only if

$$-\lambda_2 / \lambda_1 \leq \min_{j \in Q} C_{1j} / C_{2j} \quad \text{and} \quad -\lambda_2 / \lambda_1 \geq \max_{k \in R} C_{1k} / C_{2k}$$

The two inequalities together imply that $C_{1j} / C_{2j} \geq C_{1k} / C_{2k}$

$\forall j \in Q$ and $\forall k \in R$.

ii) As in i) above, if $D \neq \emptyset$, the given basis is not dual feasible.

Assume $D = \emptyset$ and $Q = \emptyset$. This implies $C_{1j} \geq 0 \quad \forall j \in N$ and $C_{2j} \geq 0 \quad \forall j \in N$ such that $C_{1j} = 0$. Then, by choosing λ_1 and $\lambda_2 > 0$, λ_1 of a sufficiently large magnitude, we can have

$$\lambda_1 C_{1j} + \lambda_2 C_{2j} \geq 0 \quad \forall j \in N$$

Therefore, the given basis is dual feasible.

Similar reasoning applies for $R = \emptyset$.

From the proof of Theorem I.19 it should be noted that a given basis remains dual feasible for any λ satisfying

$$-\lambda_2/\lambda_1 \in \left[\max_{k \in R} C_{1k}/C_{2k}, \min_{q \in Q} C_{1q}/C_{2q} \right]$$

That is, a given basic solution is the optimal solution for any set of objective weights whose ratio is within the above interval.

The following theorem provides a simple check for determining λ -adjacent dual feasible bases which is used in the bicriterion algorithm.

THEOREM I.20. Given a dual feasible basic solution, the basic solution obtained by introducing x_j into basis is a λ -adjacent dual feasible basic solution if and only if either.

$$i) C_{1j}/C_{2j} = \min_{q \in Q} C_{1q}/C_{2q}$$

$$\text{or } ii) C_{1j}/C_{2j} = \max_{k \in R} C_{1k}/C_{2k}$$

$$\text{or } iii) j \in T$$

Proof. As the given basis is dual feasible $\exists \lambda > 0$ such that $\lambda^T C \geq 0$. Moreover, we know that λ has to satisfy the following:

$$\max_{k \in R} c_{1k}/c_{2k} \leq -\lambda_2/\lambda_1 \leq \min_{q \in Q} c_{1q}/c_{2q} \quad (1)$$

Then, by Theorem I.18, the solution obtained by making x_j basic is a λ -adjacent dual feasible solution if and only if $\lambda^T C^j = 0$ for some $\lambda > 0$ satisfying (1).

Now, for $j \in S$

$\lambda^T C^j > 0 \quad \forall \lambda > 0$, Therefore x_j with $j \in S$ cannot lead to a λ -adjacent dual feasible basic solution.

For $j \in Q$ or $j \in R$

$$\lambda^T C^j = \lambda_1 C_{1j} + \lambda_2 C_{2j} = 0 \quad \text{if and only if} \quad -\lambda_2/\lambda_1 = C_{1j}/C_{2j} \quad (2)$$

Together (1) and (2) imply that the solution obtained by introducing x_j into the basis is a λ -adjacent dual feasible solution if and only if

$$\text{i) For } j \in Q \quad C_{1j}/C_{2j} = \min_{q \in Q} C_{1q}/C_{2q}$$

$$\text{ii) For } j \in R \quad C_{1j}/C_{2j} = \max_{k \in R} C_{1k}/C_{2k}$$

To show part iii) we observe that $\lambda^T C^j = 0 \quad \forall j \in T$.

Now, the bicriterion algorithm which uses these simple checks, which require only a series of divisions and comparisons, for determining λ -adjacent dual feasible bases can be presented.

I.3.2. DEVELOPMENT OF THE ALGORITHM

The bicriterion algorithm is similar to the multiobjective algorithm described in section I.2.2. The main differences are in checking the dual feasibility of the initial efficient basic solution, where simply part ii) of Theorem I.19. is used; and in identifying λ -adjacent dual feasible bases where Theorem I.20. is used. The general structure and the bookkeeping

of the bicriterion algorithm is the same as that of the multiobjective algorithm. The main steps can be given as follows; where the various sets are as defined in section I.2.2. for the MOLP algorithm.

1. Find initial dual feasible basis (which maximizes objective 2) with corresponding nonbasic variable index set N .
2. Set $V_1 = \phi$; $V_2 = \{N\}$.
3. Determine the set R using part i) of Theorem I.20. If $R = \phi$ go to step 8.
4. Set $R_1 = R - V_2$. If $R_1 = \phi$ go to step 8.
5. Set $R_2 = R_1 - V_1$. If $R_2 = \phi$ go to step 7.
6. Take last element of R_2 , denoted by N , and pivot to the associated basis. Set $V_1 = V_1 \cup R_2 - \{N\}$ and $V_2 = V_2 \cup \{N\}$. Go to step 3.
7. Take last element of R_1 , denoted by N , and pivot to the associated basis. Set $V_1 = V_1 - \{N\}$ and $V_2 = V_2 \cup \{N\}$. Go to step 3.
8. If $V_1 = \phi$ stop. All dual feasible bases have been generated. Otherwise, take last element of V_1 , denoted by N and pivot to the associated basis. Set $V_1 = V_1 - \{N\}$ and $V_2 = V_2 \cup \{N\}$. Go to step 3.

Here, initially the second objective is maximized. If there are no alternative solutions, one has a dual feasible basis at hand. Otherwise, it is checked whether $D = \phi$, i.e., if there is a $j \in N$ such that $C_{2j} = 0$ and $C_{1j} < 0$. If $D \neq \phi$, then x_j for some $j \in D$ is introduced into the basis. The same procedure is repeated until the situation $D = \phi$ is reached. If $D = \phi$, then by Theorem I.19. part ii) one has a dual feasible basis.

Next, the nonbasic variable(s) x_j , $j \in Q$ satisfying condition i) of Theorem I.20. and x_j , $j \in T$, if any, are determined. If there are ties for the entering variable, one of these, x_k , is selected and a pivot is performed where x_k enters into the basis, provided that the corresponding basis has not already

been generated. The remaining bases are stored to be generated later. When no new basis can be identified, the algorithm stops.

It is to be noted that part ii) of Theorem I.20. is not being used. This is due to the fact that after finding the initial dual feasible basis by maximizing the second objective, one needs to consider only those λ -adjacent dual feasible bases where the value of the second objective decreases or remains the same.

The intuitive reasoning behind the algorithm is that given an efficient point, in order to move to an adjacent efficient point, one should move in a direction where the rate of decrease of the second objective is a minimum while the rate of increase of the first objective is a maximum.

Assume there are no alternative efficient solutions, i.e. no two efficient solutions give the same objective vector, which is the case if there are no ties for the entering variable. Then, the algorithm starts from the basic solution maximizing the second objective and eventually moves to the basic solution maximizing the first objective and stops there. Then, the set of all efficient solutions will consist of a set of efficient edges which can be obtained as the convex combinations of two adjacent basic solutions recorded by the algorithm.

I.3.3. THE COMPUTER PROGRAM AND COMPUTATIONAL RESULTS

The bicriterion algorithm is also coded in FORTRAN IV. Again, either all efficient extreme points or only those extreme points satisfying the interval limits on objective weights, if these are specified, are enumerated. The computer program consists of a MAIN program and five subroutines which are called by the MAIN program.

The MAIN program is similar to that in the multiobjective computer program except for two sections. One is where the dual feasibility of the initial efficient solution is checked. The other is in finding the initial efficient solution for the case where intervals on objective weights are specified, where a composite objective function formed by assigning the second objective the maximum possible weight is maximized. Now, there is no need

for a separate subroutine, like the subroutine WEIGHT in the multiobjective program, to form the inequalities implied by intervals on objective weights. Simply, in the subroutine EDGE2 the ratio $-C_{1q}/C_{2q}$, which gives the lower bound for the objective weights ratio λ_2/λ_1 pertaining to the dual feasible basis obtained by introducing into basis the nonbasic variable x_q , is checked for being in the allowable region.

Subroutine EDGE2 performs the same function as subroutine EDGE of the multiobjective program, namely identifying the nonbasic variables that lead to λ -adjacent dual feasible bases with nondecreasing values of objective 2. It is however much simpler, the check given by part i) of Theorem I.20 is carried out.

The subroutines LEAV, PIVOT, MOVE and SORT are the same as for the multiobjective computer program.

Computational results for the same sample of randomly generated problems with two objectives are given in Table I.2. along with the computation times with the multiobjective algorithm for comparison purposes. It can be observed that the bicriterion algorithm is about 2-2.8 times faster than the general multiobjective algorithm.

Listing of the computer program, data input instructions and a sample output is given in Appendix II.

In comparison with the existing algorithms for bicriterion problems, the proposed algorithm has much lower computational requirements. The number of pivots needed is the same as for parametric linear programming as proposed by Geoffrion [27]. However, the computations needed to determine the entering variable(s) are much simpler. Here, only a number of divisions and comparisons ($<r$ where r is the number of nonbasic variables) are needed, whereas the sensitivity analysis for determining the entering variables requires more computational effort.

In comparison with the algorithms of Aneja and Nair [2] or Cohon et al. [14], the proposed algorithm is definitely superior. When there are $k(k>2)$

Table I.2. Sample Problem Results

Problem No	Problem m	Size n	No. of Efficient Extreme Pts.	CPU Bicrtn. (msec)	CPU M.Obj. (msec)
1	4	6	1	95	257
2	5	8	3	303	701
3	6	12	2	247	572
4	7	14	2	292	679
5	8	16	3	514	1020
6	9	18	9	1114	2804
7	10	20	7	931	2534
8	11	22	9	1362	3441
9	12	24	5	909	2359
10	13	26	2	648	1520

efficient points, these algorithms need $2k-1$ iterations, where the linear programming problem is solved with a different objective function at each iteration. In contrast, if there are no alternative efficient solutions (which is the more common case) the algorithm proposed here needs only one such iteration and $k-1$ additional pivots. The algorithm goes from the maximizing basic solution for one objective to the maximizing basic solution for the other, and enumerates all efficient extreme points with about as much effort as needed for the initialization of the algorithms cited above.

PART II. AN APPLICATION IN POWER SYSTEMS PLANNING

In recent years, with the emergence of new trends in energy systems analysis, the decision making process in electrical power system investments has become increasingly more complex. One of the new trends is the environmentalist movement while another is the concern over nuclear technology. The oil embargo of 1973; the sudden increase in energy prices and the threat of future energy shortages have caused these trends to have more significant impact on the decision making process. Supply risks associated with oil fired power plants have meant more coal and nuclear plants. These plants, in turn, give rise to even greater opposition by the environmentalist and the antinuclear groups.

The reservations voiced by these groups have already influenced the planning and operation of power plants. The plebiscites in Federal Germany, Austria and Sweden are clear evidences of direct public participation in the decision making process. Up to recent times, power systems planning was carried out by central electricity generating authorities. The main concern was economic efficiency, subject to certain technical considerations, while environmental and social factors were not influential. Today, this picture is changing rapidly with the emergence of different interest groups with different and conflicting objectives. Consequently, the need for developing and applying new planning procedures and mathematical techniques arises. These new procedures and techniques will evolve slowly after sufficient experience has accumulated and the strengths and limitations of each technique has been realized. Here one possible approach that attempts to accommodate several objectives in power systems expansion decisions is presented. The approach is based on generating relevant decision alternatives through the use of a multiobjective linear programming model. The efficient solutions of the model are grouped based on the clustering of the objective values and the similarity of the decision implications for the immediate future. The model has been applied to the Turkish electrical system.

II.1. MODELLING THE POWER SYSTEM EXPANSION PROBLEM

Modelling of the power system expansion problem is quite a complicated task due to the complexity of the system. A variety of power plants with different fixed and variable costs, availability factors, capacities, etc. form the supply system. The demand, on the other hand, is subject to fluctuations at different seasons, months, days and even hours of the day, and wide variations due to unforeseen events. Furthermore, the expansion problem must be considered in conjunction with the operating program of all power plants if suboptimal decisions are to be avoided.

The uncertainties in various elements and the size and complexity of the system make it almost impossible to accommodate all aspects within a single model. Once the need for decomposition is recognized, one has to decide on the level at which decomposition should be done. A natural approach is to first decide on the global expansion strategy, and then to plan a detailed project evaluation program. The aim of this study is to focus on the first phase where several objectives are influential.

The model given below is a modified version of a dynamic linear programming model which had been applied to the electrical system in Turkey [35], [36], [37]. In its present simplified form, only three types of power plants are included; coal, hydro, and nuclear. The model extends over five planning periods of six years' duration each. All technical and financial aspects are expressed as linear relations. Three objectives are considered to have deciding influence; namely economics, environmental impact and potential damage. Economic efficiency has been the traditional planning objective in public investment problems and will continue to be a determinant in any investment decision, be it public or private. Environmental impact combines several factors such as land use, chemical, thermal and radioactive pollution. Potential damage accounts for the probability of an accident and the extent of damage that such an accident would cause. These three objectives which are to be minimized are expressed as:

$$z_1 = \sum_{t=1}^5 \left(\frac{1}{1+r} \right)^t \sum_{i=1}^3 \alpha_i P_{it} + \alpha'_i E_{it} \quad (1)$$

$$z_2 = \sum_{t=1}^5 \sum_{i=1}^3 \beta_i E_{it} \quad (2)$$

$$z_3 = \sum_{t=1}^5 \sum_{i=1}^3 \gamma_i P_{it} \quad (3)$$

where P is added new power capacity, E is energy generated; α and α' are unit costs of installed power and energy generation; β and γ are environmental impact and potential damage coefficients; r is discount rate (over a period); i is power plant type (i=1 for coal, i=2 for hydro and i=3 for nuclear), and t is the time period. Here energy generated is taken as a surrogate for environmental impact and power installed as a surrogate for potential damage.

The constraints of the model can be given in general form as follows:

- i) Energy Demand: Demand for electrical energy (ED) must be satisfied at all time periods.

$$\sum_{i=1}^3 E_{it} \geq ED_t \quad ; \quad \forall t \quad (4)$$

- ii) Power Demand: Power demand (PD) must be satisfied at all time periods.

$$\sum_{j=0}^t \sum_{i=1}^3 P_{ij} \geq PD_t \quad ; \quad \forall t \quad (5)$$

where P_{i0} gives the initial available power capacity for plant type i.

- iii) Production Capacity: Amount of energy that can be generated at time t cannot exceed that allowed by the available power capacity

$$E_{it} \leq f_i \sum_{j=0}^{t-1} P_{ij} \quad ; \quad \forall t, \quad \forall i \quad (6)$$

where energy production is lagged by one time period (6 years) with respect to the implementation of a new project, and f_i denotes the maximum load factor of the corresponding power plant.

- iv) Build-up Rate: The amount of new capacity added for each type of power plant is limited by the development of technical capacity

$$P_{it} \leq K_i \cdot P_{i,t-1} \quad ; \quad \forall t, \forall i \quad (7)$$

where K_i is the capacity expansion coefficient for corresponding type of power plant. Because different types of technological capabilities are involved, each type of power plant is treated separately.

- v) Hydro Limitation. The hydroelectricity that can be generated is restricted by the hydropotential (HP) developed upto that time.

$$E_{2t} \leq HP_t \quad ; \quad t=4,5 \quad (8)$$

Hydroelectricity is not restricted for $t \leq 3$, because power capacity is already constrained by (7) above.

In developing this model, it was further assumed that for Turkey nuclear energy cannot contribute to the supply mix earlier than the second period and coal reserves and/or imports will not create any problems within the planning horizon. The first assumption requires minor changes in expressions (4) through (8). Specifically the variables E_{31} , E_{32} and P_{31} are deleted and P_{32} becomes a parameter giving the initial nuclear capacity. In its final form, the model has 26 variables and 37 constraints.

Uncertainties in model elements can be handled through scenario and/or sensitivity analyses. The planning horizon of 30 years will be adequate. It is not necessary to consider developments too far into the future as investment decisions are made every fiscal year. Depending on changes and unforeseen events, the analysis may be repeated to update the results.

The mathematical modelling of the power system expansion problem as a multiobjective linear program given above is the initial phase of the overall solution procedure explained in detail in the next section.

II.2. SOLUTION PROCEDURE

It was expressed before that there are three basic approaches to the multiobjective decision making process. Reviewing, these are: a priori articulation of preferences leading to the formation of a single overall objective function; generating efficient solutions and a posteriori articulation of preferences; or progressive articulation of preferences, i.e. interactive approaches. In evaluating these approaches for their relevance in actual decisions, the characteristics of the decision environment or the type of problem at hand carries utmost importance. A decision process that is highly successful in a given environment may be totally inadequate in another. Viewed from this point, the interactive approach seems very promising for decisions in a restricted environment where there is a well-defined decision maker who is actually able to cooperate. In most socioeconomic systems, however, there are several decision makers who may even be difficult to identify. In such cases, not only the interactive methods, but also a priori specification of preferences may become inoperational. In view of the complexity of the decision environment, nonexistence of a unique well-defined decision maker, and the far-reaching consequences of the power system investments program, the approach that is considered most appropriate for this particular problem is the generating approach which postpones articulation of preferences to later stages of the decision process and provides a sound information base.

The main concern in this study has been to generate relevant decision alternatives that are only a few in number. Recognizing that the generating approach usually leads to a large number of alternatives, model dimensions were kept as small as possible. Keeping model size small is consistent with decisions of a global nature where only aggregate quantities are of interest, in addition to the fact that both the number of efficient solutions as well as the computational effort required for each solution

are reduced. Furthermore, it is easier to develop and implement efficient algorithms for models with small dimensions,

The influence of uncertainties in model parameters or policy options were analyzed by developing appropriate scenarios. Sensitivity analyses were carried out to determine the significance of environmental impact and potential damage coefficients for which absolute values are difficult to determine.

The overall solution procedure employed is depicted in Figure II.1.

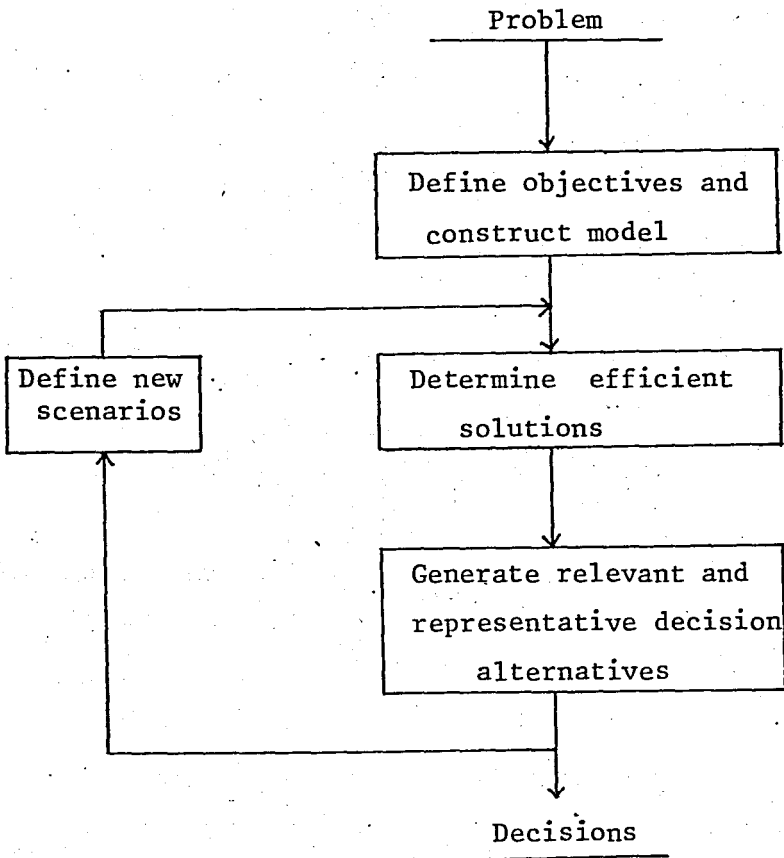


Figure II.1. The Overall Solution Procedure

The MOLP algorithm presented in Part I was used for the generation of efficient solutions. The algorithm generates a representative set of efficient solutions, i.e., efficient extreme points, with quite low computational effort. The efficient solutions obtained may then be grouped so as to generate relevant decision alternatives representing different preferences. The grouping of solutions may be based on the relative magnitudes of the objective values, or on the clustering in the values of the decision variables or on the implied weights associated with the objectives. In this study a combination of the first and second methods was used in grouping the solutions.

Another approach was the consideration of only the economics and environmental impact objectives and solving the model for a given scenario as a bicriterion problem.

II.2.1. APPLICATION OF THE MODEL WITH THREE OBJECTIVES

Four scenarios were developed in order to assess the consequences of certain policy implications. The scenarios tested were:

- A: Base case
- B: Restricted hydro
- C: Low energy demand
- D: Less restricted nuclear

The parameters and coefficients of the model for the base case were specified so that

- Energy and power demands were taken to increase at their historical rates;
- Maximum load factors for coal, hydro and nuclear power plants were taken as 0.70, 0.45 and 0.60 respectively;
- Build-up rate factor was taken as 1.5 for nuclear plants and 2.0 for other plants;
- Initial coal and hydro capacities were both assumed to be 4.0 Gw.
- Total nuclear capacity in the second period was assumed to be 1.0 Gw.

In determining the objective coefficients; the investment cost was taken as the sum of power plant and transmission costs. Similar models usually consider only power plant costs [1]. In reality, transmission costs are almost as high as plant costs and may vary significantly depending on the distance that the energy is transmitted to. In Turkey, where the hydro potential is rather distant from the main load centers, the largest unit transmission costs are incurred for hydroelectricity. Lowest transmission costs apply for coal power plants while nuclear plants would entail costs of intermediate value.

Environmental impact and potential damage aspects are not as easily quantified as financial aspects. For these two factors, an ordinal ranking was made and sensitivity analyses were carried out on the first two scenarios in order to observe the influence of the relative magnitude of the coefficients assigned to each type of power plant on the results obtained. The ordinal ranking of power plants with respect to these factors and financial aspects is given in Table II.1. from highest to lowest in going down the table.

Table II.1. Ranking of Power Plants

Cost		Environmental impact (β)	Potential damage (γ)
Fixed (α)	Variable (α')		
Nuclear	Coal	Coal	Hydro
Hydro	Nuclear	Hydro	Nuclear
Coal	Hydro	Nuclear	Coal

Environmental impact, as defined here, refers to air pollution caused by the burning of coal; land covered by the reservoir of a hydroelectric plant; and radioactive emissions from a nuclear power plant. Although these indicators imply different dimensions and necessarily entail a certain subjective evaluation, the ranking given in Table II.1. may be representative

of the assessment of most analysts. The public in general would also view a coal power plant as more objectionable in comparison with a hydro project, especially when low quality, high sulfur content brown coal is considered. Furthermore the land allocated to a coal plant is not insignificant. Few would argue that the radiation (or the excess heat release) of a nuclear power plant affects the environment anywhere nearly as much as the chemical pollution caused by the burning of fossil fuels. The area required by the nuclear plant is usually much less than a coal or a hydroplant.

As regards potential damage, the ranking given above may appear counterintuitive. Most people would probably react by asserting that the nuclear plant is more objectionable than the hydro plant in this aspect. It should be noted, however, that quite a few dams have failed whereas no significant physical damage has so far been caused by a nuclear plant. That is, the probability of failure of a nuclear plant is lower. In the event of a failure, the number of lives that would be affected is of the same order of magnitude for either type of plant.

The steps of the solution strategy followed in applying the model to the Turkish electrical system are summarized below.

The BASE CASE scenario was solved for three different sets of objective coefficient (β and γ) assumptions, which are given below.

$$1. \text{ Set } \frac{\beta_1}{\beta_3} = 3 \quad \frac{\beta_2}{\beta_3} = 2 \quad ; \quad \frac{\gamma_2}{\gamma_1} = 3 \quad \frac{\gamma_3}{\gamma_1} = 2$$

$$2. \text{ Set } \frac{\beta_1}{\beta_3} = 9 \quad \frac{\beta_2}{\beta_3} = 3 \quad ; \quad \frac{\gamma_2}{\gamma_1} = 9 \quad \frac{\gamma_3}{\gamma_1} = 3$$

$$3. \text{ Set } \frac{\beta_1}{\beta_3} = 16 \quad \frac{\beta_2}{\beta_3} = 4 \quad ; \quad \frac{\gamma_2}{\gamma_1} = 16 \quad \frac{\gamma_3}{\gamma_1} = 4$$

The efficient solutions in the decision space were identical for each of these sets, except for one efficient point which was not included in the solutions for set one. Furthermore, the points where each objective

attained its minimum were exactly the same. This led to the conclusion that results were not much sensitive to the changes in the environmental impact and potential damage coefficients.

The RESTRICTED HYDRO scenario was developed next. Observing that in some solutions hydrolic energy generation in the fourth period was as high as 130Twh, which was thought to be too high, considering the hydrolic potential that could be utilized in the fourth period, hydrolic energy generation in the fourth period was constrained to a maximum of 100 Twh. For this scenario solutions with two different sets of objective coefficients β and γ given below, were obtained.

$$1. \text{ Set } \quad \frac{\beta_1}{\beta_3} = 3 \quad \frac{\beta_2}{\beta_3} = 2 \quad ; \quad \frac{\gamma_2}{\gamma_1} = 3 \quad / \quad \frac{\gamma_3}{\gamma_1} = 2$$

$$4. \text{ Set } \quad \frac{\beta_1}{\beta_3} = 1.2 \quad \frac{\beta_2}{\beta_3} = 1.1 \quad ; \quad \frac{\gamma_2}{\gamma_1} = 1.2 \quad \frac{\gamma_3}{\gamma_1} = 1.1$$

The first case resulted in 29 efficient extreme points and the second in 30, where 28 of the points were identical. This fact supported the conclusion that the results are not very sensitive to changes in the coefficients of objectives related to environmental impact and potential damage. From then on, for coefficients β and γ , the values given by set two above were adopted.

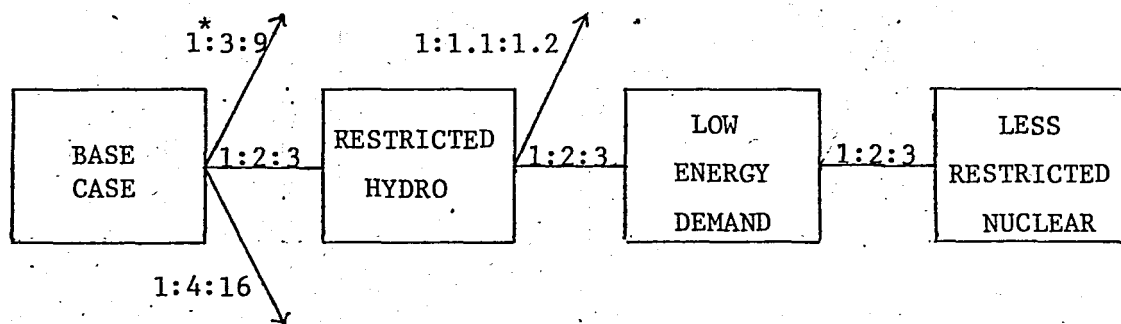
The LOW ENERGY DEMAND scenario was run next assuming lower rates of increase in energy and power demands, i.e. modeling a slower growth. This run resulted in 19 efficient extreme points.

The LESS RESTRICTED NUCLEAR scenario allowed for higher nuclear energy production by increasing the nuclear capacity expansion factor from 1.5 to 4.0. This scenario resulted in 29 efficient extreme points.

Then, the efficient solutions for each scenario were analyzed and grouped

together to form decision alternatives. Considering the dynamic structure of the model, it is not necessary to strictly lay down decisions too far into the future. Since the analysis may be repeated at will, decision implications for the near future have much greater impact than decision implications for later periods. Many of the solutions which imply different decisions in the later periods call for similar courses of action in the first few periods. Based on this observation, solutions with similar objective values and decision implications for the first three periods were grouped together forming a few decision alternatives.

The steps of the solution procedure are depicted schematically in Figure II.2.



* Numbers on branches represent coefficients of environmental impact and potential damage objectives.

Figure II.2. Solution Steps

A summary of solution statistics for the three most significant scenarios, B, C and D are given in Table II.2. Statistics of other results have not been included as different β and γ values were used, making direct objective value comparisons meaningless. Furthermore, the base case scenario is somewhat unrealistic since it allows for too high a hydro potential in the fourth period.

Table II.2. Solution Summary Statistics

Group No.	Number of Solutions	Ranges of Objective Values			Total Installed Capacity Ranges for the First Three Periods (Gw)						
		Econ.Cost (Billion TL)	Environ- mental Impact	Potentl. Damage	Hydro			Coal			Nuclear
					Period 1	Period 2	Period 3	Period 1	Period 2	Period 3	Period 3
B	29	1289-1536	1388-1737	41-137	4-12	4-28	4-33	5-9	7-17.2	11-27.8	1-2.5
C	19	1101-1311	1184-1533	37-133	4-12	4-28	4-33	4.6-7.3	5.8-13.8	8.2-25	1-2.5
D	28	1097-1336	1022-1533	37-156	4-12	4-28	4-33	4.2-7.3	4.5-13.8	5.3-25	1-5

An inspection of the statistics indicates that although the ranges of objective values are not too wide, the implications on the resulting power plant mix are significant. Results all the way from no capacity increase to quite high capacity increases for the first three periods can be observed for each type of plant. However, grouping of efficient solutions for any single scenario does not create significant problems. Detailed results for scenarios B, C and D are given in Tables II.3, II.4 and II.5 respectively.

For each scenario, solutions in Group I are evidently associated with least economic cost as well as least environmental impact; while those in Group IV are associated with least potential damage. In view of the ranking of the power plants given in Table II.1. these results may seem surprising. Summing up fixed and variable costs results in lowest unit total cost for hydro and highest for nuclear. Thus, each type of power plant ranks best in any given attribute and may be expected to be favored in solutions associated with minimal levels of that attribute; whereas power plants ranking poorly with respect to that attribute are disfavored. On closer inspection, however, it is observed that since total nuclear contribution is rather small, the competition is really between hydro and coal. Thus, Group I solutions where hydro is favored heavily result also in lowest environmental impact as hydro is superior to coal in that respect.

The investment scheduling implications, for each of the scenarios B,C, D, between the four strategies represented by the solution groups can be seen in Figures II.3, II.4. and II.5., where power capacity increases are plotted against time.

The decision maker is also supplied with additional information in the form of representative "objective weights" associated with each solution. This additional information is valuable especially for those more familiar with the classical approach of assigning weights (λ_i) to each objective. As an example, the set of all efficient extreme solutions for scenario C along with representative λ values associated with each efficient solution are presented in Table II.6.

The computation times for the various scenarios with the 1st. set of β and γ values are given in Table II.7.

Table II.3. Summary Statistics for Scenario B Solutions

Group No.	Number of Solutions	Ranges of Objective Values			Total Installed Capacity Ranges for the First Three Periods (Gw)						
		Econ. Cost (Billion TL)	Environ- mental Impact	Potentl. Damage	Hydro			Coal		Nuclear	
					Period 1	Period 2	Period 3	Period 1	Period 2	Period 3	Period 3
I	14	1289-1333	1388-1500	91-137	12	25.2-28	25.2-33	5-5.6	7-8.9	11-15.4	1-2.5
II	6	1409-1427	1581-1642	57-65	11.8-12	11.8-13.5	11.8-13.5	6.6-6.9	11.8-12.7	22.2-24.2	1-2.5
III	3	1439-1463	1630-1657	54-55	9-10.6	9-10.6	9-10.6	7.1-7.4	13.4-14.3	24-25	1-2.5
IV	6	1505-1533	1672-1737	41-48	4-5.6	4-5.6	4-5.6	8-9	16.2-17.2	26.6-27.8	1-2.5

Table II.4.. Summary Statistics for Scenario C Solutions

Group No.	Number of Solutions	Ranges of Objective Values			Total Installed Capacity Ranges for the First Three Periods (Gw)						
		Econ.Cost (Billion TL)	Environ- mental Impact	Potentl. Damage	Hydro			Coal			Nuclear
					Period 1	Period 2	Period 3	Period 1	Period 2	Period 3	Period 3
I	5	1101-1137	1184-1291	93-133	12	28	28-33	4.6-5.4	5.8-8.2	8.2-13.6	1-2.5
II	5	1162-1223	1304-1410	54-81	12	12-21.5	12-21.5	5.4-6.4	8.2-11.1	13.8-20.5	1-2.5
III	5	1243-1259	1436-1463	46-49	8-9.8	8-9.8	8-9.8	6.4-6.5	11.2-11.5	20.8-21.7	1-2.5
IV	4	1277-1311	1487-1533	37-41	4-6.4	4-6.4	4-6.4	6.8-7.3	12.4-13.8	23.7-25	1-2.5

Table II.5. Summary Statistics for Scenario D Solutions

Group No.	Number of Solutions	Ranges of Objective Values			Total Installed Capacity Ranges for the First Three Periods (Gw)						
		Econ. Cost (Billion TL)	Environ- mental Impact	Potentl. Damage	Hydro			Coal			Nuclear
					Period 1	Period 2	Period 3	Period 1	Period 2	Period 3	Period 3
I	14	1097-1140	1022-1291	93-156	12	28	28-33	4.2-5.4	4.5-8.1	5.3-13.6	1-5
II	6	1192-1255	1226-1436	54-77	10.7-12	10.7-15	10.7-15	5.6-6.3	8.9-11	15.5-20.5	1-5
III	3	1243-1254	1413-1463	48-49	8.1-9.8	8.1-9.8	8.1-9.8	6.3-6.5	11-11.5	19.5-21.5	1-5
IV	5	1277-1336	1323-1533	37-58	4-6.4	4-6.4	4-6.4	6.8-7.3	12.4-13.8	21.7-25	1-5

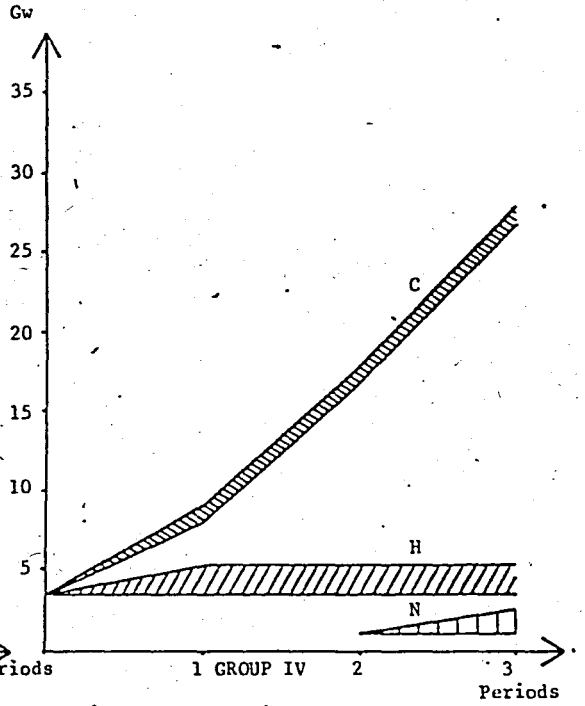
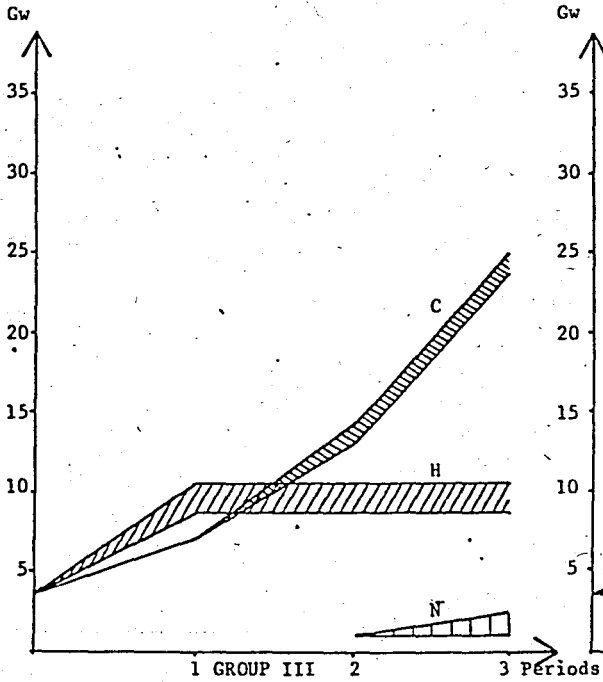
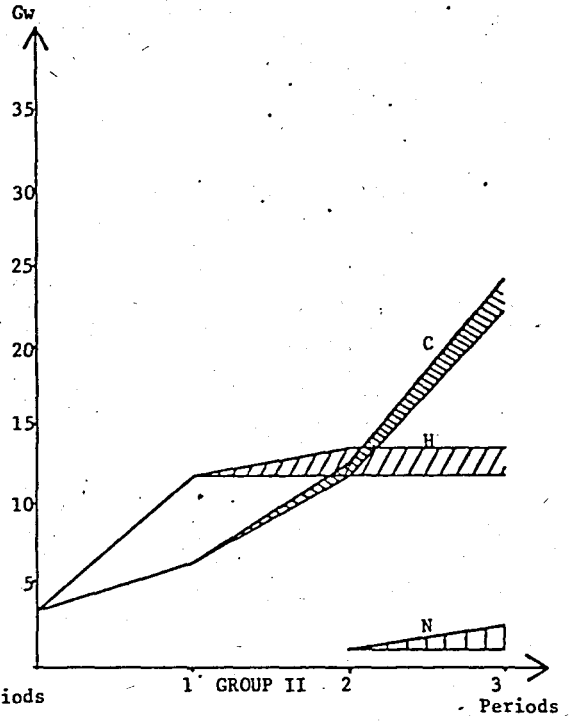
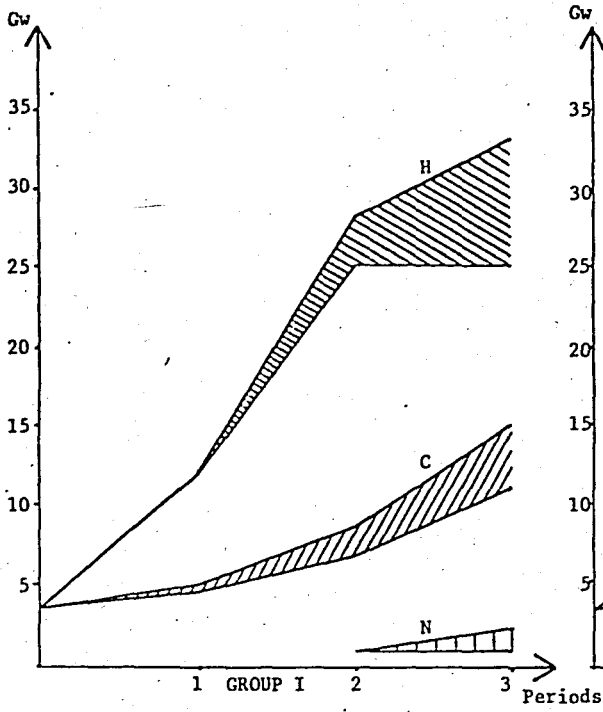


Figure II.3. Power Expansion Alternatives for Scenario B

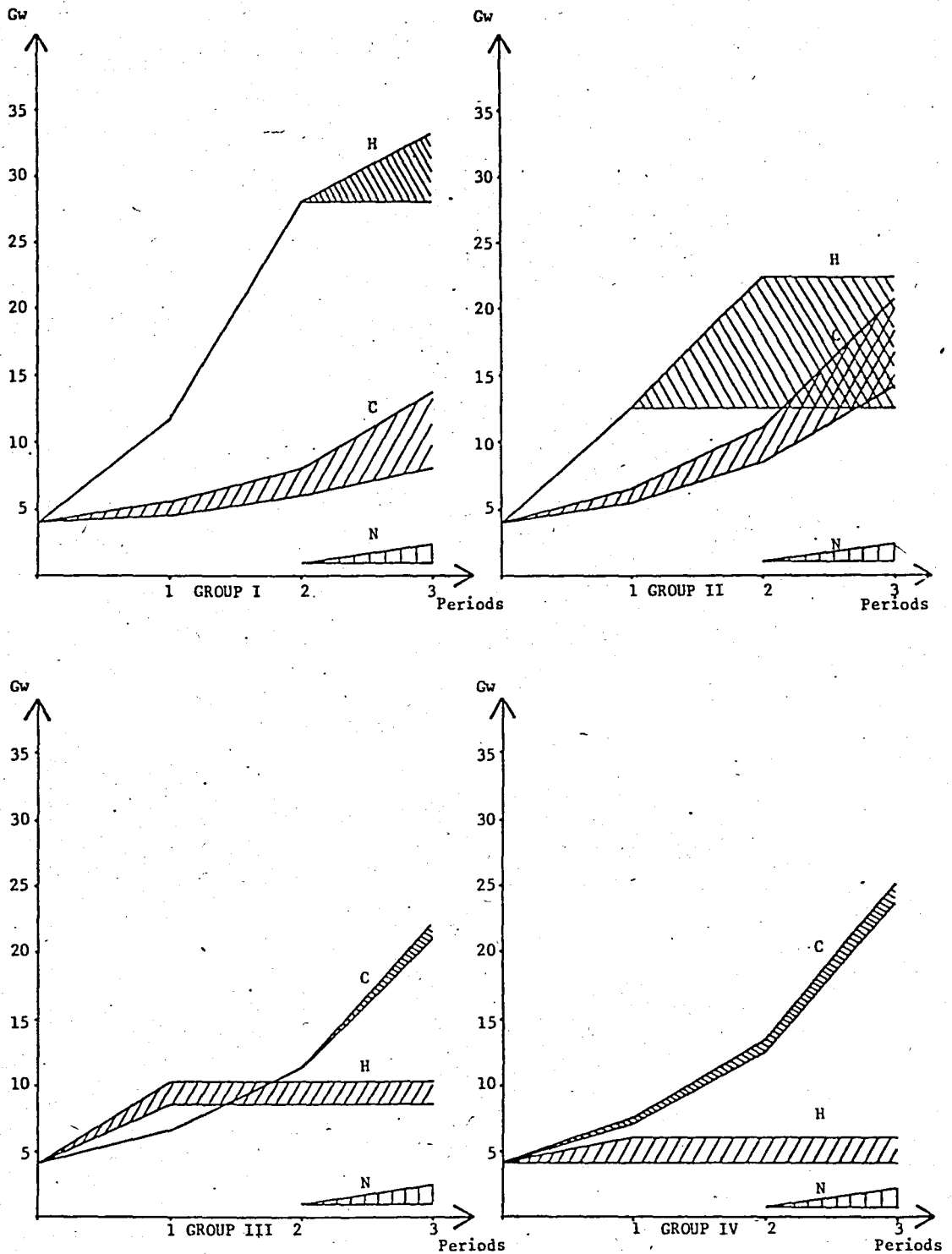


Figure II.4. Power Expansion Alternatives for Scenario C.

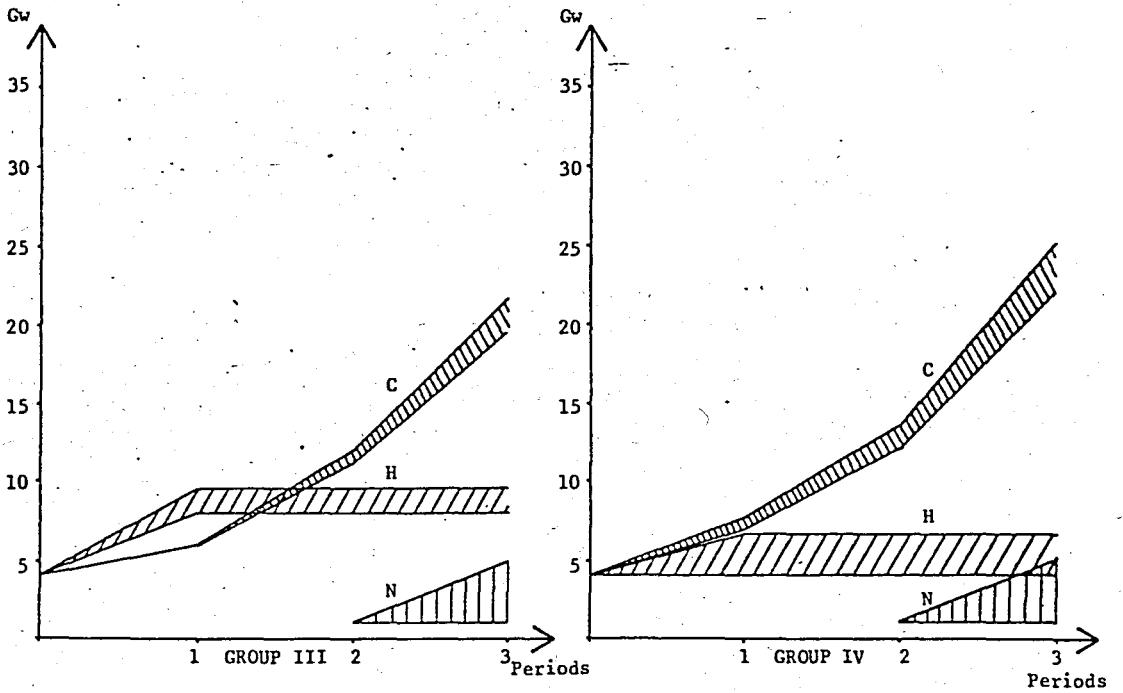
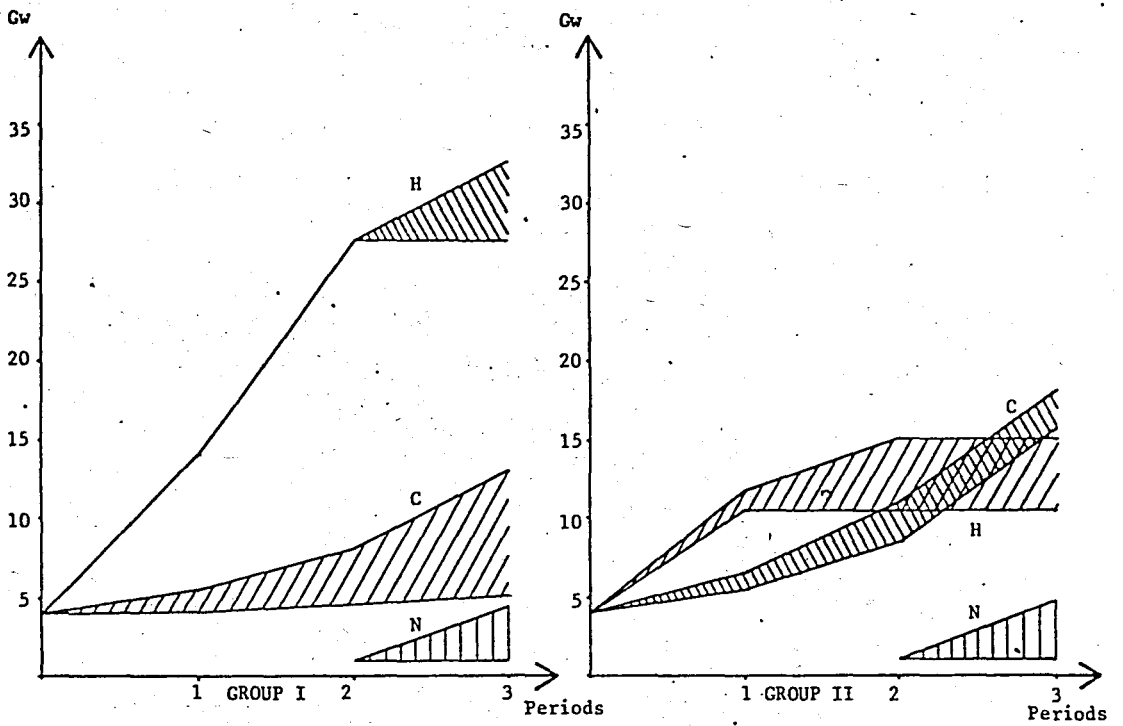


Figure II.5. Power Expansion Alternatives For Scenario D

Table II.6. Efficient Solutions and Associated Representative Weights λ_i for Scenario C

Group	Solution	Objective Values			Associated Representative Weights			Total Installed Capacity at 3rd period(Gw)		
		Economic Cost	Environml. Impact	Potential. Damage	λ_1	λ_2	λ_3	Hydro	Coal	Nuclear
I	1	1101	1184	133	.33	.33	.33	33	8.2	2.5
	2	1121	1214	109	.24	.24	.51	33	10.8	2.5
	19	1123	1260	105	.40	.04	.56	33	12.3	1.0
	3	1135	1245	97	.18	.18	.64	28	12.1	2.5
	18	1137	1291	93	.31	.05	.64	28	13.6	1.0
II	4	1162	1304	81	.14	.14	.73	21.5	13.8	2.5
	9	1176	1350	71	.12	.12	.76	18.6	15.5	2.5
	10	1219	1410	55	.11	.13	.76	12	19.3	2.5
	5	1223	1390	58	.11	.13	.76	12	19.3	2.5
	11	1223	1436	54	.19	.02	.75	12	20.5	1.0
III	12	1243	1463	48	.19	.02	.75	9.8	21.7	1.0
	16	1244	1444	49	.09	.09	.81	9.2	20.8	2.5
	17	1255	1458	46	.04	.10	.85	8	21.5	2.5
	6	1256	1436	49	.06	.12	.82	8.2	21.5	2.5
	7	1259	1439	48	.04	.11	.85	8	21.5	2.5
IV	13	1277	1504	41	.08	.05	.86	6.4	23.7	1.0
	15	1308	1507	38	.04	.10	.86	4	23.7	2.5
	14	1308	1533	37	.08	.05	.86	4	25	1.0
	8	1311	1487	40	.04	.11	.85	4	23.7	2.5

Note: $\sum_{i=1}^3 \lambda_i$ may not add up to 1.00 exactly, due to rounding error.

Table II.7. Computation Times for Model Application

Scenario	No. of efficient extreme points	cpu (secs.)
A	30	32
B	29	30
C	19	23
D	29	36

II.2.2. APPLICATION OF THE MODEL WITH TWO OBJECTIVES

Bicriterion, or two-objective analysis offers conceptual ease in that efficient solutions can be displayed graphically in a neat manner and the trade-off function between the two objectives can be observed. With these considerations, the model was solved for scenario D also as a bicriterion problem. Here, in addition to the traditional economics objective, the environmental impact objective, which has also become more or less classic, was taken.

In this solution, only five efficient extreme points were obtained. These solutions, along with the range of λ_1/λ_2 ratios for which they remain optimal solutions to the "weighted objective" problem are given in Table II.8.

These efficient solutions can be plotted as points on a graph, where the axes represent the two objectives. The plot of points when joined gives the trade-off function between these objectives, which is displayed in Figure II.6.

Here, the line segments joining two adjacent points also represent efficient solutions. The particular ratio λ_1/λ_2 for which these line segments are the optimal solutions to a weighted objective problem are indicated.

The values taken for the various parameters and coefficients of the model are given in Appendix III.

Table II.8. Efficient Solutions For Bicriterion Problem (Scenario D)

Economic Cost	Environmental Impact	Range for λ_1/λ_2
1097	1096	$[4.35, \infty]$
1100	1080	$[2.22, 4.35]$
1113.8	1050.2	$[1.52, 2.22]$
1113.9	1050	$[1.15, 1.52]$
1138	1022	$[0, 1.15]$

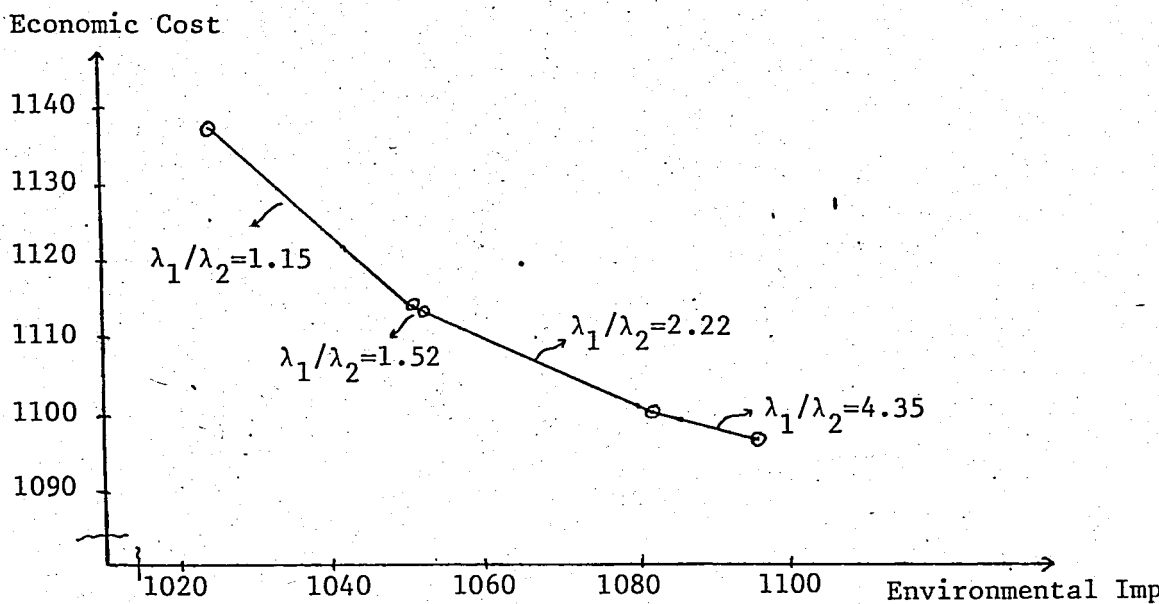


Figure II.6. Trade-off Function Between Economic Cost and Environmental Impact

11.5. EVALUATION OF RESULTS

Recent developments in environmental concern and issues of social risk have changed the nature of power systems expansion decisions. These decisions now involve several criteria and different centers of influence, necessitating multiobjective analysis.

The approach taken here is that of generating a few relevant decision alternatives from the set of efficient solutions for the medium term, which is assumed to be 10 to 15 years.

The modelling technique and the solution procedure employed in this study offers several advantages, especially when the characteristics of the decision environment are considered. The linear programming model is a well-known and established technique and most decision makers are aware of its capabilities. The method of generating efficient solutions and decision alternatives is quite efficient and calls for neither prior articulation of preferences nor an extended cooperation with the decision makers, which are quite difficult to realize in the particular decision environment. Uncertainties in certain parameters and policy implications are analyzed by developing different scenarios. The influence of hard-to-quantify aspects such as environmental impact and social risks are determined through sensitivity analysis.

A sound and reliable information base for decisions is produced without requiring very precise and sophisticated data.

The major limitation of this procedure is in model size. The whole procedure would become rather impractical if the number of decision variables is in the order of hundred. In such cases, not only would computational efforts increase, but also displaying the results and analysis of decision alternatives would become quite tedious.

PART III. MULTIOBJECTIVE INTEGER LINEAR PROGRAMMING

The multiobjective integer linear programming (MOILP) problem is written as

$$\begin{aligned} & \max_{x \in X} Cx \\ & X = \{x \mid Ax \leq b, \quad x \geq 0 \text{ and integer}\} \end{aligned}$$

where C is a $p \times n$ matrix whose rows C_i , $i=1, \dots, p$, represent the different objective functions; A is an $m \times n$ matrix; and x and b are n and m sized vectors respectively. Maximization here refers to the determination of the set of all efficient points. In multiobjective zero-one linear programming (MOZOLP) elements of X are further constrained to take on values of only zero or one.

MOILP problems arise when in problems characterized by multiple objectives, the decision variables are desired to take on integer values. For example, in the design of an urban transportation system, one could be interested in selecting the optimal transportation modes as well as determining the number of units of each to be scheduled for a desired service level, with the objectives of minimizing travel times, construction costs and operating costs. Binary variables are necessary for handling yes or no decisions in several problems; such an area of application for MOZOLP is project selection. In addition to the traditional objective of maximizing the total present value, other objectives such as minimizing risk or maximizing market share may be under consideration in project selection or capital budgeting problems.

It is natural to assume that MOLP solution techniques can be extended to handle MOILP problems since integer linear programming is closely related to linear programming and most algorithms for integer linear programming use linear programming subroutines. However, MOILP problems are of a quite different nature than MOLP problems. Parametrically optimizing linear combinations of the objectives generates some but not necessarily all of the efficient solutions. This can be observed from Figure III.1.

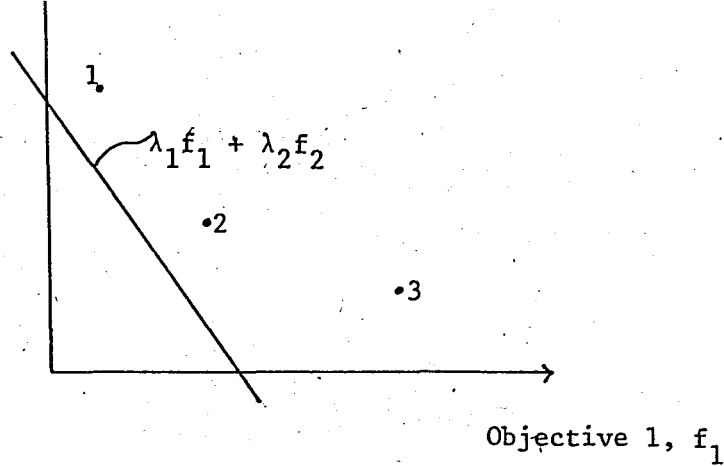


Figure III.1. Parametric Scalarization May Not Generate all Efficient Integer Solutions

Here, it is assumed that points 1, 2, and 3 give the efficient points in the objective space for a hypothetical two objective integer problem. By maximizing the linear function $\lambda_1 f_1 + \lambda_2 f_2$, $\lambda_1, \lambda_2 > 0$, points 1 and 3 can be obtained as the maximum solutions for certain appropriate values of λ_1 and λ_2 . However, point 2 does not give the maximum solution of $\lambda_1 f_1 + \lambda_2 f_2$ for any $\lambda_1, \lambda_2 > 0$. Thus, although it is efficient, point 2 cannot be generated by parametric scalarization, i.e. parametrically optimizing linear combinations of the two objectives. Furthermore, as the feasible region of integer points in an integer linear program does not generally represent a connected graph, the use of multiobjective simplex methods for identification of adjacent efficient points may not yield a convergent method for the search of all efficient points; in other words, the implied search may not locate efficient points in the unconnected portions of the integer linear program lattice. These facts indicate that methods of MOLP cannot be simply extended to MOILP. This is why MOILP has gained interest only recently and is still not a developed field of multiobjective optimization. In the following section a survey of multiobjective integer linear programming methods is presented.

III.1. MULTIOBJECTIVE INTEGER LINEAR PROGRAMMING METHODS

A classification of the approaches to MOILP is given in Table III.1.

Table III.1. Approaches to MOILP

Nature of Solutions	Problem Type	Approach
Efficient Solutions	General MOILP	Bowman [8] - A parametric method
	zero-one bicriterion	Pasternak and Passy[45] A parametric Algorithm
		Shapiro [53] Theoretical results
	MOZOLP	Bitran [7] Search algorithm based on preference cone
		Klein and Hamman[38] Sequential algorithm based on implicit enumeration
A Preferred Solution	General MOILP	Zionts [63] Interactive algorithm
		Lee [41] Integer goal programming

The first work to appear in this field is on bicriterion zero-one programming. Pasternak and Passy [45] show that all efficient points are solutions of the parametric problem

$$\begin{aligned}
 & \max \quad C_1 x \\
 & \text{s.t.} \quad Ax \leq b \\
 & \quad \quad C_2 x \geq \theta \\
 & \quad \quad x_j = 0 \text{ or } 1
 \end{aligned}$$

where θ varies from the minimum value of $C_2 x$, attained at x which maximizes $C_1 x$, to the maximum value of $C_2 x$. Some further results are derived under the assumption of a strictly quasiconcave utility function so that only a relevant subset of all efficient points would be generated. The authors also give an algorithm which is an extension of the Balas filter method.

Shapiro [53] investigates how the results of integer programming duality theory can be used for multiobjective zero-one and mixed integer problems. He focuses on identification of efficient solutions which can be generated through parametric scalarization, i.e. the solutions of the following parametric LP problem

$$\begin{aligned} \max \quad & \lambda^T Cx \\ \text{s.t.} \quad & x \in [F] \\ & \lambda \in \text{int } \Lambda \end{aligned}$$

where $F = \{x | Ax \leq b, x_j = 0 \text{ or } 1\}$ and $[F]$ denotes the convex hull of F and Λ is as defined for the MOLP problem.

Shapiro comments that the difficulty with this apparent reduction to a more manageable optimization problem is that $[F]$ is generally impossible to characterize in any practical manner. He then states that integer programming duality theory can be interpreted as approximating $[F]$ in a neighborhood of an optimal integer programming solution. This interpretation would provide the insights on how to perform the sensitivity analysis associated with parametric variation of λ .

Shapiro also mentions that integer programming duality theory could be combined with Benders' decomposition method and thus the suggested approach could be extended to multiobjective mixed integer programming problems.

The discussions of Shapiro are quite theoretical and the results presented are far from complete. The author states that a rigorous research on the structure of the family of integer programming dual problems generated as λ varies is needed.

Theoretically an integer programming dual problem which solves a given integer programming problem can always be constructed by a finite procedure, however in practice integer programming dual methods require excessive numerical calculations and are not currently being used much. It is to be expected that the computational effort associated with generating the family of integer programming dual problems as λ varies will be quite enormous. Therefore this approach does not seem computationally promising.

Bowman [8] considers the relationship between the generalized Tchebycheff norm and efficient solutions for problems where the feasible region is not convex. As this is the case in problems where some or all of the decision variables are constrained to take on integer values the results apply to such problems.

The motivation for the approach is the geometric interpretation of efficient points. A point x^* is efficient if in the p -dimensional objective space $Y = \{y | y = Cx, x \in X\}$, the nonnegative orthant with origin at $y^* = Cx^*$ contains no point $y \in Y$. Therefore, the smallest hypercube centered at any point $y^* + \alpha e$, where $\alpha > 0$ and e is a vector of appropriate dimensions with each component equal to one, containing a point $y \in Y$ is a hypercube that contains y^* on the boundary. This can be interpreted as y^* being a point of the objective space that minimizes the Tchebycheff norm derived from some point (in this case $y^* + \alpha e$), where the Tchebycheff norm of a p dimensional vector r is denoted by $\|r\|$ and is defined as

$$\|r\| = \max_i |r_i|$$

Obviously, $\|r\| = a$ defines the boundary points of the hypercube centered at the origin and with sides of length $2a$.

Similarly, the generalized Tchebycheff norm of a p -dimensional vector r is defined as

$$\|r\|_\beta = \max_i \beta_i |y_i|, \quad \beta > 0$$

where β is a p -dimensional vector.

Then, using the generalized Tchebycheff norm, Bowman considers the problem

$$\min_{x \in X} \|Cx - \bar{y}\|_\beta, \quad \bar{y}_i = \max_{x \in X} C_i x \quad (P_\beta)$$

or alternatively written in a familiar form

$$\begin{aligned} \min \quad & z \\ \text{s.t.} \quad & z \geq \beta_i (\bar{y}_i - C_i x) \\ & x \in X \end{aligned}$$

He then states that $x^* \in X$ is an efficient solution only if it is a solution to P_β for some $\beta > 0$. The converse is not generally true. A non-efficient point which is dominated only by efficient points lying on the boundary of the nonnegative orthant with origin at the non-efficient point may still be a solution to P_β . Nevertheless, Bowman shows that if uniform dominance is satisfied, i.e. if for every non-efficient point x^1 there exists an efficient point x^* such that $Cx^1 > Cx^*$ then all solutions to P_β are efficient.

This approach provides a means of generating all efficient points even though a lot of computational effort is needed; it is also difficult to vary the parameters β_i in an orderly way so that solutions representative of the entire set of efficient solutions can be obtained.

It is also noted that the formulation given by P_β can be considered as a parametric goal programming formulation, with the goal being the point \bar{y} , and β representing the weighting of each of the goals.

Integer goal programming is another approach to multiobjective integer problems, aiming at finding a single compromise solution. Sang Lee [41] incorporates three integer programming algorithms, a cutting-plane, a branch and bound and an implicit enumeration algorithm within the goal programming framework in a quite straightforward manner.

An interactive approach to multiobjective integer and mixed-integer problems is presented by Zionts [63]. The approach is an extension of the Zionts-Wallenius method discussed in section I.1.2., to incorporate integer variables. The basic assumption is that the implicit utility function of the decision maker is a linear function of the objectives. Under this assumption, it is sufficient to consider the subset of efficient solutions that can be generated through parametric scalarization. Because then, only these solutions are candidates for the decision maker's preferred solution.

Zionts considers adaptation of the Zionts-Wallenius method within a branch and bound framework. First, the corresponding continuous multiobjective problem is solved using the Zionts-Wallenius method. If the solution satisfies integer constraints, it is the preferred solution. Otherwise the branch and bound procedure is initiated by branching on an integer variable whose value is not integer in the current solution. Each of the newly generated solutions is tested and can be excluded from further consideration if i) the decision maker prefers an integer solution to it and ii) all efficient tradeoffs associated with the solution are viewed negatively or with indifference by the decision maker. Otherwise it is placed on the list of solutions to be examined further. If a new integer solution which is preferred to the best known integer solution is found, it becomes the best known integer solution and the previous solution can be dropped from the list if condition ii) holds.

Whenever the objective function weights λ_i do not satisfy the constraints constructed from decision maker's responses to tradeoff questions, a new set of λ_i satisfying these constraints are adopted and a new objective function is formed. Then, the optimal continuous solution for that objective function will have to be found first. Generally branching continues from the most preferred newly found solution, or the most preferred solution from the list. If the list is empty, the preferred solution has been found.

Zionts also considers a cutting plane approach, however he does not find this approach promising on the grounds that cutting plane methods do not work well in practice.

The branch and bound approach has not been tested and its convergence properties are not known. However it seems to be promising for finding a maximum utility solution.

The next two approaches to be discussed aim at generating all efficient solutions for MOZOLP problems and are algorithmically oriented. Since the algorithm presented in this thesis is also in the field of MOZOLP, these approaches are discussed in more detail to indicate the state of art in this area.

Theoretical results and an algorithm for generating all efficient points for MOZOLP problems are given by Bitran [7]. In this approach the relations between the main problem P

$$\begin{aligned} & \max_{x \in F} Cx \\ & F = \{x \mid Ax \leq b, \quad x_j = 0, 1\} \end{aligned} \quad (P)$$

and the auxiliary problem P'

$$\begin{aligned} & \max_{x \in F'} Cx \\ & F' = \{x \mid x_j = 0, 1\} \end{aligned} \quad (P')$$

where the constraints have been removed, are explored. P' plays a central role in the analysis, based on the fact that every efficient point of P' that is feasible in P is an efficient point of P. First, finding all efficient points of P' are considered.

A set of preference directions V is defined as

$$V = \{v^i \mid Cv^i \geq 0, \text{ and } v_j^i = 0, 1 \text{ or } -1\}$$

where if $x^2 = x^1 + v^i$ for some $v^i \in V$ then $Cx^2 \geq Cx^1$ and x^1 is dominated in P'. The set of points dominated in P' in a given direction v^i is denoted by $M(v^i)$ and defined as

$$M(v^i) = \{x \mid x_j = 0 \text{ if } v_j^i = 1; \quad x_j = 1 \text{ if } v_j^i = -1; \quad x_j = 0, 1 \text{ if } v_j^i = 0\}.$$

The sets $M(v^i)$ can be determined as the sets of extreme points that solve the linear programming problem: $\min\{v^i x, 0 \leq x_j \leq 1\}$, and $\bigcup_i M(v^i)$ gives the set of dominated points of P'.

The set V is to be obtained through an implicit enumeration scheme. However as the number of potential elements of V is of the order 3^n , some results aiming at reducing the enumeration are given

Lemma 1. Assume $v^1, v^2 \in V$ $v_j^1 = v_j^2$, $j \in J_0$ and $y_j^1 = 0$ $j \in J_0$, where J_0 is a subset of variable indices. Then $M(v^2) \subset M(v^1)$.

As a result, it is only necessary to consider v^1 in such a case. Thus enumeration will start with vectors having as many zero variables as possible.

The set of efficient points of P' , $EF(P')$ are obtained by eliminating the dominated points from F' . Although every $x \in EF(P') \cap F$ is efficient in P , a dominated point of P' is not necessarily dominated in P . In order to relate P and P' the following lemma is given.

Lemma 2. a) If $Av^i \leq 0$ for some i then any $x \in F \cap M(v^i)$ is a dominated point of P . b) If $x \in EF(P')$ but $x \notin EF(P)$, then $x + v^i \notin F$ for all i such that $x \in M(v^i)$.

An algorithm for generating all efficient points of P based on these results is given as follows.

1. Through an implicit enumeration scheme, generate the subset of V implied by lemma 1.
2. Obtain $\bigcup_i M(v^i)$ and then by eliminating these from F' obtain $EF(P')$
3. Obtain $EF(P') \cap F$ i.e. points efficient in both P and P' .
4. Obtain $\bigcup_{i \in I_1} M(v^i)$, $I_1 = \{i | Av^i \leq 0\}$ i.e. the set of dominated points of P' which if feasible in P will also be dominated in P .
5. Obtain $\Delta_i = M(v^i) + \{v^i\}$ $i \in I_2 = \{i | i \notin I_1\}$, i.e. the set of points of P' which dominate the points in $M(v^i)$ in the direction v^i .
6. Exclude the points found in 4. from $M(v^i)$. i.e. form

$$\Omega_i = M(v^i) - \bigcup_{j \in I_1} M(v^j) \quad i \in I_2$$

7. For each $i \in I_2$, eliminate all elements of Ω_i dominated by the points found in 3.

8. For each $i \in I_2$, eliminate elements of Ω_i infeasible in P .
9. For each $i \in I_2$, determine the set of points of Ω_i dominated by points in $\Delta_i \cap F$. Let these be denoted by γ_i . Form the sets $\theta_i = \Omega_i - \gamma_i$.
10. Form the set $\phi = \bigcup_{i \in I_2} \theta_i - \bigcup_{i \in I_2} \gamma_i$. This step is necessary because the sets used upto this point may not be disjoint.
11. Obtain $EF(P) = (EF(P') \cap F) \cup \phi$

Summarizing, first points which are efficient in both P' and P are obtained. Then, if a set of points dominated in P' which will also be dominated in P , if feasible, can be identified, these are eliminated. Dominated points of P' , dominated by any element of the subset of efficient points of P found initially are also discarded. Next, the remaining points are further reduced by first checking for feasibility in P and next by verifying if they are dominated in the direction v^i by a point in F . Points inefficient in P' but which are not eliminated in the process are also efficient in P and these are added to the previously found solutions to obtain all efficient points.

Computational results for a set of 30 problems are presented and compared with the results of directly applying the definition of efficiency to points of F . It is stated that results indicate the algorithm does better than directly applying the definition of efficiency when the number of points of F is large compared with 2^n which is the number of feasible points of P , i.e. number of feasible points without any constraints. However, all of the problems solved are quite small. The maximum problem size is nine variables four objectives and four constraints, and the computational results are not very encouraging. For example, a two objective function, four constraint, six variable problem requires about 3 seconds, whereas a similar problem with nine variables requires about 60 second for solution on Burroughs 6700 computer.

The mixed-integer multiobjective problem is also discussed briefly. The approach suggested is solving a series of scalarized problems using Benders' partitioning. Some conditions are given to obtain a local approximation to $\Lambda(x)$ where $\Lambda(x)$ is the set of objective weights λ for which the current solution x remains optimal.

A branch and bound algorithm for sequentially generating the complete set of efficient solutions for MOZOLP problems is proposed by Klein and Hannan [38]. The procedure consists of solving a sequence of continually more constrained single objective function problems.

The concept of applying logical operators such as "and" and "or" to linear inequalities is introduced. Let L_i , $i=1, \dots, k$ represent a set of k linear equalities and let \vee stand for "or" and \wedge stand for "and". The statement

$$\bigvee_{i=1}^k L_i = L_1 \vee L_2 \vee \dots \vee L_k$$

represents the condition that at least one of the inequalities L_i must hold. Similarly the statement

$$\bigwedge_{i=1}^k L_i = L_1 \wedge L_2 \wedge \dots \wedge L_k$$

represents the condition that all the inequalities L_i must hold simultaneously.

The general step k in the method consists of solving the problem P_k

$$\begin{aligned} \max \quad & C_1 x \\ \text{s.t.} \quad & Ax \leq b \\ & \bigwedge_{j=1}^r \left(\bigvee_{i=2}^p C_i x \geq C_i y^{j+1} \right) \quad (P_k) \\ & y^j \in Y^{k-1} = \{y^1, y^2, \dots, y^r\} \\ & x_j = 0, 1 \end{aligned}$$

where Y^{k-1} is the set of efficient points accumulated until step k . Let

\bar{Y}^k denote the set of efficient optimal solutions to P_k . If P_k has a unique solution, it is the unique efficient solution. If P_k has multiple solutions the dominated solutions are not included in \bar{Y}^k . The cumulative set of efficient points Y^k then becomes $Y^k = Y^{k-1} \cup \bar{Y}^k$.

The concepts developed by Roodman [51] for postoptimizing zero-one programs are utilized in the general solution procedure. When several related integer problems are to be solved through implicit enumeration, computational savings can be achieved if these problems are solved using the same enumeration tree. The tree developed during step zero when maximizing the first objective, is used for subsequent problems. During solution of any problem P_k , information on the causes of fathoming of nodes is stored. Subsequent problems use this information to determine which nodes remain fathomed for these problems.

Along the enumeration tree, nodes are fathomed either due to infeasibility, feasibility or "insufficiency", where insufficiency means that the upper bound for objective one is less than the current lower bound for it. Nodes fathomed for infeasibility remain infeasible for all following problems and need not be recorded. Nodes fathomed for feasibility are recorded on a list of feasible solutions FL, in decreasing values of the first objective function. A node fathomed for insufficiency cannot yield a solution to the current problem, but may give a solution to subsequent problems. These nodes are recorded on a list of potential solutions PL.

After these preliminaries an outline of the algorithm is given as follows.

1. Initialization: Solve the problem P_0

$$\begin{aligned} \max \quad & C_1 x \\ \text{s.t.} \quad & Ax \leq b \\ & x_j = 0, 1 \end{aligned} \quad (P_0)$$

and determine its set of optimal solutions Y^0 . During solution create lists FL and PL. Set $k=1$ and go to step 2.

2. Finding a feasible solution to P_k : Form the problem P_k . Use some heuristic to find a feasible solution \bar{x} to P_k . One possibility is to search through FL. If no feasible solution go to step 5, otherwise go to step 3.
3. Forming the candidate list: Form the candidate list, i.e., the list of unfathomed nodes for P_k , which is composed of all nodes from FL and PL whose upper bounds on the first objective are greater than $C_1\bar{x}$. Remove the nodes on the candidate list from FL and PL. Go to step 4.
4. Solution of P_k . Solve P_k , limiting the enumeration to nodes on the candidate list, and determine the set of efficient optimal solutions \bar{Y}_k . Set $Y_k = Y_{k-1} \cup \bar{Y}_k$. During solution augment lists FL and PL. Set $k=k+1$ and go to step 2.
5. Termination. Stop. All efficient solutions have been obtained.

The authors also make some comments regarding the finiteness and exhaustiveness of the described procedure and the nature of the logical constraints. The logical constraints added by each efficient point differ only in their right hand side coefficients. This allows treating them as $P-1$ constraints with multiple right hand sides. They also show that in the case of the bicriterion problem the logical constraints collapse to a single linear inequality and the problem P_k becomes

$$\begin{aligned}
 \max \quad & C_1 x \\
 \text{s.t.} \quad & Ax \leq b \\
 & C_2 x \geq C_2 y^r + 1
 \end{aligned}$$

where y^r is an efficient point determined at step $k-1$. Thus, it is observed that the approach of Pasternak and Passy [45] is a special case of this approach.

Some computational results are reported and it is observed that the computational effort and the number of efficient points increases as the number of variables and objectives increase and as the problem becomes

less restrictive, i.e. the number of feasible solutions increase. For example a two objective, fourteen variable four constraint problem with the right hand sides being set to 50 % of the sum of the constraint coefficients requires about 13 seconds of cpu time whereas a similar problem with four objectives and ten variables requires about 48 seconds. Actually, the computation times are not really amenable to comparison with a compiled code, as the program implementing the procedure was written in APL, which is interactive, and run on a UNIVAC 1100/89.

However, the authors comment that the computational results indicate that large amounts of computation time are required and the number of efficient points increases a lot as the number of variables and objectives increases. Observing that a great number of efficient points make it harder for the decision maker to select a preferred solution, they consider a procedure which limits the search for efficient points. The problem P_k is modified to

$$\begin{aligned} \max \quad & C_1 x \\ \text{s.t.} \quad & Ax \leq b \\ & \bigwedge_{j=1}^r \left(\bigvee_{i=2}^p C_i x \geq C_i y^j + d_{ij} \right) \quad (P_k) \\ & x_j = 0,1 \end{aligned}$$

where $d_{ij} > 1$ and integer. Setting values for d_{ij} different from one limits the search for efficient points to points sufficiently different, with respect to objectives 2 to p, from those already found.

This limited search approach and the original approach are both biased in terms of objective one which determines the order in which efficient points are discovered. Furthermore, there is bound to be a lot of backtracking and searching for feasible solutions as one moves from one problem to the next, which require considerable amounts of computation. Also storage requirements are quite high since a lot of data needs to be stored to keep track of the nodes in the lists FL and PL.

The paper of Klein and Hannan was not available at the initiation of this

thesis, when extension of implicit enumeration techniques to MOZOLP problems was considered. Independently, an implicit enumeration algorithm for MOZOLP problems, which is based on a different interpretation was developed. This algorithm, along with the studies carried out to increase its efficiency is presented in the next section.

III.2. AN ALGORITHM FOR MULTIOBJECTIVE ZERO-ONE LINEAR PROGRAMMING

The algorithm developed for MOZOLP problems is based on implicit enumeration. Therefore, in order to provide a frame of reference, a review of implicit enumeration techniques is presented.

III.2.1. REVIEW OF IMPLICIT ENUMERATION

Implicit enumeration refers to a class of branch and bound algorithms designed specifically for problems with binary decision variables. Although several reformulations and refinements have been done by various authors, the original version of the implicit enumeration algorithm is due to Balas [3]. The algorithm examines the nodes of the combinatorial tree by the application of certain tests to determine whether feasible and improved solutions can be found by moving further down from them or not.

In the implicit enumeration tree each node represents a "partial solution" which is an assignment of binary values to a subset of the decision variables. Variables which are not assigned values are called "free variables". A solution formed through an assignment of binary values to all free variables is called a "completion" of the partial solution. A node is "fathomed" when none of its completions require further investigation.

In Balas' original description practically all tentative solutions or nodes must be stored so that they may be scanned during succeeding iterations. Glover [30] applied the "backtracking" concept to implicit enumeration. Geoffrion [26] reformulated the Balas algorithm by representing the tree in vector form, which greatly improved the bookkeeping and computational

efficiency of the algorithm. In this formulation, the path P_k from the initial node of the tree to node k is stored as a vector and uniquely determines the remaining enumeration required. Variable indices appear in the vector if they correspond to assigned variables. The order of the indices represents the level in the tree. A positive subscript indicates that the variable has been assigned the value one, and a negative subscript indicates that the variable has been assigned the value zero. When branching to $x_p=1$, P_k is simply augmented by p . In backtracking, the rightmost positive element is changed to a negative element and all negative elements to the right of it are dropped. The enumeration is complete when all remaining elements are negative.

Before giving the rules for fathoming and branching, some definitions are needed. At any node k , denoted by N_k , let w_k be the index set of assigned variables, let B_k and H_k denote the index sets of variables assigned the values one and zero respectively, and let F_k be the index set of free variables. Then, the problem considered at N_k is

$$\begin{aligned} \max \quad & \sum_{j \in F_k} c_j x_j + \sum_{j \in B_k} c_j \\ \text{s.t.} \quad & \sum_{j \in F_k} A^j x_j \leq b - \sum_{j \in B_k} A^j \equiv S \\ & x_j = 0, 1 \quad j \in F_k \end{aligned}$$

Without loss of generality, it can be assumed that $c_j \leq 0$, since any x_j with $c_j > 0$ can be replaced by $x'_j = 1 - x_j$, yielding a problem with nonpositive c_j . Then, the upper bound z^{-k} at N_k is given by $z^{-k} = \sum_{j \in B_k} c_j$ and z_0 denotes the current lower bound, i.e. the best objective function value so far computed. The sum of negative coefficients of free variables in constraint i is given by t_i , i.e. $t_i = \sum_{j \in F_k} \min\{0, a_{ij}\}$. I_k , the infeasibility of N_k is defined as

$$I_k = \sum_{i=1}^m \max\{0, -s_i\}$$

and $I_k(j)$ the infeasibility at the successor node if x_j is assigned the value one as

$$I_k(j) = \sum_{i=1}^m \max \{0, -s_i + a_{ij}\}$$

Then, the fathoming conditions are

- i) Feasibility, i.e. $S \geq 0$
- ii) Infeasibility, i.e. $t_i > s_i$ for some $i=1, \dots, m$
- iii) Suboptimality, i.e. $z^k \leq z_0$

If a node is fathomed, backtracking is done to the most recently generated node. Otherwise, the next variable to be assigned the value of one, x_p , is chosen such that $I_k(p) = \min_{j \in F_k} I_k(j)$.

Many tests for reducing the enumeration have been proposed. Here some of these tests are reviewed to provide the background for the tests utilized in the MOZOLP algorithm.

TEST 1. If $c_j \leq z_0 - z^k$ then $x_j = 0$ in any optimal completion of N_k . This test is due to Balas [3]. Tests 2-4 below, are due to Glover [30] and are performed by first computing numbers G and U which give the minimum and maximum number of variables that are to equal one in any optimal completion of N_k . The set of constraints is augmented by

$$\sum_{j \in F_k} a_{oj} x_j \leq s_0$$

where $a_{oj} = -c_j$ and $s_0 = z_0 - z^k$. Then for each constraint $i=0, \dots, m$, the variables are reindexed so that the coefficients a_{ij} are in nondecreasing order. The reindexed coefficients for constraint i are denoted by $a_{i,t(i)}$ where $t < r$ implies $a_{i,t(i)} \leq a_{i,r(i)}$. The sum of r smallest coefficients of constraint i , $T_i(r)$, is defined as

$$T_i(r) = \sum_{j=1}^r a_{i,j(i)} \quad , \quad T_i(0) = 0.$$

For each i such that $s_i < 0$, G_i is defined by

$$T_i(G_i) \leq s_i < T_i(G_i - 1)$$

Then $G_i \leq G$ and G is defined to be $G = \max_{i, s_i < 0} G_i$

For each $i=0, \dots, m$, U_i is given by

$$T_i(U_i) \leq s_i < T_i(U_i + 1) \quad \text{if } T_i(g) > s_i$$

$$U_i = \infty \quad \text{if } T_i(g) \leq s_i$$

where g is the number of elements in F_k . Then $U_i \geq U$ and U is defined to be $U = \min_i U_i$.

TEST 2. If $G > 0$, N_k is fathomed since $G < \sum_{j \in F_k} x_j \leq U$ cannot be satisfied. Also if $G = g$, then each of the free variables can be set to one.

TEST 3. Let $T_i(G, U) = \min_{G \leq t \leq U} T_i(t)$ $i=0, \dots, m$

be the smallest sum that can be obtained using at least G but not more than U free variables. Assume $G \leq U$ and a_{ip} is not one of the numbers which determines $T_i(G, U)$. If $T_i(G-1, U-1) > s_i - a_{ip}$ then $x_p = 0$ in any optimal completion of N_k .

TEST 4. Assume $G \leq U$ and a_{ip} is one of the numbers that determines $T_i(G, U)$. If $T_i(L+1, U+1) > s_i + a_{ip}$ then $x_p = 1$ in any optimal completion of N_k .

Some other tests can be found in Geoffrion [26], Fleischmann [20], Glover [30], Glover and Zionts [31] and Petersen [47]. These tests can reduce the enumeration at the expense of added calculation, and so long as the extra calculations do not offset the benefits of reduced enumeration, it is worthwhile to use them.

Since there are at most 2^n possible combinations of decision variables, the finiteness of the algorithm is guaranteed. However computational

efficiency depends strongly on the effectiveness of the applied tests in curtailing the enumeration.

It is accepted that implicit enumeration is a powerful technique for single objective zero-one problems. The possibilities of extending the technique to multiobjective problems are investigated in this study. It is observed that through a vector interpretation and proper definition of upper and lower bounds, it is possible to extend implicit enumeration to multiobjective problems..

III.2.2. DEVELOPMENT OF A MULTIOBJECTIVE IMPLICIT ENUMERATION ALGORITHM

Reviewing briefly, the multiobjective zero-one linear programming problem is written as follows

$$\begin{aligned} \max \quad & Cx \\ \text{s.t.} \quad & Ax \leq b \\ & x_j = 0 \text{ or } 1 \end{aligned}$$

where the rows of the $k \times n$ matrix C represent the different objective functions, and A is an $m \times n$ constraint coefficient matrix. It is no longer possible to keep all $C_{ij} \leq 0$, since a variable x_j may have a negative coefficient in one objective and a positive one in another.

The problem considered at any node N_k of the enumeration tree is:

$$\begin{aligned} \max \quad & \sum_{j \in F_k} C^j x_j + \sum_{j \in B_k} C^j \\ \text{s.t.} \quad & \sum_{j \in F_k} A^j x_j \leq S \end{aligned}$$

where C^j and A^j represent the j th columns of C and A respectively. Now, one can define a slightly different upper bound vector \bar{z}^k as:

$$\bar{z}^k = \sum_{j \in B_k} C^j + Y^k ; \quad Y_i^k = \sum_{j \in F_k} \max \{0, C_{ij}\}$$

where Y_i^k is the i th element of the column vector Y^k .

The lower bound vector \underline{Z}_k for a feasible node, N_k , is defined as

$$\underline{Z}^k = \sum_{j \in B_k} C^j$$

Another difference from the single objective case is that instead of a single lower bound, now there will be a set of lower bound vectors. These vectors give the values of the objectives at feasible solutions which are so far undominated and which are the candidates for efficient solutions.

Branching and backtracking can be done as before, however the fathoming conditions need to be reconsidered. Since the C_{ij} 's are not constrained to be nonpositive, continuing down the tree from a feasible node N_k one may reach another feasible node N_j with a lower bound vector $\underline{Z}^j \leq \underline{Z}^k$ or even $\underline{Z}^j \geq \underline{Z}^k$. This means feasibility of a node is not sufficient for fathoming it. However, the general rule of branch and bound for fathoming a node whose upper bound equals its lower bound still applies. The fathoming conditions are then

i) Bound equality, i.e. $\underline{Z}^k = \underline{Z}_k$

ii) Infeasibility, i.e. $t_i > s_i$ for some $i=1, \dots, m$

where $t_i = \sum_{j \in F_k} \min \{0, a_{ij}\}$

iii) Domination, i.e. $\underline{Z}^k \leq LB^j$ for some $j \in L$ where $L = \{0, 1, \dots, \ell\}$ with ℓ being the current number of lower bounds and LB^j is the j th lower bound.

When a new feasible solution is found, its lower bound vector should be compared against existing lower bound vectors LB^j , $j \in L$. If the new lower bound vector is not dominated by any of the existing lower bound vectors, it is stored as a new lower bound. And any existing lower bound vectors dominated by it are discarded. When no live nodes remain, the enumeration is complete and the current lower bound vectors give the efficient solutions of the MOZOLP problem.

A rudimentary flow diagram for a multiobjective implicit enumeration algorithm is given in Figure III.2. Here the convention of branching according to minimum infeasibility criterion has been adopted. However, branching from a feasible node is a different matter. Let the set F_k be partitioned into two sets, $F_k^1 = \{j \in F_k \mid C^j \leq 0\}$ and $F_k^2 = \{j \in F_k \mid C^j \not\leq 0\}$. Then a feasible node is considered for further branching if $F_k^2 \neq \emptyset$. On the other hand, if $F_k^2 = \emptyset$, then $Y^k = 0$ and $\bar{Z}^k = Z_k$ and N_k is fathomed. If N_k is not fathomed, any variable x_j , $j \in F_k^2$ can be the branching variable, whereas a variable x_r , $r \in F_k^1$ should not be considered because branching on it leads to a dominated solution.

The steps of the algorithm may be summarized as follows

STEP 1 : Initialization. At N_0 , $F_0 = \{1, \dots, n\}$ $\bar{Z}^0 = \infty$ $\ell = 0$. Go to Step 2.

STEP 2 : Calculating bounds and feasibility check. Let $\bar{Z}^k = \sum_{j \in B_k} C^j + Y^k$. If $S \geq 0$ let $\underline{Z}^k = \sum_{j \in B_k} C^j$. Go to Step 5. If $t_i > s_i$ for any i , go to Step 6. Otherwise go to Step 3.

STEP 3 : Bounding. If $\bar{Z}^k \leq LB^j$ for some $j \in L$ go to Step 6. Otherwise go to Step 4.

STEP 4 : Branching. Branch to $x_p = 1$, where $I_k(p) = \min_{j \in F_k} I_k(j)$. Go to Step 2.

STEP 5 : Processing Feasible Node. If $\underline{Z}^k \leq LB^j$ for any $j \in L$ go to Step 6. If $LB^j \leq \underline{Z}^k$ for any $j \in L$ set $\ell = \ell - 1$ and drop LB^j . Then set $\ell = \ell + 1$ and $LB^\ell = \underline{Z}^k$. If $F_k^2 = \emptyset$ go to Step 6. Otherwise branch to $x_p = 1$, $p \in F_k^2$. Go to Step 2.

STEP 6 : Fathoming and Backtracking. Fathom N_k . If no live node exists go to Step 7. Otherwise backtrack to the newest live vertex. Go to Step 2.

STEP 7 : Termination. If $\ell = 0$, there is no feasible solution. If $\ell \geq 1$ LB^j , $j = 1, \dots, \ell$ are the desired efficient solutions.

As mentioned before, the efficiency of any enumerative algorithm depends

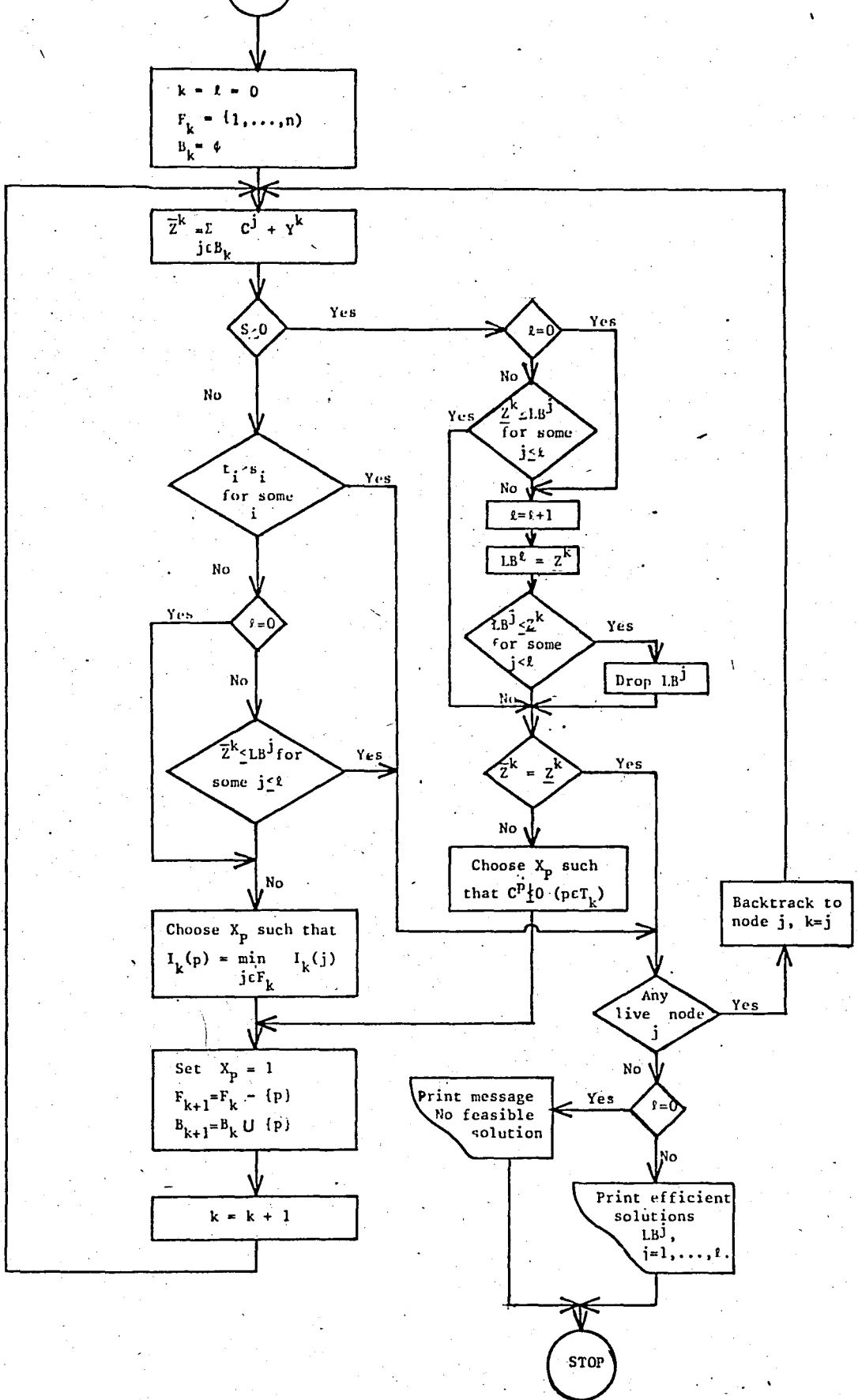


Figure III.2. Rudimentary Multiobjective Implicit Enumeration Algorithm

on the ability of its tests to chop off as great a portion of the enumeration tree as possible. The rudimentary algorithm sketched above cannot be expected to be very efficient. Of course, a lot of computational burden is inherent in most multiobjective techniques and they cannot compete with the computational efficiency of single objective methods. Here, all tests of implicit enumeration developed for concluding infeasibility can be used to advantage, but the fathoming tests involving the use of upper and lower bounds are vector comparisons and it is obvious that fewer nodes will be fathomed by them. Therefore it was decided to concentrate on developing stronger domination tests and also to test the effects of different branching criteria in order to improve the efficiency of the algorithm.

III.2.2.1. Domination Tests and Alternative Branching Criteria

Some domination tests and branching criteria which depend on the concept of "domination margin" were developed and tested. At N_k , the domination margin $D_k(j,s)$ of a free variable x_j is defined as

$$D_k(j,s) = \min_{r \in L} \left\{ \sum_{i=1}^P \max \{0, UB_i^{j,s} - LB_i^r\} + \delta_{jr} \right\}$$

$$s=0,1 \quad \delta_{jr}=1 \quad \text{if } UB_i^{j,s} = LB_i^r, \quad 0 \quad \text{otherwise.}$$

Where $UB_i^{j,1} = \bar{Z}_i^k + \min\{0, C_{ij}\}$, i.e. it is the upper bound at the successor node if x_j is assigned the value one. Similarly $UB_i^{j,0} = \bar{Z}_i^k - \max\{0, C_{ij}\}$ is the upper bound at the successor node where x_j is assigned the value zero. The domination margin is a measure of closeness to being dominated of the corresponding partial solution. Thus, we have:

TEST 1. If $D_k(j,1)=0$ for some $j \in F_k$, then $x_j=0$ necessarily in any efficient completion of N_k . Similarly, if $D_k(j,0)=0$ for some $j \in F_k$, then $x_j=1$ necessarily. If $D_k(j,1)=0 \quad \forall j \in F_k$, then N_k can be fathomed.

Proof: If $D_k(j,1)=0$ ($D_k(j,0)=0$), then for some $r \in L$ $UB_i^{j,1}$ ($UB_i^{j,0}$) $\leq LB_i^r$ and all feasible completions of N_k with $x_j=1$ ($x_j=0$) will be dominated. Also if $D_k(j,1)=0 \quad \forall j \in F_k$ then $x_j=0 \quad \forall j \in F_k$ and N_k is fathomed.

This test is an extension of Test 1, due to Balas, stated in section III.2.1. It can be viewed as a means of performing the domination check before branching to a specific node. The fathoming process can be further accelerated if a way of concluding domination not only before branching to a dominated node, but higher up the tree can be found.

Similar to that done in section III.2.1. before stating tests 2-4, let G denote the minimum number of variables that must be set to one in any feasible completion of N_k . Let $H_i(r)$ be defined as

$$H_i(r) = \sum_{j=1}^r \min\{0, C_{i,j}(i)\} \quad , \quad i=1, \dots, p$$

where the cost coefficients have been reindexed in nonincreasing order and $C_{i,j}(i)$ denotes the j th biggest coefficient, among the free variables, of objective i . Then,

TEST II N_k can be fathomed if $R = \bar{Z}_k + H(G) \leq LB^T$ for any $r \in L$, where $H(G)$ denotes the vector $[H_1(G), H_2(G), \dots, H_p(G)]^T$.

Proof. Let N_q be any feasible completion of N_k , obtained by setting at least G variables to one. The upper bound at N_{k+1} , where any variable x_j is set to one is given by

$$\bar{Z}_i^{k+1} = \bar{Z}_i^k + \{\min 0, C_{ij}\} \quad i=1, \dots, p$$

If at least G variables have to be set to one, then obviously

$$\bar{Z}^q \leq \bar{Z}^k + H(G)$$

which means $\bar{Z}^q \leq LB^T$ and N_q is dominated. Since any feasible completion of N_k is dominated N_k can be fathomed.

$R = \bar{Z}^k + H(G)$ can be termed an "advanced" upper bound which gives the best value that may be attainable for each objective by moving further down from N_k . This test is an extension of Test 2, given in section III.2.1. Here

G is computed explicitly with respect to the constraints and U is considered implicitly and only with respect to the objectives. Much of the efficiency of this test derives from the implicit consideration of U. Considering explicit calculation of U, let U_{ij} denote the maximum number of variables that can be set to one before the i th component of the advanced upper bound vector becomes less than the i th component of LB^j , i.e. U_{ij} is given by

$$H_i(U_{ij+1}) \leq LB_i^j - \bar{Z}_i^k < H_i(U_{ij}) \quad \text{if} \quad H_i(g) \leq LB_i^j - \bar{Z}_i^k$$

$$U_{ij} = \infty \quad \text{otherwise}$$

where g is the number of free variables.

Then U_j , the maximum number of variables that can be set to one if domination with respect to LB^j is avoided, is given by

$$U_j = \max_i U_{ij}$$

Since if $H_i(U_j) > LB_i^j - \bar{Z}_i^k$ for any one i domination is avoided. And U is given by

$$U = \min_j U_j = \min_j \max_i U_{ij}$$

Since the partial solution is not to be dominated with respect to any LB^j .

The calculations of the U_{ij} 's necessitates sequential formation of the partial sums $H_i(P)$, $P=1, \dots, U_{ij}+1$ and comparisons with $LB_i^j - \bar{Z}_i^k$, $j=1, \dots, \ell$ for each partial sum. However, by only implicitly considering U , these comparisons are made only once, which obviously involves much less computational effort.

Similar to defining an advanced upper bound, one can define an advanced domination margin $D_k^G(j,s)$ as

$$D_k^G(j,s) = \min_{r \in L} \{ \sum_{i=1}^p \max\{0, R_i^{j,s} - LB_i^r\} + \delta_{jr} \}$$

$$s=0,1 \quad \delta_{jr} = 1 \text{ if } R_i^{j,s} = LB_i^r, \quad 0 \text{ otherwise}$$

where $R_i^{j,1} = \bar{Z}_i^k + \min \{ [H_i(G-1) + C_{ij}], H_i(G) \}$ and

$R_i^{j,0} = \bar{Z}_i^k + \min \{ [H_i(G+1) - C_{ij}], H_i(G) \}$ are the advanced

upper bounds at the successor nodes where x_j is assigned the value one and zero respectively. Then we have

TEST III. If $D_k^G(j,1)=0$ then $x_j=0$ necessarily in any efficient completion of N_k .

Proof. If $D_k^G(j,1)=0$, then for some $r \in L$ $R_i^{j,1} \leq LB_i^r$. Let N_q be any feasible completion of N_k with $x_j=1$.

Consider objective i , $i=1, \dots, p$.

If C_{ij} is one of the numbers which determine $H_i(G)$

$$\text{Then } \bar{Z}_i^q \leq \bar{Z}_i^k + H_i(G) \quad \text{and} \quad H_i(G) \leq H_i(G-1) + C_{ij}$$

If C_{ij} is not one of the numbers which determine $H_i(G)$

$$\text{Then } \bar{Z}_i^q \leq \bar{Z}_i^k + \min\{0, C_{ij}\} + H_i(G-1)$$

$$i) \text{ if } C_{ij} \leq 0 \quad \bar{Z}_i^q \leq \bar{Z}_i^k + C_{ij} + H_i(G-1)$$

$$\text{and} \quad H_i(G-1) + C_{ij} \leq H_i(G)$$

ii) if $C_{ij} \geq 0$ then $\bar{Z}_i^q \leq \bar{Z}_i^k + H_i(G-1)$

But $C_{ij} \geq 0$ and C_{ij} does not determine $H_i(G)$ implies

$H_i(G)=0$ and $H_i(G-1)=0$. Therefore $\bar{Z}_i^q \leq \bar{Z}_i^k = \bar{Z}_i^k + H_i(G)$ and

$$H_i(G) \leq H_i(G-1) + C_{ij}.$$

Thus $\bar{Z}_i^q \leq \bar{Z}_i^k + \min \{ [H_i(G-1) + C_{ij}], H_i(G) \}$ which implies $\bar{Z}_i^q \leq R^{j,1} \leq LB^r$ and N_q is dominated. Then $x_j=0$ necessarily in any efficient completion of N_k .

An alternative implementation of Test III is to use the two-step domination margin $D_k^2(j,1)$. If $I_k(j) \neq 0$, then setting x_j to one does not lead to a feasible solution and at least one other variable must be assigned the value one which means $G \geq 2$. Therefore we have

TEST IV. If $D_k^2(j,1)=0$ and $I_k(j) \neq 0$, then $x_j=0$ necessarily in any efficient completion of N_k .

Similar to Test III, Test IV determines whether a variable has to be assigned a value of one in all efficient completions of N_k .

TEST V. If $D_k^G(j,0) = 0$ then $x_j=1$ necessarily in any efficient completion of N_k .

Proof. If $D_k^G(j,0)=0$, then for some $r \in L R^{j,0} \leq LB^r$. Let N_q be any feasible completion of N_k with $x_j=0$. Consider objective i , $i=1, \dots, p$:

If C_{ij} is one of the numbers which determine $H_i(G)$

$$\text{Then } \bar{Z}_i^q \leq \bar{Z}_i^k - \max\{0, C_{ij}\} + H_i(G+1) - \min\{0, C_{ij}\}$$

$$\bar{Z}_i^q \leq \bar{Z}_i^k + H_i(G+1) - C_{ij} \quad \text{and} \quad H_i(G+1) - C_{ij} \leq H_i(G)$$

If C_{ij} is not one of the numbers which determine $H_i(G)$

Then $\bar{z}_i^q \leq \bar{z}_i^k - \max\{0, C_{ij}\} + H_i(G)$

i) if $C_{ij} \leq 0$ then $\bar{z}_i^q \leq \bar{z}_i^k + H_i(G)$

$$\text{and } H_i(G) \leq H_i(G+1) - C_{ij}$$

ii) if $C_{ij} > 0$ then $\bar{z}_i^q \leq \bar{z}_i^k - C_{ij} + H_i(G)$

But $C_{ij} > 0$ and C_{ij} does not determine $H_i(G)$ implies

$H_i(G) = 0$ and also $H_i(G+1) = 0$. Therefore

$$\bar{z}_i^q \leq \bar{z}_i^k - C_{ij} = \bar{z}_i^k + H_i(G+1) - C_{ij}$$

$$\text{and } H_i(G+1) - C_{ij} \leq H_i(G)$$

Thus $\bar{z}_i^q \leq \bar{z}_i^k + \min\{[H_i(G+1) - C_{ij}], H_i(G)\}$ which implies $\bar{z}_i^q \leq R^j, 0 \leq LB^r$

and N_q is dominated. Then $x_j = 1$ necessarily in any efficient completion of N_k .

Tests III and V are extensions of Tests 3 and 4 due to Glover. Tests I-V can be used to check for domination either at every node or periodically, or at certain selected nodes. The break-even point between computational savings due to a smaller tree and increased computations to obtain the smaller tree should determine their frequency of usage.

For multiobjective problems a branching criterion other than the minimum infeasibility criterion could possibly be more effective. Since one is interested in enumerating efficient solutions, one can consider branching criteria based on a domination measure. Also if some of the tests given above are to be carried out at each node in an attempt to screen out variables leading to dominated solutions, the domination margin of each variable will be at hand and if could be used to advantage. Consequently, two different criteria which use the domination margin as a basis for

selecting the branching variable were formulated and compared with the minimum infeasibility criterion. These are

- i) Maximum domination margin criterion
- ii) Minimum domination margin criterion.

The effects of these branching criteria and of the proposed tests on the enumeration tree and the computation time were observed on a sample of eight problem types with different numbers of objectives, variables and constraints.

III.2.2.2. Experimentation on Branching Criteria and Use of Tests

First, the simple domination margin $D_k(j,1)$ of Test I was used in comparing the maximum and minimum domination margin criteria. It was observed that the use of minimum domination margin criterion led to the formation of a smaller sized enumeration tree. This seems surprising since the general practice in branch and bound techniques is branching to nodes which have higher upper bounds associated with them, hoping that many of the nodes with small upper bounds will never have to be considered if good lower bounds are found on the nodes that are branched to. [24, p.121]. It seems that this idea does not extend simply when one is dealing with vector bounds. Since the bound comparisons are vector comparisons, tests for fathoming by bounds are weaker, and a lot more of the nodes which were not branched to will have to be examined later. Thus, although by branching on a variable with minimum domination margin one is moving along a path which is more likely to lead to dominated solutions, one also has the advantage of recognizing and fathoming such partial solutions earlier. On the other hand, by branching on a variable with maximum domination margin, usually a longer path results and more time is spent for fathoming the partial solution even though the possibility of finally reaching a nondominated solution is greater.

Alternatively, the minimum two step domination margin $D_k^2(j,1)$ was considered for selecting the branching variable. When this change was made, it was seen that the total number of iterations, or nodes considered, decreased. This result also implied that Test IV, which uses the two-step

domination margin is more efficient than Test I which uses a one-step domination margin.

Next, the minimum domination margin (MDM) criterion using $D_k^2(j,1)$ was compared against the minimum infeasibility (MI) criterion. It was observed that although the number of iterations was generally smaller for the first case, the MI criterion did better with respect to computation time. This shows that although Test IV and a branching criterion based on it are effective in reducing the enumeration, the extra computational effort associated is quite high. Total number of iterations and cpu time for both cases are given in Table III.2. All computation times reported in this study are on UNIVAC 1106.

Table III.2. Comparison of MDM and MI Criteria for Branching

Problem	Size			No.of. Effcnt. Points	MDM Criterion		MI Criterion	
	p	n	m		Iterations	cpu (msec)	Iterations	cpu(msec)
1	2	7	3	5	20	77	22	59
2	2	9	6	5	87	427	134	365
3	2	9	9	5	87	483	132	419
4	2	10	3	4	131	454	281	520
5	2	15	10	5	204	1622	361	1676
6	2	20	10	8	1502	15128	2581	13496
7	3	9	6	6	93	474	125	342
8	3	10	3	6	148	760	314	618

Considering that reduced enumeration, when combined with more powerful domination tests could lead to reduced computational time, it was decided to introduce tests II, III, and V into the algorithm and observe the performances of minimum domination margin and minimum infeasibility criteria in combination with these tests. These tests require a lot of computations and their use at every node would be inefficient. After some experimentation, it was decided to use these tests only at nodes where at least one sixth of the variables have been set to zero and the number of variables set to

zero is greater than or equal to the number of variables set to one. The rationale behind this is that if the number of variables set to zero is greater than the number of variables set to one and also greater than a certain percentage of the total variables, the resulting partial problem will be a more constrained one and a greater number of variables will be required to take the value one, i.e. G will be higher. This in turn will result in tighter upper bounds and smaller domination margins, thus making the tests more effective. Also by requiring a percentage of the variables to be assigned to zero, using these tests too high up the tree, where they are bound to be inconclusive is prevented. Whenever $D_k^G(j,1)$ was available, it was used for selecting the branching variable.

It was also investigated whether the advantages of the two criteria, that is reduced enumeration and smaller cpu time could be combined beneficially by using Test IV only at certain selected nodes where it is expected to be more effective and using MDM criterion for branching at these nodes, but using MI criterion otherwise. In this implementation Test IV and MDM criterion were used only after a new lower bound or a dominated node were found and branching to an adjacent node (i.e. a node differing from the current node with respect to the assignment of a single variable) was to take place. Then the following partial solutions formed will be more likely to be dominated, and Test IV and MDM criterion can be used advantageously to screen out dominated solutions. When the minimum domination margin exceeded $-2pC_{av}$, where C_{av} is the arithmetic mean of the elements of C , MI criterion was resumed. When the minimum domination margin exceeds $-2pC_{av}$, on the average more than two consecutive branches of the enumeration tree have to be travelled before Test IV becomes effective and some of the free variables can be set to zero, or domination concluded. The results of these experimentations are given in Table III.3. again for the same sample of problems.

It was observed that the combined use of Tests II, III and V led to decreases in cpu time and the number of iterations for both the MDM and MI criteria. However, MI criterion still did generally better with respect to cpu time, especially for problems where the number of variables is high. The mixed use of the criteria did not seem to be advantageous, implying the use of Test IV even at selected nodes does not increase computational efficiency.

Table III.3. Comparison of MDM, MI and Mixed Branching Criteria
With Application of Tests II, III and V.

Problem	Size			No. of Effcnt. Points	MDM Criterion		MI Criterion		MIXED	
	p	n	m		Iter- ations	CPU (msec)	Iter- ations	CPU (msec)	Iter- ations	CPU (msec)
1	2	7	3	5	18	71	19	59	18	74
2	2	9	6	5	85	360	96	340	76	344
3	2	9	9	5	85	407	95	413	76	386
4	2	10	3	4	80	337	114	360	77	331
5	2	15	10	5	137	1160	134	944	144	1192
6	2	20	10	8	907	11509	1229	10066	976	11729
7	3	9	6	6	97	425	74	400	80	418
8	3	10	3	6	123	570	170	563	119	571

It was also observed that the number of times Test V was conclusive, i.e. a variable was assigned a definite value of one, was quite a small percentage of the times it was used. The results for the MI criterion without the use of Test V are given in Table III.4. where slight decreases in cpu time are observed.

Table III.4. Results for MI Criterion with Application
of Tests II and III.

Problem	p	Size		No. of Efficient Points	Iter- ations	CPU (msec)
		n	m			
1	2	7	3	5	19	50
2	2	9	6	5	99	321
3	2	9	9	5	93	384
4	2	10	3	4	115	354
5	2	15	10	5	135	889
6	2	20	10	8	1242	10101
7	3	9	6	6	97	417
8	3	10	3	6	101	520

These experimentations indicate that minimum infeasibility criterion is superior to minimum domination margin criterion and Tests II and III are stronger than the other tests. Considering larger problems where the number of efficient points and therefore the number of lower bound vectors will be large, the use of minimum domination margin criterion will involve a greater amount of comparisons and computations and thus will be even more time consuming. Also it is intuitive that if any one of Tests I, III, IV and V are to be used, it is more advantageous to use the one leading to smaller upper bounds. Much of the computational effort associated with these tests derives from the comparisons of the upper bound vectors formed for each variable with the lower bound vectors. The number of comparisons required remains roughly the same between these tests and has a tendency to decrease as the upper bounds get smaller, because once an upper bound vector is dominated by one lower bound vector, it is not compared with the others.

III.2.3. THE FINAL ALGORITHM AND COMPUTATIONAL RESULTS

On the basis of the experimentations with the branching criteria and the domination tests, it was decided to use the minimum infeasibility criterion for branching and use Tests II and III at selected nodes in the final algorithm, where these nodes are selected as described in the preceding section.

The algorithm was coded in FORTRAN and run on UNIVAC 1106. The computer program consists of a main program MAIG, and three subroutines FEAS, BRANF and ADVDM. The enumeration tree is structured by the main program, the bookkeeping scheme being along the lines given by Geoffrion [26]. However to accommodate the definite one assignments, and for avoiding a zero assignment to these variables during backtracking, two arrays are utilized. The first array indicates the level of the assigned variables on the current path, and the second array indicates whether a variable is assigned and if so, the value assigned to it. If a node for which the last variable assigned, x_r , was set to one is fathomed and backtracking to $x_r=0$ occurs, the previous set of infeasibilities $I_{k-1}(j)$ are used for selecting the new branching variable. These are already available and remain the same for the new node. All input and output is also coordinated in the main program.

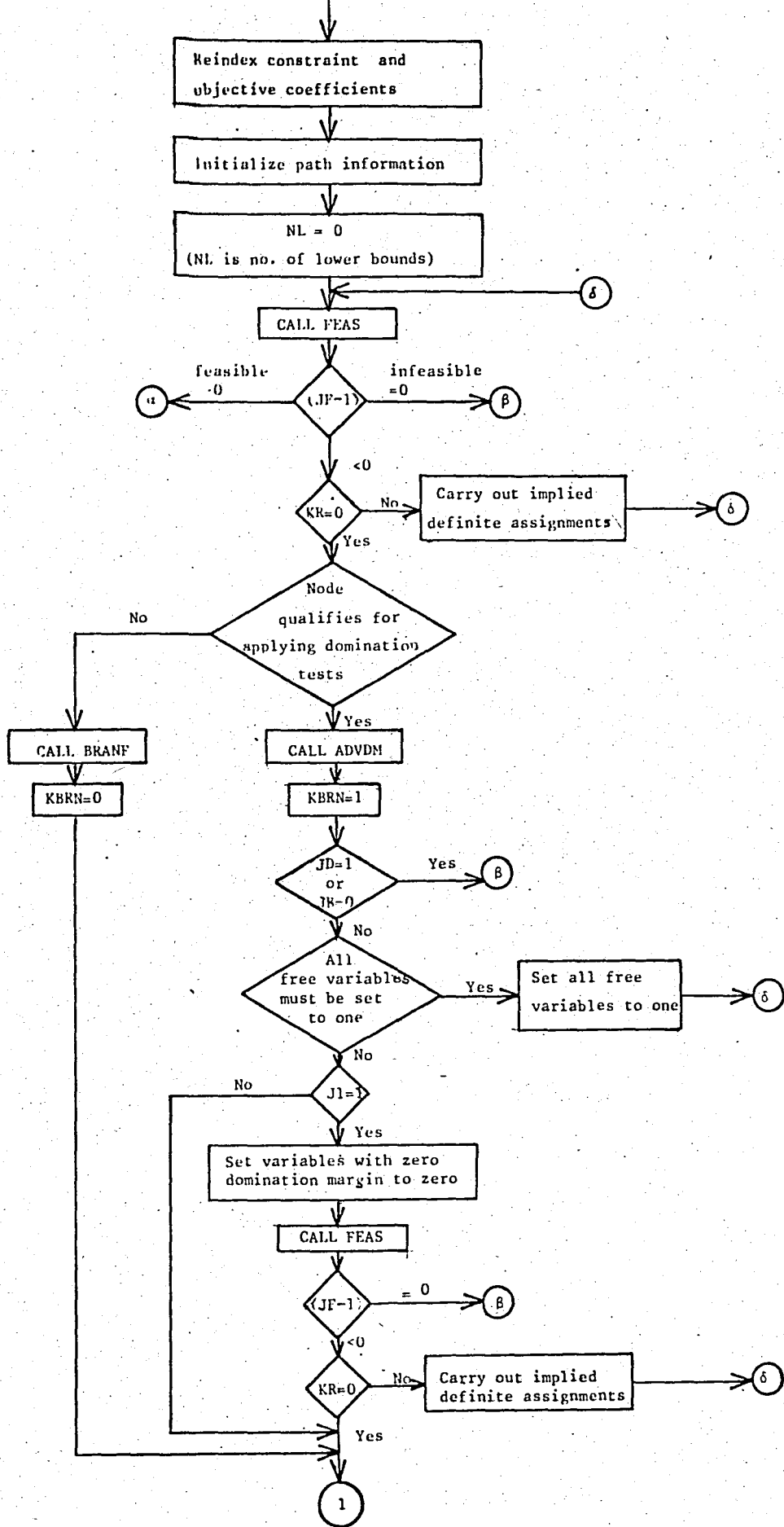
Subroutine FEAS is used to check for feasibility. Through the parameter JF specified by this subroutine and supplied to the main program, feasibility of the current node, or infeasibility of all further completions can be determined. Also if the condition $t_i = s_i$ holds for some constraint, indicating definite assignments are needed in order to achieve feasibility, the constraint number is returned to the main program via the parameter KR so that these assignments can be made.

Subroutine BRANF selects the branching variable with minimum infeasibility and returns the information to the main program through the parameters JB and MINF. JB denotes the index of the branching variable and MINF its infeasibility. If MINF is zero a feasibility check is not done after branching.

Subroutine ADVDM carries out Test II and Test III. If Test II is conclusive the parameter JD is set to one and control returns to the main program so that the current node can be fathomed. If Test II is not conclusive, Test III is carried out to determine if any variable has an advanced domination margin of zero. If so, the parameter J1 is set to one. If all variables have a zero domination margin JB is set to zero, otherwise JB stores the index of the variable with minimum domination margin. Upon returning to the main program either the current node is fathomed (JB=0) or some variables are set to zero (J1=1) or the enumeration continues by branching to X_{JB} .

The flow diagram of the algorithm is given in Figure III.3. Listings of the main program and subroutines and the definitions of the parameters used, along with a sample output are given in Appendix IV.

Computational results obtained for 19 types of problems with different number of objectives, variables and constraints are given in Table III.5. Here, iterations means the number of nodes or partial solutions generated for which feasibility test and upper and lower bound comparisons are carried out. Number of times ADVDM is called gives the number of nodes for which Tests II and III are carried out. The results give the average for four problems per problem type. The problem coefficients were generated randomly, but not so randomly as to make them too haphazard and totally



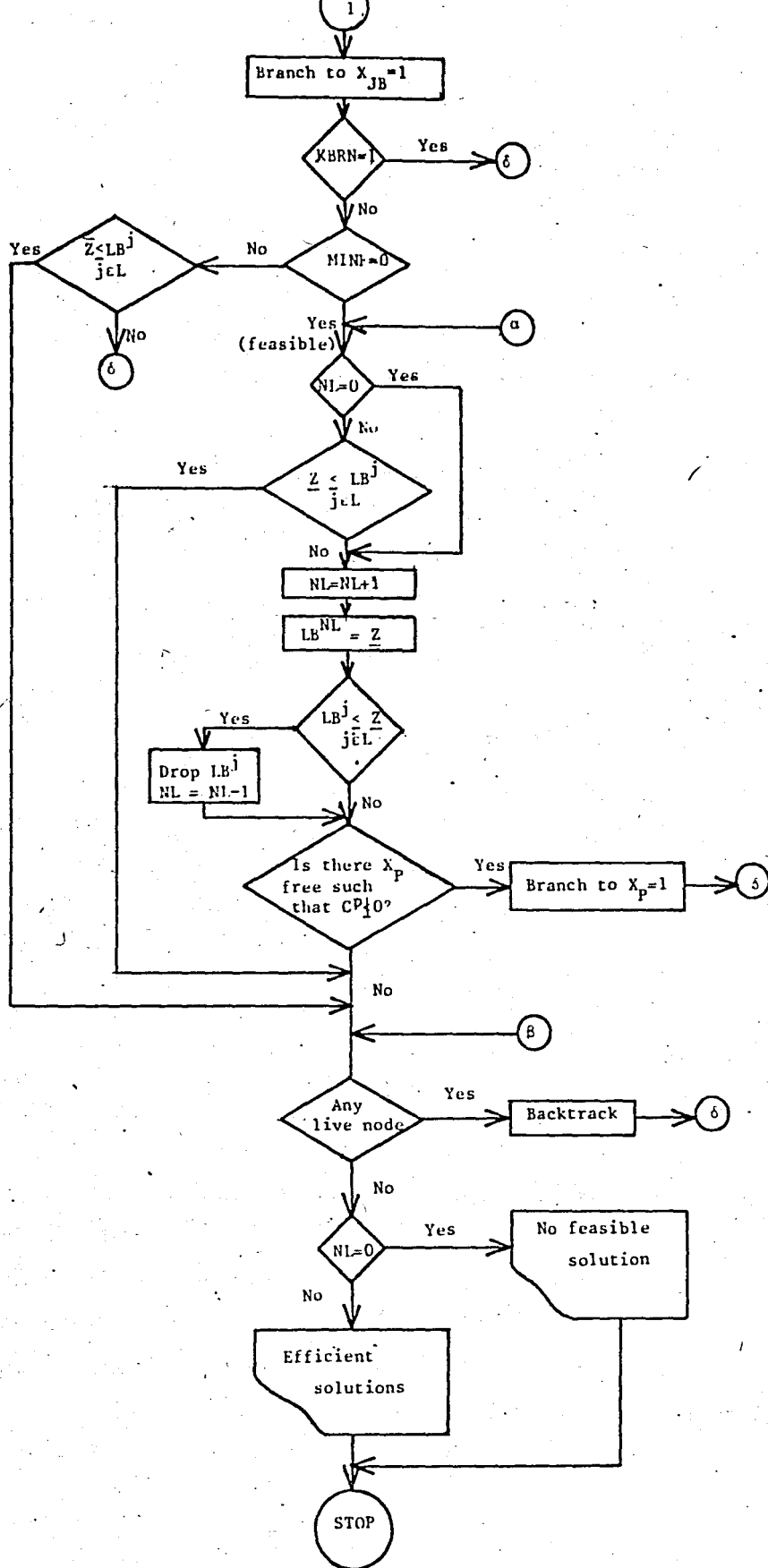


Figure III.3. Flow Diagram of Multiobjective Implicit Enumeration Algorithm

Table III.5. Computational Results

Problem	p	Size n	n	No. of Effcnt. Points	Iterations	No. of times ADVDM is called	CPU (msec)
1	2	10	6	3	77	14	306
2	2	10	8	5	84	41	503
3	2	10	10	3	53	22	338
4	2	11	10	3	71	33	470
5	2	12	10	5	128	72	971
6	2	13	10	6	221	96	1400
7	2	14	10	5	221	131	1885
8	2	15	10	8	354	218	3119
9	3	15	10	7	384	214	3396
10	3	15	12	23	936	546	12814
11	3	15	14	13	524	315	6762
12	3	15	16	9	423	252	5078
13	3	15	18	15	473	288	6413
14	3	15	20	15	573	315	7916
15	3	16	20	28	1368	821	27460
16	3	17	20	17	1135	703	22526
17	3	18	20	20	2384	1344	59437
18	3	20	20	26	3678	2390	86180
19	4	20	20	62	4073	2547	172024

unrealistic. The objective function coefficients were generated randomly in the interval $[-20, 100]$. The coefficients of the first constraint were generated randomly in the interval $[0, 100]$. For each of the following constraints, the coefficient of each variable was generated by adding a random increment in the interval $[-30, 30]$ to its coefficient in the first constraint. The right hand sides of the constraints were randomly set to between 40 % and 60 % of the sum of constraint coefficients.

The results indicate that

- i) Number of constraints does not have any systematic effect on computation time.
- ii) Computation time increases as the number of objectives and variables increase.
- iii) Computation time also increases as the number of efficient points increase (Compare problems 10 and 11).
- iv) Generally, number of efficient points increases as the number of objectives and variables increase.

The algorithm appears to be more efficient than the previous algorithms. UNIVAC 1106 is not a particularly fast machine, but still a problem with three objectives, twenty variables and twenty constraints could be solved in about 86 seconds. The computation times given by Klein and Hannan are not really comparable due to their use of APL in coding their algorithm. However they use UNIVAC 1100/89 and UNIVAC 1100/80 series are between 6.6 to 34 times faster than UNIVAC 1106. Still, the computation times reported for similar problems are much higher than the times reported here. When compared with Bitran's results, this algorithm seems definitely better. As stated before, a two objective, nine variable, four constraint problem required about 60 seconds on Burroughs 6700 whereas with this algorithm a two objective ten variable six constraint problem requires much less than a second. Although the architectures of Burroughs and Univac computers are very different and direct comparisons cannot be made, Burroughs 6700 systems are still more powerful than Univac 1106.

III.2.4. REMARKS AND EXTENSIONS ON THE ALGORITHM

The present algorithm and the algorithm of Klein and Hamman aim essentially at the same thing; generating efficient solutions utilizing implicit enumeration and in the process eliminating solutions dominated by the already available solutions. The first algorithm accomplishes this by using vectoral upper and lower bounds and the second by introduction of a set of logical

constraints within a sequential procedure. The function of the logical constraints and the lower bound vectors is the same; they also eliminate dominated solutions. The main difference between the algorithms is that the first uses a "one go" procedure aided with domination tests where each node of the enumeration tree is considered only once; whereas the second uses a sequential procedure where certain nodes have to be examined several times. An advantage of the sequential procedure is that solutions which are optimal with respect to objective one at any step of the sequential procedure are identified as efficient solutions. In the first algorithm the efficiency of no solution can be verified until enumeration is complete. However, the sequential procedure requires large amounts of storage space to keep track of nodes which require further investigation. Furthermore, passing from one problem to the next and solving each problem involves a lot of scanning and movements from one node to totally unrelated nodes which necessitate considerable amounts of calculation.

Another observation is that the screening procedure proposed by Klein and Hannan for limiting the search for efficient points can be adopted within the framework of the present algorithm. Again, the desired spacing between efficient solutions can be specified by means of a minimum variation vector d , so that the objective values of any two efficient solutions differ from each other at least by the amounts specified by the elements of the vector d .

In this case, a node, N_k , is to be fathomed if for some $j \in L$

$$\bar{z}^k \leq LB^j + d \quad \text{and} \quad \bar{z}_i^k < LB_i^j \quad \text{for some } i, \quad i=1, \dots, p.$$

because then further completions of N_k will not be sufficiently dissimilar from LB^j . The second condition is required to guarantee that the lower bound LB^j does not eliminate an efficient solution which actually dominates LB^j .

Similarly, a feasible node, N_k , is fathomed if for some $j \in L$

$$\bar{z}^k \leq LB^j + d \quad \text{and} \quad \bar{z}_i^k < LB_i^j \quad \text{for some } i, \quad i=1, \dots, p.$$

If a feasible node is not fathomed by the above condition, and a new lower bound $LB^{\ell+1}$ is found, any existing lower bound LB^j , $j=1, \dots, \ell$; for which

$$LB_j^j < LB_j^{\ell+1} + d \quad \text{and} \quad LB_i^j < LB_i^{\ell+1} \quad \text{for some } i, \quad i=1, \dots, p$$

is to be discarded.

It is shown in the following theorem that this procedure generates a subset of the efficient points of the original problem (which are dispersed at least by the specified minimum variation vector d).

THEOREM III.1. Let LB_j^1, \dots, LB_j^ℓ denote the final set of lower bound vectors obtained by the above procedure. Then LB^1, \dots, LB^ℓ are a subset of the efficient points of the original problem.

Proof. Assume LB^q for some $1 < q < \ell$ represents a dominated point. This is only possible if all efficient points dominating LB^q have been eliminated by the revised fathoming conditions.

LB^q itself cannot eliminate an efficient point, Y , which dominates LB^q , as $\bar{Z}^k \geq Y > LB^q$ for all nodes N_k along a path leading to Y , and $\bar{Z}_i^k < LB_i^q$ will never be satisfied for any $i, i=1, \dots, p$.

Also, if any lower bound eliminates an efficient point, it also eliminates all points dominated by that efficient point. Thus, if any efficient point dominating LB^q has been eliminated, LB^q must have been eliminated as well, contradicting that LB^q is contained in the final list of lower bound vectors.

With this procedure a reduced set of efficient points which are as dispersed as desired with respect to objective values can be generated. As the elements of d get bigger, fathoming will be accelerated and fewer efficient points will be generated. Thus, computation time, which is quite sensitive to number of nodes considered and to number of efficient points, will decrease considerably.

CONCLUSION

As a consequence of this study in the fields of multiobjective linear and integer linear programming, three algorithms and associated computer codes have been developed. An application in power systems expansion modelling, involving detailed analysis of various policy implications, has also been carried out.

The MOLP algorithm either generates all efficient extreme points of multiobjective linear programs, or a subset of them corresponding to a decision maker specified space of objective weights. Earlier results given by various authors are synthesized to produce an efficient algorithm facilitating the incorporation of preferences of the decision maker into the solution process. Through the observation of a monotone connectedness property, the duplication of effort is avoided so that computational efficiency can be increased.

A new algorithm for bicriterion linear programs which requires only a series of divisions and comparisons for determination of adjacent efficient extreme points, and which is computationally much more efficient with respect to existing algorithms has been given.

One possible area of utilization of the bicriterion algorithm, other than for problems with two objectives, is in generation of trade-off functions between pairs of objectives; which could be inputs to other techniques of multiobjective decision making. One such technique is the Surrogate Worth Trade Off Method developed by Haimes and Hall [33].

Another possibility is the extension of the reasoning behind the bicriterion algorithm to nonlinear problems. Starting at an efficient point, the bicriterion algorithm identifies an efficient direction where the ratio of the rate of increase of one objective to the rate of decrease of the other is maximum. A similar procedure could possibly be used in identifying efficient directions for nonlinear problems.

The major limitation of the computer codes of these two algorithms is maximum problem size. Currently both codes are restricted to problems with 50 constraints and 50 variables (excluding slack and artificial variables); and the MOLP code allows for 5 objectives. The programs utilize only core memory, and by utilizing disk storage larger sized problems could be solved. Double precision arithmetic is used in the computer programs to avoid inaccuracies due to roundoffs which could accumulate and cause problems in identification of efficient points. However, the use of double precision arithmetic also limits maximum problem size, as dimension requirements are increased.

Presently, the computer programs are suitable for educational, research and medium sized application problems. The codes are not professional ones and through further refinements and by optimizing the coding both the maximum problem size and the computational efficiency could be increased.

The MOZOLP algorithm is based on implicit enumeration which is a powerful technique for binary problems. Some domination tests aiming at identifying paths of the enumeration tree that lead to dominated solutions as higher up the tree as possible were developed and tested. The algorithm appears to be computationally more efficient than the previous algorithms for multiobjective binary linear programming.

Presently, the computer program for the MOZOLP algorithm allows for 5 objectives, 50 constraints and 70 variables. However, the maximum problem size tested was 4 objectives, 20 constraints and 20 variables as the computation time increases considerably as the number of objectives and variables increase. For problems with large number of objectives and variables, the approach may have limited applicability.

Further research and experimentation is needed to tackle large sized binary problems. One possibility is the utilization of the screening procedure discussed in section III.2.4.; either by itself or within the framework of an iterative, interactive algorithm. One could start with a sufficiently big minimum variation vector d , and generate a few efficient points. The

decision maker could be asked to evaluate these; and based on his evaluations, the problem could be modified by constraining the allowable ranges for the objectives. Continually choosing smaller values for d and constraining the problem, the decision maker's preferred solution could be generated by solving at each cycle problems having much smaller sized enumeration trees.

Reported examples of applications of multiobjective programming techniques to real-world problems, especially for integer cases, seem to be rare. A lot of insight can be gained from real-world applications and research in this direction would be valuable. Emphasis should be given to meaningful presentation of results to the decision maker, so that he can make educated judgements based on insights generated by these approaches.

It is hoped that the results and algorithms presented in this work will be beneficial to researchers in the field of multiobjective decision making.

REFERENCES

1. Anderson, D., "Investment Analysis in Electricity Supply Using Computer Models", Working Paper No.91, Economics Dept., World Bank, Wash.D.C., 1970.
2. Aneja, Y.P. and K.P.K.Nair, "Bicriteria Transportation Problem", Management Science, Vol.25, pp.73-78, 1979.
3. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", Operations Research, Vol.13, pp.517-546, 1965.
4. Beeson, R.M. and W.S.Meisel, "The Optimization of Complex Systems with Respect to Multiple Criteria", Proceedings, Systems, Man and Cybernetics Conference, IEEE, Anaheim, Calif., 1971.
5. Belenson, S.M. and K.C.Kapur, "An Algorithm for Solving Multicriterion Linear Problems with Examples", Operations Research Quarterly, Vol.21, pp.65-77, 1973.
6. Benayoun, R., J. de Montgolfier, J.Tergeny and O.Larichev, "Linear Programming with Multiple Objective Functions: Step Method (STEM)", Mathematical Programming, Vol.1, pp.366-375, 1971.
7. Bitran, G.R., "Linear Multiple Objective Programs with Zero-One Variables", Mathematical Programming, Vol.13, pp.121-139, 1977.
8. Bowman, V.J., "On the Relationship of the Tchebycheff Norm and the Efficient Frontier of Multiple-Criteria Objectives", in H.Thiriez and S.Zionts, eds., Multiple Criteria Decision Making, Springer-Verlag, Berlin, 1976.
9. Charnes, A., and W.W.Cooper, "Management Models and Industrial Applications of Linear Programming", Management Science, Vol.4, pp.38-91, 1957.

10. _____ and _____, Management Models and Industrial Applications of Linear Programming, Vol.2, John Wiley and Sons, New York, 1961.
11. Chernikova, N.V., "Algorithm for Finding a General Formula for the Nonnegative Solutions of a System of Linear Inequalities", U.S.S.R. Computational Mathematics and Mathematical Physics, Vol.5, pp.228-233, 1965.
12. Cohon, J.L. and D.H.Marks, "Multiobjective Screening Models and Water Resources Investment", Water Resources Research, Vol.9, pp.826-836, 1973.
13. _____ and _____, "A Review and Evaluation of Multiobjective Programming Techniques", Water Resources Research, Vol.11, pp.208-220, 1975.
14. _____, R.L.Church and D.P.Sheer, "Generating Multiobjective Trade-Offs: An Algorithm for Bicriterion Problems", Water Resources Research, Vol.15, pp.1001-1010, 1979.
15. Da Cunha, N.O. and E.Polak, "Constrained Minimization Under Vector Valued Criteria in Finite Dimensional Spaces", Journal of Mathematical Analysis and Applications, Vol.19, pp.103-124, 1967.
16. Ecker, J.G., and I.A.Kouada, "Finding All Efficient Extreme Points for Multiple Objective Linear Programs", Mathematical Programming, Vol.14, pp.249-261, 1978.
17. _____, N.S.Hegner and I.A.Kouada, "Generating All Maximal Efficient Faces for Multiple Objective Linear Programs", Journal of Optimization Theory and Applications, Vol.30, pp.353-381, 1980.
18. Evans, J.P. and R.E.Steuer, "A Revised Simplex Method for Linear Multiple Objective Programs", Mathematical Programming, Vol.5, pp.54-72, 1973.

19. _____ and _____, "Generating Efficient Extreme Points in Linear Multiple Objective Programming: Two Algorithms and Computational Experience", in Cochrane, J.L. and M.Zeleny, eds., Multiple Criteria Decision Making, University of South Carolina Press, 1973.
20. Fleischmann, B., "Computational Experience with the Algorithm of Balas", Operations Research, Vol.15, pp.153-155, 1967.
21. Gal, T., "Homogene Mehrparametrische Lineare Programmierung", Z.F.OR, Vol.16, pp.115-136, 1972.
22. _____, "A General Method for Determining the Set of All Efficient Solutions to a Linear Vector Maximum Problem", European Journal of Operations Research, Vol.1, pp.307-322, 1977.
23. _____ and J.Nedoma, "Multiparametric Linear Programming", Management Science, Vol.18, pp.406-422, 1972.
24. Garfinkel, R.S. and G.L.Nemhauser, Integer Programming, John Wiley and Sons, New York, 1972.
25. Gembicki, F.W. and Y.Y.Haimes, "Approach to Performance and Sensitivity Multiobjective Optimization: The Goal Attainment Method", IEEE Transactions, Vol.AC-20, pp.769-771, 1975.
26. Geoffrion, A.M., "Integer Programming by Implicit Enumeration and Balas' Method", SIAM Review, Vol.7, pp.178-190, 1967.
27. _____, "Solving Bicriterion Mathematical Programs", Operations Research, Vol.15, pp.39-54, 1967.
28. _____, "Proper Efficiency and the Theory of Vector Maximization", Journal of Mathematical Analysis and Applications, Vol.22, pp.618-630, 1968.

29. _____, J.S.Dyer and A.Feinberg, "An Interactiye Approach for Multicriterion Optimization with an Application to the Operation of an Academic Department", Management Science, Vol.19, pp.357-368, 1972.
30. Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem", Operations Research, Vol.13, pp.879-919, 1965.
31. _____ and S. Zionts, "A Note on the Additive Algorithm of Balas", Operations Research, Vol.13, pp.546-549, 1965.
32. Haimes, Y.Y., The Integration of System Identification and System Optimization, Ph.D.Dissertation, Eng. Syst.Dep., School of Eng. and Appl.Sci., Univ.of Calif., Los Angeles, 1970.
33. _____ and W.A.Hall, "Multiobjectives in Water Resources Systems Analysis: The Surrogate Worth Trade-off Method", Water Resources Research, Vol.10, pp.615-624, 1974.
34. Isermann, H., "The Enumeration of the Set of All Efficient Solutions for a Linear Multiple Objective Program", Operational Research Quarterly, Vol.28, pp.711-725, 1977.
35. Kavrakoglu, I., "Models for National Energy Policy Analysis and Planning", Automatica, Vol.16, pp.379-392, 1980.
36. _____, "Decision Analysis in the Energy Sector", Applied Mathematical Modelling, Vol.4, 1980.
37. _____, "A Dynamic Optimization Model for Energy Analysis", in I.Kavrakoglu, ed., Mathematical Modelling of Energy Systems, Sijthoff and Noordhoff, The Netherlands, 1980.
38. Klein, D. and E.Hannan, "A Branch and Bound Algorithm for the Multiple Objective Zero-One Linear Programming Problem", Working Paper No.79-2, Florida International University, 1979.

39. Koopmans, T.C., "Analysis of Production as an Efficient Combination of Activities", Activity Analysis of Production and Allocation, Cowles Commission Monograph No.13, Edited By T.C. Koopmans, John Wiley and Sons, New York, pp.33-97, 1951.
40. Kuhn, H.W. and A.W.Tucker, "Nonlinear Programming", Proceedings of The Second Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, California, pp.481-492, 1951.
41. Lee, S.M., "Interactive Integer Goal Programming: Methods and Application", paper presented at Conference on Multiple Criteria Problem Solving: Theory, Methodology, and Practice, Buffalo, New York, 1977.
42. Lin, J.G., "Multiple-Objective Problems: Pareto-Optimal Solutions by Method of Proper Equality Constraints", IEEE Transactions, Vol.AC-21, pp. 641-650, 1976.
43. Morse, J.N., "Reducing the Size of the Nondominated Set: Pruning by Clustering", Computers and Operations Research, Vol.7, pp.55-66, 1980.
44. Pareto, V., Manuale di Economia Politica, Societa Editrice Libreria, Milano, Italy, 1906; Piccola Biblioteca Scientifica No.13, Societa Editrice Libreria, Milano, Italy, 1919. Translated into English by A.S.Schwieb, as Manual of Political Economy, The MacMillan Company, New York, 1971.
45. Pasternak, H. and U.Passy, "Bicriterion Mathematical Programs with Boolean Variables", in J.L. Cochrane and M.Zeleny, eds., Multiple Criteria Decision Making, University of South Carolina Press, 1973.
46. Payne, H.J., E.Polak, D.C.Collins and W.S.Meisel, "An Algorithm for Bicriteria Optimization Based on the Sensitivity Function", IEEE Transactions, Vol.AC-20, pp. 546-548, 1975.
47. Petersen, C.C., "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R and D Projects", Management Science, Vol.13, pp.736-750, 1967.

48. Philip, J., "Algorithms for the Vector Maximum Problem", Mathematical Programming, Vol.2, pp.207-229, 1972.
49. Reid, R.W. and V.Vemuri, "On the Noninferior Index Approach to Large-Scale Multi-Criterion Systems", Journal of Franklin Institute, Vol.291, pp.241-254, 1971.
50. _____ and S.J.Citron, "On Noninferior Performance Index Vectors", Journal of Optimization Theory and Applications, Vol.7, pp.11-28, 1971.
51. Roodman, G.M., "Postoptimality Analysis in Zero-One Programming by Implicit Enumeration", Naval Research Logistics Quarterly, Vol.19, pp.435-447, 1972.
52. Roy, B. "Problems and Methods with Multiple Objective Functions", Mathematical Programming, Vol.1, pp.239-266, 1971.
53. Shapiro, J.F., "Multiple Criteria Public Investment Decision Making by Mixed Integer Programming", pp.170-182 in H.Thiriez and S.Zionts, eds., Multiple Criteria Decision Making, Springer-Verlag, Berlin, 1976.
54. Steuer, R.E., "Multiple Objective Linear Programming with Interval Criterion Weights", Management Science, Vol.23, pp.305-316, 1976.
55. _____, "A Five Phase Procedure for Implementing A Vector-Maximum Algorithm for Multiple Objective Linear Programming Problems", pp.159-169 in H.Thiriez and S.Zionts, eds., Multiple Criteria Decision Making, Springer-Verlag, Berlin, 1976.
56. _____ and A.T.Schuler, "An Interactive Multiple Objective Linear Programming Approach to a Problem in Forest Management", Operations Research, Vol.26, pp.254-269, 1978.
57. _____ and F.W.Harris, "Intra-Set Point Generation and Filtering in Decision and Criterion Space", Computers and Operations Research, Vol.7, pp.41-54, 1980.

58. Wallenius, J., H.Wallenius and P.Vartia, "An Approach to Solving Multiple Criteria Macroeconomic Policy Problems and an Application", Management Science, Vol.24, pp.1021-1030, 1978.
59. Yu, P.L. and M.Zeleny, "The Set of All Nondominated Solutions in Linear Cases and a Multicriteria Simplex Method", Journal of Mathematical Analysis and Applications, Vol.49, pp.430-468, 1975.
60. Zadeh, L.A., "Optimality and Non-Scalar Valued Performance Criteria", IEEE Transactions, Vol. AC-8, pp.59-60, 1963.
61. Zeleny, M., Linear Multiobjective Programming, Springer-Verlag, Berlin, 1974.
62. Zionts, S. and J.Wallenius, "An Interactive Programming Method for Solving the Multiple Criteria Problem", Management Science, Vol.22, pp.652-663, 1976.
63. _____, "Integer Linear Programming with Multiple Objectives", Annals of Discrete Mathematics, Vol.1, pp.551-562, 1977.
64. _____ and D.Deshpande, "A Time Sharing Computer Programming Application of a Multiple Criteria Decision Method to Energy Planning - A Progress Report", in S.Zionts, ed., Multiple Criteria Problem Solving, Springer-Verlag, 1978.

APPENDIX I: COMPUTER PROGRAM FOR MULTIOBJECTIVE LINEAR PROGRAMMING

A.I.1. DEFINITION OF VARIABLES USED IN THE PROGRAM

- M : Number of objectives
- MM : Number of constraints
- N : Number of variables
- NMM : Number of nonbasic variables in the simplex tableau
- DEL : Accuracy limit for comparison with zero
- ZZ(.) : Objective value array
- CN(.,.) : Reduced cost matrix
- BB(.) : Right hand side array
- AN(.,.) : Nonbasic coefficient matrix
- IXN(J) : Nonbasic variable associated with Jth column of CN.
- I BE(I) : Basic variable associated with Ith row of AN
- IEQ(I) : Index specifying type of the Ith constraint
(0 for \leq constraint; 1 for = constraint; 2 for \geq constraint)
- NT(.) : Nonbasic variable array of the basis to which
a move is desired
- IRT(I,.) : Nonbasic variables of Ith λ -adjacent d.f.b.(dual
feasible basis)
- IV1(I,.) : Nonbasic variables of Ith d.f.b. which is stored
to be visited later
- IV2(I,.) : Nonbasic variables of Ith already generated d.f.b.
- LOB (I,.) : Nonbasic variables of Ith alternative optimal basis
for objective M.
- L1 : Number of bases in IV1
- L2 : Number of bases in IV2
- CP(.) : Array used for forming phase one objectives, or for
other intermediary purposes
- ZP : Phase one objective value
- IPHASE : Index taking value 1 if phase one of simplex algorithm
is being carried out, 0 otherwise

IFLAG : Index taking value 0 if a move to an adjacent basis is to be made, 1 otherwise
 JT(.) : The set of nonbasic variables leading to λ -adjacent d.f.b. with nonincreasing values of objective M
 LJ : Number of elements of JT
 L(.) : The set of nonbasic variables which are candidates for being placed in the set JT
 NC : Number of elements of L
 WL(I) : Lower bound for weight of objective I
 WU(I) : Upper bound for weight of objective I
 WA(.) : The initial weighting vector
 IW : Index taking value 1 if there are limits on objective weights, 0 otherwise
 MW : Number of \geq constraints of the subproblem arising from limits on objective weights
 ALP(.,.) : The coefficient matrix of the above constraints
 BP(.) : The right hand side array of these constraints
 MWP : Number of \leq constraints of the subproblem arising from limits on objective weights
 ALN(.,.) : The coefficient matrix of these constraints
 BN(.) : The right hand side array of these constraints
 NR : Total number of constraints of the subproblem
 MR : MR = NR - MWP
 N2 : Number of artificial variables of the subproblem
 A(.,.) : The constraint coefficient matrix of the subproblem
 B(.) : The right hand side array of the subproblem
 IBQ(I) : Basic variable associated with the Ith row of A
 N1X(J) : Nonbasic variable associated with the Jth column of A
 ID(.) : Array storing nonbasic variable indices of the main problem in the order in which corresponding constraints of the subproblem are formed

A.I.2. INPUT INSTRUCTIONS

The inputs to the program consist of the parameters M, MM, N, IW and DEL; the arrays BB and IEQ; the matrices AN and CN; and if IW has the value one the arrays WL, WU and WA.

The parameters M, MM, N, IW and DEL are inputted on a single card according to the format (4I5, F12.8).

The right hand side array BB is inputted according to the format (10F8.4). The coefficient matrix AN is inputted columnwise and by its nonzero elements. First, the number of rows in which nonzero entries appear in each column is inputted according to the format (I2); then the row indices and the coefficient values are inputted according to the format (8(I2,F8.4)).

All elements of the cost matrix are inputted rowwise according to the format (10F8.4). If intervals on objective weights are specified, i.e. if IW=1, then the corresponding elements of the arrays WL, WU and WA are read for each objective and consecutively for all objectives according to the format (10F8.4).

A.I.3. PROGRAM LISTING AND SAMPLE OUTPUT

The listing of the main program and the subroutines and a sample output is given below.

```

MAY. PROBLEM SIZE IS 50 VARIABLES, 5 CONSTRAINTS, 5 OBJECTIVES
M=# OF OBJ., N=# OF VAR., M=# OF CONS., DEL=ACCURACY
CN=NONBASIC COST MATRIX, AN=NONBASIC COEFF. MATRIX, BB=RHS
IXN=NONBASIC VAR. SET, IBE=BASIC VAR. OF EQUATION
IEQ=0 FOR LE CONSTR. ; =1 FOR EQ CONSTR. ; =2 FOR GE CONSTR.
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
DIMENSION IEQ(50),NT(50),IVEC(50),LOB(10,50),IRT(50,50),IV1(50,50)
COMMON/GUL1/M,MM,NMM,DEL,BB(50),AN(50,100),CN(5,100),ZZ(5),TDF
COMMON/GUL2/CP(100),ZP,IXN(100),IBE(50),IPHASE,IFLAG,IV2(200,50)
COMMON/GUL3/LJ,JT(50),NC,L(50),WL(5),WU(5),WA(5),IW,MW,MWP,MR,ALP
*10,5),BP(10),ALN(10,5),BN(10)
COMMON/GUL4/NR,N2,A(60,65),B(60),IBQ(60),NIX(65),ID(60)
READ(5,900) M,MM,N,IW,DEL
10 FORMAT(4I5,F12.8)
WRITE(6,1883)M,N,MM
13 FORMAT(1H1,10X,,SOLUTIONS FOR THE MULTIOBJECTIVE LP PROBLEM,,//,11
*,WITH,,I1,, OBJECTIVES ,,I2,, VARIABLES ,,I2',CONSTRAINTS,,//,
*1X,45(,*,))
READ(5,901) (BB(I),I=1,MM)
11 FORMAT(10F8.4)
READ(5,902) (IEQ(I),I=1,MM)
12 FORMAT(40I2)
DO 910 J=1,N
READ(5,903) KR
READ(5,904) (I,AN(I,J),K=1,KR)
10 CONTINUE
READ(5,901)((CN(I,J),J=1,N),I=1,M)
13 FORMAT(I2)
14 FORMAT(8(I2,F3.4))
MWP=0
MW=M
DO 1 J=1,N
IXN(J)=J
DO 1 I=1,M
1 CN(I,J)=-CN(I,J)
IW=1 IF THERE ARE LIMITS ON OBJECTIVE WEIGHTS; =0 OTHERWISE
IF(IW.EQ.0) GO TO 4
READ(5,901)(WL(I),WU(I),WA(I),I=1,M)
CALL WEIGHT
DO 12 J=1,N
12 CN(M+1,J)=0.
DO 13 I=1,N
DO 13 I=1,M
13 CN(M+1,J)=CN(M+1,J)+WA(I)*CN(I,J)
M=M+1
4 K=0
N1=N
DO 10 I=1,MM
IF(IEQ(I).EQ.1) GO TO 10
IF(IEQ(I).EQ.0) GO TO 9
N1=N1+1
AN(I,N1-K)=-1.
IXN(N1-K)=N1
GO TO 10
9 N1=N1+1
K=K+1
IBE(I)=N1

```



```

DO 25 J=1,NMM
IF(IBE(I),EQ.0) GO TO 20
N1=N1+1
IBE(I)=N1
CONTINUE
N1=N1-N2
N1 IS THE # OF ARTIFICIAL VARIABLES
IF(N1.EQ.0) GO TO 100
IPHASE=1
ZP=0.
DO 25 J=1,NMM
CP(J)=0.
FORM PHASE1 OBJECTIVE
DO 30 I=1,MM
IF(IBE(I).LE.N2) GO TO 30
ZP=ZP-BB(I)
DO 30 J=1,NMM
CP(J)=CP(J)-AN(I,J)
CONTINUE
MINIMIZE PHASE1 OBJECTIVE
KE=0
DO 50 J=1,NMM
IF(CP(J).GE.-DEL) GO TO 50
KE=J
GO TO 55
CONTINUE
IF(ZP.GE.-DEL) GO TO 70
WRITE(6,805)
FORMAT(1H1,40X, 'THERE IS NO FEASIBLE SOLUTION OF THE PROBLEM, //')
STOP
CALL LEAV(KE,KMIN)
IF(KMIN.EQ.0) WRITE(6,805)
IF(IBE(KMIN).LE.N2) GO TO 65
CALL PIVOT(KE,KMIN)
NMM=NMM-1
N1=N1-1
IF(KE.EQ.(NMM+1)) GO TO 40
DO 60 J=KE,NMM
IXN(J)=IXN(J+1)
CP(J)=CP(J+1)
DO 58 I=1,M
CN(I,J)=CN(I,J+1)
DO 59 I=1,MM
AN(I,J)=AN(I,J+1)
CONTINUE
GO TO 40
KL=IBE(KMIN)
CALL PIVOT(KE,KMIN)
IXN(KE)=KL
GO TO 40
PHASE1 OBJECTIVE MINIMIZED
IF(N1.EQ.0) GO TO 100
THERE ARE STILL ARTIFICIAL VAR. IN BASIS
K=1
DO 80 I=K,MM
IF(BB(I).GT.DEL) GO TO 80
IF(IBE(I).LE.N2) GO TO 80

```

```

CONTINUE
NULL EQUATION, DROP IT
MM=M4-1
N1=N1-1
IF(I.EQ.(MM+1)) GO TO 84
DO 83 I1=I,MM
IBF(I1)=IBE(I1+1)
BB(I1)=BB(I1+1)
DO 83 J=1,NMM
AN(I1,J)=AN(I1+1,J)
3 CONTINUE
IF(N1.EQ.0) GO TO 100
K=I
GO TO 79
4 IF(N1.EQ.0) GO TO 100
ELIMINATE ARTIFICIAL VARIABLE
5 CALL PIVOT(J,I)
NMM=NMM-1
N1=N1-1
IF(J.EQ.(NMM+1)) GO TO 88
DO 86 JJ=J,NMM
IXN(JJ)=IXN(JJ+1)
DO 87 I1=1,M
87 CN(I1,JJ)=CN(I1,JJ+1)
DO 89 I1=1,MM
89 AN(I1,JJ)=AN(I1,JJ+1)
96 CONTINUE
98 IF(N1.EQ.0) GO TO 100
K=I+1
GO TO 79
AT THIS POINT WE HAVE INITIAL B.F.S.
00 IPHASE=0
01 KE=0
DO 110 J=1,NMM
IF(CN(M,J).GE.-DEL) GO TO 110
KE=J
CALL LEAV(KE,KMIN)
IF(KMIN.EQ.0) GO TO 150
KL=IBE(KMIN)
CALL PIVOT(KE,KMIN)
IXN(KE)=KL
GO TO 101
10 CONTINUE
GO TO 200
60 WRITE(6,806) M
86 FORMAT(1H1,50X,,OBJECTIVE,,12,, IS UNBOUNDED,/)
STOP
00 IF(IW.EQ.0)GO TO 204
M=M-1
GO TO 289
04 DO 1500 J=1,NMM
IF(CN(M,J).GE.DEL)GO TO 1500
CALL LEAV(J,KMIN)
DO 1510 I=1,M
AA=CN(I,J)
IF((AA*BB(KMIN)).GT.0.)GO TO 1500
0 CONTINUE

```

```

CHECK FOR DUAL FEASIBILITY
CALL EDGE
IF (IDF) 289, 289, 260
0 LA=1
LA= INDEX OF ALTERNATIVE Opt. BASIS FOR OBJ. 1
DO 202 I=1, NMM
02 IVEC(I)=IXN(I)
CALL SORT(IVEC, NMM)
DO 203 I=1, NMM
03 LOB(LA, I)=IVEC(I)
0 DO 206 J=1, NMM
IF (CB(M, J).GE.DLL) GO TO 206
KE=J
CALL LEAV(KE, KMIN)
IF (BB(KMIN).GE.DEL) GO TO 206
LA=LA+1
DO 207 I=1, NMM
07 LOB(LA, I)=IXN(I)
LOB(LA, KE)=IBF(KMIN)
DO 208 I=1, NMM
08 IVEC(I)=LOB(LA, I)
CALL SORT(IVEC, NMM)
DO 209 I=1, NMM
09 LOB(LA, I)=IVEC(I)
LA'=LA-1
DO 210 I=1, LAM
DO 211 II=1, NMM
IF (LOB(LA, II).NE.LOB(I, II)) GO TO 210
11 CONTINUE
LA=LA-1
GO TO 206
10 CONTINUE
KL=IBF(KMIN)
CALL PIVOT(KE, KMIN)
IXN(KE)=KL
CALL EDGE
IF (IDF.EQ.1) GO TO 270
06 CONTINUE
INITIAL EFFICIENT VERTEX FOUND
09 WRITE(6, 800)
00 FORMAT(///, 50X, 'INITIAL EFFICIENT VERTEX, ', //, 48X, 26(('*'), //)
WRITE(6, 802) (IBF(I), BB(I), I=1, MM)
02 FORMAT(3X, 'BASIC VARIABLES, ', //, 2X, 15(('*'), //, (7( X, J3, =, F10.3
*, //))
WRITE(6, 803) (I, ZZ(I), I=1, M)
03 FORMAT(//, 3X, 'OBJECTIVES, ', //, 2X, 10(('*'), //, 3X, 'OBJ, 5(I1, =, F10.3
*2X, 'OBJ, )
IFLAG=0
L2=# OF COMPUTED EFF. VERTICES, L1=# OF NOT YET COMPUTED EFF. VERTI
L2=1
L1=0
DO 290 K=1, NMM
90 IVEC(K)=IXN(K)
CALL SORT(IVEC, NMM)
DO 295 K=1, NMM
95 IV2(L, K)=IVEC(K)

```

```

35 FORM(77, JDX, NO. OF EFFICIENT VERTICES IS GREATER THAN 200, //, 1
  *X, RUM IS TERMINATED, )
  STOP
01 CALL EDGE
  DETERMINE FEASIBLE ADJACENT SET
  LJ=# OF EFFICIENT EDGES
  LRT=# OF ADJACENT EFF. EXTR. PTS.
11 LRT=0
  IF(LJ.EQ.0) GO TO 700
  DO 350 I=1, LJ
  J=JT(I)
  CALL LEAV(J, KMIN)
  LRT=LRT+1
  KK=JBT(KMIN)
  DO 320 I=1, NMM
20 IRT(LRT, I)=IXN(I)
  IRT(LRT, J)=KK
  SORT INDICES OF NBAS. VAR. OF EXTR. POINT
  DO 330 I=1, NMM
30 IVEC(I)=IRT(LRT, I)
  CALL SORT(IVEC, NMM)
  DO 99 I=1, NMM
99 IRT(LRT, I)=IVEC(I)
350 CONTINUE
  LR1=0
  FORM THE SET IRT-IV2
  DO 370 I=1, LRT
  DO 370 II=1, L2
  DO 360 K=1, NMM
  IF(IRT(I, K).NE.IV2(II, K)) GO TO 370
360 CONTINUE
  GO TO 390
370 CONTINUE
  LR1=LR1+1
  DO 380 K=1, NMM
380 IRT(LR1, K)=IRT(I, K)
390 CONTINUE
  IF(LR1.EQ.0) GO TO 700
  LR2=# OF ELEMENTS IN IR22
  LR2=0
  FORM THE SET IRT-IV1
  IF(L1)395, 395, 396
395 LR2=LR1
  GO TO 435
396 DO 430 I=1, LR1
  DO 410 II=1, L1
  DO 400 K=1, NMM
  IF(IRT(I, K).NE.IV1(II, K)) GO TO 410
400 CONTINUE
  GO TO 430
410 CONTINUE
  LR2=LR2+1
  DO 420 K=1, NMM
420 IRT(LR2, K)=IRT(I, K)
430 CONTINUE
  IF(LR2.EQ.0) GO TO 600
  NEW VERTEX SELECTED FROM R2

```

```

IF(LR2.EQ.0) GO TO 590
FORM THE SET IV1
DO 540 I=1,LR2
L1=L1+1
DO 540 K=1,NMM
IV1(L1,K)=IRT(I,K)
CALL MOVE(NT)
AT NEW EFF. VERTEX NEW ITERATION STARTS
GO TO 300
NEW VERTEX SELECTED FROM R1
L2=L2+1
DO 610 K=1,NMM
NT(K)=IRT(LR1,K)
IV2(L2,K)=NT(K)
DO 650 I=1,L1
DO 620 K=1,NMM
IF(NT(K).NE.IV1(I,K)) GO TO 650
CONTINUE
L1=L1-1
IF(I.EQ.L1+1) GO TO 690
DO 630 IJ=I,L1
DO 630 K=1,NMM
IV1(I,I,K)=IV1(IJ+1,K)
GO TO 690
CONTINUE
CALL MOVE(NT)
GO TO 300
NEW VERTEX SELECTED FROM v1
IF(L1.EQ.0) GO TO 9999
L2=L2+1
IFLAG=1
WRITE(6,606)IFLAG
FORMAT(2X,,IFLAG,,I2)
DO 710 K=1,NMM
NT(K)=IV1(L1,K)
IV2(L2,K)=NT(K)
L1=L1-1
CALL MOVE(NT)
IFLAG=0
GO TO 300
WRITE(6,777)
FORMAT(///,43X,,ALL EFFICIENT VERTICES HAVE BEEN ENUMERATED,)
WRITE(6,779) L2
FORMAT(///,43X,,NO.EFFICIENT EXTREME POINTS = ,,I3)
STOP
END

```

```

COMMON/GULP/CP(100),ZP,IX(100),IPHASE,IFLAG
COMMON/GUL3/LJ,JT(50),NC,L(50),WL(5),WU(5),WA(5),IW,MW,MWP,MR,ALP
*10,5),BP(10),ALN(10,5),BN(10)
COMMON/GUL4/NR,N2,A(60,65),B(60),IBQ(60),NIX(65),ID(60)
LJ=0
NC=0
NR=MW-M
N1=M
DO 1 J=1,65
DO 1 I=1,60
1 A(I,J)=0.
DO 3 J=1,NMM
3 CP(J)=0.
DO 2 J=1,NMM
DO 2 I=1,M
2 CP(J)=CP(J)+CN(I,J)
IF(NR.EQ.0) GO TO 11
DO 4 I=1,NR
N1=N1+1
4 A(I,N1)=-1.
DO 9 I=1,NR
B(I)=BN(I)
DO 9 J=1,M
9 A(I,J)=ALN(I,J)
1 DO 10 J=1,NMM
IF(CP(J).GE.0.) GO TO 10
NR=NR+1
N1=N1+1
A(NR,N1)=-1.0
ID(NR)=J
B(NR)=-CP(J)
DO 5 I=1,M
5 A(NR,I)=CN(I,J)
IF(IW.EQ.1) GO TO 7
IF(CN(M,J).LE.-DEL) GO TO 10
7 NC=NC+1
L(NC)=J
10 CONTINUE
AT THIS POINT ALL EQUATIONS WITH NEGATIVE RIGHT-HAND-SIDES STORED
N3=NR
DO 20 J=1,NMM
IF(CP(J).LT.0.) GO TO 20
IF CJ=0 EFFICIENT
DO 12 I=1,M
IF(ABS(CN(I,J)).GT.DEL) GO TO 14
12 CONTINUE
LJ=LJ+1
JT(LJ)=J
GO TO 20
4 DO 15 I=1,M
IF(CN(I,J).LT.-DEL) GO TO 16
5 CONTINUE
AS CJ>0 DOMINATED
GO TO 20
16 NR=NR+1
ID(NR)=J
N1=N1+1

```

```

1 IF (C1(M,J).LE.-DEL) GO TO 20
NC=NC+1
L(NC)=J
CONTINUE
NR=NR
MR=N1
IF (MWP.EQ.0) GO TO 350
DO 320 I=1,MWP
NR=NR+1
N1=N1+1
B(NR)=BP(I)
IBO(NR)=N1
DO 320 J=1,M
A(NR,J)=ALP(I,J)
THIS POINT EQUATIONS WITH POSITIVE RHS STORED. NOW ADD ARTIFITIAL VAR
N2=N3+M
DO 25 J=1,N2
NIX(J)=J
CP(J)=0.
IF (N3.EQ.0) GO TO 100
DO 30 J=1,N3
N1=N1+1
) IBO(J)=N1
N1=N1-N3
ELIMINATE BASIC VAR. FROM PH1 OBJECTIVE
ZP=0
DO 50 I=1,N3
ZP=ZP-B(I)
DO 50 J=1,N2
CP(J)=CP(J)-A(I,J)
MINIMIZE PH1 OBJECTIVE
) KE=0
AMIN=0.
DO 70 J=1,N2
IF (CP(J).GE.AMIN) GO TO 70
AMIN=CP(J)
KE=J
CONTINUE
) IF (KL.NE.0) GO TO 71
IF (ZP.GE.-DEL) GO TO 75
IDF=1
RETURN
1 CALL LEAV2(KE,KMIN)
IF (IBO(KMIN).LE.N1) GO TO 74
CALL PIVOT2(KE,KMIN)
N2=N2-1
N3=N3-1
IF (KE.EQ.(N2+1)) GO TO 72
DO 73 J=KE,N2
NIX(J)=NIX(J+1)
CP(J)=CP(J+1)
DO 73 I=1,NR
A(I,J)=A(I,J+1)
3 CONTINUE
2 IF (N3.EQ.0) GO TO 100
GO TO 60

```

```

5 IF(N3.EQ.0) GO TO 100
  THERE ARE STILL BASIC ARTIFICIAL VARIABLES (WITH VALUE ZERO)
  K=1
9 DO 80 I=K,NR
  IF(B(I).GT.DEL) GO TO 80
  IF(IBM(I).LE.N1) GO TO 80
  GO TO 81
0 CONTINUE
1 DO 82 J=1,N2
  IF(ABS(A(I,J)).GT.DEL) GO TO 85
2 CONTINUE
  NULL EQUATION. DROP IT
  NR=NR-1
  N3=N3-1
  IF(I.EQ.NR+1) GO TO 84
  DO 83 II=I,NR
  IBM(II)=IBM(II+1)
  B(II)=B(II+1)
  DO 83 J=1,N2
  A(II,J)=A(II+1,J)
3 CONTINUE
  IF(N3.EQ.0) GO TO 100
  K=I
  GO TO 79
4 IF(N3.EQ.0) GO TO 100
  ELIMINATE ARTIFICIAL VARIABLES
5 CALL PIVOT(J,I)
  N2=N2-1
  N3=N3-1
  IF(J.EQ.(N2+1)) GO TO 88
  DO 86 JJ=J,N2
  NIX(JJ)=NIX(JJ+1)
  DO 86 II=1,NR
  A(II,JJ)=A(II,JJ+1)
6 CONTINUE
8 IF(N3.EQ.0) GO TO 100
  K=I+1
  GO TO 79
T THIS POINT WE HAVE A FEASIBLE TABLEAU. CHECK FOR NONREDUNDANT-VARI
00 SUM=0.
  DO 90 J=1,N2
  K=NIX(J)
  IF(K.GT.M) GO TO 90
  WA(K)=1.
  SUM=SUM+WA(K)
90 CONTINUE
  DO 92 I=1,NR
  K=IBM(I)
  IF(K.GT.M) GO TO 92
  WA(K)=B(I)+1.
  SUM=SUM+WA(K)
92 CONTINUE
  DO 95 I=1,M
95 WA(I)=WA(I)/SUM
  WRITE(6,850) (WA(I),I=1,M)
50 FORMAT(//,30X,'CORRESPONDING OBJECTIVE WEIGHTS ARE :',5F6.3)
  IF(NC.EQ.0) RETURN

```



```

CALL SCAN(K)
IF(NC.EQ.0) RETURN
0 CONTINUE
DO 120 I=1,NR
IF(B(I).GT.DEL) GO TO 120
IF(ISO(I).LE.MW) GO TO 120
IF(ISO(I).GT.MR) GO TO 120
K=HO(I)-M
K=ID(K)
CALL SCAN(K)
IF(NC.EQ.0) RETURN
0 CONTINUE
DEF IF ANY WJ CAN BE MADE NONBASIC WITH ONE PIVOT
DO 130 J=1,N2
CALL LEAV2(J,KMIN)
IF(KMIN.EQ.0) GO TO 130
KK=ISO(KMIN)
IF(KK.LE.MW) GO TO 130
IF(KK.GT.MR) GO TO 130
K=KK-M
K=ID(K)
CALL SCAN(K)
IF(NC.EQ.0) RETURN
30 CONTINUE
CALL DROP
IF(NC.EQ.0) RETURN
SELECT ONE VARIABLE FROM L AND MINIMIZE
40 K=L(NC)
DO 145 I=1,NRR
IF(ID(I).EQ.K) GO TO 146
45 CONTINUE
46 K=T+M
DO 150 I=1,NR
IF(ISO(I).EQ.K) GO TO 151
50 CONTINUE
51 ZP=-B(I)
DO 160 J=1,N2
60 CP(J)=-A(I,J)
61 KE=0
AMIN=-DEL
DO 170 J=1,N2
IF(CP(J).GE.AMIN) GO TO 170
AMIN=CP(J)
KE=J
70 CONTINUE
IF(KE.NE.0) GO TO 175
NO ENTERING VARIABLE
IF(ZP.LT.-DEL) GO TO 171
VARIABLE HAS MINIMUM VALUE=0
LJ=LJ+1
JT(LJ)=L(NC)
NC=NC-1
IF(NC.EQ.0) RETURN
GO TO 105
VARIABLE HAS MINIMUM VALUE>0
71 NC=NC-1

```

```
CALL PIVOT2(KF,KMIN)  
NIX(KF)=KL  
CALL DROP  
IF(NC.EQ.0) RETURN  
GO TO 161  
END
```

```

COMMON/GUL2/CP(100),ZP,IXN(100),IBF(50),IPHASE,IFLAG,IV2(200,50)
FLAG=0 IF MOVE IS TO ADJACENT VERTEX, =1 OTHERWISE.
DO 1 II=1,NMM
DO 2 JJ=1,NMM
IF(IXN(II).EQ.INT1(JJ)) GO TO 1
2 CONTINUE
KE=II
IF(IFLAG.EQ.0) GO TO 500
(KE,TH NONBASIC VAR. WILL BE ENTERING
DO 10 IL=1,NMM
DO 11 J=1,NMM
IF(INT1(IL).EQ.IXN(J)) GO TO 10
1 CONTINUE
KL=INT1(IL)
KMIN=0
DO 13 I=1,MM
IF(KL.NE.IBF(I)) GO TO 13
KMIN=I
IF(ABS(AN(KMIN,KE)).GT.DEL) GO TO 502
GO TO 10
3 CONTINUE
0 CONTINUE
2 CALL PIVOT(KE,KMIN)
IXN(KE)=KL
1 CONTINUE
GO TO 900
0 CALL LEAV(KE,KMIN)
KL=IBF(KMIN)
CALL PIVOT(KE,KMIN)
IXN(KE)=KL
0 WRITE(6,801)
1 FORMAT(///,55X,,NEW EFFICIENT VERTEX,,//,53X,24(,*),///)
WRITE(6,802)(IBF(I),BB(I),I=1,MM)
2 FORMAT(3X,,BASIC VARIABLES,,//,2X,15(,*),//,(7( X,,J3,, =,,F10.3)
*,/))
WRITE(6,803)(I,ZZ(I),I=1,M)
3 FORMAT(//,3X,,OBJECTIVES,,//,2X,10(,*),//,3X,,OBJ,,5(I),, =,,F10.3,
*2X,,OBJ,,)
RETURN
END

```

```

COMMON/GUL4/NR,N2,A(60,65),B(60),IBQ(60),NIX(65),ID(60)
STORE WEIGHT EQNS WITH NEG. RHS
NR=0
DO 200 I=1,M
IF(WU(I).GE.(1.-DEL)) GO TO 190
RHS=M-1./WU(I)
IF(RHS.GE.-DEL) GO TO 190
NR=NR+1
DO 185 J=1,M
5 ALN(NR,J)=1.
ALN(NR,I)=1.-1./WU(I)
BN(NR)=-RHS
GO TO 200
0 IF(WL(I).LT.DEL) GO TO 200
RHS=1./WL(I)-M
IF(RHS.GE.-DEL) GO TO 200
NR=NR+1
DO 195 J=1,M
5 ALN(NR,J)=1.
ALN(NR,I)=1./WL(I)-1.
BN(NR)=-RHS
0 CONTINUE
MW=NR+M
NOW STORE EQNS WITH POS RHS
K=0
DO 300 I=1,M
IF(WU(I).GE.(1.-DEL)) GO TO 250
RHS=M-1./WU(I)
IF(RHS.LT.-DEL) GO TO 250
K=K+1
DO 240 J=1,M
0 ALP(K,J)=-1.
ALP(K,I)=1./WU(I)-1.
BP(K)=RHS
0 IF(WL(I).LT.DEL) GO TO 300
RHS=1./WL(I)-M
IF(RHS.LT.-DEL) GO TO 300
K=K+1
DO 260 J=1,M
0 ALP(K,J)=1.
ALP(K,I)=1.-1./WL(I)
BP(K)=RHS
0 CONTINUE
MWP=K
RETURN
END

```

```
DO 1 I=1,NC  
IF(K.EQ.L(I)) GO TO 2  
CONTINUE  
RETURN  
LJ=LJ+1  
JT(LJ)=K  
NC=NC-1  
IF(NC.EQ.0) RETURN  
IF(I.EQ.(NC+1)) RETURN  
DO 3 IC=I,NC  
L(IC)=L(IC+1)  
RETURN  
END
```

```
*10,5),BP(10),ALH(10,5),BN(10)
COMMON/GUL4/NR,N2,A(60,65),B(60),IBQ(60),NIX(65),ID(60)
IK=1
DO 4 I=IK,NR
DO 2 J=1,N2
IF(A(I,J).GE.DEL) GO TO 4
CONTINUE
ROW I NEGATIVE
IF(IBQ(I).LE.MW) GO TO 4
IF(IBQ(I).GT.MR) GO TO 4
K=IBQ(I)-M
K=ID(K)
DO 3 J=1,NC
IF(K.EQ.L(J)) GO TO 5
CONTINUE
GO TO 7
CONTINUE
RETURN
NC=NC-1
IF(NC.EQ.0) RETURN
IF(J.EQ.(NC+1)) GO TO 7
DO 6 IC=J,NC
L(IC)=L(IC+1)
UPDATE A-MATRIX
NR=NR-1
IF(I.EQ.(NR+1)) RETURN
DO 8 II=I,NR
IBQ(II)=IBQ(II+1)
B(II)=B(II+1)
DO 8 J=1,N2
A(II,J)=A(II+1,J)
CONTINUE
IK=I
GO TO 1
END
```

```

IBF(KMIN)=IXN(KE)
AA=AN(KMIN,KE)
NORMALIZE
DO 14 J=1,NMM
+ AN(KMIN,J)=AN(KMIN,J)/AA
BB(KMIN)=BB(KMIN)/AA
AN(KMIN,KE)=1./AA
ELIMINATE ENTERING VAR. FROM OTHER EQNS.
DO 15 I=1,MM
IF(I.EQ.KMIN) GO TO 15
AA=AN(I,KE)
BB(I)=BB(I)-AA*BB(KMIN)
DO 16 J=1,NMM
6 AN(I,J)=AN(I,J)-AA*AN(KMIN,J)
AN(I,KE)=-AN(KMIN,KE)+AA
5 CONTINUE
DO 17 I=1,M
AA=CN(I,KE)
ZZ(I)=ZZ(I)-AA*BB(KMIN)
DO 18 J=1,NMM
8 CN(I,J)=CN(I,J)-AN(KMIN,J)*AA
CN(I,KE)=-AN(KMIN,KE)*AA
7 CONTINUE
IF(IPHASE.NE.1) GO TO 10
AA=CP(KE)
ZP=ZP-AA*BB(KMIN)
DO 9 J=1,NMM
9 CP(J)=CP(J)-AN(KMIN,J)*AA
CP(KE)=-AN(KMIN,KE)*AA
10 CONTINUE
RETURN
END

```

```
IBO(KMIN)=IIX(KE)
AA=A(KMIN,KE)
NORMALIZE
DO 1 J=1,N2
A(KMIN,J)=A(KMIN,J)/AA
B(KMIN)=B(KMIN)/AA
A(KMIN,KE)=1./AA
ELIMINATE ENTERING VARIABLE FROM OTHER EQUATIONS
DO 3 I=1,NR
IF(I.EQ.KMIN) GO TO 3
AA=A(I,KE)
B(I)=B(I)-AA*B(KMIN)
DO 2 J=1,N2
A(I,J)=A(I,J)-AA*A(KMIN,J)
A(I,KE)=-A(KMIN,KE)*AA
3 CONTINUE
AA=CP(KE)
ZP=ZP-AA*B(KMIN)
DO 4 J=1,N2
CP(J)=CP(J)-A(KMIN,J)*AA
CP(KE)=-A(KMIN,KE)*AA
RETURN
END
```



```
KMIN=0  
DO 5 I=1,MM  
IF(AN(I,J).LE.DEL) GO TO 5  
AA=BB(I)/AN(I,J)  
IF(AA.GE.AMIN)GO TO 5  
AMIN=AA  
KMIN=I  
5 CONTINUE  
RETURN  
END
```

AMIN=999999

KMIN=0

DO 1 T=1,NR

IF(A(T,J).LE.DEL) GO TO 1

AA=B(J)/A(T,J)

IF(AA.GE.AMIN) GO TO 1

AMIN=AA

KMIN=T

1 CONTINUE

RETURN

END

```
IS THE DIMENSION OF THE VECTOR  
JH=I-1  
DO 1 I=1,NH  
II=I+1  
DO 1 J=II,N  
IF(IV(I).LE.IV(J)) GO TO 1  
ITEMP=IV(I)  
IV(I)=IV(J)  
IV(J)=ITEMP  
CONTINUE  
RETURN  
END
```

INITIAL EFFICIENT VERTEX

BASIC VARIABLES

X001 = 100.000 X009 = 100.000 X010 = 100.000 X011 = 100.000 X012 = 100.000 X005 = 20.000 X014 = 40.000

OBJECTIVES

OBJ1 = -80.000 OBJ2 = 100.000 OBJ3 = 60.000 OBJ4 = 40.000 OBJ

CORRESPONDING OBJECTIVE WEIGHTS ARE: .100 .300 .350 .240

NEW EFFICIENT VERTEX

BASIC VARIABLES

X001 = 94.515 X009 = 90.769 X010 = 100.000 X002 = 11.539 X005 = 14.615 X014 = 56.154

OBJECTIVES

OBJ1 = -95.233 OBJ2 = 38.452 OBJ3 = 120.769 OBJ4 = 6.154 OBJ

CORRESPONDING OBJECTIVE WEIGHTS ARE: .039 .123 .397 .450

NEW EFFICIENT VERTEX

BASIC VARIABLES

X001 = 7.005 X007 = 9.235 X010 = 53.822 X011 = 71.975 X002 = 10.191 X005 = 7.070 X014 = 78.790

APPENDIX II: COMPUTER PROGRAM FOR BICRITERION LINEAR PROGRAMMING

A.II.1. DEFINITION OF VARIABLES USED IN THE PROGRAM

The variables used in the program are defined in the same way as in the multiobjective computer program except for the variables WL and WU which are here scalar quantities, giving the lower and upper bounds respectively on the second objective. This is due to the fact that specifying intervals on the weight of the second objective also determines the allowable range for the weight of the first objective. Furthermore, in such cases, the initial weights on the objectives are taken as $1-WU$ and WU respectively.

A.II.2. INPUT INSTRUCTIONS

Data input is again in the same way as in the multiobjective computer program, except for the first data card. Here the parameters MM, N, DEL, WL and WU are read from the first data card according to the format (2I5,3F12.8). If intervals on objective weights are not to be specified, then values of 0 and 1 should be inputted respectively for WL and WU.

A.II.3. PROGRAM LISTING AND SAMPLE OUTPUT

The listing of the main program and the subroutines and a sample output is given below.

```

*****
# OF OBJ., N# OF VAR., M# OF CONS., DEL=ACCURACY
#NONBASIC COST MATRIX, AN=NONBASIC COEFF. MATRIX, BB=RHS
CN=NONBASIC VAR. SET, IBE=BASIC VAR. OF EQUATION
IEQ=0 FOR LE CONSTR. ; =1 FOR EQ CONSTR. ; =2 FOR GE CONSTR.
IMPLICIT DOUBLE PRECISION(A-H,0-Z)
DIMENSION IEQ(50),NT(50),IVEC(50),IRT(50,50),IV1(50,50),IV2(200,50
*)
COMMON/GUL1/M,MM,NMM,DEL,BB(50),AN(50,100),CN(3,100),ZZ(3)
COMMON/GUL2/CP(100),ZP,IXN(100),IRE(50),IPHASE,IFLAG
COMMON/GUL3/LJ,JT(50),WU,WL
M=2
READ(5,900)MM,N,DEL,WL,WU
FORMAT(2I5,3F12.8)
WRITE(6,1983)N,MM
FORMAT(1H1,10X,,SOLUTIONS FOR BICRITERIA LP PROBLEM,,//,11X,,WITH ,
*,I2,, VARIABLES ,,I2,, CONSTRAINTS,,//,11X,35(*,))
READ(5,901) (BB(I),I=1,MM)
FORMAT(10F8.4)
READ(5,902) (IEQ(I),I=1,MM)
FORMAT(40I2)
DO 910 J=1,N
READ(5,903) KR
READ(5,904) (I,AN(I,J),K=1,KR)
CONTINUE
READ(5,901)((CN(I,J),J=1,N),I=1,M)
FORMAT(I2)
FORMAT(8(I2,F8.4))
DO 1 J=1,N
IXN(J)=J
DO 1 I=1,M
CN(I,J)=-CN(I,J)
IF(WL.EQ.0..AND.WU.EQ.1.) GO TO 4
M=3
IW=1
DO 13 J=1,N
CN(3,J)=(1.-WU)*CN(1,J)+WU*CN(2,J)
K=0
N1=N
DO 10 I=1,MM
IF(IEQ(I).EQ.1) GO TO 10
IF(IEQ(I).EQ.0) GO TO 9
N1=N1+1
AN(I,N1-K)=-1.
IXN(N1-K)=N1
GO TO 10
N1=N1+1
K=K+1
IBE(I)=N1
CONTINUE
N2=N1
NMM=N1-K
NMM=# OF NONBASIC VARIABLES
DO 20 I=1,MM
IF(IEQ(I).EQ.0) GO TO 20
N1=N1+1
IBE(I)=N1
CONTINUE

```

```

ZP=0.
DO 25 J=1,MMM
CP(J)=0.
FORM PHASE1 OBJECTIVE
DO 30 I=1,MM
IF(IBE(I).LE.N2) GO TO 30
ZP=ZP-BB(I)
DO 30 J=1,MMM
CP(J)=CP(J)-AN(I,J)
CONTINUE
MINIMIZE PHASE1 OBJECTIVE
KE=0
DO 50 J=1,MMM
IF(CP(J).GE.-DEL) GO TO 50
KE=J
GO TO 55
CONTINUE
IF(ZP.GE.-DEL) GO TO 70
WRITE(6,805)
FORMAT(1H1,40X,,THERE IS NO FEASIBLE SOLUTION OF THE PROBLEM,/)
STOP
CALL LEAV(KE,KMIN)
IF(KMIN.EQ.0) WRITE(6,805)
IF(IBE(KMIN).LE.N2) GO TO 65
CALL PIVOT(KE,KMIN)
MMM=MMM-1
N1=N1-1
IF(KE.EQ.(MMM+1)) GO TO 40
DO 60 J=KE,MMM
IXN(J)=IXN(J+1)
CP(J)=CP(J+1)
DO 53 I=1,M
CN(I,J)=CN(I,J+1)
DO 53 I=1,MM
AN(I,J)=AN(I,J+1)
CONTINUE
GO TO 40
KL=IBE(KMIN)
CALL PIVOT(KE,KMIN)
IXN(KE)=KL
GO TO 40
PHASE1 OBJECTIVE MINIMIZED
IF(N1.EQ.0) GO TO 100
THERE ARE STILL ARTIFICIAL VAR. IN BASIS
K=1
DO 80 I=K,MM
IF(BB(I).GT.DEL) GO TO 80
IF(IBE(I).LE.N2) GO TO 80
GO TO 81
CONTINUE
DO 82 J=1,MMM
IF(ABS(AN(I,J)).GT.DEL) GO TO 85
CONTINUE
NULL EQUATION, DROP IT
MM=MM-1
N1=N1-1
IF(1.EQ.(MM+1)) GO TO 100

```

```

AN(I1,J)=AN(I1+1,J)
CONTINUE
IF(N1.EQ.0) GO TO 100
K=I
GO TO 79
ELIMINATE ARTIFICIAL VARIABLE
CALL PIVOT(J,I)
NMM=NMM-1
N1=N1-1
IF(J.EQ.(NMM+1)) GO TO 88
DO 86 JJ=J,NMM
IXN(JJ)=IXN(JJ+1)
DO 87 II=1,M
CN(II,JJ)=CN(II,JJ+1)
DO 89 II=1,MM
AN(II,JJ)=AN(II,JJ+1)
CONTINUE
IF(N1.EQ.0) GO TO 100
K=I+1
GO TO 79
AT THIS POINT WE HAVE INITIAL B.F.S.
IPHASE=0
KE=0
DO 110 J=1,NMM
IF(CN(M,J).GE.-DEL) GO TO 110
KE=J
CALL LEAV(KE,KMIN)
IF(KMIN.EQ.0) GO TO 150
KL=IBE(KMIN)
CALL PIVOT(KE,KMIN)
IXN(KE)=KL
GO TO 101
CONTINUE
GO TO 200
WRITE(6,806) M
FORMAT(1H1,50X,,OBJECTIVE,,12,, IS UNBOUNDED,/)
STOP
IF(I1.EQ.0) GO TO 202
M=M-1
GO TO 289
DO 203 J=1,NMM
IF(CN(2,J).GE.DEL) GO TO 203
IF(CN(1,J).GE.-DEL) GO TO 203
CALL LEAV(J,KMIN)
KL=IBE(KMIN)
CALL PIVOT(J,KMIN)
IXN(J)=KL
GO TO 202
CONTINUE
INITIAL EFFICIENT VERTEX FOUND
WRITE(6,800)
FORMAT(///,50X,,INITIAL EFFICIENT VERTEX,/,48X,26(*),/)
WRITE(6,802)(IBE(I),BR(I),I=1,MM)
FORMAT(3X,,BASIC VARIABLES,/,2X,15(*),/,(7( X,,J3,, =,F10.3)
*,/))
WRITE(6,803)(I,ZZ(I),I=1,M)
FORMAT(/,3X,,OBJECTIVES,/,2X,10(*),/,3X,,OBJ,,3(I1,, =,F10.3)

```



```

FORMAT(2X, 'NBSC VARS., 30I3)
IFLAG=0
J=# OF COMPUTED EFF. VERTICES, L1=# OF NOT YET COMPUTED EFF. VERTICES
L2=1
L1=0
DO 290 K=1, NMM
  IVEC(K)=IXN(K)
CALL SORT(IVEC, NMM)
DO 295 K=1, NMM
  IV2(1, K)=IVEC(K)
EFFICIENT EDGES WILL BE FOUND
IF(L2-100)301, 740, 740
WRITE(6, 745)
FORMAT(//, 10X, 'NO. OF EFFICIENT VERTICES IS GREATER THAN 200, //, 10
*X, 'RUN IS TERMINATED, )
STOP
CALL EDGE2
DETERMINE FEASIBLE ADJACENT SET
J=# OF EFFICIENT EDGES
RT=# OF ADJACENT EFF. EXTR. PTS.
LRT=0
IF(LJ.EQ.0) GO TO 700
DO 350 II=1, LJ
  J=JT(II)
CALL LEAV(J, KMIN)
LRT=LRT+1
KK=IBE(KMIN)
DO 320 I=1, NMM
  IRT(LRT, I)=IXN(I)
  IRT(LRT, J)=KK
SORT INDICES OF NBAS. VAR. OF EXTR. POINT
DO 330 I=1, NMM
  IVEC(I)=IRT(LRT, I)
CALL SORT(IVEC, NMM)
DO 99 I=1, NMM
  IRT(LRT, I)=IVEC(I)
CONTINUE
LR1=0
FORM THE SET IRT-IV2
DO 390 I=1, LRT
DO 370 II=1, L2
DO 360 K=1, NMM
IF(IRT(I, K).NE.IV2(II, K)) GO TO 370
CONTINUE
GO TO 390
CONTINUE
LR1=LR1+1
DO 380 K=1, NMM
  IRT(LR1, K)=IRT(I, K)
CONTINUE
IF(LR1.EQ.0) GO TO 700
LR2=0
FORM THE SET IRT-IV1
IF(L1)395, 395, 396
LR2=LR1
GO TO 435
DO 430 I=1, LR1
DO 410 II=1, L1
DO 400 K=1, NMM

```

```

GO TO 430
CONTINUE
LR2=LR2+1
DO 420 K=1,NMM
IRT(LR2,K)=IRT(I,K)
CONTINUE
IF(LR2.EQ.0) GO TO 600
NEW VERTEX SELECTED FROM R2
L2=L2+1
DO 500 K=1,NMM
NT(K)=IRT(LR2,K)
IV2(L2,K)=NT(K)
LR2=LR2-1
IF(LR2.EQ.0) GO TO 590
FORM THE SET IV11
DO 540 I=1,LR2
L1=L1+1
DO 540 K=1,NMM
IV1(L1,K)=IRT(I,K)
CALL MOVE(NT)
AT NEW EFF. VERTEX NEW ITERATION STARTS
GO TO 300
NEW VERTEX SELECTED FROM R1
L2=L2+1
DO 610 K=1,NMM
NT(K)=IRT(LR1,K)
IV2(L2,K)=NT(K)
DO 650 I=1,L1
DO 620 K=1,NMM
IF(NT(K).NE.IV1(I,K)) GO TO 650
CONTINUE
L1=L1-1
IF(I.EQ.L1+1) GO TO 690
DO 630 IJ=I,L1
DO 630 K=1,NMM
IV1(IJ,K)=IV1(IJ+1,K)
GO TO 690
CONTINUE
CALL MOVE(NT)
GO TO 300
NEW VERTEX SELECTED FROM V1
IF(L1.EQ.0) GO TO 9999
L2=L2+1
IFLAG=1
DO 710 K=1,NMM
NT(K)=IV1(L1,K)
IV2(L2,K)=NT(K)
L1=L1-1
CALL MOVE(NT)
IFLAG=0
GO TO 300
WRITE(6,777)
FORMAT(///,43X,'ALL EFFICIENT VERTICES HAVE BEEN ENUMERATED,')
WRITE(6,779) L2
FORMAT(///,43X,'NO EFFICIENT EXTREME POINTS = ',I3)
STOP
END

```

```
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
COMMON/GUL1/M,MM,NMM,DEL,BB(50),AN(50,100),CN(3,100),Z(3)
COMMON/GUL2/CP(100),ZP,IXH(100),IBE(50),IPHASE,IFLAG
COMMON/GUL3/LJ,JT(50),WU,WL
LJ=0
AMIN=1.0E+10
DO 10 J=1,NMM
IF(CN(2,J).LT.DEL) GO TO 10
IF(CN(1,J).GT.-DEL) GO TO 10
AA=CN(1,J)/CN(2,J)
CN(3,J)=AA
IF(AA.GE.AMIN) GO TO 10
AMIN=AA
CONTINUE
WGHT=AMIN/(AMIN+1.)
WGHT1=1.-WGHT
WRITE(6,850) WGHT1,WGHT
FORMAT(/,30X,/,CORRESPONDING OBJECTIVE WEIGHTS ARE :/,5F6.3)
IF(WGHT.GT.WU.OR,WGHT.LT,WL) RETURN
DO 30 J=1,NMM
IF(CN(2,J).LT.DEL) GO TO 30
IF(CN(1,J).GT.-DEL) GO TO 30
IF(CN(3,J).GT.AMIN+DEL) GO TO 30
LJ=LJ+1
JT(LJ)=J
CONTINUE
RETURN
END
```

```

IMPLICIT DOUBLE PRECISION(A-H,O-Z)
DIMENSION INT1(50)
COMMON/GUL1/MM,MM,NMM,DEL,IB(50),AN(50,100),CN(3,100),Z(3)
COMMON/GUL2/CP(100),ZP,IXN(100),IBE(50),IPHASE,IFLAG
IFLAG=0 IF MOVE IS TO ADJACENT VERTEX ,=1 OTHERWISE
DO 1 I=1,NMM
DO 2 J=1,NMM
IF(IXN(I).EQ.INT1(J)) GO TO 1
CONTINUE
KE=I
IF(IFLAG.EQ.0) GO TO 500
E,TH NONBASIC VAR, WILL BE ENTERING
DO 10 IL=1,NMM
DO 11 J=1,NMM
IF(INT1(IL).EQ.IXN(J)) GO TO 10
CONTINUE
KL=INT1(IL)
KMIN=J
DO 13 I=1,MM
IF(KL.NE.IBE(I)) GO TO 13
KMIN=I
IF(ABS(AN(KMIN,KE)).GT.DEL) GO TO 502
GO TO 10
CONTINUE
CONTINUE
CALL PIVOT(KE,KMIN)
IXN(KE)=KL
CONTINUE
GO TO 900
CALL LEAV(KE,KMIN)
KL=IBE(KMIN)
CALL PIVOT(KE,KMIN)
IXN(KE)=KL
WRITE(6,801)
FORMAT(///,55X,'NEW EFFICIENT VERTEX',//,53X,24(*),//)
WRITE(6,802)(IBE(I),BB(I),I=1,MM)
FORMAT(3X,'BASIC VARIABLES',//,2X,15(*),//,(7(I=1,MM) X,J3,=,F10.3)
*,//)
WRITE(6,803)(I,ZZ(I),I=1,MM)
FORMAT(//,3X,'OBJECTIVES',//,2X,10(*),//,3X,'OBJ',3(I=1,MM) X,J3,=,F10.3,
*2X,'OBJ,')
RETURN
END

```

```
COMMON/GUL1/M,MM,NMM,DEL,BB(50),AN(50,100),CN(3,100),ZZ(3)
COMMON/GUL2/CP(100),ZP,IXN(100),IRE(50),IPHASE,IFLAG
IBF(KMIN)=IXN(KE)
AA=AN(KMIN,KE)
)ORMALIZE
DO 14 J=1,NMM
AN(KMIN,J)=AN(KMIN,J)/AA
BB(KMIN)=BB(KMIN)/AA
AN(KMIN,KE)=1./AA
)IMINATE ENTERING VAR. FROM OTHER EQNS.
DO 15 I=1,MM
IF(I.EQ.KMIN) GO TO 15
AA=AN(I,KE)
BB(I)=BB(I)-AA*BB(KMIN)
DO 16 J=1,NMM
AN(I,J)=AN(I,J)-AA*AN(KMIN,J)
AN(I,KE)=-AN(KMIN,KE)*AA
CONTINUE
DO 17 I=1,M
AA=CN(I,KE)
ZZ(I)=ZZ(I)-AA*BB(KMIN)
DO 18 J=1,NMM
CN(I,J)=CN(I,J)-AN(KMIN,J)*AA
CN(I,KE)=-AN(KMIN,KE)*AA
CONTINUE
IF(IPHASE.NE.1) GO TO 10
AA=CP(KE)
ZP=ZP-AA*BB(KMIN)
DO 9 I=1,NMM
CP(J)=CP(J)-AN(KMIN,J)*AA
CP(KE)=-AN(KMIN,KE)*AA
CONTINUE
RETURN
END
```

```
SUBROUTINE LEAV(J,KMIN)
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
COMMON/GUL1/M,MM,NMM,DEL,BB(50),AM(50,100),CN(3,100),Z(3)
AMIN=.9E+10
KMIN=0
DO 5 I=1,MM
IF(AM(I,J).LE.DEL) GO TO 5
AA=BB(I)/AM(I,J)
IF(AA.GE.AMIN)GO TO 5
AMIN=AA
KMIN=I
CONTINUE
RETURN
END
```

```
COMPILE SORT (IV,N)  
DIMENSION IV(50)  
/ IS THE VECTOR TO BE SORTED  
/ IS THE DIMENSION OF THE VECTOR  
NN=N-1  
DO 1 I=1,NN  
  II=I+1  
  DO 1 J=II,N  
    IF(IV(I).LE.IV(J)) GO TO 1  
    ITEMP=IV(I)  
    IV(I)=IV(J)  
    IV(J)=ITEMP  
  CONTINUE  
  RETURN  
END
```

NEW EFFICIENT VERTEX

.....

BASIC VARIABLES

.....

X006 =	17.871	X059 =	1.452	X043 =	8.683	X035 =	31.144	X021 =	.726	X054 =	.063	X019 =	.181
X033 =	22.174	X056 =	7.819	X034 =	24.616	X001 =	12.129	X002 =	36.387	X025 =	9.132	X004 =	100.000
X005 =	130.000	X042 =	3.457	X007 =	13.613	X008 =	.716	X009 =	28.100	X010 =	28.100	X011 =	4.380
X012 =	21.900	X013 =	61.900	X016 =	4.978	X036 =	16.614	X063 =	36.530	X023 =	27.022	X017 =	9.893
X055 =	19.767	X003 =	84.904	X020 =	.363	X044 =	23.514	X015 =	16.000	X040 =	.000	X024 =	4.000
X062 =	6.868	X014 =	8.000	X									

OBJECTIVES

.....

OBJ1 = -1080.219 OBJ2 = -1100.376 OBJ

CORRESPONDING OBJECTIVE WEIGHTS ARE : .311 .689

APPENDIX III: DATA FOR THE POWER SYSTEMS EXPANSION MODEL AS APPLIED TO TURKEY

Dates corresponding to the periods:

Period	:	0	1	2	3	4	5
Mid-Period Year	:	1977	1983	1989	1995	2001	2007

Beginning capacities:

$$P_{1,0} = 4 \text{ GW}$$

$$P_{2,0} = 4 \text{ GW}$$

$$P_{3,2} = 1 \text{ GW}$$

Demand data:

Period	Energy Demand, ED_t (Twh)		Power Demand, PD_t (Gw)	
	Scenarios A,B	Scenarios C,D	Scenarios A,B	Scenarios C,D
1	30	30	6.21	6.21
2	60	50	12.44	10.37
3	108	90	22.37	18.63
4	170	150	35.25	31.13
5	240	220	49.77	45.66

Power plant data:

Plant type	Initial Cost(α) (in TL/w)	Operating Cost(α') (in TL/kwh)	Capacity Factor f_i
Coal	34	1.0	0.70
Hydro	40	-	0.45
Nuclear	65	0.5	0.60

Capacity expansion coefficients, K_i :

2.0 for coal and hydro,

1.5 for nuclear (restricted)

4.0 for nuclear (less-restricted)

Discount factor: 10 % p.a.

APPENDIX IV: COMPUTER PROGRAM FOR MULTIOBJECTIVE ZERO-ONE LINEAR PROGRAMMING

A.IV.1. DEFINITION OF VARIABLES USED IN THE PROGRAM

- K : Number of objectives
- N : Number of variables
- M : Number of constraints
- A(.,.) : Constraint coefficient matrix
- C(.,.) : Cost coefficient matrix
- S(.) : Array giving the current right hand side entries of the constraints
- T(.) : Array giving the sum of negative coefficients of each constraint
- DOM(.) : Array giving the relevant domination margins of the free variables
- UB(.) : The upper bound array
- KR : Index giving the constraint number where $T(KR)=S(KR)$, if any. (implying definite assignments.)
- JF : Index taking values in subroutine FEAS; 1 if the current node is found to be infeasible ; 2 if feasible and 0 otherwise
- IFREE : Number of free (unassigned) variables
- LZ : Number of variables that have been assigned the value zero
- J1 : Index taking value 1 if definite zero assignments are determined in subroutine ADVDM, zero otherwise
- IW(.) : Array specifying if a variable is free ($IW(J)=0$); assigned value zero ($IW(J)=10$); assigned value one ($IW(J)=11$); or assigned definitely the value one ($IW(J)=12$).
- JB : Index of the branching variable
- INF(.) : Array giving the infeasibilities of the free variables

MINF : The minimum entry of the above array
 LB(., J) : The Jth lower bound array
 NL : Number of lower bounds
 IND(.,.) : Matrix giving the indices of the variables when
 reordered in ascending order of constraint coefficients
 INC(.,.) : Matrix giving the indices of the variables when reordered
 in descending order of cost coefficients
 JD : Index taking value 1 if current node is found to be
 dominated in subroutine ADVDM
 LM : Minimum number of variables that must be set to one
 in order to reach a feasible solution
 NA : Total number of alternative solutions, if any
 LBN(I) : The lower bound number corresponding to Ith
 alternative solution
 NUB(.) : Array giving the relevant advanced upper bound vector
 IPATH(.) : Array specifying the current enumeration path in the
 order in which variables have been assigned
 IS(.) : Array used for sorting operations
 IQ(J) : Index taking value 1 if Jth variable has a positive
 cost coefficient in any objective, 0 otherwise
 IX(.,I) : The solution vector corresponding to Ith lower bound
 IXA(.,I) : The solution vector corresponding to Ith alternative
 solution
 KBRN : Index taking value 1 if branching is done according
 to minimum infeasibility; value 0 if done according
 to minimum domination margin.

A.IV.2. INPUT INSTRUCTIONS

The inputs to the program consist of the parameters K, N and M, the matrices C and A and the array S.

The parameters K, N and M are inputted on a single card according to the format (10I8).

The matrices C and A are inputted columnwise, respectively, again according to the format (10I8). The right hand side array S is also inputted according to the same format.

Since in actual problems the cost coefficients are mostly positive, the program assumes positive cost coefficient inputs. Later, through the transformation $x'_j = 1-x_j$, a cost matrix consisting basically of negative entries is obtained.

A.IV.3. PROGRAM LISTING AND SAMPLE OUTPUT

The listing of the main program and the subroutines; and a sample output is given below.

```

INTEGER A,C,S,T,DOM,UB,SUM
DIMENSION IPATH(70),IX(70,150),IS(70),IQ(70)
COMMON/BIN1/KR,JF,M,S(50),T(50),IFREE,J1
COMMON/BIN2/N,IW(70),A(50,70),JB,MINF,INF(70)
COMMON/BIN3/K,UB(5),C(5,70),LB(5,150),NL,IND(50,70),INC(5,70),JD,L
*M,NA,LBN(10),IXA(70,10),NUB(5),DOM(70)
READ(5,700) K,N,M
10 FORMAT(10I8)
READ(5,700)((C(I,J),I=1,K),J=1,N)
READ(5,700)((A(I,J),I=1,M),J=1,N)
READ(5,700)(S(I),I=1,M)
NIP=N/6+1
MAKE TRANSFORMATION 1-X
DO2 I=1,K
SUM=0
DO 1 J=1,N
IF(C(I,J).LE.0)GO TO 1
SUM=SUM+C(I,J)
1 C(I,J)=-C(I,J)
UB(I)=SUM
2 CONTINUE
DO 4 I=1,M
SUM=0
DO 3 J=1,N
SUM=SUM+A(I,J)
3 A(I,J)=-A(I,J)
S(I)=S(I)-SUM
4 CONTINUE
DO 6 J=1,N
DO 5 I=1,K
IF(C(I,J).LE.0)GO TO 5
IQ(J)=1
GO TO 6
5 CONTINUE
IQ(J)=0
6 CONTINUE
NL=0
ITER=0
NIF=0
NAOV=0
JADV=0
NA=0
SORT INDECES IN ASCENDING ORDER OF COEFFICIENTS
DO 10 I=1,M
DO 8 J=1,N
IS(J)=A(I,J)
8 IND(I,J)=J
NN=N-1
DO 15 JJ=1,NN
II=JJ+1
DO 15 J=II,N
IF(IS(JJ).LE.IS(J)) GO TO 15
ITEMP=IND(I,JJ)
IND(I,JJ)=IND(I,J)
IND(I,J)=ITEMP
ITEMP=IS(JJ)

```

7 SORT INDECS IN DESCENDING ORDER OF COSTS

```

DO 22 I=1,K
DO 17 J=1,N
IS(J)=C(I,J)
7 INC(I,J)=J
DO 20 JJ=1,NN
II=JJ+1
DO 20 J=II,N
IF (IS(JJ).GE.IS(J)) GO TO 20
ITEMP=INC(I, JJ)
INC(I, JJ)=INC(I, J)
INC(I, J)=ITEMP
ITEMP=IS(JJ)
IS(JJ)=IS(J)
IS(J)=ITEMP
0 CONTINUE
2 CONTINUE
INITIALIZE
IFREE=N
DO 25 J=1,N
5 IW(J)=0
DO 35 I=1,M
DO 35 J=1,N
IF (A(I, J)) 30, 35, 35
0 T(I)=T(I)+A(I, J)
5 CONTINUE
GO TO 56
FEASIBILITY CHECK
0 CALL FEAS
ITER=ITER+1
IF (JF-1) 45, 400, 450
5 IF (KR) 50, 50, 150
CHOOSE BRANCING VARIABLE
0 IF (NL.EQ.0) GO TO 56
KBRN=1
1 IF (LZ-NIP) 56, 52, 52
2 IF (2*LZ-N+IFREE) 56, 1152, 1152
6 CALL BRANF
GO TO 94
2 KBRN=0
CALL ADVDM
NADV=NADV+1
IF (JD.EQ.1.OR, JB.EQ.0) JADV=JADV+1
IF (JD) 54, 54, 400
+ IF (LM.EQ.IFREET) GO TO 550
IF (JB) 400, 400, 60
0 IF (J1.EQ.0) GO TO 94
DO 65 I=1, M
5 IS(I)=T(I)
DO 69 J=1, N
IF (IW(J)) 64, 64, 69
+ IF (DOM(J)) 66, 66, 69
5 DO 68 I=1, M
IF (A(I, J)) 67, 68, 68
7 T(I)=T(I)-A(I, J)
3 CONTINUE
9 CONTINUE

```

```

IF (I=J) 70,70,80
1 IF (D0(J)) 75,75,80
5 IW(J)=10
IV=IV+1
IPATH(IV)=J
LZ=LZ+1
IFREE=IFREE-1
DO 76 I=1,K
IF (C(I,J).LE.0)GO TO 76
UB(I)=UB(I)-C(I,J)
5 CONTINUE
) CONTINUE
DO 77 J=1,NL
DO 78 I=1,K
MARG=UB(I)-LB(I,J)
IF (MARG.GT.0)GO TO 77
3 CONTINUE
GO TO 400
7 CONTINUE
IF (KR)94,94,150
BRANCH TO JB=1
+ IV=IV+1
IPATH(IV)=JB
IFREE=IFREE-1
IW(JB)=11
DO 100 I=1,K
IF (C(I,JB).GE.0)GO TO 100
UB(I)=UB(I)+C(I,JB)
) CONTINUE
DO 110 I=1,M
S(I)=S(I)-A(I,JB)
IF (A(I,JB)) 105,110,110
5 T(I)=T(I)-A(I,JB)
) CONTINUE
IF (KBRN)500,500,610
2 DO 95 I=1,M
5 T(I)=TS(I)
JADV=JADV+1
GO TO 400
1 J=IPATH(IV)
JDS=0
IW(J)=10
LZ=LZ+1
DO 120 I=1,K
) UB(I)=UB(I)-C(I,J)
DO 121 I=1,M
1 S(I)=S(I)+A(I,J)
CALL FEAS
ITER=ITER+1
IF (JF-1) 122,400,450
2 IF (KR)123,123,150
3 MIN=9999999
JB=0
8 DO 140 J=1,N
IF (IW(J))135,135,140
5 IF (INF(J).GE.MIN)GO TO 140
MIN=INF(J)
JB=J

```



```

IF(MIN)612,612,613
2 JDS=1
GO TO 450
3 IF(NL)616,616,614
4 DO 620 J=1,NL
DO 615 I=1,K
MARG=UB(I)-LB(I,J)
IF(MARG)615,615,620
5 CONTINUE
GO TO 111
0 CONTINUE
6 CALL FEAS
ITER=ITER+1
IF(JF-1)45,111,111
T(I)=S(I) IN ROW KR IMPLIES DEFINITE ASSIGNMENTS
0 DO 250 J=1,N
IF(IW(J)) 160,160,250
0 IF(A(KR,J)) 170,250,190
0 IW(J)=12
IV=IV+1
IPATH(IV)=J
IFREE=IFREE-1
DO 175 I=1,K
IF(C(I,J).GE.0)GO TO 175
UB(I)=UB(I)+C(I,J)
75 CONTINUE
DO 180 I=1,M
S(I)=S(I)-A(I,J)
IF(A(I,J)) 176,180,180
76 T(I)=T(I)-A(I,J)
80 CONTINUE
GO TO 250
90 IW(J)=10
IV=IV+1
IPATH(IV)=J
IFREE=IFREE-1
LZ=LZ+1
DO 194 I=1,K
IF(C(I,J).LE.0)GO TO 194
UB(I)=UB(I)-C(I,J)
94 CONTINUE
DO 200 I=1,M
IF(A(I,J)) 195,200,200
95 T(I)=T(I)-A(I,J)
00 CONTINUE
60 CONTINUE
GO TO 500
FEASIBLE CHECK FOR DOM AND BACKTRACK
60 DO 304 I=1,K
4 NUB(I)=UB(I)
DO 306 J=1,N
IF(IW(J).NE.0)GO TO 306
DO 305 I=1,K
IF(C(I,J).LE.0)GO TO 305
NUB(I)=NUB(I)-C(I,J)
95 CONTINUE
96 CONTINUE
IF(NL)452,452,451

```

```

DO 320 I=1,K
MARG=NUB(I)-LB(I,J)
IF(MARG) 320,310,315
0 KK=KK+1
GO TO 320
5 KD=KD+1
0 CONTINUE
IF(KK.EQ.K) GO TO 350
ALTERATIVE SOLUTION
IF((KK+KD).EQ.K) GO TO 375
DOMINATES A L.B.
IF(KD.NE.0) GO TO 300
GO TO 499
0 CONTINUE
NEW LOWER BOUND
2 NIE=ITER
NL=NL+1
DO 330 I=1,K
0 LB(I,NL)=NUB(I)
DO 340 J=1,N
IF(IW(J)-10) 342,342,341
1 IX(J,NL)=0
GO TO 340
2 IX(J,NL)=1
0 CONTINUE
GO TO 499
AN ALTERNATIVE SOLUTION FOR JTH LOWER BOUND
50 DO 355 JJ=1,N
IF(IW(JJ)-10) 352,352,351
52 IF(IX(JJ,J).NE.1) GO TO 353
GO TO 355
51 IF(IX(JJ,J).NE.0) GO TO 353
55 CONTINUE
GO TO 499
53 NA=NA+1
LBN(NA)=J
DO 360 J=1,N
IF(IW(J)-10) 362,362,361
51 IXA(J,NA)=0
GO TO 360
52 IXA(J,NA)=1
60 CONTINUE
GO TO 499
JTH LOWER BOUND DOMINATED
75 DO 380 I=1,K
80 LB(I,J)=NUB(I)
DO 390 JJ=1,N
IF(IW(JJ)-10) 392,392,391
91 IX(JJ,J)=0
GO TO 390
92 IX(JJ,J)=1
90 CONTINUE
COMPARE NEW LB J WITH THE FOLLOWING LBS FOR DOMINANCE
NN=NL
DO 495 JJ=NN,J,-1
IF(JJ.EQ.J) GO TO 490
DO 485 I=1,K
MARG=LB(I,J)-LB(I,JJ)

```

```

IF (JJ.EQ.NL) GO TO 488
REPLACE LB JJ WITH LB NL
DO 486 J=1,K
86 LB(I,JJ)=LB(I,NL)
DO 487 JI=1,N
87 IX(JI,JJ)=IX(JI,NL)
88 NL=NL-1
90 IF (NA.EQ.0) GO TO 495
NB=NA
DO 395 I=NB,1,-1
IF (LBN(I).NE.JJ) GO TO 395
IF (I.EQ.NA) GO TO 394
LBN(I)=LBN(NA)
DO 397 JI=1,N
97 IXA(JI,I)=IXA(JI,NA)
94 NA=NA-1
95 CONTINUE
95 CONTINUE
99 DO 497 J=1,N
IF (IW(J).NE.0) GO TO 497
IF (IQ(J).EQ.0) GO TO 497
JB=J
KBRN=1
JDS=0
GO TO 94
97 CONTINUE
IF (JDS) 400,400,111
BACKTRACKING
400 JB=IPATH(IV)
IB=IB(JB)-10
IFREE=IFREE+1
IW(JB)=0
IV=IV-1
IF (IB.EQ.0) GO TO 430
DO 410 I=1,K
IF (C(I,JB).GE.0) GO TO 410
UB(I)=UB(I)-C(I,JB)
+10 CONTINUE
DO 420 I=1,M
+20 S(I)=S(I)+A(I,JB)
GO TO 437
+30 DO 435 I=1,K
IF (C(I,JB).LE.0) GO TO 435
UB(I)=UB(I)+C(I,JB)
+35 CONTINUE
+37 DO 470 I=1,M
IF (A(I,JB)) 440,470,470
+40 T(I)=T(I)+A(I,JB)
+70 CONTINUE
IF (IV.EQ.0.AND.IB.NE.1) GO TO 999
IF (IB=1) 471,472,400
+71 LZ=LZ-1
GO TO 400
BRANCH TO JB=0
+72 IW(JB)=10
LZ=LZ+1
IV=IV+1
IPATH(IV)=JB

```

```

35 T(I)=T(I)-A(I,JB)
50 CONTINUE
   DO 465 I=1,K
   IF(C(I,JB).LE.0)GO TO 465
   UB(I)=UB(I)-C(I,JB)
55 CONTINUE
   GO TO 500
50 DO 575 J=1,N
   IF(I/(J))560,560,575
60 IV=IV+1
   IPATH(IV)=J
   IW(J)=12
   DO 565 I=1,M
65 S(I)=S(I)-A(I,J)
   DO 570 I=1,K
   IF(C(I,J).GE.0)GO TO 570
   UB(I)=UB(I)+C(I,J)
70 CONTINUE
75 CONTINUE
   IFREE=0
   DO 580 I=1,M
80 T(I)=0
   GO TO 500
   ENUMERATION IS COMPLETE
999 WRITE(6,1883) K,N,M
863 FORMAT(1H1,10X,,SOLUTIONS FOR THE MULTIOBJECTIVE ZERO-ONE PROBLEM
*,//,11X,,WITH ,,I1,, OBJECTIVES ,,I2,, VARIABLES ,,I2,, CONSTRAINT
*,//,11X,50(,*))
   IF(NL.NE.0)GO TO 1885
   WRITE(6,1884)ITER
884 FORMAT(///,11X,,ENUMERATION WAS COMPLETED IN,,I6,, ITERATIONS,///
*,HE PROBLEM HAS NO FEASIBLE SOLUTION,)
   STOP
885 WRITE(6,800)ITER,NL,NIE
900 FORMAT(///,11X,,ENUMERATION WAS COMPLETED IN ,,I6,, ITERATIONS,,/
*,11X,74,, EFFICIENT SOLUTIONS WERE FOUND IN,,I6,, ITERATIONS,)
   DO 600 JJ=1,NL
   WRITE(6,801)(I,LB(I,JJ),I=1,K)
901 FORMAT(//,10X,5(,OBJ,,I1,, = ,,I7,5X))
   WRITE(6,802)(J,IX(J,JJ),J=1,N)
902 FORMAT(/,8X,15(, X,,I2,,=,,I1))
   IF(NA.EQ.0) GO TO 600
   DO 650 L=1,NA
   IF(LBN(L).NE.0) GO TO 650
   WRITE(6,802)(J,IXA(J,L),J=1,N)
950 CONTINUE
960 CONTINUE
   WRITE(6,806) NADV,JADV
986 FORMAT(///,10X,,ADVDOM WAS CALLED,,I4,, TIMES AND DOMINATION WAS
*ONCLUDED,,I4,, TIMES,,)
   STOP
   END

```

```

COMMON/BIN1/KR,JF,M,S(50),T(50)
COMMON/BIN2/N,IW(70),A(50,70),JB,MINF,INF(70)
COMMON/BIN3/K,UB(5),C(5,70),LB(5,150),NL,IND(50,70),INC(5,70),JD,
* M,UA, LBN(10),IXA(70,10),NUB(5),DOM(70)
DIMENSION IDT(5),NNB(5)
DOM. CHECK CONSIDERING MIN. # OF VAR. THAT MUST BE 1 FOR FEAS.
J1=0
JD=0
LM=0
DO 50 I=1,M
IF(S(I)) 10,50,50
10 L=0
SUM=0
DO 40 J=1,N
JJ=IN(I,J)
IF(IW(JJ)) 20,20,40
20 SUM=SUM+A(T,JJ)
L=L+1
IF(SUM-S(I)) 25,25,40
25 IF(LM-L) 30,50,50
30 LM=L
GO TO 50
40 CONTINUE
50 CONTINUE
LM IS MINIMUM # OF VARS. THAT MUST BE 1 FOR FEAS.
IF(LM.EQ.1)RETURN
DO 45 I=1,K
IDT(I)=UB(I)
45 NUB(I)=UB(I)
IF(LM.EQ.1)GO TO 81
DO 60 I=1,K
L=0
DO 55 J=1,N
JJ=INC(I,J)
IF(IW(JJ)) 51,51,55
51 IF(C(T,JJ))52,54,54
52 NUB(I)=NUB(I)+C(I,JJ)
54 L=L+1
IF(L-LM+1)55,53,60
53 IDT(I)=NUB(I)
55 CONTINUE
60 CONTINUE
DO 80 J=1,NL
DO 70 I=1,K
MARG=NUB(I)-LB(I,J)
IF(MARG) 70,70,80
70 CONTINUE
JD=1
RETURN
80 CONTINUE
81 DO 90 J=1,N
IF(IW(J))85,85,90
85 DO 86 I=1,K
NNB(I)=IDT(I)+C(I,J)
IF(NNB(I).GT.NUB(I))NNB(I)=NUB(I)
86 CONTINUE
DOM(J)=999999
DO 100 JJ=1,NL

```

```
MARG=INB(I)-LB(I, JJ)
IF (MARG) 95, 97, 96
06 IDOM=IDOM+MARG
IF (IDOM-DOM(J)) 95, 100, 100
07 KK=KK+1
05 CONTINUE
DOM(J)=IDOM
IF (DOM(J).NE.0) GO TO 100
IF (KK-K) 101, 102, 101
01 J1=1
GO TO 90
02 DOM(J)=1
GO TO 90
00 CONTINUE
00 CONTINUE
01 MIN=999999
JB=0
DO 120 J=1, N
IF (IW(J)) 110, 110, 120
10 IF (DOM(J)-MIN) 115, 120, 120
15 IF (DOM(J)) 120, 120, 116
16 MIN=DOM(J)
JB=J
20 CONTINUE
RETURN
END
```

```
INTEGER A,S  
COMMON/BIN1/KR,JF,M,S(50),T(50)  
COMMON/BIN2/N,IW(70),A(50,70),JB,MINF,INF(70)  
CHOSES BRANCHING VARIABLE ACCORDING TO MIN INFEASIBILITY  
MINF=999999  
DO 20 J=1,M  
IF(IW(J))1,1,20  
1 DO 15 I=1,M  
IF(S(I))2,15,15  
2 IF(A(I,J))3,15,15  
3 INF(J)=0  
DO 10 II=1,M  
ITEMP=A(II,J)-S(II)  
IF(ITEMP)10,10,4  
4 INF(J)=INF(J)+ITEMP  
10 CONTINUE  
IF(INF(J)-MINF)11,20,20  
11 MINF=INF(J)  
JB=J  
GO TO 20  
15 CONTINUE  
20 CONTINUE  
RETURN  
END
```

```
SUBROUTINE FEAS
INTEGER S,T
COMMON/BI/IL/KR,JF,M,S(50),T(50)
KR=0
JF=2
DO 30 I=1,M
IF(S(I)) 20,30,30
20 JF=0
TMS=T(I)-S(I)
IF(TMS) 30,24,22
22 JF=1
GO TO 50
24 KR=I
30 CONTINUE
50 RETURN
END
```


SOLUTIONS FOR THE MULTI-OBJECTIVE ZERO-ONE PROBLEM
 WITH 2 OBJECTIVES 12 VARIABLES 10 CONSTRAINTS

ENUMERATION WAS COMPLETED IN 145 ITERATIONS

7 EFFICIENT SOLUTIONS WERE FOUND IN 53 ITERATIONS

OBJ1 = 234 OBJ2 = 280 OBJ
 X 1=0 X 2=0 X 3=1 X 4=0 X 5=1 X 6=1 X 7=1 X 8=1 X 9=1 X10=0 X11=0 X12=1 X

OBJ1 = 236 OBJ2 = 325 OBJ
 X 1=0 X 2=1 X 3=1 X 4=0 X 5=1 X 6=1 X 7=1 X 8=0 X 9=1 X10=0 X11=0 X12=1 X

OBJ1 = 303 OBJ2 = 275 OBJ
 X 1=0 X 2=1 X 3=0 X 4=0 X 5=1 X 6=1 X 7=1 X 8=1 X 9=1 X10=0 X11=0 X12=1 X

OBJ1 = 314 OBJ2 = 270 OBJ
 X 1=0 X 2=1 X 3=1 X 4=1 X 5=0 X 6=1 X 7=1 X 8=1 X 9=1 X10=0 X11=0 X12=0 X

OBJ1 = 315 OBJ2 = 259 OBJ
 X 1=0 X 2=1 X 3=1 X 4=0 X 5=1 X 6=1 X 7=1 X 8=1 X 9=1 X10=0 X11=0 X12=0 X

OBJ1 = 256 OBJ2 = 284 OBJ
 X 1=0 X 2=1 X 3=0 X 4=1 X 5=1 X 6=1 X 7=1 X 8=0 X 9=0 X10=0 X11=0 X12=0 X

OBJ1 = 247 OBJ2 = 289 OBJ
 X 1=0 X 2=0 X 3=1 X 4=1 X 5=1 X 6=1 X 7=1 X 8=0 X 9=0 X10=0 X11=0 X12=0 X

ADJDOM WAS CALLED 57 TIMES AND DOMINATION WAS CONCLUDED 23 TIMES.

CPU TIME = 1040 MILLISECOND