IMPLEMENTATION AND MULTIPLE DIMENSIONAL EXTENSION TO RECTANGLE BLIMINAi'iurt METHODS FUK BIOBJECTIVE DECISION MAKING

Dy

Costas Spanos

DRC-01-09-82

April, 1982

IMPLEMENTATION AND MULTIPLE DIMENSIONAL EXTENSION TO RECTANGLE ELIMINATION METHODS FOR BIOBJECTIVE DECISION MAKING[1]

Costas J. Spanos

Department of Electrical Engineering
Carnegie-Mellon University
Pittsburgh, PA 15213

The *Rectangle Elimination Methods* presented in [Polak 80], were expanded for multiple dimensional use, and implemented in an APL package for graphics video terminal. The package was also interfaced to a numerical algorithm for experimental use.

---

[1] Submitted in partial fulfillment of the requirements for the degree of Masters Of Science.
This work was supported in part by the National Science Foundation under Grant ENG 78-25755.

2. TOVS yoviiS pop Ttavvii tat Ma p i a

To my parents John and Mary

ACKNOWLEDGEMENTS

I wish to express my sincere thanks to my advisor Steve Director for his invaluable help, patience and encouragement during all the stages of the preparation of this work.

I also wish to thank all persons that helped me in various ways: Luis Vidigal for providing the existing APL software to support the example and the numerical package, Tariq Samad for providing the documented APL version of the *Han - Powell* algorithm, and Greg Jordan for his discussions about the circuit simulator problem.

Finally, I want to thank the members of my thesis committee, M. Callaham and S. Talukdar for their criticisms and propositions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Today's electrical engineering design, especially in the IC area, is characterized by a high number of conflicting objectives and an even higher number of designable parameters. The designer has to face the sometimes too burdening task, of performing multidimensional trade-offs in order to locate a feasible combination of input parameters that corresponds to a nominal combination of objective values. During the last decade, a great deal of research has been devoted to this area. This research deals with circuit simulation techniques, multiple optimization methods and a plethora of interactive algorithms.

In this work we describe the implementation and applications of an interactive algorithm for biobjective decision making. We also propose an extension for higher dimensions. The interaction scheme was originally presented in [Poiak 80].The material presented here is organized as follows:

The first chapter contains a rather general overview of the mathematical formulation of the problem. The multiple objective optimization concept is established and various solution approaches are discussed. The chapter closes with an overview of the classification of interactive CAD algorithms.

The second chapter is devoted to the presentation of the Rectangle Representation and Rectangle Elimination concepts. A general interactive algorithm for biobjective decision is presented. This algorithm is based on some optimality criteria also to be discussed in this chapter.

The third chapter describes the implementation of the algorithm and the various problems encountered. The necessary supporting software packages are described and an example of a MOSFET NAND gate design is given.

The fourth chapter discusses the possibility of expanding the algorithm for

multidimensional use. Implementation problems are discussed and the same example is solved for a three dimensional demonstration.

The fifth chapter contains the conclusions from the use of the algorithm. Future work concerning implementation improvements and expansion possibilities is also discussed here.

Costas J. Spanos

CHAPTER 1
GENERAL BACKGROUND

In this chapter we present a brief overview of the Multiple Objective Optimization problem. We give the definition, the basic theory and the various numerical approaches for the solution of this problem. At the end of this chapter we interpret existing Computer Aided Design algorithms from the point-of-view of *Decision Maker*[2]*-Anatyst*[3] interaction. This part may be skipped without loss of continuity, for the reader familiar with this material.

1.1 The Multiple Objective Decision Making Problem

A *Multiple Objective Decision Making problem* arises when the *Decision Maker* is trying to simultaneously optimize several objective functions. This is often the case in electrical circuit design. Specifically, a designer is trying to choose values for a set of designable parameters, and meet a set of physical and constructive constraints which are imposed on these parameters and on the set of circuit outputs.

The general formulation of the problem can take the form:

$$\max \; [f_1(p), f_2(p), \ldots, f_n(p)] \qquad (1.1)$$
$$\text{subject to } g_i(p) \leq 0, \; i = 1, \ldots, r$$

We employ the term *max* to mean either maximize or minimize. The goal of the *Decision Maker* here, is to find those input parameter combinations that

[2] The DECISION MAKER is the person who mikes the trade-off decisions during the engineering design process

[3] The ANALYST is the person responsible for the mathematical solution of the engineering problem. In this work is the computerized algorithm which performs this task

will optimize these objective functions. Typically the objective functions are competing, meaning that the absolute optimum for one of them may correspond to a poor value for the others. Thus the notion of trade-off is introduced in the procedure, meaning that the designer will have to compromise for a less satisfactory value for some of the functions in order to obtain a better response for the rest.

The problem as presented here can easily be seen as consisting of two major parts or subproblems. The first part is imposed by the volume of numerical calculations that are needed for the analysis even for a rather modest circuit. The mathematical formulation and the numerical philosophy in circuit analysis and nonlinear programming in general, is the subject of this part.

The second part has to do with an inherent weak point in *Multiple Objective Decision Making* problems. This point is that very often the various objective functions belong to a quite diverse spectrum of interpretation (for example we may have in the same problem objectives expressed in terms of voltages, currents, times, gains, noise figures and even cost), which cannot be compared in a strict mathematical manner. So it is of great importance that the engineering judgment of the *Decision Maker* will have to take place during the trade-off procedure and of course the algorithm must permit this involvement.

The last part of the *Multiple Objective Decision Making* aspect is about a classification of the various algorithms from the point-of-view of the amount of interaction that they permit between them and the *Decision Maker*

1.2 Multiple Objective Optimization

We begin with a brief survey of the mathematical background in the areas of Multiple Objective Optimization and Constrained Optimization. Consider the general unconstrained multiple objective optimization problem:

$$\min f_i(p) \quad i = 1,\ldots,n \tag{1.2}$$

where p is the m-dimensional vector of design parameters. One straightforward way to solve (1.2), is to combine all of the objective functions into a single

objective function, and then use classical techniques for optimizing this function. The simplest way to do this is the *weighted sum approach:*

$$F(p) = \sum_i w_i f_i(p) \qquad (1.3)$$

where the weight $w_i > 0$ can be thought of as a measure of the *relative importance* of $f_i(p)$.

An alternate approach to solving (1.2) is to formulate the *minimax problem:*

$$\min_p \max_i \{ w_i f_i(p) \} \qquad (1.4)$$

Typically the optimal solution of (1.4) will yield a value p^* such that:

$$w_i f_i(p^*) \leq w_j f_j(p^*) \qquad (1.5)$$

for a subset of the objective functions, while all others will end up with lower values.

Observe that in both cases we have to make an *a priori* and somehow arbitrary choice (meaning before having a concrete opinion on the behavior of the circuit) of weighting factors. It is obvious that it is beneficial for the *Decision Maker* to have the opportunity to change these weights, especially after watching the first solutions and thus obtain a feeling about the circuit behavior. The necessity of user-algorithm interaction is obvious here. This necessity is to be discussed in some detail very soon.

Observe that in Multiple Objective Decision Making problems, there are no *optimum* solutions in the strict sense of the word. However, there are solutions clearly *inferior* compared to others. (for example, we will never prefer a solution that gives *n* poor output responses, over one that gives *n-m* poor output responses, identical or worse than the previous and *m* good output responses.)• On the contrary, looking for the best design, starting at a point p in the input space, we would like to move towards a direction Ap such that:

$$f_i(p - \Delta p) \leq f_i(p) \qquad i = 1, \ldots, n \qquad (1.6)$$

We can say that this change in Ap gives rise to a change Af * f(p+Ap) - f(p). We would like Af to have only non-positive components.

While moving along this *good* direction we will find a point p from which no direction for further improvement exists. This point p is a special point in the input space called a Pareto Critical Point.

In the output space, when no direction for further improvement exists* we may say that we hit a barrier that constrains further movement. Such a point f on the barrier of the output space is called a Noninferior Point

In order to go further and present some important theorems, we need to introduce some formal definitions.

> Definition 1: The set of points q « R_n such that there exists p « R_m where q « f(p) is called the realizable set of outputs.

$$Q \ll \{ q \in R_n \mid f(p) * q \} \tag{1.7}$$

In general Q * R_n. so that Q has a boundary 3CL This boundary is of special interest since it forms the barrier that actually contains the noninferior points of the output space.

> Definition 2: A point p € R_m is a Pareto critical point, for a mapping f, if no small change Ap exists such that:

$$f_i(p*Ap) \leq f_i(p) \qquad i * 1,\sim,n \tag{1.8}$$

> where not all of the above relationships are equalities.

Pareto critical points correspond to local minima of our objectives. For global minima we have Pareto points (not critical), and we can relax the restriction imposed by the previous definition, that Ap must be small.

Two observations give us a view of the importance of Pareto points (both critical and global). Let us define the set of Pareto points by *P,* and the set of critical Pareto points P_c. Note that $P \supset P_c$.

The first observation is the plausible statement that the best circuit design occurs at a point p € *P.*

We can also observe that if p « P_c and each f_i is continuous, then f(p) < *id.* In case that the boundary of Q, £Q, is smooth, one can define at each point q (5Q the outward normal to q, n(q). In that case:

$$f(c) = \{ q \ll R_n \ ! \ q \# 5Q.\ n(q) \pounds 0 \} \tag{1.9}$$

Observe from Fig. 1*1, that at a Pareto point in a two objectives problem, the objectives are in exact conflict, that is:

$$\lambda df_1/dp = -df_2/dp \tag{1.10}$$
for some $\lambda > 0$.

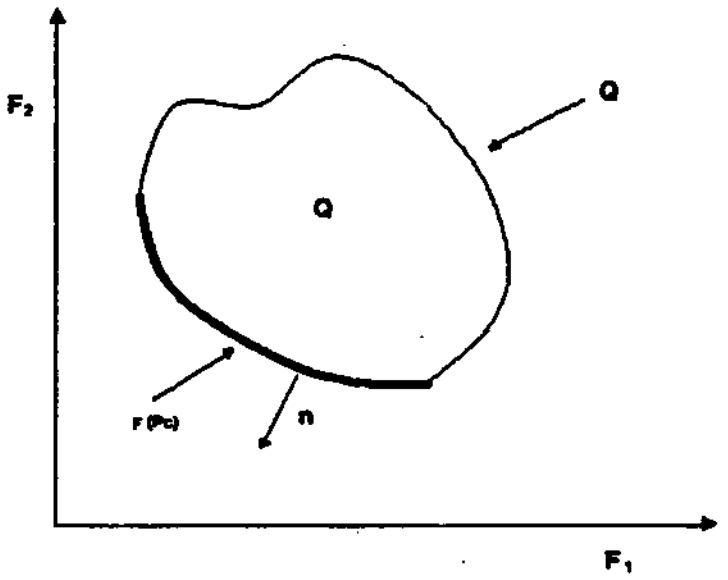

Figure 1-1: Two Objective Functions representation.

However this condition is only necessary, and this of course means that there *are* points p € P_c that cannot be reached by solving this equation.

Nevertheless, from this relationship we can extract the following important

observation: *if p* is the minimizer of F(p) * ZivY. and iQ is smooth at ftp*), then* -iv *is the outward normal to Q at f(p*J.*

The two theorems that follow, are to establish the fact that not all best solutions can be obtained by the weighted sum approach.

Theorem 3: Let MQ) denote the convex hull of Q and 3MQ) its boundary. Then f(p*) < f(p) A *SH(Q)* if and only if there exists w > 0 such that:

$$w^{r}f(p^*) \ll \min_{P} lwf(p) \qquad (1.11)$$

Thus the set of points *p* < *P* which are obtained by solving the equation (1.11) if the minimum is set equal to 0, is the set:

$$\{\ p\ /\ p\ f\ P \text{ and } f(p)\ \#\ 5MQ)\ \} \qquad (1.12)$$

Theorem 4: If each f. is a convex function, then $P^{L}P_{c}$ and f(p) fc MQ).

We now discuss some properties of the minimax approach. Specifically we can show that with the minimax approach we can reach all Pareto Critical Points , and not just the convex subset described in the previous theorems. Our problem now has the form:

$$\min_{p} \max_{i} \{\ wf(p)\ \} \qquad w_i > 0 \qquad (1.13)$$

Theorem 5: If the solution x* of the above problem is unique, then x* is a Pareto point.

It is instructive to consider a geometrical interpretation of the problem. The multiple dimensional solution must satisfy the relationship:

$$\min_{q} c\ Q!!w_{q}\ iioo, \text{ where } w\ *\ (w,\ f\sim.,w\ )^{T} \qquad (1.14)$$

The level contours of $!!\ w\ !!_{Q\ OO}$ is a m-dimensional orthogonal parallelepiped centered at 0. By varying the weights the shape of the orthogonal is altered, thus we can change the point of the contact q*. This is easily illustrated in the two dimensional case where the m-dimensional parallelepiped is reduced to a rectangle (see figure 1-2).

Theorem 6: If x^* is a Pareto point, then there exists a vector w with $w_i > 0$, such that x^* is a minimizer of:

$$\min_x \max_i \{ w_i f_i(x) \} \qquad (1.15)$$

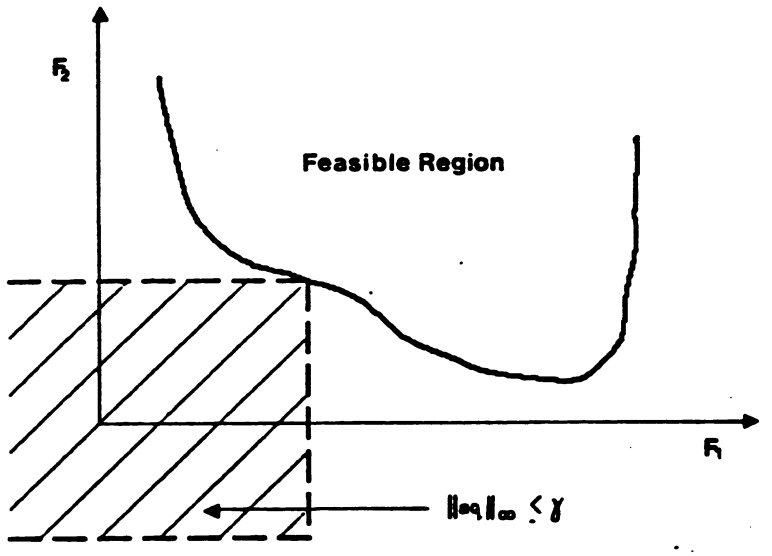

Figure 1-2: Minimax approach into a two dimensional space.

It is obvious now that minimax optimization has an apparent advantage over the weighted sum method. However there is a disadvantage as well. Namely, the constrained optimization algorithms (which are used to solve the minimax problem) are much slower, and more computationally expensive, than the unconstrained optimization algorithms that are used to solve the weighted sum problem. Furthermore, we have to state that there are approaches similar to the weighted sum method that can reach all Pareto points and still be solved as an unconstrained optimization problem.

These methods are based on the observation that if we consider objective functions of the form:

$$f_{P}\ Ot)\_.fP\ (x) \qquad (1.16)$$

then by increasing the parameter p, the realizable set in the output space tends to become progressively convex along the line:

$$f^*_1 \ll f^*_2 \sim\ll f^*_j \qquad (1.17)$$

Thus by formulating the sum consisting of factors { (wf_i P }, and by increasing p, we can transform the output space enough to become convex near any image of a Pareto point. (Note that if p -> oo_f the formulation becomes that of the minimax problem.)

Lightner [Lightner 79] shows that for each Pareto point x*, there is a minimum value of p, p* such that x* is the minimizer of:

$$\min_x \{ (Z\ tw_i.f_i.MpV'** \} \qquad (1.18)$$

and for each q *t* p*, there exists a set of weights w(q) such that x* is also the minimizer of:

$$\min_x \{ (Z\ (w(q)f_i(x))\ Q\ )\ 1/Q \} \qquad (1.19)$$

Finishing this part we present a theorem that will further clarify the relation between weighted sum and minimax methods.

Theorem 7: Let { X } be the lagrange multipliers obtained in the constrained minimization:

$$\begin{aligned} &\text{minimize} : g \\ &\text{such that} : g - f_i(x) \geq 0 \qquad i = 1,\sim.,r \end{aligned} \qquad (1.20)$$

(which is equivalent to minimax { $f_i(x)$ }). Let x* be the solution to the previously defined constrained minimization problem, and suppose $f(x^{\#})$ « *iHid).* Then:

$$\min_x \max_i \{f_i(x)\} \sim\ll Z\ Xf_i(x^*) \ll \min_x Z\ Xf_i(x) \qquad (1.21)$$

In other words, if x* is reachable by a weighted sum minimization, then the Lagrange multipliers obtained from the minimax problem are the correct weights to reach x*.

On summary, we have shown that by the weighted sum methods (or at least by similar methods) we can reach any point and still have to solve a convenient unconstrained optimization problem. However, in almost all physical design situations, there are constraints anyway. Thus we still have to solve a constrained minimization problem. In the next part we are going to give an overview of the constrained optimization techniques.

1.3 Methods for constrained Optimization

Most electrical engineering design problems are in fact Constrained Optimization problems. The usual reason for this, is that there are always physical limitations (e.g. resistors with no negative values) or practical limitations (e.g. limited resistor values on an IC due to restricted area). These limitations are the cause of the problem constraints.

The methods available to solve this problem are of quite a diverse spectrum of mathematical interpretation. In this section we briefly survey these methods. We start with the basic problem formulation along with two very important theorems.

A typical constrained optimization problem can be stated as follows:

minimize : f(x) (1.22)
such that : g(x) £ 0 i * 1,~.,r

The *region of feasibility* i s f « { x | g £ 0 , i* 1,~.,r }. Now we will state a fundamental theorem that deals with the necessary conditions for the existence of the optimum solution of the problem presented here.

> Theorem 8: (Kuhn-Tucker Necessity Theorem) If a) f and g are differentiable at x* and b) if for any z for which z^T (Ag(x*)/Ax) £ 0 for all i where g(x*) « 0, we have z pointing into *F* from[1] $x^{\#}$, then the necessary conditions that x* be a local minimum of the problem (1.22) is that there exist X\ j * 1,...,r, such that

$$g(x^*) \leq 0 \qquad (1.23)$$
$$g \cdot g(x^*) = 0$$
$$X \cdot \wedge O$$
$$dt/dx\ (x^*) \cdot I \wedge Ag\gamma dx\ (x^*) \ll 0$$

The λ_i^*, i = 1,....,r, are called the Lagrange Multipliers.

If f(x) and $g_i(x)$, i = 1,....,r are convex, (convex programming problem), the Kuhn-Tucker conditions are both necessary and sufficient. This is stated in a more formal way in the next Theorem:

> **Theorem 9:** (Kuhn-Tucker Sufficiency Theorem) If f and g_i are convex and continuously differentiable, then a sufficient condition that x^* is a global minimum of (1.22) is that there exist multipliers λ_i^* such that the KT equations are satisfied.

We have to admit though that most problems are not convex. However, if we can solve the KT equations, local optima can be obtained. What follows is a brief description of some of the existing methods to find this optima.

1.3.1 Methods of feasible directions

The problem basically consists of determining a point $x_j \in F$, and then finding a direction that goes through the acceptability region. Then we have to find the step such that the function decreases. The most prominent disadvantage of this method, is that we often encounter the so called *zigzagging* phenomenon, meaning that we wander a long time around the solution before we hit it. A remedy for this problem, is to take into account the *almost active constraints*, a method that tries to avoid *wall bouncing* while looking for the solution.

The disadvantage of this method is that the structure does not guarantee that the sequence of points found will satisfy the first order optimality conditions. However, if some parameters concerning the *almost active* constraints are dynamically changed during the execution, a convergent algorithm is obtained.

Methods of this kind are particularly attractive in engineering design because, once a feasible point has been produced, the sequence of points generated by the algorithm remains feasible.

1.3.2 Linear Programming Methods

The method is based on linearizing each function locally and using linear programming to select a local best step. In this part of the method description, for practical reasons we will use the minimax problem formulation[4]:

$$\min \max_i \{ f_i(x) \} \tag{1.24}$$

The basic linear interpretation here is the following:

$$g_i \approx \Delta f_i / \Delta x \ (x^k) \tag{1.25}$$

(where x^k is the current best point)

$$f_i(x^k + h) - f_i(x^k) \bullet g_i^T h \quad i = 1,\ldots,r \tag{1.26}$$

In order to find a step $h = (h_1, h_2, \ldots, h_m)$ we have to apply the following linear program :

$$\begin{aligned} &\text{minimize} \quad y \\ &y,h \\ &\text{such that } f_i(x^k) \bullet g_i^T h \le y \quad i = 1,\ldots,r \\ &-\delta_k \le h_j \le \delta_k \quad j = 1,\ldots,m. \end{aligned} \tag{1.27}$$

After that, the step is tested if a decrease is achieved in the actual function. If not, the step is rejected. Various tests are proposed for this part, the performance depends on the specific problem, as well as the amount, both that of the objective functions and design parameters.

The typical solution of this problem (as a minimax problem) can be expected to be a point (x,y) where:

$$f_1(x) \approx \ldots \approx f_r(x) \tag{1.28}$$

---

[4] This formulation preference does not by any means restrict the generality, since we have already shown in theorem 7. m page 14 that both formulations (minimax and weighted sum) *me* equivalent

for a subset of the objective functions.

A good observation may reveal that in some sense, the linear programming methods can be thought to be a special case of the *constrained Quasi-Newton methods*, to be discussed latter, but where the quadratic approximation B is never updated.

1.3.3 Penalty-Multiplier Methods

In order to describe this family of methods in a brief and comprehensive way, we have to present the case with only equality constraints. This of course is not any kind of restriction for the problem.

$$\begin{aligned} &\text{minimize : } f(x) \\ &\text{such that : } g(x) = 0 \\ &\text{here } g^T(x) = (g_1(x),....,g_r(x)) \end{aligned} \tag{1.29}$$

The penalty-multiplier method combines the virtues of the penalty function methods with the multiplier methods. We have first to explain what we mean by penalty functions.

A good and simple way to eliminate the constraints of the problem, but still have a good *barrier* that will keep us in the acceptability region, is the idea of *penalizing* the main function for near constraint violation:

$$f(x)+a^{-1}\Sigma\ \Phi(g_i(x)) \tag{1.30}$$

and the penalty function Φ has the properties:

$$\Phi(0) = 0,\ \Phi(t) > 0\ \ t\neq 0,\ \Phi'(0) = 0,\ \Phi''(0) > 0 \tag{1.31}$$

It is almost clear that the apparent role of the penalty function:

$$\Sigma\ \Phi(g_i(x)) \tag{1.32}$$

is that of making the function convex near the solution x^*. Now we are ready to introduce the penalty-multiplier concept: Simply, we have a combination of

the well-known Lagrange multiplier method applied to the *penalized* problem. Thus we minimize:

$$\text{ffccKa}''^{1}! \ \text{«g(x))*X.g.(x)} \qquad (1.33)$$

for some X_i. These of course must be repeated with X_i changing according to some scheme. However we do have a very distinct advantage over the basic penalty methods, meaning that in these methods we have to take an infinite sequence a_k converging to 0 to get the correct answer. Some very important theorems are stated by *Bertsekas* [Bertsekas 76] that provide proof that a finite a* always exists such that for all a, 0 £ a £ $a^{\#}$, if we can choose the correct Lagrange multipliers X_i, then we can find the desired solution to (1.30) just by solving (1.33).

These methods seem to be promising, however we are not going to elaborate more on them, because in this work other, even more powerful methods will be used.

1.3.4 Quasi-Newton Methods for Constrained Optimization

These techniques allow one to mimic Newton's method for constrained optimization, in a manner similar to unconstrained optimization. For the sake of simplicity, we again discuss the constrained problem with equality constraints. Namely:

$$\begin{aligned} &\text{minimize} : f(x) \\ &\text{such that} : g_i(x) * 0 \quad i \ \text{«} \ 1,\sim,r \end{aligned} \qquad (1.34)$$

According to the KT equations, there exist Lagrange multipliers X_i such that

$$h(x,X) \ \text{«} \ Af/Ax \bullet I \ X_i Ag_i/Ax * 0 \qquad (1.35)$$

Note that the equations:

$$\begin{aligned} &Af/Ax \ \text{«•} \ Z \ X_i Ag_i/Ax \ \text{«} \ 0 \\ &g_i \ W \bullet 0 \quad i \bullet 1,L,r \end{aligned} \qquad (1.36)$$

constitute a system with m«r equations and m+r unknowns. The actual Newton's method applied to this would give:

$$\begin{bmatrix} h(x.X) \\ g(x) \end{bmatrix} = \begin{bmatrix} f_{xx} + Z\ X_i g_{ixx} & g_x^r \\ g_x & 0 \end{bmatrix} * \begin{bmatrix} Ax \\ AX \end{bmatrix} \qquad 0.37)$$

Then, after finding Ax. AX, we should approximate by x*«x*Ax, X*»X*AX. The Quasi-Newton approach consists of approximating the Lagrangian function:

$$L(x) > F(x) * Z\ X_i g_i(x) \qquad (1.38)$$

by a quadratic:

$$L(x*Ax.X) \ « \ 1/2Ax^rBAx*h^r(x.X)Ax-«-L(x.X) \qquad (1.39)$$

Then the system we should solve is approximated by:

$$\begin{bmatrix} h(x.X) \\ g(x) \end{bmatrix} = \begin{bmatrix} B & g_x^T \\ g_x & 0 \end{bmatrix} * \begin{bmatrix} Ax \\ AX \end{bmatrix} \qquad (1.40)$$

That of course means that our original problem is actually approximated by:

$$\begin{aligned} &\text{minimize} : 1/2Ax^rB\ Ax \bullet h^T(x.X)Ax \bullet L(x.X) \qquad (1.41) \\ &\text{such that} : g_i(x) \bullet g_i^rAx * 0 \end{aligned}$$

Clearly the problem is to approximate $f_{xx} \bullet £\ X\ g_{i\ ixx}$ with B. and then solve the quadratic problem for each step. Another problem is that of the successful calculation of Ax. We know the direction along which we search for the minimum of some function. The question is which function. If we search along Ax for the minimum of f(x). we may violate the constraints g(x) * 0. *Han (* [Han ??]) has suggested searching along Ax to minimize the function:

$$f(x) * I\ m.|\ g_{(.}(x)\ | \qquad (1.42)$$

where $m_i > |X_i + AX_i|$. This is motivated by the fact that if B is positive definite, then we can prove that Ax is a descent direction for (1.41).

For a more detailed description on these methods, the user is urged to consult [Hwang 80], [Powell 77], [Brayton 80a], [Brayton 80b] and numerous other sources.

1.4 Decision Maker - Analyst interaction

As we often stated in the previous pages, the Multiple Decision Making problem, is often too qualitatively defined to be solved without the intervention of the *Decision Maker.* Thus almost all the existing Computer Aided Design Algorithms permit this intervention. However, there are substantial differences concerning the time, the quality and the quantity of the information needed, as well as the way that this information is actually passed to the *Analyst*.*

The situation can be described as the analyst taking information (usually concerning the preference criteria among the various objective functions), processing this information, solving the numerical problem and finally presenting the response to the *Decision Maker.* Usually this response consists of one or more *noninferior solutions.* After that the *Decision Maker* may accept one of them as the final solution, or he may continue after giving information to the *analyst* about his new area of interest.

We now give a brief overview of the various *Computer Aided Design Techniques* as seen from the point-of-view of *interactivity.*

1.4.1 Classification of Interactive Computer Aided Design Algorithms

The classification we will present, has been made in three steps:

- The stage at which the preference information is needed by the algorithm.
- The type of information needed.
- The major methods in any branch formed from the previous steps.

This classification can be represented by a tree-structure as can be seen in

---

[5] See the footnote Jt page 7

Fig. 1-3 In the following sections we give a brief description of the various branches that are defined by the criteria given above.

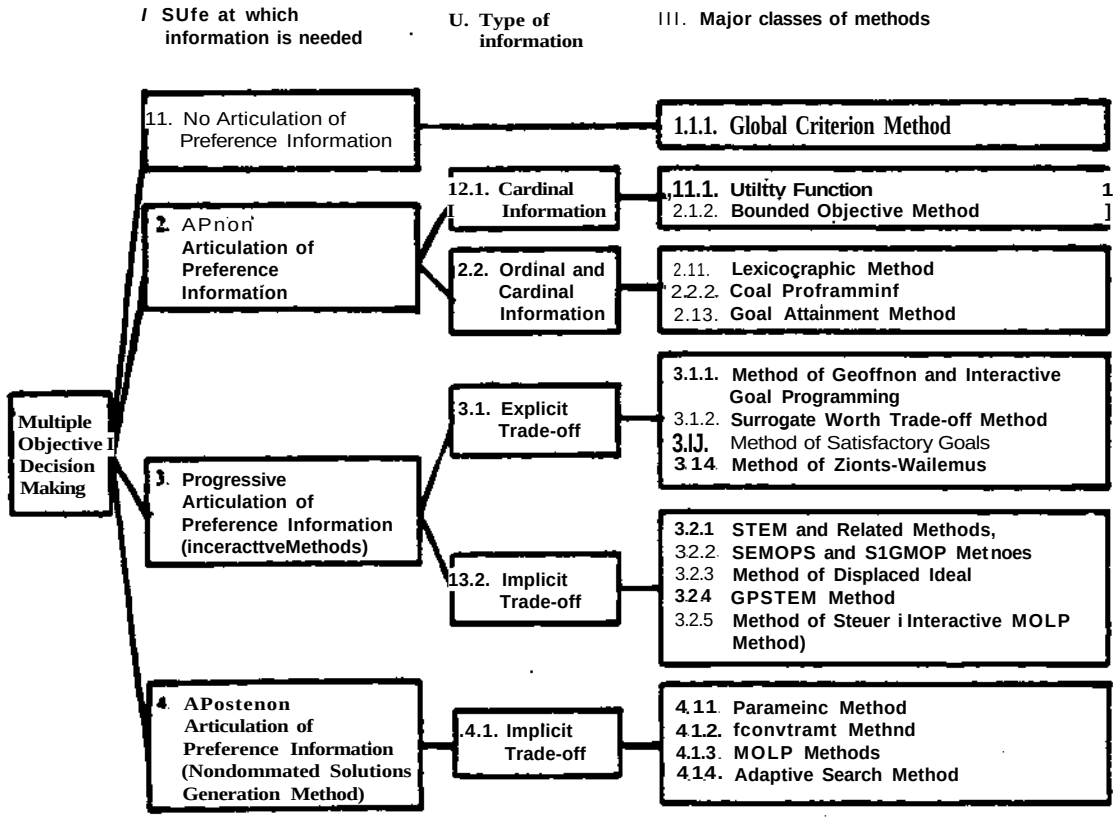

Figure 1-3: Classification Methods ([Hwang 80]).

Methods for no articulation of preference information given : By definition, the methods following this approach do not need any information from the Decision Maker once the problem constraints and objectives have been defined. This means that the problem is formulated only once (by determining the weights or the $_x$ ***preference criteria*** **in general), before it is given to the analyst. After that, the problem is passed to the analyst who will find the solution based on these *a priori* set criteria.**

Various methods have adopted this scheme. Typical representative is the so called *global criterion method* . According to this method an optimum vector is a vector that minimizes some global criterion. This criterion is supposed to embody the *preference measure* among the objective functions and is set by the Decision Maker. The advantage of these methods is that the solution is found without having the *Decision Maker* to make *analytical justifications.* On The other hand* is often very difficult to make all the critical assumptions at the beginning of the process.

Methods for a priori articulation of preference information given: *A priori* here means that the preference information must be given to the algorithm before the solution of the problem. This information may be either *cardinal* or *mixed* (*ordinal* and *cardinal*). The difference between *cardinal* and *ordinal* information is defined to be the following: In the case of *cardinal* information, the *Decision Maker* must give some judgement about specific objective preference levels, or specific trade-off. If the information is *mixed* (*ordinal* and *cardinal*), the *Decision Maker* must rank the objectives in the order of their importance.

Methods in use to solve problems in this area, are the *Utility Function methods (* weighted sum) and the *bounded objective functions for cardinal information.* For mixed *ordinal* and *cardinal* information, methods like the *lexicographic, goal programming* and *goal attainment* are used.

The advantage here is that the *Analyst* has a better (from the mathematical formulation point-of-view)image of the *Decision maker's* preferences. On the other hand, the *Decision Maker* must give some quantitative information to the *Analyst,* and that is considered difficult.

Methods for Progressive articulation of preference information given (Interactive methods) : These methods are what we general call *Interactive Methods.* They rely on the progressive definition of the Decision Maker's preferences along with the exploration of the criterion space. These methods assume that the Decisioq Maker is unable to indicate *a priori* preference information due to the complexity of the problems, but that he/she is able to give preference information on a local level to a particular solution.

It is obvious that with these assumptions, both the *Analyst* and the *Decision Maker* have good and frequent feedback, thus they can drive the solution to the preferred one. This is considered the great advantage of this method. However, the way of communication between the *Analyst* and the *Decision Maker (* very often a CRT terminal), is not efficient in conveying information representing a multiple dimensional problem. Thus we are facing the task of deriving efficient yet comprehensive ways of transferring and presenting this information.

Methods for a posteriori articulation of preference information given: These methods rely on the fact that they generate a set of nondominated solutions and then the Decision Maker chooses the most satisfactory one. making implicit trade-offs between objectives based upon some previous uninidicated or nonquantifiable criteria.

The advantage here is that the *Analyst* does not require any assumptions from the *Decision Maker.* The apparent disadvantage of this class of methods, is that they usually generate a huge amount of nondominated solutions, making the task of choosing among them often very difficult. This family has members like the e-*constraint, adaptive search* and *parametric method.*

1.4.2 The code availability problem

In summary, the classification method presented here, can also include a very important topic, that of the *availability of the appropriate computer code.* It is true that all the methods presented have been implemented, however only few of these software packages can deal with large scale problems. The main reason is that the real-life problems have very often complicated non-linear parts that make the need of efficient computer codes very apparent.

Another aspect that will be of interest later, is the lack of an efficient method to compare the effectiveness of various interaction schemes. The only good effort in this direction was made in [Wallenius 75] that compares three algorithms. For more details on this subject the reader is referred to [Hwang 80].

CHAPTER 2
THE RECTANGLE REPRESENTATION CONCEPT

In this chapter we will give the the description of a rather new family of interactive algorithms for *biobjective decision making*. These algorithms were originally presented in [Polak 80] and use graphical *man - machine* interaction.

The description is divided in three parts: We first introduce the *biobjective topological problem* and we give some formal definitions, then we present the rectangle elimination philosophy along with some optimality criteria, and finally we describe the mathematical and numerical problem associated with this method, combined with a general interactive algorithm based on this philosophy.

By the end of the chapter, we propose some modifications to the algorithm that will expand the interaction capability and will prepare the algorithm for *multiple objective problems*.

2.1 Biobjective Decision Making: A Topological description

The simplest multiple objective optimization case is the biobjective case in which there are only two objective functions. There are two reasons because of which this case is of interest.

- First because the *biobjective optimization problem* has a very convenient 2-dimensional representation that will greatly help both the understanding from the *Decision Maker* and the interaction between him and the *Analyst*[6].

- Second, because an arbitrary n-dimensional problem can be fully represented by n-1 two dimensional cross sections out of the initial n-dimensional space and yet be reasonably comprehensible to enable communication between the *Decision Maker* and the *Analyst*.

---

[6] See footnotes at page 7

Thus* by working and understanding the two dimensional problem, we actually working towards the solution of a higher-dimensional problem.

In biobjective decision making problem the *Decision Maker* has a set of feasible alternatives x < *X* and a biobjective function f: *X* -> R^2 representing two conflicting decision objectives. In making the trade-off, the *Decision Maker* need not consider the entire set of feasible objective values:

$$Y « \{ y « R^2 ! \ y « f(x). \ x « X \} \quad (2.1)$$

but only the set of *nondominated* (Pareto critical points) objective values:

Definition 1: Consider the set defined by the relationship:

$$r • \{ y \# Y ! \ y' \# Y \text{ and } y \wedge y^{\wedge}, \ y \wedge y'_2 \ «> \ y « y' \} \quad (2.2)$$

The set F(y) will be called *trade-off curve,* or *trade-off set* in general (since there are cases that F(y) is not a curve - see Fig. 2-1).

We have already discussed the *III definition* of the multiple objective problem in the previous chapter. Thus it must be obvious, that since there is no systematic way way to find the optimum point on F(y), what we need is an approximation of F(y) from which the *Decision Maker* will choose a solution.

In this work we have chosen the rectangle elimination method as a tool of approximating the trade off curve. As we will see, this method has the following advantages:

- It yields a good representation of the trade-off curve, giving a good image of both its shape and the accuracy of the approximation (by the area of the rectangle).
- It is well suited for interaction between the *Decision Maker* and the *Analyst,* because of the immediate information that is given by the rectangle (the two corners are known points, while the interior contains an unknown segment of the curve). If the *Decision Maker* feeis that he must refine his image before he makes his choice, he has just to indicate the rectangle of interest and the number of curve points he wishes to find. No quantitative information is needed.

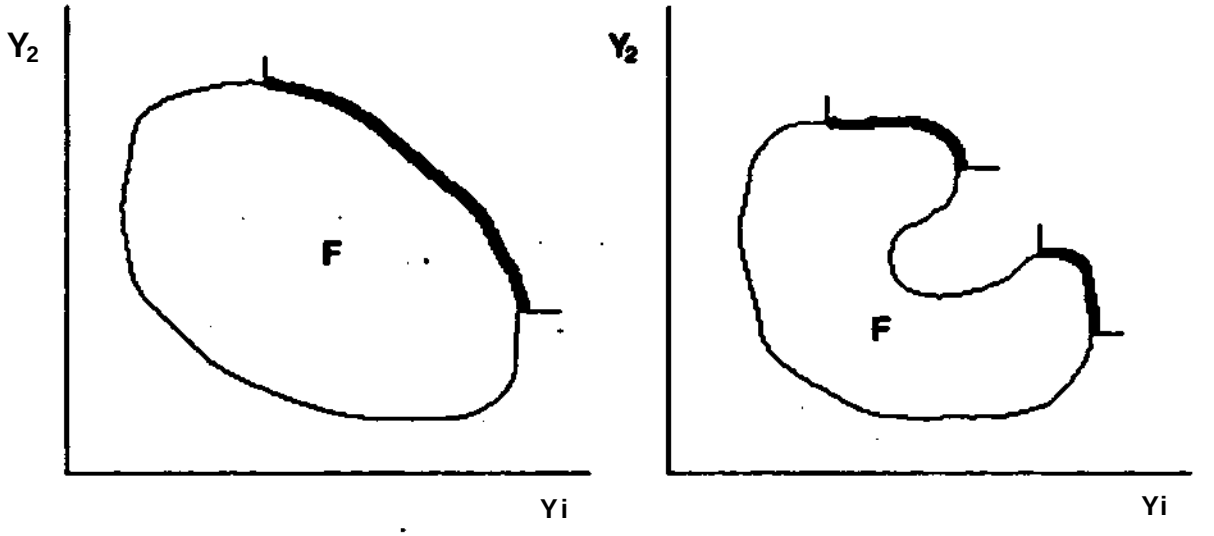

Figure 2*1: Continuous and Non-continuous trade-off curves.

2.2 Rectangle Elimination Methods

In this section we will present the basic theory on which this new interaction scheme is based. We will also present some optimization criteria and some models based on them.

2.2.1 Rectangle Representation of Biobjective trade-off curve.

Before we go on we have to give some formal definitions. First we have to define the terms *strictly* and *general dominating points:*

> **Definition 2:** We say that y' *dominates* y if both components of y' are greater or equal to those of y, and y' is not equal to y.

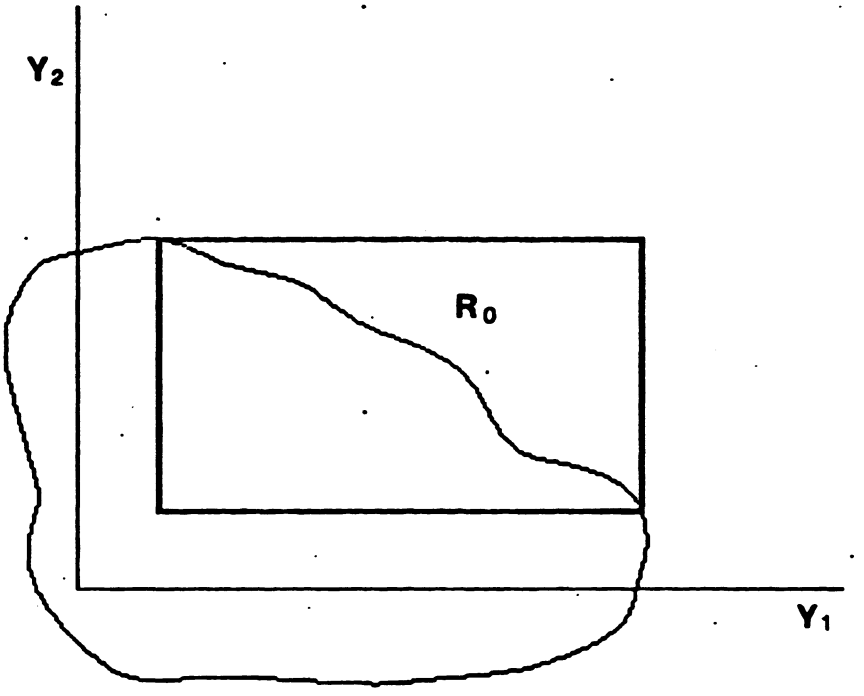

Figure 2-2: One Rectangle Approximation to a trade-off curve.

Definition 3: We say that y' *strictly dominates* y if both relations $y_1 < y'_1$ and $y_2 < y'_2$ are true.

Definition 4: For any $\underline{z}$, $\bar{z}$, $\in R^2$ with $\bar{z}_1 \leq \underline{z}_1$ and $\bar{z}_2 \geq \underline{z}_2$, we denote by $[\bar{z} \backslash \underline{z}]$ the rectangle (see also Fig. 2-2)

$$[\bar{z} \backslash \underline{z}] = \{ y \in R^2 \mid \bar{z}_1 \leq y_1 \leq \underline{z}_1 \, , \, \bar{z}_2 \geq y_2 \geq \underline{z}_2 \} \quad . \tag{2.3}$$

From Definition 1, it is obvious that to find a point on the trade-off curve, we either have to maximize y_1 while keeping y_2 constant, or maximize y_2 while keeping y_1 constant or deriving another scheme maximizing a function that contains both. Of course we also have to be always consistent with the physical constraints of the problem. We denote this class of operations (especially the case when one of the two objectives is kept constant) by the term *Argmax*. So we have the proposition:

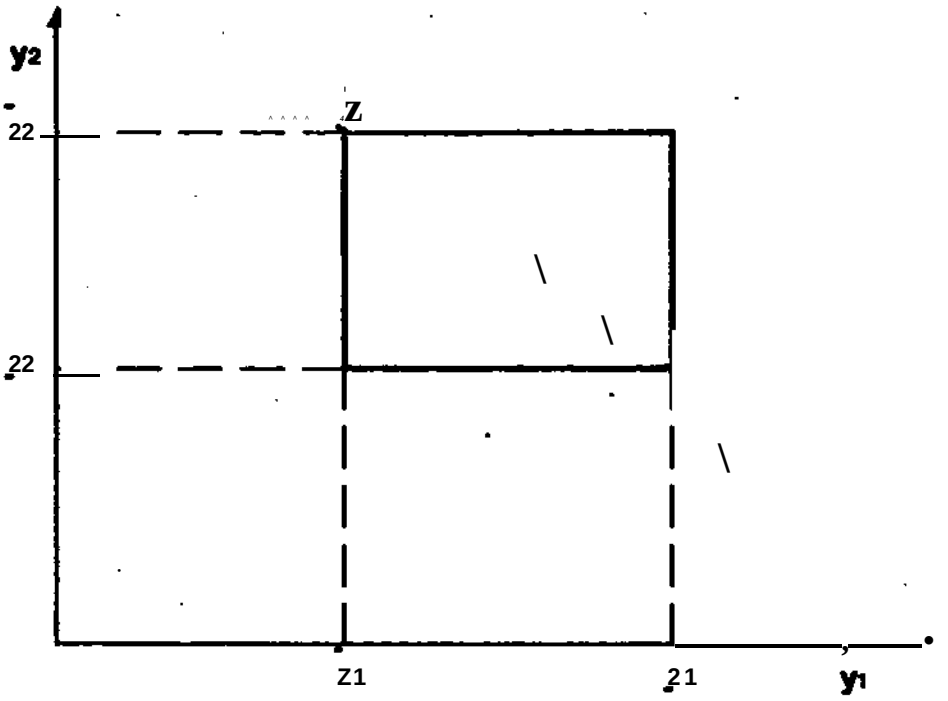

Figure 2*3: The rectangle [z\z].

Proposition 5: If y < Argmax {y_1 !y € Y} and ȳ € Argmax {y_2 !y * Y}# then HY) c C ȳ\y).

Definition 6: Let R_0 « t ȳ\y] and E_0 « {ȳ.y} where y € Argmax {y_1 !y # Y} and ȳ # Argmax (y_2 !y ' Γ}. We then define the class[1] of *nonempty trade-off sets T* in R_0 by

$$Q(R_0) = \{\Gamma/\Gamma \text{ c } R_Q,\ T \# /:\ y,y' \# r \text{ and } y \wedge y' \ll\!\gg y \ll y' \} \qquad (2.4)$$

What we have to understand here, is that if F C Q(R$_0$ ^ then *no* point in Γ dominates another point in F.

From the generality of the previous definition we can understand that the trade-off set F may be a connected curve or just a set of disconnected arcs (see Fig. 2-1).

If the set F is a connected curve in R_0 then there are no points in R_0 that

strictly dominate or are strictly dominated by any point in F. However, since discontinuity in F *is* permitted, *there are* such points.- This set of points is defined:

$$E(r,R_Q) \text{ « } \{ y \text{ € } R_Q \text{ ! } y' \text{ * } F \text{ «> } y < y' \text{ and } y' < y \} \quad \text{<2.5)}$$

In general $E(F,R_0)$ ^ F. When F « $E(F_r R_0)$, the F is continuous and has not vertical or horizontal segments.

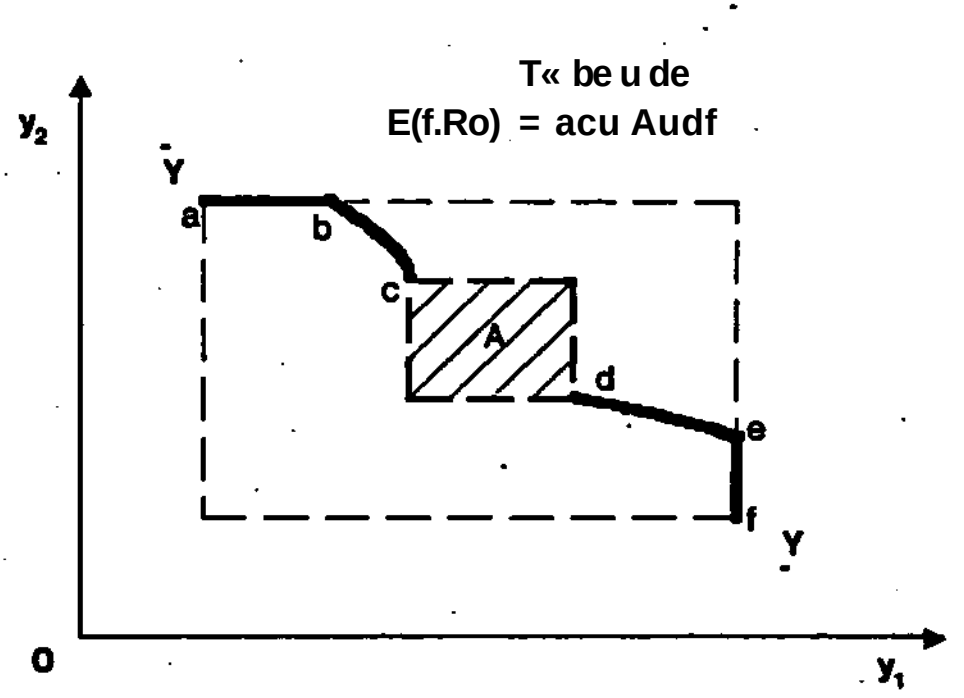

Figure 2-4: The set $E\{r,R_Q\}$.

Now consider the case that we have F C *Q^r)* and we find a Point V € $E(F.R_0)$. From what we have said up to now, we know that F must have no points that dominate y or are dominated by y. Thus F must be contained in the subset of R_0 obtained from R_0 by removing these two sets, that is the union of two rectangles [ȳ\y] U [y\yl. In the same way, we can see that by defining * points, we end up by having *k+1* rectangles whose union contains F.

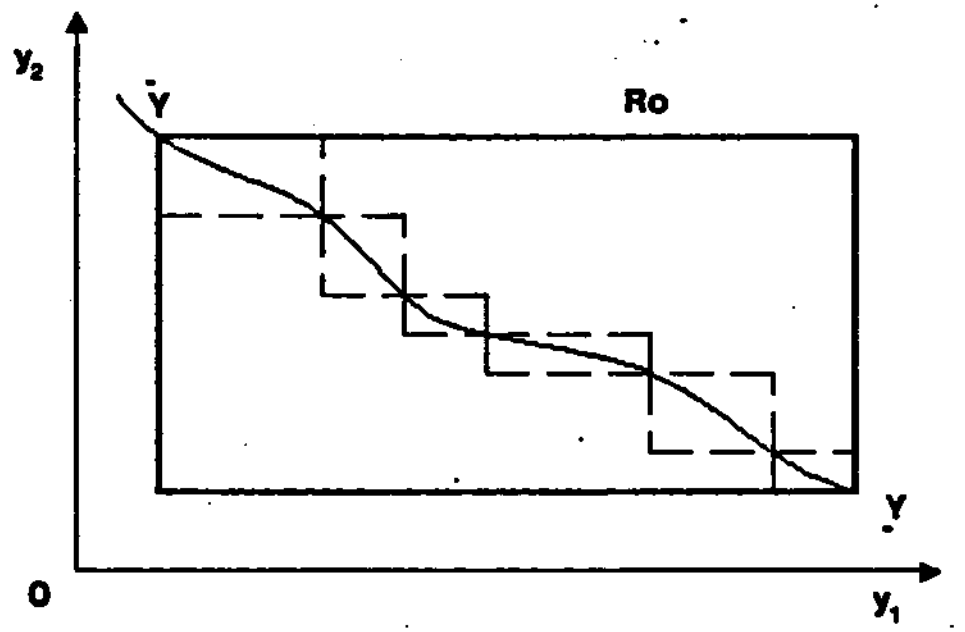

Figure 2-5: N-Experiment rectangle representation of the trade-off set.

Proposition 7: Let T C $Q(R_0)$ and let E C E(I\R$_0$) be a set of *k* points such that:

$$y,y' \in E \text{ thus } y' < y \text{ and } y < y' \tag{2.6}$$

Then:

1. E can be ordered so that E « { $y\backslash y^2,\sim.,y^k$ } with y^1_1 in ascending order and y^1_2 in descending order.

2. The following equation is valid:

$$\Gamma\langle R_0 \bar{}_{U|y} * E \{y' \in R_0 | y<y_1 \text{ Or } y_1<y\} = U^k_{i=0}[y^i \backslash y^{i+1}] \tag{2.7}$$

with y° « $\bar{y}$ and $y^{k+1} = \underline{y}$

and from all this we can justify the following definition:

Definition 8: If we have an initial rectangle that contains F and we

find k points that belong to $E(\Gamma,R_0)$ and no two of them are equal, then this set of points is said to be a *set of experiments* for Γ. Let us refer to this set with E. When $E \cup E_0 = \{ y^0, y^1, \ldots, y^{k+1} \}$ is such that we have y^i in ascending order and y^i_2 in descending order for $i = 0,1,\ldots,k+1$, the $k+1$ rectangles $[y^i \backslash y^{i+1}]$, $i = 0,1,\ldots,k$ are said to give a *k-experiment rectangle representation of* Γ.

Proposition 9: Let $\Gamma \subseteq Q(R_0)$ and R(E) be any rectangle in the rectangle representation of Γ defined by a set of experiments E. If $E' \subseteq R(E)$ (a new set of experiments for Γ), then $E \cup E'$ is a set of experiments for Γ.

2.2.2 The Rectangle Elimination Model

Now we are ready to describe the mechanism with which the interaction between the *Decision Maker* and the *Analyst* takes place.

Our goal is to derive a procedure for interactively approximating and/or exploring $\Gamma(Y)$ for an estimate of a preferred objective value y^*. We will call the procedure to be described the *Rectangle Elimination Method*.

We assume that the *Decision Maker* initially knows that $\Gamma(Y)$ is contained in a rectangle[7] R_0. What the *Decision Maker* is supposed to do, is to derive some experiments[8] in R_0 in order to refine his image of $\Gamma(Y)$, then choose a smaller rectangle *in* R_0 (that is derived from the new set of experiments). This new rectangle (called the *area of interest*) will serve as a new initial rectangle, inside of which the *Decision Maker* will derive a new set of experiments for further refinement of the $\Gamma(Y)$ in this region.

At each stage of the process, the Decision Maker learns more about $\Gamma(Y)$, and his preferences should become increasingly better defined. The process stops after N stages, when the *Decision Maker* eliminates all rectangles except one that contains his preferred point y^*, *yet is small enough that any point on* $\Gamma(Y)$ *in this rectangle will be accepted instead of* y^*.

Before we close this part, we give two useful definitions. One to formally

---

[7] The procedure of establishing this initial rectangle is described later in the chapter.

[8] The term experiment means "the procedure of deriving a new point on the trade-off curve". This term will be formally defined later in this chapter.

define the *Decision Maker Response Function,* and another to define a very important figure of merit, the *uncertainty.*

> Definition 10: A correspondence that will connect a set of experiments E with a rectangle selection R(E) that will satisfy the assumptions:
>
> 1. For any set of experiments E for I\ R(E) is a rectangle representation of *T* defined by E, and
>
> 2. If E and E' are sets of experiments for *T* such that E *C* E\ then R(E') *Q* R(E)
>
> is said to be a *Decision Maker Response Function.* This class of response functions is denoted by *R.*
>
> Definition 11: Let *T* £ $^{Q(R}$J and R # *R.* Given a set of experiments E for r, R(E) is said to be the *rectangle of uncertainty* for E, and the area of R(E) a(R(E)) is said to be the *uncertainty.*

2.2.3 Sequential Search Strategies

Generally, the procedure followed by the *Decision Maker* in exploring the trade* off curve is as follows:

The session starts with an initial rectangle of uncertainty R_0«[ỹ\y]. The *Decision Maker* decides on a number of experiments to be executed in this rectangle by employing the *Analyst.* After the experiments are executed and the points are displayed,, the *Decision Maker* makes his choice among the rectangles that are formatted both from the new points and the initial rectangle's upper left and lower right corners. This procedure is repeated N times until the final rectangle is small enough.

A quite important aspect is that the experiments must be *in* the rectangle defined in the previous stage. This assumption[9] is only made because *experiments outside this rectangle do not help in reducing the uncertainty.*

> Definition 12: Any map Z defined on Q(RJ X *R* by any procedure of the form we just described, creating a mapping from (I\R) into a sequence { E_1,E_2,.... E_N } of sets of experiments for I\ is said to be a *N-stage sequential search strategy.* For given integers k_i>0, i«l,2,..., N. Nwe let $S_N(k_1,k_2,\text{-}., k_N)$ or $S_N(k_i)$ denote the class of 'all *N-stage*

[9] This assumption will be relaxed later.

sequential search strategies for which the number of experiments in E_i is k_i.

Definition 13: For $Z < S_N(k_i)_9$ the worst case uncertainty A(£) is defined by:

$$A(\Sigma) = \sup_{R \in R \text{ and } \Gamma \in \Omega(R_0)} \alpha(R(U_{i-1}^{N} E_i)) \qquad (2.8)$$

The last definition suggests that the worst case uncertainty depends only upon the search strategy, and not upon the specific problem or the *Decision Maker* response. Thus it makes sense to compare strategies by comparing the worst case uncertainty.

Definition 14: An optimal $S_N<k_i>$ search strategy is the one that gives the lowest worst case uncertainty, for all the families that belong to the same family.

Since the computational cost of deriving a point on the trade-off curve is high, we have good reason to use an optimal strategy that will give us good accuracy with a low number of experiments.

We will present without proof the following theorem which suggests a lower limit for the optimum worst case uncertainty:

Theorem IS: For all I * S^k_I we have:

$$A_0^{I1}!^g \ll i^{/} (V^{1>2} * A(Z > f \text{ Where } A_0 * tf < R_0) \qquad (2\#9)$$

Consider now a partition of R' * $[\bar{z}\backslash\underline{z}]$ defined by k parallel lines which have slope $(f_1 - z_2)/(z_1 - \bar{z}^{\wedge}$ (parallel to the first diagonal) and which cut the second diagonal into k+1 segments of equal length (see fig. 2-6). These lines are defined by the family of equations:

$$g(y;\bar{z}.\underline{z}) \ll b_i \quad i \ll 1,2,\sim, k \qquad (2.10)$$

where

$$g(y:\bar{z}.\underline{z}) = (y_1 - \bar{z}_1)/(\underline{z}_1 - \bar{z}_1) + \langle f_2 - y_2 \rangle / \ll_2 - 1_2) \qquad \text{an)}$$

and

$$b_i = 2i/(k+1) \qquad (2.12)$$

We can prove (see [Polak 80]) that any rectangle defined by the intersection of two lines and the actual trade-off curve, is smaller that the one defined-by the intersection of two lines and the diagonal P. thus the partitioning derived in (2.10) has as an upper limit of uncertainty the case when the trade-off curve coincides with the second diagonal of the rectangle. All other cases give a smaller (better) uncertainty.

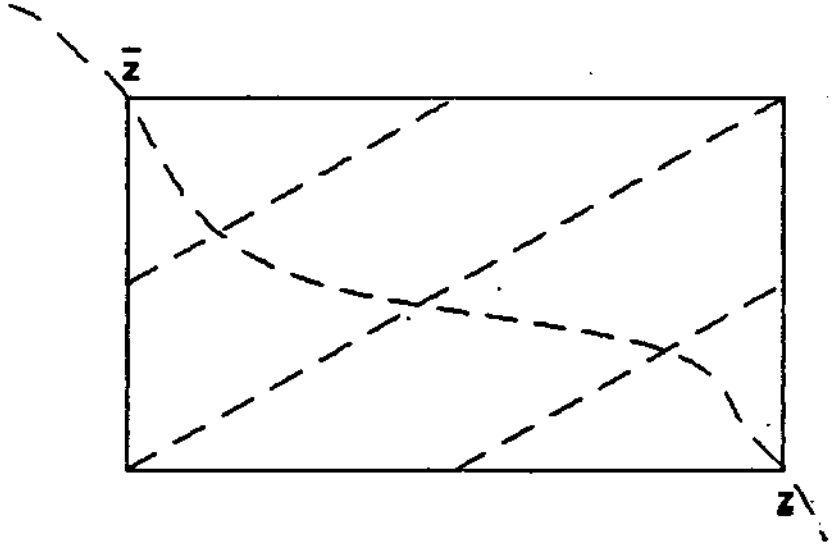

Figure 2-6: A 3-line partitioning of the initial rectangle.

Thus, in order to get an optimal strategy, we have to employ the partitioning method described in the previous section. Thus the appropriate step of the algorithm that implements an optimal search strategy will be:

Compute a set of experiments E_i «{y\~. y^{kl}) C E_o(I\R$_{i-1}$ and R such that:

$$g(y:\bar{z}.\underline{z}) \ll 2i/(k_i*1) \quad i \ll 1,2,\sim. \ k_s \tag{2.13}$$

This is an *optimal strategy* and we have the theorem:

Theorem 16: The minimum worst case uncertainty for an $S_N(k_i)$ strategy is:

$$V(k.)_{N\ i} \ll A_{ft\ 0}\ n^N\ /(k-1)^2_{i\ll 1\ I} \tag{2.14}$$

And also we can state that at stage i of $\pounds(k)$, it is guaranteed that the uncertainty is reduced by a factor at least $(k_{n\ i}+1)^2$.

2.3 A General Rectangle Elimination Algorithm

In the previous sections we discussed the concepts *"biobjective-trade-off sets", "rectangle representation", "search strategies"* and *"optimal search strategies".* We also used the term *"experiment",* to mean the act of finding the intersection of a known line (usually parallel to the main[10] diagonal of a rectangle approximating the curve) and the unknown trade-off curve. Now we will formally define the concept of *experiment* and we will also state the reasons why we delayed this definition.

2.3.1 The generalized Experiment

In the previous sections we avoided further discussion of the term *experiment,* simply because there is a possibility (see Fig. 2-1) to have a gap in the trade-off curve, and so the desired point (defined as the intersection of the unknown trade-off curve and a known line), may not exist. Thus we must give a careful formal definition to cover this case, and this really deserves a separate part of this chapter. We start by determining the problem we have to solve:

Problem 17: Given R' * $[\bar{z}\backslash\underline{z}]$ and b *t* (0,2). find a $\hat{y}$ « $E(r(Y).R_0)$ A R', satisfying $g(\hat{y};\bar{z}.\underline{z})$ * b.

This problem always has a solution because the set $E(F(Y)_{\#}R_0)$, (see equation

[10] We call MAIN DIAGONAL the one with the positive slope. We are going to refer to the other one by the term SECONDARY DIAGONAL

(2.5)) will always have a common point with the line that is defined by this equation.

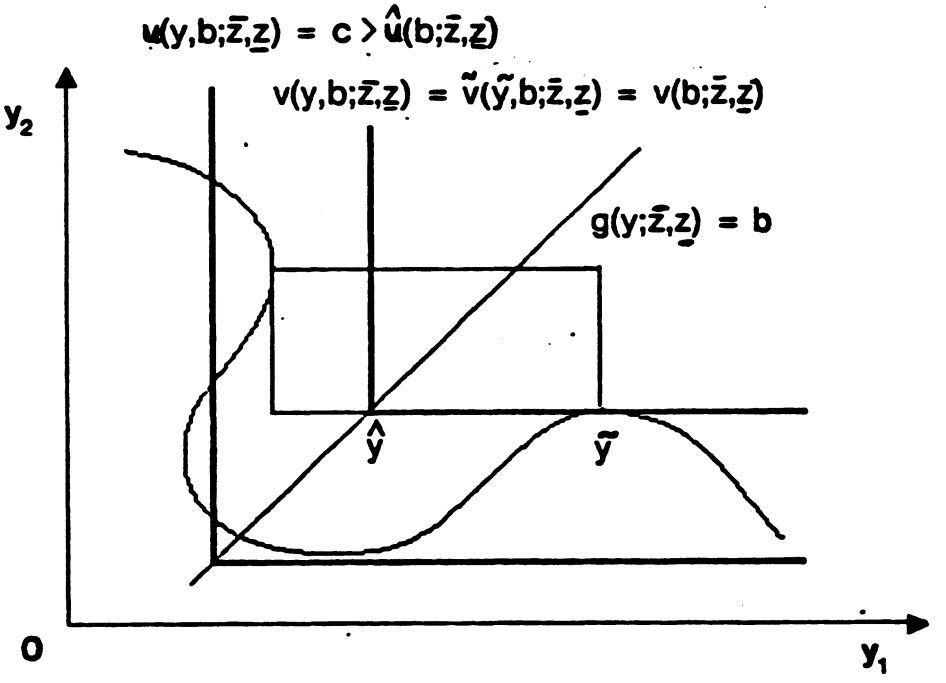

Figure 2-7: The generalized experiment concept.

Our problem now takes the following form:

> **Problem 18:** Find a convenient way of deriving a point on the trade-off curve, provided that you have a solution of the problem 17 and this solution is not a point on the trade-off curve.

In order to pursue our goal, we define the function $u(\cdot,b;\bar{z};\underline{z})$: $R^2 \rightarrow R$ by:

$$u(\cdot,b;\bar{z};\underline{z}) = \max \{ b - (y_1-\bar{z}_1)/(\underline{z}_1-\bar{z}_1),\ (\bar{z}_2-y_2)/(\bar{z}_2-\underline{z}_2) \} \qquad (2.15)$$

Level sets of $u(\cdot,b;\bar{z};\underline{z})$ are shown in Fig. 2-7 along with the line defined by the equation: $g(y;\bar{z},\underline{z}) = b$. Note that $g(y;\bar{z},\underline{z}) = b$ *if and only if the two maximands in* (2.15) *are equal*. And now we can give a more convenient definition of the problem:

> **Problem 19**: Given $R' = [\bar{z}\backslash\underline{z}]$, and $b \in (0,2)$, minimize $u(\cdot,b;\bar{z};\underline{z})$ subject to $y \in Y$.

It is easy to prove (see [Polak 80]), that *if there is a* $\hat{y} \in \Gamma(Y)$ *that satisfies* $g(\hat{y};\bar{z},\underline{z}) = b$, *then* $\hat{y}$ *is the unique solution of Problem 19.*

If the solution of the problem gives a $\hat{y}$ that satisfies $g(\hat{y};\bar{z},\underline{z}) = b$ but is not on the trade-off curve, then *there is a* $\tilde{y} \in \Gamma(Y)$ *such that* $\tilde{y} \geq y$ *with* $\tilde{y}_1 = y_1$ *or* $\tilde{y}_2 = y_2$.(See Fig. 2-7.)

From the previous statements we can make the following useful observation: by finding y and $\tilde{y}$ and seeing if $y \neq \tilde{y}$, we can detect the existence of gaps in the trade-off curve.

And now some notes on the mathematical formulation of our problem. Clearly, this problem has an inherent minimax formulation. This is deceiving though, because in every engineering problem there are also physical constraints to be met by the designer. Thus the actual form of the mathematical problem which we will have to solve, is the one of constraint optimization. In this work we will solve the constrained optimization form of the problem by using the Han-Powell algorithm. The reasons for this choice are explained later.

2.3.2 General Rectangle Optimization Algorithm

The generalized optimal[11] search strategy that is based on the previous observations, can be described with algorithmic steps:

- We start with an initial rectangle R_0 and a maximum number of

[11] with the criteria derived in the beginning of this chapter

experiments allowed k[12].

a The Decision Maker decides on the number of experiments to be executed in this rectangle. This number must be less or equal to the number of remaining experiments.

For each line defined inside our rectangle (one per experiment), we solve the mathematical problem derived in the previous subsection (we find both $\bar{y}$ and $\hat{y}$ (see Fig. 2-7) and by comparing them we can decide if there are any gaps).

- The decision maker now has the following alternatives:

1. Decide that what he has is a good approximation of the trade* off curve and thus terminate the algorithm without more additional.

2. Choose a nondegenerate[13] rectangle that represents his new area of interest, and go back to the previous step for this rectangle.

3. Choose an arbitrary rectangle outside the initial one.

4. Choose an arbitrary[14] line and conduct a single experiment by finding the intersection of this line and the unknown trade-off curve.

2.4 Remarks on the Generalized Algorithm

It must be clear to the reader by now, that the algorithm presented in the previous section is not always an optimal strategy. It really depends to what alternatives the Decision Maker will choose at the last step.

Actually, only alternatives 1 and 2 preserve the optimality. The other two alternatives (initially forced by the fact that we can end up trying to define experiments in a rectangle that has interior points not in common with the trade-off curve, thus we would like to recover by choosing another rectangle), eventually are to help the decision maker to recover from previous misjudgements and of course to set the very, first rectangle to start the algorithm.

---

[12] This Yestriction is based of course on the ultimate CPU time restrictions.

[13] Meaning that is a rectangle contained *m* the initial and defined by two adjacent points

[14] It must have nonnegattve slope

The reader must note that the original algorithm presented by Polak and Payne [Polak 80], provides only the "arbitrary rectangle choice" (apart from the main optimal strategy implementation). The term "arbitrary" means here: "another rectangle that was found in a previous stage and has been rejected". In our implementation though, we found it convenient for the user to be able to choose an arbitrary rectangle (defined by old couples of points not necessarily adjacent, or by new points or by one old and one new point).

Finally, the last alternative that involves the definition of lines with nonnegative slope, is simply a feature to support the previously discussed "arbitrary rectangle definition". It also can serve to give the user the way out from a "complete loss" (for instance, initial rectangle defined on a gap).

Closing this chapter, we have to add that the same algorithm can be used to explore multidimensional spaces (one 2-dimensional cross-section of the space at a time). This possibility will be discussed further later. In the following chapter we will present an implementation of the general algorithm described here.

CHAPTER 3
RECTANGLE ELIMINATION ALGORITHM IMPLEMENTATION

In this chapter we describe the implementation of the general algorithm presented in Chapter 2. All the software is written in APL and the graphics package is compatible with the HP 2647A graphics terminal.

The presentation is divided into three major parts. The first part is devoted to the description of the general structure of the program and it's subsystems. The second part describes the supporting software, that is the numerical package and the circuit simulator to be attached to our program. The third part is devoted to a sample run of the algorithm (for a "real-life" engineering problem), simply to point out the various features.

3.1 Main Structure

This part is to be divided into three subparts. We will first describe the way that the program is constructed, then we will discuss the APL graphics package and finally we will elaborate on the interaction capabilities of the package.

3.1.1 The Command Level Concept

For ease of construction and maintenance, the whole program is written in a hierarchical manner. We used six main levels called *Command Levels.* There is one *Command Level* for each separate form of *Decision Maker - Analyst*[15] interaction.

The main part of each Command Level, is the set of commands available to the user. This is what distinguishes one Level from the other. At present the six levels available are:

[15] See footnotes at page 7.

1. Level one The user will decide if he wants to continue a saved problem[16], or if he wants to clear the buffer to use it for a new purpose. In a later version of the problem the user will decide *which* saved problem to continue or to inspect, and he will also decide on where the current problem is to be saved.

2. Level two: The user is prompted for the definition of the axis for the 2-dimensional space to be inspected. We can always come back at this level and redefine the axis. The first time through the program this level cannot be skipped.

3. Level three: This level is to help the user to set up the first rectangle. This is done with numerical input, by defining two vertical or two horizontal lines to include the first rectangle. This level can be skipped, or can be revisited whenever the user wishes.

4. Level four. Here the user will be prompted to define the maximum number of experiments[17] allowed. This level can be skipped or revisited, however, if the number of experiments is exhausted, we have to pass through this level to continue. This part can be considered as being a *watch-dog* in order to spare CPU time.

5. Level five: This is the main Command Level of the implementation, meaning that here the rectangle elimination algorithm is actually implemented. Here the user has a wide variety of choices, starting from the ones that constitute the main algorithm, (choice of a nondegenerate rectangle, arbitrary line definition, local experiments number) up to some special aides to improve the interaction (like the plus or minus zooming capability).

   All the information here is passed back and forth via the graphics screen and the graphics cursor position. This is found to be very helpful for the *Decision Maker - Analyst* interaction.

6. Level six: This is the "special facilities" level. Since the "interaction convenience" can only be seen from a personal point-of-vtew, this level provides the opportunity the decision maker to "adjust" some of the features of the program to his personal taste. Things like *grid existence, zooming power, point labeling* can be adjusted from this Command Level.

In general, the user is expected to follow a default logical path through the modes, however he is free to move within the level structure arbitrarily. (see Fig. 3-1).

---

[16] All problems solved are saved into an internal buffer.

[17] See the definition of the term at Problems 17 and 19.

During the progress of this work, the *mode construction* seemed to be helpful for both the user and the programmer. From the programmer point-of-view, it gives a clear image of the code, and so helps both in construction and maintenance. From the user point-of-view, it gives a clear image of the logical structure and it helps during the execution because the user is not distracted from commands that belong to a different level.

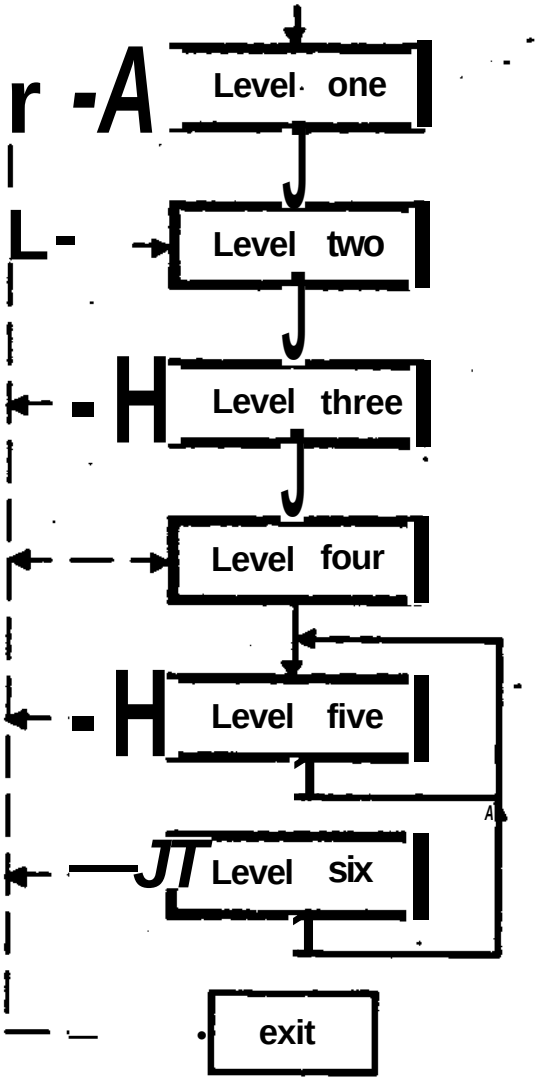

Figure 3-1: The level structure.

3.1.2 The APL Graphics Package

The term *graphics package* is used here to indicate all the commands and construction designed to drive the special functions of the HP 2647A terminal. Since the functions are not necessarily graphical (functions like "read keyboard" or "change page" are also used), this term might be misleading. However, we will use the term *graphics* since the main purpose of this package *is* to drive the graphic functions.

The graphics package has been built and initialized outside our main program. This is done to help us transfer it, modify it, and even use it as a general purpose package (with minor modifications).

The initialization has a *tree-like* logical structure with progressive specialization. At the root we define the *escape* "$" variable to be used for all the commands in the package. Around the root we define some low-level general purpose commands. As we are moving from the root to the remote branches, we find more complicated specialized functions (see figure 3-2).

Effort has been made for the complete documentation of the package, that will permit easy maintenance and transferability.

3.1.3 Interaction Capabilities

Since the purpose of this package is to implement a highly interactive algorithm, special care has been taken to make this interaction as convenient as possible. We assume that the *convenience of interaction* depends mainly on three features of the software. These features are: The *logical structure* to be learned and followed by the user, the *information form* to be exchanged between the *Decision Maker* and the *Analyst*, and the *error tolerance* of the program. We will elaborate now on these topics.

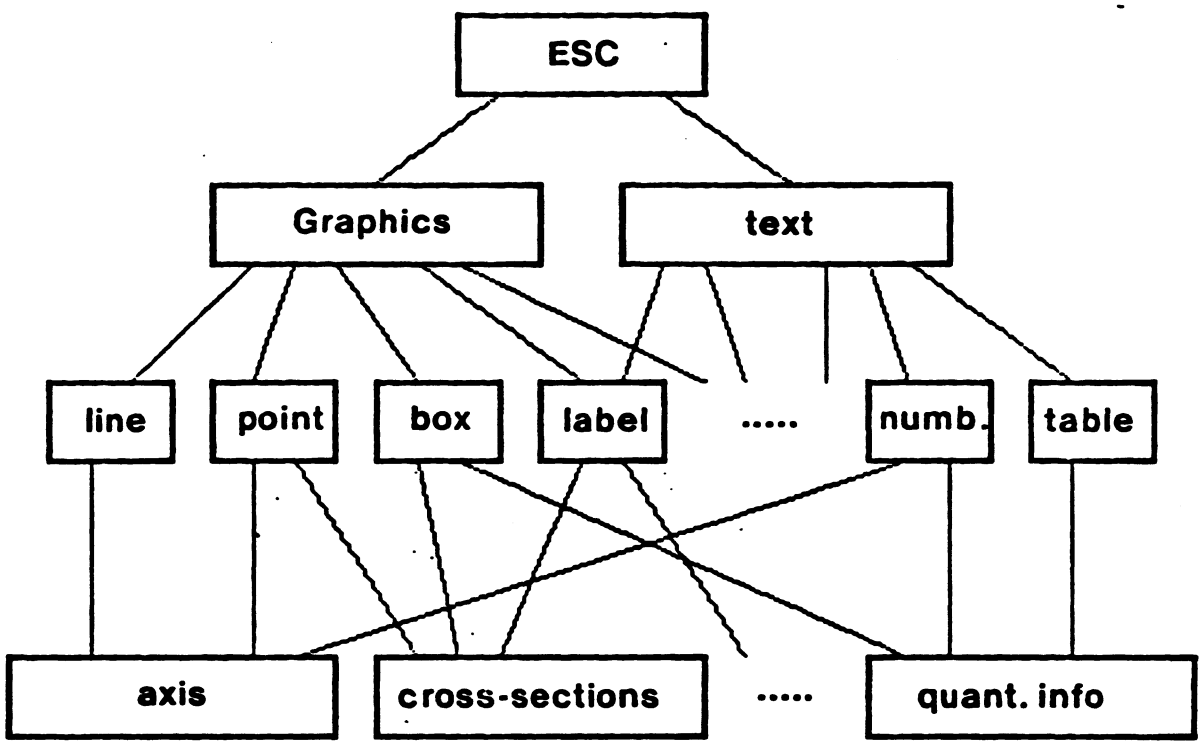

Figure 3-2: The graphics package tree construction.

3.1.3.1 Logical Structure

We have already discussed the logical structure of our program in 3.1.1. Here we will discuss the mode structure from the user point-of-view.

The first important feature in this structure, is that we have a good separation among commands that belong to different areas. Thus the user is not distracted by them, and at each specific point has specific tasks to perform. This provides both convenience and ease of understanding. It is also easy to provide *on line* documentation (giving *two levels* of help, both local and general).

The second important feature is that we can *guide* the user to move within the structure in the way which is suggested by the algorithm, and yet leave the user free to move within wider limits of choice. We found that in this way we can provide the greatest flexibility, a really important aspect in the implementation of an interactive algorithm.

3.1.3.2 Information Form

Since the interaction is actually an information exchange, the form of the information to be transferred is of great importance. This information must be both accurate and concise, both for the *Decision Maker* and for the *Analyst.* However, for the *Decision Maker* "concise" information is a piece of information that is comprehensible, easy to understand and to process. On the other hand, the *Analyst* prefers information expressed in numerical form.

Thus, there is an obvious trade-off that must take place here between *accuracy* and *convenience.* It is accurate to type a number on the keyboard, or to read a number on the screen, but it is not convenient, especially when one has to do that repeatedly. It is convenient to read graphics information and to pass information just by pointing at a certain position on the graphics screen. There are cases though in which this is not accurate enough.

In our case we went midway. This was possible because the algorithm we implement *does not ask the decision maker to make explicit quantitative judgments* (see paragraph 2.1), and this leaves us enough room to use the graphics screen as the interaction media. So, a big portion of the information is passed through the graphics screen. When accuracy was important, we simply used numerical form of information. In this implementation, numerical information is used to define the diagram axis and to set up the first rectangle. Graphical information is used to choose a new nondegenerate rectangle, to set up single experiments and to use some of the graphics facilities like the zooming facility.

3.1.3.3 Error Tolerance

A good interactive program, must include *error recovery techniques,* so no matter what the user types on the terminal, the program can sustain operation. This feature has been difficult to implement in the APL environment. However, since an APL user is expected to have a certain level of sophistication, he should be satisfied with a program not fully *error proofed.*

The precautions we have taken to improve the *error tolerance* of this package can be divided into two main categories. The first is defined at points where *non-numerical information* is expected. The second is defined at all the points when *numerical information* is expected as input. In cases that belong to the first category, the *error tolerance support* includes extended help upon request, local help in case of error and (using a special function of the HP 2647A terminal) reading and reflecting of the command without hitting <cr>. In all other cases (numerical input), error checking is more "loose", since the only feature that is checked is the dimensionality of the input. We found however that it is a rare case to make another error here. In both cases the error tolerance is improved by providing adequate prompting information.

3.2 Supporting Software

It should be obvious to the reader that this algorithm normally needs the support of at least two other software packages. We are going to refer to these packages by the term *supporting software.* The supporting software discussed here, includes the numerical package to solve the problem posed by the *experiment* (see Problems 17 and 19), and the circuit simulator to "transform" the problem from the circuit form, to the equations form.

The coupling of all the *supporting software* to the main algorithm implementation, is of great importance, and in the "ultimate" version will provide us a really automated computer aided circuit design tool. However, in the current version of the program, only the numerical package is really integrated to the algorithm, mainly because the proper circuit simulator was not available by the time this work was completed. In the rest of this section we are going to discuss the choice and the use of these two *supporting software* packages.

3.2.1 Numerical Package

On the development of the numerical package, three subjects are of importance: The mathematical formulation of the problem, the choice (and the criteria for that choice) of the appropriate algorithm, and the implementation problems that are posed by the previous aspects.

3.2.1.1 Problem Formulation

Recall Problem definitions 17 and 19 and the equation (2.15) that is derived from these problems. The form of the problem is:

$$\min_{p} \max \{ b - (y_1(p) - \bar{z}_1) / (z_1 - \bar{z}_1) \ge (z_2 - y_2(p))) / (z_2 - z_{2j}) \quad (3.1)$$

$$\text{subject to : } g_i \le 0 \qquad i = 1,\dots,r$$

where p is the vector of n designable parameters. $y_i(p)$, i*1,2 are the two objective functions that we have to maximize. (From now on we will use the term "maximize" in order to achieve a minimal level of consistency with the graphical interpretation). The rest parameters have to do with the line that is supposed to cut the trade-off curve (see Fig. 2-7).

For reasons of *hand/ing convenience,* it is useful to transform this *constrained minimax problem* to an equivalent *constrained minimization problem.* This transformation is fully justified by the theorem 7. The new form is:

$$\min_{P,T} y \quad (3.2)$$

such that:

$$y - (b - (y_1(p) - \bar{z}_1) / (z_1 - \bar{z}_1)) \ge 0$$

$$y - ((\bar{z}_2 - y_2(p)) / (\bar{z}_2 - z_2)) \ge 0$$

and

$$g_i(p) \le 0 \qquad i = 1,\dots,r$$

This is the form that eventually we will use to solve the problem. This specific subject is discussed next.

3.2.1.2 Numerical Algorithm Choice

It should be obvious that if we really want this to be an interactive program, what we need is a fast numerical algorithm to find the noninferior points. Thus the first constraint in our choice is posed: it is *speed.* Another fact that we should take note of is the observation made in 1.2. This tells us that the point we *are* looking for is actually on the boundary, that means that at least some of the constraints are active at the solution.

The previous two observations rule out of our choices the method of Feasible Directions (see 1.3.1) and the Penalty * Multiplier methods (see 1.3.3). Another observation is that the equations extracted from a circuit representation are usually strongly non-linear. This rules out the Linear Programming Methods (see 1.3.2).

Thus we what we finally have, is the *Quasi - Newton Methods* (see 1.3.4). The numerical package we are going to use is a version of the one written by *Luis Vidigaf* and *Tariq Samad.* The problems of coupling this package to the main program are discussed next.

3.2.1.3 Coupling Problems

Our requirement of having an interactive algorithm poses an implementation problem in the coupling of the numerical package to the main program. Specifically, operation of the numerical package must be invisible to the user. However, there is always the possibility that the numerical algorithm will not converge to the proper solution. Thus the user must have a level of attendance over this part. This attendance starts with the selection of the appropriate initial values and ends with the inspection of the solution to assure proper convergence. We now describe how these two cases have been covered:

Initial Values Selection : We created a separate routine to perform this task. This is done by reading the buffer (where all the previous solutions are stored) and finding the closest one in case of single experiments, or calculating a linear approximation in multiple experiments cases. We found that this scheme performs well when we are in a small region on the trade-off curve. More complicated schemes may improve this performance, however we

found that the *Han-Powell* algorithm is not very sensitive to the initial solution selection.

Solution Inspection : A straightforward way to inspect the validity of a new point, is to observe its position on the screen. Points that are clearly out of position (on the graphics screen) mean that the algorithm did not converge. We provide the opportunity to the user to "reject" such points (by erasing them from the buffer) and repeat the experiment with manually selected initial values and error tolerance.

3.2.2 Circuit Simulator

The existence of the appropriate circuit simulator is of great importance. This circuit simulator should be used only once to initially define the circuit, and then will be called as an invisible subroutine every time the parameters are changed, in order to eliminate the volume of unnecessary calculations, the "parameter update" must be done after the equations of the specific circuit are formed.

At the time that this work was about to be submitted, the only close approach (in our working environment) to the simulator we just described, was the one under development by G. *Jordan.* This simulator is written in APL and can solve reasonably big circuits in a reasonable amount of CPU time. It has an interactive input for initially defining the circuit, and there is a rather straightforward way of coupling it to our program in a way that will satisfy the requirements discussed in the previous paragraphs.

However, there is always the problem of how fast the system will respond to calculate a new experiment. With simple calculation we have seen that a fairly small circuit needs 5 to 10 seconds of CPU time to be solved by the simulator. If we assume that we need 5 to 10 iterations to conclude one experiment (that is to find a new point on the screen), it is obvious that we need about one minute of CPU time per point. If the system is loaded this could very easily be 10 minutes of real time.

It is obvious that the slow reaction of the system does not favor the

interactivity of our program. Thus the need of rewriting all the software in a faster language (like *Pascal*) seems to be vital for the real-life application of the algorithm.

3.3 Example

In order to illustrate the application of the algorithm presented in Chapter 2, we will present and solve a sample engineering design problem in this section. The problem consists of making the "best" decision on the values of some designable parameters in order to achieve the "best" output characteristics in a MOSFET NAND gate design [Lightner 79] .

3.3.1 Problem Description

The problem is defined by a given circuit topology (see Fig. 3-3), the given model we are going to use to represent the MOSFET transistor element (see table 3-1), and the set of preset parameter values (see table 3-2).

The design objectives are to minimize the delay and the area of the gate, while satisfying a set of constraints on the range of the designable parameters, the maximum allowable delay and area, and the maximum minimum allowable ON output voltage to ensure the proper noise immunity[18].

3.3.2 Mathematical Formulation

The mathematical formation of the problem is derived by the following equations:

1. Objectives

   a. The first objective is the time delay (minimize)[19]

$$\Phi_1 = A_F(L_1/W_1) \tau a \qquad (3.3)$$

---

[18] The ON output voltage was the third objective in the original problem. Here we take care of it just by using it as a constraint to ensure that it will not take values close to the OFF output voltage.

[19] For more details on the methodology used to determine and calculate the delay, the reader should consult Lightner79 (pages 155-157).

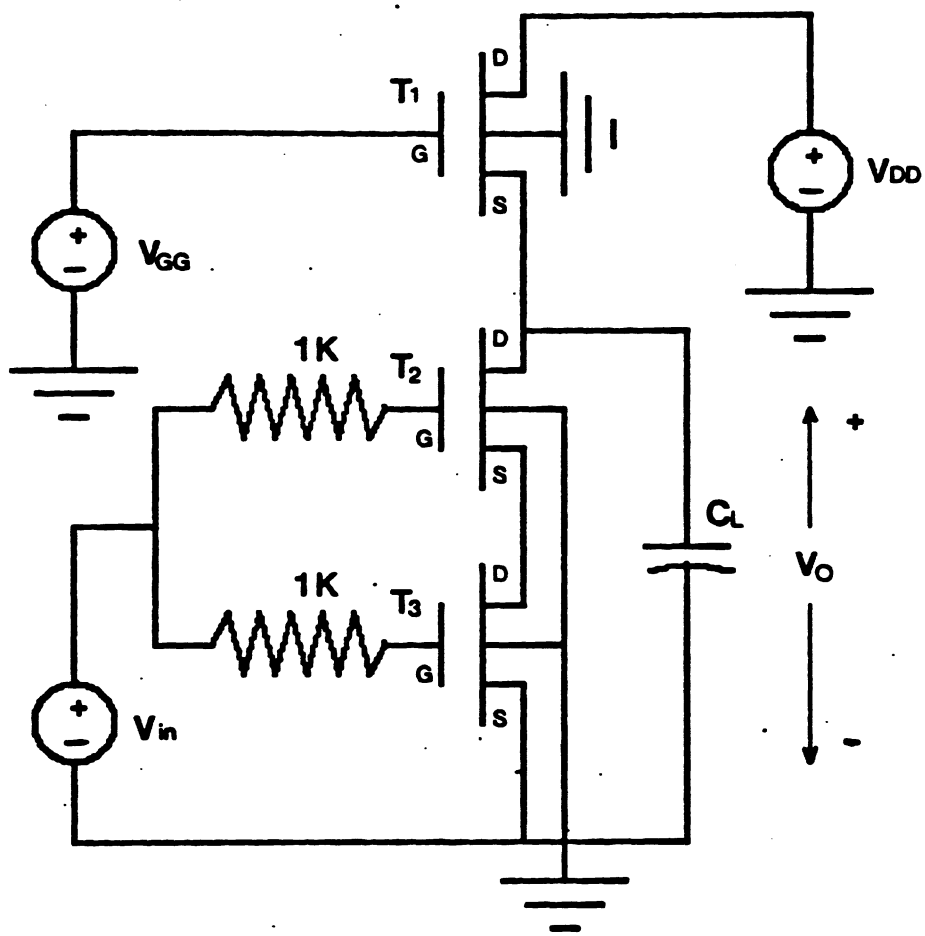

Figure 3-3: The NAND gate topology.

JDS « GM'WL'fVGS-VT)^2 above pinch-off

JDS * GM»WL*VDS»(VGS-VT-VDS/2) below pinch-off

YTsV +k*(VSSUB + PSIf
FB

where

GM = Normalized transconductance
WL = Width-to-length ratio of the device
VT s Gate threshold voltage
VGS s Gate-to-substrate voltage
YDS s Drain-to-source voltage
VFB = Flat band voltage
k s constant
PSI = Electrostatic potential on surface
VSSUB s Source-to-substrate voltage

Table 3-1: MOSFET model equations.

b. The second objective is the area occupied by the integrated gate (minimize)

$$\Phi_2 = W_1 L_1 * {}^2 L_{23} W_{23} \quad (3.4)$$

2. Constraints

a. The first constraint is to secure the lower limit of the ON output voltage. The calculation is based on an iterative DC analysis of the circuit.

$$V_o * -0.7 \text{ Volts} \quad (3.5)$$

b. The second constraint is to limit the maximum delay time allowed.

$$\bullet_1 \pounds 110 \text{ nsec} \quad (3.6)$$

c. The third constraint is to limit the maximum integration area allowed.

PSIs.5771	
* " •*	Designable parameters
GMs.006	
., - • , •	VFB.W1.W23
LI «12.7 microns	
L23 s 5.08 microns	
V00 s -6.5 volts	
VG6*-14.5 volts	
Vinx.14.5 volts	
Af« 1.03657	
CLsSpF	

Table 3-2: Problem Parameters.

$$\bullet_2 * 2500 \text{ mils}^2 \quad (3.7)$$

d. The fourth constraint has to do with physical limits on the *Flat-Band Voltage* values

$$\bullet 2 \text{ i } V_{ffl} \text{ S } -1 \quad (3.8)$$

e. The fifth constraint in to restrict the width of the *load transistor* (1).

$$5 \text{ £ } W_1 \text{ £ } 50 \quad (3.9)$$

f. And the last constraint is to restrict the width of the *gate transistors (2* and 3).

$$50 \text{ £ } W_{23} \text{ £ } 250 \quad (3.10)$$

In order to make this form suitable for solution by the *Han-Powell* algorithm, we rewrite the objectives:

$$\bullet'_1 * 110 - \bullet_1 \quad (3.11)$$

$$\bullet *_2 \ll 2500 - *_2 \qquad (3.12)$$

After that we have to rewrite the constraints to be of the form proposed in (3.1). To get to the final form of the problem (see equation (3.2)) . we have to add one more variable *y* and two more constraints. This final form is: -

$$\min_{7.w_1.w_{23}.v_{pB}} r_w w \quad v \quad y \qquad (3.13)$$

such that:

$$_r - (b - \{+\ - l_y) I (\underline{1},- I J) * 0$$

$$y - ((\bar{z}_2 - \Phi'_2) / (\bar{z}_2 - z_2)) \geq 0$$

$$V \bullet 0.7 \geq 0$$

$$\bullet^{o} \wedge i. 0$$

$$\bullet\bullet_2 \ i \ 0$$

$$-1 - V_{FB} \geq 0$$

$$V^{2 i O}$$

$$50 - W_1 * 0$$

$$W1 - 5 \wedge 0$$

$$250 - W_{23} \ 2 \ ^{0}$$

$$W23 - 50 \ 2 \ 0$$

3.3.3 Sample Run

We present here two sample runs of the rectangle elimination algorithm. In both cases we have the limit of deriving our preference with at most 10 points. In Fig. 3-4 we have followed a strictly optimal strategy. The initial rectangle is derived by the points 1 and 2. These points were obtained as the cross section of two (Decision Maker defined) vertical lines and the trade-off curve. Points 3 to 7 are the result of a 5-partitioning of the initial rectangle. Then the Decision Maker had chosen the rectangle derived from points 5 and 6 as the new area of interest and devoted the three experiments left in a 3-partitioning of this rectangle. The formal definition of this sequence (except of points 1 and 2) is $S_2(k_5,k_3)$ (see Definition 12 at Chapter 2).

In Fig. 3-5 we still have the same restriction* but the decision maker now follows a heuristic scheme. Thus, points 1, 2, 3, 9 and 10 are results of independent experiments. The rest points are results of two optimal sequences ($S_1(k_2)$ and $S_1(k_3)$ with initial rectangles 1-3 and 3-2 respectively).

In the next section we will apply the same example to discuss the expansion of this algorithm to multidimensional spaces.

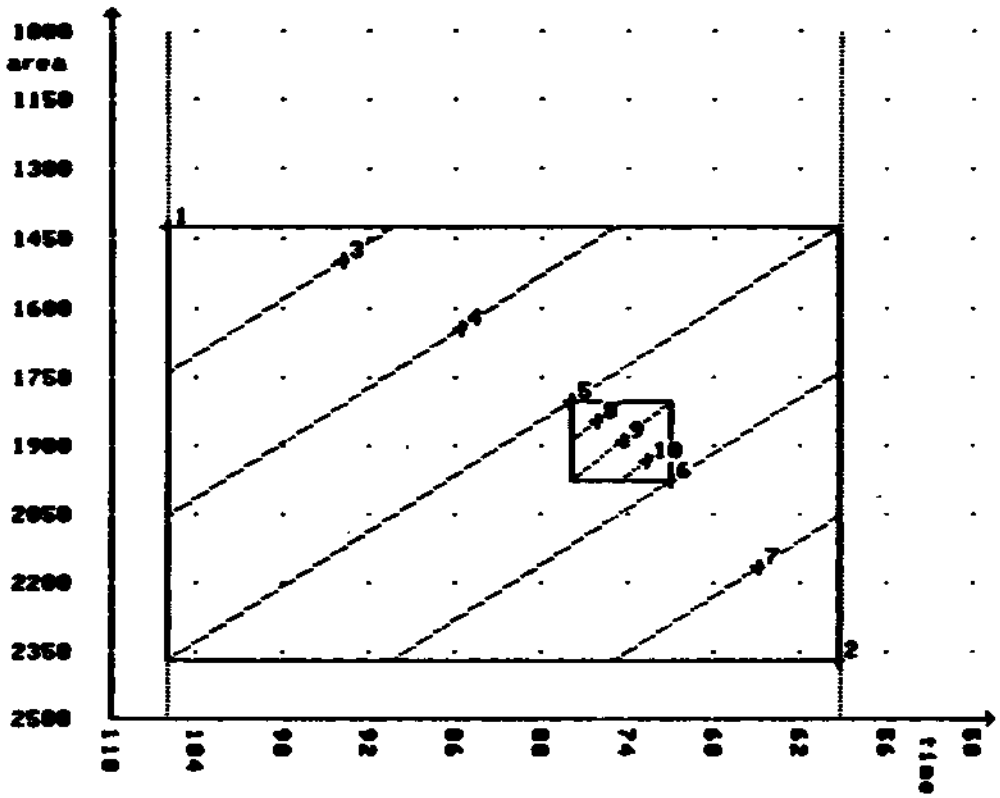

Figure 3-4: Nand gate sample run - Optimal strategy.

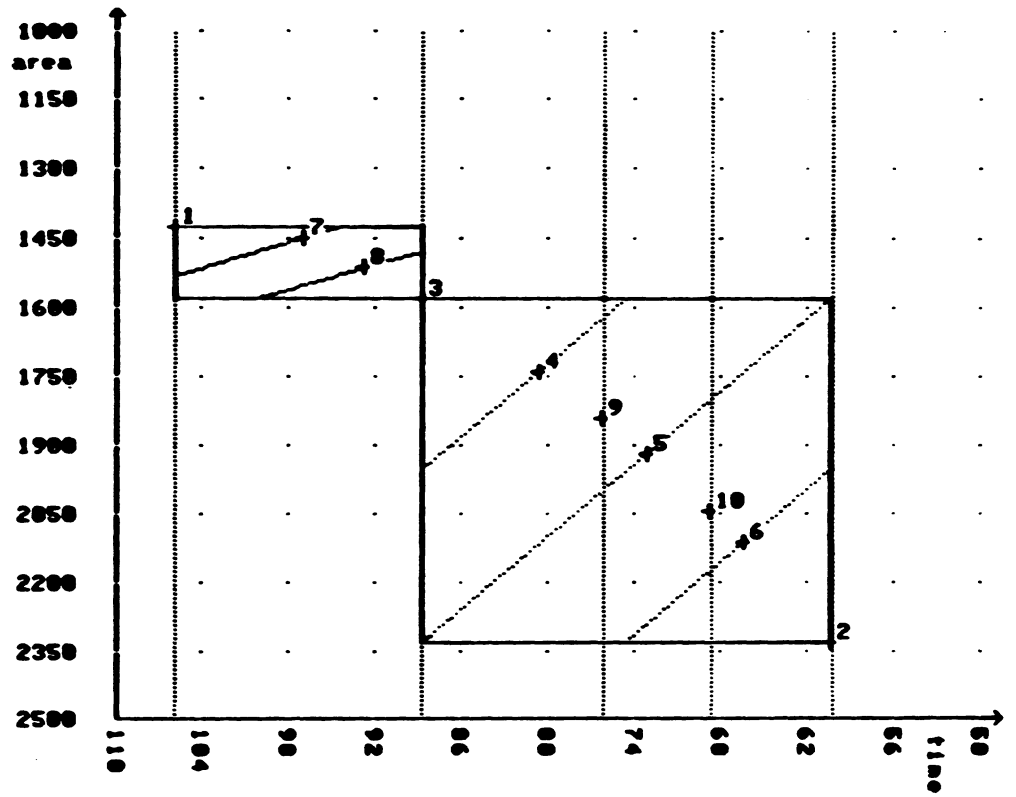

Figure 3-5: Nand gate sample run - Non optimal strategy.

CHAPTER 4
MULTIDIMENSIONAL APPLICATION

In this chapter we consider expansion of this algorithm by presenting an example of a multiple dimensional application. We begin with a short discussion of the concept of a multiple dimensional experiment.

4.1 The Multiple Dimensional Experiment

In order to move from the two dimensional space to the multiple dimensional one, we have to consider some new aspects. The first is the problem of *adequate representation* or *user perception* of the multidimensional space.

The *Decision Maker* will have to perceive the information concerning the multiple dimensional space through the use of a two dimensional screen. In order to simplify this task, we decided to have the user explore this space by means of two dimensional cross sections of it. These cross sections can in general be oriented in an arbitrary way in the n-dimensional space. However, in order both to make our programming task easier, and to keep the information on the screen clear and consice, we will use cross sections *orthogonal* to all the non displayed axis. This simply means that we keep all other objectives (except the two we are trading) constant[20].

Another problem is that of *storing* and *retrieving* the multiple dimensional points calculated thus far. This problem is solved by storing this information by *groups*. Each *group* contains the points which belong to a separate plane. The group is stored along with a label identifying the plane it represents. The user will be able to retrieve these groups, and, even superimpose groups which represent parallel planes.

---

[20] There is the opportunity to keep an objective not constant, but limited with upper and lower bounds that are close enough to be considered essentially the same from the Decision Maker.

The last aspect that we have to tackle is that of the mathematical formulation of the problem. We need a proper way of *informing* the numerical package about the current "plane of interest" without having to rewrite the equations. This is accomplished by creating a "flexible" mathematical form that will be internally *modified.* Suppose that our original problem has the form:

max { y,(p),yjp),~.,y (p)) (4.1)

such that : g(p) * 0 i * 1,_,r

where p_ i * 1,~,m are the designable parameters. Now assume that we decide to move into the s,t *plane of interest.* This means that we will be allowed to vary only the y_ and y_ objectives while keeping all other constants at predefined values. The form of the problem now becomes:

max { y$(p), y^p) } (4.2)

such that :

g(p) 2 0 i • 1,~.,r
y'(p) * V_ j « 1,~.. s-1, s*1_. t-1, t*1,~,n (1 S s < t £ n)

If this problem is put in the form we discussed in the previous chapter, it will be:

min_ *y* (4.3)

such that :
b - (y (p) - z_) / (z_ - z_) * 0

(z_ - y_(p) 1 / l f ^ l i O

and :
g_(p) **i 0 i «** U r

y (P) r. v. j « 1,~, s-1. s+1,..., t-1, t+1,.~.n (1 £ s < -t £ n)

Another form similar to this, but computationally more convenient, can be **derived.** This riew form is based on the fact that very often the Decision **Maker** is not interested on exact values for some secondary objectives. Thus, these objectives can be taken into account by defining an *area of tolerance ,* rather than specific values. The formulation of the problem now becomes:

max { $y_s(p)$, $y_t(p)$) (4.4)

such that :

$g_i(p)$ i O i « ' 1_.r

$y_j(p)$ « V_j j f S

V_{-q} £ y_Q £ V_q q € S'

with S U S' * { 1,-.. s-1. s*1.~. t-1, t*1,.~.n }

and 1 £ s < t £ n

And the final form is (the one proposed in (3.2) h

min$_{p.y}$ y (4.5)

such that :

b - < $y_{\$}$<p) - $\bar{z}_s$) / (2 - f$_{\sim s}$ s ≥ 0

($\bar{V}$ y_t<p)) / (K*> * o

and:

9_S<P) *0i = 1,...,r

y_jXp) * V_j j « S

V $_{—q}$ £ y_Q 2S V_q q « S'

with S U S' * { 1,.... s-1, s+1,.... t-1, t+1,....n }

and 1 ^ s < t £ n

These formulations are used in the the multidimensional example that follows.

4.2 Three Dimensional Example

We will now present the solution of the NAND gate which is described in Chapter 3. The search is now performed in the three-dimensional space with objectives: the delay time (to be minimized) the **area** (to be minimized) and the ON output voltage (to be maximized). The mathematical formulation of the problem is basically the same as the one described in (3.13). The main difference is that now we consider a third objective $\bullet_3$:

$$\bullet_{-} \text{ « } -V \qquad \textbf{(4.6)}$$

Since we want all objectives to be in a maximizable form, we rewrite $\bullet_3$ to get a new objective. This objective is derived in a way that will simplify the expression of the constraint:

$$V_o \; * \; -0.7 \qquad (4.7)$$

that ensures the noise immunity of the gate. The third objective of our final formulation is:

$$\bullet'_3 \text{ « } 0.7 - *_3 \qquad (4.8)$$

We now have three[21] alternative constraint sets and every one of them represents a search in a different group of parallel planes.

Thus we have one set of constraints that represents all the planes orthogonal to the *Delay* (group A), a group for planes orthogonal to the *Area* (group B) and a group for planes orthogonal to *Output Voltage.* In every group plane definition there is a parameter (in direct correspondence with the non-defined objective) that specifies the specific plane out of the group. These (three) parameters have the name P_{sk} where s and k are the subscripts of the variable objectives in each group. The formulation is:

$$\min_{y, w_1, w_{23}, v_{FB}} y \qquad (4.9)$$

---

[21] It is obvious that there ere six possible sets. However we assume that the sequence of the two objectives in one group definition is not important and we arbitrarily choose one of the possible two.

such that:

Group A (Trade-off between area and output voltage. The delay is kept constant at 110 - P_{23} ns.)

y ' (b - <*'$_3$ - f_3) / <i_3- 2$_3$)) * 0

y - < (i_2- •'j) / <V 2_2>) * 0

Φ'_1 = P_{23}

•'$_2$ * 0

Φ'_3 ≥ 0

-1 - v_{FB} * 0

V_{FB}• 2 2: 0

50 - W_1 ≥ 0

W_r 5 *i* 0

250 •- W_{23} * 0

w_{23}" ' 50 ≥ 0

Group B (Trade-off between delay and output voltage. The area is kept constant at 2500 - P_{13} mils^2.)

y - (b - M'$_1$ - z$_1$) / (2^ 2^) ^ 0

T ~ ((f_3 - 1'$_{^3}$ } / (z$_3$-2$_3$)) ^ 0

Φ'_1 *i* 0

V B P_{13}

*3 ' *t* 0

-1 - v_{FB} ≥ 0

v_{FB}^• 2 * 0

50 v• w_i ≥ 0

W_r 5 2 0

250 - W_{23} 2 0

W_{23} - 50 ≥ 0

Group C (Trade-off between delay and area. The output voltage is kept constant at P_{12} - 0.7 volts.)

$\gamma - (b - (\Phi'_1 - \bar{z}_1) / (\underline{z}_1 - \bar{z}_1)) \geq 0$

$\gamma - ((\bar{z}_2 - \Phi'_2) / (\bar{z}_2 - \underline{z}_2)) \geq 0$

$\bullet\bullet_1 * 0$

•1 - V^ i 0

$V^2 * o$

50 - W, 2 **0**

W1 - 5 * 0

250 - W_{23} * 0

W_{23} - 50 * 0

In the next pages we present a sequence of screen snapshots from this three-dimensional search. This search is done with the optimality criteria we discussed in Chapter 3. Specifically, we present three groups of parallel planes. The search at every single plane is done by an arbitrary initial rectangle definition (points 1 and 2) followed by a $S_1(k_3)$ strategy.

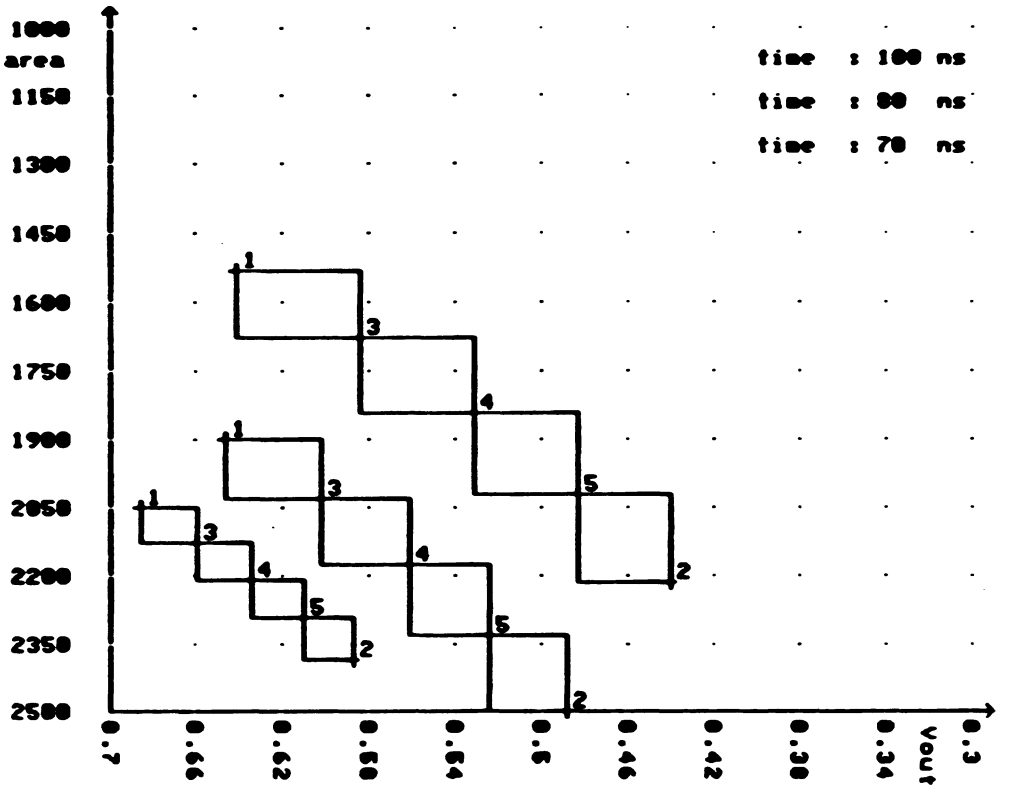

Figure 4-1: Area - Vout planes, Delay Time = 100, 80 and 70 ns

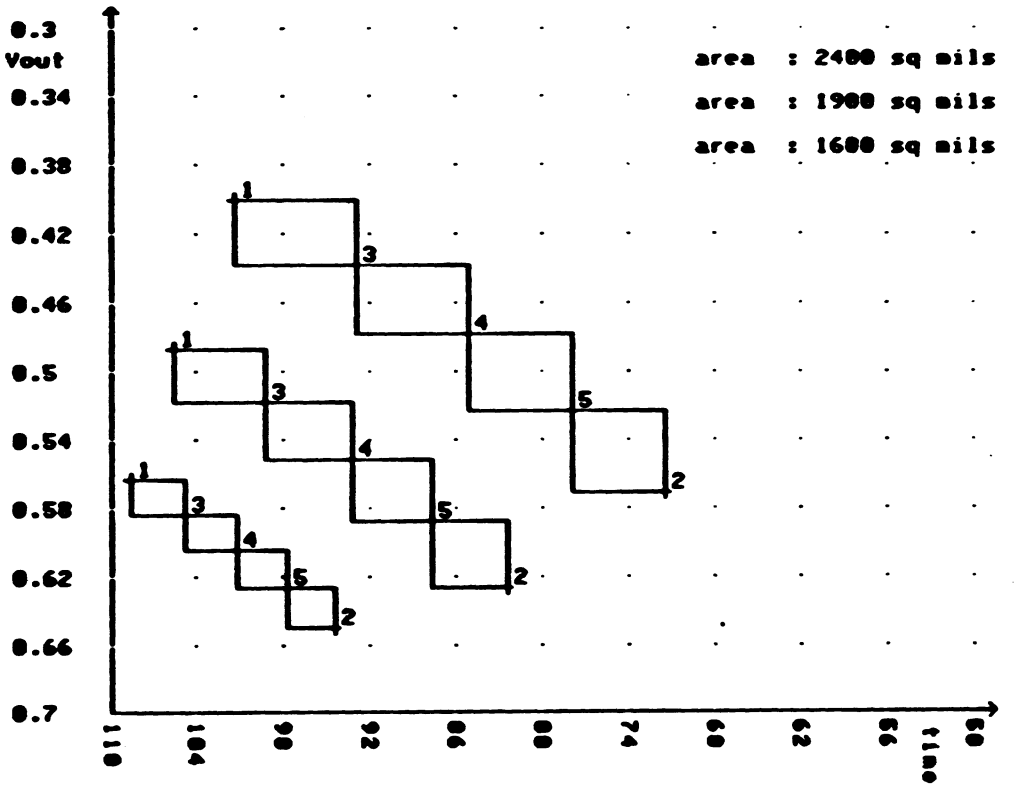

Figure 4-2: Time - Vout planes, Area = 2400, 1900 and 1600 mils^2

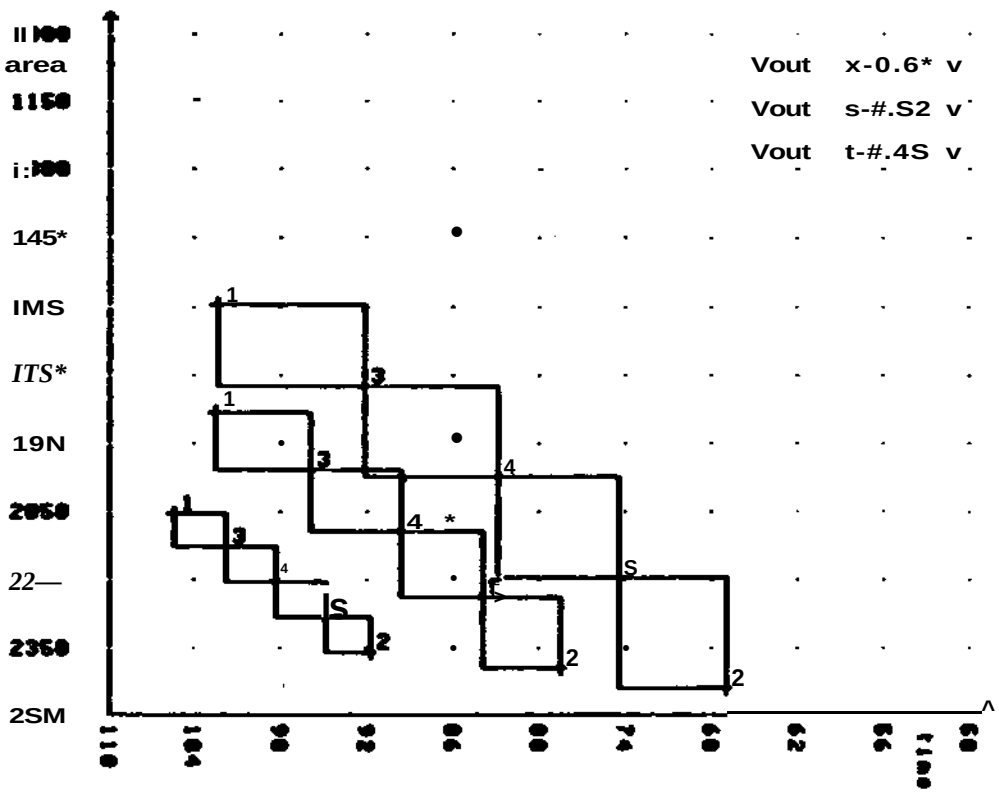

Figure 4-3: Area - Time planes, Vout « -0.60, -0.52 and -0.45 volts

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

In this work we have presented methods for exploring the trade-off curves associated with multiobjective optimization. In this chapter we will summarize our work and we will give our conclusions and some proposals for future work in this direction.

5.1 Brief Description of Work

In this work we presented the *rectangle elimination* family of methods for biobjective optimization. In the first two chapters, these methods where associated with the current mathematical background on multiobjective optimization, and the algorithm for optimal biobjective search (originally presented in [Polak 80]) is expanded.

In the third Chapter we discuss the implementation of the scheme using APL, and we present and solve a two dimensional example. Then, in the fourth Chapter, the same scheme is expanded for multiple dimensional use and a three-dimensional example is presented and solved.

5.2 General Conclusions

The main problem in this area is found to be the *perceptibility* of the image we present to the decision maker. For good interaction, the decision maker must be able to interpret the information quickly and preferably on a qualitative basis. However, the complexity of the information rises rapidly as the dimensionality of the problem increases and at some point no decision can be made without quantitative considerations.

5.2.1 Rectangle Elimination Scheme

What was said in the previous paragraph holds true for the algorithm we presented in this work. The user must not think that one can explore a set of two-dimensional cross-sections. Very soon this task will get very confusing and pointless.

In order to cope with the high dimensionality of the problem in real life engineering design, we still have to use some of the methods described in chapter 1, methods like weighted sum or minimax approach. The designer may use these methods to make up his mind for some not very important objective functions, and then ^use a more interactive method (like the one presented here) to perform the trade-off between the rest.

It is our belief that a more general scheme must be devised to employ both interactive and non-interactive methods. This scheme will first apply a non interactive method for multi objective optimization, to reduce the dimensionality of the problem to a more comfortable level, and then we can apply the interactive method for "fine tuning" our decision on the most important objectives. This can be repeated like an iterative procedure, until the decision maker has a good combination of all objectives.

5.3 Future Work Proposals

5.3.1 Multiple Dimensional Cross Sections

Sometimes it may be favorable for the user who is trying to accomplish an n-dimensional trade-off, if he explores this space by using three-dimensional cross sections instead of two-dimensional ones. This will obviously reduce the necessary number of cross sections, and it may also provide the opportunity for more fine decisions.

However, in order to implement this scheme, we must devise a way of exploring three^v dimensional spaces. This may be based on an extrapolation of the existing 2-dimensional search. The scheme will include a three-dimensional projection on the graphics screen, a more sophisticated way of interaction and

probably an optimal strategy for approximating the trade-off surface using orthogonal parallelepipeds.

The three-dimensional search will be ideal for three dimensional problems and should also be very convenient for small dimensionalities like 4 or 5. For dimensionalities higher than these though, the information presented to the user will be rather difficult to process.

5.3.2 Point Filtering and Curve Estimation

While exploring a trade-off space, the user may find it convenient to have a way of uniformly spreading points over the entire surface. This may be done using the *Point Filtering Techniques* described in [Steuer 80]. These techniques have to do with an *optimal point generation strategy* and a "point rearrangement strategy". The method may be useful when we do not really want to find a specific point on the curve, but would rather gain an insight into the general shape of the curve.

References

[Bertsekas 76] Bertsekas P. Dimitri.
Multiplier Methods: A Survey.
AUTOMATICA 12:133-145, 1976.

[Brayton 80a] Robert Spence (editor).
CAD of Electrinic Circuits. Volume 2: *Sensitivity and Optimization.*
ELSEVIER SCIENTIFIC PUBLISHING COMPANY, P.O. Box 330,1000 AH Amsterdam, The Netherlands, 1980.

[Brayton 80b] Brayton R.K. Hachtel G.D. Sangiovanni A. Vincentelli.
A Survey of Optimization Techniques for IC Design.
1980.

[Han ??] Han S. P.
A Globally Convergent Method For Nonlinear Programming.
Report No. 75-275.
1975.

[Hwang 80] Hwang C.L. , Paidy S.R. and Yoon K.
Mathematical Programming with Multiple Objectives: A Tutorial.
COMPUTER AND OPERATIONAL RESEARCH 7:5-31, 1980.
Pergamon Press, Printed in Great Britain.

[Lightner 79] Lightner M.R.
Multiple Criterion Optimization And Statistical Design For Electronic Circuits..
PhD thesis, Carnegie-Mellon University, February, 1979.

[Polak 80] Polak E. and Payne A.
An Interactive Rectangle Elimination Method for Biobjective Decision Making.
IEE TRANSACTIONS ON AUTOMATIC CONTROL ac-25(3):421-432, June, 1980.

[Powell 77] Powell M.J.D.
A Fast Algorithm For Nonlinearly Constrained Optimization Calculations.
Unpublished , June, 1977.
Presented at the 1977 Dundee Conference on Numerical Analysis.

[Steuer 80] Steuer E. Ralph.
Intra-Set Point Generation and Filtering in Decision and Criterion Space.
COMPUTER AND OPERATIONAL RESEARCH 7:41-53, 1980.
Pergamon Press ltd, printed in Great Britain.

[Wallenius 75] Wallenius Jurki.
Comparative Evaluation Of Some Interactive Approaches To Multicriterion Optimization.
*MANAGEMENT SCIENCE 2*1(9): 1387-1396. May. 1975.