2-1-1979

Multilevel Approach To Urban Water Resources Systems Analysis-Application To Medium Size Communities: Interactive Multiple Objective Optimization

K. J. Musselman

J. J. Talavage

Follow this and additional works at: http://docs.lib.purdue.edu/watertech

http://docs.lib.purdue.edu/watertech/121

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
Multilevel Approach to Urban Water Resources
Systems Analysis-Application to Medium Size Communities

INTERACTIVE MULTIPLE OBJECTIVE OPTIMIZATION

by
Kenneth J. Musselman
Joseph J. Talavage

February 1979

PURDUE UNIVERSITY
WATER RESOURCES RESEARCH CENTER
WEST LAFAYETTE, INDIANA
Water Resources Research Center
Purdue University
West Lafayette, Indiana 47907

INTERACTIVE MULTIPLE OBJECTIVE OPTIMIZATION

by

Kenneth J. Musselman
and
Joseph J. Talavage

The work upon which this report is based was supported in part by funds provided by the United States Department of the Interior, Office of Water Research and Technology, as authorized by the Water Resources Research Act of 1974 (PL88-379) as amended.

Period of Investigation: January 1977 - August 1979
Partial Completion Report for OWRT-B-083-IND
Matching Fund Agreement 14-31-0001-5213

Purdue University Water Resources Research Center
Technical Report No. 121
February 1979
ACKNOWLEDGEMENTS

The present report covers part of the work done under Matching Grant Fund OWRT-B-083-IND and was supported in part by the Office of Water Research and Technology, Department of the Interior, and in part by Purdue University. The authors wish to express their appreciation to Dr. Dan Wiersma, Director of the Purdue University Water Resources Research Center, to Dr. John F. McLaughlin, Head of the School of Civil Engineering and to Dr. Wilbur L. Meier, Jr., Head of the School of Industrial Engineering at Purdue University for their assistance in the administration of the project.

The present report is a slightly condensed version of the Doctoral Thesis "An Interactive Tradeoff Cutting Plane Approach to Continuous and Discrete Multiple Objective Optimization" submitted by K. Musselman to the Graduate Faculty at Purdue University in December, 1978, which may be obtained from University Microfilms, Ann Arbor, Michigan. The work was performed under the supervision of Professor J. J. Talavage.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>x</td>
</tr>
<tr>
<td>CHAPTER I - CONCEPTS AND TERMINOLOGY</td>
<td>1</td>
</tr>
<tr>
<td>Definition of the Problem</td>
<td>2</td>
</tr>
<tr>
<td>Preliminary Definitions and Assumptions</td>
<td>4</td>
</tr>
<tr>
<td>Tradeoffs</td>
<td>9</td>
</tr>
<tr>
<td>Methods of Solution</td>
<td>12</td>
</tr>
<tr>
<td>The Problem to be Studies</td>
<td>14</td>
</tr>
<tr>
<td>Additional Assumptions</td>
<td>15</td>
</tr>
<tr>
<td>Superiority Test</td>
<td>17</td>
</tr>
<tr>
<td>A Usable Direction</td>
<td>18</td>
</tr>
<tr>
<td>Algorithmic Convergence</td>
<td>19</td>
</tr>
<tr>
<td>Organization of the Report</td>
<td>23</td>
</tr>
<tr>
<td>CHAPARER II - THE GEOFFRION-DYER-FEINBERG INTERACTIVE</td>
<td>25</td>
</tr>
<tr>
<td>ALGORITHM</td>
<td></td>
</tr>
<tr>
<td>General Overview</td>
<td>25</td>
</tr>
<tr>
<td>Description</td>
<td>26</td>
</tr>
<tr>
<td>The Interactive Frank-Wolfe Algorithm</td>
<td>29</td>
</tr>
<tr>
<td>Significant Attributes</td>
<td>29</td>
</tr>
<tr>
<td>Some Limitations of the GDF Algorithm</td>
<td>30</td>
</tr>
<tr>
<td>CHAPTER III - THE TRADEOFF CUTTING PLANE ALGORITHM</td>
<td>33</td>
</tr>
<tr>
<td>Intent of the Tradeoff Cutting Plane Algorithm</td>
<td>33</td>
</tr>
<tr>
<td>Concepts Behind a Cut</td>
<td>36</td>
</tr>
<tr>
<td>Locating the Next Tradeoff Point</td>
<td>41</td>
</tr>
<tr>
<td>Tradeoff Cutting Plane Algorithm</td>
<td>45</td>
</tr>
<tr>
<td>Proof of Convergence</td>
<td>53</td>
</tr>
<tr>
<td>Tradeoff Sensitivity</td>
<td>68</td>
</tr>
<tr>
<td>CHAPTER IV - STORM DRAINAGE - AN APPLICATION</td>
<td>76</td>
</tr>
<tr>
<td>Overview</td>
<td>76</td>
</tr>
<tr>
<td>Original Formulation</td>
<td>80</td>
</tr>
<tr>
<td>Multiple Objective Formulation</td>
<td>80</td>
</tr>
<tr>
<td>Problem Statement</td>
<td>83</td>
</tr>
<tr>
<td>Tradeoff Policy A</td>
<td>89</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Continued)

Tradeoff Policy B ................................................. 94
Tradeoff Policy C .................................................. 101
Tradeoff Policy D .................................................. 101
Discussion .......................................................... 106

CHAPTER V - DISCRETE OPTIMIZATION VIA THE TRADEOFF
CUTTING PLANE ALGORITHM ....................................... 110

General Approach .................................................. 110
Development of the Algorithm .................................... 111
Ancillary Definitions ............................................... 118
Discrete Version of the TCP Algorithm ......................... 121
Finiteness of the Algorithm ...................................... 123
Results Relating to the Final Potential Set .................... 126
Modifications ...................................................... 134
Examples ............................................................ 141

CHAPTER VI - SUMMARY AND RECOMMENDATIONS .................. 153

Overview ........................................................... 153
Areas for Future Research ........................................ 154

BIBLIOGRAPHY ....................................................... 159

APPENDICES

Appendix A: An Amalgam of Definitions of Lemmas .......... 164
Appendix B: Method of Centers ................................... 172
Appendix C: Program Listing. ..................................... 177
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>82</td>
</tr>
<tr>
<td>4.2</td>
<td>84</td>
</tr>
<tr>
<td>4.3</td>
<td>88</td>
</tr>
<tr>
<td>4.4</td>
<td>91</td>
</tr>
<tr>
<td>4.5</td>
<td>92</td>
</tr>
<tr>
<td>4.6</td>
<td>93</td>
</tr>
<tr>
<td>4.7</td>
<td>102</td>
</tr>
<tr>
<td>4.8</td>
<td>103</td>
</tr>
<tr>
<td>4.9</td>
<td>104</td>
</tr>
<tr>
<td>4.10</td>
<td>105</td>
</tr>
<tr>
<td>4.11</td>
<td>107</td>
</tr>
<tr>
<td>Appendix Table</td>
<td></td>
</tr>
<tr>
<td>C.1</td>
<td>178</td>
</tr>
</tbody>
</table>

4.1 Analytical Model of Subbasin 13's Storm Drainage Problem ........................................... 82
4.2 Regression Models of the Simulation Output Parameters ..................................................... 84
4.3 Bounds Associated with Constraints $e_l$ through $e_{35}$ .................................................. 88
4.4 Sequence of Points Generated from SP1 under Policy A ....................................................... 91
4.5 Sequence of Points Generated from SP2 under Policy A ....................................................... 92
4.6 Sequence of Points Generated from SP3 under Policy A ....................................................... 93
4.7 Sequence of Points Generated from SP1 under Policy B ....................................................... 102
4.8 Sequence of Points Generated from SP2 under Policy B ....................................................... 103
4.9 Sequence of Points Generated from SP3 under Policy B ....................................................... 104
4.10 Sequence of Points Generated under Policy C ........................................................................ 105
4.11 Sequence of Points Generated under Policy D ........................................................................ 107

C.1 FORTRAN Representation of Terms ............................................. 178
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>Objective Space</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Decision Space</td>
<td>39</td>
</tr>
<tr>
<td>3.2</td>
<td>Objective Space</td>
<td>39</td>
</tr>
<tr>
<td>3.3</td>
<td>Translated Cut in Decision Space</td>
<td>40</td>
</tr>
<tr>
<td>3.4</td>
<td>Flowchart of the TCP Algorithm</td>
<td>48</td>
</tr>
<tr>
<td>3.5</td>
<td>Tradeoff Sensitivity in the Case of Two Objectives</td>
<td>72</td>
</tr>
<tr>
<td>3.6</td>
<td>Tradeoff Sensitivity Printout</td>
<td>73</td>
</tr>
<tr>
<td>4.1</td>
<td>Conceptualization of Subbasin 13's Storm Drainage Problem</td>
<td>78</td>
</tr>
<tr>
<td>4.2</td>
<td>Response Surface and Regression Model of the Expected Economic Loss Due to Flooding</td>
<td>85</td>
</tr>
<tr>
<td>4.3</td>
<td>Response Surface and Regression Model of the Average Number of Pounds per Year of Suspended Solids</td>
<td>86</td>
</tr>
<tr>
<td>4.4</td>
<td>Total Cost of the Drainage System</td>
<td>90</td>
</tr>
<tr>
<td>4.5</td>
<td>Drainage System Cost as a Function of the Iteration Number</td>
<td>95</td>
</tr>
<tr>
<td>4.6</td>
<td>Drainage Network Cost Under Policy B</td>
<td>96</td>
</tr>
<tr>
<td>4.7</td>
<td>Storage Facility Cost Under Policy B</td>
<td>97</td>
</tr>
<tr>
<td>4.8</td>
<td>Treatment Facility Cost Under Policy B</td>
<td>98</td>
</tr>
<tr>
<td>4.9</td>
<td>Expected Flood Damage Cost Under Policy B</td>
<td>99</td>
</tr>
<tr>
<td>4.10</td>
<td>Expected Economic Loss Due to Flooding Under Policy B</td>
<td>100</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES (Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Tradeoff Cut Elimination Scheme</td>
<td>114</td>
</tr>
<tr>
<td>5.2</td>
<td>Inability to Locate the Best-Compromise Solution</td>
<td>116</td>
</tr>
<tr>
<td>5.3</td>
<td>The Plausible Set Ω = { B, C }</td>
<td>120</td>
</tr>
<tr>
<td>5.4</td>
<td>Flowchart of the TCP Algorithm (Discrete Version)</td>
<td>124</td>
</tr>
<tr>
<td>5.5</td>
<td>Variation in the Final Potential Set</td>
<td>135</td>
</tr>
<tr>
<td>5.6</td>
<td>Pseudocut Illustration</td>
<td>137</td>
</tr>
<tr>
<td>5.7</td>
<td>Hypercube Caveat</td>
<td>140</td>
</tr>
<tr>
<td>5.8</td>
<td>Tradeoff Cuts Connected with the Discrete Version of the TCP Algorithm</td>
<td>144</td>
</tr>
<tr>
<td>5.9</td>
<td>First Iteration Pseudocut</td>
<td>147</td>
</tr>
<tr>
<td>5.10</td>
<td>Tradeoff Cuts Connected with the Pseudocut Routine</td>
<td>149</td>
</tr>
<tr>
<td>5.11</td>
<td>First Iteration of the Hypercube Routine</td>
<td>150</td>
</tr>
<tr>
<td>5.12</td>
<td>Second Iteration of the Hypercube Routine</td>
<td>152</td>
</tr>
</tbody>
</table>

### Appendix

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>Locating $z^{i+1}$ from $z^i$</td>
<td>176</td>
</tr>
</tbody>
</table>
ABSTRACT

Decision problems often arise in which several, non-commensurate and conflicting objectives must be considered simultaneously. The subject of this report is concerned with first investigating a general nonlinear class of these problems and then developing a means of solving problems found in this class. The results of this research are divided into four related areas:

(1) identifying properties and relationships which exist within this class of problems and which prove useful to their resolution,

(2) developing a user-oriented algorithm, using these properties, to solve nonlinear multiple objective optimization problems,

(3) applying the algorithm to a representative problem, and

(4) modifying the algorithm to address discrete nonlinear multiple objective optimization problems.
In identifying the properties and relationships which exist between the decision space and the objective space, insight is gained into how the decision maker can avoid considering alternatives which are inferior to his present state. It is shown, by means of a "tradeoff cut", how the feasible region in decision space is able to be reduced without jeopardizing the final (best-compromise) solution.

The concept of a tradeoff cut is fundamental to the optimization technique developed in this dissertation. This optimization technique, termed the Tradeoff Cutting Plane algorithm, uses these cuts to reformulate the original multiple objective problem into a series of pairwise decisions. By repeatedly locating a feasible solution and then modifying the problem by means of these tradeoff cuts, one converges to the best-compromise solution. The algorithm makes prudent use of the decision maker by closely matching his ability to supply the information necessary to unify the various objectives. Furthermore, upon termination, the algorithm conducts a sensitivity analysis of the decision maker's inputs supplied during the solution process.

A decision problem concerned with the expected storm drainage needs of an urban subbasin is analyzed by means of this algorithm. Specifically, the problem involves minimizing various pollutant loads and costs by adjusting the drainage system's local detention storage capacity, maximum treatment rate, and maximum allowable overflow rate. Convergence results are shown for several tradeoff policies.

Finally, the Tradeoff Cutting Plane algorithm is modified to resolve the question of how to find integer solutions to these non-linear multiple objective optimization problems. The peculiar difficulties involved with directly applying the continuous version
of the algorithm are discussed in detail. It is also explained how the modified version of the algorithm is able to circumvent these difficulties.
CHAPTER I

CONCEPTS AND TERMINOLOGY

Decision makers, be they individuals or collective bodies, are frequently beset with having to find solutions to large-scale problems. These problems, found both in the public and private sector, have become more and more complex due to the active participation of public-interest groups, legislative bodies, and the public media. This widening of the participating community creates the incentive to individually represent and treat those factors relevant to the various interest groups. As a consequence, these problems become characterized by multiple, non-commensurate, and often conflicting objectives. For example, water resource projects must often be concerned about flood control, water quality, land use, erosion and sedimentation, fish and wildlife, outdoor recreation, flooding and drainage, water supply, water and land policy, and so forth. As a result, decision making has become increasingly more difficult.

Being inundated with such a vast amount of information, the decision maker often chooses to either ignore or oversimplify the original problem [13]. He begins to disregard relevant considerations in order to avoid the multiple objective aspect of the problem. So as the number of objectives increases, the ability of the decision maker to comprehend and correctly solve the problem decreases. With the volume of information exceeding his ability to process it, his own personal values are
inclined to suppress those of the organization, be it a public committee, a private firm, a government group, a university, or some similar body. One can easily recognize that this type of approach has a tendency to serve the needs of the decision maker more than the needs of the organization.

How then should one systematically analyze these complex problems?

A great deal of effort has been and is now being devoted to the development of various methods to assist the decision maker in these situations. Techniques range from those which consider only one objective at a time to those which look simultaneously at all the objectives. A discussion of these methods is deferred until the problem and the types of solutions associated with it are more clearly defined. Nonetheless, it can be stated that these methods all share the common goal of formulating a means to consider more than one objective in a way that is theoretically sound and operationally efficient.

1.1. Definition of the Problem

The type of problem we have been discussing is generally termed a multiple objective optimization problem. It concerns the solution of mathematical programming models with more than a single objective. Simply stated, the object is to maximize \( r \) \((r > 1)\) distinct objective functions defined over the set of feasible decisions. Numbering the objectives from 1 to \( r \) reflects no relative importance among them. In mathematical terms the problem is defined to be:
maximize \( f(x) = (f_1(x), f_2(x), \ldots, f_r(x)) \) \hspace{1cm} (1.1)
subject to
\[ g_j(x) \leq 0, \quad j=1,2,\ldots,m \] \hspace{1cm} (1.2)

where \( x \) is an \( n \)-dimensional vector of decision variables.

\( f_i(x) \), \( i=1,2,\ldots,r \), are \( r \) distinct objective functions.

\( g_j(x) \), \( j=1,2,\ldots,m \), are \( m \) constraint functions.

Since equations can be replaced by two inequalities in opposite directions, equality constraints will not be explicitly represented.

Although many writers prefer to distinguish among attributes, criteria, objectives and goals, this author will not hold to any distinction. Furthermore, when examining complex problems the objectives are occasionally subjective in nature and are not known with any certainty. Here, however, it is assumed that the objectives are precise analytical expressions which accurately reflect a decision's worth or value. Associated with any point in decision space there is a certain known consequence in objective space. The analysis is assumed to start at a stage where the decision situation has been explicitly defined -- the various alternatives have been identified and are able to be evaluated against known objectives.

Throughout this analysis it is assumed that there is a well-specified, identifiable, unitary decision maker. This is not to deny that many problems are incapable of being resolved in this manner. One need not search very far to find numerous examples where the decision involves an intricate composite of opposing individual viewpoints. Approaches which ostensibly recognize multiple decision makers are discussed by Sen [50], Fishburn [18], Haines and Hall [25], and Keeney
and Raiffa [34]. However, at some point in most decision processes, a single decision maker (or a group of individuals that may be viewed as a single decision maker) is called upon to make a choice. Thus, limiting the scope of this report to only those situations involving single decision makers is not excessively restrictive.

1.2. Preliminary Definitions and Assumptions

First, attention is focused on what is termed the feasible region of the decision maker's "decision space".

**Definition. (Feasible Region in Decision Space)**

The region defined by the constraint set (1.2) in n-dimensional Euclidean vector space,

\[ X = \{ x \mid g_j(x) \leq 0, \quad \forall j \} \]

will be referred to as the feasible region in the decision space.

The r-dimensional objective function maps the feasible region in decision space \( X \) into the feasible region in objective space \( f(X) \), defined on the r-dimensional Euclidean vector space.

**Definition. (Feasible Region in Objective Space)**

The set

\[ Y = \{ y \mid y = f(x), \quad x \in X \} \]

will be referred to as the feasible region in the objective space.
Ultimately, the decision maker seeks a course of action represented by some $\bar{x}$ in $X$. However, it will be the feasible region $Y$ which will be of most immediate concern to him for it is the points in $Y$ which serve to indicate the levels of the objectives.

Since it is ambiguous to speak of optimizing $r$ objective functions, it is important to clarify what is meant by an optimal solution and further explain the type of solution one hopes to reach.

**Definition.** (Superior Solution)

A superior (optimal) solution is one which maximizes all the objectives simultaneously; a solution $\bar{x}^0$ to problem (1.1)-(1.2) is said to be superior if and only if $\bar{x}^0 \in X$ and $f(\bar{x}^0) \geq f(x)$ for all $x \in X$.

It often arises, however, that objectives will conflict with each other in that improvement in any one objective is done at the expense of some other objective. The decision maker is faced with the problem of having to decide how much to sacrifice of one objective in order to gain in another. It becomes a problem involving value tradeoffs among the objectives. Later, it will be shown how one is able to constructively use these tradeoffs to resolve this problem of conflicting objectives.

It is assumed that the decision maker prefers higher values of all the objectives to lower values. If an objective is used in which lower values are preferred to higher values, then the corresponding objective function is defined as its negative. In this way, higher values will always be the preferred values. Also, the relative preferences that the decision maker has for points in $X$ are based entirely upon their respective functional values in $Y$. The following definition will help clarify this last statement.
Definition. (Preferred)

If \( f(x^1), f(x^2) \in Y, f(x^1) \geq f(x^2), \) and \( f_1(x^1) > f_1(x^2) \)

for some \( i=1,2,\ldots,r, \) then \( x^1 \) is preferred to \( x^2. \)

For this preference relation to hold, it is assumed that each objective is ordinarily independent of the others. This means that if

\[
(f_i, \overline{f}_i) \succeq (\overline{f}_i, f_i),
\]

then

\[
(f_i, \overline{f}_i) \succeq (\overline{f}_i, f_i),
\]

where \( \overline{f}_i \) is the complementary set of objectives with respect to \( f_i, \)

and \( \overline{f}_i \) represents a different set of objective values than \( \overline{f}_i. \)

From the above definitions and assumptions it becomes clear that the decision maker should restrict his attention to what are called efficient solutions.

Definition. (Efficient Solution)

An efficient (Pareto-optimal, noninferior, nondominate, admissible) solution is a feasible solution, \( x^* \in X, \) for which there does not exist any other feasible solution, \( x \in X, \) such that \( f(x^*) \leq f(x) \) and \( f_i(x^*) < f_i(x) \) for some \( i=1,2,\ldots,r. \)

In other words, an efficient solution is a feasible solution for which there exists no other feasible solution which is preferred to it.

Clearly, an efficient solution is, in general, not unique. The collection of all efficient solutions, \( X^*, \) is a subset of the feasible
region in the decision space $X$. The objective function $f$ maps these efficient solutions into what is referred to as the efficient frontier $f(X^*)$. Note that the efficient frontier is defined in the objective space and $f(X^*) \subset f(X)$.

Without further preference information regarding the objective functions $f_i$, a complete ordering of the feasible solutions cannot be obtained. A complete ordering is possible, however, if the decision maker is willing to indicate the points in objective space for which he is indifferent. With this added preference information, a continuous, real-valued function $U$ can be defined on this space for which $U(f_1) = U(f_2)$ if and only if $f_1$ is indifferent to $f_2$ [32]. The function $U$ is called an overall preference function.

It has already been assumed that the decision maker prefers higher values of any objective. This assumption is termed non-satiation. Assuming $U$ is differentiable, this means that all $1^{st}$ order partial derivatives of the overall preference function $U$ are positive. That is,

$$\frac{\partial U}{\partial f_i}(f(x)) = \left( \frac{\partial U}{\partial f_1}(f(x)), \ldots, \frac{\partial U}{\partial f_r}(f(x)) \right) > 0.$$ 

Another assumption required of the overall preference function is for it to be concave. Usually, to guarantee the concavity of $U$ on $X$, one of the following two conditions are assumed to exist: either (i) $U$ is concave on $Y$ and each $f_i$ is linear on $X$, or (ii) $U$ is concave increasing on $Y$ and each $f_i$ is concave on $X$. Since condition (ii) is the situation commonly encountered in practice (i.e., non-satiation), it will be the one assumed in this analysis.
These restrictions on U are primarily made in order to simplify
discussion and lend some exactness to the subject. They do not
normally impose any significant practical limitations on the decision
maker.

Previously, it was shown that a decision maker should focus his
attention of efficient solutions. However, without additional pre-
ference information regarding the objective space, one particular
efficient solution cannot be considered preferable to any other. In
the presence of the overall preference function U, the "best" efficient
solution can now be identified.

Although some authors choose to use the term "optimal" or "preferred"
for this "best" solution, this author will use the phrase "a best-
compromise solution". Originally coined by Belenson and Kapur [3], it
alludes to the fact that such a solution is "best" only in terms of
the particular preference function used.

Definition. (Best-Compromise Solution)

A best-compromise solution is an efficient solution which
is chosen as the final decision based on the decision maker's
overall preference function U.

By fixing U at some constant value a, an indifference curve
(for U = a) can be established between any two objectives. For example,
the indifference curve between \( f_2 \) and \( f_4 \) is constructed by varying
\( f_2 \) and \( f_4 \) while holding U and the other objective functions (i.e.,
\( f_1, f_3, f_5, f_6, \ldots, f_r \)) fixed. The resultant indifference curve, which
is in two-dimensional space, has certain properties associated with it.

Taken from Baumol [1], they are as follows:
P1) An indifference curve, which lies above and to the right of another, represents preferred combinations of the objectives.

P2) Indifference curves have a negative slope.

P3) Indifference curves can never meet or intersect, so that only one indifference curve will pass through any one point (in the objective space).

P4) The absolute slope of an indifference curve diminishes toward the right. Consequently, an indifference curve is convex to the origin.

Figures 1.1 and 1.2 illustrate some of the concepts defined to this point. An example of a feasible region in a two-dimensional decision space is shown in Figure 1.1. Also shown in the figure is the best-compromise solution. Figure 1.2 pictures the corresponding feasible region in objective space, the feasible region Y, the efficient frontier, indifference curves, and the best-compromise.

1.3. Tradeoffs

Indifference curves are used to depict the tradeoffs which exist between two objectives. The negative slope of the tangent to these indifference curves is termed the marginal rate of substitution or tradeoff ratio.

Definition. (Tradeoff Ratio)

The tradeoff ratio (marginal rate of substitution) between objectives \( f_1 \) and \( f_2 \) at a point \( x^o \) in Y is the ratio
Figure 1.1 Decision Space

Figure 1.2 Objective Space
\[
\begin{align*}
\frac{\partial u}{\partial r_2} & \quad | \quad f(x) = f(x^0) \\
\frac{\partial u}{\partial r_1} & \quad | \quad f(x) = f(x^0) .
\end{align*}
\]

To further illustrate, a tradeoff ratio of \( r \) is the marginal rate of substitution of objective \( f_1 \) for \( f_2 \). It means that at the current point \( f(x^0) \) the decision maker is indifferent to a decrement of \( r \) units of \( f_1 \) provided \( f_2 \) is incremented by one unit. This ratio will, in general, depend on the levels of both \( f_1 \) and \( f_2 \). It might also depend on the levels of the other objective functions.

Independence between the (conditional) indifference curves of two objectives and the levels of the other objectives is termed preferential independence [53]. It implies that the indifference curves of any two objectives are the same regardless of the other functional values. In this paper only local information about the overall preference function \( U \) is needed. As a result, the restrictive assumption of preferential independence is averted.

Throughout this thesis it will be assumed that the decision maker can provide tradeoffs between the various objective functions. Since these local changes which preserve indifference can be made around any point in objective space, the objectives are considered compensatory. Accordingly, one generally refers to this as a compensatory tradeoff model. In contrast to this are the noncompensatory models which include lexicographic ranking schemes reviewed by Fishman [17], elimination by aspects discussed in Tversky [54, 55], and several others described in MacCrimmon [38].
1.4. Methods of Solution

Haines et al. [26] point out that basically the methods used in identifying the best-compromise solution fall into two categories: 1) those which identify efficient solutions before trying to determine the best-compromise solution and 2) those which endeavor to directly solve for it. Cohon and Marks [10] have further subdivided this latter category into methods which rely on "prior articulation of preferences" and those which rely on "progressive articulation of preferences".

Techniques which rely on the prior articulation of preferences include Goal Programming [9, 31, 36], Optimal Weighting [39, 40], ELECTRE [6, 48], and the Surrogate Worth Tradeoff Method [24, 26]. The decision maker arrives at a complete ordering or at least a "more complete" ordering (as in the case of ELECTRE) by the use of these techniques. Preferences laid down by the decision maker prior to the start of the problem act as a basis for these orderings.

The techniques which rely on the progressive articulation of preferences follow a general pattern. First, they locate a feasible (and perhaps efficient) solution. Next, they modify the problem according to the decision maker's preferences. By repeating this general procedure, the decision maker either converges towards the best-compromise solution, or he terminates the process prior to reaching this limiting point.

The objective of this approach is to yield a satisfactory decision after a reasonable number of iterations and within a reasonable amount of time [52]. This final decision should be one in which the decision
maker is relatively convinced that no other significantly improved decision exists.

Methods representative of this approach include Step Method [4], Sequential Multiobjective Problem Solving [41], Iterative Weighting Method [35], Progressive Orientation Procedure [5], Zionts-Wallenius Interactive Programming Method [64], and the Geoffrion-Dyer-Feinberg Interactive (GDF) algorithm [21, 22].

More recently, Oppenheimer [43] has attempted to blend these two approaches (viz., prior and progressive articulation of preferences) by using a global preference function in conjunction with the Geoffrion-Dyer-Feinberg algorithm. A prior analysis is conducted to determine what functional form best approximates the decision maker's overall preference function. Local tradeoff information then progressively updates this function as one approaches the best-compromise solution.

A major difficulty with Oppenheimer's "proxy" approach is the arduous, confusing and very involved prior analysis which is required. Once this analysis has been completed, however, the process has been shown to progress extremely well. Essentially, the question is whether or not this involved prior analysis is worth the improvement one gains in terms of the rate of convergence.

To date, very little attention has been paid to discrete multiple objective mathematical programming problems. Only recently have there been articles treating this important area. Zionts [63] has suggested an interactive approach based on his and Wallenius' interactive programming method, Lee [37] and Ignizio [31] have extended the goal programming simplex method to integer and zero-one goal programming
problems, and Bitran [7] has developed some theoretical results for
the zero-one linear multiple objective case.

The author's interest lies primarily with the Geoffrion-Dyer-
Feinberg algorithm. A formal presentation of this algorithm will be
given in Chapter II. In anticipation of this, the specific class
of problems to which the GDF algorithm applies and, in particular,
to which this dissertation addresses itself will now be discussed.

1.5. The Problem to be Studied

The multiple objective optimization problem defined in Section 1.1
requires additional structure if it is to be resolved in the general
framework of a man-machine interactive algorithm. To this end, the
specific problem of interest is as follows:

\[
\text{maximize } U[f_1(x), f_2(x), \ldots, f_r(x)] \tag{1.3}
\]

subject to

\[
x \in X = \{x | g_j(x) \leq 0, \ j=1,2,\ldots,m\} \tag{1.4}
\]

where \(x\) is an \(n\)-dimensional vector of decision variables,
\(f_i(x), \ i=1,2,\ldots,r\), are \(r\) distinct objective functions
of the decision vector \(x\), \(X\) forms the constrained set of
feasible decisions, and \(U\) is the decision maker's overall
preference function defined on the range of \(f_i\).

The set \(X\) and each \(f_i\) have been previously defined by the decision
maker. The overall preference function \(U\), however, is not identified
explicitly; rather, it is established locally at various iteration
points by questioning the decision maker.
It is assumed that the functions $U$, $f_i$, and $g_j$ are continuously differentiable. The functions $U$ and $f_i$ are concave while the functions $g_j$ are convex. Furthermore, $U$ is increasing in each $f_i$; and $X$ is a nonempty, compact, convex set.

Since $U$ is increasing in $f_i$, the optimal solution to (1.3)-(1.4) is an efficient point. Because the optimal solution also maximizes the decision maker's overall preference function, it is the best-compromise solution. Conversely, the best-compromise solution is obviously optimal. Hence, problem (1.3)-(1.4) constructively characterizes the best-compromise solution.

It is interesting to note that even under this characterization, it is impossible to disregard any efficient point. The following lemma by Soland shows that every efficient point is still a candidate for the best-compromise solution.

**Lemma 1.1 [52]**

For every efficient point $x^e$ there exists an overall preference function $U$ defined on $R^r$ which is nondecreasing, concave, and possesses continuous derivatives of arbitrary order $s$, such that $U(f(x))$ is maximized over $X$ at $x^e$ and $U(f)$ is maximized over $Y$ uniquely at $f^e = f(x^e)$.

1.6. Additional Assumptions

The objective functions, $f_i$, should also satisfy two additional assumptions. The first of these states that all objectives are relevant to the decision maker's preferences. This simply means that each objective function is germane to the decision maker's judicial process of determining preferences among the points in the objective space. Stated mathematically,
\[ \frac{\partial u}{\partial f_i} \neq 0 \quad \text{for } i=1,2,\ldots,r. \]

In cases where irrelevant objectives exist, they should be omitted since they in no way affect the decision maker's preferences.

The second assumption requires the objectives to be functionally independent. This refers to the fact that there must not exist a functional relationship among any of the objectives used. Essentially, this precludes redundancy from occurring. More formally, for any set of objective functions, there is no non-zero function \( H \) of these objectives for which \( H(x) = 0 \) for all \( x \) in \( X \).

To illustrate why functional independence is necessary, consider the following example. Suppose there are three objective functions \( f_1, f_2, \) and \( f_3 \). In addition, assume that \( f_1 = 5f_2 \). Then, the first two of these objectives are related by the function

\[ H(f_1, f_2) = f_1 - 5f_2 = 0. \]

Thus, they are functionally dependent.

Now suppose one would want to investigate the tradeoffs between \( f_1 \) and \( f_3 \), for example. Since \( f_2 \) is necessarily fixed at some level, \( f_1 \) must also be fixed. Thus, it becomes meaningless to establish tradeoffs between \( f_1 \) and \( f_3 \).

On the other hand, functional independence among \( f_1, f_2, \) and \( f_3 \) would make it reasonable to consider such a tradeoff. In fact, it would permit us to consider tradeoffs between any pair of these objectives.
1.7. Superiority Test

In attempting to solve the multiple objective optimization problem presented in Section 1.5, it would be advantageous to be able to repeatedly restrict the decision space to those points which represent favorable alternatives to the decision maker. In this way, the decision maker would avoid considering courses of action which are inferior to his present state. With this aim, Lemma 1.2 is presented. Although it is not as discriminatory as one might hope, it does reveal a very effective test. This lemma and its contrapositive are repeatedly referred to in later chapters.

**Lemma 1.2**

Suppose $U$ is concave and differentiable on $\mathbb{R}^n$. Then for any scalar $\alpha > 0$,

$$\alpha \nabla_x U(x^2)^t \cdot (x^1 - x^2) < 0 \implies U(x^1) < U(x^2).$$

**Proof.**

Since $\alpha \nabla_x U(x^2)^t \cdot (x^1 - x^2) < 0$ and $\alpha > 0$,

$$\nabla_x U(x^2)^t \cdot (x^1 - x^2) < 0. \tag{1.5}$$

$U$ being concave and differentiable,

$$U(x^1) \leq U(x^2) + \nabla_x U(x^2)^t \cdot (x^1 - x^2)$$

$$= U(x^1) - U(x^2) \leq \nabla_x U(x^2)^t \cdot (x^1 - x^2).$$

Using (1.5), it follows that

$$U(x^1) < U(x^2).$$

Q.E.D.
By successfully passing this test (i.e., \( \alpha \nabla_x U(x^2)^t \cdot (x^1 - x^2) < 0 \)), one has necessarily found a point \( x^2 \) which is preferred to \( x^1 \). Moreover, the function \( U \) does not have to be known explicitly in order to make this test; any positive multiple of its gradient is sufficient.

This lemma is also instrumental in defining a region which contains all points of promising value in terms of \( U \). By taking the contrapositive of the above lemma, namely

\[
U(x) \geq U(x^*) = \alpha \nabla_x U(x^*)^t \cdot (x - x^*) \geq 0,
\]

we see that to find any point \( x \) which has a value equal to or greater than \( U(x^*) \), the search need only concentrate in that half-space for which

\[
\alpha \nabla_x U(x^*)^t \cdot (x - x^*) \geq 0.
\]

Again to emphasize, the function \( U \) need not be known explicitly. The gradient of \( U \) at \( x^* \) or a positive multiple of it is sufficient to determine this inequality.

1.3. A Usable Direction

From a given feasible solution in decision space, a direction will be determined which will align itself with preferred solutions. Exactly how this is accomplished is the subject of Chapters II and III. However, before this discussion can take place, the concept of a direction must be made more precise.

**Definition.** (Direction)

Any non-zero, \( n \)-dimensional vector \( d \) is a direction.
Definition. (Feasible Direction)

Let \( x \in X \subseteq \mathbb{R}^n \), where \( X \) is a convex set. A direction \( d \) at \( x \) is feasible if and only if \( x + \lambda d \in X \) for sufficiently small positive \( \lambda \).

Definition. (Usable Direction)

A feasible direction which improves the overall objective function is said to be usable. In particular, \( d \) is a usable direction if it is a feasible direction which also satisfies

\[
\nabla U(x)^t \cdot d > 0,
\]

where \( U \) is a differentiable concave function.

By repeatedly moving in a usable direction from one feasible point to another, it is hoped that one's location in decision space will continually improve and that the process will eventually terminate. Earlier, a test (Lemma 1.2) was presented which showed how one can determine if he has improved his position in decision space. Chapter III will explain in what way this relates to the multiple objective optimization problem. As for process termination, this is the subject of the next section.

1.9. Algorithmic Convergence

The discussion here concerning the concept of an algorithm and the meaning of its convergence is attributed to Zangwill [62]. He points out in his book, entitled *Nonlinear Programming: A Unified Approach*, that determining an optimal point, except in very special cases, is not a finite process. Unless one fortuitously finds an optimal point, the
iterative process of going from one feasible solution to the next will never end. For this reason it must be made clear what is meant by the expression "a convergent algorithm".

To begin, we define what is known as a point-to-set map.

**Definition. (Point-to-Set Map)**

A point-to-set map $A_k : V_k \rightarrow V_{k+1}$ is a map which associates with any point $z \in V_k$, a set $A_k(z)$ in $V_{k+1}$.

From this, it is possible to define the notion of an algorithm.

**Definition. (Algorithm)**

An algorithm is an iterative process consisting of a sequence of point-to-set maps

$$A_k : V_k \rightarrow V_{k+1}.$$ 

Given an initial point $z^1 \in V_1$, assume that the points $z^2, \ldots, z^k$ have been generated. The algorithm then proceeds as follows:

If $A_k(z^k) = \emptyset$, the process stops.

Otherwise, select the next point in the sequence by use of the recursion

$$z^{k+1} \in A_k(z^k)$$

where $z^{k+1}$ is any point in the set $A_k(z^k)$.

Under this definition any arbitrary sequence of points $y^1, y^2, \ldots, y^i, \ldots$ can be construed as an algorithm. Simply define $z^1 = y^1$, $V_1 = \{y^1\}$, and $A_i(z^i) = \{y^{i+1}\}$ for all $i$. Clearly, this algorithm generates the desired sequence.
This general interpretation of an algorithm is useful in proving convergence. As Zangwill indicates, given a cumbersome or complicated algorithm which generates a sequence of points, it is often easier to redefine the sequence as a second algorithm. In establishing convergence for this second algorithm, one has necessarily implied convergence of the original algorithm.

Because algorithmic convergence generally occurs only in a limiting sense, it is prudent to be precise in discussing the limit of a sequence.

**Definition. (Limit)**

A sequence \( z^1, z^2, z^3, \ldots \), has a limit \( \ell \) if for any \( \epsilon > 0 \), there exists \( N \) such that \( \forall k \geq N \)

\[
\| z^k - \ell \| < \epsilon,
\]

where

\[
\| z^k - \ell \| = \left( \sum_{i=1}^{n} (z^k_i - \ell_i)^2 \right)^{1/2}.
\]

**Definition. (Convergent)**

A sequence is convergent if it has a limit.

To say that a sequence has a limit \( \ell \) is to imply that all but a finite number of terms of that sequence are within \( \epsilon \) of \( \ell \). A much weaker requirement is to state that infinitely many terms of the sequence are within \( \epsilon \) of \( \ell \).

**Definition. (Limit Point)**

A sequence \( z^1, z^2, z^3, \ldots \), has a limit point (cluster point, accumulation point) \( \ell \) if for any \( \epsilon > 0 \) and any \( N \), there exists \( k \geq N \) such that
\[ \| z^k - \xi \| < \epsilon. \]

Although a limit is a limit point, the converse is not necessarily true. For example, define \( z^k = (-1)^k \). The sequence \( z^1, z^2, z^3, \ldots \), has limit points +1 and -1, but it has no limit.

Lemma 1.3 further explains the relationship between limit points and limits.

**Lemma 1.3 [49]**

A sequence \( z^1, z^2, z^3, \ldots \), has a limit point \( \xi \) if and only if there exists a subsequence of this sequence which converges to \( \xi \).

The definition of a convergent algorithm can now be stated. Zangwill asserts that the following definition is reasonable, generally applicable, and very useful.

**Definition. (Convergent Algorithm) [62]**

Given a problem and a solution set \( \Omega \subseteq \mathbb{R}^n \), a convergent algorithm is an algorithm with the following properties:

(a) If the algorithm stops at a point \( z \), then it indicates either that no solution exists or that \( z \) is a solution. Also, if a point \( z \) is a solution, then either \( A_k(z) = \emptyset \) or \( y \in A_k(z) \) implies \( y \) is a solution.

(b) Assume the algorithm generates an infinite sequence of points none of which is a solution. Then if all points are not on a compact set, no solution point exists, while if all
points are on a compact set, the limit of any convergent subsequence is a solution.

In essence, an algorithm converges if it either calculates the optimal point or is able to determine it in a limiting sense. On the other hand, if an algorithmic sequence converges to a non-optimal point, the algorithm is said not to converge. This definition of algorithmic convergence will be used in Chapter III to show that a particular algorithmic process will eventually converge.

1.10. Organization of the Report

As previously mentioned, Chapter II details the Geoffrion-Dyer-Feinberg Interactive algorithm. Man-machine interactive decisions are its salient feature. The algorithm is described in the context of the Frank-Wolfe mathematical programming algorithm. Some of the limitations of this algorithm are presented in order to emphasize the reasons why one should consider the alternative approach discussed in the next chapter.

The Tradeoff Cutting Plane algorithm is introduced in Chapter III. The intent of the algorithm and the concepts behind it are also explained. A detailed presentation of the algorithm and a proof of its convergence are included. The chapter concludes with a discussion about a sensitivity analysis which is automatically performed upon termination of the algorithm.

In order to illustrate an application of the Tradeoff Cutting Plane algorithm to a specific example, a storm drainage problem, similar to that faced by many medium size communities in the United States, is presented in Chapter IV. Convergence results are shown for various policies associated with this application.
The discrete multiple objective optimization problem is considered in Chapter V. Although conceptually the optimization approach is the same, this chapter shows how the algorithm is modified to address this situation. The nature of the problem and its peculiar qualities are also discussed.

In conclusion, Chapter VI outlines possible areas for further research. It discusses areas relevant to the algorithm itself and multiple objective optimization in general.

In Appendix A an assortment of definitions and lemmas relevant to this report are listed. Appendix B follows with a presentation of a programming technique concerned with locating the optimal solution to a nonlinear programming problem. The technique is a specific type of those generally referred to as methods of centers, a clever cross between penalty function methods and methods of feasible directions. Finally, Appendix C displays a complete listing of the program.
CHAPTER II

THE GEOFFRION-DYER-FEINBERG INTERACTIVE ALGORITHM

In Section 1.4 it was stated that there are basically two categories of multiple objective techniques. In the category labeled "progressive articulation of preferences", reference was made to a man-machine interactive process termed the Geoffrion-Dyer-Feinberg (GDF) algorithm. In this algorithm use is made of tradeoffs between the objectives to determine the best-compromise solution.

2.1. General Overview

The method is essentially a search procedure which questions the decision maker at each step in order to advance to "improved" feasible solutions. In the performance of this search, explicit knowledge of the overall preference function is not essential. Instead, by questioning the decision maker, local information regarding his preferences is received; and this, in turn, is sufficient to determine a direction in which to proceed to preferred solutions. The step size is found by once again interacting with the decision maker to find the point on this ray which is most preferred. Termination occurs when the optimal is reached or when the improvement between steps is less than some specified value.
2.2. Description

This section will briefly describe the algorithm as it is presented by Geoffrion, Dyer, and Feinberg [22]. The technique was originally proposed by Geoffrion [21] in a more general framework. He, together with Dyer and Feinberg, later formalized the basic technique and applied it to the operation of an academic department.

The description of the algorithm is presented with specific reference to the Frank-Wolfe algorithm [19]. The reasons behind the selection of this particular algorithm include its simplicity, convenience, and appropriate theoretical properties [22].

Recall from Section 1.5 that the multiple objective optimization problem of interest is of the following form:

\[
\begin{align*}
\text{maximize } & U(f_1(x), \ldots, f_r(x)) \\
\text{subject to } & x \in X,
\end{align*}
\]  

(2.1)

where \( f_1, f_2, \ldots, f_r \) are \( r \) functionally independent objective functions defined on the convex, compact, feasible decision space \( X \) and \( U \) is the decision maker's overall preference function defined on the objective space \( Y \). Each \( f_i \) is concave and continuously differentiable on \( X \), and the preference function \( U \) is concave increasing and continuously differentiable on \( Y \). The function \( U \) is not assumed to be known explicitly, whereas each \( f_i \) and the set \( X \) are assumed given.

In order to solve (2.1) via the Frank-Wolfe algorithm, one proceeds as follows:

Step 0. Choose an initial feasible solution \( x^0 \in X \).

Let \( k=1 \).
Step 1. Determine an optimal solution $\bar{z}^k$ of the direction finding problem

$$\begin{align*}
\text{maximize} & \quad \nabla_x U[f_1(x^k), f_2(x^k), \ldots, f_r(x^k)] \cdot z \\
\text{subject to} & \quad z \in X.
\end{align*}$$

Let $d^k = \bar{z}^k - x^k$.

Step 2. Determine an optimal solution $t^k$ of the step-size problem

$$\begin{align*}
\text{maximize} & \quad U[f_1(x^k + t d^k), f_2(x^k + t d^k), \ldots, f_r(x^k + t d^k)] \\
\text{subject to} & \quad 0 \leq t \leq 1.
\end{align*}$$

If the solution is optimal, terminate. Otherwise, let

$$\begin{align*}
x^{k+1} &= x^k + t d^k, \\
k &= k+1,
\end{align*}$$

and return to Step 1.

In view of the fact that $U$ is not known, this straightforward application of the Frank-Wolfe algorithm is presumptuous. Fortunately, adjustments can be made which circumvent the problem areas.

The first difficulty arises in Step 1 where the gradient of $U$ is required. By applying the chain rule, one finds

$$\nabla_x U[f_1(x^k), \ldots, f_r(x^k)] = \sum_{i=1}^r \frac{\partial U}{\partial x_i} f_i(x^k) \nabla_x f_i(x^k)$$

(2.5)
where \((\partial U/\partial x_i^k)^k\) is the \(i^{th}\) partial derivative of \(U\) evaluated at \([f_1(x^k), \ldots, f_r(x^k)]\), and \(\nabla f_i(x^k)\) is the gradient of \(f_i\) evaluated at \(x^k\). Recalling that the optimal solution \(z^k\) of (2.2) is not affected by multiplying the objective function by a positive scalar, one may divide the objective function by any positive quantity (e.g., \((\partial U/\partial x_i^k)^k\)).

**Definition.** (Reference Criterion) [22]

The objective function on which the decision maker bases his tradeoffs is termed the reference criterion.

Assume that \(f_1\) is the reference criterion. Now by dividing equation (2.5) by \((\partial U/\partial x_i^k)^k\) and appropriately modifying the objective function in equation (2.2), Step 1 becomes equivalent to

\[
\text{maximize } \sum_{i=1}^{r} w_i^k \sum_{i=1}^{r} \frac{\nabla f_i(x^k)^t \cdot z^k}{\partial U/\partial x_i^k}\]

subject to \(z \in X\),

where \(w_i^k = (\partial U/\partial x_i^k)^k / (\partial U/\partial x_1^k)^k\) is the tradeoff ratio between objectives \(f_i\) and \(f_1\) at the current point.

Once these tradeoffs are obtained from the decision maker, problem (2.2) can be solved. There are several methods by which these tradeoffs can be approximated. An exposition of these methods will not be presented here. Instead, one is referred to [15, 22].

Having completed Step 1, Step 2 is solved by graphically representing

\[f_i(x^k + t d^k), \ 0 \leq t \leq 1\]

for all \(i=1, 2, \ldots, r\), or by giving the value of the vector

\([f_1(x^k + t d^k), \ldots, f_r(x^k + t d^k)]\) for various values of \(t\) between 0 and 1.
From this selection the decision maker chooses his preferred decision point which, as a result, solves problem (2.3). Step 2 is then completed, and the decision maker has the option of either terminating the process or returning to Step 1.

2.3. The Interactive Frank-Wolfe Algorithm

To be more concise, the algorithm works interactively with the decision maker in the following way:

Step 0. The decision maker chooses an initial point

\[ x^1 \in X. \]  Let \( k = 1 \).

Step 1. a) The decision maker assesses his tradeoff ratios \( w_i^k \) at the current base point.

b) Compute an optimal solution \( z^k \) to problem (2.6).

Let \( d^k = z^k - x^k \).

Step 2. The decision maker subjectively determines an optimal solution \( t^k \) to problem (2.3). He then either decides to terminate or proceeds on to Step 1 after redefining as in Equation (2.4).

2.4. Significant Attributes

Worthy of note is that the decision maker interacts only with matters regarding the objective space. Since the dimension of the objective space is usually significantly less than the dimension of the decision space, this fact is of practical importance. Furthermore, working only in the objective space, the decision maker determines tradeoffs between the objectives without being ostensibly influenced by the decision variables.

Secondly, as the decision maker progresses from iteration to iteration, he accumulates, either consciously or unconsciously, more
and more information. Consequently, he becomes increasingly more aware of the implications regarding his tradeoffs and the nature of his problem.

Finally, note that the decision maker only considers relative preferences; exact specification of the overall preference function is needless and impractical. The information of significant importance is the tradeoffs made between pairs of objectives. The \( r - 1 \) tradeoffs between objective \( i \) \((i = 2, 3, \ldots, r)\) and objective 1 are the minimum required. Consideration of all possible pairwise tradeoffs is unnecessary, unless one is interested in a check on consistency.

2.5. Some Limitations of the GDF Algorithm

For each cycle of the GDF algorithm, there are two interactions with the decision maker. Different types of information are required at each of these interactions.

The first interaction asks for estimates of the decision maker's marginal rates of substitution between the objectives. The difficulty in determining these estimates is of obvious concern. Dyer [15] has investigated this area and has developed a tradeoff routine designed to arrive at these estimates through ordinal comparisons. Even with this improvement, there is still difficulty in determining these values and a lack of confidence in the solution based on these estimates [57].

The second interaction requires the decision maker to resolve a step-size problem. Geoffrion [21] originally proposed that this be done by plotting the values of all \( r \) objectives \( f_i(x^k + td^k), i = 1, 2, \ldots, r \), as a function of \( t \). As \( t \) varies between 0 and 1, the decision maker
examines the superimposed plots and selects the most preferred value
of t. Geoffrion acknowledges that as the number of objective functions
increases so also does the task of determining this optimal step-size.

Dyer [15] modified this procedure by displaying the objective
vector for equally spaced values of t between 0 and 1. Once the decision
maker selects his preferred vector, he has determined his step-size,
and the algorithm is ready for another iteration.

Although these methods don't insure locating the maximum as
requested in Step 2 of the GDF algorithm, Hogan [28] argues that
the maximum is not essential for convergence. As long as a "big
enough" improvement is made on each iteration, suboptimization of
the step-size is permissible.

Next, some mention should be made of the fact that the Frank-
Wolfe algorithm is the optimization procedure used to implement
the Geoffrion-Dyer-Feinberg interactive program. Although it is not
the only technique able to be incorporated, it was chosen for its
computational simplicity and impressive convergence properties [28,
61]. Yet, the Frank-Wolfe procedure is limited by the fact that it
addresses a nonlinear objective function subject to linear constraints.
As stated in Oppenheimer [43], nonlinear constraints require the use
of more complex nonlinear methods. The extent to which this might
adversely affect the solution process is not clear.

Montgomery and Bettencourt [42], in applying the GDF algorithm
to a study involving the effect of crew training on the effectiveness
of a main battle tank, had to augment the GDF algorithm to accommodate
constraints of the form

\[ \sum_{j=1}^{n} x_j^2 = b. \]
They used a code [2] based on a cyclic coordinate algorithm to deal with this specific type of quadratic constraint.

Another important limitation of the Geoffrion-Dyer-Feinberg algorithm concerns the likelihood of transferring misleading information to the decision maker. Notice that in Step 1 of the algorithm, the objective function is maximized over the entire decision space \( X \). Under the assumptions laid forth in Section 1.5, regions of this decision space can be removed from further consideration based on the decision maker's tradeoffs. (This will be further explained in Chapter III.) Consequently, the solution \( z^k \) in Step 1 might possibly be a point outside the region of interest. This, in turn, causes the decision maker (in Step 2) to needlessly consider points located in an unfavorable region.

Although theoretically the Geoffrion-Dyer-Feinberg algorithm is able to be used in conjunction with standard branch and bound procedures to affect a discrete solution, the amount of information required of the decision maker makes it an impractical approach [16]. Unless an improved branching scheme is developed, the extension of this algorithm to a discrete mathematical programming context is injudicious.

A number of the limitations explained here motivated the development of the Tradeoff Cutting Plane algorithm, which is covered in the next chapter. In describing this algorithm, reference is made to a methods of centers programming technique presented in Appendix B. A modification of this method is used as a subprocedure in the Tradeoff Cutting Plane algorithm. Consequently, occasional reference is made to Appendix B for further clarification. Coverage of the discrete optimization problem will be deferred until Chapter V.
CHAPTER III

THE TRADEOFF CUTTING PLANE ALGORITHM

In this chapter the Tradeoff Cutting Plane (TCP) algorithm is introduced and shown to converge. Although the algorithm is termed a cutting plane algorithm, it does not conform to the established meaning of this type of an algorithm. With cutting plane algorithms, each added constraint "cuts" off a portion of the previous region leaving the feasible region intact but eliminating the present "optimal" solution. The TCP algorithm, on the other hand, will cut across the feasible region and will retain both the present solution and the best-compromise solution. It can be viewed as an internal cutting plane approach as opposed to the former external approach. The author is aware of the possible terminological inconsistency, but believes that this purposeful abuse in terms will serve to simplify the presentation, not cause any unwarranted confusion, and permit a more natural descriptive label of the technique.

First, the concepts behind the algorithm together with its intent are discussed.

3.1 Intent of the Tradeoff Cutting Plane Algorithm

As formerly mentioned, the TCP algorithm was motivated by a number of the problems associated with the GDF algorithm. Primary
concern centered on the problems of tradeoff sensitivity, step-size resolution, and discrete optimization. By addressing these problems, it was anticipated that the needs and judgmental capacities of the decision maker would be better served. This should not be construed as a condemnation of the GDF algorithm for the TCP algorithm is closely related to this approach. Besides, the GDF algorithm has been shown to be a useful and innovative tool [22, 43]. Instead, what is presented here should be viewed as a viable alternative to GDF method.

The information requirements of the two algorithms are different. Whereas the GDF algorithm requires interaction with the decision maker on each iteration to resolve the step-size problem and to acquire tradeoffs, the TCP algorithm only requires the tradeoff interaction. In other words, the TCP algorithm seeks to make more efficient use of the available information. By eliminating the need for a step-size interactive decision, one type of value judgment is completely removed. This is significant in view of the fact that the aim of man-machine interactive algorithms is to minimize the information burden of the decision maker while still allowing for an effective solution process.

As noted, the task of supplying tradeoffs between objectives is continued in the TCP algorithm. Its purpose here, however, is to establish a cutting plane in objective space in order to remove from further consideration large areas of the decision space. Although the decision maker will not be aware of the extent to which he has reduced his decision space, he is assured that the best-compromise solution has not been carelessly eliminated. This reduced space is then decreased again by another cut. By continuing
this process, the decision maker tends towards the area of the original decision space which contains best-compromise solution.

The tradeoffs supplied by the decision maker function to a large extent as a means of further restricting the decision space. The cutting plane established from these tradeoffs focuses in on the most relevant portion of the decision space, but the accomplishment of this is based more on eliminating the inferior portion of the space than on identifying the superior portion. Simply put, the cut acts as a restriction on where not to proceed. Upon termination of the algorithm, the decision maker has identified the neighborhood of the best-compromise solution, if not the solution itself.

This cutting plane approach also makes possible a post-optimal analysis of the decision maker's tradeoffs. Assessing value tradeoffs between objectives has always been a troublesome and difficult task. Although this problem of tradeoff assessment still remains, it is somewhat lightened by the fact that the tradeoffs are able to be tested against "critical" limits. It gives the decision maker an indication of the amount of latitude he had in making his tradeoffs.

An interesting by-product of the TCP algorithm is that the same solution procedure is used regardless of whether the original constraints are linear or not. The algorithm decomposes the evaluation process in a way similar to that of the GDE algorithm in that the overall optimization problem is divided into simpler subproblems. The original problem is actually restructured into a sequence of linear programs and one-dimensional searches. Each linear program requires the usual decision maker tradeoff interaction, whereas the
one-dimensional (step-size) searches require no decision maker interaction. As a result, the information burden on the part of the decision maker is reduced and a greater number of objectives are more easily accommodated.

3.2. Concepts Behind a Cut

At each iteration of the TCP algorithm, the decision maker evaluates one alternative which is represented by some point \( f' = f(x') \) in objective space. If one considers the \( r+1 \) dimensional space of the concave, overall preference function \( U \) and the objective space, the equation

\[
U(f) = U(f') + \sum_{i=1}^{r} w_i (f_i - f'_i)
\]  

where \( w_i = \frac{\partial U}{\partial f_i} \bigg|_{f=f'} \), is a linear approximation of \( U \) at \( U(f') \). It describes the \( r \) dimensional hyperplane which is tangent to the indifference surface at \( f' \). Focusing on just the \( r \) dimensional objective space, the \( r-1 \) dimensional hyperplane satisfying

\[
\sum_{i=1}^{r} w_i (f_i - f'_i) = 0
\]  

provides a boundary between those values of \( U \) which are promising and those which are unfavorable. A promising region refers to a region containing at least those points whose values in terms of \( U \) is equal to or greater than \( U(f') \). Thus, in searching for improved values of \( U \), one need only concentrate in that half-space for which

\[
\sum_{i=1}^{r} w_i (f_i - f'_i) \geq 0.
\]
This is what is discussed in Section 1.7 with regards to the contrapositive of Lemma 1.2. The difference here being that the half-space is in objective space and not, as before, in decision space.

With the $\partial U/\partial f_i$, for $i=1,2,\ldots,r$, being unknown, the weights $w_i$ and, therefore, inequality (3.3) cannot be directly determined. However, if one were to divide this inequality by the positive scalar $(\partial U/\partial f_1) |_{f = f'}$, it would read like this:

$$
\sum_{i=1}^{r} \left[ (\partial U/\partial f_i)/(\partial U/\partial f_1) \right] (f_i - f'_i) \geq 0.
$$

(3.4)

Since $(\partial U/\partial f_i)/(\partial U/\partial f_1) |_{f = f'}$, for $i=2,3,\ldots,r$, are the decision maker's tradeoffs at $f'$, inequality (3.4) is known. Furthermore, because those $f$ which satisfy (3.3) also satisfy (3.4) and vice versa, the promising region of the objective space is now known.

It should be emphasized that need for exact knowledge of $U$'s functional form has been circumvented. The tradeoffs furnished by the decision maker are sufficient information to further restrict the space containing the best-compromise solution.

In Section 1.2 it was stated that the slope of an indifference curve is negative. That is, while holding all other objectives constant, an indifference change in the values of objective 1 and $i$ by the respective amounts $\Delta_1$ and $\Delta_i$ forms the positive tradeoff ratio

$$
- \frac{\Delta_1}{\Delta_i} = \frac{\partial U/\partial f_1}{\partial U/\partial f_i}.
$$

(3.5)
The sum in (3.4) is a linear combination of concave functions. Since the weights \( ((\partial U/\partial x_i)/(\partial U/\partial x_1), i=2,3,\ldots,r) \) of this linear combination are positive, the left-hand-side of the expression in (3.4) is itself a concave function (Appendix A, Lemma A.4). In requiring this expression to be greater than or equal to zero, the corresponding set of feasible decisions which satisfy this inequality is convex (Appendix A, Lemma A.2).

Succinctly, the process works as follows. The decision maker furnishes tradeoffs at each iteration. From these tradeoffs a cut is formed in objective space. This restriction, which limits the promising region, defines a convex set in decision space. The intersection of this set with the initial feasible region in the decision space defines a convex set which must contain the best-compromise solution.

To illustrate, let's assume that there are two objectives, \( f_1 \) and \( f_2 \), defined in two decision variables, \( x^1 \) and \( x^2 \). Figure 3.1 shows an original convex set, \( X \), before any tradeoffs are made. Figure 3.2 shows the corresponding set, \( Y \), in objective space. The current point is \( x' \) in \( X \). This means that the decision maker furnishes tradeoffs at the point \( f(x') \) in \( Y \). From these tradeoffs the half-space of promising points is established in objective space. This half-space cut translates to the convex set diagrammed in Figure 3.3. The shaded region in Figure 3.2 illustrates the promising region; the one in Figure 3.3 shows the restricted region where the best-compromise solution is located.
Figure 3.1 Decision Space

Figure 3.2 Objective Space
Figure 3.3 Translated Cut in Decision Space
3.3. Locating the Next Tradeoff Point

The question which now arises is how does one select a new point in the promising region? The problem relates to that of trying to locate a point in the restricted region of the decision space.

The point $x'$ in $X$ as well as the equation of the constraint through this point (introduced by the translated cut) are known. The constraints associated with the original feasible region are also known. A point near the "center" of this constrained set is desirable for it would eliminate an ample portion of this set on the next cut. A method of centers technique is applicable (Appendix B).

Before considering how this is accomplished, we should be assured that if one locates a point in this restricted set, then its image under the mapping $f$ is in the promising region. Lemma A.7 in Appendix A is directed at this specific situation. It, in effect, states that no matter where one locates in the restricted set of the decision space, the corresponding point in objective space must be in the promising region.

Another pertinent question is whether or not the restricted set of the decision space contains the best-compromise solution. Lemma A.8 answers this question, for it states that if there is a best-compromise in objective space, then a solution exists, and it must be located within the restricted set of the decision space.

Now let's proceed in trying to locate this best-compromise solution. The image of this solution is obviously on the boundary of the feasible region in objective space. However, a decision
point whose image is on the boundary is not necessarily found on the border of the feasible set in decision space. Thus, to concentrate on the extreme points or even to focus on the boundaries of the decision space would be a mistake. One must consider the entire restricted region when searching for the best-compromise solution. Consequently, a method similar to the methods of centers technique discussed in Appendix B will be used.

Given that we are located at some point in the decision space and the constraint through this point has been determined, we move along a ray emanating from here until the location is reached at which the maximum distance from any constraint is minimized. The image of this new point is the next tradeoff point.

**Definition.** (Tradeoff Point)

A tradeoff point is any point in objective space at which the decision maker assesses his tradeoffs.

The direction of this ray is very important. It must orient itself in a feasible direction which will increase the unknown, overall preference function \( U \). To select this usable direction \( \mathbf{h} \) at a point \( \mathbf{x}^0 \), consider the following linear programming problem:

\[
\text{minimize} \quad M \\
\text{subject to} \quad -M - \alpha \sum \frac{\nabla}{\nabla \mathbf{x}} U(\mathbf{x}^0)^t \cdot \mathbf{h} \leq 0 \\
-M + g_1(\mathbf{x}^0) + \sum g_i(\mathbf{x}^0)^t \cdot \mathbf{h} \leq 0 \\
\mathbf{h} = \mathbf{h}_1 \mathbf{h}_2, \ldots, \mathbf{h}_m, \\
-1 \leq \mathbf{h}_j \leq 1, \quad j=1,2,\ldots,n
\]

where

\[
\alpha = \frac{1}{\partial f_1 / \partial \mathbf{x}_i} \bigg|_{\mathbf{x}^0} = f(\mathbf{x}^0).
\]
Note the similarity with the LP problem (B.6) in Appendix B. The constraints (3.8) insure feasibility, whereas (3.7) guarantees that the vector \( h \) points in a direction which will increase \( U \). Since \( U \) is unknown, constraint (3.7) is not exactly like the one found in (B.6). Instead, it reflects the known, local information the decision maker furnishes about \( U \). Consequently, interaction with the decision maker is required to establish the LP problem (3.6)-(3.9).

For example, let there be three objective functions associated with the multiple objective problem:

\[
\begin{align*}
  f_1 &= 2x_1 + x_2, \\
  f_2 &= -x_1^2 - x_2^2 + 10, \\
  f_3 &= 4x_1 + 6x_2 - 2x_1^2 - 2x_1x_2 - 2x_2^2.
\end{align*}
\]

Assume that \( f_1 \) is the reference criterion. Now suppose the decision maker gives \((1, 1/2, 5)\) as the tradeoff vector at the current point \( f(x^0) = f(1,2) = (4,5,2) \). With this information, constraint (3.7) becomes

\[
\begin{align*}
-M - \alpha \left( \begin{array}{c}
  x_1 \\
  x_2 \\
  x_3
\end{array} \right) & \geq \left( \begin{array}{c}
  h_1 \\
  h_2
\end{array} \right), \\
-M - (1, 1/2, 5) \cdot \begin{bmatrix} 2 & 1 \\ -2x_1 & -2x_2 \\ 4 & -4x_1 - 2x_2 & 6 - 2x_1 - 4x_2 \end{bmatrix} & \cdot \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \\
-M - (1, 1/2, 5) \cdot \begin{bmatrix} 2 & 1 \\ -2 & -4 \\ -4 & -4 \end{bmatrix} & \cdot \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}
\end{align*}
\]
\[ -M - \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \cdot \begin{bmatrix} -19 \\ -21 \end{bmatrix} \] 
\[ = -M + 19h_1 + 21h_2 \leq 0. \] 
(3.14)

The other constraints are derived directly from the previously known information. The LP problem (3.6)-(3.9) can now be solved to determine the usable direction \( \mathbf{h} \).

It should be noted that for any \( \mathbf{x} \) in

\[ C_k = \{ \mathbf{x} | g_i(\mathbf{x}) \leq 0, \ i = 1, 2, \ldots, m_k \} \]

\((M, \mathbf{h}) = (0, 0)\) is a feasible solution. Consequently, \( M \leq 0 \) for all \( \mathbf{x} \) in \( C_k \). Moreover, since \( U \) and \( g_i \) are continuously differentiable, \( M \) is the maximum of a set of continuous functions and, as such, is itself a continuous function. Use of these observations will be made later in proving convergence of the algorithm.

The constraints in (3.8) are associated with the original \( m \) constraints plus the tradeoff cuts which have been derived on each iteration. Assuming the current tradeoff point is \( \mathbf{f}(\mathbf{x}^0) \), the left-hand-side of the last constraint in (3.8) (i.e., the one associated with the current tradeoff cut) is

\[ -M + g_{m_k}(\mathbf{x}^0) + \mathbf{V}_X g_{m_k}(\mathbf{x}^0)^t \cdot \mathbf{h} \]

\[ = -M - \sum_{i=1}^{r} t_i \left( f_i(\mathbf{x}^0) - f_i(\mathbf{x}^0) \right) \]

\[ - \mathbf{V}_{X=\mathbf{x}^0} \left[ \sum_{i=1}^{r} t_i \left( f_i(\mathbf{x}) - f_i(\mathbf{x}^0) \right) \right]^t \cdot \mathbf{h} \]
where \( t_i = \left( \frac{\partial \Omega}{\partial x_i}, \frac{\partial \Omega}{\partial x_{-i}} \right) \bigg|_{x = x}\)

\[
= -M - \left[ \sum_{i=1}^{r} t_i \nabla_x f_i(x) \right]^{t} \cdot h \\
= -M - \left( \frac{1}{\partial \Omega/\partial x_{-i}} \right) \left[ \sum_{i=1}^{r} \left( \frac{\partial \Omega}{\partial x_i} \right) \nabla_x f_i(x) \right]^{t} \cdot h \\
= -M - \alpha \nabla_x U(x) x^{t} \cdot h .
\]

Thus, the last constraint in (3.8) is redundant with (3.7). In practice, (3.7) is actually dropped in favor of this newest constraint. This means that on each iteration of the algorithm the best feasible direction is determined directly from the "feasibility" constraints. The improvement of the decision maker's overall preference function \( U \) is given implicit consideration through one of these constraints.

### 3.4. Tradeoff Cutting Plane Algorithm

Having explained various phases of the Tradeoff Cutting Plane algorithm, the algorithm itself will now be presented. Let \( m_0 = m \), the initial number of constraints associated with problem (1.3)-(1.4) in Section 1.5. Define

\[
c_0 = \{ x \mid g_i(x) \leq 0, \ i = 1, 2, \ldots, m_0 \}.
\]

At each iteration a new constraint is added, reflecting the tradeoff cut which was made in objective space. Therefore, after the \( k \)th iteration, let
\[ C_k = \{ x \mid g_i(x) \leq 0, \ i=1,2,\ldots,m_k \} \]

where \( m_k = m_\infty + k \) and \( g_i(x) \leq 0, \ i=1,2,\ldots,m_k \) represent the initial 
\m constraints plus the k tradeoff cuts. The set \( C_k \) is assumed to have 
an interior and contain at least one point \( x^c \) for which \( g_i(x^c) < 0 \), 
for all \( i=1,2,\ldots,m_k \). This together with the fact that the functions 
g_i(x) are convex and continuously differentiable ensures that the 
constraint qualification (Appendix A) holds at any feasible point \( x \) 
in \( C_k \) [27, 51, 62]. As before,

\[ \alpha = 1/(\partial u/\partial f_1) \]

evaluated at the current decision point. Let \( d \) be defined as the 
negative of the distance function (Appendix B)

\[ d(x) = \max \{ g_i(x), \ i=1,2,\ldots,m_k \} \]

Two constants, \( \rho_1 \) and \( \rho_2 \), are used in the algorithm's step length 
determination schemes. They are set by the user prior to the algorithm's 
execution and are assumed to have values greater than one. The function 
of these constants will be explained after the algorithm has been 

Algorithm.

Step 0. Compute \( x^0 \in C_0 \) and set \( k=0 \).

Step 1. Determine tradeoffs at \( f(x^0) \).

Step 2. Set \( x = x^k \) and \( t=1 \).

Step 3. Solve (3.6)-(3.9) to obtain \( (M(x), x^k) \).
Step 4. If $M(x) \geq 0$, STOP!

Otherwise, continue.

Step 5. Compute $\lambda^*$, the smallest positive scalar such that

$$d(x + \lambda^* h) = \min \{ d(x + \lambda h) | \lambda \geq 0 \}.$$ 

Step 6. Set $z = \lambda^* h$.

Step 7. If $\alpha \nabla_x U(x + t_z)^t \cdot (-t_z) \begin{cases} < 0, \text{go to Step 12} \\ = 0, \text{go to Step 14} \\ > 0, \text{continue.} \end{cases}$

Step 8. Set $t = t / \rho_1$.

Step 9. If $\alpha \nabla_x U(x + t_z)^t \cdot (-t_z) \geq 0$, return to Step 8. Otherwise, go to Step 14.

Step 10. If $x + t_z \in C_k$, continue.

Otherwise, go to Step 13.

Step 11. If $\alpha \nabla_x U(x + t_z)^t \cdot (-t_z) \begin{cases} < 0, \text{continue} \\ = 0, \text{go to Step 14} \\ > 0, \text{go to Step 13.} \end{cases}$

Step 12. Set $t = \rho_2 t$ and return to Step 10.

Step 13. Set $t = t / \rho_2$.

Step 14. Set $x^{k+1} = x + t_z$, $k = k + 1$, and return to Step 2.
Figure 3.4a. Flowchart of the TCP Algorithm
Figure 3.4b. Flowchart of the TCP Algorithm (Continued)
The TCP algorithm is composed of four parts. The first part, Steps 0-3, determines the termination index $M$ and the usable direction $h$ to be used in locating the center of the current feasible region. The next part is the algorithm's termination rule which is found at Step 4. The third part, Step 5, involves locating the center of the current feasible region. Finally, Steps 6-14 encompass the algorithm's step length determination schemes.

This last part is where the constants $\rho_1$ and $\rho_2$ appear. Specifically, they are used in the step length determination schemes denoted as routines $\#1$ and $\#2$. $\rho_1$ is seen in Step 8 of the algorithm. At this point in the process (Routine $\#1$), too large a step-size has been taken. The result is that the point at which $U$ reaches its maximum (along $x + tz$) has been passed. Since no statement can be made concerning the degree of improvement (if there has even been any), a decrease in the step-size is necessary. In Routine $\#1$ the step-size is decreased by a factor of $\rho_1$ until the point at which $U$ reaches its maximum has been passed again (this time in the other direction). The algorithm then sets the image of this new point as the next tradeoff point and continues.

In the present version of the algorithm, $\rho_1$ is set to 2. A higher value for $\rho_1$ would possibly result in reaching a point of guaranteed improvement in fewer iterations and, in turn, require fewer interactions with the decision maker. On the other hand, a smaller value for $\rho_1$ would allow for a "better" cut to be taken. By this it is meant that the cut intersects the path $x + tz$ at a point closer to where $U$ reaches its maximum. In taking cuts close to this maximizing point, one reduces the feasible decision space more judiciously.
The other constant \( p_2 \) appears in Step 12. Entering Routine \#2 of the TCP algorithm, the maximum of \( U \) along the path \( x + tz \) has not been passed yet. Similar to the situation with \( p_1 \), the considerations one must address in determining a value for \( p_2 \) result in conflict. A large \( p_2 \) is beneficial, for the next step would most likely be infeasible and, consequently, not require any interaction on the part of the decision maker. Yet, a smaller value for \( p_2 \) would increase the possibility of obtaining a better cut.

As previously mentioned, Routines \#1 and \#2 of the algorithm are step length determination schemes. In both of these routines the final step length is less than or equal to the step length determined by the optimal gradient method (i.e., step length \( \hat{\theta} \) where

\[
U(x + \hat{\theta} V_x(U(x))) \leq \max_{\theta} U(x + \theta V_x(U(x))).
\]

This "undershooting" of the optimal gradient step is not as detrimental as it might at first appear. In fact, this tactic is often recommended with feasible direction methods to counteract the general tendency of this optimal step-size to overshoot the best step length in terms of the overall rate of convergence [51]. The best "relaxation" of this optimal step-size differs according to the objective function, but the fact remains that missing this optimal step-size on the short side usually yields improved overall rates of convergence. In practice, Simmons [51] has found that a relaxation factor of 0.9 furnishes generally good results. The relaxation factor associated with the TCP algorithm varies on each iteration. However, a shortened
step length is nearly always used. Exceptions to this occur when the optimal step-size corresponds with the center of the feasible region, in which case this step is retained.

In reference to the algorithm's flow chart, the bold box and diamonds indicate where interaction with the decision maker is required. Although these interactions appear at a number of places in the algorithm, one is precluded from interacting at all of these points due to the algorithm's logic. Note also that each interaction requests the same type of information, namely, local functional tradeoffs. These same tradeoffs are then used again in Step 3 to define the next LP problem.

Plainly, it is advantageous to limit the number of constraints necessary to solve for M in LP problem (3.6)-(3.9). The tradeoff cutting plane algorithm was designed with this in mind. As a result not all of the decision maker's tradeoffs are retained. Since each decision maker interaction produces tradeoffs which are in turn translated into constraints, it becomes clear that not necessarily all the information received from the decision maker continues to be pertinent.

For example, constraints derived from adjacent interaction points along the path $x + t\pi$ can produce nested subsets. The perceptive choice among the possible constraints to retain would be the final cut. This, in essence, is what is done in implementing the TCP algorithm.

The procedure works as follows. If the expression in Step 7 is zero, only one constraint (viz., the current constraint) is added. If this expression is not zero, one must enter either Routine #1 or Routine #2.
In Routine #1, the decision maker interactions continue until 
the same expression (viz., $\alpha \nabla U(x+tz)^{-t}(-tz)$) becomes non-positive. 
When this occurs, cuts produced by this last interaction and the previous 
one are retained. Stated differently, cuts are taken at both ends of 
an interval containing a maximizing point for $U$ along the path $x+tz$.

In Routine #2, three different situations can occur. If a point 
is located for which the same test expression is zero, only this cut 
is retained. The second possibility occurs when the expression becomes 
positive before one experiences infeasibility. If this happens, the 
situation is the same as explained above for Routine #1. Finally, 
if infeasibility occurs (recall that the initial step is always 
feasible), the cut obtained on the last decision maker interaction 
is the only one retained.

Admittedly, the above scheme concerning what information to 
retain may not be the most prudent under all circumstances, but it 
does offer a sensible compromise between over specifying and in-
adequately defining the new feasible decision space.

3.5. Proof of Convergence

This section presents a convergency proof of the TCP algorithm. 
As explained in Section 1.9, convergence means that either the best-
compromise point is located after a finite number of iterations or 
else it is determined in a limiting sense.

Prior to presenting the theorem on convergence, a number of 
lemmas are proven in order to facilitate a better understanding of 
the proof. In these lemmas reference is often made to $t_{\text{max}}$. Assuming 
$U$ to be a concave function defined on a compact convex set $C$ and $h$ to 
be a usable direction at a point $x$ in $C$, $t_{\text{max}}$ determines a point
(viz., \( x + t_{\text{max}} \ h \)) at which

\[
U(x + t_{\text{max}} \ h) = \max_t U(x + th).
\]

\( x + th \in C \)

Two facts concerning \( t_{\text{max}} \) should be emphasized:

(i) \( t_{\text{max}} \) will always exist, and

(ii) since \( h \) is a usable direction, \( t_{\text{max}} \) will always

be greater than zero.

Lemmas 3.1 and 3.2 give, in terms of the gradient of \( U \), conditions of \( t \) for which the value of \( t \) is known in relationship to \( t_{\text{max}} \).

**Lemma 3.1**

Let \( U \) be a differentiable concave function defined on

the compact convex set \( C \), and let \( h \) be a usable direction

at \( x \in C \). If

\[
\nabla_x U(x + th)^t \cdot (-th) < 0 \quad \text{where } t \geq 0, \]

then \( t \leq t_{\text{max}} \).

**Proof.**

Assume \( t > t_{\text{max}} \) (>0). By definition,

\( U(x + th) \leq U(x + t_{\text{max}} h) \). As a result,

\[
\nabla_x U(x + th)^t \cdot [(x + t_{\text{max}} h) - (x + th)] \geq 0 \quad \text{(Lemma A.6)}
\]

\[
\Rightarrow \nabla_x U(x + th)^t \cdot [(t_{\text{max}} - t)h] \geq 0
\]

\[
\Rightarrow \nabla_x U(x + th)^t \cdot [-th] \geq 0 \quad \text{since } t > 0 \text{ and}
\]

\[
t_{\text{max}} < t.
\]
But this contradicts the fact that

\[ \nabla_x U(x + th)^t \cdot (-th) < 0. \]

Hence, \( t \leq t_{\text{max}} \).

Q.E.D.

**Lemma 3.2**

Let \( U \) be a differentiable concave function defined on the compact convex set \( C \), and let \( \overline{h} \) be a usable direction at \( x \in C \). If

\[ \nabla_x U(x + th)^t \cdot (-th) > 0 \quad \text{where } t \geq 0, \]

then \( t_{\text{max}} \leq t \).

**Proof.**

Analogous to the proof of Lemma 3.1.

Q.E.D.

Lemmas 3.3 and 3.4 give statements concerning the value of \( U \) at \( x + th \), where \( t \) is contained in the closed interval \( \left[ \frac{t_{\text{max}}}{\rho}, t_{\text{max}} \right] \) and \( \rho \geq 1 \).

**Lemma 3.3**

Let \( U \) be a concave function defined on the compact convex set \( C \), and let \( \overline{h} \) be a usable direction at \( x \in C \). For any constant \( \rho \geq 1 \) and for all \( t \) such that \( \frac{t_{\text{max}}}{\rho} \leq t \leq t_{\text{max}} \),

\[ U(x + th) \geq U(x + \frac{t_{\text{max}}}{\rho} h). \]
Proof.
Since \( \frac{t_{\text{max}}}{\rho} \leq t \leq t_{\text{max}} \), there must exist a \( \lambda \) such that \( 0 \leq \lambda \leq 1 \) and

\[
t = \frac{t_{\text{max}}}{\rho} + \lambda (t_{\text{max}} - \frac{t_{\text{max}}}{\rho}).
\]

Thus,

\[
U(x + th) = U\left[ x + \left( \frac{t_{\text{max}}}{\rho} + \lambda \left( t_{\text{max}} - \frac{t_{\text{max}}}{\rho} \right) \right) h \right]
\]

\[
= U\left[ (1-\lambda) \left( x + \frac{t_{\text{max}}}{\rho} \right) h + \lambda (x + t_{\text{max}}) h \right]
\]

\[
\geq (1-\lambda) U(x + \frac{t_{\text{max}}}{\rho} h) + \lambda U(x + t_{\text{max}} h)
\]

\[
= U(x + \frac{t_{\text{max}}}{\rho} h) + \lambda U(x + t_{\text{max}} h) - U(x + \frac{t_{\text{max}}}{\rho} h).
\]

Finally, since \( \lambda \geq 0 \) and \( U(x + t_{\text{max}} h) - U(x + \frac{t_{\text{max}}}{\rho} h) \geq 0 \),

\[
U(x + th) \geq U(x + \frac{t_{\text{max}}}{\rho} h).
\]

Q.E.D.

**Lemma 3.4**

Let \( U \) be a concave function defined on the compact convex set \( C \), and let \( h \) be a usable direction at \( x \in C \). For any constant \( \rho \geq 1 \) and for all \( t \) such that \( \frac{t_{\text{max}}}{\rho} \leq t \leq t_{\text{max}} \),

\[
U(x + th) - U(x) \geq \frac{1}{\rho} \left( \max_{t} \left( \frac{t_{\text{max}}}{\rho} \right) U(x + th) - U(x) \right).
\]

\[\forall x + th \in C\]
Proof.
Because \( \frac{t_{\max}}{p} \leq t \leq t_{\max} \) and \( U \) is concave,
\[
U(x + th) \geq U(x + \frac{1}{p} t_{\max} h)
\]

(Lemma 3.3)
\[
= U[(1 - \frac{1}{p}) x + (\frac{1}{p})(x + t_{\max} h)]
\]
\[
\geq (1 - \frac{1}{p}) U(x) + \frac{1}{p} U(x + t_{\max} h)
\]
\[
= U(x) + \frac{1}{p} [U(x + t_{\max} h) - U(x)].
\]

Therefore, using the definition of \( t_{\max} \),
\[
U(x + th) - U(x) \geq \frac{1}{p} (\max_t U(x + th) - U(x)).
\]
\[
\forall x + th \in C
\]

Q.E.D.

The next lemma refers to the step-size routine in the TCP algorithm and shows what improvement in \( U \) is guaranteed by this routine.

Lemma 3.5

Let \( U \) be a differentiable concave function defined on the compact convex set \( C \). Let \( x' = x + \theta h \) where \( h \) is a usable direction at \( x \in C \). Assume \( h \) solves the LP problem (3.6)-(3.9), and assume \( \theta \) is determined by Steps 7-13 of the TCP algorithm (i.e., \( x' \in A(x) \)). Then
\[
U(x') - U(x) \geq \frac{1}{p} [\max_{\varphi} U(x + \varphi h) - U(x)]
\]
\[
\forall x + \varphi h \in C
\]

where \( p = \max\{p_1, p_2\} \).
Proof. (Note: $\theta_h = t \lambda_h^* = t_z$)

Determining $\theta' = t^o \lambda^*$ from Steps 7-13 of the TCP algorithm falls into three cases. The separate cases arise at Step 7 where $\alpha \nabla_x U(x + t z) (-t z)_{-z}$ is tested to see if it is positive, negative, or zero.

Case 1: $\alpha \nabla_x U(x + t z)^t (-t z) > 0$

By construction, $x + t z$ is initially in $C_k$. That is, the first step is always feasible. After this, $t$ is repeatedly divided by $\rho_1$ until $\alpha \nabla_x U(x + t_{n+1} z)^t (-t_{n+1} z) < 0$

where $t_m \equiv \frac{t_m}{\rho_1}$ for $m \geq 0$. Thus, by Lemma 3.1, $t_{n+1} \leq t_{\text{max}}$.

By construction, $\alpha \nabla_x U(x + t_n z)^t (-t_n z) > 0$. Therefore, by Lemma 3.2, $t_{\text{max}} \leq t_n$. Consequently,

$$t_{\text{max}} \leq \frac{t}{\rho_1^n}$$

$$= \frac{t_{\text{max}}}{\rho_1} \leq \frac{t}{\rho_{n+1}}$$

(since $\rho_1 > 1$)

$$= \frac{t_{\text{max}}}{\rho_1} \leq t_{n+1}.$$

As a result,

$$\frac{t_{\text{max}}}{\rho_1} \leq t_{n+1} = t^o \leq t_{\text{max}}.$$
Finally, by Lemma 3.4,

\[
U(x + t\lambda^* h) - U(x) = U(x + t^* \lambda^* h) - U(x)
\]

\[
= U(x + t^* z) - U(x)
\]

\[
\geq \frac{1}{\rho_1} \left[ \max_t U(x + tz) - U(x) \right]_{x + tz \in C}
\]

\[
= \frac{1}{\rho_1} \left[ \max_{\varphi} U(x + \varphi h) - U(x) \right]_{x + \varphi h \in C}
\]

\[
\geq \frac{1}{\rho} \left[ \max_{\varphi} U(x + \varphi h) - U(x) \right]_{x + \varphi h \in C}
\]

Case 2: \( \alpha \nabla_x U(x + tz)^t \cdot (-tz) < 0 \)

Again, \( x + tz \) is initially in \( C_k \). After this, \( t \) is continually increased by a factor of \( \rho_2 \) until either \( x + t_{n+1} z \notin C_k \) or \( \alpha \nabla_x U(x + t_{n+1} z)(-t_{n+1} z) \geq 0 \) where \( t_m = \rho_2^m t \) for \( m \geq 0 \).

Subcase 2A: \( x + t_{n+1} z \notin C_k \).

By construction, \( x + t_n z \in C_k \) and \( \alpha \nabla_x U(x + t_n z)^t \cdot (-t_n z) < 0 \).

Thus, by Lemma 3.1, \( t_n \leq t_{\text{max}} \).

Recall from the discussion concerning tradeoff cuts that for all \( y \in C \cap \overline{C_k} \), \( U(y) < U(x) \). Therefore, along the path \( x + tz \), \( U \) must achieve its maximum in \( C_k \). Since \( x + t_{n+1} z \notin C_k \), \( t_{\text{max}} \leq t_{n+1} \).
Now,
\[ t_{n+1} \geq t_{\text{max}} \]

\[ \rho_2^{(n+1)} t \geq t_{\text{max}} \]

\[ \rho_2 (\rho_2^n t) \geq t_{\text{max}} \]

\[ \rho_2^n t \geq \frac{t_{\text{max}}}{\rho_2} \quad \text{since } \rho_2 > 1 \]

\[ t_n \geq \frac{t_{\text{max}}}{\rho_2} . \]

As a result,
\[ \frac{t_{\text{max}}}{\rho_2} \leq t_n = t^o \leq t_{\text{max}} . \]

Finally, by Lemma 3.4 and reasoning analogous to that found in Case 1,
\[ U(x + \theta^o h) - U(x) \geq \frac{1}{\rho} \left[ \max_{\varphi} U(x + \varphi h) - U(x) \right] . \]

\[ x + \varphi h \in C \]

Subcase 2B: \[ x + t_{n+1} z \in C_k \] and \[ \max_{x} U(x + t_{n+1} z)^t (-t_{n+1} z) > 0 . \]

By construction, \[ x + t_n z \in C_k \] and \[ \max_{x} U(x + t_n z)^t (-t_n z) < 0 . \]

Hence, \[ t_n \leq t_{\text{max}} \] by Lemma 3.1.

Since \[ \max_{x} U(x + t_{n+1} z)^t (-t_{n+1} z) > 0 , \] \[ t_{\text{max}} \leq t_{n+1} \] from Lemma 3.2.

Thus, \[ t_n \leq t_{\text{max}} \leq t_{n+1} . \] Therefore, as in Subcase 2A with \[ t_n = t^o , \]
\[ U(x + \theta h) - U(x) \geq \frac{1}{\rho} \left[ \max_{\varphi \in C} U(x + \varphi h) - U(x) \right]. \]

Subcase 2C: \( x + t_{n+1} z \in C_k \) and \( \alpha \nabla_x U(x + t_{n+1} z)^t \cdot (-t_{n+1} z) = 0. \)
Define \( t_{n+1} = t^o \) and refer to Case 3.

Case 3: \( \alpha \nabla_x U(x + t z)^t \cdot (-t z) = 0 \)
Define \( t^o = t. \) Then since \( \alpha > 0, \)
\[ \nabla_x U(x + t^o z)^t \cdot (-t^o z) = 0. \]
Furthermore, since \( t^o > 0, \)
\[ \nabla_x U(x + t^o z)^t \cdot (z) = 0. \]
Therefore, \( \nabla_x U(x + t^o z)^t \cdot (t_{\max} - t^o z) = 0 \)
\[ \Rightarrow \nabla_x U(x + t^o z)^t \cdot [(x + t_{\max} z) - (x + t^o z)] = 0 \]
\[ \Rightarrow U(x + t_{\max} z) \leq U(x + t^o z) \quad \text{by Lemma A.6.} \]

By the definition of \( t_{\max}, \)
\[ U(x + t_{\max} z) \geq U(x + t^o z). \]

Consequently,
\[ U(x + t_{\max} z) = U(x + t^o z). \]
Hence,

\[ U(x') - U(x) = U(x + \theta h) - U(x) \]
\[ = U(x + t^\circ z) - U(x) \]
\[ = \max_{t} \left\{ \frac{1}{\rho} \left[ \max_{\phi} U(x + \phi h) - U(x) \right] : x + \phi h \in C \right\} \]

These three cases resolve all possibilities. It follows, therefore, that the lemma's conclusion is valid.

Q.E.D.

In showing convergence of the TCP algorithm, use is made of a surrogate algorithm developed by Polak [46]. After comparing the TCP algorithm to this surrogate algorithm, the convergence theorem associated with this algorithm is employed to prove the convergence of the TCP algorithm. This approach makes possible the establishment of convergence without becoming intricately involved with the specific complexities of the TCP algorithm.

In his book Polak refers to points which satisfy some predetermined optimality condition as "desirable" points. His algorithm for locating these points in a closed subset \( T \) of \( \mathbb{R}^n \) utilizes a set-valued search function \( A \) which maps \( T \) into the set of all non-empty subsets of \( T \) (i.e., \( A: T \rightarrow \mathcal{P}(T) \)) and a stopping rule \( f: T \rightarrow \mathbb{R} \).
Algorithm. (Polak) [46]

Step 0. Compute \( x^0 \in T \).

Step 1. Set \( k = 0 \).

Step 2. Compute \( y \in A(x^k) \).

Step 3. Set \( x^{k+1} = y \).

Step 4. If \( f(x^{k+1}) \geq f(x^k) \) (or a direct test for determining \( x^k \) to be desirable), then set \( x^{k+1} = x^k \) and stop. Otherwise, set \( k = k+1 \) and return to Step 2.

What follows is a convergence theorem which addresses this algorithm.

Theorem 3.1 [46]

Suppose that

(i) \( f \) is either continuous at all nondesirable points \( x \in T \), or else \( f \) is bounded below for \( x \in T \);

(ii) for every \( x \in T \) which is not desirable,

\[ \exists \varepsilon(x) > 0 \text{ and a } \delta(x) < 0 \text{ such that} \]

\[ f(x') - f(x) \leq \delta(x) < 0, \]

for all \( x' \in T \) such that \( ||x' - x|| \leq \varepsilon(x) \), and for all \( x'' \in A(x') \).

Then, either the sequence \( x^0, x^1, x^2, \ldots \), constructed by Polak's algorithm is finite and its next to last element is desirable, or else it is infinite and every limit point of the sequence is desirable.
The next theorem makes use of the above result to show that the TCP algorithm converges to a desirable point. The proof follows, in part, the reasoning used in the modified method of centers proof found in [46].

**Theorem 3.2**

Let $\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^k, \ldots$, be a sequence generated by the TCP algorithm. Then this sequence is either finite, ending with $\mathbf{x}^{k+1} = \mathbf{x}^k$ and $M(\mathbf{x}^k) \geq 0$, or it is infinite and every limit point $\mathbf{x}^*$ of this sequence satisfies $M(\mathbf{x}^*) \geq 0$.

**Proof**

Since the TCP algorithm only stops after a finite number of iterations if $M(\mathbf{x}^k) = 0$, the case of a finite sequence is straightforward.

Next, consider the case in which this sequence if infinite. In relationship to Theorem 3.1 (and Polak's algorithm), $T = C$ and $f = -U$. $A$ is defined by the instructions in the TCP algorithm, and $\mathbf{x} \in C$ is termed desirable if $M(\mathbf{x}) \geq 0$. Theorem 3.1 will be shown to be satisfied by the maps $f$ and $A$. As a result, every limit point $\mathbf{x}^*$ of this sequence will satisfy $M(\mathbf{x}^*) \geq 0$.

By assumption, $U$ is continuously differentiable. Therefore, $-U$ satisfies part (i) of Theorem 3.1.

To establish part (ii), it is necessary to show that for any $\mathbf{x}^* \in C$ such that $M(\mathbf{x}^*) < 0$, there exist an $\varepsilon^* > 0$ and a $\delta^* < 0$ such that for all $\mathbf{x} \in D(\mathbf{x}^*, \varepsilon^*) = \{ \mathbf{x} \in C \mid \| \mathbf{x} - \mathbf{x}^* \| \leq \varepsilon^* \}$,

$$-U(\mathbf{x}') + U(\mathbf{x}) < \delta^*$$

for all $\mathbf{x}' \in A(\mathbf{x})$. 

Recall that for any \( x \in C \), \( M(x) \leq 0 \). Let \( x^* \) be any element in \( C \) for which \( M(x^*) = \mu < 0 \) (i.e., non-desirable). Since \( M \) is continuous, there exists an \( \varepsilon > 0 \) such that

\[
M(x) \leq \frac{\mu}{2} \quad \text{for all } x \in D(x^*, \varepsilon). \tag{1}
\]

By assumption, \( \frac{\partial U}{\partial x_1} > 0 \) for all \( x \in C \). \( U \) being continuously differentiable and \( C \) being compact, it follows from Lemma A.9 that

\[
\gamma = \inf \{ \frac{\partial U}{\partial x_1} \mid x \in C \} > 0. \tag{2}
\]

Since \( S = \{ h \in \mathbb{R}^n \mid |h_i| \leq 1, \ i = 1, 2, \ldots, n \} \) is a compact set and \( U \) is continuously differentiable, there exists (Lemma A.10) an \( \varepsilon^* \in (0, \varepsilon] \) and a \( \lambda' > 0 \) such that

\[
| \nabla_x U(x + \lambda h)^t \cdot h - \nabla_x U(x)^t \cdot h | \leq \frac{-\mu \gamma}{4} \tag{3}
\]

for all \( x \in D(x^*, \varepsilon^*) \), for all \( h \in S \), for all \( \lambda \in [0, \lambda'] \).

Next, let \( x \in D(x^*, \varepsilon^*) \) be arbitrary. For this \( x \), let \( h \) be any vector in \( S \) which solves the linear programming problem (3.6)-(3.9). As a result of this and (1),

\[
- \alpha \nabla_x U(x)^t \cdot h \leq M(x) \leq \frac{\mu}{2}. \tag{4}
\]

Since \( \alpha = 1 / (\partial U / \partial x_1) \),

\[
- \nabla_x U(x)^t \cdot h \leq \frac{\mu}{2\alpha} \leq \frac{\mu \gamma}{2} \tag{5}
\]

By the Mean-Value Theorem (Theorem A.1), there exists an \( \xi \in [0, \lambda'] \) such that
\[ U(x + \lambda'h) - U(x) = \lambda' \left( \nabla_x U(x + \xi h)^t \cdot h \right). \quad (6) \]

Making use of (6), (5), and (3)

\[ - U(x + \lambda'h) + U(x) = \lambda' \left( - \nabla_x U(x + \xi h)^t \cdot h \right) \]
\[ = \lambda' \left[ - \nabla_x U(x + \xi h)^t \cdot h - \left( - \nabla_x U(x)^t \cdot h \right) \right] \]
\[ = \lambda' \left[ - \nabla_x U(x + \xi h)^t \cdot h - \left( - \nabla_x U(x)^t \cdot h \right) \right] \]
\[ = \lambda' \left( - \frac{\mu \gamma}{4} + \frac{\mu \gamma}{2} \right) \]
\[ = \frac{\lambda' \gamma \mu}{4} < 0. \quad (7) \]

Next let \( x' = x + \theta'h \) where \( \theta' \) is determined by Steps 7-13 of the TCP algorithm (i.e., \( x' \in A(x) \)). Because of Lemma 3.5 and (7),

\[ - U(x') + U(x) = - U(x + \theta'h) + U(x) \]
\[ \leq - \frac{1}{\rho} \left[ \max_{\varphi} U(x + \varphi h) - U(x) \right]_{x + \varphi h \in C} \]
\[ \leq - \frac{1}{\rho} \left[ U(x + \lambda'h) - U(x) \right] \text{ where } \lambda' \text{ is as in (6)} \]
\[ \leq \frac{1}{\rho} \left( \frac{\lambda' \gamma \mu}{4} \right) = \delta' < 0. \]
Since (8) holds for all \( x \in D(\mathbf{x}^*, \mathbf{e}^*) \) and for all \( x' \in A(x) \), the theorem must be true.

Q.E.D.

Theorem 3.2 has just shown that the TCP algorithm converges. This raises the following question: to what type of point has the algorithm converged?

The optimality condition in the TCP algorithm concerns \( M \). A sequence is constructed which is either finite and the last point \( x \) satisfies \( M > 0 \), or else it is infinite and has the property that any of its limit points must also satisfy this condition. The reason for this test is that

\[
M(x) \geq 0 \quad (\text{LP formulation})
\]

\[
\min_{h \in S} \left( \max_{x} \left\{ -\alpha \nabla_x U(x)^{t} \cdot h, g_1(x) + \nabla_x g_1(x)^{t} \cdot h, \quad i=1,2,\ldots,m_x \right\} \right) \geq 0
\]

\[
\min_{h \in S} \max_{i \in A(x)} \left\{ -\alpha \nabla_x U(x)^{t} \cdot h, \nabla_x g_1(x)^{t} \cdot h \right\} \geq 0
\]

where \( A(x) = \{ i \mid g_i(x) = 0, \quad i=1,2,\ldots,m_x \} \)

\[
-\alpha \nabla_x U(x)^{t} \cdot h \geq 0 \quad \text{for all} \quad h \in D(x)
\]

where \( D(x) \) is set of all feasible directions at \( x \)

\[
\nabla_x U(x)^{t} \cdot h \leq 0 \quad \text{for all} \quad h \in D(x)
\]

[62] \( \Rightarrow \) \( \Rightarrow \)

\( x \) is optimal to (1.3)-(1.4) \( \Rightarrow \)

\( x \) is the best-compromise solution. (Section 1.5)
Thus, as anticipated, the TCP algorithm converges to the best-compromise solution.

3.6. Tradeoff Sensitivity

Decision makers are often unable to make confident assessments of their tradeoffs between objectives. They feel compelled (and rightly so) to justify their tradeoffs, making it at times extremely difficult for them to supply a specific value. Relaxing this requirement by asking instead a range within which the tradeoff should exist would be a more manageable request. On the other hand, this type of tradeoff ambiguity leads to an ill-defined problem. The decision maker's overall preference function becomes obscured, making it impossible to arrive at the best-compromise solution.

In connection with the TCP algorithm, a specific tradeoff value is necessary in determining the current tradeoff cut. Since past cuts are retained, a poor approximation could preclude one from ever finding the "true" best-compromise solution. A practical approach to avoid this type of problem would be to retain a given number of the latest tradeoff cuts. Thus, if a particular cut did eliminate the best-compromise solution, it would eventually be dropped, once again opening up the region containing the best-compromise solution. Still, this approach does not entirely alleviate the difficulties surrounding tradeoff approximations.

The problem of eliminating the best-compromise solution only manifests itself when a tradeoff cut is established for which this final solution is no longer feasible. In the TCP algorithm, decision maker supplied tradeoffs are used as a means of further restricting the decision space, not for approximating global preference functions.
or establishing the best search direction. For this reason the importance of correctly assessing tradeoffs is somewhat reduced. As long as the tradeoffs are such that the corresponding tradeoff cut does not eliminate the best-compromise solution, the process is not in jeopardy. The difficulty is that one does not know a priori where the best-compromise solution is located. As a result, one is unable to ascertain whether or not a given tradeoff cut has eliminated this point.

The problem surrounding tradeoff sensitivity is compounded when one realizes that tradeoffs are only known for those points at which the algorithm has interacted with the decision maker. One is unable to artificially generate the decision maker's tradeoffs. This precludes the possibility of studying alternative paths leading to the final solution point.

In an effort to give the decision maker a better understanding of the latitude he is afforded with respect to tradeoffs, the Tradeoff Cutting Plane program (Appendix C) is designed to perform a tradeoff sensitivity analysis upon termination of the algorithm. For each tradeoff, upper and lower bounds are computed for which violation of these bounds would cause any succeeding point and/or the best-compromise solution to be eliminated or contradict an assumption of the algorithm.

The program assumes the last point generated by the algorithm is the best-compromise solution. It then calculates the amount one is able to perturb each tradeoff before forcing the best-compromise solution to become infeasible. Similarly, it also calculates what latitude is allowed before any succeeding point becomes infeasible.
Mathematically, the logic works as follows. Letting

\[ w_i = \frac{\partial u/\partial f_i}{\partial u/\partial f_j}, \]

the tradeoff cut becomes

\[ \sum_{i=1}^{r} w_i(f_i - f')_i \geq 0, \]

where \( f'_i \) is the value of the \( i \)th objective at the current tradeoff point. Since the best-compromise must satisfy this cut,

\[ \sum_{i=1}^{r} w_i(f^b_i - f'_i) \geq 0, \]

where \( f^b_i \) is the value of the \( i \)th objective at the best-compromise.

For any given tradeoff \( w_j \),

\[ w_j(f^b_j - f'_j) + \sum_{i=1}^{r} \sum_{i \neq j} w_i(f^b_i - f'_i) \geq 0. \]

This establishes the following restrictions on \( w_j \):

\[
\left\{ \begin{array}{l}
- \sum_{i=1}^{r} \sum_{i \neq j} w_i(f^b_i - f'_i) \\
- \sum_{i=1}^{r} \sum_{i \neq j} w_i(f^b_i - f'_i) \\
0 \leq w_j \\
\leq \infty, \quad \text{if } (f^b_j - f'_j) = 0.
\end{array} \right.
\]
Figure 3.5 illustrates what takes place when there are only two objective functions. Assuming $f(x)$ to be the current tradeoff point, the objective space is partitioned into quadrants with $f(x)$ being the point of reference. The computer routine bases all tradeoffs on a unit decrease in $f_1$ and assumes all other tradeoffs are fixed.

If the best-compromise is located in quadrant I, the upper and lower bounds are infinity and zero respectively. The reason for this stems from the fact that the decision maker's overall preference function is assumed to have a negative slope. Therefore, for a unit decrease in $f_1$, one is allowed to vary $f_2$ from zero to infinity without affecting the feasibility of the best-compromise. Equally straightforward is the case wherein the best-compromise is found in quadrant III. Any point located in this quadrant is necessarily infeasible. This being the case, it is impossible for quadrant III to contain the best-compromise.

If the best-compromise happens to land in quadrant IV, the results become more interesting. In decreasing $f_1$ by one unit, one determines the amount of increase allowed in $f_2$ before infeasibility occurs on the part of the best-compromise (see Figure 3.5). The lower bound on this perturbation is obviously zero; the upper bound is the maximum increase allowed before eliminating the best-compromise. An analogous result holds for the situation in which the best-compromise is located in quadrant II. Only this time one calculates a lower bound, and the upper bound is set at infinity.

An example of the program's tradeoff sensitivity printout is given in Figure 3.6. For each tradeoff point, the functional values
Figure 3.5. Tradeoff Sensitivity in the Case of Two Objectives
### Tradeoff Sensitivity Based on Final Solution

#### Tradeoff Point 1 (F11) = -6.592

<table>
<thead>
<tr>
<th>TRADEOFF POINT 1 (F11) = -6.592</th>
<th>ALLOW (SUC PTS)</th>
<th>ALLOW (GC PT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADECORF</td>
<td>F</td>
<td>GIVEN</td>
</tr>
<tr>
<td>1 VS 2</td>
<td>-6.576</td>
<td>2.115E+01</td>
</tr>
<tr>
<td>1 VS 3</td>
<td>-5.725</td>
<td>5.110E-02</td>
</tr>
<tr>
<td>1 VS 4</td>
<td>-5.532</td>
<td>1.000E+00</td>
</tr>
<tr>
<td>1 VS 5</td>
<td>-5.107</td>
<td>1.000E+01</td>
</tr>
</tbody>
</table>

#### Tradeoff Point 2 (F12) = -6.440

<table>
<thead>
<tr>
<th>TRADEOFF POINT 2 (F12) = -6.440</th>
<th>ALLOW (SUC PTS)</th>
<th>ALLOW (GC PT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADECORF</td>
<td>F</td>
<td>GIVEN</td>
</tr>
<tr>
<td>1 VS 2</td>
<td>-6.433</td>
<td>5.000E+01</td>
</tr>
<tr>
<td>1 VS 3</td>
<td>-3.746</td>
<td>5.000E+02</td>
</tr>
<tr>
<td>1 VS 4</td>
<td>-6.485</td>
<td>1.000E+01</td>
</tr>
<tr>
<td>1 VS 5</td>
<td>-5.462</td>
<td>1.000E+01</td>
</tr>
</tbody>
</table>

#### Tradeoff Point 3 (F13) = -6.527

<table>
<thead>
<tr>
<th>TRADEOFF POINT 3 (F13) = -6.527</th>
<th>ALLOW (SUC PTS)</th>
<th>ALLOW (GC PT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADECORF</td>
<td>F</td>
<td>GIVEN</td>
</tr>
<tr>
<td>1 VS 2</td>
<td>-6.316</td>
<td>1.000E+02</td>
</tr>
<tr>
<td>1 VS 3</td>
<td>-4.731</td>
<td>1.000E-01</td>
</tr>
<tr>
<td>1 VS 4</td>
<td>-6.472</td>
<td>1.000E+01</td>
</tr>
<tr>
<td>1 VS 5</td>
<td>-6.523</td>
<td>1.000E+01</td>
</tr>
</tbody>
</table>

#### Tradeoff Point 4 (F14) = -6.560

<table>
<thead>
<tr>
<th>TRADEOFF POINT 4 (F14) = -6.560</th>
<th>ALLOW (SUC PTS)</th>
<th>ALLOW (GC PT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADECORF</td>
<td>F</td>
<td>GIVEN</td>
</tr>
<tr>
<td>1 VS 2</td>
<td>-7.654</td>
<td>2.000E+01</td>
</tr>
<tr>
<td>1 VS 3</td>
<td>-5.264</td>
<td>5.000E-02</td>
</tr>
<tr>
<td>1 VS 4</td>
<td>-5.144</td>
<td>1.000E+00</td>
</tr>
<tr>
<td>1 VS 5</td>
<td>-7.710</td>
<td>1.000E+01</td>
</tr>
</tbody>
</table>

---

Figure 3.6. Tradeoff Sensitivity Printout
are printed together with the tradeoffs supplied by the decision maker. The values listed in the column entitled "given" are the tradeoffs supplied by the decision maker adjusted to correspond to a unit perturbation in $f_1$. The minimum and maximum perturbations allowed are given for both the best-compromise point and all points which follow the current tradeoff point (labeled "BC PT" and "SUC PTS" respectively). If the upper bound is infinity, the program is instructed to print $1.000E+30$. In the case of an error (i.e., the best-compromise is located in quadrant III), the program prints $-9.00E+00$ for both the minimum and maximum values.

Under "TRDBOFF POINT 1" in Figure 3.6, $f_1$ and $f_2$ are shown to have values of $-6.598$ and $-6.000$ respectively. In addition, it shows the decision maker willing to trade 20 units of $f_2$ to gain 1 unit of $f_1$. Based on the final solution, the decision maker could have traded as little as $0.3396$ of a unit of $f_2$ for the 1 unit gain in $f_1$, without eliminating any successive point (including the best-compromise point). As for $f_3$, it is shown to have a value of $-5.707$, for which the decision maker is willing to accept 1 unit of $f_1$ in trade for $0.05$ of a unit of $f_3$. The decision maker could actually have traded as much as $0.1783$ of a unit of $f_3$ without eliminating any of the successive points and as much as $0.2988$ before eliminating the best-compromise point.

Practically speaking, care must be taken to understand what these limits mean. One should not assume that by using one of these extreme tradeoff values, the final solution will necessarily be the same. Indeed, even one change at a given tradeoff point would most likely
cause a different sequence of points to be followed. Yet, what these limits do represent is the degree of criticality associated with each tradeoff. If, for example, a given tradeoff is near the maximum allowable, one would realize what amount of tradeoff error would jeopardize the solution. So if a decision maker establishes a range within which a particular tradeoff is known to exist, this range can be judged against these limits to indicate to what extent his tradeoff indecision undermines the final result. An important fact to underscore is that in judging this acceptable margin of error, all other tradeoffs are assumed correct and fixed.
CHAPTER IV

STORM DRAINAGE - AN APPLICATION

In this chapter the Tradeoff Cutting Plane algorithm is used to analyze the expected storm drainage needs of an urban subbasin. Although a subbasin may be interesting from several water resource perspectives, this application focuses exclusively on the problem of storm drainage. Throughout the discussion, reference is made to the specific application of this analysis to the storm drainage problem associated with the city of West Lafayette, Indiana. First, a brief overview of this problem is given.

4.1. Overview

The medium size community of West Lafayette is able to be partitioned into a number of hydrologically independent watersheds. Each one may be viewed as having a separate drainage system composed of detention storage facilities, treatment plants, drainage networks, and a means of restitution into the receiving body of water. These watersheds can be further subdivided into subbasins. One such subbasin (number 13) is examined in terms of its storm drainage system. This subbasin is assumed, for the purposes of this analysis, to be hydrologically independent of the other subbasins, having its own drainage network, on-site detention storage facility, treatment plant, and tributary to a receiving water body.
The various factors involved in this problem as well as how they conceptually interlink are illustrated in Figure 4.1. Runoff due to rainfall and snowfall is channeled to a treatment facility before being discharged into the receiving body of water. That which is not able to be immediately treated is re-routed to a temporary storage facility to await treatment. Once the storage tank has filled, any further runoff either overflows into the receiving body of water, carrying with it pollutants, or backs-up, causing local flooding to occur.

Subbasin 13's storm drainage system is assumed to be characterized by three decision variables:

- $x_1$ - local detention storage capacity (basin-inches),
- $x_2$ - maximum treatment rate (basin-inches/hour), and
- $x_3$ - maximum allowable overflow rate (basin-inches/hour).

Restricting the overflow rate makes it possible to incorporate into the analysis damages caused by local flooding. Once the detention storage facility is filled and the overflow rate is at its maximum, the drainage system's conveying capacity is significantly reduced. This results in local flooding, the consequence of which is structural damage and economic disruption to the area.

The hydrologic performance of subbasin 13 was examined by using an urban hydrologic simulation program termed LANDSTORM [14]. The program coupled a landuse forecasting model with an urban hydrologic simulation model called STORM [47, 56]. Based on such inputs as surface characteristics, growth requirements and zoning policies, the simulation program generated an urban growth scenario for the
Figure 4.1. Conceptualization of Subbasin 13's Storm Drainage Problem
subbasin on a lot-by-lot basis. Following this, it simulated the performance of the given storm drainage system (i.e., \(x_1, x_2, x_3\)) over a specified period of time and under weather conditions representative of the area. Twenty-three years of recorded precipitation data from the West Lafayette area was used as input to the simulation. The model was then simulated for a 20 year period to obtain an estimate of the hydrologic performance of the subbasin's drainage system.

Output from the simulation consisted of the drainage system's yearly statistical performance. Relevant data included:

1. Average number of floods per year \((y_1)\)
2. Average flood volume per year \((y_2)\)
3. Average number of pounds per year of suspended solids \((y_{31})\)
4. Average number of pounds per year of settleable solids \((y_{32})\)
5. Average number of pounds per year of biochemical oxygen demand - BOD \((y_{33})\)
6. Average number of pounds per year of total nitrogen - N \((y_{34})\)
7. Average number of pounds per year of orthophosphate - PO_4 \((y_{35})\).
4.2. Original Formulation

The original conceptualization, formulation, and analysis of this problem was done by Dr. Stergios Dendrou of the Civil Engineering Department of Purdue University. His comprehensive study ([14]) was of a much broader scope than the problem considered here. He investigated the larger scaled watershed drainage problem with its interdependent subbasins. The subbasins were not assumed to be necessarily hydrologically independent of each other. For example, situations were allowed for which the same treatment facility might service several subbasins. In order to coordinate the drainage of these subbasins, he used a multilevel optimization scheme. This scheme sought to minimize the cost of procuring, operating, and maintaining the watershed drainage system through the coordination of the individual subbasins. At the subbasin level, attention was paid to the detention storage capacities, treatment rates, and allowable overflow rates; while at the watershed level, interest turned to the allocation of these detention storage facilities and treatment plants among the various subbasins. By iterating between these two levels, the overall optimum was eventually identified.

4.3. Multiple Objective Formulation

In his study, Dendrou aggregated the various objectives into a single economic objective function. In doing so, he was implicitly assuming a constant rate of substitution among the objectives. That is, his local tradeoffs were invariable and global in that the same rate of substitution was applicable at any point in objective space. In the application described here, although limited to a single subbasin, the same "lower-level" storm drainage problem is considered,
but from a more representative, multiple objective standpoint. In fact, various sets of objectives are used as well as various trade-off policies between the objectives.

In Dendrou's model cost functions were representative of the storage and drainage network facilities, treatment plants, and damages (both structural and economic) caused by local flooding. Constraints on his system included hydraulic continuity equations, detention storage capacities, overflow threshold restrictions, and reliability conditions regarding pollutant concentrations during overflows, frequency of local floods, and street-flooding exceeding a threshold depth.

Only a summary of subbasin 13's analytical model is given here since details into the reasoning behind the model are well documented in Dendrou [14]. Table 4.1 lists each factor of interest together with its representative algebraic expression. Note that the objectives are now treated individually. All cost expressions take into account both the capital cost and the operating and maintenance cost appropriately discounted over the entire planning horizon.

Inasmuch as the basic submodel of this problem is a simulation program, it became necessary to approximate the behavior of the simulation output parameters by response surfaces. Over the region of interest,

\[ 0.01 \leq x_1 \leq 0.45 \]
\[ 0.01 \leq x_2 \leq 0.10 \]
\[ 0.01 \leq x_3 \leq 0.10 \]
Table 4.1. Analytical Model of Subbasin 13's Storm Drainage Problem

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>NAME</th>
<th>ALGEBRAIC EXPRESSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>Drainage Network Cost</td>
<td>( 106,780.37x_2 + 106,780.37x_3 + 61,704.67 )</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>Storage Facility Cost</td>
<td>( 3,000x_1 )</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>Treatment Facility Cost</td>
<td>( (305,700/(.06 \times 2289)^{.65}) \times 2289 \times 2 )</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>Expected Flood Damage Cost</td>
<td>( 250 \times 2289 \times y_2 )</td>
</tr>
<tr>
<td>( f_5 )</td>
<td>Expected Economic Loss Due to Flooding</td>
<td>( 25,000y_1 )</td>
</tr>
<tr>
<td>( g_1 )</td>
<td>Average Number of Floods per Year</td>
<td>( y_1 )</td>
</tr>
<tr>
<td>( g_2 )</td>
<td>Probability of Exceeding a Flood Depth of 0.01 Basin-inches</td>
<td>( \frac{0.0000006}{x_1 x_2} + 0.1082x_3 - 0.00986 )</td>
</tr>
<tr>
<td>( g_{31} )</td>
<td>Average Number of Pounds per Year of Suspended Solids</td>
<td>( y_{31} )</td>
</tr>
<tr>
<td>( g_{32} )</td>
<td>Average Number of Pounds per Year of Settleable Solids</td>
<td>( y_{32} )</td>
</tr>
<tr>
<td>( g_{33} )</td>
<td>Average Number of Pounds per Year of BOD</td>
<td>( y_{33} )</td>
</tr>
<tr>
<td>( g_{34} )</td>
<td>Average Number of Pounds per Year of ( N )</td>
<td>( y_{34} )</td>
</tr>
<tr>
<td>( g_{35} )</td>
<td>Average Number of Pounds per Year of ( PO_4 )</td>
<td>( y_{35} )</td>
</tr>
</tbody>
</table>
a response surface was established for each of the seven output
parameters listed in Section 4.1. Since the TCP algorithm assumes
convexity, it was necessary to selectively choose regression models
which not only satisfied this assumption but also allowed for a
respectable fit. Numerous models were attempted for each of the
seven parameters. In each case the model with the best "quality" fit (in terms of the coefficient of multiple determination \( R^2 \)) was
selected. The resulting multiple regression curves together with
their corresponding \( R^2 \) values are given in Table 4.2.

To gain further insight into the quality of the fit associated
with each expression found in Table 4.1, response surfaces were
plotted over a two-dimensional plane (one variable being held fixed
if necessary). Figures 4.2 and 4.3 show representative examples
from a number of these plots. The top plot in each figure illustrates
the results obtained from the storm drainage simulation model; the
bottom plot shows the surface obtained by using the corresponding
multiple regression curve.

4.4. Problem Statement

The multiple objective formulation of subbasin 13's storm drainage
problem can now be explicitly stated as follows:

minimize

\[
\begin{align*}
    f_1 &= 106,780.37x_2 + 106,780.37x_3 + 61,704.67 \\
    f_2 &= 3,000x_1 \\
    f_3 &= (305,700/(0.06 \cdot 2289)^{65}) \cdot 2289x_2 \\
    f_4 &= 250(2289) \exp(-39.75x_2 + 9.9x_3 + 2.74) \\
    f_5 &= 25 \left( \frac{1.39}{x_1 x_2} + 4.940x_3 - 80 \right)
\end{align*}
\]
Table 4.2. Regression Models of the Simulation Output Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>REGRESSION CURVE</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>$\frac{0.00139}{x_1x_2} + 4.94x_3 - 0.08$</td>
<td>.921</td>
</tr>
<tr>
<td>$y_2$</td>
<td>$\exp(-39.75x_2 + 9.9x_3 + 2.74)$</td>
<td>.951</td>
</tr>
<tr>
<td>$y_{31}$</td>
<td>$\frac{12.307}{x_1x_2} + 49.408x_2 + 2x_3 - 4.051.02$</td>
<td>.906</td>
</tr>
<tr>
<td>$y_{32}$</td>
<td>$\frac{2.098}{x_1x_2} + 8046.33x_3 - 696.71$</td>
<td>.914</td>
</tr>
<tr>
<td>$y_{33}$</td>
<td>$\frac{2.138}{x_1x_2} + 7883.39x_3 - 705.04$</td>
<td>.910</td>
</tr>
<tr>
<td>$y_{34}$</td>
<td>$\frac{0.447}{x_1x_2} + 1.721.26x_2 + 136.54$</td>
<td>.900</td>
</tr>
<tr>
<td>$y_{35}$</td>
<td>$\frac{0.164}{x_1x_2} + 631.13x_3 - 54.48$</td>
<td>.912</td>
</tr>
</tbody>
</table>
Figure 4.2. Response Surface and Regression Model of the Expected Economic Loss Due to Flooding
Table 4.2. Regression Models of the Simulation Output Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>REGRESSION CURVE</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>$\frac{0.00139}{x_1 x_2} + 4.94 x_3 - 0.08$</td>
<td>.921</td>
</tr>
<tr>
<td>$y_2$</td>
<td>$\exp(-39.75 x_2 + 9.9 x_3 + 2.74)$</td>
<td>.951</td>
</tr>
<tr>
<td>$y_{31}$</td>
<td>$\frac{0.307}{x_1 x_2} + 49.408.24 x_3 - 4.051.02$</td>
<td>.906</td>
</tr>
<tr>
<td>$y_{32}$</td>
<td>$\frac{2.098}{x_1 x_2} + 8.046.33 x_3 - 696.71$</td>
<td>.914</td>
</tr>
<tr>
<td>$y_{33}$</td>
<td>$\frac{2.138}{x_1 x_2} + 7.883.39 x_3 - 705.04$</td>
<td>.910</td>
</tr>
<tr>
<td>$y_{34}$</td>
<td>$\frac{0.417}{x_1 x_2} + 1.721.26 x_3 - 136.54$</td>
<td>.900</td>
</tr>
<tr>
<td>$y_{35}$</td>
<td>$\frac{0.164}{x_1 x_2} + 631.13 x_3 - 54.48$</td>
<td>.912</td>
</tr>
</tbody>
</table>
Figure 4.2. Response Surface and Regression Model of the Expected Economic Loss Due to Flooding
Figure 4.3. Response Surface and Regression Model of the Average Number of Pounds per Year of Suspended Solids
subject to

\[
\frac{0.00139}{x_1 x_2} + 4.94 x_3 - 0.08 \leq c_1
\]

\[
\frac{0.0000306}{x_1 x_2} + 0.1082 x_3 - 0.00986 \leq c_2
\]

\[
\frac{12.307}{x_1 x_2} + 494.0824 x_3 - 4051.02 \leq c_3
\]

\[
\frac{2.098}{x_1 x_2} + 8046.33 x_3 - 696.71 \leq c_4
\]

\[
\frac{2.138}{x_1 x_2} + 7883.39 x_3 - 705.04 \leq c_5
\]

\[
\frac{0.417}{x_1 x_2} + 1721.26 x_3 - 136.54 \leq c_6
\]

\[
\frac{0.164}{x_1 x_2} + 631.13 x_3 - 54.48 \leq c_7
\]

0.01 \leq x_1 \leq 0.45

0.01 \leq x_2 \leq 0.10

0.01 \leq x_3 \leq 0.10

The values used for \( c_1 \) through \( c_7 \) are given in Table 4.3 along with the meaning of each constraint.
Table 4.3. Bounds Associated with Constraints $g_1$ through $g_{33}$

<table>
<thead>
<tr>
<th>CONSTRAINT</th>
<th>MAXIMUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average number of floods per year</td>
<td>1</td>
</tr>
<tr>
<td>Probability of exceeding a flood depth of 0.01 basin-inches</td>
<td>0.10</td>
</tr>
<tr>
<td>Average number of pounds per year of suspended solids</td>
<td>50,000</td>
</tr>
<tr>
<td>Average number of pounds per year of settleable solids</td>
<td>16,000</td>
</tr>
<tr>
<td>Average number of pounds per year of BOD</td>
<td>10,000</td>
</tr>
<tr>
<td>Average number of pounds per year of N</td>
<td>2,000</td>
</tr>
<tr>
<td>Average number of pounds per year of PO$_4$</td>
<td>550</td>
</tr>
</tbody>
</table>
4.5. Tradeoff Policy A

The first tradeoff policy (A) used with the TCP algorithm in solving this multiple objective problem was a global policy. It reflected Dendrou's formulation of this problem for it aggregated the various objectives into a single function representative of the drainage system's total cost (Figure 4.4). This total sum objective was run primarily for comparison purposes. Since the single objective optimization problem can be solved using conventional methods, this policy allowed for a comparison of the solution with known results. Moreover, it gave an indication of the type of solution arrived at under Dendrou's formulation.

In order to study convergence characteristics and to allow for a check on the final solution, three separate starting points were used. These same three points were used with each policy so as to provide a means of studying the effect of policy changes. The three starting points were:

SP1 \((.40, .01, .08)\),
SP2 \((.20, .02, .02)\), and
SP3 \((.30, .08, .01)\).

Each time, the algorithm was run for ten iterations, unless there was early termination due to the best-compromise solution being reached.

The optimal solution under policy A, found by applying the constrained minimization technique of Griffith and Stewart [23], is \((0.4198, 0.0657, 0.0100)\) with a final objective function value of 2,594,869. Tables 4.4, 4.5, and 4.6 reveal the sequence of points
Figure 4.4. Total Cost of the Drainage System
Table 4.4. Sequence of Points Generated from SPL under Policy A

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>Total Cost</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.4000</td>
<td>.0100</td>
<td>.0800</td>
<td>13,523,101</td>
<td>421.1</td>
</tr>
<tr>
<td>1</td>
<td>.3067</td>
<td>.0895</td>
<td>.0214</td>
<td>2,942,645</td>
<td>13.4</td>
</tr>
<tr>
<td>2</td>
<td>.3196</td>
<td>.0685</td>
<td>.0211</td>
<td>2,746,206</td>
<td>5.8</td>
</tr>
<tr>
<td>3</td>
<td>.3276</td>
<td>.0703</td>
<td>.0204</td>
<td>2,743,639</td>
<td>5.7</td>
</tr>
<tr>
<td>4</td>
<td>.3331</td>
<td>.0672</td>
<td>.0199</td>
<td>2,738,432</td>
<td>5.5</td>
</tr>
<tr>
<td>5</td>
<td>.3416</td>
<td>.0697</td>
<td>.0192</td>
<td>2,734,377</td>
<td>5.4</td>
</tr>
<tr>
<td>6</td>
<td>.3459</td>
<td>.0670</td>
<td>.0189</td>
<td>2,730,480</td>
<td>5.2</td>
</tr>
<tr>
<td>7</td>
<td>.3526</td>
<td>.0693</td>
<td>.0183</td>
<td>3,727,256</td>
<td>5.1</td>
</tr>
<tr>
<td>8</td>
<td>.3561</td>
<td>.0669</td>
<td>.0180</td>
<td>2,724,145</td>
<td>5.0</td>
</tr>
<tr>
<td>9</td>
<td>.3616</td>
<td>.0689</td>
<td>.0176</td>
<td>2,721,493</td>
<td>4.9</td>
</tr>
<tr>
<td>10</td>
<td>.3644</td>
<td>.0668</td>
<td>.0173</td>
<td>2,718,939</td>
<td>4.8</td>
</tr>
<tr>
<td>Iteration</td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>Total Cost</td>
<td>% Error</td>
</tr>
<tr>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>0</td>
<td>0.2000</td>
<td>0.0200</td>
<td>0.0200</td>
<td>5,524,345</td>
<td>112.9</td>
</tr>
<tr>
<td>1</td>
<td>0.3316</td>
<td>0.0895</td>
<td>0.0222</td>
<td>2,945,373</td>
<td>13.5</td>
</tr>
<tr>
<td>2</td>
<td>0.3363</td>
<td>0.0684</td>
<td>0.0216</td>
<td>2,750,223</td>
<td>6.0</td>
</tr>
<tr>
<td>3</td>
<td>0.3437</td>
<td>0.0709</td>
<td>0.0209</td>
<td>2,747,224</td>
<td>5.9</td>
</tr>
<tr>
<td>4</td>
<td>0.3485</td>
<td>0.0572</td>
<td>0.0204</td>
<td>2,741,514</td>
<td>5.7</td>
</tr>
<tr>
<td>5</td>
<td>0.3561</td>
<td>0.0699</td>
<td>0.0196</td>
<td>2,737,138</td>
<td>5.5</td>
</tr>
<tr>
<td>6</td>
<td>0.3598</td>
<td>0.0670</td>
<td>0.0192</td>
<td>2,732,921</td>
<td>5.3</td>
</tr>
<tr>
<td>7</td>
<td>0.3657</td>
<td>0.0694</td>
<td>0.0186</td>
<td>2,729,444</td>
<td>5.2</td>
</tr>
<tr>
<td>8</td>
<td>0.3687</td>
<td>0.0669</td>
<td>0.0183</td>
<td>2,726,118</td>
<td>5.1</td>
</tr>
<tr>
<td>9</td>
<td>0.3735</td>
<td>0.0690</td>
<td>0.0178</td>
<td>2,723,316</td>
<td>5.0</td>
</tr>
<tr>
<td>10</td>
<td>0.3759</td>
<td>0.0668</td>
<td>0.0176</td>
<td>2,720,586</td>
<td>4.8</td>
</tr>
</tbody>
</table>
Table 4.6. Sequence of Points Generated from SP3 under Policy A

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>Total Cost</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.3000</td>
<td>.0800</td>
<td>.0100</td>
<td>2,762,576</td>
<td>6.5</td>
</tr>
<tr>
<td>1</td>
<td>.3159</td>
<td>.0668</td>
<td>.0116</td>
<td>2,676,679</td>
<td>3.2</td>
</tr>
<tr>
<td>2</td>
<td>.3179</td>
<td>.0659</td>
<td>.0116</td>
<td>2,676,250</td>
<td>3.1</td>
</tr>
<tr>
<td>3</td>
<td>.3194</td>
<td>.0663</td>
<td>.0115</td>
<td>2,676,163</td>
<td>3.1</td>
</tr>
<tr>
<td>4</td>
<td>.3204</td>
<td>.0659</td>
<td>.0115</td>
<td>2,676,058</td>
<td>3.1</td>
</tr>
<tr>
<td>5</td>
<td>.3220</td>
<td>.0663</td>
<td>.0115</td>
<td>2,675,957</td>
<td>3.1</td>
</tr>
<tr>
<td>6</td>
<td>.3230</td>
<td>.0659</td>
<td>.0115</td>
<td>2,575,857</td>
<td>3.1</td>
</tr>
<tr>
<td>7</td>
<td>.3245</td>
<td>.0663</td>
<td>.0115</td>
<td>2,675,755</td>
<td>3.1</td>
</tr>
<tr>
<td>8</td>
<td>.3254</td>
<td>.0659</td>
<td>.0115</td>
<td>2,675,661</td>
<td>3.1</td>
</tr>
<tr>
<td>9</td>
<td>.3268</td>
<td>.0663</td>
<td>.0115</td>
<td>2,675,570</td>
<td>3.1</td>
</tr>
<tr>
<td>10</td>
<td>.3277</td>
<td>.0659</td>
<td>.0115</td>
<td>2,575,480</td>
<td>3.1</td>
</tr>
</tbody>
</table>
generated by the TCP algorithm from each starting point. Also listed is, for each point in the sequence, the total cost and the amount of error from the optimal solution (i.e.,

\[
\% \text{ Error } = \frac{\text{Total Cost} - 2,594,869}{2,594,869} \cdot 100\%.
\]

For each starting condition, Figure 4.5 shows the total cost as a function of the iteration number. Notice from this figure the relatively large initial improvement in the total cost function. One quickly moves into the neighborhood of the true solution after only three iterations. From a practical standpoint this is significant for decision makers are typically unwilling, under a man-machine interactive environment, to spend much time searching for the best solution.

4.6. Tradeoff Policy B

Turning to the second tradeoff policy (B), an effort was made to once again consider a global policy, but this time make it dependent upon the values of the objectives at the various local tradeoff points. The policy stated that the amount one was willing to trade of a given objective for a unit decrease in \( f_1 \) was considered inversely proportional to a percentage value of the objective. In other words, if a given objective had a high cost in comparison to \( f_1 \), the decision maker favored (in reverse proportion to their values) increasing the cost associated with \( f_1 \) over that of the given objective.

Under this policy, one finds again this rather rapid stabilization in the values of the objectives. As seen in Figures 4.6 through 4.10, all five objectives have begun to dampen their oscillation pattern
Figure 4.5. Drainage System Cost as a Function of the Iteration Number
Figure 4.6. Drainage Network Cost Under Policy B
Figure 4.7. Storage Facility Cost Under Policy B
Figure 4.8. Treatment Facility Cost Under Policy B
Figure 4.9. Expected Flood Damage Cost Under Policy B
Figure 4.10. Expected Economic Loss Due to Flooding Under Policy B
after only seven iterations. The best-compromise level of each objective function is quickly sought irrespective of the starting condition. Tables 4.7 through 4.9 list the sequence taken from each starting point.

4.7. Tradeoff Policy C

The third tradeoff policy (C) retained the intrinsic value associated with each of the five objectives and used a decision maker to assess the local tradeoffs. Dr. Dendrou, the original modeler of the system, supplied the necessary information. He served as a well-informed decision maker, fully understanding the intricate characteristics of the drainage system as well as the type of information required on each iteration of the algorithm.

Policy C's sequence of points for each starting condition is given in Table 4.10. Notice that the local tradeoffs supplied by the decision maker resulted in inconsistent solutions. $x_1$ and $x_3$ tended to converge to the same level, whereas $x_2$ ranged from .0146 to .0829. Comments concerning this type of discrepancy will be made shortly. Observe also that the results here do not coincide with those under policy A. Apparently, the total cost objective under policy A does not resemble the decision maker's preference function represented under policy C. The decision maker failed to follow (as is his preference) the same pattern of behavior expected of a predominately cost conscious person. Influences beyond the obvious monetary ones swayed his tradeoff assessments.

4.8. Tradeoff Policy D

A reformulation of the multiple objective problem accompanied this final tradeoff policy (D). The cost functions $f_1$, $f_2$, and $f_3$
Table 4.7. Sequence of Points Generated from SPL under Policy B

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>$f_4$</th>
<th>$f_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.4000</td>
<td>.0100</td>
<td>.0800</td>
<td>71,315</td>
<td>1,200</td>
<td>285,347</td>
<td>13,148,671</td>
<td>16,568</td>
</tr>
<tr>
<td>1</td>
<td>.3533</td>
<td>.0498</td>
<td>.0507</td>
<td>72,431</td>
<td>1,060</td>
<td>1,419,765</td>
<td>2,025,718</td>
<td>6,238</td>
</tr>
<tr>
<td>2</td>
<td>.2724</td>
<td>.0535</td>
<td>.0129</td>
<td>68,788</td>
<td>817</td>
<td>1,525,639</td>
<td>1,201,929</td>
<td>1,975</td>
</tr>
<tr>
<td>3</td>
<td>.2659</td>
<td>.0561</td>
<td>.0127</td>
<td>69,057</td>
<td>798</td>
<td>1,601,739</td>
<td>1,079,472</td>
<td>1,899</td>
</tr>
<tr>
<td>4</td>
<td>.1829</td>
<td>.0559</td>
<td>.0103</td>
<td>68,773</td>
<td>549</td>
<td>1,594,551</td>
<td>1,064,614</td>
<td>2,673</td>
</tr>
<tr>
<td>5</td>
<td>.1800</td>
<td>.0566</td>
<td>.0103</td>
<td>68,849</td>
<td>540</td>
<td>1,615,202</td>
<td>1,034,369</td>
<td>2,683</td>
</tr>
<tr>
<td>6</td>
<td>.1785</td>
<td>.0557</td>
<td>.0103</td>
<td>68,754</td>
<td>535</td>
<td>1,589,800</td>
<td>1,071,605</td>
<td>2,767</td>
</tr>
<tr>
<td>7</td>
<td>.1400</td>
<td>.0563</td>
<td>.0102</td>
<td>68,809</td>
<td>420</td>
<td>1,606,687</td>
<td>1,045,876</td>
<td>3,672</td>
</tr>
<tr>
<td>8</td>
<td>.1395</td>
<td>.0562</td>
<td>.0102</td>
<td>68,792</td>
<td>419</td>
<td>1,602,329</td>
<td>1,052,230</td>
<td>3,698</td>
</tr>
<tr>
<td>9</td>
<td>.1387</td>
<td>.0563</td>
<td>.0102</td>
<td>68,812</td>
<td>416</td>
<td>1,607,697</td>
<td>1,044,359</td>
<td>3,710</td>
</tr>
<tr>
<td>10</td>
<td>.1356</td>
<td>.0563</td>
<td>.0102</td>
<td>68,807</td>
<td>407</td>
<td>1,606,631</td>
<td>1,045,804</td>
<td>3,812</td>
</tr>
<tr>
<td>Iteration</td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$f_1$</td>
<td>$f_2$</td>
<td>$f_3$</td>
<td>$f_4$</td>
<td>$f_5$</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>0</td>
<td>0.2000</td>
<td>0.0200</td>
<td>0.0200</td>
<td>65,976</td>
<td>600</td>
<td>570,694</td>
<td>4,878,417</td>
<td>9,158</td>
</tr>
<tr>
<td>1</td>
<td>0.2658</td>
<td>0.0548</td>
<td>0.0211</td>
<td>69,805</td>
<td>797</td>
<td>1,562,474</td>
<td>1,238,804</td>
<td>2,994</td>
</tr>
<tr>
<td>2</td>
<td>0.2486</td>
<td>0.0613</td>
<td>0.0190</td>
<td>70,287</td>
<td>746</td>
<td>1,750,064</td>
<td>934,682</td>
<td>2,631</td>
</tr>
<tr>
<td>3</td>
<td>0.1651</td>
<td>0.0509</td>
<td>0.0105</td>
<td>68,258</td>
<td>495</td>
<td>1,452,116</td>
<td>1,300,488</td>
<td>3,131</td>
</tr>
<tr>
<td>4</td>
<td>0.1323</td>
<td>0.0569</td>
<td>0.0106</td>
<td>68,912</td>
<td>397</td>
<td>1,624,468</td>
<td>1,023,729</td>
<td>3,919</td>
</tr>
<tr>
<td>5</td>
<td>0.1384</td>
<td>0.0569</td>
<td>0.0106</td>
<td>68,907</td>
<td>415</td>
<td>1,623,593</td>
<td>1,024,861</td>
<td>3,717</td>
</tr>
<tr>
<td>6</td>
<td>0.1348</td>
<td>0.0564</td>
<td>0.0105</td>
<td>68,854</td>
<td>404</td>
<td>1,609,902</td>
<td>1,044,363</td>
<td>3,871</td>
</tr>
<tr>
<td>7</td>
<td>0.1385</td>
<td>0.0560</td>
<td>0.0105</td>
<td>68,810</td>
<td>416</td>
<td>1,599,191</td>
<td>1,059,776</td>
<td>3,773</td>
</tr>
<tr>
<td>8</td>
<td>0.1406</td>
<td>0.0569</td>
<td>0.0100</td>
<td>68,853</td>
<td>422</td>
<td>1,624,527</td>
<td>1,018,067</td>
<td>3,577</td>
</tr>
<tr>
<td>9</td>
<td>0.1385</td>
<td>0.0560</td>
<td>0.0100</td>
<td>68,758</td>
<td>415</td>
<td>1,599,151</td>
<td>1,054,703</td>
<td>3,714</td>
</tr>
<tr>
<td>10</td>
<td>0.1364</td>
<td>0.0561</td>
<td>0.0100</td>
<td>68,768</td>
<td>409</td>
<td>1,601,883</td>
<td>1,050,694</td>
<td>3,776</td>
</tr>
</tbody>
</table>
Table 4.9. Sequence of Points Generated from SP3 under Policy B

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>$f_4$</th>
<th>$f_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.3000</td>
<td>.0800</td>
<td>.0100</td>
<td>71,315</td>
<td>900</td>
<td>2,282,775</td>
<td>406,903</td>
<td>683</td>
</tr>
<tr>
<td>1</td>
<td>.2661</td>
<td>.0517</td>
<td>.0125</td>
<td>68,557</td>
<td>798</td>
<td>1,475,441</td>
<td>1,283,850</td>
<td>2,065</td>
</tr>
<tr>
<td>2</td>
<td>.2514</td>
<td>.0558</td>
<td>.0122</td>
<td>68,971</td>
<td>754</td>
<td>1,592,982</td>
<td>1,087,307</td>
<td>1,985</td>
</tr>
<tr>
<td>3</td>
<td>.1682</td>
<td>.0561</td>
<td>.0103</td>
<td>68,793</td>
<td>505</td>
<td>1,600,790</td>
<td>1,055,084</td>
<td>2,952</td>
</tr>
<tr>
<td>4</td>
<td>.1667</td>
<td>.0555</td>
<td>.0103</td>
<td>68,726</td>
<td>500</td>
<td>1,583,111</td>
<td>1,081,358</td>
<td>3,027</td>
</tr>
<tr>
<td>5</td>
<td>.1613</td>
<td>.0567</td>
<td>.0103</td>
<td>68,860</td>
<td>484</td>
<td>1,619,196</td>
<td>1,028,250</td>
<td>3,065</td>
</tr>
<tr>
<td>6</td>
<td>.1380</td>
<td>.0560</td>
<td>.0102</td>
<td>68,777</td>
<td>414</td>
<td>1,598,297</td>
<td>1,058,135</td>
<td>3,758</td>
</tr>
<tr>
<td>7</td>
<td>.1355</td>
<td>.0563</td>
<td>.0102</td>
<td>68,807</td>
<td>406</td>
<td>1,606,550</td>
<td>1,045,978</td>
<td>3,817</td>
</tr>
<tr>
<td>8</td>
<td>.1393</td>
<td>.0565</td>
<td>.0102</td>
<td>68,822</td>
<td>418</td>
<td>1,611,270</td>
<td>1,033,851</td>
<td>3,675</td>
</tr>
<tr>
<td>9</td>
<td>.1352</td>
<td>.0562</td>
<td>.0102</td>
<td>68,787</td>
<td>405</td>
<td>1,602,494</td>
<td>1,051,399</td>
<td>3,834</td>
</tr>
<tr>
<td>10</td>
<td>.1402</td>
<td>.0559</td>
<td>.0101</td>
<td>68,758</td>
<td>421</td>
<td>1,595,896</td>
<td>1,060,670</td>
<td>3,683</td>
</tr>
</tbody>
</table>
Table 4.10. Sequence of Points Generated under Policy C

<table>
<thead>
<tr>
<th>Iteration</th>
<th>SP1</th>
<th></th>
<th></th>
<th>SP2</th>
<th></th>
<th></th>
<th>SP3</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
</tr>
<tr>
<td>0</td>
<td>.4000</td>
<td>.0100</td>
<td>.0800</td>
<td>.2000</td>
<td>.0200</td>
<td>.0200</td>
<td>.3000</td>
<td>.0800</td>
<td>.0100</td>
</tr>
<tr>
<td>1</td>
<td>.3805</td>
<td>.0266</td>
<td>.0678</td>
<td>.4211</td>
<td>.0130</td>
<td>.0130</td>
<td>.3215</td>
<td>.0829</td>
<td>.0161</td>
</tr>
<tr>
<td>2</td>
<td>.3887</td>
<td>.0161</td>
<td>.0161</td>
<td>.4347</td>
<td>.0157</td>
<td>.0114</td>
<td>.4496</td>
<td>.0829</td>
<td>.0100</td>
</tr>
<tr>
<td>3</td>
<td>.3911</td>
<td>.0195</td>
<td>.0159</td>
<td>.4464</td>
<td>.0149</td>
<td>.0104</td>
<td>.4500</td>
<td>.0829</td>
<td>.0100</td>
</tr>
<tr>
<td>4</td>
<td>.3931</td>
<td>.0212</td>
<td>.0157</td>
<td>.4489</td>
<td>.0147</td>
<td>.0101</td>
<td>.4500</td>
<td>.0829</td>
<td>.0100</td>
</tr>
</tbody>
</table>
| 5         | .3943 | .0227 | .0155 | .4497 | .0146 | .0100 |       |       |     *
| 6         | .3969 | .0245 | .0153 | .4499 | .0146 | .0100 |       |       |     *
| 7         | .4460 | .0217 | .0104 | .4499 | .0146 | .0100 |       |       |     *
| 8         | .4496 | .0213 | .0100 | .4500 | .0146 | .0100 |       |       |     *
| 9         | .4496 | .0213 | .0100 | .4500 | .0146 | .0100 |       |       |     *
| 10        | .4496 | .0213 | .0100 |       |       |     * |       |       |     *

* Termination due to best-compromise solution being reached
were added to form a single objective, as were the local flooding costs $f_4$ and $f_5$. Furthermore, the expressions reflecting pollutant loads (i.e., constraints $g_{31}$, $g_{32}$, $g_{33}$, $g_{34}$, $g_{35}$) were dropped as constraints and reformulated to function as objectives. Each pollutant type was represented by a separate objective function. Once again, the aim was to minimize the set of objectives under the local trade-off preferences of Dr. Dendrou.

Policy D's performance is given in Table 4.11. It shows, for each starting point, the path taken to the best-compromise solution.

### 4.9. Discussion

For comparison purposes, let the solution to each policy be represented by the following derived answers:

- **Policy A** - (.3277, .0659, .0115),
- **Policy B** - (.1364, .0561, .0100),
- **Policy C** - (.4500, .0829, .0100),
- **Policy D** - (.3508, .0666, .0111).

Looking at the differences in these solutions, one quickly recognizes that the maximum allowable overflow rate ($x_3$) is the least significant variable across policies. In each case the overflow rate was forced to its minimum or near-minimum level. Note also the similar results obtained under policies A and D. Evidently, the inclusion of pollutant loads as objectives (as well as using a decision maker) failed to cause any notable differences. This result is contrary to the one observed earlier between policies A and C wherein the inclusion of the decision maker did cause discrepancies between the two policies.
Table 4.11. Sequence of Points Generated under Policy D

<table>
<thead>
<tr>
<th>Iteration</th>
<th>SP1</th>
<th>SP2</th>
<th>SP3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
</tr>
<tr>
<td>0</td>
<td>.4000</td>
<td>.0100</td>
<td>.0800</td>
</tr>
<tr>
<td>1</td>
<td>.2300</td>
<td>.0320</td>
<td>.0780</td>
</tr>
<tr>
<td>2</td>
<td>.2300</td>
<td>.0484</td>
<td>.0780</td>
</tr>
<tr>
<td>3</td>
<td>.2300</td>
<td>.0837</td>
<td>.0780</td>
</tr>
<tr>
<td>4</td>
<td>.2624</td>
<td>.0673</td>
<td>.0631</td>
</tr>
<tr>
<td>5</td>
<td>.3401</td>
<td>.0910</td>
<td>.0160</td>
</tr>
<tr>
<td>6</td>
<td>.3394</td>
<td>.0696</td>
<td>.0176</td>
</tr>
<tr>
<td>7</td>
<td>.3439</td>
<td>.0669</td>
<td>.0173</td>
</tr>
<tr>
<td>8</td>
<td>.3563</td>
<td>.0713</td>
<td>.0165</td>
</tr>
<tr>
<td>9</td>
<td>.3611</td>
<td>.0672</td>
<td>.0163</td>
</tr>
<tr>
<td>10</td>
<td>.3591</td>
<td>.0656</td>
<td>.0157</td>
</tr>
</tbody>
</table>
It should be emphasized that the TCP algorithm is not confined
to just local tradeoff policies. Although the algorithm was developed
on the premise that local tradeoffs would be given by the decision
maker, this does not preclude the use of this algorithm in situations
involving global tradeoff policies, for knowledge of one's global
preference function implies an understanding of local behavior.
Moreover, in requiring these local tradeoff assessments, considerably
less is demanded of the decision maker than if he were asked to be
definitive about his global preference function.

Finally, a comment about including the human element in the
solution process. As was illustrated under policy C, inconsistencies
can arise when a decision maker is asked to supply the necessary
information to effect a solution. The inconsistency shown by the
decision maker in starting from different locations in decision
space exposes the fact that he does not always behave according to
a unique overall preference function. Reduction of some of this
erratic behavior is usually accomplished by performing tradeoff
consistency checks at the various iteration points. Testing consistency
at a single point is a standard procedure with tradeoff assessment
schemes [15, 22, 34, 43]. It involves using a different reference
criterion and the chain rule to reassess the tradeoffs. Because
there can be only r - 1 unique tradeoffs among the objectives, one
is able to measure the disagreement connected with each tradeoff.
A predetermined tolerance level is then used to ascertain whether or
not it is necessary to interact any further with the decision maker.

Experience has shown that decision makers tend to be cognizant
of only a few objectives at any one time. Those objectives which
exhibit unusual or critical values are inclined to receive the most attention. Consequently, the decision maker might be more conscious of certain objectives at one location than at another. This phenomenon is reflected in the tradeoffs the decision maker assesses between these objectives and, thus, causes his overall preference function to deviate according to the starting point.

Often, by starting the interactive solution process from different starting points, certain decision variables repeatedly seek the same level while others seem to disguise their true intentions. Under tradeoff policy C, both \( x_1 \) and \( x_2 \) sought the same levels each time, while \( x_2 \) never did. Although the human element may be the cause of such behavior, it should not be viewed as a destructive influence. By incorporating these human inconsistencies into the decision process, one is able to recognize those decision variables having rather robust solution levels. Further analysis of the problem can then be carried out with these variables fixed at their appropriate levels. In fact, having a condensed version of the original problem might even enhance the chances of better judgements on behalf of the decision maker.

This last variable reduction technique reflects the type of thinking sometimes found in the statistical analyses of a process. Being able to identify those variables which do not significantly affect the process is quite often helpful in resolving the problem. In fact, the TCP algorithm, when used in this way, might prove to be a very useful means of further refining the original problem.
CHAPTER V

DISCRETE OPTIMIZATION VIA THE
TRADEOFF CUTTING PLANE ALGORITHM

In this chapter the TCP algorithm is modified in order to solve discrete multiple objective mathematical programming problems. The direct use of the former algorithm is prohibited by the inherent nature of this type of problem. The chapter begins with a general interpretation of the discrete version of the TCP algorithm.

5.1. General Approach

The optimization method developed in this chapter, being gradient-based, works with the continuous feasible space \( X \) defined in Section 1.5. The solution, however, must be found in the set \( X_\perp \) which consists of those points in \( X \) which have integral valued components. The set \( X_\perp \) is assumed to contain at least one lattice point. \( U \) and \( f_\perp \) are still assumed to satisfy (with respect to the feasible set \( X \)) all the properties previously specified in Section 1.5. To be practical, interaction with the decision maker is only allowed at points in \( X_\perp \). Also, in an effort to simplify the presentation, the expression best-compromise set is introduced.

Definition. (Best-Compromise Set)

The best-compromise set is the collection of all best-compromise solutions. In set-builder form, it is
\[ \{ z \in U(z) \geq U(x), \quad \forall \ x \in X_i \}; \text{ and } z \in X_i \}. \]

In addressing the problem of discrete multiple objective optimization, the basic approach is slightly modified from what has previously been explained with regards to continuous solutions, but the basic concept of shrinking the original decision space by means of tradeoff cuts remains germane to the process. Instead of locating a continuous solution on each iteration, an integer solution is selected to act as the next iteration point. By establishing the tradeoff cut, the decision space is further restricted. Continuing in this manner reduces the original decision space to a set of integer points (or a single integer point) which contains the best-compromise solution.

Although the best-compromise solution is contained in this final integer point set, it is possible for this final set to contain integer points other than the best-compromise solution. At first this may appear discouraging. Upon closer examination, however, one learns that this final set is "nearly" identical to the best-compromise set.

Exactly how "close" this set is to being the best-compromise set is the subject of some later propositions. Before investigating this relationship, the TCP algorithm for discrete optimization will be discussed in more detail.

5.2. Development of the Algorithm

The TCP algorithm starts with a feasible integer solution. This point is termed the initial iteration point and is designated \( x^0 \). Upon interacting with the decision maker to determine his local functional tradeoffs, a tradeoff cut is established at \( x^0 \). In contrast to what
is done in the continuous case, these local tradeoffs play a direct role in determining the direction to improve the overall preference function $U$. Similar to the Geoffrion-Dyer-Feinberg approach, the gradient direction of $U$ is used. This means that local tradeoffs $\mathbf{i}^o$ combine with the gradients of each objective function $f_i$ to form a linear approximation of the overall preference function $U$. This approximation of $U$ (viz., $\alpha \nabla_x U(x^o)^T \cdot x$) is used as the objective function of an integer programming (IP) problem. Since the solution of an IP problem is invariant to positive multiplications of the linear objective function, using this positive multiple of $U$'s linear approximation does not affect the outcome.

The purpose of the integer programming problem is to locate the next iteration point. The linear approximation of $U$ is maximized over the integer points contained in the original feasible set intersected with the convex set defined by the translated tradeoff cut. As a result, the next iteration point is discrete and feasible. The TCP algorithm automatically designates each iteration point as a "potential point". Its classification as a potential point continues until it fails to satisfy a tradeoff cut.

**Definition. (Potential Point)**

A potential point is any integer point $\mathbf{x} \in X_+$ which has been generated by the Tradeoff Cutting Plane algorithm and which currently satisfies every tradeoff cut.

**Definition. (Potential Set)**

The collection of all potential points forms the potential set.
At each new iteration point, the IP problem is again formed. Upon solving it, local tradeoffs are again asked of the decision maker. A check is made to see which potential points (i.e., past iteration points which have remained feasible) violate the present tradeoff cut. Observe that the current iteration point will never violate its own cut, so attention is only paid to potential points other than the current one. For this example the initial iteration point, considered a potential point by definition, is now tested to see if it violates the current tradeoff cut. If it violates the cut it is no longer considered a member of the potential set and, consequently, no longer a candidate to become a best-compromise solution.

Figure 5.1 shows how iteration points are eliminated and the way in which the decision space is continually reduced in size. After receiving the decision maker’s tradeoffs at point $f(A)$, the translated tradeoff cut is superimposed on the original convex decision space to form a new, more stringent, feasible region. The linear approximation of $U$ at point $A$ is then used to find the new (integer) iteration point. This iteration point (point $B$) is labeled a potential point, meaning that it is now a candidate for the best-compromise solution. Notice that the decision maker's tradeoffs at $f(B)$ result in a translated tradeoff cut which eliminates point $A$, a former potential point. Taking the linear approximation of $U$ at $B$ and solving the associated IP problem produces the next potential point (point $C$).

Ideally, this process will continue to locate new improved potential points and eliminate past poor selections until the
Figure 5.1. Tradeoff Cut Elimination Scheme
updated feasible region contains only one integer point, that being
the best-compromise solution. Unfortunately, this cannot be guaranteed.

Taking the example one step further, Figure 5.2 illustrates the
type of difficulty that is encountered. The linear approximation
of U at B has generated point C. The tradeoff cut at \( f(C) \) has not
eliminated \( f(B) \). Consequently, the translated tradeoff cut at point C
does not eliminate point B. Using a linear approximation at C causes
the IP problem to return point B as a solution without any consideration
being given to other, possibly superior, interior points. As a result,
cycling occurs between these two potential points, preventing the best-
compromise solution from ever being realized.

To counteract this behavior, a strategy must be devised which
will exclude a point from further consideration if it has previously
been designated as a potential point. Fortunately, a technique [33]
exists. Its approach is to literally circumvent the potential point
by adding constraints. The method is particularly well suited to
this problem of cycling for it allows elimination of the potential
points without affecting the eligibility of the other feasible solutions.

In essence the idea is to represent each potential point (which
is a vector) as a unique positive integer. By forming constraints
which preclude the occurrence of this unique integer representation,
the corresponding potential point is excluded from the feasible region.

On each iteration the potential set is updated with respect to
the current tradeoff cut. Constraints are added to force all but one
potential point in this set to be infeasible to the IP problem. The
one potential point which is allowed to remain feasible is the current
iteration point. Solving the IP problem returns the next iteration
Figure 5.2. Inability to Locate the Best-Compromise Solution
point. This IP problem will always have a solution since the current (integer) iteration points remains feasible.

If the IP problem returns an integer point other than the current iteration point, this new solution becomes the next iteration point. Accordingly, it is designated a potential point and the potential set is once again updated.

On the other hand, if the current iteration point is returned, it indicates that the present solution is locally best. This means that there is no other integer point in the revised feasible region for which the linear approximation of \( U \) will give a larger value. At this stage alternate optimal solutions are investigated. If such solutions exist, the process continues; otherwise, the final potential set is complete and the process terminates.

Shortly, it will be shown that this final potential set contains the best-compromise solution. However, there is a possibility that this set might also contain points inferior to this "optimal" solution. The reason for this stems from the algorithm's inability to effectively observe these inferiorities. Knowledge of the decision maker's local tradeoffs is not always enough to insure ending with the best-compromise set.

If further distinction between the final integer points is necessary, it requires additional analysis. The decision maker needs to provide more information about his overall preferences. One means of accomplishing this is to provide the decision maker with a list of these final integer points together with their corresponding functional values. He might then be able to decide which point is best in this list. Another approach,
under a more analytical setting, is a technique called ELECTRE [6, 48]. Based on certain parameters set by the decision maker, it determines those points considered best in terms of a "concordance-discordance" criteria.

5.3. Ancillary Definitions

Although this need for further analysis may arise, it would be careless to quickly deduce from this that the final potential set is of questionable value. In fact, these final potential points satisfy certain properties which make this final set very useful and important. Moreover, one can readily recognize when the best-compromise set and this final potential set are identical. The definitions and propositions which follow will help to clarify these remarks.

**Definition.** (Preferred (under U))

A point \( x^1 \in X \) is preferred (under U) to a point \( x^2 \in X \) if \( U(x^1) > U(x^2) \).

When \( x^1 \) is preferred to \( x^2 \), it must also be preferred (under U) to \( x^2 \) since \( U \) satisfies the nonsatiation property (with respect to the \( f_i \)). The converse to this is not necessarily true, however. For instance, if \( x^1 \) and \( x^2 \) are both efficient and \( U(x^1) > U(x^2) \), then \( x^1 \) is preferred (under U) to \( x^2 \); but neither point is preferred to the other.

**Definition.** (Tradeoff Noninferior)

Any two points \( x^1 \) and \( x^2 \) in \( X \) are tradeoff noninferior to each other if \( x^1 \) satisfies the decision maker's tradeoff cut at \( x^2 \) and vice-versa.
Points B and C in Figure 5.2 illustrate two points being tradeoff noninferior to each other. The feasibility of these two points is unaffected by the tradeoff cuts at $f(B)$ and $f(C)$. One should understand that being tradeoff noninferior does not necessarily mean that the two points are efficient. The property only addresses the feasibility of these points in relationship to the two tradeoff cuts.

**Definition.** (Plausible Set)

A set $\Omega \subset X_1$ is said to be plausible provided the following two conditions are met:

1. **Internal Stability**
   - All points in $\Omega$ are efficient.
   - Any two points in $\Omega$ are tradeoff noninferior to each other.

2. **External Stability**
   - For any point $x \in X_1 - \Omega$, there exists at least one point $x^* \in \Omega$ which is preferred (under $U$) to $x$.

**Definition.** (Plausible Points)

The elements of $\Omega$ are termed plausible points.

The set $(B,C)$ in Figure 5.3 is a plausible set. Notice that point $B$ is actually preferred (under $U$) to point $C$, yet they are both members of the plausible set. This is not in violation of the definition, for elements of $\Omega$ may be preferred (under $U$) to one another. At the same time, being efficient points, no one point is preferred to any other.
Figure 5.3. The Plausible Set $\Omega = \{B, C\}$
Point A, outside of \( \Omega \), is actually preferred (under U) to point 
C, a plausible point. On the other hand, point B in \( \Omega \) is preferred
(under U) to point A. Thus, there is at least one element in \( \Omega \) which
is preferred (under U) to point A.

From this example one can understand the meaning behind a set
being "nearly" a best-compromise set. The set comprised of point B
is the best-compromise set. \( \Omega \) contains not only point B but also
point C. The tangent to U at \( f(B) \) allows for the inclusion of \( f(C) \)
and, thus, point C. No further reduction in \( \Omega \) can be accomplished
without additional information from the decision maker. Functional
tradeoffs at \( f(B) \) can only produce a linear approximation of U, and
this approximation is unable to recognize the inferiority of point C.

The reason for the interest in plausible sets is that the Tradeoff
Cutting Plane algorithm for discrete optimization terminates when only
plausible points remain in the reduced feasible region. Before proving
this fact, it is necessary to state the algorithm.

\[ \] 5.4. Discrete Version of the TCP Algorithm

The TCP algorithm for discrete multiple objective optimization
is as follows:

\[ \begin{align*}
\text{Algorithm.} \\
\text{Step 1. Start with a feasible integer solution } x^0. \\
\text{Set } k=0. \\
\text{Step 2. Evaluate the functions } f_j, \quad j=1,2,\ldots,r, \\
\text{at the current iteration point } x^k \\
\text{(i.e., } f_j(x^k), \quad j=1,2,\ldots,r). \\
\end{align*} \]
Step 3. Establish tradeoffs at \( f(x^k) = (f_1(x^k), \ldots, f_r(x^k)) \). Let \( t^k \) stand for the vector whose components are the tradeoffs established at \( f(x^k) \).

Step 4. Form the new tradeoff cut

\[
g_m(x) = \sum_{i=1}^{r} t_i^k \left[ f_i(x^k) - f_i(x) \right] \leq 0.
\]

Step 5. Update the set of potential points. That is, delete from the potential set any potential points which fail to satisfy the new tradeoff cut.

Step 6. Solve the following integer programming problem to obtain \( x^* \).

\[
\max \sum_{i=1}^{r} t_i^k \nabla_x f_i(x^k)^t \cdot x
\]

s.t. \( g_i(x) \leq 0, \ i = 1, 2, \ldots, s_k \)

\( x_j, \ j = 1, 2, \ldots, n \) integer

where \( s_k = m_0 + k + \ell_k \) and \( g_i(x) \leq 0, \ i = 1, 2, \ldots, s_k \)

represent the initial \( m \) constraints, \( k \) tradeoff cuts, and \( \ell_k \) constraints (at the \( k^{th} \) iteration) which are added to preclude all but the current potential point from occurring.
Step 7. If $x^* = x^k$, go to Step 9. Otherwise, continue.

Step 8. Designate $x^*$ as a potential point; set $x^{k+1} = x^*$, set $k = k+1$ and return to Step 2.

Step 9. Test if there is an alternate optimal solution.
   a) If there is, denote it as $x^*$ and go to Step 8.
   b) If not, stop. The current potential set is plausible.

If the original constraints and objective functions are linear, the IP problem in Step 6 of the algorithm is also linear. Solution procedures are readily available to solve this type of problem [44, 45]. On the other hand, if any one of these constraints (i.e., original constraints and tradeoff cuts) is nonlinear, the problem becomes more difficult and one's choice of a solution procedure is more limited. Possible selections in this case include several techniques discussed by Garfinkel and Nemhauser in [20], a dynamic programming approach by Cooper [11, 12], Witzgall's all-integer, parabolic programming method [60], and various cutting plane procedures.

5.5. Finiteness of the Algorithm

On each iteration, the algorithm seeks out integer points which have not already been designated as potential points. Past potential points are never considered again for either constraints are added to exclusively avoid them or they are eliminated by tradeoff cuts.
Input Initial Point $x^0$, $k \leftarrow 0$

Evaluate $f(x^k)$

Assess Tradeoffs at $f(x^k)$

Determine Tradeoff Cut and Update Potential Set

Solve Associated IP Problem ($x^*$)

$X^* = X^k$ ?

Yes

No

Designate $x^*$ as a Potential Point; $x^{k+1} \leftarrow x^*$, $k \leftarrow k+1$

Figure 5.4a. Flowchart of the TCP Algorithm (Discrete Version)
Figure 5.4b. Flowchart of the TCP Algorithm (Discrete Version) (Continued)
In the former case they are members of the potential set; in the latter case they simply no longer lie in the updated feasible region. Once the algorithm fails to locate a new potential point, it stops. Since there is only a finite number of points in $X_f$ and no point is ever considered twice, the finiteness of the algorithm is a direct result.

5.6. Results Relating to the Final Potential Set

Let $X_f$ be the final feasible set defined by the algorithm (i.e., the original feasible decision space $X_f$ intersected with each of the translated tradeoff cuts). Let $X_p$ be defined as the potential set upon termination of the algorithm.

**Lemma 5.1**

$$X_p = X_f$$

**Proof.**

Case 1: $X_p \subseteq X_f$

By construction any $x \in X_p$ must be feasible and satisfy all the accumulated tradeoff cuts. This being the case, $x \in X_f$. Hence, $X_p \subseteq X_f$.

Case 2: $X_f \subseteq X_p$

Let $y \in X_f$. In addition, assume $y \notin X_p$. Let $x^k$ be the final iteration point of the algorithm. Since $y$ is not a potential point and $x^k$ is returned at this final stage, the value of the objective function (in Step 6 of the algorithm) at $x^k$ exceeds the value at $y$. That is,
\[
\sum_{i=1}^{r} t_{i}^{k} \nabla f_{i}(x^{k}) \cdot y < \sum_{i=1}^{r} t_{i}^{k} \nabla f_{i}(x^{k}) \cdot x^{k}
\]

\[
= \alpha \nabla_{x} u(x^{k})^{t} \cdot y < \alpha \nabla_{x} u(x^{k})^{t} \cdot x^{k} \tag{5.1}
\]

Both \(x^{k}\) and \(y\) being elements of the convex set \(X_{p}\) implies that

\[
x^{k} + \lambda (y - x^{k}) \in X_{p} \quad \text{for } 0 \leq \lambda \leq 1.
\]

Therefore, \(y - x^{k}\) is a feasible direction at \(x^{k}\). This being a feasible direction implies that

\[
\nabla_{x} g_{m}^{k}(x^{k})^{t} \cdot (y - x^{k}) \leq 0
\]

where \(g_{m}^{k}(x^{k})\) is the tradeoff cut at \(x^{k}\). By substitution,

\[
[ \nabla_{x} = x^{k}\left( \sum_{i=1}^{r} t_{i}^{k} \left[ f_{i}(x^{k}) - f_{i}(x) \right] \right) ]^{t} \cdot (y - x^{k}) \leq 0
\]

\[
\Rightarrow [ \nabla_{x} = x^{k}\left( \sum_{i=1}^{r} t_{i}^{k} f_{i}(x) \right) ]^{t} \cdot (y - x^{k}) \geq 0
\]

\[
\Rightarrow \alpha \nabla_{x} u(x^{k})^{t} \cdot (y - x^{k}) \geq 0 \quad \text{(where } \alpha = 1/(\partial j/\partial x_{1}) \text{)}
\]

\[
\Rightarrow \alpha \nabla_{x} u(x^{k})^{t} \cdot y \geq \alpha \nabla_{x} u(x^{k})^{t} \cdot x^{k}
\]

But this is a contradiction to (5.1). Hence, \(y\) must be a potential point. Since \(y\) was arbitrarily selected in \(X_{p}\), \(X_{p} \subset X_{p}^{*}\).

Q.E.D.
Lemma 5.1 means that when the algorithm terminates every point in the final feasible region $X_f$ has been screened as a potential point. As a result, the final potential set and $X_f$ are actually identical.

Considering now those feasible points outside of $X_f$, Lemma 5.2 investigates their relationship to this final set.

Lemma 5.2

For any feasible integer point not in $X_f$, there exists at least one point in $X_f$ which is preferred (under $U$) to it.

Proof.

The initial feasible set $X_I$ is assumed nonempty. Furthermore, the algorithm's final iteration point is integer and feasible. Thus, there always exists at least one point in $X_f$.

Let $\chi$ be any feasible integer point which is not a member of $X_f$ (i.e., $\chi \notin X_f \cap X_p$). Since it is not in $X_f$, it must violate at least one tradeoff cut. Say $\chi$ violates the tradeoff cut at $x^k$. Then,

$$\sum_{i=1}^{r} t_i^k [f_i(x^k) - f_i(\chi)] > 0.$$  

As the result of Lemma 1.2,

$$U(\chi) < U(x^k).$$

Therefore, $x^k$ is preferred (under $U$) to $\chi$. Thus, there exists an integer point (viz., $x^k$) which is preferred (under $U$) to $\chi$. 

If \( x^k \in X_p \), then a point in \( X_p \) has been found which is preferred (under U) to \( y \).

If \( x^k \in X_I - X_p \), consider the following. Any iteration point in \( X_I - X_p \) must violate at least one tradeoff cut. Therefore, since the algorithm is finite and transitivity holds, there exists at least one iteration point in \( X_p \) which is preferred (under U) to \( x^k \in X_I - X_p \). By transitivity and the fact that \( x^k \) is preferred (under U) to \( y \), there must exist a point (viz., an iteration point) in \( X_p \) which is preferred (under U) to \( y \).

Q.E.D.

Lemma 5.2 opens the possibility of excluding some efficient points from \( X_p \). A point in \( X_p \) can never be preferred to an efficient point, but it can be preferred (under U). Thus, requiring preference according to U enables further refinement of the final set.

Lemma 5.3 establishes the type of point found in \( X_p \) and states what the relationship is between these potential points.

**Lemma 5.3**

All points in \( X_p \) are:

1. efficient and
2. tradeoff noninferior to each other.

**Proof.**

Since each potential point also serves as an iteration point, any two points in \( X_p \) must be tradeoff noninferior to each other.
A two-part proof is used to show that all points in $X_p$ are efficient. The first part shows that for any point $x$ in $X_p$ there is no $y \in X_p$ which is preferred to it. The second part shows that the same is true for any $y \in X_1 - X_p$.

Part I.

Let $f(x) = (f_1(x), f_2(x), \ldots, f_r(x))$ and, correspondingly, assume $t = (t_1, t_2, \ldots, t_r)$ to be the decision maker's tradeoff vector at $x$. Recall that $t_i > 0$ for $i=1,2,\ldots,r$. Also, let $x$ and $y$ be any two potential points in the final set $X_p$.

Since $X_p \subseteq X_1$, $x$ and $y$ are both feasible points. If $f_i(x) = f_i(y)$ for each $i=1,2,\ldots,r$, then clearly neither point is preferred to the other. Without loss of generality, assume $f_1(y) > f_1(x)$. Because $x \in X_p$, it must satisfy the tradeoff cut at $y$. That is,

$$\sum_{i=1}^{r} t_i [f_i(y) - f_i(x)] \leq 0.$$

It follows from this that

$$t_1 f_1(x) + \sum_{j=2}^{r} t_j f_j(x) \geq t_1 f_1(y) + \sum_{j=2}^{r} t_j f_j(y).$$

Since $f_1(y) > f_1(x)$,

$$t_1 f_1(y) + \sum_{j=2}^{r} t_j f_j(x) > t_1 f_1(x) + \sum_{j=2}^{r} t_j f_j(y).$$

Hence, $\sum_{j=2}^{r} t_j f_j(x) > \sum_{j=2}^{r} t_j f_j(y)$.

Since $t_1 > 0$, $\exists i, 2 \leq i \leq r$, for which

$$f_i(x) > f_i(y).$$
This implies that there is a decrease in at least one of the objective functions other than \( f_1 \). As a result, since \( x \) and \( y \) were arbitrarily selected in \( X_p \), no point in \( X_p \) is preferred to any other point in \( X_p \).

Part II.

Now let \( y \) be any integer point in \( X_p \). Furthermore, assume \( y \) is preferred to \( x \in X_p \). That is,

\[
f_i(y) \geq f_i(x) \quad \text{for } i = 1, 2, \ldots, r
\]

and

\[
f_j(y) > f_j(x) \quad \text{for some } j = 1, 2, \ldots, r.
\]

Since \( y \in X_p \), it must violate a tradeoff cut at some iteration point of the algorithm. Call this point \( w \). Since \( x \in X_p \), it must satisfy this same tradeoff cut. That is,

\[
\sum_{i=1}^{r} t_i [f_i(w) - f_i(x)] \leq 0.
\]

From the fact that \( y \) is preferred to \( x \),

\[
\sum_{i=1}^{r} t_i [f_i(x) - f_i(y)] < 0,
\]

where \( t_i \) is the \( i \)th component of the tradeoff vector at \( y \).

Combining these two sums,

\[
\sum_{i=1}^{r} t_i [f_i(x) - f_i(y)] + \sum_{i=1}^{r} t_i [f_i(w) - f_i(x)] < 0
\]

\[
= \sum_{i=1}^{r} t_i [f_i(w) - f_i(y)] < 0.
\]
This, however, contradicts the fact that $y$ violates the tradeoff cut at $w$. Therefore, $y$ cannot be preferred to $x$. Thus, there does not exist an integer point in $X'_1 - X'_p$ which is preferred to $x$.

From Parts I and II, one must conclude that $x$ is an efficient point. Since $x$ was arbitrarily selected in $X'_p$, all points in $X'_p$ are efficient.

Q.E.D.

Having established the internal and external stability of $X'_p$, the next result is an immediate consequence of Lemmas 5.1, 5.2, and 5.3.

**Theorem 5.1**

$X'_p$ is a plausible set.

As discussed earlier the best-compromise set and the plausible set are very similar. It is possible, though, for the plausible set to contain points which are preferred (under $U$) to one another. For this reason, $X'_p$ will not, in general, be identical to the best-compromise set. Nevertheless, Theorem 5.2 proves that the best-compromise set must be a subset of $X'_p$. The accompanying corollary shows when the two sets are necessarily the same.
Theorem 5.2

The best-compromise set is a subset of the plausible set $X_p$.

Proof.

Assume there is an integer point $y$ in the best-compromise set which is not a member of the plausible set $X_p$. By Lemma 5.2 there exists an integer point $x$ in $X_p$ which is preferred (under U) to $y$. But this contradicts the fact that $y$ is a best-compromise solution. Therefore, $y$ must also be a member of $X_p$. Hence, the best-compromise set is contained in the plausible set.

Q.E.D.

Corollary 5.2.1

If there is only one element in the final plausible set $X_p$, then the best-compromise set and the plausible set are identical.

Proof.

From Lemma 5.2 the unique element of $X_p$ is preferred (under U) to any other point in $X_f$. Consequently, it is the best-compromise solution. In other words, the plausible set is contained in the best-compromise set.

According to Theorem 5.2, the best-compromise set is a subset of the plausible set. Hence, the two sets are identical.

Q.E.D.
The plausible set may be dependent upon which starting point is used to initiate the algorithm. For instance, if one starts at the tradeoff point \( f(C) \) in Figure 5.5, then \( X_p = \{A, C\} \); whereas, by starting at \( f(B) \), \( X_p = \{C\} \). In either case point \( C \), the best-compromise solution, is an element of \( X_p \).

Knowledge of this dependency may be helpful if further analysis of the plausible set is requested. By initiating the algorithm from various starting points and taking the intersection of the resulting plausible sets, further refinement of this final set might be possible. Take note that this intersection is again a plausible set and still must contain the best-compromise solution.

The above argument assumes that the decision maker remains consistent with his preference function over different starting points. As observed in the application presented in Chapter IV, this may not always be the case in practice. Consequently, the intersection of the final plausible sets might turn out to be empty. This would immediately signal that the decision maker is inconsistent and should reconsider his tradeoff assessments.

5.7. Modifications

As it stands, the discrete version of the TCP algorithm might have a tendency to "zig-zag" across the decision space from one boundary to the next. By adding distance restrictions or by employing artificial constraints, one is able to improve the placement of the tradeoff cuts and, thereby, dampen this effect. This section discusses two such approaches to this problem.
Figure 5.5. Variation in the Final Potential Set
The first of these is an artificial constraint technique which uses what is called a pseudocut. The idea is to solve for $x^t = x^\star$ in Step 6 of the algorithm as usual. Then, the midpoint (or some other suitable point) of the line segment joining the current iteration point $\bar{x}^k$ and $\bar{x}^t$ is calculated. Next, a temporary cut (or pseudocut) is formed which passes through this midpoint and purposefully disconnects the decision space. By maximizing the negative of the objective function (found in Step 6) subject to the same constraints plus this cut, one is forced away from the opposite boundary and focuses the direction of the search towards the "center" of the decision space. The new integer solution becomes $x^\star$ and the process continues to Step 7. The existence of a solution is guaranteed since $x^t$ is not a member of the potential set.

An example of a pseudocut is shown in Figure 5.6. It is a displaced linear approximation of $U$ in decision space. After the point $\bar{x}^t$ is determined, the midpoint of the line segment joining $\bar{x}^k$ and $\bar{x}^t$ is calculated. The pseudocut is then centered to pass through this midpoint. It restricts the area of feasibility to that half-space containing $\bar{x}^t$. By maximizing $- \nabla_x U(\bar{x}^k)^t \cdot x$ subject to the updated feasible region and this half-space, added importance is given to those integer points located along the pseudocut. Once this IP problem is solved, the pseudocut is dropped.

Numerous variations to this technique exist. They include retaining some past pseudocuts (thereby causing further restrictions on the search region), temporarily displacing the tradeoff cut to pass through this midpoint, and/or limiting the distance one is allowed
Figure 5.6. Pseudocut Illustration
to search from $x^t$. With any of these methods caution is advised for
one must remain cognizant of the increased difficulty that may result
in trying to solve the associated IP problem.

A second and perhaps superior approach to help prevent zig-zagging
is a method which uses a hypercube to restrict movement from the center
of the feasible space. It mixes the discrete version of the TCP
algorithm (Section 5.4) with the modified method of centers technique
explained earlier (Section 3.3). It is termed a hypercube technique
and it works as follows. From the current location $x^k$, the "center"
c of the reduced feasible decision space is determined by first
solving (3.6)-(3.9) to obtain $h_x$, next choosing $\lambda^*$ to be the smallest
positive scalar such that

$$d(x^k + \lambda^* h_x) = \min \{ d(x^k + \lambda h_x) | \lambda \geq 0 \},$$

and then setting $c = x^k + \lambda^* h_x$. Although $x^k$ has integral coordinates,
the center $c$, in general, does not. Having located the center,
the smallest possible hypercube whose vertices have integral coordinates
is placed so that it circumscribes this center point. The next iteration
point is found by maximizing the linear approximation of $U$ at $x^k$ subject
to the restrictions imposed by this hypercube and the reduced feasible
region. (Recall that the original feasible region only contains
integer points.) The image (under $f$) of the optimal solution to this
subproblem is the next tradeoff point. Having located this point, a
new tradeoff cut is established through interaction with the decision
maker and the process begins to repeat itself.
If the IP subproblem is infeasible, the dimensions of the hypercube are increased by one unit in each direction. This revised IP subproblem will either be feasible, in which case the process can continue; or infeasible, in which case the dimensions of the hypercube are once again increased. Since the current point $x^k$ will eventually be included in the hypercube's expanding boundaries, this IP subproblem must at some point in time admit feasible solutions.

If $x^k$ is returned as the solution, the algorithm continues according to the original discrete version of the algorithm. The process can only terminate from this original version. The reason for this is that it is possible to have the current point returned as the optimal solution under this "nested" hypercube approach without detecting the existence of other feasible solutions in the reduced feasible region. For example, let point A in Figure 5.7 be the current location in decision space. Point C is then calculated to be the center point. The algorithm's first hypercube involves, in this case, the square STUV. Since this subproblem is infeasible, the hypercube is expanded to form the square WXYZ. The solution to this problem is point A, the current point. Therefore, having the current point returned as the optimal solution does not imply that the reduced feasible region has been entirely examined, for point B is a feasible solution and it has not been recognized. Now, by returning to the original discrete version of the algorithm, point B is located on the next iteration and the process terminates.

Attaching this hypercube routine to the original discrete version of the TCP algorithm often serves to improve the types of tradeoff cuts used to reduce the decision space. Admittedly, there are situations where the original discrete TCP version is superior to this hybrid
Figure 5.7. Hypercube Caveat
approach. Yet, as a rule, the best cut is usually located near the center of the current feasible region and this hypercube modification is more likely to yield this type of a cut.

In comparing the hypercube and pseudocut routines, it is again not clear as to which of these methods is superior. Depending upon the problem, either routine can be shown to converge in fewer iterations. However, it is the author's contention that the hypercube routine will generally produce a more advantageous tradeoff cut. The reason is again the greater likelihood it has of producing a cut near the center of the current feasible region.

In both of these modification routines (pseudocut and hypercube), the role of the decision maker is unchanged. His only interaction with the algorithm remains to be his assessments of the proportional rates of change among the objectives at various discrete iteration points. Thus, differences among these routines and the basic algorithm are not apparent to the user.

5.8. Examples

In this section the TCP algorithm for discrete optimization is applied to an example problem. Also shown are the pseudocut and nested hypercube routines. The problem used to illustrate these procedures is:

maximize $x_1$

subject to $2x_1 + 3x_2 - 12 \leq 0$

$-x_1 + 2x_2 - 4 \leq 0$

$x_1, x_2 \geq 0$ and integral valued.
The decision maker's overall preference function is assumed to behave as if

\[ U = 20r_1 + 8r_2 - r_1^2 + 2r_1r_2 - 2r_2^2. \]  

(5.2)

Consequently,

\[ \Delta U/\Delta r_1 = 20 - 2x_1 + 2x_2 \quad \text{and} \]  

\[ \Delta U/\Delta r_2 = 8 + 2x_1 - 4x_2. \]  

(5.3)

The first procedure shown is the original discrete version of the algorithm. The point \((1,2)\) is used to start the process. The tradeoff vector at this point, found by using (5.3) and (5.4), is \((1,1/11)\). From this, the new tradeoff cut is calculated to be

\[ 1(1-x_1) + (1/11)(2-x_2) \leq 0 \]

\[ \Rightarrow -11x_1 - x_2 + 13 \leq 0. \]  

(5.5)

Since only the current point is potential, no updating of the potential set is necessary. Next, the following integer programming problem is solved.

\[ \text{maximize} \quad x_1 + \frac{x_2}{11} \]  

(5.6)

subject to \(2x_1 + 3x_2 - 12 \leq 0\)  

\(-x_1 + 2x_2 - 4 \leq 0\)  

\(-11x_1 - x_2 + 13 \leq 0\)  

\[ x_1, x_2 \geq 0 \quad \text{and integral valued.} \]  

(5.7)
The solution to this problem is easily identified in Figure 5.8 to be \((6,0)\). Thus, \((6,0)\) is designated as a potential point and it becomes the next iteration point.

Once the objective functions are evaluated at this point, the decision maker again assesses his tradeoffs. His tradeoff vector according to (5.3)-(5.4) is \((1, 5/2)\). Accordingly, the second tradeoff cut is determined to be

\[
(6-x_1) + (5/2)(0-x_2) \leq 0
\]

\[
= -2x_1 - 5x_2 + 12 \leq 0.
\]

Since the first potential point \((1,2)\) also satisfies this constraint, no changes are made in the potential set. To preclude the potential point \((1,2)\) from being returned as the optimal solution to the upcoming integer programming problem, the constraints

\[
x_1 + 10x_2 - My_1 \leq 20
\]

\[
-x_1 - 10x_2 - My_2 \leq -22
\]

\[
y_1 + y_2 = 1
\]

are added where \(M\) is a large positive number and the \(y_i\) are zero-one variables. The subsequent IP problem is:

maximize \(x_1 + (5/2)x_2\)

subject to

\[
2x_1 + 3x_2 - 12 \leq 0
\]

\[
-x_1 + 2x_2 - 4 \leq 0
\]

\[
-11x_1 - x_2 + 13 \leq 0
\]

\[
-2x_1 - 5x_2 + 12 \leq 0
\]

\[
x_1 + 10x_2 - My_1 \leq 20
\]

\[
-x_1 - 10x_2 - My_2 \leq -22
\]

\[
y_1 + y_2 = 1
\]

\[
y_1, y_2 = 0,1; \quad x_1, x_2 \geq 0 \text{ and integral valued.}
\]
Figure 5.8. Tradeoff Cuts Connected with the Discrete Version of the TCP Algorithm
The optimal solution can again be seen graphically to be \((3,2)\). Its corresponding tradeoff cut translates into

\[-3x_1 - x_2 + 11 \leq 0.\]

In updating the potential set, the point \((1,2)\) must be eliminated for it fails to satisfy this latest tradeoff cut. The constraints used to circumvent this point, namely

\[
\begin{align*}
x_1 + 10x_2 - My_1 & \leq 20 \\
x_1 - 10x_2 - My_2 & \leq -22 \\
y_1 + y_2 & = 1 \\
y_1, y_2 & = 0, 1,
\end{align*}
\]

are now also discarded. The new potential set contains only \((6,0)\) and \((3,2)\).

On the next iteration the feasible point \((4,1)\) is discovered and its tradeoff cut rejects \((3,2)\) as a potential point. At this stage the algorithm terminates for there are no new feasible points remaining in the reduced feasible region (Figure 5.8). The final potential set, \(((4,1),(6,0))\), is plausible. As for it being the best-compromise set, this cannot be determined for there is more than one element in the set.

Using the decision maker's overall preference function \((5.2)\), the best-compromise solution is established to be \((6,0)\). Tradeoffs alone, however, cannot distinguish between \((4,1)\) and \((6,0)\). More information from the decision maker is required before a distinction between these two points can be made.
Had the original problem included, for example, the constraint

$$2x_1 - 9x_2 - 9 \leq 0,$$
then the final potential set would have contained only the point (4,1), since (6,0) fails to satisfy this constraint. This being the case, the final potential set and the best-compromise set would be identical, revealing (4,1) as the best-compromise solution.

Under the pseudocut modification, the sequence of points followed to the final potential set changes slightly. Starting again at (1,2), the tradeoff cut is as before in (5.5) and the same IP problem (5.6)-(5.10) is solved to establish (6,0) as the solution. Here, however, the process does not accept (6,0) as the next iteration point, but instead forms a pseudocut which dissects the line segment between (1,2) and (6,0). The linear approximation of $U$ at (1,2) is displaced to pass through the midpoint of this line segment which is at (7/2,1). The pseudocut restricts the feasible region to include that portion of the space containing (6,0) (Figure 5.9). This cut is determined to be

$$11x_1 + x_2 \geq 79/2.$$

The IP problem is now changed to include this pseudocut (which temporarily replaces the former tradeoff cut) and to maximize the negative of the previous objective function. It reads as follows:
Figure 5.9. First Iteration Pseudocut
maximize \(-x_1 - \frac{x_2}{11}\)

subject to \[2x_1 + 3x_2 - 12 \leq 0\]
\[- x_1 + 2x_2 - 4 \leq 0\]
\[-11x_1 - x_2 + 79/2 \leq 0\]

The optimal solution to this problem, \((4,0)\), becomes the next iteration point. In updating the potential set, one finds that the tradeoff cut at \((4,0)\) eliminates \((1,2)\) as a potential point. The next iteration point under this pseudocut approach turns out to be \((4,1)\). Its tradeoff cut eliminates the former potential point \((4,0)\). On the next iteration, the point \((6,0)\) is found to be feasible. The process then terminates with the final potential set again including \((4,1)\) and \((6,0)\) (Figure 5.10).

With the hypercube approach, the direction to the center of the feasible region is determined by using (3.6)-(3.9). Starting at point \((1,2)\), this direction is calculated to be \((1,0)\). Recall that the tradeoff cut at \((1,2)\) is included as a constraint in determining this direction. By minimizing the maximum distance to all the constraints, the center of this reduced feasible region is determined to be at \((2.35,2)\). The points \((2,2)\) and \((3,2)\) define the smallest "hypercube" which circumscribes this center point and whose vertices have integral coordinates (Figure 5.11).

The next iteration point, \((3,2)\), is determined by using the linear approximation of \(U\) at \((1,2)\) and maximizing it subject to the restrictions imposed by the original constraints, the accumulated
Figure 5.10. Tradeoff Cuts Connected with the Pseudocut Routine
Figure 5.11. First Iteration of the Hypercube Routine
tradeoff cuts, and the boundaries of the hypercube. The image (under \( f \)) of \((3,2)\) is the next tradeoff point. The feasible region is then reduced again by introducing the next tradeoff cut,

\[-x_1 - (1/3)x_2 + (11/3) \leq 0.\]

Solving for the next usable direction determines the direction to the center of this newly revised feasible region. This direction is established to be \((0.667, -0.833)\) and results in the new center point being located at \((4.09, 0.64)\). The points \((4,0)\), \((4,1)\), \((5,0)\) and \((5,1)\) define the boundaries of the next "hypercube". As can be seen in Figure 5.12, the point \((5,0)\) becomes the next iteration point.

The next two iterations produce the remaining two feasible points, \((4,1)\) and \((6,0)\). Since their respective tradeoff cuts eliminate all other feasible candidates, the process terminates with the final set once again containing these two points.
Figure 5.12. Second Iteration of the Hypercube Routine
CHAPTER VI

SUMMARY AND RECOMMENDATIONS

This chapter summarizes the material presented in the previous chapters and discusses its significance to the field of multiple objective decision analysis. Following this, several recommendations for further analysis are briefly noted.

6.1. Overview

As previously discussed in Chapter I, it is becoming increasingly important to develop realistic, user-oriented, mathematical techniques for the resolution of decision problems in which several objectives must be considered simultaneously. More effort is continually being devoted to the development of these solution procedures. Presently, there exist many, widely varied approaches to this type of problem. One such approach, the Geoffrion-Dyer-Feinberg algorithm, has received a great deal of attention in this area. The method answers the complex multiple objective question by allowing interaction between the computer and the decision maker during the solution process. The express purpose for this interaction is to indirectly unify the diverse and often conflicting objectives associated with the problem.

This same approach is adopted with the technique developed in this thesis, the Tradeoff Cutting Plane algorithm. However, contrary
to what is seen in the literature and the GDF algorithm in particular, this technique focuses its attention on the decision space. Through the use of "tradeoff cuts", the convex envelope containing the best-compromise solution is continually refined. The approach concentrates on reducing the feasible region in decision space, rather than improving one's knowledge of the overall preference function. In so doing, the problem is reduced to a series of pairwise decisions between the objective functions. This obviates the need for any type of vector decision on the part of the decision maker and stays reasonably within his capability to supply the necessary information to resolve this type of complex decision problem. Equally noteworthy is the fact that this approach is applicable to both continuous and discrete optimization problems, making it a practical and versatile technique to consider.

6.2. Areas for Future Research

While the Tradeoff Cutting Plane algorithm is a promising and useful approach to the multiple objective problem, there are still difficulties which need to be overcome. Foremost among these is one's tendency to blindly apply multiple objective decision analysis techniques. Hopefully, the TCP algorithm will help to bridge the gap which still exists between the skills needed to effectively employ these types of algorithms and the aptitude possessed by the practitioners who are anxious to benefit through their use. Any multiple objective optimization technique should be applied cautiously, for the success of any of these methods is limited by the competence of those who apply it.
The TCP algorithm is only viewed as another step in the quest to find a universally acceptable means of resolving multiple objective optimization problems. Still further extensions and improvements are envisioned for this algorithm. Challenging areas for future work can be divided into two general categories: (i) algorithmic improvements and (ii) general suggestions. The following list includes recommendations in both of these categories.

Algorithmic Improvements

1. Examine better ways of determining a center point for the reduced feasible region. Currently, the center is located on the basis of a distance measure which is dependent upon the magnitudes of the constraints. Freedom from this dependence would simplify the implementation of the algorithm.

2. Investigate the effect of using different "centers". Presently, the center is defined as the point along the linear program's usable direction ray which maximizes the minimum distance to all the constraints. Possibly a better center point could be obtained by more adeptly incorporating the particular features of the problem.

3. Blend the two versions of the TCP algorithm into a unified procedure which will extend the application of this method to problems involving mixed integer solutions. This would preclude the need for the existence of two separate versions of the algorithm and greatly improve its applicability to real-world decision models.
4. Improve the convergence characteristics of the algorithm in the neighborhood of the best-compromise solution. One approach to this problem is to assume or acquire more information about the overall preference function $U$. Presently, a linear approximation to $U$ is used throughout the analysis. By using a quadratic approximation to $U$, for example, the convergence rate should improve. As mentioned above in (2), using different schemes to locate the center point is another relevant consideration. Possibilities here range from devising a distance measure which does not rely on the magnitude of the constraints to running experiments to determine what "relaxation" factor best accommodates this general class of problems. Finally, an examination of the effect of using different reference criteria as one progresses towards the final solution might also prove fruitful.

5. Improve computational efficiency by devising ways of discarding former tradeoff cuts once they no longer border on the current feasible region.

6. Formulate various program termination schemes. Presently, the TCP algorithm terminates whenever there is an error, $M \geq -0.00001, |h_i| \leq 0.00001$ for all $i$, or it is instructed to do so by the decision maker.

7. Improve the type of information given the decision maker as he is progressing towards the best-compromise solution,
helping him to better understand his progress and the particular characteristics of his problem. Suggestions include determining the Euclidean distance between successive iteration points and displaying the distance to each constraint. Also associated with this area is the need to better equip the decision maker with information pertinent to his tradeoff assessments. Devising schemes to indirectly arrive at these tradeoffs (by more closely matching the decision maker's background and understanding) would prove to be very useful.

General Suggestions

1. Develop realistic, accurate and consistent means of assessing tradeoffs. An advancement in this area would result in notable improvements in not only the TCP algorithm, but also many other procedures which rely heavily on the reliability of these tradeoffs.

2. Extend the concepts introduced with the TCP algorithm to the much broader class of problems associated with decision making under uncertainty. In Chapter IV regression approximations to the STORM simulation output were used as a means of handling the stochastic nature of that problem. Yet, this does not, generally speaking, completely resolve the uncertainty question. Considerably more remains to be done in the way of transferring this uncertainty through the analysis to the final solution set.
3. Extend the applications of the TCP method to analyses involving multiple decision makers. Some progress has been made toward this end by the elimination of the difficult vector decisions which previously had to be made along the usable direction, but further advancements need to be made before multiple-member decision processes can be realistically and effectively modeled.

4. Weaken some of the more restrictive assumptions of the problem (e.g., the concavity of the $f_i$ and the convexity of the $g_j$) while maintaining conditions sufficient to ensure the convergence of the algorithm.
BIBLIOGRAPHY
BIBLIOGRAPHY


APPENDICES
APPENDIX A

AN AMALGAM OF DEFINITIONS AND LEMMAS

This appendix presents various lemmas, theorems and definitions. They are introduced here with the intention of standardizing notation and making it possible for a more lucid discussion throughout the main text. Lemmas peculiar to this thesis are proved in detail.

Definition (Convex Set)

A set $X \subset E^n$ is convex if and only if $x_1, x_2 \in X$ implies $x = \lambda x_1 + (1-\lambda)x_2$ is also in $X$ for any $\lambda$, $0 \leq \lambda \leq 1$.

Definition (Compact)

A compact set is one which is closed and bounded.

Definition (Concave)

Given a convex set $X$, a function $U$ is concave on $X$ if and only if $x_1, x_2 \in X$ implies $U(\lambda x_1 + (1-\lambda)x_2) \geq \lambda U(x_1) + (1-\lambda)U(x_2)$ where $0 \leq \lambda \leq 1$. 
Definition (Increasing)

A function $U$ is termed increasing if $y^1 < y^2$ implies $U(y^1) \leq U(y^2)$.

Lemma A.1

Let $X_i$, $i=1,2,...,m$, be convex sets. Then the set

$$X_m = \bigcap_{i=1}^{m} X_i$$

is also convex.

Lemma A.2

Let $f$ be a concave function. Then for any fixed scalar $c$, the set in $E^n$

$$F_c = \{ x \mid f(x) \geq c \}$$

is convex.

Lemma A.3

Let $f_i$, $i=1,2,...,m$, be concave functions on the convex set $X$. The set

$$F = \{ x \mid f_i(x) \geq c_i, x \in X, i=1,2,...,m \}$$

is convex.

Lemma A.4

Let $f_i$, $i=1,2,...,r$, each be concave on a convex set $X$.

If $\alpha_i \geq 0$, $i=1,2,...,r$, the function

$$g(x) = \sum_{i=1}^{r} \alpha_i f_i(x)$$

is concave on $X$. 
Lemma A.5

If $U$ is concave increasing on $Y$ and each $f^i_1$ is concave on $X$, then $U$ is concave on $X$.

Lemma A.6 [62]

Let $U$ be differentiable on $E^n$. Then $U$ is concave on $X$ if and only if $U(y) \leq U(x) + \nabla U(x)^t(y - x)$ for any $x$ and $y$ in $X$.

The next two lemmas explain some of the relationships between the decision space and the objective space. Preliminary to this, define

$$X_m = \{ x \mid x \in \bigcap_{i=0}^{m} X_i \}$$

where $X^0 \equiv X$, a convex set,

$$X_1 = \{ x \mid \sum_{k=1}^{r} \alpha_k^1 f_k(x) \geq c_1 \}$$

$$\vdots$$

$$X_m = \{ x \mid \sum_{k=1}^{r} \alpha_k^m f_k(x) \geq c_m \}$$

where $c_i$, $i=1,2,\ldots,m$, and $\alpha_k \geq 0$, $k=1,2,\ldots,r$, are scalars; and $f_k$, $k=1,2,\ldots,r$, are concave functions defined on the convex set $X$. Also define

$$F_m = \{ f \mid f \in \bigcap_{i=0}^{m} F_i \}$$

where $F^0 \equiv Y = \{ f \mid f = f(x), x \in X \}$.
\[ F^1 = \{ f \mid \sum_{k=1}^{r} \alpha_{k} f_k \geq c_1 \} \]

\[ \vdots \]

\[ F^n = \{ f \mid \sum_{k=1}^{r} \alpha_{k} f_k \geq c_n \} \].

**Lemma A.7**

\[ x \in X_m \Rightarrow f(x) \in F_m. \]

**Proof.**

\[ x \in X_m \Rightarrow x \in \bigcap_{i=0}^{m} X_i \]

\[ \Rightarrow \begin{cases} x \in X \\ \sum_{k=1}^{r} \alpha_{j} f_k(x) \geq c_j, \quad j=1,2,\ldots,m \end{cases} \]

\[ \Rightarrow \begin{cases} f(x) \in Y \\ f(x) \in F^j, \quad j=1,2,\ldots,m \end{cases} \]

\[ \Rightarrow f(x) \in \bigcap_{i=0}^{m} F_i \]

\[ \Rightarrow f(x) \in F_m. \quad \text{Q.E.D.} \]

**Lemma A.8**

If \( f^* \in F_m \), then \( \exists x \in X \) such that \( f(x) = f^* \). Furthermore, for any \( x \in X \) for which \( f(x) = f^* \), \( x \in X_m \).
Proof.

Part I.

\[ f^* \in F_m \Rightarrow f^* \in \bigcap_{i=0}^{m} F_i \Rightarrow f^* \in F^o = F \Rightarrow \exists x \in X \text{ such that } f(x) = f^*. \]

Part II.

Let \( x \in X \) and \( f(x) = f^* \in F_m \). Then

\[ \sum_{k=1}^{r} \alpha^i_k \cdot f_i(x) \geq c_j, \quad j=1,2,...,m \]

\[ \Rightarrow x \in \bigcap_{i=0}^{m} X^i \Rightarrow x \in X_m. \quad \text{Q.E.D.} \]

Lemma A.9 \[62]\]

Let \( f: V \to \mathbb{R} \) be a continuous function, and \( S \subset V \) where \( S \) is a compact set. Suppose

\[ f(w) > 0, \quad \forall w \in S \]

then \( \inf(f(w)|w \in S) > 0. \)

Lemma A.10 \[46]\]

Let \( f: \mathbb{R}^n \to \mathbb{R} \) be a continuously differentiable function, and \( S \subset \mathbb{R}^n \) where \( S \) is a compact set. Then for each \( z' \in \mathbb{R}^n \) and for each \( \delta > 0 \), there exists an \( \epsilon' > 0 \) and \( \lambda_m > 0 \) such
that for all \( z \in D (z', \varepsilon') = \{ z \mid \| z - z' \| < \varepsilon' \} \) and for all \( h \in S \),

\[
| \nabla_z f(z + \lambda h) \cdot h - \nabla_z f(z) \cdot h | \leq \delta \quad \text{for all } \lambda \in [0, \lambda_m].
\]

Theorem A.1 (Mean-Value Theorem)

Let \( f: \mathbb{R}^n \to \mathbb{R} \) be a continuously differentiable function. Then, for any \( z, h \in \mathbb{R}^n \) and for any \( \lambda \in \mathbb{R} \),

\[
f(z + \lambda h) = f(z) + \lambda \nabla_z f(z) \cdot h
\]

where \( \xi \in [z, z + \lambda h] \).

Condition. (Constraint Qualification) [51]

Let \( x^0 \) be any feasible solution to

Max \( U(x) \)

subject to \( g_i(x) \leq 0, \quad i=1,2,\ldots,m_k \);

and let \( h \) be any vector originating at \( x^0 \) which satisfies

\[
\nabla_x g_i(x)^t \cdot h \leq 0 \quad \text{for all } i \in A(x) = \{ i \mid g_i(x) = 0, \quad i=1,2,\ldots,m_k \}
\]

Then if the constraint qualification is to hold, there must exist an \( n \)-dimensional vector function \( g(t) \) of a nonnegative real variable \( t \) such that
(a) \( \varphi(0) = \bar{x} \); 

(b) \( \varphi(t) = x \) is a feasible point for all \( 0 \leq t \leq T \), 
where \( T \) is some positive number; and

(c) \( \lim_{t \to 0} \frac{\varphi(t) - \varphi(0)}{t} = h \).

In essence, this condition ensures that every direction \( h \)
satisfying \( \nabla_x g_i(x) \cdot h \leq 0 \) for all \( i \in A(x) \) is a feasible
direction.

**Lemma A.11**

Assume \( U \) to be concave increasing and differentiable on
\( C_k = \{ x \in \mathbb{R}^n | g_i(x) \leq 0, \ i=1,2,\ldots,m_k \} \) where \( g_i(x) \) are
convex differentiable functions. Let \( x = \frac{\partial U}{\partial x} \) and
\( S = \{ h \in \mathbb{R}^n | |h_i| \leq 1, \ i=1,2,\ldots,n \} \). Then

\[
\min \left( \max \left\{ -\alpha \nabla_x U(x)^t \cdot h, g_i(x) + \nabla_x g_i(x)^t \cdot h, \quad \right.\right.
\left. \left. i=1,2,\ldots,m_k \right\} \right) \geq 0
\]

\[
\Leftrightarrow \min_{h \in S} \max_{i \in A(x)} \left( -\alpha \nabla_x U(x)^t \cdot h, \nabla_x g_i(x)^t \cdot h \right) \geq 0,
\]

where \( A(x) = \{ i | g_i(x) = 0, i=1,2,\ldots,m_k \} \).

**Proof.**

**Case 1.** \( (\Rightarrow) \)

If \( \min_{h \in S} \max_{i \in A(x)} \left( -\alpha \nabla_x U(x)^t \cdot h, \nabla_x g_i(x)^t \cdot h \right) < 0, \)
then \(-\alpha \nabla_x U(x)^t \cdot h < 0\) and
\[
\nabla_x g_i(x)^t \cdot h < 0 \quad \text{for } i \in A(x).
\]

Therefore, \(g_i(x) + \nabla_x g_i(x)^t \cdot h < 0\) for \(i \in A(x)\).

But, for \(i \notin A(x)\), \(g_i(x) < 0\). Accordingly, \(\exists h \in S\) such that \(g_i(x) + \nabla_x g_i(x)^t \cdot h < 0\).

Thus,
\[
\min_{\substack{h \in S}} \left( \max_{\substack{i = 1, 2, \ldots, m_k}} \{-\alpha \nabla_x U(x)^t \cdot h, g_i(x) + \nabla_x g_i(x)^t \cdot h\} \right) < 0.
\]

The first case is established.

\underline{Case 2. (=)}

If \(\min_{\substack{h \in S}} \left( \max_{\substack{i = 1, 2, \ldots, m_k}} \{-\alpha \nabla_x U(x)^t \cdot h, g_i(x) + \nabla_x g_i(x)^t \cdot h\} \right) < 0\),

then \(-\alpha \nabla_x U(x)^t \cdot h < 0\) and \(g_i(x) + \nabla_x g_i(x)^t \cdot h < 0\), for some \(h \in S\). For \(i \in A(x)\), \(\nabla_x g_i(x)^t \cdot h < 0\).

Thus,
\[
\min_{\substack{h \in S}} \max_{i \in A(x)} \{-\alpha \nabla_x U(x)^t \cdot h, \nabla_x g_i(x)^t \cdot h\} < 0.
\]

This establishes the second case and the lemma.
APPENDIX B

METHOD OF CENTERS

In the decision space a compact convex set encloses what is being termed the best-compromise solution. The goal is to divide this space by means of a cut. Clearly, a cut near the "center" of the eligible region would be advantageous since a significant portion of the region would be eliminated each time.

Wilde [58] proposes this type of an idea in his article "Optimization by the Method of Contour Tangents". In order to find the middle of the feasible region, he suggests locating one of the following four points: the midpoint, the minimax point, the center of volume, or the centroid.

Because these points are computationally too unreasonable to locate with respect to the problem being studied, another means must be used. A technique following the lines of those known as "methods of centers" is remarkably applicable in the context of the problem under study. This appendix is devoted to describing Huard's Modified Method of Centers [29, 30] as presented in Polak [46].

B.1. Preliminary Observations

In the modified method of centers, the problem is to

\[
\text{minimize } f_0(x) \quad \text{(B.1)}
\]

subject to \( f_i(x) \leq 0, \ \ i=1,2,\ldots,m \)
where \( f_i : \mathbb{R}^m \to \mathbb{R}, \ i = 0, 1, 2, \ldots, m \), are continuously differentiable functions.

Two assumptions are made on the set

\[
C = \{ z \mid f_i(z) \leq 0, \ i = 0, 1, 2, \ldots, m \} \tag{B.2}
\]

and on the objective function \( f_0(z) \).

A1. There is a point \( z^0 \in C \) such that the set

\[
C'(z^0) = \{ z \mid f_0(z) - f_0(z^0) \leq 0, \ f_i(z) \leq 0, \ i = 1, 2, \ldots, m \} \tag{B.3}
\]

is compact and has an interior.

A2. a) The set \( C \) in (B.2) has an interior and the closure of the interior of \( C \) is equal to \( C \).

b) For every \( z \) in the interior of \( C \), \( f_i(z) < 0, \ i = 1, 2, \ldots, m \).

Since the object is to find the middle of the feasible region at each step, a distance measure is necessary.

**Definition (Distance)**

Given any point \( x \) in a convex set \( C \), the distance \( d \) of \( x \) to the boundary of \( C \) is a function which is continuous and possesses the following properties:

1) \( d = 0 \) for all points \( x \) on the boundary of \( C \),

2) \( d > 0 \) for all points \( x \) in the interior of \( C \).
Definition (Center)

Given a convex set C and a distance function d defined on this set, a point \( \bar{x} \) in C which maximizes \( d \) over the set C is called the center of the set.

Given that the current objective value is \( f_0(z') \), let \( d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \) be defined by

\[
d(z, z') = \max \{ f_0(z) - f_0(z'), f_i(z), \ i=1,2,...,m \}.
\]

(B.4)

The function \( d \) is a non-positive function for which \( -d \) satisfies the requirements of a distance function. Although the choice of a distance function is not unique [8, 29], the one defined above is fitting for the type of situation which is encountered in this report.

Note, also, that if the functions \( f_i \), \( i=0,1,2,...,m \), are all convex, then the distance function \( d \) is also convex for any \( z \in C \).

Lemma B.1

If \( \bar{z} \) is optimal to (B.1), then

\[
\min_{h \in S} \left( \max \{ \nabla_z f_0(\bar{z})^t h; \nabla_z f_i(\bar{z})^t h, \ i=1,2,...,m \} \right) = 0
\]

where \( S = \{ h | |h_i| \leq 1, \ i=1,2,...,n, \ h \in \mathbb{R}^n \} \).

The vector \( h \) is interpreted to be a direction vector. By minimizing the left-hand-side expression over all \( h \in S \), the resultant \( h \) is also a usable direction.

A more practical version of Lemma B.1 can be stated by defining \( M(\bar{z}) \), for any \( h \in S \), as
\[ M(z) = \max \left\{ \nabla_z f_0(z)^T \cdot h; \quad \nabla_z f_i(z)^T \cdot h, \quad i=1,2,\ldots,m \right\} \]  \hspace{1cm} (B.5)

Lemma B.2

If \( \hat{z} \) is optimal to (B.1), then \( M(\hat{z}) = 0 \) where \( (M(z), h) \in \mathbb{R}^{n+1} \) is characterized by the linear programming problem:

minimize \( M \)

subject to: \[-M + \nabla_z f_0(z)^T \cdot h \leq 0 \]  \hspace{1cm} (B.6)

\[-M + \nabla_z f_i(z)^T \cdot h \leq 0, \quad i=1,2,\ldots,m \]

\[-1 \leq h_j \leq 1 \quad , \quad j=1,2,\ldots,n \]

Three points should be emphasized. First, since the functions \( f_i, \ i=0,1,2,\ldots,m \), are continuously differentiable, \( M \) is the maximum of a set of continuous functions. Therefore, it must be a continuous function. Secondly, for any point \( z \in C, M(z) \leq 0 \). Finally, the vector \( h \) might not be uniquely defined by the LP problem in (B.6). Nevertheless, any optimal vector \( h \) is acceptable.

B.2. Huard's Modified Method of Centers [29]

Huard proposes the following modified method of centers which, according to Polak, is the most efficient version.

Algorithm

Step 0. Determine a point \( z^0 \in C \), and set \( i=0 \).

Step 1. Set \( z = z^i \).
Step 2. Solve (B.6) to obtain \((M(z), h)\).

Step 3. If \(M(z) = 0\), stop; otherwise, continue.

Step 4. Determine \(\mu(z)\), the smallest positive scalar such that
\[
d(z + \mu(z)h, z) = \min \{ d(z + \mu h, z) \mid \mu \geq 0 \}
\]
where \(d\) is defined as in (B.4).

Step 5. Set \(z^{i+1} = z + \mu(z)h\), \(i = i + 1\), and return to Step 1.

Figure B.1 illustrates what occurs on one iteration of Huard's algorithm. Starting from a point \(z^i\) in \(C\), the usable direction \(h\) is determined by the LP problem (B.6). The successor point, \(z^{i+1}\), is then ascertained by minimizing along \(h\) the maximum distance from all the boundaries of \(C\), and the process continues. It is this type of reasoning which is used in the Tradeoff Cutting Plane algorithm to locate the next point in decision space.

![Figure B.1. Locating \(z^{i+1}\) from \(z^i\)](image-url)
APPENDIX C

PROGRAM LISTING
PROGRAM TCP (INPUT,TERM,OUTPUT,TAPE5=INPUT,TAPE9=TERM,TAPE6=OUTPUT)
10 COMMON //TCP0//, A1, NC, NC, NAME
COMMON //TCP1//, TRADE(20,10), X(HLOD(20)), H(H20)
COMMON //TCP2//, CCL(41,121), C(121), GBAR(121)
DIMENSION X(20), XI(20), XP(20), X0LD(20), Z(20), W(10)
DIMENSION F(I10), OLF(20,10), NAME(I10), BB(10), GF(200), G(I20), GTC
100 WRITE (6,105)
20 FORMAT (1H1)
105 WRITE (6,110)
WRITE (A,110)
120 FORMAT (/20X, "INTERACTIVE MULTIPLE OBJECTIVE OPTIMIZATION",/)
WRITE (6,112)
WRITE (A,112)
125 FORMAT (5X, 45hin TEN LETTERS OR LESS, INPUT EACH OBJECTIVE, 14H)
WRITE (6,120) HOLD
WRITE (A,120) HOLD
130 FORMAT (5X, 35hin IS THE NAME ASSOCIATED WITH F(I12, 215), 215)
WRITE (6,130) I
WRITE (A,130) I
135 FORMAT (5X, A10) NAME(I)
140 CONTINUE
WRITE (6,140)
WRITE (A,140)
145 FORMAT (5X, 31hin MANY CONSTRAINTS ARE THERE)
WRITE (6,150) NC
WRITE (A,150) NC
150 FORMAT (5X, 12)
WRITE (6,155)
WRITE (A,155)
155 FORMAT (5X, 41hin MANY INDEPENDENT VARIABLES ARE THERE)
WRITE (6,160) NX
WRITE (A,160) NX
160 FORMAT (5X, 12)
WRITE (6,170)
WRITE (A,170)
165 FORMAT (5X, 4H0)
WRITE (6,175) I
WRITE (A,175) I
170 FORMAT (5X, 4H0)
CALL RR(X(I))
WRITE (6,190) X(I)
190 FORMAT (1X,F10.4)
X(I)=X(I)
195 CONTINUE
CALL GFXNS (X,G)
GO 205 KM=1,NC
GIN(K)=G(K)
IF (GIN(K)=.LE.0.1) GO TO 201
WRITE (6,190)
WRITE (8,190)
WRITE (8,190)
WRITE (6,190)
WRITE (6,190)
WRITE (6,190)
190 FORMAT (1X,0.5X,3HWHAT VALUES ARE NOT FEASIBLE?)
G3 400 YOU MUST TRY AGAIN IF SO, TYPE Y, OTHERWISE, TYPE N.
1H TO TRY AGAIN IF SO, TYPE Y, OTHERWISE, TYPE N.
CALL RC (ITRY)
WRITE (6,195) ITRY
195 FORMAT (1X,F11)
IF (ITRY.EQ.1) GO TO 165
WRITE (6,200)
WRITE (8,200)
WRITE (8,200)
WRITE (8,200)
WRITE (8,200)
WRITE (8,200)
200 FORMAT (5X,3HSTATIONARY DUE TO INFEASIBILITY.)
GO TO 955
205 CONTINUE
WRITE (6,210)
WRITE (8,210)
WRITE (8,210)
WRITE (8,210)
WRITE (8,210)
WRITE (8,210)
210 FORMAT (5X,3HYOU ARE LOCATED IN THE FEASIBLE REGION.)
WRITE (6,215)
WRITE (8,215)
WRITE (8,215)
WRITE (8,215)
WRITE (8,215)
WRITE (8,215)
215 FORMAT (5X,5HINPUT THE STEP SIZE FOR THE GOLDEN SECTION SEARCH.)
CALL RC (HR)
WRITE (6,220) HR
WRITE (8,220) HR
WRITE (8,220) HR
WRITE (8,220) HR
WRITE (8,220) HR
WRITE (8,220) HR
220 FORMAT (1X,8HRC)
WRITE (8,225)
WRITE (8,225)
WRITE (8,225)
WRITE (8,225)
WRITE (8,225)
WRITE (8,225)
225 FORMAT (5X,4HINPUT THE INITIAL ERROR TOLERANCE FOR THE GOLDEN
SECTION SEARCH.)
CALL RC (ERR)
WRITE (6,230) ERR
WRITE (8,230) ERR
WRITE (8,230) ERR
WRITE (8,230) ERR
WRITE (8,230) ERR
WRITE (8,230) ERR
230 FORMAT (1X,6HRC)
WRITE (8,235)
WRITE (8,235)
WRITE (8,235)
WRITE (8,235)
WRITE (8,235)
WRITE (8,235)
235 FORMAT (5X,4HCC YOU WISH TO USE THE GRADIENT APPROXIMATION?,
9HOPTIONS 1/5X, 3HIF SO, TYPE Y, OTHERWISE, TYPE N.)
1H TO USE THE GRADIENT APPROXIMATION ?
CALL RC (ITGRAD)
WRITE (6,240) ITGRAD
WRITE (8,240) ITGRAD
WRITE (8,240) ITGRAD
WRITE (8,240) ITGRAD
WRITE (8,240) ITGRAD
WRITE (8,240) ITGRAD
240 FORMAT (1X,8HRC)
WRITE (8,245)
WRITE (8,245)
WRITE (8,245)
WRITE (8,245)
WRITE (8,245)
WRITE (8,245)
245 FORMAT (5X,4HINPUT THE CENTRAL DIFFERENCE PARAMETER FOR CALCULATING
GRADIENTS.)
CALL RC (OIF)
WRITE (6,250) OIF
WRITE (8,250) OIF
WRITE (8,250) OIF
WRITE (8,250) OIF
WRITE (8,250) OIF
WRITE (8,250) OIF
250 FORMAT (1X,18HRC)
WRITE (6,255)
WRITE (6,255)
WRITE (6,255)
WRITE (6,255)
WRITE (6,255)
WRITE (6,255)
255 FORMAT (1X,6HRC)
WRITE (6,260)
WRITE (6,260)
WRITE (6,260)
WRITE (6,260)
WRITE (6,260)
WRITE (6,260)
260 FORMAT (5X,3HCC YOU WISH TO USE THE TRADEOFF SCHEMES, 1/5X, 3HIF SO,
15HOPTIONS 1/5X, 3HIF SO, TYPE Y, OTHERWISE, TYPE N.)
CALL PC (ITRAD)
WRITE (6,265) ITRAD
WRITE (6,265) ITRAD
WRITE (6,265) ITRAD
WRITE (6,265) ITRAD
WRITE (6,265) ITRAD
WRITE (6,265) ITRAD
265 FORMAT (1X,8HRC)
C
C
THIS SECTION PRESENTS THE DECISION MAKER WITH ALTERNATIVES AND READS HIS TRADEOFF INPUTS.

MACC=0
M(1)=1
IC=1
L=1
T=1
CALL FFXMS (X,F)
DC 270 I=1,4X
DC 270 >OLD(I)=X(I)
WRITE (6,275)
WRITE (6,275)

275 FORMAT (5X, 42H THE INITIAL OBJECTIVE FUNCTION VALUES ARE [I])
CC 275 I=1,4DF
WRITE (6,280) NAME(1),I,F(I)
WRITE (6,280) NAME(2),I,F(I)

280 FORMAT (1X, A13, 10X, 2HF(1), 1H1, 5X, F20.5)
OLDF(1,1)=F(1)

285 CONTINUE
290 WRITE (6,295)
WRITE (6,295)

295 FORMAT (5X, 27H INPUT PERTURBATION OF F(I))
CC 295 F(IC,E)
WRITE (6,300) FFONE

300 FORMAT (1X, F15.7)
BE(I)=F(1)+FFONE
DO 305 JOB=2,4NF
BR(JE)=F(JP)

305 CONTINUE
310 FORMAT (1X, F15.7)
BE(1)=F(1)+FFONE
DO 315 JOB=2,4NF
BR(JE)=F(JP)

315 FORMAT (1X, F15.7)
BE(1)=F(1)+FFONE
BR(JE)=F(JP)

320 FORMAT (1X, F15.7)
IF (JT,EC,2) CC TO 320
BE(JT)=F(JT)

325 FORMAT (1X, F15.7)
IF (JTAD,EC,1HN) GO TO 370
WRITE (6,330)
WRITE (6,330)

330 FORMAT (30X, A10, 24X, 1HB)
DO 340 NK=1,NF
WRITE (6,335) NAME(NK),F(NK),DB(NK)
WRITE (6,335) NAME(NK),F(NK),DB(NK)

335 FORMAT (1X, A10, 5X, F20.5, 5X, F20.5)

340 CONTINUE

THIS SECTION ADJUSTS THE ALTERNATIVES PRESENTED TO THE DECISION MAKER UNTIL HE IS INDIFFERENT.

WRITE (6,345)
WRITE (6,345)

345 FORMAT (1X, A10, 5X, A10) DO YOU PREFERENCES IF YOU ARE INDIFFERENT, TYPE TCP 1420
CALL RC (IPREF)
WRITE (6,350) IPREF

350 FORMAT (1X,A1)
   IF (IPREF.EQ.'A') GO TO 370
   IF (IPREF.EQ.'B') GO TO 325

C
C PREFER A
C
355 DELTA=DELTA/2.
   BB(KT)=BE(KT)+DELTA
C
INPUT PREFERENCE
C
360 WRITE (6,330)
WRITE (6,330)
DO 365 JH=1,ACF
   WRITE (6,335) NAME(JH),F(JH),BB(JH)
   WRITE (6,335) NAME(JH),F(JH),BE(JH)
365 CONTINUE
WRITE (6,345)
WRITE (6,345)
WRITE (6,350) IPREF
   IF (IPREF.EQ.'A') GO TO 370
   IF (IPREF.EQ.'B') GO TO 355

C
C PREFER B
C
365 DELTA=DELTA/2.
   BB(KT)=BE(KT)+DELTA
   GO TO 360
C
THIS SECTION COMPUTES THE TRADEOFF VALUES.
C
370 W(KT)=IF(KT)+EP(KT)/Pfone
375 CONTINUE
NACG=NACG+1
   IF (ITRADE.EQ.1) GO TO 385
   WRITE (6,390)
380 FORMAT (5X,1AH1E TRADEOFFS ARE1)
385 DC 400 LT=1,NCF
   IF (ITRADE.EQ.1) GO TO 385
   WRITE (6,390) NAME(LT),W(LT)
   WRITE (6,390) NAME(LT),W(LT)
390 FORMAT (1X,A10,S1,F10.5)
395 TRADE(WACC,LT1,W(LT))
400 CONTINUE
C
THIS SECTION CHECKS TO SEE IF THE DECISION MAKER IS IN THE
PROMISING REGION. IF REQUESTED, IT ALSO APPROXIMATES THE
GRADIENT OF EACH OBJECTIVE FUNCTION USING A CENTRAL DIFFERENCING SCHEME.

CALL HCLDC (NACC,F)
IF (ISGRAD.EQ.1) GO TO 405
CALL FGRAD (X,F)
GO TO 430
405 GO TO 410 T=1, AX
410 XP(I)=X(I)
DO 425 J=1, NX
XP(J)=X(J)+0.1DF
CALL PPXKS (X,F)
DO 415 I=1, NCF
IK=NX*I+NX*J
415 GF(IK)=F(I)
XP(J)=X(J)+0.1DF
CALL PPXKS (X,F)
DO 420 I=1, NCF
IK=NX*I+NX*J
420 GF(IK)+=(GF(IP)-F(IP))/12.*0.1DF
XP(J)=X(J)
425 CONTINUE
CALL PPXKS (X,F)
430 IF (IC.EQ.1) GO TO 510

TEST IF THERE IS GUARANTEED IMPROVEMENT.

R=.3
DO 440 K=1, NX
ROC=0.
DO 435 I=1, NCF
IK=NX*I+I*K
435 ROC=ROC+H(I)*GP(INX)
440 R=R+ROC*(I-1)*7(K)
IF (R.LT.6.) GO TO 490

TEST IF PREFERRED DECISION POINT SATISFIES THIS NEW CUT.
IF NOT, THIS NEW POINT IS A PREFERRED POINT.

CALL PPXKS (XOLC,F)
GCX=0.0
DO 445 I=1, NCF
GCX=GCX+H(I)*F(I)
445 CONTINUE
GCX=GCX-HOLC(NACC)
IF (GCX.GT.0.0) GO TO 445

THE CUT IS RETAINED, BUT THE STEP SIZE IS REDUCED.

WRITE (6,450)
WRITE (6,450)
450 FORMAT (/5X, 36F15.0, 36F15.0)
1 1/2 THE DISTANCE IS NOW ATTEMPTED.)
T=T/2.
WRITE (6,455)
WRITE (6,455)

456 FORMAT (/5X, 36X, 'THE NEW ONE=HALF STEP SIZE POINT IS')
DO 465 K=1, NX
   X(K)=X0+L*K
WRITE (6,460) X(K)
WRITE (8,460) K, X(K)
465 CONTINUE

467 FORMAT (/5X, 2H4, 62, 4H=, F10.4)

470 CONTINUE
CALL FFXMS (X, F)
WRITE (6,470)
WRITE (8,470)

475 CONTINUE
WRITE (6,475) NAME(I), F(I)
WRITE (8,475) NAME(I), F(I)

475 FORMAT (/5X, 2H4, 10X, 2H4, 10X, 1H1, 5X, F20.5)
OLDIF(NACC=1, I)=F(I)

480 CONTINUE
GC TO 290

C

THE DECISION MAKER IS AT A PREFERRED POINT.

C

490 CALL FFXMS (X, F)
490 WRITE (6,495)
WRITE (8,495)

495 FORMAT (/5X, 36H-YOUR NEW POINT IS A PREFERRED POINT)

C

ASK WHETHER TO TERMINATE THE PROCESS OR NOT.

C

WRITE (6,500)
WRITE (8,500)

500 FORMAT (/5X, 33H-IF YOU WISH TO TERMINATE, TYPE T, 20H OTHERWISE.
1 TYPE C)
CALL RC (ITC)
WRITE (6,505) ITC

505 FORMAT (/5X, 1H1)
IF (ITC.EQ.1) GO TO 720

C

THIS SECTION INPUTS THE COEFFICIENTS OF THE SUBOPTIMIZATION
LINEAR PROGRAMMING PROBLEM WHICH REMAIN FIXED ON EACH ITERATION
OF THE TRADEOFF CUTTING PLANE ALGORITHM.

C

510 DO 525 IR=1, NX
   DO 520 ICOL=1, NX
      IF (IR.EQ. ICOL) GC TO 515
      COL(IR, ICOL)=0.
      AK=NX+ICOL
      COL(IR, AK)=0.
      NI=AK+IR
      COL(NI, ICOL)=0.
      COL(NI, AK)=0.
   GO TO 520

C

TCP 3050
TCP 3060
TCP 3070
TCP 3080
TCP 3090
TCP 3100
TCP 3110
TCP 3120
TCP 3130
TCP 3140
TCP 3150
TCP 3160
TCP 3170
TCP 3180
TCP 3190
TCP 3200
TCP 3210
TCP 3220
TCP 3230
TCP 3240
TCP 3250
TCP 3260
TCP 3270
TCP 3280
TCP 3290
TCP 3300
TCP 3310
TCP 3320
TCP 3330
TCP 3340
TCP 3350
TCP 3360
TCP 3370
TCP 3380
TCP 3390
TCP 3400
TCP 3410
TCP 3420
TCP 3430
TCP 3440
TCP 3450
TCP 3460
TCP 3470
TCP 3480
TCP 3490
TCP 3500
TCP 3510
TCP 3520
TCP 3530
TCP 3540
TCP 3550
TCP 3560
TCP 3570
TCP 3580
TCP 3590
TCP 3600
TCP 3610
TCP 3620
TCP 3630
TCP 3640
TCP 3650
515 \text{COL}(i,k,ICOL) = 1,0
\text{nx} = \text{nx} + \text{icol}
\text{COL}(i,nx) = 1,0
\text{ai} = \text{nx} + \text{ir}
\text{COL}(i,k,ICOL) = 1,0
\text{COL}(i,nx) = -1,0

520 \text{CONTINUE}

525 \text{CONTINUE}
\text{nx} = 2 \times \text{nx}
\text{go} 530 \text{i} = 1, n \times \text{nx}

530 \text{COL}(2 \times \text{nx} + 1, \text{i}) = 0
\text{cc} 535 \text{i} = 1, n \times \text{nx}
\text{c(i)} = -1,0

535 \text{CGAR}(-1,0)
\text{c}
\text{c}
\text{this section determines the coefficients of the suboptimization}
\text{linear programming problem which change on each iteration of the}
\text{tradeoff cutting plane algorithm. included in this section is}
\text{also a central differencing scheme to approximate, if so}
\text{requested, the gradient of each constraint. furthermore, this}
\text{section checks if the original constraints extend their previous}
\text{pin or max values.}
\text{c}
\text{c}

\text{if} \text{igrad} < \text{ec} < \text{1,hy} \text{go to 550}
\text{call ggmao(x,gg)}
\text{go} 545 \text{j} = 1, n \times \text{nx}
\text{jj} = n \times \text{nx} + \text{j}
\text{do} 540 \text{i} = 1, n \times \text{nc}
\text{ir} = 2 \times \text{nx} + \text{i}
\text{ig} = n \times \text{nx} \times (i r - 2 \times \text{nx}) - n \times \text{+j}
\text{col(j,j,ir)} = \text{gg}(i g)
\text{col(j,j,ir)} = \text{gg}(i g)

540 \text{CONTINUE}

545 \text{CONTINUE}
\text{go to 575}

550 \text{do} 555 \text{i} = 1, n \times \text{nx}

555 \text{xp}(i) = x(i)
\text{go} 570 \text{j} = 1, n \times \text{nx}
\text{jj} = n \times \text{nx} + \text{j}
\text{xp(j,j) = x(j) + dif}
\text{call gfxxs(xj,g)}
\text{do} 560 \text{i} = 1, n \times \text{nc}
\text{ir} = 2 \times \text{nx} + \text{i}

560 \text{col(j,j,ir)} = 1(1)
\text{xp(j,j) = x(j) + dif}
\text{call gfxxs(xj,g)}
\text{do} 565 \text{t} = 1, n \times \text{nc}
\text{ip} = 2 \times \text{nx} + \text{i}
\text{ccl(j,i,ir)} = (\text{col(j,i,ir)} - g(i)) / (2 \times \text{dif})
\text{col(j,j,ir)} = -\text{ccl(j,i,ir)}
\text{xp(j) = x(j)}

570 \text{CONTINUE}

575 \text{call gfxxs(xj,g)}
\text{go} 595 \text{i} = 1, n \times \text{nc}
\text{ix} = 2 \times \text{nx} + \text{i}
\text{if} \(-g(i) + \text{le} = \text{gmax}(i)) \text{go to 590}
\text{gmax}(i) = -g(i)

580 \text{if} \(-g(i) + \text{le} = \text{gmin}(i)) \text{go to 585}
\text{gmin}(i) = -g(i)
585  COL(2*NX+1,IX)=1.0
    C(IK)=G(I)
    CBnF(IK)=G(I)
    IF (C(IK),LE,0.0) GO TO 595
    WRITE (6,590) IK
    WRITE (3,590) IK

590  FORMAT (/1X,'I,J3, 4th ELEMENT OF THE LP COST VECTOR IS POSITIVE.*/
     1X, 22HAWAIIAN TERMINATION,)
    MARK=1
    GO TO 725

595  CONTINUE

C
C THIS SECTION FILLS THE COEFFICIENT MATRIX FOR THE NEWLY ADDED
C CONSTRAINTS (E.G., TRADEOFF CUTS).  NOTE:  THE LAST CONSTRAINT
C IS THE SAME AS THE USEABLE DIRECTION CONSTRAINT.  THIS SECTION
C ALSO CHECKS TO SEE IF A TRADEOFF CUT EXCEEDS ITS PREVIOUS MAX
C VALUE.
C
IK=NC+(2*NX)+1
IJ=NC+(2*NX)+NACC
CO 620  I=IK,IJ
IL=I-NC-(2*NX)
DO 605  J=1,NX
       JJ=NX+J
       COL(J,J)=G(J)
       GO 600  IF=1,NCF
       IF=NX*IF-NX+J

600  COL(J,J)=COL(J,J)*THRD(IL,IF)*GF(IF)
605  COL(J,J)=COL(J,J)*GF(IF)
    COL(2*NX+1,J)=-1.0
    C(J)=G(J)
    QD 610  J=1,NCF
    DO 610  I=1,NX

610  C(I)=C(I)-TRACF(IL,J)*F(J)
    C(I)=G(I)-HOLD(IL)
    CMDF(I)=C(I)
    IF (-C(I),GE,TMAX(IL)) TMAX(IL)=C(I)
    IF (-C(I),LT,TMAX(IL)) GO TO 620
    WRITE (6,615) I
    WRITE (3,615) I

615  FORMAT (/1X, 4th ELEMENT OF THE LP COST VECTOR IS POSITIVE.*/
     1X, 22HAWAIIAN TERMINATION,)
    MARK=1
    GO TO 725

620  CONTINUE

C
C THIS SECTION DETERMINES THE USEABLE DIRECTION K VIA A LINEAR
C PROGRAMMING SUBROUTINE.
C
CALL UDFALU (RM,IFLAG)

IF (IFLAG) 625 645 635
625  WRITE (6,645)

630  FORMAT (/1X, 3#ERROR IN SUBROUTINE UDFALU UDDH INPUT DATA.,1/1X, 29HCFP
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HAWAIIAN , 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
     1(J) NEED TO BE NON-POSITIVE.,1/10X, 10HTERM.
WRITE (A,640)

640 FORMAT (/10X, 42HMINIMUM RATIO RULE FAILED IN DUAL SIMPLEX, 8H080TCP 4880
15LINE, /10X, 22HAUTOMATIC TERMINATION, )

MARK=1

GO TO 725

C

CHECK WHETHER R'M OR H ARE NEAR ZERO.

C

645 IF (R'M,LT,-0.00001) GO TO 650

650 GC 655 IH=1,HX

IF (ABS(I'H),GT,0.00001) GO TO 660

IF (I'H,EQ,0,HX) GO TO 710

655 CONTINUE

C

FINDS THE CENTER OF THE DECISION SPACE.

C

660 CALL GSS (RKC, ERR, X, DIST, RLA MDA)

ERRN=ERR

IF (DIST,LE,ERROR) GO TO 665

ERRN=ERR/2.

GO TO 660

C

THIS SECTION LOCATES THE NEW POINT IN DECISION SPACE.

C

AND EVALUATES IT IN OBJECTIVE SPACE.

C

665 T=1.

GO 670 K=1,NX

Z(K)=R RLA MDA*(K)

670 XCLD(K)=X(K)

WRITE (6,675) IC

WRITE (9,675) IC

675 FORMAT (/15X, 4AH***********/5X, 7HMA THE ,12, 24H ITERATION, THE DECISION, 1TCP 5279

20H POINT IS I/I

GO 680 K=1,NX

X(K)=XCLD(K)+Z(K)

WRITE (6,680) K, X(K)

WRITE (9,680) K, X(K)

680 FORMAT (15X, 2H1X,412, 4H) = ,2F10,4)

685 CONTINUE

CALL FFXNS (X,F).

WRITE (6,690) IC

WRITE (9,690) IC

690 FORMAT (/15X, 7HMA THE ,12, 35H ITERATION, THE OBJECTIVE FUNCTION

1, 11VALUES ARE11

GO 700 I=1,6OF

WRITE (6,695) NAME(I), I,F(I)

WRITE (9,695) NAME(I), I,F(I)

695 FORMAT (1X,412,1X, 2MF(I,1H, ,1H),5X,F20,5)

ODYF(NACC+1,1)+F(I)

700 CONTINUE

WRITE (6,705)

WRITE (9,705)

705 FORMAT (/1X, 47H***********, 1TCP 5400)
**Termination in a Finite Number of Steps**

**710** WRITE (6,715)
**WRITE** (6,715)

**715** FORMAT (4DX, 43X, AUTOMATIC TERMINATION, THE BEST-COMPROMISE, 10H, 1H, HAS BEEN REACHED.)
**MARK** = 2
**GO TO** 725

**720** **MARK** = 3

**725** WRITE (6,730)

**730** FORMAT (1H1)
**WRITE** (6,735)

**735** FORMAT (14X, 2M10.10)** SUMMARY** **RTP** **TCP** **5400**
**1** 11 **INPUT** **TCP** **5500**
**CC TO** (740, 570, 570, 570, 570)

**740** WRITE (6,745)

**745** FORMAT (5X, 46HTERMINATION DUE TO MINIMUM RATIO RULE FAILURE.)
**GO TO** 740

**750** **ITER** = IC - 1

**755** WRITE (6,755) **IC**

**760** WRITE (6,765)

**765** FORMAT (5X, 46HTERMINATION DUE TO ERROR IN UCOLUMN INPUT DATA.)
**GO TO** 740

**770** **IC** = IC - 1

**775** FORMAT (5X, 49HTERMINATION DUE TO BEST-COMPROMISE BEING REACHED.)
**1** 6 **ITER** = 12, 12 **ITER** = 12

**780** WRITE (6,785)

**785** FORMAT (5X, 40HTHE INITIAL POINT IN DECISION SPACE WAS)
**CC** 790 **IC** = 1, **IC**

**790** WRITE (6,790) **IC**

**795** CONTINUE

**800** FORMAT (5X, 43HTHE INITIAL OBJECTIVE FUNCTION VALUES WERE)
**CC** 810 **IC** = 1, **IC**

**810** WRITE (6,815)

**815** FORMAT (15X, 37HTHE FINAL POINT IN DECISION SPACE IS)
**CC** 820 **IC** = 1, **IC**

**825** CONTINUE

**830** WRITE (6,830)

**835** FORMAT (15X, 40HTHE FINAL OBJECTIVE FUNCTION VALUES ARE)
**CC** 840 **IC** = 1, **IC**
DO 840 I1=1,NCF
   WRITE (6,A35)  NAME(I1),I1,F(I1)
835 FORMAT (1X,A18,5X,2HF(1,1I1,1H),5X,F20.5)
840 CONTINUE
   WRITE (6,A45)
845 FORMAT (1/1X,17HCONSTRAINT VALUES/,5X,3HMIN.,6X,5HMAX.
   1HINITIAL,6X,3HMIN.+10X,3HMAX)
CALL GFX35 (X+G)
   FDIST=G(I)
   ICON=I
850 CONTINUE
   WRITE (6,A55) I+HG,GIN(I),GMIN(I),GMAX(I)
855 FORMAT (5X,12,5X,F8.3,5X,F8.3,5X,F8.3,5X,F8.3,5X,F8.3)
860 CONTINUE
   WRITE (6,A65)
865 FORMAT (/13X,16HTRADEOFF CLT VALUES/,5X,3HMIN.,4X,7HCU.
   1HNO.,6X,TCP 6330
   AACC=MACC=1
   EC 885 K=1,NACC
   KK=NC*K
   TCV=0.
   IL=NC-3*X K)
   DO 870 J=1,NCF
870 TCV=TCV+TCV(1)+0(1)
   IF (TCV.GT.TCMAK(II) TCMAK(II)=TCV
   IF (TCV.GT.PDIST) GO TO 875
   PDIST=TCV
   ICON=KK
875 WRITE (6,A95) KK,K,TCV,TCMAK(K)
880 FORMAT (5X,12,7X,12,7X,5X,F8.3,5X,F8.3)
885 CONTINUE
   WRITE (6,A100) FDIST,ICON
890 FORMAT (/18X,3H1ST LOWEST CONSTRAINT VALUE IS /F8.3/5X,25HTHE TCP
   10 CONSTRAINT IS NUMBER /I2, 1H,
   WRITE (6,A95)
895 FORMAT (/19X,44HTRADEOFF SENSITIVITY BASED ON FINAL SOLUTION
   DO 950 I1=1,NACC
   WRITE (6,A90) I1,CLOF(I1)
900 FORMAT (/13X,15HTRADEOFF POINT /I2, 11H ( F(1), /I10.3, 2H/,TCP
   15X, /I10.3, 10X, 15HMIN.2, 14X, 15GIVEN, 14X, 15HALLOW (SUC PTS), 18TCP
   2X, 13HALLOW (EC FT)/5X, 3HMIN., 13X, 3HMAX., 13X, 3HMIN., 13X, 3HMAX.
   3A)
   DO 945 J=2,NCF
   SMIN=0.0
   SMAX=1.0
   IK=I
   DO 920 II=1,NACC
   OF=CLCF(II,J)-CLCF(I,J)
   IF (AES(OF),LE.,0.06G001) GO TO 920
   SC=0.0
   DO 905 K=1,NCF
   TCP 6100
   TCP 6110
   TCP 6120
   TCP 6130
   TCP 6140
   TCP 6150
   TCP 6160
   TCP 6170
   TCP 6180
   TCP 6190
   TCP 6200
   TCP 6210
   TCP 6220
   TCP 6230
   TCP 6240
   TCP 6250
   TCP 6260
   TCP 6270
   TCP 6280
   TCP 6290
   TCP 6300
   TCP 6310
   TCP 6320
   TCP 6330
   TCP 6340
   TCP 6350
   TCP 6360
   TCP 6370
   TCP 6380
   TCP 6390
   TCP 6400
   TCP 6410
   TCP 6420
   TCP 6430
   TCP 6440
   TCP 6450
   TCP 6460
   TCP 6470
   TCP 6480
   TCP 6490
   TCP 6500
   TCP 6510
   TCP 6520
   TCP 6530
   TCP 6540
   TCP 6550
   TCP 6560
   TCP 6570
   TCP 6580
   TCP 6590
   TCP 6600
   TCP 6610
   TCP 6620
   TCP 6630
   TCP 6640
   TCP 6650
   TCP 6660
   TCP 6670
   TCP 6680
   TCP 6690
   TCP 6700
IF (K>M,K,M) GO TO 905
SC = 5 + (1 - (1 - SC)) * (CLDF(I, k) - OLDF(I, k))
CONTINUE
GO TO 920
IF (SFG, SFG, 0.0) GO TO 915
IF (SFG, SFG, 0.0) GO TO 910
SMAX = -9,
SMAX = -9,
GO TO 920
IF (SFG, SFG, 0.0) GO TO 920
SMAX = -9,
TSMAX = 1.TMIN
IF (TSMAX, TSMAX, TSMAX) TSMAX = TSMAX
CONTINUE
GO TO 920
EMIN = 0.0
EMIN = 0.0
DFO = F(I) - CLDF(I, J)
IF (AENSDFC, 0.0, 0.0) GO TO 935
EMIN = 1./TMAX
EMIN = 1./TMAX
GO TO 935
IF (SFG, SFG, 0.0) GO TO 935
TMAX = SC/I-CDF
EMIN = 1./TMAX
EMIN = 1./TMAX
GO TO 935
CONTINUE
DART = 1./TRADE(I, J)
WRITE (6, 940) J, OLDF(I, J), DART, SMIN, SMAX, BMIN, BMAX
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
CONTINUE
STOP
END
SUBROUTINE UNDUAL (RM, IFLAG)
COMMON /TCOM1/ AX, KOF, NC, NACC
COMMON /TCOP2/ TRADE(25, 101), M(20, 4)
COMMON /TCOMS/ CI(41), E11, E11, CI(121), CI(121)
COMMON /TCOMS/ CI(41), E11, E11, AX(41), NK+4
M = 2*NK+1
N = 4*NK+NG+NACC+1
L = 2*NK+NG+NACC+1
LL = 2*NK+NG+NACC
EPS=.5E-6
IT=1
IFLAG=0

C
C  INITIALIZE RHS
C

MM=M-1
DC 105 I=1,MM
105 RHS(I)=0.
RHS(M)=1.0
C
C  TEST WHETHER THE COST COEFFICIENTS HAVE THE NECESSARY
C  NON-POSITIVE FORM. IF THERE IS A VIOLATION, INDICATE
C  AN ERROR.
C
C  DO 110 J=M,LL
    IF (C(J),GT,0.0) GO TO 115
110 CONTINUE
    GO TO 120
115 IFLAG=-1
    GO TO 220
C
C  THIS SECTION FILLS IN THE IDENTITY MATRIX AND COSTS ASSOCIATED
C  WITH THE BASIC VARIABLES.
C
120 DC 130 I=1,M
    II=2*NX+N+NC+NNC+I
    GO 125 J=L,N
125 COL(I,J)=0.0
    CI(I)=1.0
    CNAP(I)=0.0
    CI(I)=0.0
    IVAP(I)=II
130 CLI(I,II)=1.0

C
C  THIS SECTION LOCATES THE PIVOT ROW IR.
C
135 IR=0
    DC 140 I=1,M
    IF (RHS(I),GE,EPS) GO TO 140
    IR=I
    GO TO 145
140 CONTINUE
    IF (IR,EQ,0.0) GO TO 195
145 IT=IT+1
C
C  THIS SECTION LOCATES THE PIVCT COLUMN IS.
C
IS=0
    THETA=.9E30
    DO 150 J=1,N
IF (COL(I, J) .GE. EPS) GO TO 150
RATIO = CBAR(I,J) / COL(I, J)
IF (RATIO .GT. THETA) GO TO 150
THETA = RATIO
IS = J
150 CONTINUE
IF (IS .EQ. 0) GO TO 190
C C
C UPDATE RHS
C C
OC 155 I = 1, M
IF (I .EQ. IR) GC TO 155
RHS(I) = RHS(I) * (COL(I, IS) / COL(IR, IS)) * RHS(IR)
C C
C CHECK FOR ROUND-OFF ERROR CLOSE TO ZERO AND CORRECT IF NECESSARY.
C C
IF (ABS(RHS(I) - (1.0 .EQ. 0)) .LT. 1.0E-10) RHS(I) = 0.
155 CONTINUE
C C
C UPDATE RHS(IR)
C C
RHS(IR) = RHS(IR) / COL(IR, IS)
C C
C UPDATE CBAR RCH FOR THE NEXT ITERATION.
C C
160 CONTINUE
C C
C THE new COL(I,J) MATRIX IS CALCULATED.
C C
170 CONTINUE
OD 165 I = 1, M
IF (I .EQ. IR) GC TO 165
IF (J .EQ. IS) GC TO 170
COL(I, J) = COL(I, J) + COL(IS) / COL(IR, IS) * COL(IR, J)
IF (ABS(COL(I, J) - LT .LT. 1.0E-14) .OR. COL(I, J) > L)
165 CONTINUE
170 CONTINUE
OC 190 J = 1, N
175 CONTINUE
OC 185 J = 1, N
185 CONTINUE
C C
C C C
UPDATES INDEX ARRAY OF BASIC VARIABLES IVAR(I), BASIC VARIABLE COSTS CI(I), AND THE TOLERANCE ERROR. IT RETURNS TO CONTINUE THE DUAL SIMPLEX PROCEDURE.

CI(IR)=CI(IS)
IVAR(IW)=IS
EPS=EPS+.5E-6
GO TO 139

MINIMUM RATIO RULE FAILED. PROBLEM IS INFEASIBLE.

190 IFLAG=1
GO TO 220

DETERMINES PRIMAL SOLUTION AND FILLS H VECTOR.

195 VALUE=0.,0.
DC 200 I=1,M
200 VALUE=VALUE+RHS(I)*CI(I)
RM=VALUE
GG 210 K=1,M
KK=2*NX+NC+NACC+K
RHS(K)=0.
DO 205 I=1,M
RHS(K)=RHS(K)+CI(I)*CCL(I,KK)
205 CONTINUE
IF (ABS(RHS(K)) .LT. 1.0E-6) RHS(K)=0.
210 CONTINUE
DC 215 I=1,NX
II=NX+I
215 +H(I)=RHS(I)-RHS(II)
220 RETURN

END

SUBROUTINE GSS (RC,ERR,X,DI,FLA40A)

GSS 10
GSS 20
GSS 30
GSS 40
GSS 50
GSS 60
GSS 70
GSS 80
GSS 90
GSS 100
GSS 110
GSS 120
GSS 130
GSS 140
GSS 150
GSS 160
GSS 170
GSS 180
GSS 190
GSS 200
GSS 210

THIS SECTION CONFUTES AN INTERVAL (RIM1,RJ) CONTAINING A MINIMIZING POINT.

RIM1=0.
RI=0.
CALL EVALU (X,RI,FI,ICON)
FIM1=FI
105 RJ=RI+RHO
CALL EVALU (X,RJ,FJ,ICON)
IF (FJ,GE,FI) GO TO 110
FIM1=FI
FI=FJ
RIM1=RI
RI=RF
GO TO 105

110 CONTINUE

A MINIMIZING POINT EXISTS BETWEEN RIM1 AND RJ.

A=RIM1
E=RI
FJ=FIM1
FE=FJ

THIS SECTION REJECTS THE LENGTH OF THE INTERVAL CONTAINING THE MINIMIZING POINT.

K=1.8619033
RSD=P*R
XH=B-A
XL=A+RSD*W
XR=A+R*W
CALL EVALO (X,XL,FL,ICON)
CALL EVALO (X,XR,FR,ICON)

115 IF (FR,XL,FL) GO TO 120

TEST THE REMAINING INTERVAL (A,XR). DISCARD ALL BEYOND X=RIGHT, AND COMPUTE A NEW X=LEFT.

REGION=XR-A
IF (REGION.LE.ERRI) GO TO 125
E=XR
FE=FR
XH=XR-A
XL=XH
FR=FL
XR=A+RSD*W
CALL EVALO (X,XL,FL,ICON)
GO TO 115

TEST THE REMAINING INTERVAL (XL,B). DISCARD ALL BELOW X=LEFT, AND COMPUTE A NEW X=RIGHT.

REGION=B-XL
IF (REGION.LE.ERRI) GO TO 130
A=XL
FA=FL
XH=E-XL
XL=XH
FR=FR
XR=A+R*W
CALL EVALO (X,XR,FR,ICON)

120 CONTINUE

130 CONTINUE

GSS 220
GSS 230
GSS 240
GSS 250
GSS 260
GSS 270
GSS 280
GSS 290
GSS 300
GSS 310
GSS 320
GSS 330
GSS 340
GSS 350
GSS 360
GSS 370
GSS 380
GSS 390
GSS 400
GSS 410
GSS 420
GSS 430
GSS 440
GSS 450
GSS 460
GSS 470
GSS 480
GSS 490
GSS 500
GSS 510
GSS 520
GSS 530
GSS 540
GSS 550
GSS 560
GSS 570
GSS 580
GSS 590
GSS 600
GSS 610
GSS 620
GSS 630
GSS 640
GSS 650
GSS 660
GSS 670
GSS 680
GSS 690
GSS 700
GSS 710
GSS 720
GSS 730
GSS 740
GSS 750
GSS 760
GSS 770
GSS 780
GSS 790
GSS 800
GSS 810
GSS 820
GO TO 115

C THE SOLUTION LIES BETWEEN A AND XR.

125 XR=XR
   XL=A
   GO TO 135

C THE SOLUTION LIES BETWEEN XL AND B.

130 XR=B
   XL=XL
   PRINTOUT

135 FLAMDA=(XR+XL)*.5
   CALL EVALD(XR,XL,FLAMDA,DI=4,ICON=)
   FI=FI-DIST
   WRITE (6,140) PC,-DI=1,ICON=-
   RETURN

C END

SUBROUTINE EVALD(XR,XL,FLAMDA,ICON=)

C THIS SUBROUTINE EVALUATES THE MINIMUM DISTANCE TO THE C
C BOUNDARY.

COMMON /TCOLY/ NY,NCF,NC,NGACC
COMMON /TCOLY/ IDADE(20,10),HCL(20),H(20)
DIMENSION XNEW(20),X(20),F(10),G(20)

CG 105 I=1,AX
105 XNEW(I)=X(I)+D*F(I)
   CALL FFXNS(XNEW,F)
   CALL GFXNS(XNEW,G)

C FINDS THE MINIMUM DISTANCE TO ALL THE ORIGINAL CONSTRAINTS.

ICON=1
CVM=G(I)
   I=2
110 IF (G(I),LE.,CVM) GO TO 115
   ICON=I
   CVM=G(I)
115 I=I+1
IF (1.0 <= NC) GO TO 110

FINDS THE MINIMUM DISTANCE TO ALL THE CONSTRAINTS.

CC 125 K=1, NACC
GCNX = 0,
DO 120 K=1, NACC
   GCNX = GCNX + TRADE(K, KK) * F(K)
120 CONTINUE
   GCNX = GCNX - HOLD(K)
IF (GCNX .LE. CVM) GO TO 125
ICON = NC + K
CVM = GCNX
125 CONTINUE
RETURN
C
END
SUBROUTINE HOLDC (NCN, F)
CC
THIS SUBROUTINE DETERMINES THE CONSTANT TERM ASSOCIATED WITH EACH NEW CONSTRAINT (I.E. TRADEOFF CUT).
CC
COMMON /TCOP1/ K, NACC, NC, NACC
COMMON /TCOP2/ TRADE(20, 10), HOLD(20), H(20)
DIMENSION F(10)
C
HOLD(NCN) = 0,
CC 105 K=1, NCF
105 HOLD(NCN) = HLC(NCN) - TRADE(NCN, K) * F(K)
RETURN
C
END
SUBROUTINE FFYNS (X, F)
DIMENSION X(21), F(2)
F(1) = -(X(1) ** 2) / 10. + 1.5 * X(1)
F(2) = -(X(2) ** 2) / 10. + X(2)
RETURN
C
END
SUBROUTINE GFYNS (X, G)
DIMENSION X(2), G(4)
G(1) = -(X(1) ** 2) - 6. + X(1) + 4. * X(2) - 16
G(2) = EXP(X(1) - 3.) - X(1) * X(2) + 3. * X(3) - 1.
G(3) = X(1)
G(4) = X(2)
RETURN
C
END
SUBROUTINE FGPAE (X, GF)
DIMENSION X(2), GF(4)
GF(1) = 0.2 * X(1) + 1.6
GF(2) = 0.
GF(3) = 0.
GF(4) = 0.2 * X(2) + 1.
RETURN

C EAD
SUBROUTINE GGRAC (X,GG)
DIMENSION X(2), GG(4)
GG(1) = 2 * X(1) - 6
GG(2) = 4
GG(3) = EXP(X(1)) - 3
GG(4) = -X(1) + 3
GG(5) = -1
GG(6) = 0
GG(7) = 0
GG(8) = 1
RETURN

C EAD
/*EGF*/
/*EGF*/