INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of “sectioning” the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.
Ranganathan, Anbil Nityanand

INVESTIGATION OF THREE RESOURCE ALLOCATION ALGORITHMS FOR CONTROLLING MINE DRAINAGE POLLUTION

The Ohio State University

University Microfilms International

Copyright 1984 by Ranganathan, Anbil Nityanand

All Rights Reserved
INVESTIGATION OF THREE RESOURCE ALLOCATION ALGORITHMS
FOR CONTROLLING MINE DRAINAGE POLLUTION

DISSertation

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of The Ohio State University

By
Anbil N. Ranganathan, B.Tech., M.S.

* * * * *

The Ohio State University
1984

Reading Committee:
Dr. Gordon M. Clark
Dr. Clarence H. Martin
Dr. Clark A. Mount-Campbell
Dr. Charles H. Reilly

Approved By

Gordon M. Clark
Adviser
Department of Industrial and Systems Engineering
ACKNOWLEDGEMENTS

I would like to thank Professor Gordon M. Clark for his constant support, encouragement and guidance both personally and professionally throughout my many years of education and research at the Ohio State University. I would also like to thank Dr. Clarence Martin for encouraging and guiding me in linear modeling and integer linear programming considerations in this research. I am grateful to David Gonsalvez for providing and helping me with his integer linear programming code. And I owe mostly to my wife for her unwavering confidence in me.
VITA

August 15, 1950  Born - Sivaganga, Tamil Nadu, India

1972  B. Tech., The Indian Institute of Technology, Madras, India

1974  M.S., The Ohio University, Athens, Ohio

1974-1977  Research Associate, Department of Industrial and Systems engineering, The Ohio State University, Columbus, Ohio

1977-1979  Research Associate, Computer Center, The Ohio State University, Columbus, Ohio

1979-1981  Programmer Analyst, Owens Corning Fiberglas, Toledo, Ohio

1982-1983  Research Associate, Computer Center, The Ohio State University, Columbus, Ohio

PUBLICATIONS


FIELDS OF STUDY

Major Field: Industrial and Systems Engineering

Studies in Simulation and Statistics. Professor Gordon M. Clark

Studies in Operations Research. Professor Walter C. Giffin

Studies in Mathematical Programming. Professors Subash Sarin, Clark A. Mount-Campbell and Clarence H. Martin
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>ii</td>
</tr>
<tr>
<td>VITA</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>Chapter</td>
<td></td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. A MINIMUM COST RESOURCE ALLOCATION PROBLEM</td>
<td>6</td>
</tr>
<tr>
<td>DEFINITIONS</td>
<td>6</td>
</tr>
<tr>
<td>FORMULATION</td>
<td>15</td>
</tr>
<tr>
<td>Pollution Flows</td>
<td>21</td>
</tr>
<tr>
<td>Average Annual Pollutant Loads</td>
<td>25</td>
</tr>
<tr>
<td>Cost, Revenues, and the Criterion Function</td>
<td>26</td>
</tr>
<tr>
<td>Quality Constraints</td>
<td>29</td>
</tr>
<tr>
<td>The Minimum Cost Problem</td>
<td>30</td>
</tr>
<tr>
<td>Literature Review</td>
<td>31</td>
</tr>
<tr>
<td>III. MODIFICATIONS AND EXTENSIONS TO THE LAWLER-BELL ALGORITHM</td>
<td>42</td>
</tr>
<tr>
<td>THE LAWLER-BELL ALGORITHM</td>
<td>43</td>
</tr>
<tr>
<td>OVERVIEW OF ALCOT</td>
<td>45</td>
</tr>
<tr>
<td>REDEFINITION OF THE SOURCE DECISION VARIABLES</td>
<td>50</td>
</tr>
<tr>
<td>ORDERING DECISION VARIABLES IN A VECTOR X</td>
<td>58</td>
</tr>
<tr>
<td>NEXT FEASIBLE SOLUTION PROCEDURE IN CONJUNCTION</td>
<td>59</td>
</tr>
<tr>
<td>WITH THE BRANCH AND BOUND AND DECOMPOSITION METHODS</td>
<td></td>
</tr>
<tr>
<td>Case: 1 First Decision Variable in Watershed h, h&lt;H</td>
<td>60</td>
</tr>
<tr>
<td>Case 2: Instream Treatment Decision Variable</td>
<td>61</td>
</tr>
<tr>
<td>Case 3: Source Decision Variable Defined at a Potential Multiple Effluent Mine Node</td>
<td>62</td>
</tr>
<tr>
<td>Case 4: Source Decision Variable Defined at an Abandoned or Active Multiple effluent Mine Node</td>
<td>64</td>
</tr>
<tr>
<td>Case 5: Source Decision Variable at a Single Effluent Mine; Abandoned, Active or Potential</td>
<td>64</td>
</tr>
<tr>
<td>Application of Branch and Bound</td>
<td>65</td>
</tr>
<tr>
<td>STORING MINIMUM COST SUBVECTOR SOLUTIONS</td>
<td>73</td>
</tr>
<tr>
<td>SKIPPING FROM X TO X</td>
<td>75</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A Reservoir's Potential for Abatement</td>
<td>14</td>
</tr>
<tr>
<td>2. Test Problem Features</td>
<td>157</td>
</tr>
<tr>
<td>3. Test Problem Characteristics</td>
<td>158</td>
</tr>
<tr>
<td>4. ALCOT Results</td>
<td>158</td>
</tr>
<tr>
<td>5. Lagrangian Relaxation Results</td>
<td>160</td>
</tr>
<tr>
<td>6. ENUMCODE Results</td>
<td>161</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Stream Network Representation</td>
<td>7</td>
</tr>
<tr>
<td>2. Representation of Mine Effluents and Nodes</td>
<td>8</td>
</tr>
<tr>
<td>3. Optimum Abatement Level Representation at a Multiple Effluent Mine</td>
<td>114</td>
</tr>
<tr>
<td>4. Subproblems in a Watershed</td>
<td>131</td>
</tr>
</tbody>
</table>
CHAPTER I
INTRODUCTION

This dissertation concerns the acid mine drainage pollution problem. Acid mine drainage originates when pyritic compounds alongside ore strata, usually coal, become exposed to atmospheric oxygen during and after mining and undergo oxidation. Once oxidized, the compounds dissolve in rain water, and the pH in the mine water effluent can drop considerably. At this low pH, metals that are usually insoluble, also dissolve, and constitute further pollution. In 1964, the U.S. Fish and Wildlife Service estimated that fish and wildlife habitats in 6000 miles of streams in the United States were adversely affected by acid mine drainage [9].

Perhaps the most detrimental feature of acid generation in the mines is its interminable characteristic, at least for all practical purposes. Acid generation is a slow combustion process analogous to rusting. A mine can continue to drain acid for decades, even centuries, long after it has been abandoned.

Prior to the Nineteen Sixties, proper regulations for mine site abandonment and pollution control did not exist, the result of which is many severely polluting abandoned mine sites, especially in Pennsylvania and West Virginia where coal was mined extensively to meet the rail,
steel and power industries' demands. Restoring these polluted areas will cost millions of dollars, and the government must absorb these costs either directly or indirectly by compensating miners or private institutions undertaking to control pollution from abandoned mines.

Considering the present budget deficit facing the U.S. Government, it is reasonable to expect the cleanup to be gradual, and it is important to find the more cost-effective sites for pollution abatement.

Two basically different methods exist for controlling acid mine drainage: at-source abatement and chemical or physical treatment of the drainage. Examples of at-source abatement are [15]: 1) sealing and flooding of underground mines, and 2) regrading, covering, and revegetation of surface mines and gob piles. All at-source methods have the common aim of eliminating the exposure of pyritic materials to atmospheric oxygen. The primary advantage of abatement is maintenance costs are low; however, acid drainage is not completely eliminated, the maximum effectiveness attainable by abatement may be achieved only after a long period of time, and the cost-effectiveness of abatement depends on the individual characteristics of mine sites. Chemical or physical treatment on the other hand, can neutralize all of the acid drainage and remove all dissolved metals [1]. However, operating costs of treatment installations are significant and the required period of treatment is, for all practical purposes, interminable.

Many areas in the United States that have significant coal deposits also have abandoned mine sites in the same watershed. This makes the problem more complex and potential economies more significant because
options referring to the types of mine development, or not to mine at all, exist in addition to possible pollution control measures.

Two primary mining methods, surface and underground, are employed in mineral extraction. Basic factors involved in determining the type of mining method to be utilized for a particular mineral deposit are: grade of ore, physical characteristic of ore deposit and associated ratio of overburden thickness to mineral thickness. Surface mining is preferred, where possible, over underground mining because of its higher mineral recovery, much greater production per man-day, more economical operation in terms of capital costs and cost per unit, better health and safety conditions, and more flexibility of operation.

Current legislative controls and guidelines in most states help minimize operational and post operational effects from mining. These legislative actions require the miner to consider pollution control as an integral part of the overall mining effort. For example, the design of underground mines for down-dip development can facilitate inundation and sealing by lessening the potential hydrostatic head pressure against seal installations. Similarly, surface mining operations using "overburden segregation" can provide for the proper burial of pollutant bearing and toxic materials and the recovery of top soil.

Another complexity of the mine pollution control problem is that pollution control expenditure can often be justified only after predictable enhancements to the socio-political-economic conditions existing in the affected watersheds, over and above improvements to
water quality, ecosystems and aesthetics. However, it is usually difficult to establish a direct link between socio-economic growth and environmental quality if for no other reason than the problem of quantifying "environmental quality". One study [13] itemizes various factors for evaluating the socio-economic impacts due to pollution, and ultimately arrives at a weighting scheme to sort out the relative importance of individual areas in the affected watersheds. Typically considered factors are: population and migration rates, existing and proposed water and land uses (residential, commercial, industrial, agricultural, recreational, etc.) along individual stream reaches, and community sensitivity and policy towards pollution control. Another study [15] also uses a weighting scheme to identify the relative socio-economic importance of individual stream reaches and combines these weights with ecosystem requirements. Thus the value of water may vary from point to point within a watershed, and it may be unrealistic to specify a uniform quality standard for the entire watershed.

Typically, the problem of allocating resources for cost-effective control of acid mine drainage cannot be viewed individually for the mine pollution sources; rather the global view of total watershed degradation is necessary. That is, the pollution experience at points downstream in the watershed will be greater than that experienced at upstream points because of pollution aggregation in the direction of flow. On the other hand, a stream's capacity for assimilating pollution increases as the stream flows downstream because of flow volume increase. These interactions between pollution aggregation and the pollution
assimilative capacities of streams play a crucial role in the selection of mining and pollution control methods at the various sites. So, a careful mathematical analysis of the problem can illuminate solution strategies and justify cost savings.

The effort here extends from a E.P.A. sponsored research contract [2] where a Multiple Watershed Resource Allocation Model was developed. This discourse confines to this model and presents three promising discrete optimization algorithms for a resource allocation problem formulated from the model. The dissertation concludes with a comparative evaluation of the three algorithms.
CHAPTER II
A MINIMUM COST RESOURCE ALLOCATION PROBLEM

This chapter presents a problem and its formulation for minimizing the cost of resource allocation and achieving specified quality standards across one or more watersheds. As noted, this problem comes from a contractual undertaking with the E.P.A. [12]. This chapter also reviews the literature for solution approaches to this problem.

DEFINITIONS

Let the mining complex of interest consist of a mix of abandoned, active, and potential mine pollution sources spread over one or more adjacent watersheds. Let each watershed have a single watershed outlet and call it the level 3 stream. Define tributaries of the level 3 stream, if they exist, as level 2 streams. Likewise, if one or more level 2 streams each has tributaries, define them as level 1 streams. Assume level 1 streams do not have tributaries. Figure 1 illustrates the representation of streams in two watersheds.

Along each stream denote a set of points where the mine effluents merge with the streams. Call these points "nodes". Figure 2 illustrates mine sources, mine effluents, and nodes. Note that a mine
may emit a single pollutant effluent or multiple pollutant effluents. A multiple effluent mine can exhibit any of the following possibilities:

1. Effluents enter a single stream at different nodes.
2. Effluents enter different streams in a single watershed.
3. Effluents enter streams in different watersheds.
4. Effluents are distributed in a pattern displaying any combination of possibilities 1, 2, and 3, above.

In general, let each node receive a unique mine effluent.
Figure 2. Representation of Mine Effluents and Nodes
Allow for three mining alternatives at each potential mine site, namely, do not mine, develop a surface mine and develop an underground mine. It is possible that the number and distribution of effluents from the potential mine site depends on the mining alternative selected; i.e., surface or underground mining. However, for ease of formulation, we regard the surface and deep mining alternatives to result in the same number and distribution of effluents. Nevertheless, in the event the number and distribution of effluents vary with the mining alternatives, the variation can be portrayed implicitly by allowing for all possible effluents and supplying zero pollutant flow data when the effluents do not apply.

Conventional procedures [12] can provide the mining costs and revenues associated with the surface and underground mining alternatives. Let all cost and revenue data be in equivalent annual dollars spread over a viable planning horizon, say one, two or three decades.

Assume zero pollutant flow in each effluent of a potential mine site when the "do not mine" decision is in effect. For the surface and deep mine options, existing at-source simulation models [15,12], parametrized using nearby abandoned or active mine pollution data whenever available, can be used to predict the pollutant flow in each potential mine effluent. These models can also provide a more complete representation of pollutant flows from abandoned and active mines than that estimable from available data.
Let all pollutant flow data be in kilograms of acid per hour. In actuality, this flow is time varying and stochastic principally due to precipitation patterns. However, studies [15,12] indicate the existence of certain precipitation patterns which produce adverse pollution concentrations. For example, a sudden summer storm after a long dry spell results in adverse surface mine pollutant flows. Similarly, when the ground thaws in Spring, a long winter pollutant build-up in the deep mines can be flushed out. Thus, let the pollution and stream flow data here represent such typical "worst-case" situations. This simplifies the time-varying stochastic pollutant and stream flows to time-invariant deterministic flows, the justification being any set of decisions that meets the quality standard for each worst-case situation will stand-up for all times.

Let there exist the decision to abate or not to abate at each single effluent source, whether abandoned, active or potential. Assume that abatement reduces the pollutant flow in the effluent but does not necessarily eliminate it. Since existing at-source simulation models [15,12] are designed to respond to changes in the pyrite in the system and to changes in the hydrologic regimes affecting the system, these changes can be estimated for a given abatement technique and the post-abatement pollutant flow in the mine effluent can be predicted for each worst-case situation. Regarding the cost of abatement, standard procedures [4] can provide an estimate. Assume abatement incurs a fixed investment cost and negligible maintenance cost.
At each multiple effluent mine source, whether abandoned, active, or potential, several alternative levels of abatement are possible, where the first level means a certain amount of abatement, the second level means a certain additional extent, and so on. This view comes from abatement practiced at multiple opening underground mines where openings on the lowest level or elevation are abated first, openings on the next higher level are abated next, and so on. However, at a multiple effluent surface mine, it may be most economical to abate the entire site, once the decision has been made to abate at the site. In this case, a single level of abatement would be appropriate. Thus, the abatement strategy at a multiple effluent potential mine can depend on the mining alternative -- surface or underground. However, through judicious specifications of zero pollutant flow data whenever pollution flows do not apply, the number of alternative levels of abatement can be held constant regardless of the type of mine development. This simplifies formulation. In general, let the first level of abatement reduce the pollutant flow in certain mine effluents, the second level in certain additional effluents, and so on, the final level implying reductions in all mine effluents.

For each mine effluent, let there exist the decision to treat or not to treat the effluent. Assume treatment neutralizes all pollutants in the effluent yielding zero pollutant flow, with the contention that it would be most economical to remove all pollutants once the decision has been made to install the treatment processor. Regarding costs, standard procedures [4] can provide the investment and operating costs of
treatment. In general, the cost of treatment is a function of flow and loading with unit costs decreasing with increasing scale of the treatment system. Plant size depends largely on the flow rate of the drainage while the cost of chemicals rises directly with the average annual acid load to the plant. Thus, let treatment incur a fixed investment cost and a variable cost directly proportional to the average annual acid load to the plant. Existing simulation models [15,12] can provide the pre- and post-abatement average annual acid loads in kilograms via each mine effluent, and the cost of treating one unit of pollution in dollars per kilogram of acid can be estimated depending on the particular chemical reagent to be deployed, usually lime or limestone slurry.

Let there also exist a set of nodes that are likely sites for treatment in the stream channel. Because stream flow volumes are higher than mine effluent volumes, instream treatment will incur a relatively higher cost than source treatment. However, since unit cost decreases with plant size, the simultaneous treatment of the aggregate pollution flow from upstream mine sources can be more economical than controlling the pollution flows individually at these sources. Like source treatment, let instream treatment incur a fixed investment cost and a variable cost directly proportional to the aggregate average annual acid load to the plant. Further, let instream treatment neutralize all of the pollutants in the stream yielding zero pollutant flow. But assume it will not affect alkaline flows, since alkaline flows can alleviate part of any potential acid mine drainage. Alkaline flows in the stream channel can be due to natural causes.
Natural pollutants can occur throughout the streams in the watersheds. Let the distribution of natural pollutants be uniform in the stream channel, but let each stream reach between adjacent nodes have its own unique natural pollutant occurrence rate. Assume that the natural pollutant occurrence rate in each stream reach for each worst-case situation is given data in kilograms of acid per hour. Assume likewise for the average annual natural pollutant load in each reach in kilograms. These worst-case and average annual data may be positive indicating acidity or negative indicating alkalinity.

Each watershed may also have a reservoir, which can be a natural means for pollution abatement. In the present context, a reservoir can be viewed as a stabilizer of each worst-case situation. That is, a reservoir can delay for some time the worst-case pollutant flows from upstream mine sources. When these flows eventually enter reaches downstream of the reservoir, there can be a reduction in the adversity of pollutant flows from mine sources in the vicinity. This concept can be applied by splitting a worst-case situation into several situations. For example, consider Table 1 which contains the pollutant flows in kilograms per hour at times 0, 1, and 2 hours, from two mine sources, one upstream of the reservoir, and another downstream of the reservoir. Assume that the reservoir delays the pollutant flow by an hour. If we had to represent the worst-case situation by a single set of values, we would be forced to select the flows 14 and 15 occurring after one hour for logical reasons. However, if we represented the worst-case situation by two sets of values, i.e., 8 and 15, and 14 and 6, which
Table 1. A Reservoir's Potential for Abatement

<table>
<thead>
<tr>
<th>Time (hr)</th>
<th>Upstream Source</th>
<th>Downstream Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

account for the one hour delay in the reservoir, then the maximum possible flow would be 23 as opposed to 29. In general, the greater the number of sets of pollution flow values picked around a worst-case situation, the more accurate will be the delay representation. However, the problem increases in complexity with increasing numbers of these sets. So, as few sets as are necessary, can be chosen initially; later, depending on the significance of the reservoir on the minimum cost solution, additional sets of values may be gathered if warranted. For ease of formulation, regard each set of values as a unique worst-case situation.

The purpose of resource allocation here is to achieve a specified quality standard at each node in the stream networks in the cheapest possible way. Let all costs and revenues (regarded as negative costs) be totalled together regardless of the agency that incurs these costs,
i.e., the miner or the government. The idea is to minimize this total cost.

Most states have acidity pollution standards based upon pH since biological processes are related to pH; however, given a pH standard, an analysis of the acid buffering capacities of the watersheds can produce an equivalent standard based upon total acidity. Thus, let the quality standard at a node be expressed as pollution concentration in parts of acid per million parts of solution (water plus acid), and this concentration should not be exceeded for each worst-case situation. Either a uniform quality standard may be applied throughout the watersheds, or a standard specific to each node may be applied. As noted, the relative socio-economic importance of individual stream reaches can dictate different quality requirements for different reaches.

FORMULATION

Let the nodes defined in the stream networks of all the watersheds be numbered from one to J, and let j be the node index. Define

\[ J^a = \{ j | j \text{ receives an effluent from an abandoned or active mine} \} \]

and

\[ J^p = \{ j | j \text{ receives an effluent from a potential mine} \}. \]

Note that

\[ J^a \cup J^p = \{ j | j=1, \ldots, J \} \]

and

\[ J^a \cap J^p = \emptyset. \]
Also let

\[ J_t = \{ j | j \text{ is a potential instream treatment site} \} \].

Elemental analysis of the decisions indicated earlier is convenient when we define decisions at nodes rather than at mine sites. Define the decision variables as follows, where the phrase "abate the effluent" means abate at the site releasing the effluent so as to reduce the pollutant flow in the effluent.

\[ x^a_{j} = \begin{cases} 
1 & \text{abate the mine effluent entering } j \\
0 & \text{but do not treat the effluent} \\
0 & \text{otherwise} 
\end{cases} \quad j \in J^a
\]

\[ x^{a}_{j} = \begin{cases} 
1 & \text{treat the mine effluent entering } j \\
0 & \text{but do not abate the effluent} \\
0 & \text{otherwise} 
\end{cases} \quad j \in J^a
\]

\[ x^{at}_{j} = \begin{cases} 
1 & \text{abate and treat the mine effluent to } j \\
0 & \text{otherwise} 
\end{cases} \quad j \in J^a
\]

\[ x^{s}_{j} = \begin{cases} 
1 & \text{develop a surface mine at the site releasing the effluent to } j \\
0 & \text{but implement no pollution control} \\
0 & \text{otherwise} 
\end{cases} \quad j \in J^p
\]

\[ x^{sa}_{j} = \begin{cases} 
1 & \text{develop a surface mine at the site releasing the effluent to } j \text{ and abate the effluent,} \\
0 & \text{but do not treat the effluent} 
\end{cases} \quad j \in J^p
\]

\[ x^{st}_{j} = \begin{cases} 
1 & \text{develop a surface mine at the site releasing the effluent to } j \text{ and treat the effluent, but do not abate} \\
0 & \text{otherwise} 
\end{cases} \quad j \in J^p
\]

\[ x^{sat}_{j} = \begin{cases} 
1 & \text{develop a surface mine at the site releasing the effluent to } j, \text{ and abate and treat the effluent} \\
0 & \text{otherwise} 
\end{cases} \quad j \in J^p
\]
\[
x^d_j = \begin{cases} 
1 \text{ develop a deep mine at the site} \\
\text{releasing the effluent to } j, \text{ but} \\
\text{implement no pollution control} \\
0 \text{ otherwise}
\end{cases} \quad j \in J^P
\]

\[
x^{da}_j = \begin{cases} 
1 \text{ develop a deep mine at the site} \\
\text{releasing the effluent to } j \text{ and abate} \\
\text{the effluent, but do not treat} \\
0 \text{ otherwise}
\end{cases} \quad j \in J^P
\]

\[
x^{dt}_j = \begin{cases} 
1 \text{ develop a deep mine at the site} \\
\text{releasing the effluent to } j \text{ and treat} \\
\text{the effluent, but do not abate} \\
0 \text{ otherwise}
\end{cases} \quad j \in J^P
\]

\[
x^{dat}_j = \begin{cases} 
1 \text{ develop a deep mine at the site} \\
\text{releasing the effluent to } j, \\
\text{and abate and treat the effluent} \\
0 \text{ otherwise}
\end{cases} \quad j \in J^P
\]

\[
x^z_j = \begin{cases} 
1 \text{ perform instream treatment at } j \\
0 \text{ otherwise}
\end{cases} \quad j \in J^t
\]

We require

\[
x^{a}_j + x^{t}_j + x^{at}_j \leq 1 \quad \text{for each } j \in J^a \tag{2.1}
\]

and

\[
x^{s}_j + x^{sa}_j + x^{st}_j + x^{sat}_j + x^{d}_j + x^{da}_j + x^{dt}_j + x^{dat}_j \leq 1 \quad \text{for each } j \in J^p. \tag{2.2}
\]

Note that when an abandoned or active mine node's decision variables are all zero, the implication is "no pollution control", and likewise, when a potential mine node's decision variables are all zero, the implication is "do not mine".
We also require additional logical relationships between decision variables defined at nodes of each multiple effluent mine. Let

\[ M = \text{total number of multiple effluent mines}, \]
\[ m = \text{multiple effluent mine index}, \]
\[ M^a = \{m|m \text{ is an abandoned or active mine}\}, \]
\[ M^p = \{m|m \text{ is a potential mine}\}, \]
\[ B_m = \text{total number of abatement levels considered at mine } m, \]
\[ b = \text{abatement level index}, \]
\[ J_{bm}^m = \{j|\text{node } j\text{'s effluent is affected by abatement at level } b \text{ or abatement at a higher level}\}, \] and

\[ J_{*m} = \bigcup_{b=1}^{B_m} J_{bm}^m = \{j|\text{the effluent entering node } j \text{ is released by mine } m\}. \]

Note that effluents released from a multiple effluent mine are classified by abatement level. That is, an effluent belonging to an abatement level has its pollution flow reduced only when abatement is performed at its level or at a higher level. As noted, this construction is based on abatement practiced at multiple opening underground mines.

We require, when we do specify a mining decision at a node denoted for a potential multiple effluent mine, the same decision must be reflected at every other node of the same mine. This can be accomplished as follows. Let
\[ y^s_m = \begin{cases} 1 & \text{develop a surface mine at } m, m \in M^p \\ 0 & \text{otherwise} \end{cases} \]

and \[ y^d_m = \begin{cases} 1 & \text{develop a deep mine at } m, m \in M^p \\ 0 & \text{otherwise} \end{cases} \]

Then \[ x^s_j + x^{sa}_j + x^{st}_j + x^{sat}_j = y^s_m \text{ for each } j \in J_m, m \in M^p \] (2.3)

and \[ x^d_j + x^{da}_j + x^{dt}_j + x^{dat}_j = y^d_m \text{ for each } j \in J_m, m \in M^p \] (2.4)

For future reference, we make the following point. That is, we need not explicitly require that \( y^s_m \) and \( y^d_m \) be binary variables. These variables are implied to be binary because of the constraints (2.2) to (2.4). Therefore we regard these variables as unrestricted in Chapter IV.

We also require that the abatement decisions at nodes denoted for multiple effluent mines conform with abatement levels. This can be accomplished as follows. Let

\[ z^a_{bm} = \begin{cases} 1 & \text{perform abatement at level } b \text{ of mine } m; \text{ i.e., abate to reduce the pollutant flow in the effluents belonging to levels } 1 \text{ to } b \\ 0 & \text{otherwise} \end{cases} \]

\[ m \in M^a \]

\[ z^{sa}_{bm} = \begin{cases} 1 & \text{perform abatement at level } b \text{ of potential surface mine } m; \text{ i.e., abate to reduce the pollutant flow in the potential surface mine effluents belonging to levels } 1 \text{ to } b \\ 0 & \text{otherwise} \end{cases} \]

\[ m \in M^p \]

and \[ z^{da}_{bm} = \begin{cases} 1 & \text{perform abatement at level } b \text{ of potential deep mine } m; \text{ i.e., abate to reduce the pollutant flow in the potential deep mine effluents belonging to levels } 1 \text{ to } b \\ 0 & \text{otherwise} \end{cases} \]

\[ m \in M^p \]
Then \( x_j^a + x_j^{sat} = z_{bm}^a + z_{b+1,m}^a + \cdots + z_{B_m,m}^a \)
for each \( j \in J_{bm}, b = 1 \) to \( B_m \) and \( m \in \mathbb{M}^a \) \hspace{1cm} (2.5)

\( x_j^{sa} + x_j^{sat} = z_{bm}^{sa} + z_{b+1,m}^{sa} + \cdots + z_{B_m,m}^{sa} \)
for each \( j \in J_{bm}, b = 1 \) to \( B_m \) and \( m \in \mathbb{M}^p \) \hspace{1cm} (2.6)

and \( x_j^{da} + x_j^{dat} = z_{bm}^{da} + z_{b+1,m}^{da} + \cdots + z_{B_m,m}^{da} \)
for each \( j \in J_{bm}, b = 1 \) to \( B_m \) and \( m \in \mathbb{M}^p \) \hspace{1cm} (2.7)

For future reference, we make the following point. That is, we need not explicitly require \( z_{bm}^a, z_{bm}^{sa} \) and \( z_{bm}^{da} \) each to be binary. These variables are forced to be binary because of the constraints (2.1) and (2.2) in conjunction with the constraints (2.5) to (2.7) at \( b = B_m \). So we regard these variables as unrestricted in Chapter IV. Similarly, the less stringent restrictions:

\[
\begin{align*}
  z_{bm}^a & \geq 0 \text{ for each } b = 1 \text{ to } B_m - 1 \\
  z_{bm}^{sa} & \geq 0 \text{ for each } b = 1 \text{ to } B_m - 1 \\
  z_{bm}^{da} & \geq 0 \text{ for each } b = 1 \text{ to } B_m - 1
\end{align*}
\]

are sufficient to force these variables to be binary in view of the constraints (2.1), (2.2) and (2.5) to (2.7). Hence, in Chapter IV, we regard these variables as having lower bounds of zero but having no upper bounds.
Pollution Flows

There are three alternative pollutant flows possible from each abandoned or active mine effluent, for each worst-case situation. One flow refers to the zero flow when the effluent is treated. The others are:

\[ w_{ij} = \text{pollutant flow in kilograms per hour in the abandoned or active mine effluent entering node } j \text{ for worst-case } i \text{ when no control action is taken} \]

and \[ w_{ij}^a = \text{pollutant flow in kilograms per hour in the abandoned or active mine effluent entering node } j \text{ for worst-case } i \text{ after abatement}. \]

The number of worst-case situations can vary from watershed to watershed depending on the presence or absence of reservoirs. Let

\[ H = \text{total number of watersheds} \]
\[ h = \text{watershed index} \]
\[ I_h = \text{total number of worst-case situations considered in watershed } h \]

and \[ h_j = \text{watershed containing node } j. \]

Then the definitions \[ w_{ij} \text{ and } w_{ij}^a \] apply for \( i = 1 \text{ to } I_h \text{ and for each } j \in j^a. \)

There are five alternative pollutant flows possible from each potential mine effluent for each worst-case situation, one being the zero pollutant flow as a result of treatment or when the "do not mine" decision is in effect. The others are:
\( w_{ij}^s \) = potential surface mine pollutant flow in kilograms per hour in the effluent to \( j \) for worst-case \( i \) when no control action is taken,

\( w_{ij}^{sa} \) = potential surface mine pollutant flow in kilograms per hour in the effluent to \( j \) for worst-case \( i \) after abatement,

\( w_{ij}^d \) = potential deep mine pollutant flow in kilograms per hour in the effluent to \( j \) for worst-case \( i \) when no control action is taken

and \( w_{ij}^{da} \) = potential deep mine pollutant flow in kilograms per hour in the effluent to \( j \) for worst-case \( i \) after abatement.

The above definitions are true for each \( i = 1 \) to \( I_{h_j} \) and \( j \in J^p \).

In general, let

\( w_{ij}^e \) = pollutant flow in kilograms per hour in the mine effluent to node \( j \) for worst-case \( i \) per the decision taken at the source.

Then

\[
\begin{align*}
w_{ij}^e &= w_{ij} - (w_{ij}^{a} - w_{ij}^a)x_j^a - w_{ij}^t x_j^x - w_{ij} x_j \text{at} \\
\text{for each } i &= 1 \text{ to } I_{h_j} \text{ and } j \in J^{a}
\end{align*}
\]  

(2.8)

and

\[
\begin{align*}
&\text{for each } i = 1 \text{ to } I_{h_j} \text{ and } j \in J^p.
\end{align*}
\]  

(2.9)

The pollution flow past a node is taken as the sum of three inputs:

1) the flow past the next upstream node or nodes, 2) natural pollutant
flow occurring between the next upstream nodes and the current node, and
3) the pollutant flow in the node’s mine effluent. Of course, if the
node hosts an active instream processor, then the sum of these three
inputs will be zeroed out, but only when the sum is positive. Let all
pollutant flows be conserved quantities, and let any dissipation of
pollutants due to natural processes be accounted for by the natural
pollutant inputs.

Let there exist at most two nodes immediately upstream of any node,
one being the next upstream node on the same stream and the other the
last node on the tributary that merges immediately upstream of the
current node. Call nodes having two nodes immediately upstream as
"confluence" nodes. Note that level 1 streams cannot have confluence
nodes, and the first (most upstream) node on any stream cannot be a
confluence node because we can regard the merging tributary and the
receiving stream as one single stream.

Define

\[ p_{1j} = \begin{cases} 
\text{k if node k is immediately upstream of node j on the same stream} \\
0 \text{ if node j is the most upstream node on a stream} 
\end{cases} \]

and \[ p_{2j} = \begin{cases} 
\text{k if node k is the most downstream node on the merging tributary} \\
0 \text{ if j is not a confluence node.} 
\end{cases} \]
Also let

\[ w_{ij}^n = \text{natural pollutant flow in kilograms per hour occurring between node } j \text{ and its next upstream node or nodes for worst-case } i; \text{ if } j \text{ is the first node on a stream, the flow is the aggregate from upstream}, \]

and \( w_{ij}^* = \text{pollutant flow in kilograms per hour past node } j \text{ for worst-case } i \text{ per the decisions taken at node } j \text{ and upstream nodes.} \)

Then \[ w_{ij} = w_{ip1j}^* + w_{ip2j}^* + w_{ij}^n + w_{ij}^e \]

for each \( i = 1 \) to \( I_h \) and \( j = 1, \ldots, J \) but \( j \notin J^t \) \hspace{1cm} (2.10)

and \[ w_{ij}^* = (1-x_j^z)(w_{ip1j}^* + w_{ip2j}^* + w_{ij}^n + w_{ij}^e) \]

+ \( x_j^z L[(w_{ip1j}^* + w_{ip2j}^* + w_{ij}^n + w_{ij}^e)] \)

for each \( i = 1 \) to \( I_h \) and each \( j \in J^t \) \hspace{1cm} (2.11)

where \( w_{10}^* = 0 \)

and \( L \) is a clip function which specifies the instream treatment action, i.e.,

\[ L(x) = \begin{cases} 0 & \text{if } x > 0 \\ x & \text{if } x < 0. \end{cases} \]

Note that instream treatment zeroes out positive flows but will not affect negative flows.
Average Annual Pollutant Loads

Average annual pollutant loads are considered analogous to the worst-case pollutant flows. Let

\[ a_j = \text{average annual pollutant load in kilograms via the abandoned or active mine effluent to node } j \text{ when no control action is taken}, \]

\[ a^a_j = \text{average annual pollutant load in kilograms via the abandoned or active mine effluent to node } j \text{ after abatement}, \]

\[ a^s_j = \text{average annual pollutant load in kilograms via the potential surface mine effluent to node } j \text{ when no control action is taken}, \]

\[ a^{sa}_j = \text{average annual pollutant load in kilograms via the potential surface mine effluent to node } j \text{ after abatement}, \]

\[ a^d_j = \text{average annual pollutant load in kilograms via the potential deep mine effluent to node } j \text{ when no control action is taken}, \]

\[ a^{da}_j = \text{average annual pollutant load in kilograms via the potential deep mine effluent to node } j \text{ after abatement}, \]

\[ a^e_j = \text{average annual pollutant load in kilograms via the mine effluent to node } j \text{ as per the decision taken at the source}, \]

\[ a^n_j = \text{average annual natural pollutant load in kilograms between node } j \text{ and its next upstream nodes; if } j \text{ is the first node on a stream, the load is the aggregate from upstream, and} \]
\( a_j^* = \text{average annual pollutant load in kilograms flowing past node } j \text{ per the decisions at node } j \text{ and upstream nodes.} \)

Then

\[
a_j^e = a_j - (a_j - a_j^a)x_j^a - a_j^t - a_j^at \quad \text{for each } j \in J^a \tag{2.12}
\]

\[
a_j^e = a_j^s + a_j x_j^s + a_j^d x_j^d + a_j^d x_j^d \quad \text{for each } j \in J^p \tag{2.13}
\]

\[
a_j^* = a_j^p + a_j^s + a_j^d + a_j^e
\]

for each \( j=1,\ldots,J \) and \( j \notin J^t \tag{2.14} \)

and

\[
a_j^* = (1 - x_j^z)(a_j^p + a_j^s + a_j^d + a_j^e)
\]
\[
+ x_j^z L[(a_j^p + a_j^s + a_j^d + a_j^e)]
\]

for each \( j \in J^t. \tag{2.15} \)

where \( a_0^* = 0. \)

**Costs, Revenues, and the Criterion Function**

Recall there are the mining costs and revenues, and the abatement and treatment costs, which are all in equivalent annual dollars. Let

\[
c_m^s = \text{cost minus revenue (negative profit) from surface mining at multiple effluent potential site } m
\]
\[
c_m^d = \text{cost minus revenue (negative profit) from deep mining at multiple effluent potential site } m
\]
\[c^s_j = \text{cost minus revenue (negative profit) from surface mining at the single effluent site feeding node } j\]
\[c^d_j = \text{cost minus revenue (negative profit) from deep mining at the single effluent site feeding node } j\]
\[c^{sa}_{bm} = \text{cost of abatement at level } b \text{ at multiple effluent potential mine site } m \text{, after the site is surface mined}\]
\[c^{da}_{bm} = \text{cost of abatement at level } b \text{ at multiple effluent potential mine site } m \text{, after the site is deep mined}\]
\[c^a_{bm} = \text{cost of abatement at level } b \text{ at multiple effluent abandoned or active mine } m\]
\[c^{sa}_j = \text{cost of abatement at the single effluent potential mine site feeding node } j \text{, after the site is surface mined}\]
\[c^{da}_j = \text{cost of abatement at the single effluent potential mine site feeding node } j \text{, after the site is deep mined}\]
\[c^a_j = \text{cost of abatement at the single effluent abandoned or active mine feeding node } j\]
\[c^{st}_j = \text{fixed cost of treating the potential surface mine effluent entering node } j\]
\[c^{dt}_j = \text{fixed cost of treating the potential deep mine effluent entering node } j\]
\[c^t_j = \text{fixed cost of treating the abandoned or active mine effluent entering node } j\]
\[c^z_j = \text{fixed cost of performing instream treatment at node } j\]
\[c^v = \text{variable cost of treating one unit of pollution in dollars per kilogram of acid}\]
\[c^e_j = \text{cost incurred at the single effluent mine feeding node } j \text{ per the decision taken at the mine, or the cost incurred per the treatment decision for the effluent to node } j \text{ when } j \text{ is fed by a multiple effluent mine}\]

and \[c^m = \text{cost incurred per the mining and(or) abatement decisions at multiple effluent mine } m.\]
Then

\[ c_j = c_j^a x_j + (c_j^t + c_j^v) x_j^t + (c_j^a + c_j^t + c_j^v) x_j^a \]

for each \( j \in J^a \) but \( j \notin J^m \) for any \( m = 1 \) to \( M \) \hspace{1cm} (2.16)

\[ c_j = c_j^s x_j + (c_j^s + c_j^s^a) x_j^a + (c_j^s + c_j^s^t + c_j^s^v) x_j^{st} \]

\[ + (c_j^s + c_j^s^a + c_j^s^t + c_j^s^v) x_j^{sat} + c_j^{d_s} x_j + (c_j^{d_s} + c_j^{d_s}) x_j^{dat} \]

\[ + (c_j^{d_s} + c_j^{d_s}) x_j^{dt} + (c_j^{d_s} + c_j^{d_s}) x_j^{dat} \]

for each \( j \in J^p \) but \( j \notin J^m \) for any \( m = 1 \) to \( M \) \hspace{1cm} (2.17)

\[ c_j = (c_j^t + c_j^v) x_j^t + (c_j^t + c_j^v) x_j^a \]

for each \( j \in J^m \) and \( m \in M^a \) \hspace{1cm} (2.18)

\[ c_j = (c_j^{st} + c_j^{st}) x_j^{st} + (c_j^{st} + c_j^{st}) x_j^{sat} \]

\[ + (c_j^{dt} + c_j^{dt}) x_j^{dt} + (c_j^{dt} + c_j^{dt}) x_j^{dat} \]

for each \( j \in J^m \) and \( m \in M^p \) \hspace{1cm} (2.19)

\[ c_m = c_m^{1m} z_m^{1m} + \ldots + c_m^{z_m} z_m^{z_m} \]

for each \( m \in M^a \) \hspace{1cm} (2.20)

\[ c_m = c_m^s y_m^s + c_m^s x_m^s + \ldots + c_m^s z_m^s \]

\[ + c_m^{d_d} y_m^{d_d} + c_m^{d_d} y_m^{d_d} + \ldots + c_m^{d_d} y_m^{d_d} \]

for each \( m \in M^p \) \hspace{1cm} (2.21)
The criterion function can now be stated as

\[ C^T = \sum_{j=1}^{J} c^e_j + \sum_{m=1}^{M} c_m + \sum_{j \in J^t} (c^z_j - c^v_L[-(a^*_p_1j + a^*_p_2j + a^n_j + a^e_j)]x^z_j) \]  \hspace{1cm} (2.22)

where \( C^T \) is the total cost to be minimized and \( L \) is the clip function introduced earlier.

**Quality Constraints**

Let

\[ q^s_j = \text{pollution concentration in ppm that should not be exceeded at node } j \]

\[ q^{ij}_j = \text{stream flow exclusive of pollutant in cubic meters per second at node } j \text{ for worst-case } i \]

\[ f = \text{conversion factor from cubic meters per second to kilograms per hour} \]

Then the quality constraints can be expressed as

\[ w^*_{ij} \leq q^{ij}_j = f q^{ij}_j (q^{s10^{-6}}/(1 - q^{s10^{-6}})) \]

for each \( i = 1 \) to \( I_{h_j} \) and \( j = 1 \) to \( J \) \hspace{1cm} (2.23)

where \( w^*_{ij} \) is given by equation (2.10) or (2.11).
The Minimum Cost Problem

The minimum cost problem can be stated as: minimize $C^T$ given by equation (2.22), subject to the constraints given by (2.1) to (2.7), and (2.23). Equations (2.12) to (2.21) support equation (2.22) and equations (2.8) to (2.11) support (2.23).

It is clear that this problem is an INLP, i.e., an integer nonlinear program, where the nonlinearities arise only from the clipping effect of instream treatment specified in equations (2.11), (2.15) and (2.22). It is also a special case of INLP's in that the decision variables are all binary. Further, the recurring worst-case pollution flow terms in the quality constraints constitute a special structure.

There are basically two different approaches which may be used to arrive at the minimum cost solution, if one exists. One is the "cutting plane" approach, and the other is the "enumeration" approach. The idea in the cutting plane method is to introduce a succession of linear cuts to the continuous feasible space of the minimum cost problem until the reduced feasible space has an integral extreme point that is optimal. This method requires the feasible region to be convex at least in the neighborhood of the eventual extreme point. It may be possible that this method applies to the problem at hand; however, such an approach seems unattractive because the method is generally slow to converge and its construction and implementation can be very difficult.
The second approach, the enumeration approach, proves optimality through sheer enumeration. The ideas motivating this approach fall under various labels such as "branch and bound" and "relaxation and restriction", all of which essentially permit the fathoming of most of the total enumeration space. In the author's judgement, enumeration is the only promising approach for constructing optimization algorithms for the minimum cost problem.

The coming chapters present and compare the performances of three promising enumeration algorithms. One is an extension of the Lawler-Bell algorithm [10], another uses Lagrangian relaxation, and the third is based on a linear formulation of the minimum cost problem and uses LP relaxation. They all vary in spirit and implementation. We shall now briefly review the literature on the above three enumerative approaches.

**Literature Review**

The algorithm of Chapter III is based mainly on Clark's ideas in his exploratory work [15] where he formulates and solves the problem of cost-effective resource allocation to control mine drainage pollution. These ideas of Clark, while general in scope, require certain special problem structures for successful algorithmic construction, and are particularly suited to the minimum cost problem at hand. In his pioneering work, Clark considers a simpler problem than the minimum cost problem we have here. That is, he only considers single effluent abandoned mine sites polluting streams in a single watershed. At each
such site, he considers the decision to abate or not to abate and the
decision to treat or not to treat the site's effluent. He also
considers the decision to treat or not to treat in the stream channel at
sites that are likely candidates for instream treatment. His criterion
is to minimize pollution control costs and his constraints comprise the
achievement of the quality standard at each stream node for a single
worst-case situation. This problem being a INLP, he uses the Lawler-
Bell algorithm [10] designed for any general binary program. However,
because the Lawler-Bell algorithm is slow to converge for medium and
large problems, he improves its efficiency by incorporating several
extensions and modifications. First, he pools the abatement and
treatment decision variables at each abandoned mine site into a single
integer variable having at most three integer values (he eliminates
redundant alternatives). Second, he orders these integer variables and
the instream treatment binary decision variables in a single vector so
as to yield the lexicographic numerical ordering of solution vectors, as
required in the Lawler-Bell algorithm. Third, he provides a procedure
for finding the next feasible solution vector beyond any given vector in
the lexicographic ordering of vectors. Fourth, he decomposes the
problem at each active instream treatment site whenever the
corresponding instream treatment variable takes the value 1 during
enumeration. This decomposition is possible because the pollution
interaction between mine sources upstream of an active instream
processor and those that pollute the remainder of the basin is nil. His
last extension is, he computes a vector he calls the "stopping vector."
Any vector beyond the stopping vector in the lexicographic ordering of
vectors is nonoptimal. He thus obviates the search beyond the stopping vector.

Clark [15] reports good computational times on an IBM 370 Computer for a hypothetical problem involving more than a hundred binary variables. He also concludes that the minimum cost solution is sensitive to the worst-case situation analyzed. So he recommends the simultaneous analysis of multiple worst-case situations.

In a subsequent effort, Clark and Ranganathan [1] extend the Clark algorithm to consider multiple worst-case situations. Here, they also implement a branch and bound method in conjunction with the next feasible solution procedure for fathoming less optimal vectors. They report good computational times on an IBM 370 Computer for a hypothetical problem involving more than a hundred binary variables.

The most recent effort by Clark and Ranganathan [2] involves further extensions and modifications to the Clark algorithm. This recent algorithm solves the minimum cost problem at hand. Clark and Ranganathan [3] have also applied this algorithm to a real life case depicting the mine pollution situation near Elkins, West Virginia, and have reported a CPU time of about 6.5 minutes on the AMDAHL 470 V.8 Computer. Chapter III describes this algorithm but in addition includes two improvements the author has made more recently. One improvement is a Dynamic Programming lower bound computation method and another makes the stopping vector determination more efficient.
A class of branch and bound algorithms that has acquired prominence in the Seventies for solving specially structured integer programming problems involves the use of Lagrangian relaxation for the calculation of bounds. General frameworks for this class of algorithms have been discussed by a number of authors including Geoffrion [6], Fisher [5] and Shapiro [14]. In many applications, the given integer programming problem contains a set of easy constraints complicated by a set of side constraints. Dualizing the side constraints produces a Lagrangian problem that is relatively easy to solve and whose optimal value is a lower bound (for minimization problems) on the optimal value of the original problem. Fisher [5] reports a number of successes with the Lagrangian method for a variety of integer programming problems. We now briefly discuss the Lagrangian problem in the context of a general discrete programming problem, and outline some algorithms available in the literature for solving this problem.

Consider a general discrete program, say \((P)\), which can be, for all practical purposes, stated as follows:

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad g(x) \leq 0 \\
& \quad h(x) = 0 \\
& \quad x \in X = \{x^t, t=1,\ldots,T\} \subseteq \mathbb{R}^n
\end{align*}
\]

(\(P\))

where \(X\) is a finite set containing points in \(\mathbb{R}^n\), \(f\) is a numerical function, \(g\) and \(h\) are each a vector of numerical functions, and \(f, g\) and \(h\) are assumed to be finite over \(X\). The set \(X\) may be thought of as being
defined by some simple constraints, for example, \( x \) is a vector of binary variables. And the constraints \( g(x) \) and \( h(x) \) are the hard constraints.

An intuitive approach to try to avoid the difficult constraints is to use the construction

\[
\min_{x \in X} (f(x) + ug(x) + rh(x)) \quad (LR_{ur})
\]

where \( u \) and \( r \) are conformable vectors of Lagrangian multipliers, but \( u \) in addition is required to be nonnegative. We call \((LR_{ur})\) the Lagrangian problem. Letting \( v(.) \) denote the optimal value of problem \((.)\), it is well known that

\[
v(LR_{ur}) \leq v(P) \text{ for all } u \geq 0 \text{ and all } r.
\]

This is easily verified. Let \( S \) denote the set of feasible solutions to \((P)\). Then

\[
v(LR_{ur}) = \min_{x \in X} (f(x) + ug(x) + rh(x)) \leq \min_{x \in S} (f(x) + ug(x) + rh(x)) \leq \min_{x \in S} (f(x)) = v(P)
\]

where the left-most inequality is due to \( S \subseteq X \) and the right-most inequality is because \( u \geq 0 \) and \( ug(x) + rh(x) \leq 0 \) for all \( x \in S \). Hence, for any \( u \geq 0 \) and any \( r \), \( v(LR_{ur}) \) is a lower bound for \( v(P) \) and thus serves in a branch and bound algorithm for \((P)\). The solution procedure for \((LR_{ur})\) is generally easy but dependent on the particular problem structure. So, it is left to the concerned analyst to devise his particular technique for solving \((LR_{ur})\). However, the problem of maximizing \( v(LR_{ur}) \) over \( u \) and \( r \) possesses a uniform structure for the general
discrete program. Hence, many works in the literature have concentrated on this problem. A solution or near optimal solution to this problem is important because it provides a tight bound on \( v(P) \) thus leading to good fathoming efficiency in a branch and bound algorithm or it can lead to good heuristics for \( (P) \). This problem is customarily referred to as the dual of \( (P) \) and can be expressed as the following linear program with many constraints:

\[
\begin{align*}
\max_{u,r} & \quad w \\
\text{s.t.} & \quad w \leq f(x_t^*) + u g(x_t^*) + r h(x_t^*) \quad \text{for } t = 1, \ldots, T \\
& \quad u \geq 0
\end{align*}
\]

(D)

Note that for any given \( u \geq 0 \) and \( r \), solving \( (LR_{ur}) \) yields the tightest R.H.S. (Right Hand Side) on \( w \) among all the R.H.S.'s 1 through \( T \) shown in (D). Thus, \( v(LR_{ur}) \) relative to \( u \) and \( r \) is actually the upper envelope formed from the intersections of the half spaces 1 to \( T \) in (D). Therefore, maximizing \( v(LR_{ur}) \) over \( u \) and \( r \) translates to finding the upper most point on this envelope. In particular, the piece-wise linear and concave properties of \( v(LR_{ur}) \) relative to \( u \) and \( r \) have made life easy for the construction of hill climbing algorithms in practice. In the literature, these algorithms fall in three categories. One is a group of algorithms based on "subgradient optimization", the second uses "generalized linear programming" and the third comprises "multiplier adjustment techniques".

We see that if \( x^k \), for some \( k \) between 1 and \( T \), is optimal in \( (LR_{0f}) \), then the hyperplane defined by
\[ w = f(x^k) + ug(x^k) + rh(x^k) \]

meets \( v(LR_{ur}) \) at \( u=\bar{u} \) and \( r=\bar{r} \) and generally lies above \( v(LR_{ur}) \) in the \((w,u,r)\) space. Hence

\[ v(LR_{ur}) \leq f(x^k) + ug(x^k) + rh(x^k) \]

and \[ v(LR_{uf}) = f(x^k) + \bar{u}g(x^k) + \bar{r}h(x^k) \]

Or, subtracting the second relation from the first, we obtain

\[ v(LR_{ur}) \leq v(LR_{uf}) + g(x^k)(u-\bar{u}) + h(x^k)(r-\bar{r}) \]

This last relation implies that \( g(x^k) \) and \( h(x^k) \) are subgradients of \( v(LR_{ur}) \) at \( u=\bar{u} \) and \( r=\bar{r} \) respectively.

The subgradient method for solving (D) is a brazen adaptation of the gradient method where gradients are replaced by subgradients. Given an initial value \( u^0 \geq 0 \) and an initial value \( r^0 \), a sequence \( \{u^k, r^k\} \) is generated from the rules

\[ u^{k+1} = \max \{0, u^k + t_k g(x^k)\} \]
\[ r^{k+1} = r^k + t_k h(x^k) \]

where \( x^k \) is optimal in \( LR_{ur} \) and \( t_k \) is a positive scalar step size. Because the subgradient method is easy to program and has worked well on many practical problems, it has become the most popular method for (D). Computational performance and theoretical convergence properties of the subgradient method are discussed by Held, et al. [8]. The fundamental theoretical result is that \( v(LR_{ur} k, k) \) tends to \( v(D) \) if \( t_k \) tends to zero and if the sum \( t_i \) over \( i \) from 0 to \( k \) tends to infinity. The step size used most commonly in practice is
\[
\frac{\lambda_k (v - v(LR_u r_k k))}{\|g(x^k)\|^2 \|h(x^k)\|^2}
\]

where \(\lambda_k\) is a scalar satisfying \(0 < \lambda_k \leq 2\) and \(v\) is an upper bound on \(v(D)\) frequently obtained by applying a heuristic to \((P)\). Justification of this formula is given in [8]. Often the sequence \(\{\lambda_k\}\) is determined by setting \(\lambda_0 = 2\) and halving \(\lambda_k\) whenever \(v(LR_{ur})\) has failed to increase in some fixed number of iterations. This rule has performed well, even though it is not guaranteed to satisfy the sufficient condition given above for optimal convergence.

Unless we obtain a \(u^k\) and a \(r^k\) for which \(v(LR_{u r k k})\) equals the cost of a known feasible solution to \((P)\), there is no way of proving the optimality of \((D)\) with the subgradient method. To resolve this difficulty, the method is usually terminated after reaching an arbitrary iteration limit. Its final solution may then be used in a branch and bound algorithm or as a starting point for a heuristic procedure for a feasible solution to \((P)\).

The second class of algorithms for \((D)\) is called "generalized linear programming." In fact, generalized linear programming is an approach with a long history, initially constructed to decompose and solve large scale linear programs (see Shapiro [14]). Its use for solving \((D)\) has been quite recent (see Fisher [5]). The basic idea here is to use the dual simplex method or the primal-dual simplex method for solving \((D)\). However, the constraints 1 to T are not known apriori. So the idea is
to generate the constraints 1 to T from one iteration to the next one constraint at a time until an iteration is reached when it is proved that all remaining (not generated) constraints in 1 to T are redundant (nonbinding) at the optimum of (D), so that the solution to (D) at this iteration is optimal in (D). To elucidate, suppose at the end of iteration k-1, we have generated the constraints 1 to k-1 corresponding to the points \(x^1, \ldots, x^{k-1}\). Then at iteration k, we solve (D) ignoring all other constraints. Let \(u^k\) and \(r^k\) be optimal in (D) at this iteration. Then for this \(u^k\) and \(r^k\), we solve (LR) to obtain \(x^k\) that is optimal in (LR). This \(x^k\) produces the new constraint for iteration k+1. Eventually, an iteration t will be reached when \(u^{t-1} = u^t\) and \(r^{t-1} = r^t\). This condition implies \(u^t\) and \(r^t\) solve (D) in general [14].

Generally, the simplex-based methods are harder to program and have not performed quite so well computationally as the subgradient method. But they prove the optimality of (D). Fisher [5] recommends they be used with a starting point determined by the subgradient method. But this recommendation carries value only when we are seeking good heuristics for (P). For in a branch and bound algorithm, the additional expense due to the application of such a method will not be worth the price in general.

The third approach, multiplier adjustment methods, are specialized algorithms for (D) that exploit the structure of a particular application. In these methods, a sequence \(\{u^k, r^k\}\) is generated by the rules

\[
u^{k+1} = u^k + t_k d_k^1
\]

\[
r^{k+1} = r^k + d_k^2
\]

where \(t_k\) is a positive scalar and \(d_k^1\)
and $d^2_k$ are rectional changes for $u$ and $r$ respectively. To determine $d^1_k$ and $d^2_k$, a small set of primitive directions $S_1$ is defined apriori for $u$ and likewise a set $S_2$ for $r$, for which it is easy to evaluate the directional derivative of $v(LR_{ur})$. At any given iteration, the directions in $S_1$ and $S_2$ are scanned in fixed order, and $d^1_k$ and $d^2_k$ are taken to be either the first directions found along which $v(LR_{ur})$ increases or directions of steepest ascent in $S_1$ and $S_2$ respectively. The step size $t_k$ is chosen either to maximize $v(LR_{ur})$ along the direction $(d^1_k, d^2_k)$ or to take us to the first point at which the directional derivative of $v(LR_{ur})$ changes. If $S_1$ and $S_2$ contain no improving directions, we terminate.

Successful implementation of the multiplier adjustment method requires artful specifications of $S_1$ and $S_2$. Fisher [5] reports some practitioners have had success with the multiplier adjustment method. On the other hand, he cites other implementations where the performance has been poorer than the subgradient method.

A research area of current interest is Lagrangian heuristics. The aim of these heuristics is to move the Lagrangian solution that is often infeasible (but nearly feasible) to (P) to a good feasible solution to (P). However, most heuristics to date are application specific, and no general technique is available.

Chapter IV presents a branch and bound algorithm for solving the minimum cost problem. This algorithm uses the Lagrangian method for calculating lower bounds and a Lagrangian heuristic for generating
feasible solutions. Except for the subgradient method, all other techniques presented in the chapter are entirely new.

Besides the Lawler-Bell and Lagrangian approaches, another promising approach for the minimum cost problem, branch and bound methods using linear programming relaxation, are methods with a long history. These methods apply to the minimum cost problem because the minimum cost problem can be reformulated as an integer linear program, which we do in Chapter V. The advantage of these methods is they apply to any mixed integer linear program, and their applications have grown largely because of the availability of many standard codes including the IBM MPSX/370. Their major disadvantage is they involve many simplex iterations which can be very time consuming and which can lead to unacceptable numerical inaccuracies. Reducing these inaccuracies to tolerable levels requires the reinversion of the LP basis, which again consumes significant computational time. In addition, reinversions of the LP bases may be necessary when nodes in the enumeration tree are backtracked. Researchers in the past have suggested various techniques to minimize these disadvantages. The list of papers is long and it is not the intent of this dissertation to discuss the various issues involved. Rather, the intent is to use a standard code and evaluate its performance against the two alternative approaches of interest. In particular, we use a code written by Gonsalvez [7] which we describe briefly in Chapter V.
CHAPTER III
MODIFICATIONS AND EXTENSIONS
TO THE LAWLER-BELL ALGORITHM

This chapter presents an optimization algorithm for the minimum cost resource allocation problem based on an algorithm developed by Lawler and Bell [10] for binary nonlinear programs. For the most part, the algorithm was developed in stages under two E.P.A. contracts [15,12], but the author has made two improvements since. The algorithm will be referred to by ALCOT (Acid Least Cost) hereafter.

The primary motive for ALCOT is it is additive unlike the algorithms of Chapters IV and V. Secondly, it lends itself to the use of dynamic programming on two occasions. Thirdly, special purpose devices are appended easily. And lastly, it is easy to implement. The drawback of ALCOT is that it is not global in its outlook as are the algorithms of the next two chapters. This drawback comes from the apriori definition of the enumeration tree. In any case, only computational experience can demonstrate the superior algorithm.
THE LAWLER-BELL ALGORITHM

This algorithm is designed to solve any general binary nonlinear program which may be expressed as:

Minimize \( g_0(X) \)

Subject to \( g_{11}(X) - g_{12}(X) \geq 0 \)
\( g_{21}(X) - g_{22}(X) \geq 0 \)
\[ \vdots \]
\( g_{m1}(X) - g_{m2}(X) \geq 0 \)

where \( X = (x_1, x_2, \ldots, x_n) \), \( x_j \) for any \( j = 1 \) to \( n \) is binary, and \( g_0(X), g_{11}(X), \ldots, g_{m2}(X) \) are each monotone nondecreasing in each \( x_1 \) to \( x_n \).

The algorithm is enumerative in that each possible vector \( X \) is considered explicitly or implicitly to determine the optimal solution vector.

A lexicographic or numerical ordering of vectors is first defined by giving each vector \( X \) the value

\[ n(x) = x_1 2^{n-1} + x_2 2^{n-2} + \ldots + x_n 2^0. \]

In addition, a vector partial ordering is obtained by skipping from a current vector \( X \) to a vector \( X^* \) under certain conditions, where \( X^* \) is the first vector beyond \( X \) in the ordering of vectors such that

\[ X \not\leq X^*. \]

This implies that each vector in
\{Y | n(Y) = n(X) + 1, n(Y) = n(X) + 2, \ldots, n(Y) = n(X^*) - 1 \}

satisfies the condition

\[ Y \geq X \text{ or } y_j \geq x_j \text{ for each } j = 1 \text{ to } n. \]

Lawler and Bell show that \( X^* \) can be obtained from \( X \) readily on a digital computer by

1. regarding \( X \) as binary,
2. subtracting 1 from \( X \),
3. logically "or" \( X \) and \( X - 1 \) to obtain \( X^* - 1 \), and
4. adding 1 to obtain \( X^* \).

The procedure described by Lawler and Bell to identify the optimal solution involves proceeding through the list of possible solutions and keeping a record of the least costly solution encountered during the search. Let \( \hat{X} \) denote this least costly solution and \( \hat{g}_0(\hat{X}) \) the current least cost. The procedure starts with \( X = (0,0,\ldots,0) \) and ends when \( X = (1,1,\ldots,1) \). Letting \( X \) denote the vector that is currently being examined, the following rules indicate which vector is examined after \( X \):

1. If \( g_0(X) \geq \hat{g}_0(\hat{X}) \), skip to \( X^* \). Since \( X^* \) is the first vector in the numerical order following \( X \) where \( X \not\leq X^* \), then each \( Y \) in
\[ \{Y | n(Y) = n(X) + 1, n(Y) = n(X) + 2, \ldots, n(Y) = n(X^*) - 1 \} \]
is greater than or equal to \( X \). Also because \( g_0(X) \) is monotone nondecreasing in \( X \),
\[ g_0(Y) \geq g_0(X) \geq \hat{g}_0(\hat{X}) \text{ for each indicated } Y. \]

2. If \( X \) is a feasible solution and \( g_0(X) < \hat{g}_0(\hat{X}) \), set \( \hat{X} = X \), and skip to \( X^* \) for the same reasons explained in step 1.
3. If $X$ is infeasible, if $g_0(X) < g_0(\hat{X})$ and if $g_{11}(X^{*-1}) - g_{12}(X) \not\geq 0$ for any $i = 1$ to $m$, skip to $X^*$. Note that $Y = X^{*-1}$ maximizes $g_{11}(Y)$ and $Y = X$ minimizes $g_{12}(Y)$ among the vectors between $X$ and $X^{*-1}$. Thus when $g_{11}(X^{*-1}) - g_{12}(X) \not\geq 0$, then no vector $Y$ between $X$ and $X^{*-1}$ can satisfy constraint $i$.

4. Skip to $Y$ where $n(Y) = n(X) + 1$ when rules 1 to 3 above do not apply.

Lawler and Bell give representative computer times to solve typical problems to illustrate the potential efficiency of their method. Using the procedure listed above, they solved problems involving as many as 30 binary variables with computer times ranging from 10 to 20 minutes on an IBM 7090. However, current generation computers are considerably faster than an IBM 7090. Although the increase in computer time for increasing numbers of variables appears to be less than an exponential function, this increase is faster than a linear function. Thus, there appears to be an upper limit on the number of nodes that can be analysed in a single problem, and it is important to select an algorithm that is rapid. With emphasis on this efficiency objective, the algorithm ALCOT was constructed involving some modifications and several extensions to the Lawler-Bell algorithm.

OVERVIEW OF ALGORITHM ALCOT

The value of the Lawler-Bell algorithm lies in the skip from $X$ to $X^*$ when the vectors between $X$ and $X^*$ are determined implicitly as being too
costly or infeasible. ALCOT uses this feature for skipping the too costly vectors. That is, the binary decision variables $x^a_j$, $x^c_j$ and $x^{at}_j$ subject to the constraint (2.1) at each $j \in J^a$ are pooled into a single integer variable $x^e_j = (0, 1, 2, 3)$ such that $x^e_j = 0$ is the least cost decision, $x^e_j = 1$ the next higher cost decision, and so on. Likewise, the binary decision variables $x^s_j$, $x^{sa}_j$, $x^{st}_j$, $x^{st}_j$, $x^d_j$, $x^{da}_j$, $x^{dt}_j$ and $x^{dat}_j$ subject to the constraint (2.2) at each $j \in J^d$ are pooled into a single integer variable $x^z_j = (0, 1, \ldots, 8)$ such that $x^z_j = 0$ is the least cost decision, $x^z_j = 1$ the next higher cost decision, and so on. The definition $x^e_j$ at each $j \in J^a$ is preserved since the criterion function is already monotone nondecreasing in each $x^e_j$. Next the decision variables $x^e_j$ for $j = 1$ to $J$ and $x^z_j$ for each $j \in J^d$ are ordered in a vector $X = (x^e_1, x^e_2, \ldots, x^e_n)$. Note that a decision variable $x_k$ may take on as many as nine values as opposed to zero and one in the Lawler-Bell algorithm. Nevertheless, the lexicographic ordering of $X$ is obtained readily. That is, $x_n$ will cycle through all its values for each $(x_1, x_2, \ldots, x_{n-1})$, $x_{n-1}$ will cycle through all its values for each $(x_1, x_2, \ldots, x_{n-2})$, and so on.

A simple and rapid procedure determines the next feasible vector beyond any given $X$ in the lexicographic ordering of vectors. This is clearly an improvement over step 3 of the Lawler-Bell algorithm. Further it eliminates step 4 of the Lawler-Bell algorithm. Note that step 4 increases the computational effort of the Lawler-Bell algorithm.

The next feasible solution procedure incorporates the lexicographic ordering of vectors by maximizing the potential for feasibility of the higher hierarchical variables in the vector $X$ in relationship to their
counterparts, the lower hierarchical variables. This scheme naturally produces the next feasible solution vector. $x_1$ is the highest hierarchical variable, which is considered first, then $x_2$, and so on until $x_n$. The potential for feasibility of $x_1$ is maximized by assuming zero pollutant output in the mine effluents associated with each source decision in $x_2$ through $x_n$, and eliminating positive pollutant flow in the stream channel at sites associated with each instream treatment decision variable in $x_2$ through $x_n$. The given value of $x_1$ is retained if it achieves the quality standard at its node and all downstream nodes. Otherwise $x_1$ is increased to the next higher value that satisfies the quality requirements, and $x_2$ to $x_n$ are each set to zero in accordance with the lexicographic ordering of vectors. Similarly the value of $x_2$ is examined for feasibility assuming minimum pollutant output decisions in $x_3$ to $x_n$ but taking into account the pollutant flow due to the feasible value $x_1$ and any multiple effluent mine constraint (equations 2.3 to 2.7) that $x_1$ imposes on $x_2$. If the given value of $x_2$ is feasible, the value is retained; otherwise $x_2$ is increased to the next higher value that is feasible and $x_3$ to $x_n$ are each set to zero. This scheme is continued with $x_3$ to $x_n$ and characterizes the next feasible solution procedure.

In conjunction with the next feasible solution procedure, the calculation of a lower bound on the total systems cost provides the potential for fathoming vectors which are more expensive than the incumbent feasible vector. Each time a decision variable is made feasible, this lower bound is calculated. Specifically, once $x_k$ has been made feasible, then by regarding
1. \((x_1, x_2, \ldots, x_k)\) as fixed, and
2. \((x_{k+1}, x_{k+2}, \ldots, x_n)\) as free,

a lower bound on the total systems cost is calculated and compared against the currently known minimum cost. If this lower bound is not less than the current minimum, then each \((x_{k+1}, x_{k+2}, \ldots, x_n)\) has been fathomed; consequently, \(x_{k+1}\) through \(x_n\) are all set to their maximum values and the next feasible solution procedure is halted. This constitutes the application of the branch and bound concept in ALCOT.

Note that if the next feasible solution procedure in conjunction with the branch and bound mechanism yields the next feasible solution, then this solution will be better than the previously held minimum cost solution.

Because many (but not all) decisions affecting the quality of streams in one watershed bear no effect on a different watershed's quality, the total vector \(X\) is decomposed into sub-vectors, each sub-vector constituting the decisions in a unique watershed, and each optimized under the quality constraints. Let \(X\) be written as \((X^H, X^{H-1}, \ldots, X^1)\), where \(X^H\) is the sub-vector representing the decisions in watershed \(H\), and so on. The lexicographic ordering dictates that \(X^1\) will cycle through all its values for each \((X^H, X^{H-1}, \ldots, X^2)\), \(X^2\) will cycle through all its values for each \((X^H, X^{H-1}, \ldots, X^3)\), and so on. Because the decision variables in \(X\) are defined at nodes rather than at the mine sources, each \((X^H, X^{H-1}, \ldots, X^2)\) can impose mining and (or) abatement restrictions upon decision variables defined at those nodes in watershed 1 that are denoted for multiple effluent mines draining into
watershed 1 and one or more of the higher numbered watersheds (see constraints 2.3 to 2.7). These imposed restrictions on \( x^1 \) due to \((x^H, x^{H-1}, \ldots, x^2)\) can be the same for several values of \((x^H, x^{H-1}, \ldots, x^2)\).

Likewise, several values of \((x^H, x^{H-1}, \ldots, x^3)\) will impose the same set of mining and (or) abatement restrictions upon \((x^2, x^1)\). In general, several \((x^H, x^{H-1}, \ldots, x^{h+1})\) will impose the same set of restrictions on \((x^h, x^{h-1}, \ldots, x^1)\). Consequently, the minimum solution \((x^h, x^{h-1}, \ldots, x^1)\) obtained for a set of restrictions on \((x^h, x^{h-1}, \ldots, x^1)\) or a lower bound on this solution's cost can be reimplemented each time \((x^h, x^{h-1}, \ldots, x^1)\) is subject to the same set of restrictions. This implementation is performed in the next feasible solution procedure after \((x^H, x^{H-1}, \ldots, x^{h+1})\) is made feasible. This characterizes the application of the "decomposition" principle in ALCOT.

The determination of the "stopping vector" is another improvement over the Lawler-Bell algorithm. Any vector beyond the stopping vector in the lexicographic ordering of vectors is more expensive than the stopping vector; consequently, the search beyond the stopping vector is obviated. The fact that a stream's capacity for carrying pollution within the quality standard increases as the stream flows downstream (due to flow volume increase) is taken advantage of in the determination of the stopping vector. The greater the variation in the pollution assimilative capacity of a stream as it flows downstream, the better will be the stopping vector.

In summary, each iteration of ALCOT consists of two basic steps: 1) the next feasible solution procedure in conjunction with the branch
and bound and decomposition principles, and 2) the skipping procedure for skipping the too costly vectors between \( X \) and \( X^* \). ALCOT begins with \( X=0 \) and terminates when \( X \) equals or exceeds the stopping vector.

**REDEFINITION OF THE SOURCE DECISION VARIABLES**

This section addresses the pooling of the binary source decision variables at each node into a single integer variable in order that the criterion function is monotone nondecreasing in the integer variable. Moreover, redundant decision alternatives at single effluent source nodes are eliminated.

Recall there are three binary variables subject to the constraint (2.1) at each abandoned or active mine node, constituting a total of four alternatives, and there are eight binary variables subject to the constraint (2.2) at each potential mine node, constituting nine alternatives. Define

\[
d_j = \begin{cases} 
0 & \text{if } x^a_j = x^t_j = x^{at}_j = 0 \\
1 & \text{if } x^a_j = 1 \\
2 & \text{if } x^t_j = 1 \\
3 & \text{if } x^{at}_j = 1 
\end{cases} \quad \text{at each } j \in J^a
\]
Also define

\[
d_j = \begin{cases} 
0 & \text{if } x_j^s = x_j^a = x_j^{st} = x_j^{sat} = x_j^d = x_j^d = x_j^t = 0 \\
1 & \text{if } x_j^s = 1 \\
2 & \text{if } x_j^a = 1 \\
3 & \text{if } x_j^{st} = 1 \\
4 & \text{if } x_j^{sat} = 1 \\
5 & \text{if } x_j^d = 1 \\
6 & \text{if } x_j^d = 1 \\
7 & \text{if } x_j^t = 1 \\
8 & \text{if } x_j^t = 1 
\end{cases} 
\]

at each \( j \in J \). 

Also define

\[
c_j^a = \frac{c_{bm}^a - c_{b-1,m}^a}{|J_{bm}|} 
\]

for each \( j \in J_{bm} \), \( b=1 \) to \( B_m \) and \( m \in M^a \) \hspace{1cm} (3.1)

\[
c_j^s = \frac{c_j^s}{|J_m^s|} \quad \text{for each } m \in M^s \hspace{1cm} (3.2)
\]

\[
c_j^d = \frac{c_j^d}{|J_m^d|} \quad \text{for each } m \in M^d \hspace{1cm} (3.3)
\]

\[
c_j^{sa} = \frac{c_j^{sa}}{|J_{bm}^s|} \quad \text{for each } j \in J_{bm} \text{, } b=1 \text{ to } B_m \text{ and } m \in M^s \hspace{1cm} (3.4)
\]

\[
c_j^{da} = \frac{c_j^{da}}{|J_{bm}^d|} \quad \text{for each } j \in J_{bm} \text{, } b=1 \text{ to } B_m \text{ and } m \in M^d \hspace{1cm} (3.5)
\]

where \( |J_{bm}| \) and \( |J_m^s| \) mean the cardinalities of the sets \( J_{bm} \) and \( J_m^s \) respectively. Note that equations (3.1) to (3.5) specify a uniform distribution of mining and (or) abatement costs at multiple effluent mines among their respective nodes, and thus obviate the need for equations (2.18) to (2.21); i.e., (2.16) and (2.17) are now sufficient to express the costs incurred at all mine sources.
Next rewrite equations (2.8), (2.9), (2.12), (2.13), (2.16), (2.17) and (2.22) respectively as follows:

\[
\begin{align*}
\hat{w}_{ij}^e &= \begin{cases} 
  w_{ij} & \text{if } d_j = 0 \\
  a_j & \text{if } d_j = 1 \\
  0.0 & \text{if } d_j = 2 \text{ or } 3 
\end{cases} \\
\text{for each } i = 1 \text{ to } I_h \text{ and } j \in J^a
\end{align*}
\]  

(3.6)

\[
\begin{align*}
\hat{w}_{ij}^e &= \begin{cases} 
  w_{ij}^s & \text{if } d_j = 1 \\
  w_{ij}^a & \text{if } d_j = 2 \\
  w_{ij}^d & \text{if } d_j = 5 \\
  w_{ij}^{da} & \text{if } d_j = 6 \\
  0.0 & \text{if } d_j = 0, 3, 4, 7 \text{ or } 8 
\end{cases} \\
\text{for each } i = 1 \text{ to } I_h \text{ and } j \in J^p
\end{align*}
\]  

(3.7)

\[
\begin{align*}
\hat{a}_j &= \begin{cases} 
  a_j & \text{if } d_j = 0 \\
  a_j & \text{if } d_j = 1 \\
  0.0 & \text{if } d_j = 2 \text{ or } 3 
\end{cases} \\
\text{at each } j \in J^a
\end{align*}
\]  

(3.8)

\[
\begin{align*}
\hat{a}_j &= \begin{cases} 
  a_j & \text{if } d_j = 1 \\
  a_j & \text{if } d_j = 2 \\
  a_j & \text{if } d_j = 5 \\
  a_j & \text{if } d_j = 6 \\
  0.0 & \text{if } d_j = 0, 3, 4, 7 \text{ or } 8 
\end{cases} \\
\text{at each } j \in J^p
\end{align*}
\]  

(3.9)

\[
\begin{align*}
\hat{c}_j &= \begin{cases} 
  0.0 & \text{if } d_j = 0 \\
  a_j & \text{if } d_j = 1 \\
  c_j + c^a_j & \text{if } d_j = 2 \\
  c_j^a & \text{if } d_j = 3 \\
  c_j^a & \text{if } d_j = 3 
\end{cases} \\
\text{at each } j \in J^a
\end{align*}
\]  

(3.10)
\[ c_j^e = \begin{cases} 
0.0 & \text{if } d_j = 0 \\
\frac{1}{h} & \text{if } d_j = 1 \\
\frac{1}{h} + \frac{1}{h} & \text{if } d_j = 2 \\
\frac{1}{h} + \frac{1}{h} + \frac{1}{h} & \text{if } d_j = 3 \\
\frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} & \text{if } d_j = 4 \\
\frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} & \text{if } d_j = 5 \\
\frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} & \text{if } d_j = 6 \\
\frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} & \text{if } d_j = 7 \\
\frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} + \frac{1}{h} & \text{if } d_j = 8 
\end{cases} \]

and

\[ T = \sum_{j=1}^{J} c_j^e + \sum_{j \in J} (c_j^z - c_j^z_{\text{L}} - (a_{p1j} + a_{p2j} + a_{n_j}^s + a_{n_j}^e)) \times_j \]

Also, rewrite constraints (2.3) to (2.7) as follows. Let

\[ f^s(d_j) = \begin{cases} 
1 & \text{if } d_j = 1, 2, 3 \text{ or } 4 \\
0 & \text{otherwise} 
\end{cases} \quad \text{for each } j \in J^P \]

\[ f^d(d_j) = \begin{cases} 
1 & \text{if } d_j = 5, 6, 7 \text{ or } 8 \\
0 & \text{otherwise} 
\end{cases} \quad \text{for each } j \in J^P \]

\[ f^{sa}(d_j) = \begin{cases} 
1 & \text{if } d_j = 2 \text{ or } 3 \\
0 & \text{otherwise} 
\end{cases} \quad \text{for each } j \in J^P \]

\[ f^{da}(d_j) = \begin{cases} 
1 & \text{if } d_j = 6 \text{ or } 8 \\
0 & \text{otherwise} 
\end{cases} \quad \text{for each } j \in J^P \]

\[ f^a(d_j) = \begin{cases} 
1 & \text{if } d_j = 1 \text{ or } 3 \\
0 & \text{otherwise} 
\end{cases} \quad \text{for each } j \in J^a \]
Redefine

\[ z_{sa}^{bm} = \begin{cases} 
1 & \text{if the surface mine effluents on abatement level } b \text{ at mine } m \text{ are to be abated} \\
0 & \text{otherwise}
\end{cases} 
\]

for each \( b = 1 \) to \( B_m \) and \( m \in M^p \)

\[ z_{da}^{bm} = \begin{cases} 
1 & \text{if the deep mine effluents on abatement level } b \text{ at mine } m \text{ are to be abated} \\
0 & \text{otherwise}
\end{cases} 
\]

for each \( b = 1 \) to \( B_m \) and \( m \in M^p \)

\[ z_{ba}^{bm} = \begin{cases} 
1 & \text{if the effluents on abatement level } b \text{ at mine } m \text{ are to be abated} \\
0 & \text{otherwise}
\end{cases} 
\]

for each \( b = 1 \) to \( B_m \) and \( m \in M^a \)

so that we require

\[ z_{1m}^{sa} \geq z_{2m}^{sa} \geq \ldots \geq z_{B_m}^{sa} \quad \text{for each } m \in M^p \] (3.13)

\[ z_{1m}^{da} \geq z_{2m}^{da} \geq \ldots \geq z_{B_m}^{da} \quad \text{for each } m \in M^p \] (3.14)

\[ z_{1m}^{a} \geq z_{2m}^{a} \geq \ldots \geq z_{B_m}^{a} \quad \text{for each } m \in M^a \] (3.15)

Now the constraints (2.3) to (2.7) transform to

\[ f^s(d_j) = y^s_m \quad \text{for each } j \in J_{sa}, \quad m \in M^p \] (3.16)

\[ f^d(d_j) = y^d_m \quad \text{for each } j \in J_{da}, \quad m \in M^p \] (3.17)

\[ f^{sa}(d_j) = z_{bm}^{sa} \quad \text{for each } j \in J_{bm}, \quad b=1 \text{ to } B_m, \quad m \in M^p \] (3.18)

\[ f^{da}(d_j) = z_{bm}^{da} \quad \text{for each } j \in J_{bm}, \quad b=1 \text{ to } B_m, \quad m \in M^p \] (3.19)

\[ f^{a}(d_j) = z_{bm}^{a} \quad \text{for each } j \in J_{bm}, \quad b=1 \text{ to } B_m, \quad m \in M^a \] (3.20)
The minimum cost problem formulated in chapter II now translates to minimize $C^T$ given in (3.12) subject to the constraints (3.13) to (3.20) and (2.23), where $C^T$ is supported by (3.1) to (3.5), (3.8) to (3.11), and (2.14) to (2.15); and (2.23) is supported by (3.6), (3.7), (2.10) and (2.11).

We are now required to reorder $d_j$ such that $C^T$ is monotone nondecreasing in the new ordering. There is however a complexity to consider. That is, the cost of $d_j$ at a node $j$ that is attributable to $C^T$ depends on the presence or absence of instream treatment downstream of the node. If there is no instream treatment downstream of node $j$, then (3.10) or (3.11), whichever applies, gives the cost of $d_j$. But when there is instream treatment downstream, then the cost of $d_j$ must include the cost of treating the average acid annual output as given by (3.8) or (3.9), whichever applies, over and above the cost in (3.10) or (3.11). Note that according to equation (3.12), only positive average annual loads to active instream processors incur treatment costs, and there is the possibility that the average annual acid output due to $d_j$ is not treated at all at the downstream processor or treated only partially (natural alkaline flow could neutralize all or some of this acid). However, by assuming that the optimal solution would remain unchanged if we used the following equation for $C^T$ instead of (3.12),

$$C^T = \sum_{j=1}^{J} c^e_j + \sum_{j \in J} \left( c^v_j + c^v (a^p_{1j} + a^p_{2j} + a^q_j) \right) x^z_j$$

(3.21)

this complexity can be circumvented. In other words, we assume that
instream treatment being very expensive can be economical only when a sufficiently large level of pollution is treated, and that the case when an instream treatment processor experiences an alkaline annual load would imply wasteful expenditure. Secondly, average annual natural pollutant loads are generally much smaller in magnitude than mine outputs, as a result, the fictitious negative cost accrued from treating a negative annual load will not be sufficient to outweigh the otherwise high expenditure.

There are thus two possible orderings for the source alternatives at each node, one when there is no instream treatment downstream of the node and the second when there is instream treatment downstream. Since the lexicographic ordering of vectors implies that lower hierarchical variables cycle through their values for each fixed set of the higher hierarchical variables, we are required to order the decision variables in a vector X such that all source decision variables upstream of each instream treatment site are to the right of the instream treatment decision variable. This ensures the implementation of the appropriate ordering for the source decision alternatives depending on the presence or absence of instream treatment downstream. The next section addresses the ordering of the decision variables in a vector X.

When there is no instream treatment downstream of a node, say j, then the alternatives in d_j are ordered in ascending costs per equation (3.10) or (3.11), whichever applies. Let x^e_j represent this new ordering, i.e., x^e_j=0 is the least cost alternative, x^e_j=1 is the next higher cost alternative, and so on. And let r_j relate d_j to x^e_j; i.e.,
Then it is clear that

\[ d_j = r_j(x_j^e) \text{ for each } j = 1 \text{ to } J \]  

(3.22)

where \( c_j^e \) is given by (3.10) or (3.11) accordingly, and \( x_j^e \) is the maximum value \( x_j^e \) can take on. This maximum value can vary from node to node among single effluent source nodes because we can eliminate redundant alternatives at these nodes. A redundant alternative is one that incurs a cost not less than the cost of some other alternative and produces a pollutant flow not less than the flow produced by this other alternative for each worst-case situation. For example, five alternatives at a single effluent potential mine source yield the zero pollutant flow for each worst-case situation. We need consider only the cheapest of these alternatives. Regarding multiple effluent source nodes, the constraints (3.16) to (3.20) prohibit any such elimination.

When there is instream treatment downstream of a node, say \( j \), let \( x_j^f \) portray the ascending cost ordering, and let \( g_j(x_j^f) \) relate \( x_j^f \) to \( d_j \); i.e.,

\[ d_j = g_j(x_j^f) \text{ for each } j = 1 \text{ to } J \]  

(3.23)

and

\[ c_j^e(g_j(0)) + c_{v_j}^e(g_j(0)) \leq c_j^e(g_j(1)) + c_{v_j}^e(g_j(1)) \leq \ldots \leq c_j^e(g_j(x_j^f)) + c_{v_j}^e(g_j(x_j^f)) \]
where $x^f_j$ is the maximum value that $x^f_j$ can take on. $x^f_j$ can vary from node to node among the single effluent source nodes after the elimination of redundant alternatives at these nodes as above.

ORDERING DECISION VARIABLES IN A VECTOR X

In the previous section, we came up with two orderings for the source decision alternatives in $x^e_j$ and $x^f_j$ for each $j = 1$ to $J$ such that $C^T$ is monotone nondecreasing in each $x^e_j$ or $x^f_j$, as the case may be. And $C^T$ is already monotone nondecreasing in each of the instream treatment decision variables defined in $x^z_j$ at each $j = J$. Thus we have achieved the required monotonicity of the criterion function. In this section, we arrange the decision variables in a vector $X$, to yield the lexicographic ordering of decision vectors.

Let the decision variables be ordered from downstream to upstream loosely speaking.

1. Let $x^e_j$ or $x^f_j$, as the case may be, be placed to the left of $x^e_k$ or $x^f_k$, if $j$ is downstream from $k$ on the same stream.

2. Let $x^e_j$ or $x^f_j$ be placed to the left of $x^e_k$ or $x^f_k$, if $j$ and $k$ are on streams of the same level in the same watershed and $j$ belongs to a higher numbered stream than $k$.

3. Let $x^e_j$ or $x^f_j$ be placed to the left of $x^f_k$ or $x^f_k$, if $j$ and $k$ are in the same watershed but $j$ belongs to a higher level stream than $k$. 

4. Let $x_j^e$ or $x_j^f$ be placed to the left of $x_k^e$ or $x_k^f$, if $j$ belongs to a higher numbered watershed than $k$.

5. Let $x_j^z$ be placed immediately to the left of $x_j^e$ or $x_j^f$ for each $j \in J^t$.

Thus, decisions in a higher numbered watershed have higher hierarchy than those in a lower numbered watershed. Likewise, decisions for a stream on a certain level have higher hierarchy than those on a lower level stream; and decisions for a certain stream on a given level have higher hierarchy than those for a lower numbered stream on the same level. And lastly, decisions at downstream nodes have higher hierarchy than upstream nodes on the same stream. This ordering of variables in a vector $X$ permits convenient pollutant flow analysis and two dynamic programming implementations. Also note that instream treatment decision variables are always to the left of their upstream source decision variables.

To be general, let $x_k$ denote a variable in $X$ and $\bar{x}_k$ its maximum value. Thus, $x_k$ may refer to $x_j^e$ or $x_j^f$, as the case may be, or it may refer to $x_j^z$. And $\bar{x}_k$ may refer to $\bar{x}_j^e$ or $\bar{x}_j^f$, or $\bar{x}_k=1$ if $x_k$ refers to $x_j^z$. Also, regard $X$ as $(x_1, x_2, \ldots, x_n)$, where $n$ is the total number of variables.

**NEXT FEASIBLE SOLUTION PROCEDURE IN CONJUNCTION WITH THE BRANCH AND BOUND AND DECOMPOSITION METHODS**

Each iteration of ALCOT begins with this procedure. Before the first iteration, $X$ is set to zero.
The given vector X is examined from left to right, a variable at a time. Let \( x_k \) be the current variable that is being examined, implying we have already considered \( x_1, x_2, \ldots, x_{k-1} \). The following cases are special in this procedure.

1. \( x_k \) is the first decision variable in a watershed, say \( h \), and \( h<H \).
2. \( x_k \) is an instream treatment decision variable.
3. \( x_k \) is a source decision variable and \( x_k \) belongs to a potential multiple effluent mine.
4. \( x_k \) is a source decision variable and \( x_k \) belongs to an abandoned or active multiple effluent mine.
5. \( x_k \) is a source decision variable and \( x_k \) belongs to a single effluent mine, whether abandoned, active, or potential.

These cases are considered one at a time.

**Case 1: First Decision Variable in Watershed \( h \), \( h<H \)**

Here, the decomposition principle is applied. That is, we are interested in implementing the minimum cost solution in watersheds 1 to \( h \), if this solution has been determined previously under the set of mining and(or) abatement restrictions imposed by \( (x_1, x_2, \ldots, x_{k-1}) \) upon \( (x_k, x_{k+1}, \ldots, x_n) \). These restrictions are carried in the \( y \) and \( z \) variables stipulated in the constraints (3.13) to (3.20). Initially, at the start of the next feasible solution procedure, these variables, namely the \( y \) and \( z \) variables, are all set equal to \(-1\) indicating that there are no mining or abatement restrictions imposed on the given \( X \).
Later, as $X$ is examined from left to right, and as decisions in $X$ are made feasible, appropriate $y$ and $z$ variables are updated with values zero or one to satisfy one or more of the constraints in (3.13) to (3.20) as required. These updated $y$ and $z$ variables in turn restrict the consideration of source decision alternatives at appropriate nodes that are yet to be examined in $X$. The mechanism for updating the $y$ and $z$ variables and how they in turn restrict the remaining variables in $X$ are discussed later. For now, assume that the current values given to the $y$ and $z$ variables reflect the mining and(or) abatement restrictions imposed by $(x_1, x_2, \ldots, x_{k-1})$ upon $(x_k, x_{k+1}, \ldots, x_n)$. We then compare these values of the $y$ and $z$ variables against a list of stored restrictions. If there is a match, we implement the corresponding stored minimum cost solution for $(x_k, x_{k+1}, \ldots, x_n)$ and halt the next feasible solution procedure. Otherwise, we pass control to case 2, 3, 4 or 5 described below. We discuss later how a minimum cost subvector solution under a set of restrictions is identified and stored in the course of ALCOT.

Case 2: Instream Treatment Decision Variable

The given $x_k$ is either zero or one. If it is one, it implies the decision to implement the instream processor. Since we assume the processor to be active for setting $x_1$ to $x_{k-1}$ at feasible values, $x_k=1$ must be feasible. If $x_k$ is zero, we determine if the absence of an instream processor at node $j$ can meet the quality standard at $j$ and at nodes downstream from $j$ for each worst-case. We do this by maximizing the potential for $x_k=0$ to be feasible. That is, we assume minimum
pollutant output values in each \(x_{k+1}\) to \(x_n\). With this assumption, and
given the previously fixed values for \(x_1\) to \(x_{k-1}\), it is quite
straightforward to use equations (3.22), (3.23), (3.6), (3.7), (2.10),
(2.11) and (2.23) to establish if \(x_k = 0\) is feasible. If the value zero
is feasible, it is retained. Otherwise, \(x_k\) is set to one, and \(x_{k+1}\) to
\(x_n\) are each set to zero in accordance with the lexicographic ordering of
vectors.

Case 3: Source Decision Variable defined
at a Potential Multiple Effluent Mine Node

Let \(j\) be the node, \(m\) the mine and \(b\) the abatement level of the effluent
to node \(j\). Suppose \(j\) is the highest hierarchy node among all nodes
belonging to mine \(m\). Then it is clear, \(y_m^d = y_m^d = z_{bm}^d = z_{bm}^d = 1\); i.e., there
are no mining and abatement restrictions on \(x_k\). So we are only required
to check if the given value of \(x_k\) satisfies the quality standard at node
\(j\) and nodes downstream of \(j\) for each worst-case. As said earlier, we
assume minimum pollutant output decision values in each \(x_{k+1}\) to \(x_n\); fix
\(x_1\) to \(x_{k-1}\) at their previously set values; and use equations (3.22) or
(3.23), (3.6), (3.7), (2.10), (2.11) and (2.23) to test if the given
value of \(x_k\) is feasible. If so, we retain its value; otherwise, we
increase \(x_k\) to the next higher value that is feasible, and set \(x_{k+1}\) to
\(x_n\) each to zero in accordance with the lexicographic ordering of
vectors. Having obtained a feasible value for \(x_k\), we determine \(d_j\) from
(3.22) or (3.23), then set \(y_m^s\), \(y_m^d\), \(z_{bm}^s\) and \(z_{bm}^d\) according to equations
(3.16) to (3.19), and update
\[ z_{sa} = z_{sa} = \ldots = z_{sa} = 1 \quad \text{if} \quad z_{sa} = 1 \]
\[ z_{1m} = z_{2m} = \ldots = z_{b-1,m} = 1 \quad \text{if} \quad z_{b-1,m} = 1 \]
\[ z_{b+1,m} = z_{b+2,m} = \ldots = z_{b,m} = 0 \quad \text{if} \quad z_{b,m} = 0 \]
\[ z_{da} = z_{da} = \ldots = z_{b-1,m} = 1 \quad \text{if} \quad z_{b-1,m} = 1 \]
\[ z_{1m} = z_{2m} = \ldots = z_{b,m} = 0 \quad \text{if} \quad z_{b,m} = 0 \]

in accordance with the requirements in (3.13) and (3.14).

Suppose next that \( j \) is not the highest hierarchical node among the nodes denoted for mine \( m \). Then we must have \( y_m^S = 0 \) or 1 and \( y_m^d = 0 \) or 1 since we have previously considered the highest hierarchical node among the nodes denoted for mine \( m \). In consequence, we must restrict \( d_j \) such that (3.16) and (3.17) are achieved, and then restrict \( x_k \) according to (3.22) or (3.23). We now face one of the following possibilities:

1) \( y_m^S = y_m^d = 0 \),
2) \( y_m^S = 1 \) and \( z_{sa} = -1 \),
3) \( y_m^d = 1 \) and \( z_{da} = -1 \),
4) \( y_m^S = 1 \) and \( z_{bm} = -1 \),
5) \( y_m^d = 1 \) and \( z_{bm} = -1 \).

If we have possibility 1, we set \( x_k \) to imply the "do not mine" decision. If we have possibility 2, we set \( x_k \) to imply surface mining and such that it achieves the quality standard for each worst-case. Also, we find \( d_j \) corresponding to the feasible value of \( x_k \), set \( z_{sa} \) according to (3.18) and set \( z_{lm}^{sa}, \ldots, z_{b-1,m}^{sa} \) each to 1 if \( z_{bm}^{sa} = 1 \) or set \( z_{b+1,m}^{sa}, \ldots, z_{b,m}^{sa} \) each to zero otherwise. Possibility 3 is like possibility 2, with the deep mine restriction instead of the surface mine restriction. Possibilities 4 and 5 are extensions of possibilities 2 and 3 respectively, where we are required to meet the abatement restriction in addition. Note that for each possibility above, we assume minimum pollutant output decisions for each decision variable to the right of \( x_k \). Also, we set these variables to the right to zero if we are forced to increase \( x_k \) from its given value.
Case 4: Source Decision Variable defined at an Abandoned or Active Multiple Effluent Mine

This case is like case 3 except we do not contend with the mining restriction. Let \( x_k \) be defined at node \( j \), let \( j \) belong to mine \( m \), and let \( b \) be the abatement level of the effluent to node \( j \). There are two possibilities: 1) \( z_{bm}^a = -1 \) and 2) \( z_{bm}^a = 0 \) or 1. In the former case, there is no abatement restriction on \( x_k \). So, we set \( x_k \) to satisfy the quality standard at \( j \) and nodes downstream of \( j \). We also set \( z_{bm}^a \) per equation (3.20) and set \( z_{1m}^a, z_{2m}^a, \ldots, z_{b-1,m}^a, z_{b+1,m}^a \) each to 1 if \( z_{bm}^a = 1 \) or set \( z_{b+1,m}^a \) each to zero if \( z_{bm}^a = 0 \). In the latter case, we restrict \( x_k \) per the \( z_{bm}^a \) value and achieve the value of \( x_k \) that satisfies the quality standard at \( j \) and nodes downstream of \( j \). In either case, we assume minimum pollutant output decisions in \( x_{k+1} \) to \( x_n \), and set these decision variables to zero in the event we are forced to increase \( x_k \) from its given value.

Case 5: Source Decision Variable at a Single Effluent Mine, whether Abandoned, Active, or Potential

This is the simplest case. There is no mining or abatement restriction to contend with. We merely achieve the value of \( x_k \) that satisfies the quality standard for each worst-case as before by assuming minimum pollutant output decision values in \( x_{k+1} \) to \( x_n \). We also set \( x_{k+1} \) to \( x_n \) to zero if we are forced to increase \( x_k \) from its given value.
Application of Branch and Bound

Having processed $x_k$ as in case 2, 3, 4 or 5, we compute a lower bound on $C^T$, the total systems cost, by regarding

1) $(x_1, x_2, \ldots, x_k)$ as fixed at their feasible values, and
2) $(x_{k+1}, x_{k+2}, \ldots, x_n)$ as free,

and if this lower bound is not less than the cost of the currently held minimum cost solution, then we fathom $(x_{k+1}, x_{k+2}, \ldots, x_n)$ by setting it to $(\bar{x}_{k+1}, \bar{x}_{k+2}, \ldots, \bar{x}_n)$ and halt the next feasible solution procedure. Otherwise, we simply continue with $x_{k+1}$.

We have experimented with two methods to calculate the lower bound of interest. The first is a quick and dirty method and the second uses Dynamic Programming.

In the first method, we consider the mining and abatement restrictions imposed by $(x_1, x_2, \ldots, x_k)$ upon $(x_{k+1}, x_{k+2}, \ldots, x_n)$ as stipulated by the values in the $y$ and $z$ variables. However, we ignore any such restriction that the variables $x_{k+1}$ to $x_n$ can impose upon each other. Secondly, we set each instream treatment decision variable in $(x_{k+1}, x_{k+2}, \ldots, x_n)$ to one but ignore both the fixed and variable costs of treatment at each of these instream treatment sites. Thirdly, we assume the zero pollutant decision value in every other source decision variable in $(x_{k+1}, x_{k+2}, \ldots, x_n)$ when we consider a particular source decision variable. Under these conditions, it is quite straightforward to set a source decision variable in $(x_{k+1}, x_{k+2}, \ldots, x_n)$ to a value such
that 1) any mining and(or) abatement restrictions imposed by
\((x_1, x_2, \ldots, x_k)\) on the source variable is satisfied, 2) the quality
standard is achieved at each node for each worst-case, and 3) the value
refers to the least cost alternative among all the alternatives that
each achieves requirements 1 and 2 above. The cost of these source
decisions in \((x_{k+1}, x_{k+2}, \ldots, x_n)\) plus the cost of \((x_1, x_2, \ldots, x_k)\) then
yields the lower bound.

In the second method, we ignore all mining and abatement restrictions
imposed by \((x_1, x_2, \ldots, x_k)\) upon \((x_{k+1}, x_{k+2}, \ldots, x_n)\) as well as any such
restriction the variables \(x_{k+1}\) to \(x_n\) can impose on one another. We then
minimize the total systems cost over the free variables separately for
each worst-case situation using dynamic programming, and take the
required lower bound as the maximum of these minimum costs. Consider
now the problem of minimizing the total cost for a particular worst-case
situation, say \(i\). This problem may be stated as minimize \(c^T\) given by
\((3.21)\) subject to the quality constraints in \((2.23)\) at each \(j=1\) to \(J\) for
worst-case \(i\), where \((3.21)\) is supported by \((3.1)\) to \((3.5)\), \((3.8)\) to
\((3.11)\), \((2.14)\) and \((2.15)\); and \((2.23)\) is supported by \((3.6)\), \((3.7)\),
\((2.10)\) and \((2.11)\). Clearly, since we are ignoring the multiple effluent
mine constraints, this problem constitutes \(H\) independent subproblems,
each defined in a unique watershed. Secondly, because of the particular
ordering of the decision variables in \(X\), the subproblems in watersheds
\(h+1\) to \(H\) are trivial, letting that \(x_{k+1}\) to \(x_n\) are defined in watersheds
\(1\) to \(h\). Thirdly, the subproblems in watersheds \(1\) to \(h-1\) are independent
of \(x_1\) to \(x_k\). So they each need be solved once only during the entire
run of ALCOT. This leaves us with only the subproblem in watershed h.
Here again, the particular ordering of the decision variables in X renders the use of recursive functions that are independent of $x_1$ to $x_k$. To elucidate, let watershed h have only a single stream, and let the nodes along the stream be numbered 1 to m. Further, assume there is no instream treatment site. Then the subproblem of interest has the form

$$\begin{align*}
\text{minimize} & \quad c^e_1(d_1) + c^e_2(d_2) + \ldots + c^e_m(d_m) \\
\text{s.t.} & \quad w^e_{ii}(d_i) \leq q_{ii} - w^n_{ii} = \bar{q}_i \\
& \quad w^e_{ii}(d_i) + w^e_{i2}(d_2) \leq q_{i2} - w^n_{i2} = \bar{q}_2 \\
& \quad \vdots \\
& \quad w^e_{ii}(d_i) + w^e_{i2}(d_2) + \ldots + w^e_{im}(d_m) \leq q_{im} - \sum_{j=1}^{m} w^n_{ij} = \bar{q}_m
\end{align*}$$

where $d_1$ to $d_m$ are the source decisions we defined earlier, $c^e_1$ to $c^e_m$ correspond with the definitions in (3.10) or (3.11) accordingly, $w^e_{ii}$ to $w^e_{im}$ correspond with the definitions in (3.6) or (3.7) accordingly, and $w^n_{ii}$ to $w^n_{im}$ are the ith worst-case natural pollutant inputs. The above problem can be restated in the traditional dynamic programming form as follows:
Find $f_m(\infty)$, where

$$f_j(q_j) = \min_{d_j} \left[ c_j^e(d_j) + f_{j-1}(\min(q_j, \bar{q}_j) - w_{ij}^e(d_j)) \right]$$

s.t. $w = w_{ij}^e(d_j) + g_{j-1}(\min(q_j, \bar{q}_j) - w_{ij}^e(d_j))$

$$w \leq \min(q_j, \bar{q}_j)$$

for each $j = 1$ to $m$,

$g_j(q_j) = \text{the corresponding optimal value of } w$

for each $j = 1$ to $m$, and

$f_0(q_0) = g_0(q_0) = 0$ regardless of $q_0$.

It is clear that the recursive functions $f_j$ and $g_j$ are independent of the decisions $d_{j+1}$ to $d_m$. So, letting $x_k$ refer to $d_{j+1}$, the recursive functions $f_1$ to $f_j$ and $g_1$ to $g_j$ are independent of $x_1$ to $x_k$ since $x_1$ to $x_k$ have higher hierarchy than all decisions upstream of node $j$ and therefore cannot include any of the decisions $d_1$ to $d_j$. In consequence, the required minimum cost in watershed $h$, namely $f_m(\infty)$, is simply $f_j(q_j)$ plus the cost of the fixed decisions $d_{j+1}$ to $d_m$, where

$$q_j = \min_{r=j+1}^{m} \left( \bar{q}_{ir} - \sum_{s=j+1}^{r} w_{is}^e(d_s) \right)$$

Thus, by parametrizing these recursive functions over $q$, as and when necessary during the course of ALCOT, we can rapidly compute $f_m(\infty)$ in watershed $h$. And regarding the subproblems in watersheds 1 to $h-1$, the corresponding $f_m(\infty)$'s, once computed, give their respective minimum costs. This characterizes the efficiency of this lower bound computation method.
We have so far dealt with the simplest case: a single stream and no instream treatment site. This simple case is easily extended when multiple streams exist. Letting

\[ q_j = q_{ij} - w_{ij}^{n*} \]

where \[ w_{ij}^{n*} = w_{ij} + w_{iplj} + w_{ip2j} \]

then the recursive functions \( f_j \) and \( g_j \) for the simple case continue to hold at each nonconfluence node if we regard \( f_{j-1} \) and \( g_{j-1} \) as \( f_{p1j} \) and \( g_{p1j} \) respectively. These functions also hold at each confluence node if we replace \( f_{j-1} \) and \( g_{j-1} \) with \( f^*_j \) and \( g^*_j \), where

\[ f^*_j(q^*_j) = \min_{q} (f_{p1j}(q) + f_{p2j}(q_j - g_{p1j}(q))) \]

\[ s.t. \ w = g_{p1j}(q) + g_{p2j}(q_j - g_{p1j}(q)) \leq q_j \]

\[ g^*_j(q^*_j) = \text{the corresponding optimal value of } w \]

However, the recursive functions \( f^*_j \) and \( g^*_j \) can depend on \((x_1, x_2, \ldots, x_k)\). This is because the decisions at level 3 stream nodes have higher hierarchy than those at level 2 stream nodes, and these latter decisions in turn are of higher hierarchy than those at level 1 stream nodes. In effect, there may exist a stream confluence downstream of \( x_k \)'s node whose merging tributary (or a tributary of this tributary) may have lower hierarchical decisions than \( x_k \). Resolving the dynamic program at this stream confluence will require the corresponding recursive functions \( f^* \) and \( g^* \). Clearly, these functions are dependent on \( x_k \) and perhaps on others in \( x_1 \) to \( x_{k-1} \). In essence, we are forced to recompute them.
To obtain an efficient dynamic programming implementation, we maintain and update the following: 1) $f_j(q_j)$ and $g_j(q_j)$ at each $j = 1$ to $J$ when the decisions at $j$ and upstream of $j$ are free to vary, and 2) $f_j^*(q_j^*)$ and $g_j^*(q_j^*)$ at each stream confluence when the decisions upstream of the confluence are free to vary. But we recompute $f_j^*$ and $g_j^*$ when one or more decisions upstream of the confluence are fixed. Regarding $f_j(q_j)$ and $g_j(q_j)$ when $d_j$ is fixed, they are merely translations of upstream recursive functions that we maintain or recompute. This follows because of the particular ordering of the decision variables in $X$.

Consider now the effect of potential instream treatment sites on the dynamic program of interest. Let us redefine the cumulative natural pollutant flow as the aggregate of the natural pollutant inputs assuming active instream treatment sites; i.e.,

$$w_{ij}^n = \begin{cases} w_{ij}^{nc} & \text{at each } j \notin J^t \\
\min(0, w_{ij}^{nc}) & \text{at each } j \in J^t
\end{cases}$$

where $w_{ij}^{nc} = w_{ij}^n + w_{ip1j}^n + w_{ip2j}^n$.

Then the recursive functions we indicated for the multiple stream case continue to hold at each $j \notin J^t$ provided there is no active instream treatment site downstream of $j$. When there is such a site downstream of $j$, the recursive functions $f_j$ and $g_j$ do not apply because $f_j$ does not include the chemical treatment costs incurred at the downstream processor. So, we are required to develop new recursive functions for this case. However, the principles remain the same. That is, if we
define $f_j^t$ and $g_j^t$ as the new recursive functions, and likewise, $f_j^{*t}$ and $g_j^{*t}$ as the new recursive functions at each stream confluence that is upstream of an active instream processor, then the recursive functions we indicated for the multiple stream case continue to hold at each $j \not\in J^t$ when we replace $f_j$, $g_j$, $f_j^*$ and $g_j^*$ with $f_j^t$, $g_j^t$, $f_j^{*t}$ and $g_j^{*t}$ respectively, and include the treatment cost term $c v_j^e(d_j)$ in the objective of $f_j^t(q_j)$, where $a_j^e(d_j)$ is the average annual load due to $d_j$ that is treated at the downstream processor. So, what we have left to consider are the recursive functions at each instream treatment node $j \in J^t$.

Consider first the case when the instream treatment site at $j$ is inactive. Then the recursive functions we indicated for each $j \not\in J^t$ apply here if we replace $w_{ij}^e(d_j)$ in $f_j(q_j)$ and $g_j(q_j)$, or in $f_j^t(q_j)$ and $g_j^t(q_j)$, as the case may be, with $w_{ij}^e(d_j) + \max(0, -w_{ij}^c)$. We require this because we assume $w_{ij}^e = 0$ when $w_{ij}^c$ is positive, as a consequence, node $j$ and nodes downstream of $j$ will experience this positive flow when the instream treatment site $j$ is inactive. We note that when $x_j^z$, the instream treatment decision, is free to vary, $w_{ij}^c$ may be greater than $\bar{q}_j$ or $q_j$, in which case, $x_j^z = 0$ is infeasible, and we regard $f_j(q_j)|x_j^z = 0$ (the symbol "|" means "given that") or $f_j(q_j)|x_j^z = 0$, as the case may be, as infinity.

When the instream treatment site at $j$ is regarded active, letting $q'_j$ denote the upper limit on the $i$th worst-case flow entering the instream processor, then $q'_j$ is infinity if $-w_{ij}^c > q_j$ since $q_j$ is always nonnegative and the maximum possible flow out of the processor is $\max(0, -w_{ij}^c)$, which
in turn is a consequence of the definition \( w_{ij}^{n*} = \min(0, w_{ij}^{nc}) \). Otherwise, \( q_j' = q_j \). Note that \( q_j' \) does not enter the picture because by definition it is not less than \( -w_{ij}^{nc} \). The recursive functions can now be stated as:

\[
\begin{align*}
(f_j(q_j)|x_j^z=1) &= c_j^z + c_v a_j^{n*} + f_j'(q_j) \\
(g_j(q_j)|x_j^z=1) &= \max\{0, \min(-w_{ij}^{nc}, g_j'(q_j))\}
\end{align*}
\]

when there is no active instream treatment site downstream from the current site, or

\[
\begin{align*}
(f_j'(q_j)|x_j^z=1) &= c_j^z + f_j'(q_j) \\
(g_j(q_j)|x_j^z=1) &= (g_j(q_j)|x_j^z=1)
\end{align*}
\]

otherwise, where \( a_j^{n*} \) is the cumulative average annual natural pollutant load entering the processor at \( j \) and given by

\[
a_j^{n*} = a_j + a_j^{n*} + a_j^{n*}
\]

and \( f_j' \) and \( g_j' \) correspond in form with \( f_j \) and \( g_j \) defined at nodes that do not host instream treatment.

When \( x_j^z \) is free to vary and there is no active downstream processor, \( f_j(q_j) \) is taken as the minimum of \( f_j(q_j)|x_j^z=0 \) and \( f_j(q_j)|x_j^z=1 \), and \( g_j(q_j) \) corresponds. \( f_j'(q_j) \) and \( g_j'(q_j) \) are obtained similarly when there exists an active instream processor downstream. When \( x_j^z \) is fixed, we use the corresponding conditional recursive functions.

We have now covered all the recursive functions we need. We mentioned earlier that we maintain and update the recursive functions \( f_j, g_j, f_j^* \) and \( g_j^* \) when the decisions at \( j \) and upstream of \( j \) are free to
vary. We do likewise for \( f_j^t, g_j^t, f_j^* \), and \( g_j^* \). In addition, as noted, we recompute \( f_j^* \) and \( g_j^* \), or \( f_j^t \) and \( g_j^t \), as the case may be, when one or more decisions upstream of the stream confluence are fixed. Regarding \( f_j \) and \( g_j \) or \( f_j^t \) and \( g_j^t \) when \( d_j \) is fixed, they are simply translations of upstream recursive functions that we maintain or recompute.

Both methods, the quick and dirty method and the dynamic programming method are rapid, the quick and dirty method being the faster but expected to give poorer lower bounds. Included in both methods is a special purpose device. To elucidate, let \((x_1, x_2, \ldots, x_k)\) be defined in watersheds \( h+1 \) to \( H \). Then the mining and(or) abatement restrictions imposed by \((x_1, x_2, \ldots, x_k)\) upon those variables in \((x_{k+1}, x_{k+2}, \ldots, x_n)\) that are defined in watersheds \( 1 \) to \( h \), are compared against a list of stored restrictions. If there is a match, then the corresponding stored cost is either the cost of the minimum cost solution in watersheds \( 1 \) to \( h \) or a lower bound on this cost (the details are deferred to the next section). So, in this event, the lower bound computation procedure need not include watersheds \( 1 \) to \( h \).

STORING MINIMUM COST SUBVECTOR SOLUTIONS

This scheme is used between the next feasible solution procedure and the skipping procedure. It applies only when the next feasible solution procedure yields a feasible solution.

Recall that when the next feasible solution procedure in conjunction with branch and bound method yields a feasible solution, it must be
better than the previously held minimum cost solution. It is also the case that each subvector \((X_{H-1}^1, \ldots, X_1^1), (X_{H-2}^1, \ldots, X_1^1), \ldots, (X_{h+1}^1, \ldots, X_1^1)\) under the current set of mining and(or) abatement restrictions has an improved solution, where \(X_i^1\) in general refers to the decision vector in watershed \(i\), and \(h\) is the watershed at which a stored minimum cost solution was reimplemented in the next feasible solution procedure (if \(h=0\), this implementation did not occur). Each of the indicated subvectors has an improved solution because of the lexicographic ordering of vectors. That is, \((X_{h+1}^1, \ldots, X_1^1)\) cycles through all its values for each \((X_H^1, X_{H-1}^1, \ldots, X_{h+2}^1)\), and since we have an improved overall solution, it means the current \((X_{h+1}^1, \ldots, X_1^1)\) must be improved for the fixed \((X_H^1, X_{H-1}^1, \ldots, X_{h+2}^1)\). Likewise, the argument extends to the subvectors \((X_{h+2}^1, \ldots, X_1^1), (X_{h+3}^1, \ldots, X_1^1), \ldots, (X_{H}^1, \ldots, X_1^1)\). Also, the fixed \((X_H^1, X_{H-1}^1, \ldots, X_{h+2}^1)\) imposes a fixed set of mining and(or) abatement restrictions on \((X_{h+1}^1, \ldots, X_1^1)\), the fixed \((X_H^1, X_{H-1}^1, \ldots, X_{h+3}^1)\) imposes a fixed set of mining and(or) abatement restrictions on \((X_{h+2}^1, \ldots, X_1^1)\), and so on. Thus, these minimum cost subvector solutions are stored in a list against their respective restrictions and updated each time an improvement is realized. Eventually, when the subvector cycles through all its solutions, the currently stored solution is declared the minimum and used in the next feasible solution procedure as indicated. It is also possible that a subvector solution may never have been stored for a set of restrictions. This can happen when the branch and bound method takes effect throughout the scan of the subvector space. This in turn means that the minimum cost of the subvector must be greater than or equal to the difference between the current overall
minimum cost and the cost of the fixed higher hierarchical subvector. This difference is stored as a lower bound on the cost of the subvector under the current set of mining and (or) abatement restrictions, and used in the branch and mound method as indicated.

SKIPPING FROM X TO X*

The method here is applied to the subvector \((x^H, x^{H-1}, \ldots, x^{h+1})\) obtained from the next feasible solution procedure, where \(h\) is the watershed at which a stored minimum cost solution was reimplemented in the next feasible solution procedure. Since the subvector \((x^h, \ldots, x^1)\) is currently optimal, it would be pointless to consider it. However, since its optimality is conditional upon the current set of mining and (or) abatement restrictions that \((x^H, \ldots, x^{h+1})\) imposes upon \((x^h, \ldots, x^1)\), we require to consider any change in this set of restrictions as \((x^H, \ldots, x^{h+1})\) changes.

Let \(x_e\) be the right most decision variable in \(x^{h+1}\). One requirement is that any \(Y=(y_1, y_2, \ldots, y_e)\) between \((x_1, x_2, \ldots, x_e)\) and \((x^*_1, x^*_2, \ldots, x^*_e)\) must satisfy \(y_1 > x_1, y_2 > x_2, \ldots, y_e > x_e\). A second is that each \(Y\) identified above, must impose the same set of mining and (or) abatement restrictions on \((x^h, \ldots, x^1)\) as \((x_1, x_2, \ldots, x_e)\). These two requirements can be met quite readily. Let \(x_m\) be the first nonzero decision from right to left in \((x_1, x_2, \ldots, x_e)\). Consequently, any vector

\[Y=(x_1, x_2, \ldots, x_{m-1}, y_m, y_{m+1}, \ldots, y_e)\]
beyond $X$ in the lexicographic ordering of vectors will be greater than or equal to $X$, where $y_{m+1}, y_{m+2}, \ldots, y_e \geq 0$. So by setting $x_{m}^* = 0$ and adding 1 to $(x_1^*, x_2^*, \ldots, x_{m-1}^*)$ to obtain $(x_1^*, x_2^*, \ldots, x_{m-1}^*)$, we derive the largest ascent in the lexicographic ordering of vectors without violating requirement 1. However, requirement 2 above may be violated. To ensure this, we consider the subvector $(x_m, x_{m+1}, \ldots, x_e)$ from right to left, and letting $x_k^*$ denote the current variable, we determine if any value between $x_{k+1}^*$ and $\overline{x}_k^*$ alters the set of restrictions implied by $(x_1^*, x_2^*, \ldots, x_e^*)$. If there is alteration, we set $x_k^*$ to the next higher value from $x_k^*$ that causes the alteration, keep $x_i^* = x_i$ at each $i=1$ to $e$ but $i \neq k$, and set $x_i^* = 0$ for each $i=e+1$ to $n$. Otherwise, we simply proceed with the next $x_k^*$. Eventually, if the procedure runs to its end without encountering a change in the mining and abatement restrictions of interest, then we set $X = X^*$ per the Lawler-Bell criterion that we exposed above.

STOPPING VECTOR

The main idea behind the determination of the stopping vector involves the computation of the maximum feasible pollutant flow past a given node for a given worst-case situation, under assumptions that are consistent with the lexicographic ordering of vectors in $X$. Assume, for simplicity, that both multiple effluent mines and potential instream treatment sites are absent in the given problem. Consider now the determination of the stopping vector value of $x_k^L$, implying $X^L$ denotes the stopping vector. Suppose we assume zero pollutant outputs in the
mine effluents associated with the decisions \( x_1 \) to \( x_{k-1} \). Clearly, since \( x_1 \) to \( x_{k-1} \) are of higher hierarchy than \( x_k \), this assumption is consistent with the lexicographic ordering of vectors in \( X \); i.e., it implies \( x_1 \) to \( x_{k-1} \) are fixed at \( \bar{x}_1 \) to \( \bar{x}_{k-1} \). Further, if we assume \( x_k \) is also fixed at \( \bar{x}_k \), implying zero pollutant output in its mine effluent, and then consider the problem

\[
\text{Find } s_{ij}
\]

\[
s_{ij} = \min_{r \in U_j} \left( \bar{q}_{ir} - \bar{w}_{ir} \right)
\]

\[
\bar{w}_{ir} = \text{value of the problem:}
\]

\[
\begin{align*}
&\text{maximize } \bar{w}_{ir} \\
&x_{k+1} \text{ to } x_n \\
&\text{s.t. } \bar{w}_{it} \leq \bar{q}_{it} \text{ for some } t \text{ and each } t = 1 \text{ to } J
\end{align*}
\]

where \( x_k \) is defined at node \( j \), \( U_j \) is the set of nodes including \( j \) and nodes downstream from \( j \), and \( \bar{w}_{it} \) is given by (2.10) with \( t \) replacing \( j \); then clearly, \( s_{ij} \) signifies the minimum \( i \)th worst-case quality slack or the maximum \( i \)th worst-case restriction that \( x_{k+1} \) to \( x_n \) impose upon the \( i \)th worst-case mine effluent output due to \( x_k \). Conversely, any value of \( x_k \) whose corresponding \( i \)th worst-case mine effluent output is within \( s_{ij} \) has no influence on the \( i \)th worst-case quality constraints that \( x_{k+1} \) to \( x_n \) must satisfy. Consequently, since the criterion is monotone nondecreasing in \( x_k \), the smallest value of \( x_k \) whose mine effluent output is within \( s_{ij} \) for each \( i = 1 \) to \( I^* \), can be taken as \( x_k^L \). Since \( k \) is arbitrary, \( X^L \) can be obtained in general.
The idea motivating the above approach is that the pollution assimilative capacity of a stream generally increases as the stream flows downstream (due to flow volume increase). This in turn implies \( s_{ij} \) can be significant from node to node.

From a computational viewpoint, what is of necessity is the calculation of any \( \dot{w}_{ir}^* \); i.e., the maximum feasible ith worst-case flow past node \( r \) over the free decision variables \( x_{k+1} \) to \( x_n \) and assuming zero pollutant outputs in the effluents associated with \( x_1 \) to \( x_k \). This calculation can be expressed as the following dynamic program:

Find \( \dot{w}_{ir}^* = g_r(\infty) + w_{ir}^{n*} \)

where \( g_m(q_m) = \max \{w_{im}^e(d_m) \}
\)
\[d_m + g_{p_{1m}}(\min(q_m,q_{im}^*)-w_{im}^e(d_m)) \]
\[\leq \min(q_m,q_{im}^*) \]
for \( m=r \), each \( m \) upstream of \( r \), and \( m \) is not a confluence node,

\[g_m(q_m) = \max \{w_{im}^e(d_m) \}
\]
\[d_m + g_{m}(\min(q_m,q_{im}^*)-w_{im}^e(d_m)) \]
\[\leq \min(q_m,q_{im}^*) \]
for \( m=r \), each \( m \) upstream of \( r \), and \( m \) is a confluence node,

\[g_m(q_m^*) = \max \{g_{p_{1m}}(q) + g_{p_{2m}}(q^* - g_{p_{1m}}(q)) \}
\]
\[\leq q_m \]
for \( m=r \), each \( m \) upstream of \( r \), and \( m \) is a confluence node,

\[g_0(q_0) = 0 \] regardless of \( q_0 \),

\[q_{im}^* = q_{im} - w_{im}^{n*} \]
and \( w_{im}^* \) is given by (3.6) or (3.7) accordingly. Note that \( d_m \) varies freely only at nodes associated with \( x_{k+1} \) to \( x_n \); at each remaining node, it takes only the single value corresponding to \( w_{im}^e = 0 \). An analysis similar to that exposed for the dynamic programming lower bound computation method, reveals that an efficient overall implementation can be obtained by 1) maintaining and updating \( g_m \) at each \( m \) where \( d_m \) is free, 2) maintaining and updating \( g_m^* \) at each stream confluence whose upstream decisions are free, and 3) recomputing \( g_m^* \) at each stream confluence that has one or more fixed upstream decisions. Also, at each \( m \) where \( d_m \) is fixed, \( g_m \) is simply a translation of an upstream recursive function that we maintain or recompute. Thus, \( s_{ij} \) can be determined quite rapidly for each \( i \) and \( j \) in a recursive manner, and \( X^L \) can be obtained consequently.

We have so far considered only the simplest case: no multiple effluent mine and no instream treatment site. When multiple effluent mines are present (but no instream treatment site), we first obtain the \( s_{ij} \)'s as above, ignoring the multiple effluent mine constraints. Then we proceed from left to right in \( X \), and letting \( x_k \) be the current decision variable we are examining, we obtain all the candidate decision alternatives of \( x_k \). Each candidate decision must satisfy any multiple effluent mine constraint imposed by \( x_1^L \) to \( x_{k-1}^L \) on \( x_k \) and release a mine effluent output that is within \( s_{ij} \) for each \( i=1 \) to \( I^j_k \), where \( j \) as usual is \( x_k \)'s node. If \( j \) is a single effluent mine's node, then \( x_k^L \) is set equal to the least cost candidate decision as noted earlier. Otherwise,
$x_k^L$ is set such that 1) it is a candidate decision, and 2) there does not exist a smaller valued (less costly) candidate decision that (together with $x_1^L$ to $x_{k-1}^L$) imposes the same set of mining and(or) abatement restrictions on $(x_{k+1}, \ldots, x_n)$ as $x_k^L$.

Finally, regarding the case when instream treatment decisions are present, we simply set $x_k^L=1$ for each $k$ referring to an instream treatment decision. This avoids a complexity that arises because instream treatment does not affect alkaline flows. We note that we redefine $U_j$ to refer to node $j$ and nodes downstream from $j$ until the first instream treatment site. But $U_j$ excludes the node hosting the instream treatment site; if the site is at $j$, $U_j$ is empty and $s_{ij}$ is infinity for each $i=1$ to $I_j^*$. We also assume zero worst-case pollutant output from each instream processor; so $w_{im}^{n^*}$ in the various recursive functions which we portrayed earlier refers now to the cumulative natural pollutant flow past node $j$ assuming active instream treatment processors.

Because of the similarities between the dynamic programming lower bound computation method and the stopping vector procedure, we do not recommend the use of the stopping vector when this lower bound computation method is employed in ALCOT. That is, in this case, we expect ALCOT to scan all vectors beyond $x_k^L$ in a time that is less than the computational time for the stopping vector. Of course, only computational experience can verify this contention.
CHAPTER IV
A BRANCH AND BOUND ALGORITHM
USING LAGRANGIAN RELAXATION

This chapter presents a branch and bound enumeration algorithm for the minimum cost problem using Lagrangian relaxation of both the multiple effluent mine constraints in (2.3) to (2.7) and the quality constraints in (2.23). The algorithm has theoretic parallels with the LP based branch and bound algorithm of Chapter V, and it would be interesting to see how its performance compares with that of the LP based algorithm.

The primary motive for this Lagrangian relaxation is it lends itself to many of the special structures constituting the minimum cost problem. These special structures are:

1. the nonlinear clipping effect of instream treatment,
2. the recurring terms in the quality constraints in (2.23),
3. the integrality property of the multiple choice and multiple effluent mine constraints in (2.1) to (2.7),
4. most decisions render pollution flow reductions simultaneously across all worst-case situations, and
5. for any fixed set of instream treatment decisions, the minimum cost problem transposes to an ILP.

The weakness of this relaxation is it tends to give poor lower bounds. But as is typically the case, this weakness is compensated by the ease and speed with which the Lagrangian optimum can be computed.
We now give our attention to the Lagrangian relaxation of interest, covering some theory and discussing why the relaxation can yield poor lower bounds. We then pursue the algorithm.

THE LAGRANGIAN PROBLEM

Consider the problem

\[
\min C^R = C^T + \sum_{j=1}^{J} \sum_{i=1}^{I_j} u_{ij}(w_i^* - q_i^j) \\
+ \sum_{m \in M^p} \sum_{j \in J_m} r_s(x_j^s + x_j^{sa} + x_j^{st} + x_j^{sat} - y_m^s) \\
+ \sum_{m \in M^p} \sum_{j \in J_m} r_d(x_j^d + x_j^{da} + x_j^{dt} + x_j^{dat} - y_m^d) \\
+ \sum_{m \in M^p} \sum_{b=1}^{B_m} \sum_{j \in J_{bm}} r_s(x_j^{sa} + x_j^{sat} - \sum_{k=b}^{B_m} z_{km}^{sa}) \\
+ \sum_{m \in M^p} \sum_{b=1}^{B_m} \sum_{j \in J_{bm}} r_d(x_j^{da} + x_j^{dat} - \sum_{k=b}^{B_m} z_{km}^{da}) \\
+ \sum_{m \in M^a} \sum_{b=1}^{B_m} \sum_{j \in J_{bm}} r_a(x_j^a + x_j^{at} - \sum_{k=b}^{B_m} z_{km}^{a})
\]

s.t.

\[
\sum_{j \in J_{m}} r_j^s = c_m^s \text{ for each } m \in M^p
\]

\[
\sum_{j \in J_{m}} r_j^d = c_m^d \text{ for each } m \in M^p
\]
\[ \sum_{j \in J_x^m} r_{ja} = c_{B_m}^{sa} \text{ for each } m \in M^p \tag{4.4} \]

\[ \sum_{j \in J_x^m} r_{jd} = c_{B_m}^{da} \text{ for each } m \in M^p \tag{4.5} \]

\[ \sum_{j \in J_x^m} r_{ja} = c_{B_m}^a \text{ for each } m \in M^a \tag{4.6} \]

\[ \sum_{k=1}^{b} \sum_{j \in J_{km}} r_{ja} \leq c_{bm}^{sa} \text{ for } b=1 \text{ to } B_m-1 \text{ and } m \in M^p \tag{4.7} \]

\[ \sum_{k=1}^{b} \sum_{j \in J_{km}} r_{jd} \leq c_{bm}^{da} \text{ for } b=1 \text{ to } B_m-1 \text{ and } m \in M^p \tag{4.8} \]

\[ \sum_{k=1}^{b} \sum_{j \in J_{km}} r_{ja} \leq c_{bm}^a \text{ for } b=1 \text{ to } B_m-1 \text{ and } m \in M^a \tag{4.9} \]

and the constraints in (2.1) and (2.2) \hspace{1cm} \text{(LR}_{ur})

where \( C^\top \) is given by equations (2.12) to (2.22), \( w_{ij}^* \) by equations (2.8) to (2.11), \( u \) is a vector of nonnegative multipliers associated with the quality constraints, and \( r \) is a vector of multipliers (unrestricted in sign) associated with the multiple effluent mine constraints. The vector \( r \) must satisfy the constraints (4.2) to (4.9) in order that the Lagrangian problem is bounded. To elucidate, the equalities (4.2) to (4.6) must hold because of the definition of \( c_m \) in (2.20) or (2.21) accordingly, and because \( y_m^s, y_m^d, z_{B_m}^{sa}, z_{B_m}^{da} \) and \( z_{B_m}^a \) are all unrestricted at each appropriate \( m \). Similarly, the inequalities in (4.7) to (4.9) are necessary because of the definition of \( c_m \), and because \( z_{B_m}^{sa}, z_{B_m}^{da} \) and \( z_{B_m}^a \), for \( b=1 \) to \( B_m-1 \), and for each appropriate \( m \), do not have upper bounds. Problem \( \text{(LR}_{ur}) \) is the Lagrangian relaxation of the minimum cost problem relative to the multiple effluent mine and quality constraints.
It is well known that
\[ v(LR_{ur}) \leq v(P) \]
for all \( u \geq 0 \) and all \( r \), where \((P)\) is the minimum cost problem and \( v(.) \) denotes the optimal value of problem \((.)\). So the usual idea is to maximize \( v(LR_{ur}) \) over \( u \) and \( r \), for this gives the tightest lower bound on \( v(P) \).

Of theoretical and practical interest is the gap between the maximum of \( v(LR_{ur}) \) over \( u \) and \( r \) and \( v(P) \). This gap may be viewed as the gap between the maximum of \( v(LR_{ur}) \) over \( u \) and \( r \) and the maximum of \( v(LR_{u}) \) over \( u \), plus the gap between the maximum of \( v(LR_{u}) \) over \( u \) and \( v(P) \), where problem \((LR_{u})\) is the Lagrangian relaxation of only the quality constraints in \((P)\). In particular, it is the latter gap between \( v(LR_{u}) \) and \( v(P) \) that is significant; the former gap is not very significant as will become apparent later on.

The gap between the maximum of \( v(LR_{u}) \) over \( u \) and \( v(P) \) has to with the nonconvexities in the \( b \)-perturbation function associated with the quality constraints in \((P)\). This \( b \)-perturbation function may be defined by

\[
\phi_b(\theta) = \min_c \quad c^T \\
\text{s.t. } w_{ij}^* \leq c_{ij} + \theta_{ij} \\
\text{for each } i \text{ and } j \\
\text{and the constraints (2.1) to (2.7)}
\]
where $\theta$ is the vector of the $\theta_{ij}$'s, and $C^T$ and $\omega_{ij}^*$ are determined as before according to equations (2.8) to (2.22) as appropriate. Let us now relate $v(LR_u)$ to $\phi_b(\theta)$.

It is well known that

$$v(LR_u) \leq v(P) \text{ for all } u \geq 0.$$

That is,

$$\min \left[ C^T + \sum_{j=1}^{I_h} \sum_{i=1}^{I_l} u_{ij}(\omega_{ij}^* - \eta_{ij}) \right] \leq v(P)$$

s.t. the constraints (2.1) to (2.7)

or

$$\min \left[ C^T + \sum_{j=1}^{I_h} \sum_{i=1}^{I_l} u_{ij}(\omega_{ij}^* - (\eta_{ij} + \theta_{ij})) \right] \leq \phi_b(\theta)$$

s.t. the constraints (2.1) to (2.7)

or

$$v(LR_u) - u\theta \leq \phi_b(\theta) \text{ for all } u \geq 0.$$

This final result indicates that the function $\phi_b(\theta)$ lies above the hyperplane

$$v(LR_u) - u\theta = \emptyset$$

in the $(\theta, \phi)$ space, where $\emptyset$ refers to the coordinate for $\phi_b(\theta)$ and $v(LR_u)$ and $u$ are regarded as constants. Thus, one envisions an infinite family of such hyperplanes, each being defined by a unique $u$, and the lower envelope formed from the intersections of their half spaces containing $\phi_b(\theta)$ subtends $\phi_b(\theta)$. We also see that for each hyperplane

$$v(LR_u) - u\theta = \emptyset$$
there exists a

$$\hat{\theta} = \{\hat{\theta}_{ij} | \hat{\theta}_{ij} = \theta_{ij}^* - \eta_{ij} \text{ for each } i \text{ and } j\}$$

such that

$$v(LR_u) - u\hat{\theta} = \phi_b(\hat{\theta})$$

where the $\theta_{ij}^*$'s are the values of the $w_{ij}^*$'s at the optimum of $(LR_u)$. Thus the envelope we just introduced generally lies below $\phi_b(\theta)$ but meets $\phi_b(\theta)$ at some points. These meeting points are finite because values of $\phi_b(\theta)$ are finite; i.e., $\phi_b(\theta)$ corresponds with values of $C^T$ and $C^T$ is defined over binary decision variables. Geoffreon [6] shows that the lower convex envelope of $\phi_b(\theta)$ is precisely the lower envelope formed from the intersections of the half spaces noted above.

One now sees the relationship between $(LR_u)$ and $(P)$. That is, the construction $(LR_u)$ can at best find those points of $\phi_b(\theta)$ that are at the intersection between $\phi_b(\theta)$ and its lower convex envelope. All other points of $\phi_b(\theta)$ cannot be accessed by the construction $(LR_u)$. These inaccessible points constitute what is commonly called the "duality gap". Thus the nearness of the maximum of $v(LR_u)$ over $u$ to $v(P)$ depends on the extent to which $\phi_b(\theta)$ is convex at $\theta=0$. Or more practically speaking, this nearness depends on how convex $\phi_b(\theta)$ is, in general.

Because $(P)$ is a discrete program, $\phi_b(\theta)$ is monotonic nonincreasing and piece-wise linear relative to $\theta$. But $\phi_b(\theta)$ is almost always nonconvex for most discrete programs. For the minimum cost problem, this nonconvexity can be highly pronounced when instream treatment enters the solution. This is due to the clipping effect of instream
treatment which causes a sharp discontinuity in an otherwise linear problem. Linearizing this effect as we do in Chapter V does not help because this linearization introduces the big M. This M quantifies the greatest clip caused by instream treatment, where as Lagrangian relaxation attacks the clip directly, as the clip varies with the decision variables. In other words, the clipping effect of instream treatment has at least as bad an impact on the LP relaxation as it has on Lagrangian relaxation. In essence, we do have to contend with this effect, and we can expect a significant gap between \(v(LR_u)\) and \(v(P)\) in general. One way of minimizing this effect is to branch early on the instream treatment decision variables. But now we have the question: what is the gap between \(v(LR_u)\) and \(v(P)\) when the instream treatment decisions are each fixed at either zero or one?

For any fixed set of instream treatment decisions, problem \((P)\) reduces to an ILP if we make one basic assumption. That is, we assume the average annual pollutant load and the worst-case pollutant flows to each active instream processor are all nonnegative at the optimum. This is a fairly reasonable assumption which we point out later on. With this the case, \((LR_u)\) is also an ILP and in particular has the integrality property; i.e., the constraints \((2.1)\) to \((2.7)\) constitute a matrix whose basic solutions are each integral. In consequence, the maximum of \(v(LR_u)\) over \(u\) is equal to \(v(P)\) \([6]\), where \((P)\) is the LP relaxation of \((P)\). Thus the construction \((LR_u)\) can at best equal the LP relaxation when the instream treatment decision variables are fixed, and once again we are faced with a potential gap between \(v(LR_u)\) and \(v(P)\).
Nevertheless, because the construction \((LR_{ur})\) can be quickly solved and lends itself to the consideration of the special structures constituting problem \((P)\), we explore it. We note that there is no gap between the maximum of \(v(LR_{ur})\) over \(u\) and \(r\) and the maximum of \(v(LR_u)\) over \(u\), for any fixed set of instream treatment decisions. This follows directly when \((LR_u)\) has the integrality property.

We defer the proof that \((LR_u)\) with fixed instream treatment decisions has the integrality property. We now proceed with an overview of the branch and bound method.

**BRANCH AND BOUND OVERVIEW**

At the start of enumeration, we begin with one "vertex" where all the binary decision variables in \((P)\) are free. We then select some binary variable for branching and produce two vertices. We continue in this branching process producing several vertices. The term vertex here refers to a restricted problem of \((P)\) at a given stage in the branching process. That is, those variables we have previously branched upon to arrive at the current vertex are fixed at their branch values (zero or one) and the other variables are free in the restricted problem of \((P)\) at the current vertex. So the union of the restricted problems at all the vertices we have generated so far in the branching process constitute problem \((P)\).

We associate with each vertex the term "live" or "fathomed". A live vertex is one whose restricted problem remains to be resolved and
contrarily, a fathomed vertex is one whose restricted problem has previously been resolved. There are basically two ways by which we resolve a live vertex and change it to a fathomed vertex. The common way is we obtain a lower bound on the minimum cost of the restricted problem (the lower bound is regarded as infinity if the given restricted problem is infeasible) and find that it is not less than the cost of the current best feasible solution to (P). Note that the cost of a feasible solution to (P) is an upper bound on \( v(P) \), and that the current best feasible solution is the one with the least cost among the costs of all the feasible solutions we have generated so far for (P). The second way we fathom a live vertex is we solve the restricted problem of (P) at this vertex. This generally happens when all the decision variables are fixed and occasionally when the lower bound on the cost of the restricted problem is equal to the cost of a feasible solution to the restricted problem.

In order to fathom live vertices as we have indicated, we naturally require a method for generating feasible solutions to (P) as we journey through the branching process. One occurrence of a feasible solution is when all the decision variables are fixed at a live vertex. We already mentioned this. In addition, we use a heuristic method on restricted problems at live vertices to obtain feasible solutions to (P).

There are two issues involved in the guiding of the branching process. One is a strategic issue, and the other a tactical one. The strategic issue concerns the selection of the next live vertex among competing live vertices at any stage in the branching process. The idea
here is to choose that vertex that contains the optimal solution to (P).
The tactical issue concerns the selection of a free variable to branch
upon, after having decided on the live vertex. The common idea here is
to widen the gap between the expected minimum costs of the restricted
problems at the resulting two new vertices so that the worse vertex can
be fathomed as early as possible. We shall first discuss the tactical
implementation and then the strategic.

We branch first on the instream treatment decision variables, then on
the mining and (or) abatement decision variables at multiple effluent
mines, and lastly on the remaining decision variables. Branching first
on the instream treatment decision variables serves two purposes. One
is it enables us to nullify the clipping effect of instream treatment
upon the Lagrangian lower bound as early as possible. The second is
that problem (P) decomposes at active instream treatment processors into
individual ILP's that are interlinked only by the multiple effluent mine
constraints (2.3) to (2.7). Taking the Lagrangian breaks these
connections and we can compute the lower bounds separately for each
problem.

Having branched on all the instream treatment decision variables, we
begin breaking the multiple effluent mine links by branching on the
mining and (or) abatement decision variables at multiple effluent mines.
Eventually, when the instream treatment decisions and the mining and
abatement decisions at the multiple effluent mines are all fixed, we
will obtain a restricted problem of (P) that constitutes one or more
independent ILP's. Each of these ILP's is then solved entirely through
further branching on its free decision variables.
For selecting the instream treatment decision variable to branch upon when two or more such variables are free, we use a preassigned order. That is, we order the instream treatment decision variables in descending processor costs, so that we branch first on the most expensive variable, then on the next lower cost variable, and so on. This apriori tactic saves significant computation in our judgement. For to employ a dynamic tactic such as computing the lower bound at each possible successor vertex and choosing the instream treatment decision variable that contends the widest lower bound margin between its two successor vertices would involve quite extensive computations.

Regarding the choice between competing mining and (or) abatement decision variables at multiple effluent mines, the linear structure of the associated restricted problem provides readily the lower bound margins between successor vertices, in effect, we branch on the variable that provides the widest such margin. The same is true for the selection of the branching variable after having branched on all the mining and abatement decisions at the multiple effluent mines.

Consider now the strategic issue, that of choosing between live vertices. We had mentioned that the ideal here is to pick the vertex that has the optimal solution to (P). Naturally, this enables us to find a good upper bound on \( v(P) \) as early as possible, so that many vertices can be fathomed quickly. Of course, we can never know apriori the vertex that contains the optimal solution to (P). We can only guess. We have experimented with two types of guesses. One is simply based on the Lagrangian functional value. That is, we guess that the
live vertex that has the smallest Lagrangian value (lower bound) is the one that contains the optimal solution. Our second type of guess is more sophisticated but risking more computation as is typically the case. Here we use a heuristic to find a good feasible solution at each live vertex. We then estimate that the best feasible solution respectively at each vertex has a cost that is the sum of the Lagrangian value (lower bound) and a fraction of the difference between the feasible solution's cost and the lower bound. We have experimented with values in the range 0.6 to 0.9 for this fraction.

There is still one other strategy we have adopted. That is, when we do choose a vertex where the instream treatment decisions and the mining and abatement decisions at the multiple effluent mines are all fixed, we explode from this vertex until we entirely resolve this vertex. The idea here is twofold. One is, it permits programming ease to solve the independent ILP's at this vertex. The second is that it allows us to maintain a list of solved ILP's. Since the solution of each ILP is conditional only on the fixed mining and (or) abatement decisions at multiple effluent mines, it can be reimplemented each time the same conditions arise during the remainder of enumeration. This is akin to the implementation of the decomposition concept in ALCOT.

This completes the overview. We now proceed with methods to compute lower bounds for (P) and its restricted problems.
Recall that \( v(LR_{ur}) \) is a lower bound for \((P)\) or a restricted problem of \((P)\), given any \( u \geq 0 \) and an \( r \) that satisfies (4.2) to (4.9). In particular, we would like to find that value of \( u \) and \( r \) that maximizes \( v(LR_{ur}) \). But in practice, finding these values is computationally expensive, and one resorts to improving \( v(LR_{ur}) \) over \( u \) and \( r \) as cheaply as is reasonable. Nevertheless, a formal statement to maximize \( v(LR_{ur}) \) over \( u \) and \( r \) lends insight into how one can improve \( v(LR_{ur}) \) over \( u \) and \( r \).

The problem of maximizing \( v(LR_{ur}) \) over \( u \) and \( r \) can be expressed as the following linear program with many rows:
\[
\max \ g \\
\text{s.t.} \\
g \leq C^k + \sum_{j=1}^{J} \sum_{i=1}^{I} u_{ij}(w_{ij}^* - q_{ij}) \\
+ \sum_{m \in M^p} \sum_{j \in J_{km}} r^s_j(x_{jk}^s + x_{jk}^{sa} + x_{jk}^{st} + x_{jk}^{sat}) \\
+ \sum_{m \in M^p} \sum_{j \in J_{km}} r^d_j(x_{jk}^d + x_{jk}^{da} + x_{jk}^{dt} + x_{jk}^{dat}) \\
+ \sum_{m \in M^p} \sum_{j \in J_{km}} r^{sa}_j(x_{jk}^{sa} + x_{jk}^{sat}) \\
+ \sum_{m \in M^p} \sum_{j \in J_{km}} r^{da}_j(x_{jk}^{da} + x_{jk}^{dat}) \\
+ \sum_{m \in M^a} \sum_{j \in J_{km}} r^{a}_j(x_{jk}^{a} + x_{jk}^{at})
\]

for each \( k = 1 \) to \( K \)

and the constraints (4.2) to (4.9) \( (D) \)

where \( K = \) total number of combinations of the binary decisions per the constraints (2.1) and (2.2)

\( C^k = \) the total cost of decisions in combination \( k \)

discounting the costs of the \( y \) and \( z \) variables in \( C^T \), or

\[
C^k = C^T_k - \sum_{m=1}^{M} c_m
\]

\( C^T_k = C^T \) given the binary variables are fixed at combination \( k \), and

\( w_{ij}^* \) = \( i \)th worst-case flow past node \( j \) for decision combination \( k \).
and the subscript $k$ of each $x$ identifies the specific value of each $x$ in combination $k$. Note that the $y$ and $z$ variables are absent in the statement of (D). This is due to equations (4.2) to (4.9), and the fact that $C^R$ in (4.1) is to be minimized. We explain this later when we address the solution procedure for $(LR_{ur})$.

Among the many possible approaches to solve or nearly solve (D) as cited in the literature [5], we consider only the subgradient method. We discard the generalized linear programming approach because it would be too time consuming. Regarding the multiplier adjustment technique, preliminary investigation did not reveal any viable implementation. However, as an alternative to the subgradient method, we have devised a new technique which we call the "ascent method". This ascent method is based on the LP complementary slackness theorem and takes advantage of the recurring terms in the quality constraints. It is not as global in outlook as the subgradient method but accurate within its scope. However, it requires linearity of the Lagrangian problem and therefore does not apply when the instream treatment decisions are free to vary. Neither method is guaranteed to solve (D). We now describe the subgradient method and take up the ascent method later on.

**The Subgradient Method**

We see that the intersection of the hyperplanes defining the half spaces for $k=1$ to $K$ in (D) forms a piece-wise linear and concave upper envelope in the $(u,r,g)$ space, in effect, problem (D) translates to finding the
upper most point on this envelope. Further, the supporting hyperplane on this envelope at any given \( u \), say \( \bar{u} \), and an \( r \), say \( \bar{r} \), that satisfies the constraints (4.2) to (4.9), is precisely that obtained from solving \((LR_{uf})\). Hence, if \((\varphi^* - \eta)\) is the vector of the \((\varphi^*_{ij} - \eta_{ij})\)'s at the optimum of \((LR_{uf})\), and likewise, if \( \mathcal{X} \) refers to the sums

\[
\begin{align*}
\mathcal{X}_j^s + \mathcal{X}_j^{sa} + \mathcal{X}_j^{st} + \mathcal{X}_j^{sat} \\
\mathcal{X}_j^d + \mathcal{X}_j^{da} + \mathcal{X}_j^{dt} + \mathcal{X}_j^{dat} \\
\mathcal{X}_j^{sa} + \mathcal{X}_j^{sat} \\
\mathcal{X}_j^{da} + \mathcal{X}_j^{dat} \\
\mathcal{X}_j^{a} + \mathcal{X}_j^{at}
\end{align*}
\]

at each \( j \in J_{m} \) and \( m=1 \) to \( M \), which are optimal in \((LR_{uf})\), then \((\varphi^* - \eta)\) and \( \mathcal{X} \) form a subgradient of \( v(LR_{ur}) \) at \( u=\bar{u} \) and \( r=\bar{r} \) since \( v(LR_{ur}) \) relative to \( u \) and \( r \) really refers to the envelope just introduced and

\[
v(LR_{ur}) \leq v(LR_{uf}) + (\varphi^* - \eta)(u - \bar{u}) + \mathcal{X}(r - \bar{r})
\]

holds for all \( u \geq 0 \) and all \( r \) that satisfies (4.2) to (4.9). Considering now the step size for changing \( u \) in the direction \((\varphi^* - \eta)\) and \( r \) in the direction \( \mathcal{X} \), we need not limit ourselves to the range of \( u \) and \( r \) where the above relationship is an equality; rather we can extend beyond this range, for, after all, we are climbing a piece-wise linear hill. 

Because \( r \) must satisfy the constraints (4.2) to (4.9), we have constructed a specialized procedure for changing \( r \). The strategy we use
to obtain an approximate solution to (D) is we fix r and employ the subgradient method to change u, and then fix u to change r according to our special procedure. We will presently explain the details of this strategy. For now, we describe the subgradient method for changing u for a fixed r.

The subgradient method generates a sequence \( \{u^n\} \), where

\[
u^{n+1} = \max \{ u^n + t_n (w^n - q), 0 \}
\]

\( w^n \) is optimal in \( (LR_{ur}^n) \), and \( t_n \) is a positive scalar step size. The step size most commonly used in practice is

\[
t_n = \lambda_n (\bar{V} - v(LR_{ur}^n)) / ||w^n - q||^2
\]

where \( \lambda_n \) is a scalar satisfying \( 0 < \lambda_n \leq 2 \) and \( \bar{V} \) is an upper bound on the maximum of \( v(LR_{ur}^n) \) over u. Justification of this formula is given in [8]. Often the sequence \( \{ \lambda_n \} \) is determined by setting \( \lambda_0 = 2 \) and halving \( \lambda_n \) whenever \( v(LR_{ur}^n) \) has failed to increase in some fixed number of iterations.

At the head vertex, we first initialize r so it satisfies (4.2) to (4.9); we defer the description of this initialization. We also initialize u=0. We note now that problem \( (LR_{ur}) \) constitutes as many subproblems as there are watersheds. So, we can generate the sequence \( \{u^n\} \) individually for each subproblem. But we require the value of V for each subproblem. Recall that this V is an upper bound on the maximum of \( v(LR_{ur}^n) \) over u for the respective subproblem. This V can be taken as the cost of a solution that satisfies the constraints (2.1),
(2.2) and (2.23) in the subproblem's watershed. We use a heuristic to find such a solution for each subproblem. We then generate the sequence \{u^n\} for each subproblem by solving (LR^\text{}_{ur}) on every iteration and changing u per the subgradient. Regarding the sequence \{\lambda_n\}, we set \lambda_0 to 2. Based on intuitive considerations, we have experimented as follows. We halve \lambda_n if v(LR^\text{}_{ur}) fails to increase in 2 or 4 iterations. We terminate the sequence \{u^n\} if v(LR^\text{}_{ur}) fails to increase after successively halving \lambda_n 2 or 3 times or when the total number of iterations reaches 12, 18 or 24.

Having terminated the sequence \{u^n\} for each subproblem, we use a special procedure for changing r. Having changed r, we identify the subproblems affected by this change. Note that r refers to the mining and abatement cost allocations, and there is the possibility that these allocations may remain the same for some subproblems. Consequently, the old u, i.e., the terminal u from the preceding sequence \{u^n\}, is still satisfactory for these subproblems. So, new sequences \{u^n\} need be generated only for the affected subproblems. To generate these sequences, we again apply the feasible solution heuristic to each affected subproblem to update the corresponding V's, and then employ the subgradient method by starting with the terminal u from the preceding sequence. Our method for halving \lambda_n and terminating the sequence \{u^n\} remains as we discussed previously. Having terminated the new sequence \{u^n\} for each affected subproblem, we again contemplate the change of r. We continue in this iterative fashion for an arbitrary number of times or when r remains invariant between iterations. We have experimented
with 1 to 3 such iterations; each iteration here refers to generating the sequences \( u^n \) for each affected subproblem and then changing \( r \).

Having obtained a satisfactory \( u \) and \( r \) for the head vertex, we apply the feasible solution heuristic to each watershed to find an overall feasible solution to (P). As noted, this heuristic considers only the constraints (2.1), (2.2) and (2.23); i.e., it disregards the multiple effluent mine constraints (2.3) to (2.7). So, to ensure that these latter constraints are met, we fix the mining and(or) abatement decisions at multiple effluent mines before applying the heuristic. We detail these considerations later on.

The overall feasible solution to (P) not only provides an upper bound for (P) but also aids to establish \( V \) at successor vertices. To elucidate, let there be \( K \) subproblems at a successor vertex, and let \( L_k \) be equal to \( v(LR_{ur}) \) of the kth subproblem, where \( u \) and \( r \) are the final \( u \) and \( r \) obtained previously at the parent vertex. The initial lower bound at the successor vertex is simply the sum of the \( L_k \)'s taken over \( k \) in \( K \); let \( L \) denote this sum. Then \( V \) for the kth subproblem is taken as \( C + L_k - L \), where \( C \) is the total cost of the feasible solution to (P). This equation for \( V \) in general saves the use of the feasible solution heuristic for each subproblem at every vertex succeeding the head vertex. While \( V \) obtained in this way does not truly represent what it should, it suffices from the fathoming viewpoint.

Regarding our implementation of the subgradient method at vertices succeeding the head vertex, it is much the same as for the head vertex.
but with a few exceptions. One is the establishment of $V$ which we just indicated. A second is that the starting $u$ and $r$ is the parent vertex's terminal $u$ and $r$; but on occasions, we do modify this $u$ and $r$, which we address in a later section. A third is that we reduce the lengths of the sequences $\{u^n\}$ as we proceed down the enumeration tree. That is, we have experimented with lengths of 10, 14 or 18 for the middle portion of the tree and 8, 10 or 12 at the bottom portions of the tree. Lastly, at those vertices where the mining and abatement decisions at multiple effluent mines are all fixed, we do not contend with $r$; we merely generate the sequences $\{u^n\}$ for the respective subproblems.

In our judgement, $v(LR_{ur})$ is more sensitive to $u$ than $r$. Hence our more frequent change of $u$ than $r$. Secondly, we point out that there is no way of proving the optimality of (D) with the subgradient method unless by coincidence $v(LR_{ur})$ equals the cost of a feasible solution to (P).

We now describe the solution procedure for $(LR_{ur})$. We then consider the change of $r$ for a fixed $u$. After that we consider the ascent method.

**Solving the Lagrangian Problem for a given $u$ and $r**

The definition of the clipping function in Chapter II poses a difficulty here. To resolve this difficulty, we assume that for most real world data, the optimal solution in (P) is invariant over the following two models for instream treatment:
1. Instream treatment zeroes out positive worst-case acid flows and positive average annual acid load, but does not affect these quantities when they are negative.

2. Instream treatment zeroes out both worst-case acid flows and the average annual acid load regardless of the sign (positive or negative) of these quantities.

Then \( \text{(LR}_\text{ur}) \)'s objective function, i.e., \( C^R \), formulated earlier using model 1 above, can be simplified using model 2. How reasonable is the above assumption?

Let \( S \) be the feasible solution space of \( (P) \) and \( S_1 \) a subset of \( S \) where each solution specifying one or more active instream processors also produces nonnegative worst-case and average annual flows to each of these processors. Let \( S_1 \) also include solutions having no instream treatment. Then we see that for all solutions in \( S_1 \), the construction of \( (P) \) remains the same whether we use model 1 or 2 for instream treatment.

It is reasonable to expect \( S_1 \) to contain the solution that minimizes \( (P) \) for most real world data because instream treatment being very expensive can be economical only when significant pollution levels are treated. Moreover, natural alkaline flows are generally much smaller in magnitude relative to mine acid flows, and a situation where an active instream processor experiences negative pollution input would indicate either an excess of pollution control upstream or a waste of instream treatment.

Using model 2 for instream treatment, equations (2.11), (2.15) and (2.22) simplify respectively as follows:
Now problem \((P)\) translates to minimize \(C^T\) given by (4.12) subject to the constraints (2.1) to (2.7) and (2.23) where (4.12) is supported by equations (2.12) to (2.14), (4.11), and (2.16) to (2.21); and (2.23) is supported by (2.8) to (2.10) and (4.10). Consequently, \((LR_{ur})\) attains a simpler form which we describe next.

Let

\[ w_{ij}^* = (1 - x_j^z)(w_{ip_{1j}}^* + w_{ip_{2j}}^* + w_{ij}^n + w_{ij}^e) \]

for each \(i = 1\) to \(I_h^j\) and \(j \in J^t\) \hspace{1cm} (4.10)

\[ a_j^* = (1 - x_j^z)(a_{p_{1j}}^* + a_{p_{2j}}^* + a_j^n + a_j^e) \] for each \(j \in J^t\) \hspace{1cm} (4.11)

\[ C^T = \sum_{j=1}^{J} c_j^e + \sum_{m=1}^{M} c_m 
+ \sum_{j \in J^t} \left[ c_j^e + c_j^v(a_{p_{1j}}^* + a_{p_{2j}}^* + a_j^n + a_j^e)\right] x_j^z \] \hspace{1cm} (4.12)

Now problem \((P)\) translates to minimize \(C^T\) given by (4.12) subject to the constraints (2.1) to (2.7) and (2.23) where (4.12) is supported by equations (2.12) to (2.14), (4.11), and (2.16) to (2.21); and (2.23) is supported by (2.8) to (2.10) and (4.10). Consequently, \((LR_{ur})\) attains a simpler form which we describe next.

Let

\[ T_j = \text{the ordered set of potential instream treatment sites encountered along the path from node } j \text{ to the watershed outlet node; i.e.,} \]

\[ T_j = \{k \mid k = j \text{ or } k \text{ is downstream from } j, \ k \in J^t\} \]

\[ u_{ij}^* = \text{sum of the Lagrangian multipliers for the } \]

\[ \text{ith worst-case over all nodes downstream from } j \text{(including } j)\), i.e., \]

\[ u_{ij}^* = \sum_{k=j}^{J} u_{ik} \] \hspace{1cm} (4.13)

or \(k\) downstream from \(j\)
$u^e_{ij} = \text{Effective sum of the Lagrangian multipliers for the } \text{ith worst-case for node } j; \text{ i.e., the sum over all nodes downstream from } j \text{ until the first active instream processor, or}$

$$u^e_{ij} = u^*_{ij} - x^z_{k_1} u^*_{ik_1}$$

$$- (1 - x^z_{k_1}) x^z_{k_2} u^*_{ik_2} - \ldots - (1 - x^z_{k_1}) \ldots (1 - x^z_{k_{n-1}}) x^z_{k_n} u^*_{ik_n}$$

where $k_1, \ldots, k_n$ are the ordered sequence in $T_j$ \hfill (4.14)

$$x^e_j \begin{cases} 1 \text{ if } x^e_k = 1 \text{ for any } k \in T_j \\ 0 \text{ otherwise} \end{cases}$$ \hfill (4.15)

$c^s_j = r^s_j, c^sa_j = r^sa_j, c^d_j = r^d_j, c(da) = r(da)$

be defined for each $j \in J_{sm}$ and $m \in M^p$

$c^a_j = r^a_j$ be defined for each $j \in J_{sm}$ and $m \in M^a$

$$e^s_j = c^s_j + c^va_j x^e_j + \sum_{i=1}^{I_{hj}} w^s_{ij} u^e_{ij} \text{ for each } j \in J^p$$ \hfill (4.16)

$$e^{sa}_j = c^s_j + c^sa_j + c^va_j x^e_j + \sum_{i=1}^{I_{hj}} w^{sa}_{ij} u^e_{ij} \text{ for each } j \in J^p$$ \hfill (4.17)

$$e^{st}_j = c^s_j + c^st_j + c^va_j \text{ for each } j \in J^p$$ \hfill (4.18)

$$e^{sat}_j = c^s_j + c^sa_j + c^st_j + c^va_j \text{ for each } j \in J^p$$ \hfill (4.19)

$$e^d_j = c^d_j + c^va_j x^e_j + \sum_{i=1}^{I_{hj}} w^d_{ij} u^e_{ij} \text{ for each } j \in J^p$$ \hfill (4.20)

$$e^{da}_j = c^d_j + c^da_j + c^va_j x^e_j + \sum_{i=1}^{I_{hj}} w^{da}_{ij} u^e_{ij} \text{ for each } j \in J^p$$ \hfill (4.21)
\[ e^d_j = c^d_j + c^dt_j + c^v a^d_j \text{ for each } j \in J^p \] (4.22)

\[ e^{dat}_j = c^d_j + c^{dat}_j + c^{v a^{dat}}_j \text{ for each } j \in J^p \] (4.23)

\[ e_j = c^v a^x_j x^e_j + \sum_{i=1}^{I_{h_j}} w_{ij} u^e_{ij} \text{ for each } j \in J^a \] (4.24)

\[ e^a_j = c^a_j + c^v a^x_j x^a_j + \sum_{i=1}^{I_{h_j}} w_{ij} u^e_{ij} \text{ for each } j \in J^a \] (4.25)

\[ e^t_j = c^t_j + c^v a^x_j \text{ for each } j \in J^a \] (4.26)

\[ e^{at}_j = c^a_j + c^t_j + c^v a^x_j \text{ for each } j \in J^a \] (4.27)

Then \((LR_u r)^u\)'s objective using model 2 for instream treatment takes the form

\[
C^L = \sum_{j \in J^p} (x^{s e}_{j e j} + x^{s a}_{j e j} + x^{s t}_{j e j} + x^{s a t}_{j e j} \\
+ x^{d e}_{j e j} + x^{d a}_{j e j} + x^{d t}_{j e j} + x^{d a t}_{j e j} \\
+ \sum_{j \in J^a} ((1 - x^a_j - x^t_j - x^{at}_j) e_j + x^{a e}_j + x^{a t}_j + x^{a t e}_j) \\
+ \sum_{j=1}^{J} c^v a^{n e}_{j e j} + \sum_{j=1}^{J} \sum_{i=1}^{I_{h_j}} w_{ij} u^e_{ij} \\
+ \sum_{j=1}^{J} c^z x^z_j - \sum_{j=1}^{J} \sum_{i=1}^{I_{h_j}} u_{ij} q_{ij} \] (4.28)

Note that the sum of \(c_m\) over \(m=1\) to \(M\) in \(C^T\) and the summations over the multiple effluent mine constraints' Lagrangian terms in \(C^R\) have all
disappeared. This is partly because the variables $y_m^s, y_m^d, c_{bm}^s, c_{bm}^d$ and $c_{bm}^a$, for each $b$ and $m$, all vanish from the objective $C^R$. That is, these variables appear only in the summation of the $c_m$'s in $C^T$; and because of (4.2) to (4.9), they cancel out in $C^R$. This cancellation is clear for $y_m^s, y_m^d, z_{bm}^s, z_{bm}^d$, and $z_{bm}^a$ at each appropriate $m$. Regarding $z_{bm}^sa, z_{bm}^da$ and $z_{bm}^a$, for each $b=1$ to $B_m-1$ and each appropriate $m$, they are all involved in isolated terms with nonnegative coefficients; and since $C^R$ is to be minimized, they can all be set to zero. Also the new definitions for $c_{j}^s, c_{j}^a, c_{j}^d, c_{j}^da$, and $c_{j}^a$, at each appropriate multiple effluent mine node, permit the rearrangement of the multiple effluent mine constraints' Lagrangian terms involved in the summations in $C^R$. Thus the Lagrangian multipliers in $r$ are actually the allocations of the mining and (or) abatement costs at multiple effluent sources among their respective nodes.

Given a $u \geq 0$ and an $r$ that satisfies the constraints (4.2) to (4.9), problem (LR$_{ur}$) translates to: minimize $C^L$ as given by (4.28) subject to the multiple choice constraints (2.1) and (2.2). Note that the symbol $e$ now identifies the cost of each mine source decision. We have used this symbol to mean "effective" cost.

Note first of all that for any fixed set of instream treatment decisions, $u_{ij}^e$ for each $i$ and $j$ and $x_j^e$ at each $j$ are constants. Consequently, the effective costs are constants. In turn, (LR$_{ur}$) constitutes $J$ independent subproblems; i.e., as many subproblems as there are nodes. The subproblem at any node is solved simply by picking the decision that has the least effective cost. In essence, (LR$_{ur}$) can
be solved readily at all vertices where the instream treatment decisions are fixed. Regarding the remaining vertices where one or more instream treatment decisions are free to vary, \((LR_{ur})\) can again be solved quite readily. To begin with, note that \(u^e_{ij}\) for each \(i\) and \(j\) and \(x^e_j\) at each \(j\) depends only on the first active instream treatment site in the ordered sequence \(T_j^*\). To elucidate, if \(k_1, k_2, \ldots, k_n\) is the ordered sequence in \(T_j\), and if the processor at \(k_1\) is active, then \(u^e_{ij}\) and \(x^e_j\) are invariant with respect to the presence or absence of instream treatment at any of the sites \(k_2\) to \(k_n\). Similarly, if site \(k_1\) is inactive but \(k_2\) is active, then again \(u^e_{ij}\) and \(x^e_j\) are invariant relative to the presence or absence of active processors at sites \(k_3\) to \(k_n\). This inductive reasoning reveals that there are at most \(n+1\) possible dictates on \(u^e_{ij}\) and \(x^e_j\) corresponding to the \(n\) distinct nearest active downstream instream treatment sites plus the case when none of the \(n\) sites is active, where \(n\) is the cardinality of \(T_j\). Correspondingly, there are \(n+1\) possible dictates on the effective costs at node \(j\). For each such dictate, the optimal source decision is that which has the least effective cost. For explicitness, let us define

\[
d_{jk} = \text{the mine source decision that is optimal at node } j \text{ when } k \text{ in } T_j \text{ is the nearest active downstream instream processor; } k=0 \text{ when none of the processors in } T_j \text{ are active.}
\]

and \(c_{jk} = \text{effective cost of } d_{jk}^*\).

Let us now collect nodes having identical \(T_j\)'s. In fact, each such collection can be associated with a distinct instream treatment site.
That is, for each \( j \in J^t \), all nodes upstream of \( j \) and including \( j \) whose first downstream instream treatment site is at \( j \) have identical \( T_j \)'s; all other nodes do not have this \( T_j \). Let

\[
U_j = \{ f | f = j \text{ or } f \text{ is upstream from } j \text{ and } T_f = T_j \}
\]

for each \( j \in J^t \) and \( j \neq \emptyset \). Let \( c_{jk} = \sum_{f \in U_j} (c_{fk} + c_{nfk} + c_{nf}) \) for \( k = 0 \), each \( k \) in \( T_j \) and each \( j \in J^t \)

where \( c_{nfk} = \begin{cases} c_{nf} & \text{if } k \text{ is the first active instream processor in } T_f \\ 0 & \text{if } k = 0 \text{ implying none in } T_f \text{ are active} \end{cases} \)

\[
I_h, j = \sum_{i=1}^{j} w_{if} u_{if}^* \text{ when } k \text{ is the first active instream processor in } T_f
\]

and

\[
c_{nfk} = \begin{cases} I_h, j & \text{if } k = 0 \text{ implying none in } T_f \text{ are active} \\ \sum_{i=1}^{j} w_{if} u_{if}^* & \text{if } k \neq 0 \text{ and } k \text{ is the first active instream processor in } T_f \end{cases}
\]

Note that \( C_{jk} \) is a constant for each \( j \) and \( k \). Now the following procedure solves \((LR_{ur})\) and is self explanatory.

1. Consider each instream treatment site that has no other such site upstream, and perform this step for each of these sites in turn. Let \( j \in J^t \) be the current site. If \(|T_j| > 1\), let \( j_k \) be the \( k \)th downstream site in \( T_j \) (\( j_1 = j \), \( j_2 \) is the next instream treatment downstream from \( j \), and so on), and set

\[
a) \quad C_{jj_k}^* = \min (C_{jj} + c_{j}, C_{jj_k})
\]

for \( j_k = 0 \) and for \( k = 2 \) to \(|T_j|\).
and b) \( C_{j_k^1} = C_{j_k^1} + C_j^* \)

for \( j_k = 0 \) and for \( k = 2 \) to \( |T_j| \).

If \( |T_j| = 1 \), set

\[
C_{j^*} = C_{j^*} + \min \left( C_{j^0} + c_j^z, C_{j^0} \right)
\]

where \( C_{j^*} \) accumulates \( v(L_{ur}) \) and is initialized to zero.

2. Flag all instream treatment sites considered previously, and collect all the unflagged instream treatment sites that each has no other unflagged site upstream. Perform step 1 for each of these sites in turn. Repeat this step as many times as is necessary until all the instream treatment sites are flagged.

Update

\[
C_{j^*} = C_{j^*} + \sum_{\text{over each } j^0} (c_{j^0} + c_{j^0}^{nw})
\]

where \( T_j = \emptyset \)

Thus \( C_{j^*} \) obtained at the end of procedure (R4.1) equals \( v(L_{ur}) \) and an appropriate bookkeeping of decisions during procedure (R4.1) can reveal the optimal decisions in \( (L_{ur}) \).

**Changing \( r \) for a Fixed \( u \)**

Recall that this step forms a part of the procedure to improve \( v(L_{ur}) \) over \( u \) and \( r \). That is, we first fix \( r \) and change \( u \), and then fix \( u \) to institute this change to \( r \). But we render this change to \( r \) only at those vertices where at least one mining or abatement decision at a multiple effluent mine is free to vary.
We mentioned earlier that at the head node, we initialize \( r \) so it satisfies the constraints (4.2) to (4.9). We address this initialization later. For the present, assume that the given \( r \), say \( p \), satisfies the constraints (4.2) to (4.9). Our task now is to change \( r \) from \( p \) to \( r \) so that \( v(LR_{ur}) \) improves and that the constraints (4.2) to (4.9) continue to hold.

Recall that if \( \xi \) is optimal in \( (LR_{uf}) \) and \( \xi \) refers to the sums

\[
\begin{align*}
\xi^s_j + \xi^{sa}_j + \xi^{st}_j + \xi^{sat}_j & \text{ at each } j \in \mathcal{J}_m \text{ and } m \in \mathcal{M}^p \\
\xi^d_j + \xi^{da}_j + \xi^{dt}_j + \xi^{dat}_j & \text{ at each } j \in \mathcal{J}_m \text{ and } m \in \mathcal{M}^p \\
\xi^{sa}_j + \xi^{sat}_j & \text{ at each } j \in \mathcal{J}_m \text{ and } m \in \mathcal{M}^p \\
\xi^{da}_j + \xi^{dat}_j & \text{ at each } j \in \mathcal{J}_m \text{ and } m \in \mathcal{M}^p \\
\xi^a_j + \xi^{at}_j & \text{ at each } j \in \mathcal{J}_m \text{ and } m \in \mathcal{M}^a
\end{align*}
\]

then \( \xi \) is a subgradient of \( v(LR_{ur}) \) at \( r=p \); i.e.,

\[
v(LR_{ur}) \leq v(LR_{uf}) + \xi(r-p)
\]

is true for all \( r \) that satisfies (4.2) to (4.9). It is clear that \( \xi \) is a vector of zeroes and ones. So, in order that the change we render to \( r \) conforms with the subgradient \( \xi \), we cannot decrease (we may increase) those elements of \( r \) whose corresponding elements in \( \xi \) are ones; but the other elements of \( r \), we may increase or decrease. With this simple rule in mind, we institute the change to \( r \) from \( p \) to \( r \) such that (4.2) to (4.9) continue to hold, and such that
\[ v(LR_{ur}) \leq v(LR_{ur}|n \in N) = \max_r v(LR_{ur}|n \in N) = v(LR_u|n \in N) \]

where \( N \) = the set containing all combinations of the instream treatment decisions

\( n \) = the particular combination of the instream treatment decisions that is optimal in \( LR_{ur} \)

\( LR_{ur}|n \in N) = (LR_{ur}) \) with the restriction that the instream treatment decisions are fixed at combination \( n \)

and \( LR_u|n \in N) = (LR_u) \) with the restriction that the instream treatment decisions are fixed at combination \( n \).

The basic idea here is to use the integrality property of problem \( LR_u|n \in N \). This property indicates that the basic solutions associated with the constraint matrix defined by (2.1) to (2.7) are all integral, in consequence, there must exist an \( \tau \) that corresponds with the dual variable values of (2.3) to (2.7) at the LP optimum of \( LR_u|n \in N \). So we set about finding this \( \tau \). Specifically, all we are required to prove is that \( v(LR_{ur}|n \in N) = v(LR_u|n \in N) \).

We digress here to mention an important point. All that \( \tau \) assures us is that \( v(LR_{ur}|n \in N) \geq v(LR_{uf}) \). However, when one or more instream treatment decisions in problem \( LR_{ur} \) are free to vary, it is possible that a different combination of treatment decisions, other than \( n \), is optimal in \( LR_{ur} \). In consequence, \( v(LR_{ur}) \) may be less than \( v(LR_{uf}|n \in N) \). In essence, there is the possibility that \( v(LR_{ur}) \) is less than \( v(LR_{uf}) \), which is contrary to our objective. However, by keeping the norm of \( \tau - \tilde{\tau} \) relatively small and by ensuring that \( \tau - \tilde{\tau} \) is in line with the subgradient \( \tilde{\tau} \), we can minimize the occurrence of this undesirable possibility. Note that this occurrence is theoretically
possible only when one or more instream treatment decisions are free to vary; at other vertices where the instream treatment decisions are fixed but where one or more mining and(or) abatement decisions at multiple effluent mines are free to vary, we do not contend with it. For at these latter vertices, the set $N=\{n\}$ implying $v(L_{ur})$ equals $v(L_{ur}|n \in N)$.

We are now concerned with going from $f$ to $\tilde{f}$ such that 1) the step is in line with $\delta$, 2) the constraints (4.2) to (4.9) continue to hold, and 3) $v(L_{ur}|n \in N) = v(L_{u}|n \in N)$. Of course it is possible that $v(L_{ur})$ equals $v(L_{u}|n \in N)$, in which event, we need not change $f$. So, hereafter, we assume this is not the case.

Let us compare problems $(L_{ur}|n \in N)$ and $(L_{u}|n \in N)$. We had mentioned that $(L_{ur}|n \in N)$ constitutes as many independent subproblems as there are nodes, namely $J$ subproblems, and that each subproblem is solved simply by selecting the decision with the least effective cost (the effective costs are defined in (4.16) to (4.27)). Likewise, problem $(L_{u}|n \in N)$ constitutes a number of independent subproblems but with a difference. That is, these subproblems are defined at the individual mine sources rather than at the nodes. However, the subproblems defined at the single effluent sources are identical to those in $(L_{ur}|n \in N)$ that are defined at the corresponding single effluent mine nodes; so we can eliminate these problems out of the ensuing analysis. Our concern then is with matching the multiple effluent mine nodes' subproblems in $(L_{ur}|n \in N)$ with the multiple effluent mine source subproblems in $(L_{u}|n \in N)$. 
Consider first an abandoned or active multiple effluent mine source, say \( m \in \mathbb{M} \). The subproblem of \((LR_u|n \in N)\) at this source has the form

\[
\min C_m = \sum_{b=1}^{B_m} c_{bm}^a z_{bm}^a + \sum_{j \in J_m^*} (e_j(1 - x_j^a - x_j^t - x_j^{at}) + e_j^a x_j^a + e_j^t x_j^t + e_j^{at} x_j^{at})
\]

s.t. \( x_j^a + x_j^t + x_j^{at} \leq 1 \) for each \( j \in J_m^* \)

and \( x_j^a + x_j^{at} = z_{bm}^a + \ldots + z_{B_m}^a \) for each \( j \in J bm \) and \( b=1 \) to \( B_m \)

where \( e_j^a, e_j^t, e_j^{at} \) and \( e_j^{at} \) are as defined in equations (4.24) to (4.27) except that \( c_j^a \) in \( e_j^a \) and \( e_j^{at} \) is taken as zero. We note that 1) any one \( z \) variable can be set to unity or all must be set to zero corresponding to the \( B_m+1 \) abatement alternatives, and 2) for any fixed set of values for the \( z \) variables, the subproblem decomposes further into as many problems as there are nodes in \( J_m^* \). Thus the following procedure solves the above subproblem:

1. Compare \( z_{1m}^a = z_{2m}^a = \ldots = z_{B_m}^a = 0 \) against \( z_{1m}^a = 1 \) and

\[
\sum_{j \in J_{1m}} \min(e_j^a, e_j^t) < c_{1m}^a + \sum_{j \in J_{1m}} \min(e_j^a, e_j^{at})
\]

For any fixed set of values for the \( z \) variables, the subproblem decomposes further into as many problems as there are nodes in \( J_m^* \). Hence if

\[
\sum_{j \in J_{1m}} \min(e_j^a, e_j^t) < c_{1m}^a + \sum_{j \in J_{1m}} \min(e_j^a, e_j^{at})
\]
we would choose the former combination; or else the latter. If the quantities are equal, we can choose either. Let \( c^* \) be the current best cost; i.e., \( c^* = e_{1m} \) if we choose the former or \( c^* = c_{1m} + e_{2m} \) otherwise.

2. Compare \( z^a_{2m} = z^a_{3m} = \ldots = z^a_{B_m} = 0 \) against \( z^a_{2m} = 1 \) and similarly. That is, if

\[
\begin{align*}
\frac{c^*}{e^a_{2m}} + \sum_{j \in J_{2m}} \min(e^t_j, e^a_j) < c^a_{2m} + e^a_{2m} + \sum_{j \in J_{2m}} \min(e^a_j, e^a_j)
\end{align*}
\]

choose the former and update \( c^* = c^* + e^a_{2m} \). Otherwise choose the latter and set \( c^* = c^a_{2m} + e^a_{1m} + e^a_{2m} \).

3. Continue in the above fashion until all abatement levels are exhausted. That is, at any abatement level \( b \), choose the combination associated with the smaller of \( c^* + e_{bm} \) and \( c^a_{bm} + e^a_{1m} + \ldots + e^a_{bm} \), and update \( c^* \) accordingly.

\[(R4.2)\]

A pictorial way to interpret procedure \((R4.2)\) is to regard

\[
\sum_{b=1}^{B_m} \sum_{j \in J_{bm}} \min(e^a_j, e^a_j)
\]  

as the origin for the objective \( C_m \) and represent alternative \( C_m \) values by constructing two strips as shown in Figure 3, where the top strip identifies the abatement costs at levels \( 1 \) to \( B_m \) \((B = 6 \) in the figure\), the bottom strip portrays the costs of not abating at levels \( 1 \) to \( B_m \), and
Figure 3. Optimum Abatement Level Representation at a Multiple Effluent Mine.

$$
\varepsilon_{bm}^{a} = \sum_{k=1}^{b} \sum_{j \in J_{km}} \varepsilon_{j}
$$

$$
\varepsilon_{j} = \min(e_{j}, e_{j}^{t}) - \min(e_{j}, e_{j}^{at})
$$

In the figure, $C_{m}$ relative to the stipulated origin can be taken as the length between point 0 of the top strip and point $B'_{m}$ of the bottom strip.
after vertically aligning 0 with 0', or 1 with 1', and so on, depending upon whether we wish no abatement, or abatement at level 1, or abatement at respective higher levels. The optimum abatement level is that which gives the least cost length. In Figure 3, the optimum abatement level is 3. Note that 0 is to the right of 0', 1 to the right of 1', and so on. In general, the distance b-b' for b=0 to Bm gives the cost increase over the optimum when b is the chosen level of abatement. Thus, letting h be the optimum abatement level, the following inequalities are true:

\[ c_{hm}^a - c_{bm}^a \leq e_{hm}^a - e_{bm}^a \quad \text{for each } b \text{ in } 0 \leq b < h \tag{4.29} \]

\[ c_{bm}^a - c_{hm}^a \geq e_{bm}^a - e_{hm}^a \quad \text{for each } b \text{ in } h \leq b < B_m \tag{4.30} \]

Our aim now is to use (4.29) and (4.30) and find \( r_{j}^a \) at each \( j \in J_{*m} \) such that

\[ r_{j}^a \leq e_{j}^a \quad \text{at each } j \in J_{bm} \text{ and } b < h \tag{4.31} \]

\[ r_{j}^a \geq e_{j}^a \quad \text{at each } j \in J_{bm} \text{ and } b > h \tag{4.32} \]

\[ \sum_{b=1}^{h} \sum_{j \in J_{bm}} r_{j}^a = c_{hm}^a \tag{4.33} \]

\[ \sum_{b=h+1}^{B_m} \sum_{j \in J_{bm}} r_{j}^a = c_{B_m}^a - c_{hm}^a \tag{4.34} \]

and \( \sum_{k=1}^{b} \sum_{j \in J_{km}} r_{j}^a \leq c_{bm}^a \) for \( b=1,2,\ldots,h-1,h+1,\ldots,B_m-1 \) \( (4.35) \).
Equations (4.31) to (4.33) assure that \( v(LR_{u|n\in N}) \) equals \( v(LR_{u|n\in N}) \) for the subproblem at \( m \), and equations (4.33) to (4.35) imply that (4.6) and (4.9) hold at \( m \). Note that our eventual aim is to regard \( \tau_j^a \) relative to \( \tau_j^a \) so that we not only establish (4.31) to (4.35) but also ensure that the change \( \tau - \tau \) is in line with the subgradient \( \tau \). But for the moment, let us prove the existence of (4.31) to (4.35).

Consider first the problem of achieving (4.31), (4.33) and (4.35) for \( b=1 \) to \( h-1 \). Suppose, with reference to Figure 3, the interval 0 to 1 is not greater than the interval 0' to 1'. This implies \( c_{lm}^a < e_{lm}^a \). Clearly now, if we set

\[
\tau_j^a = c_{lm}^a (e_j^a/e_{lm}^a) \quad \text{for each } j \in J_{lm} \tag{4.36}
\]

then (4.31) would be achieved for each \( j \in J_{lm} \) and (4.35) would hold at \( b=1 \) since the sum of the \( \tau_j^a \)'s over \( J_{lm} \) equals \( c_{lm}^a \). Moreover, the problem that remains, namely the problem of distributing \( c_{hm}^a - c_{lm}^a \) across the \( \tau_j^a \)'s at nodes belonging to levels 2 to \( h \) such that (4.31) holds at these nodes and (4.35) holds for \( b=2 \) to \( h-1 \), has a structure identical to that of the original problem, because (4.29) continues to hold at \( b=1 \) to \( h-1 \). Therefore the allocations to the \( \tau_j^a \)'s as portrayed in (4.36) are consistent with an inductive approach when \( c_{lm}^a < e_{lm}^a \). On the other hand, when \( c_{lm}^a > e_{lm}^a \), we can set \( \tau_j^a = \gamma_j \) at each \( j \in J_{lm} \) since it satisfies (4.31) for each \( j \in J_{lm} \), satisfies (4.35) at \( b=1 \), and renders the remaining problem, that of allocating \( c_{hm}^a - e_{lm}^a \) across the \( \tau_j^a \)'s at nodes belonging to levels 2 to \( h \) such that (4.31) holds at these nodes and (4.35) holds at \( b=2 \) to \( h-1 \), to be identical in structure to that of
the original problem. Note that we can conceive the cost of abating at
level 1 to be equal to \( e_{1m}^a \) so that point 1 in Figure 3 shifts to the
left by the distance \( c_{1m}^a - e_{1m}^a \). Even with this shift, point 1 remains to
the right of 1' because 0 is to the right of 0' to begin with. Thus the
stated remaining problem possesses a structure identical to that of the
original problem indicating that the allocations \( r_j^a = e_j \) at each \( j \in J_{1m} \)
are consistent with an inductive approach. In general, by ordering the
abatement levels from 1 to \( h \) and using the formula

\[
 r_j^a = (c_{bm}^a - \sum_{k=1}^{b-1} \sum_{j \in J_{km}} r_j^a) \left( e_j / (e_{bm}^a - e_{b-1,m}^a) \right)
\]

at each \( j \in J_{bm} \) \hspace{1cm} (4.37)

if \( c_{bm}^a - \sum_{k=1}^{b-1} \sum_{j \in J_{km}} r_j^a < e_{bm}^a - e_{b-1,m}^a \)

or the formula

\[
 r_j^a = e_j \hspace{1cm} \text{at each } j \in J_{bm} \hspace{1cm} (4.38)
\]

if \( c_{bm}^a - \sum_{k=1}^{b-1} \sum_{j \in J_{km}} r_j^a > e_{bm}^a - e_{b-1,m}^a \)

we can achieve (4.31), (4.33) and (4.35) for \( b=1 \) to \( h-1 \). Note that at
\( b=h \), the formula that applies is (4.37) since point \( h-1 \) would be to the
right of \( (h-1)' \) per our inductive reasoning.

Similarly, regarding the proof for (4.32), (4.34) and (4.35) for \( b>h \),
there are two possibilities to begin with: either \( c_{B_m}^a - c_{B_m-1,m}^a > e_{B_m}^a - e_{B_m-1,m}^a \)
or otherwise. In the former case, setting
\[ r_j^a = (c_{B_m}^a - c_{B_m - 1}^a, j)(e_j/(e_{B_m}^a - e_{B_m - 1, m, j})) \] for each \( j \in J_{B_m} \) \hspace{1cm} (4.39)

achieves (4.32) for each \( j \in J_{B_m} \) and assures (4.35) at \( B_m - 1 \) after assuming (4.34). Moreover, the problem that remains, namely that of allocating \((c_{B_m - 1}^a - c_{B_m}^a, j)\) at nodes belonging to the levels \( B_m - 1 \) such that (4.32) holds at these nodes and (4.35) holds for \( b = H + 1 \) to \( B_m - 2 \), has the same structure as the original problem because of (4.30). Hence the allocations in (4.39) are consistent with an inductive approach. Regarding the latter case, when \( c_{B_m}^a - c_{B_m - 1}^a, j \geq e_{B_m}^a - e_{B_m - 1, m, j} \), setting \( r_j^a = e_j \) at each \( j \in J_{B_m} \) assures (4.32) for each \( j \in J_{B_m} \), assures (4.35) at \( B_m - 1 \) after assuming (4.34), and agrees with an inductive approach. Note that \( c_{B_m}^a - c_{B_m - 1}^a, j \) may be conceived as being decreased by \( e_{B_m}^a - e_{B_m - 1, m, j} \) minus \( c_{B_m}^a - c_{B_m - 1, m, j} \), in effect, point \( B_m - 1 \) as portrayed in Figure 3 will shift to the left by this decrease. But even after this shift, \( B_m - 1 \) will be to the right of \( B_m' - 1 \) because \( B_m \) is to the right of \( B_m' \) to begin with. In general, we can achieve (4.32), (4.34) and (4.35) for \( b > h \), by ordering the abatement levels backward from \( B_m \) to \( h + 1 \), and using the formula

\[ r_j^a = (c_{B_m}^a - \sum_{k=b+1}^{B_m} \sum_{j \in J_{km}} r_j^a - c_{b-1, m, j}) (e_j/(e_{B_m}^a - e_{b-1, m})) \] for each \( j \in J_{B_m} \) \hspace{1cm} (4.40)

if \( c_{B_m}^a - \sum_{k=b+1}^{B_m} \sum_{j \in J_{km}} r_j^a - c_{b-1, m, j} \geq e_{B_m}^a - e_{b-1, m} \)

or the formula (4.38) otherwise. Note that when \( b \) eventually is \( h + 1 \), point \( h + 1 \) will be to the right of point \( h' + 1 \), so that (4.40) applies and (4.34) will be met.
The above inductive approach applies for the initialization of \( r \) at each \( m \in \mathbb{N}^d \) at the head vertex. Recall that at the head vertex, \( u \) is initialized to zero. For \( u=0 \), it turns out that all instream processors are best turned off, in effect, it is straightforward to compute \( e_j \) at each abandoned or active multiple effluent mine node. Thus, the above inductive approach can be applied.

Consider next the change from \( \mathbf{t} \) to \( \mathbf{r} \) at any \( m \in \mathbb{N}^d \) given that \( \mathbf{t} \) satisfies (4.6) and (4.9), and such that \( \mathbf{r} \) satisfies (4.31) to (4.35) and that the change \( \mathbf{r} - \mathbf{t} \) is along the subgradient \( \mathbf{t} \). Since \( \mathbf{t} \) satisfies (4.6) and (4.9), it must also satisfy (4.35). However, it may violate (4.31) to (4.34).

Consider first the change from \( \mathbf{t} \) to \( \mathbf{r} \) such that 1) (4.31) holds, 2) (4.33) holds, 3) (4.35) continues to hold for \( b<h \) and 4) the change \( \mathbf{r} - \mathbf{t} \) is in line with the subgradient \( \mathbf{t} \). Let

\[
J_{b+1}^V = \{ j \mid j \in J_{b+1}^V, t_j > e_j \} \text{ for } b=0 \text{ to } h
\]

denote the set of nodes where (4.31) is violated. We also know that at these nodes, the corresponding elements in \( \mathbf{f} \) are all zeroes, in effect, we can decrease \( r_j \) relative to \( t_j \) at these nodes. In particular, let us set

\[
r_j = e_j \text{ at each } j \in J_{b+1}^V \text{ and } b<h
\]  

(4.41)

Considering now the \( r_j \)'s at each \( j \in J_{b+1}^V \) and \( b<h \), and the \( r_j \)'s as given by (4.41), we note that they continue to satisfy (4.35) for \( b<h \) and that they also satisfy (4.31). However, the L.H.S. of (4.33) may be
less than the R.H.S. of (4.33). So, our task is to close the gap between the two sides of (4.33) by increasing $r^a_j$ relative to $t^a_j$ at one or more nodes in $J_{bm}^V - J_{bm}^V, b=1$ to $h$. Note that the elements in $\mathfrak{R}$ corresponding to the nodes in $J_{bm}^V - J_{bm}^V, b=1$ to $h$, are all ones. So the contemplated increases are in line with the $\mathfrak{F}$. But we must also assure that (4.31) and (4.35) for $b<h$ both continue to hold.

Fortunately, the task of interest can be accomplished because of (4.29) as illustrated in Figure 3. Consider the following translation to the quantities $c^a_{bm}$ and $e^a_{bm}$ for $b<h$:

$$
c^a_{bm} = c^a_{bm} - t_b \quad \text{for } b<h
$$

$$
e^a_{bm} = e^a_{bm} - t_b \quad \text{for } b<h
$$

where

$$
t_b = \sum_{k=1}^{b} \sum_{j \in J_{km}^V} s_j + \sum_{j \in J_{km}^V - J_{km}^V} f^a_j
$$

Clearly because of (4.29), we have

$$
c^a_{hm} - c^a_{bm} \leq e^a_{hm} - e^a_{bm} \quad \text{for } b=0 \text{ to } h-1.
$$

In other words, the points $0'$ to $h'-1$ in Figure 3 shift to the right by distances that correspond with the right shifts of points $0$ to $h-1$. Now the length $c^a_{hm}$ is the difference between the R.H.S. and the L.H.S. of (4.33) which must be exhaustively allocated to the nodes in $J_{bm}^V - J_{bm}^V, b<h$. Thus the same inductive approach portrayed earlier applies except that (4.37) transforms to
\[ r_{j}^{a,k} = (c_{0}^{a,k} \sum_{k=1}^{b-1} \sum_{j \in J_{km}} r_{j}^{a,k} (e_{j}^{*}/(e_{0}^{a,k} - e_{0}^{a,k} - e_{j}^{a,k})) \]

for each \( j \in J_{b,m} \) \( J_{b,m} \)

and (4.38) transforms to

\[ r_{j}^{a,k} = e_{j}^{*} \quad \text{for each } j \in J_{b,m} \]

(4.43)

where \( r_{j}^{a,k} = r_{j,k}^{a} \) and \( e_{j}^{*} = e_{j}^{a} - e_{j}^{a,k} \). Note that (4.42) applies when

\[ c_{0}^{a,k} \sum_{k=1}^{b-1} \sum_{j \in J_{km}} r_{j}^{a,k} \leq e_{0}^{a,k} - e_{0}^{a,k} - e_{j}^{a,k} \]

and (4.43) applies otherwise.

Considering next the problem of moving \( f \) to \( T \) such that 1) (4.32)
holds, 2) (4.34) holds, 3) (4.35) holds for \( b \geq h \), and 4) \( T \)-\( f \) is along the
subgradient \( \alpha \), it can be resolved similarly using an inductive approach.

We note first of all that the given \( f \) is such that

\[ \sum_{k=b}^{B_{m}} \sum_{j \in J_{b,m}} f_{j}^{a} \geq c_{b,m} - c_{b-1,m} \quad \text{for } b \geq h+1 \]

(4.44)

However, we require (4.44) to be an equality at \( b = h+1 \). In addition, the
given \( f_{j}^{a} \)'s may violate (4.32) at some nodes. Let

\[ J_{b,m}^{V} = \{ j \mid j \in J_{b,m}, f_{j}^{a} < e_{j}^{a} \} \quad \text{for } b = h+1 \text{ to } B_{m} \]

be the sets of nodes where (4.32) is violated. Since we are allowed to
increase \( f_{j}^{a} \) whenever \( f_{j}^{a} < e_{j}^{a} \), this contention being due to the
subgradient \( \alpha \), we can set
\[ r_j = u_j \text{ at each } j \in J^V_{bm} \text{ and } b > h \]  

(4.45)

Considering now the \( r_j \)'s at each \( j \in J^V_{bm}, b > h \), and the \( r_j \)'s as given by (4.45), we note they satisfy (4.44) and (4.32). However, the L.H.S. of (4.34) may be greater than the R.H.S. of (4.34). So, our task here is to close the gap between the two sides of (4.34) by decreasing \( r_j^a \) relative to \( r_j^a \) at one or more nodes in \( J^V_{bm}, b > h \). But we must also ensure that (4.32) and (4.35) for \( b > h \) + 1 continue to hold. Note that maintaining (4.44) for \( b > h \) + 1 together with satisfying (4.34) implies that (4.35) holds for \( b > h \).

Consider the following translation to \( c^a_{bm} \) and \( u^a_{bm}, b > h \):

\[
c^a_{bm} = c^a_{bm} - \sum_{k=h+1}^{b} \sum_{j \in J^V_{km}} u_j \text{ for } b = h+1 \text{ to } B^a_m
\]

\[
e^a_{bm} = e^a_{bm} - \sum_{k=h+1}^{b} \sum_{j \in J^V_{km}} u_j \text{ for } b = h+1 \text{ to } B^a_m
\]

Clearly because of (4.30), we have,

\[ c^a_{bm} = c^a_{hm} > e^a_{bm} = e^a_{hm} \text{ for each } b > h. \]

In other words, the points \( h+1 \) to \( B^a_m \) as portrayed in Figure 3 shift to the left by distances that correspond with the left shifts of points \( h+1 \) to \( B^a_m \). So, we could use the inductive approach to allocate the \( r_j^a \)'s over the nodes in \( J^V_{bm}, b > h \). However, we also require \( r_j^a < r_j^a \) at each of these nodes. Thus, the method we use is as follows. Order the abatement levels backward from \( B^a_m \) to \( h+1 \), and letting \( b \) be the current abatement level, use the formula:
or the formula (4.43) otherwise, where \( t_j^{a*-} = t_j^a - r_j. \) This completes the method for going from \( t \) to \( \tau \) at any \( m \in M_\lambda \) such that 

\[
(L_{u | n \in N}^T)(L_{u | n \in N}) = (L_{u | n \in N}^T) \text{ at } m, \text{ and that the change } \tau - t \text{ is in line with the subgradient } \tau.
\]

Consider next the problem of moving \( t \) to \( \tau \) at any potential multiple effluent mine \( m \in M_\lambda \) such that 

\[
(L_{u | n \in N}^T)(L_{u | n \in N}) = (L_{u | n \in N}^T) \text{ at } m \text{ and that the change } \tau - t \text{ is along the subgradient } \tau. \]  

The subproblem of 

\[
(L_{u | n \in N}^T)(L_{u | n \in N}) \text{ at } m \text{ can be stated as:}
\]
\[
\begin{align*}
\min C_m &= c^s y^s_m + \sum_{b=1}^{B_m} c_{ba} z_{bm}^s \\
&\quad + \sum_{j \in J_{am}} (e_j^s x_j^s + e_j^a x_j^a + e_j^t x_j^t + e_j^{sat} x_j^{sat}) \\
&\quad + c^d y^d_m + \sum_{b=1}^{B_m} c_{da} z_{bm}^d \\
&\quad + \sum_{j \in J_{am}} (e_j^d x_j^d + e_j^{da} x_j^{da} + e_j^{dt} x_j^{dt} + e_j^{dat} x_j^{dat})
\end{align*}
\]

\text{s.t. } x_j^s + x_j^a + x_j^t + x_j^{sat} + x_j^d + x_j^{da} + x_j^{dt} + x_j^{dat} \leq 1 \text{ for each } j \in J_{am}

\begin{align*}
x_j^s + x_j^a + x_j^t + x_j^{sat} &= y^s_j \text{ for each } j \in J_{am} \\
x_j^d + x_j^{da} + x_j^{dt} + x_j^{dat} &= y^d_j \text{ for each } j \in J_{am} \\
x_j^s + x_j^{sat} &= \sum_{k=b}^{B_m} z_{km}^s \text{ for each } j \in J_{bm} \text{ and } b=1 \text{ to } B_m
\end{align*}

\text{and } x_j^d + x_j^{dat} = \sum_{k=b}^{B_m} z_{km}^d \text{ for each } j \in J_{bm} \text{ and } b=1 \text{ to } B_m

where \( c_j^s, c_j^d, c_j^a, \) and \( c_j^{da} \) in the effective cost definitions in (4.16) to (4.23) are to be regarded as zero. We note that 1) at most one \( y \) variable can be set to 1 or both must be zero in view of the three mining alternatives, 2) \( y^s_m=0 \) or \( y^d_m=0 \) implies that the corresponding decision variables in \( z \) and \( x \) are also zero, and 3) \( y^s_m=1 \) or \( y^d_m=1 \) restricts the subproblem to a structure identical to the subproblem at an abandoned or active multiple effluent source, implying we can use a
procedure similar to (R4.2) to determine the associated optimal decisions in z and x. Thus we merely are required to select the mining alternative and its associated optimal decisions that give the least cost. Note that \( y_m^d = y_m^c = 0 \) implies the "do not mine" alternative and the associated zero cost, stipulating \( y_m^d = 1 \) or \( y_m^c = 1 \) must yield a negative cost to be chosen.

Suppose we restrict the subproblem at \( a \) by requiring \( y_m^d = 1 \). Then procedure (R4.2) applies but for the symbols used, so that if \( h \) is the optimal abatement level at the potential surface mine, then we can deduce

\[
\begin{align*}
&c_{h,m}^{\text{sa}} - c_{b,m}^{\text{sa}} \leq \sum_{k=b+1}^{h} \sum_{j \in J_{km}} \mathbf{v}_j^s \quad \text{for each } b \in 0 \leq b < h \\
&c_{h,m}^{\text{sa}} - c_{b,m}^{\text{sa}} \geq \sum_{k=b+1}^{h} \sum_{j \in J_{km}} \mathbf{v}_j^s \quad \text{for each } b \in h \leq b \leq m
\end{align*}
\]

(4.47)

and

(4.48)

where \( \mathbf{v}_j^s = \min(e_j^s, e_j^c) - \min(e_j^s, e_j^c) \)

\( c_j^s - c_j^c = 0 \) is assumed in \( e_j^s, e_j^c, e_j^{sat} \) and \( e_j^{sat} \), and

\( c_{0m}^{\text{sa}} = 0 \).

Here, as we had discussed for any abandoned or active multiple effluent source, we can use (4.47) and (4.48) to move \( \text{f}_j^{\text{sa}} \) to \( \text{f}_j^{\text{sa}} \) at each \( j \in J_{sa} \), such that 1) (4.4) and (4.7) continue to hold, 2) \( v(LR_{u|n \in N}) \) equals \( v(LR_{u|n \in N}) \) at \( m \) with the restriction that \( y_m^d = 1 \), and 3) the change is in line with the subgradient \( \mathbf{g} \). Note that although we require \( \text{f}_j \) at each \( j \in J_{sa} \) to compute \( v(LR_{u|n \in N}) \) at \( m \), we need not be concerned because we
are restricting $y_{m}^{s}=1$. In other words, we merely require the sum of the
$T_{j}^{s}$'s over $j$ in $J_{m}$, and this we know is the constant $c_{m}^{s}$.

Similarly, we can restrict the subproblem at $m$ with $y_{m}^{d}=1$ and move $r_{j}^{da}$
to $r_{j}^{da}$ at each $j \in J_{m}$. Now let $C_{m}^{s}$ and $C_{m}^{d}$ be the conditional minimum
costs of $(LR_{u}|n \in N)$ at $m$ under the restrictions $y_{j}^{s}=1$ and $y_{j}^{d}=1$
respectively. Then using the definitions $c_{j}^{s}=c_{0}^{d}=0$, $c_{j}^{sa}=r_{j}^{sa}$ and $c_{j}^{da}=r_{j}^{da}$,
and because $v(LR_{u}|n \in N,y_{j}^{s}=1)$ equals $v(LR_{u}|n \in N,y_{j}^{d}=1)$ and
$v(LR_{u}|n \in N,y_{j}^{d}=1)$ equals $v(LR_{u}|n \in N,y_{j}^{d}=1)$, we have

$$C_{m}^{s}=c_{m}^{s}+\sum_{j \in J_{m}}\min(e_{j}^{s},e_{j}^{sa},e_{j}^{st},e_{j}) \quad (4.49)$$

and

$$C_{m}^{d}=c_{m}^{d}+\sum_{j \in J_{m}}\min(e_{j}^{d},e_{j}^{da},e_{j}^{dt},e_{j}) \quad (4.50)$$

Now, in order that $v(LR_{u}|n \in N)$ equals $v(LR_{u}|n \in N)$ at $m$, we require
(4.2) and (4.3) to hold, and that

$$c_{m}^{s} \leq c_{m}^{d} \Rightarrow r_{j}^{s}+\min(e_{j}^{s},e_{j}^{sa},e_{j}^{st},e_{j}) \leq r_{j}^{d}+\min(e_{j}^{d},e_{j}^{da},e_{j}^{dt},e_{j})$$

at each $j \in J_{m}$

$$c_{m}^{s} \leq 0 \Rightarrow r_{j}^{s}+\min(e_{j}^{s},e_{j}^{sa},e_{j}^{st},e_{j}) \leq 0 \quad \text{at each } j \in J_{m}$$

$$c_{m}^{d} \leq 0 \Rightarrow r_{j}^{d}+\min(e_{j}^{d},e_{j}^{da},e_{j}^{dt},e_{j}) \leq 0 \quad \text{at each } j \in J_{m}$$

In addition, in order that the moves $r_{j}^{s}$ to $r_{j}^{s}$ and $r_{j}^{d}$ to $r_{j}^{d}$ are along the
subgradient $f$, we require that if

$$f_{j}^{s}+\min(e_{j}^{s},e_{j}^{sa},e_{j}^{st},e_{j}) \leq 0$$

then $r_{j}^{s} \geq f_{j}^{s}$
and if
\[ r_j^d + \min(e_j^d, e_j^a, e_j^s, e_j^at) \begin{cases} \leq 0 \\ \leq r_j^s + \min(e_j^s, e_j^a, e_j^s, e_j^at) \end{cases} \]
then \( r_j^d \geq r_j^d \).

Let us first compare \( y_m^s = 1 \) with \( y_m^d = 0 \). Suppose \( c_m^s < 0 \). Then we require
\[ r_j^s + \min(e_j^s, e_j^a, e_j^s, e_j^at) \leq 0 \quad \text{at each} \quad j \in J_m^s. \quad (4.51) \]

Now, for the given \( f_j^s \)'s, let
\[ J_m^v = \{ j | j \in J_m^s, r_j^s + \min(e_j^s, e_j^a, e_j^s, e_j^at) > 0 \} \]
refer to the set of nodes where the above requirement is violated. We know, from the point of view of the subgradient \( z \), we may increase or decrease \( r_j^s \) at these nodes. In particular, let us decrease \( r_j^s \) at these nodes; i.e., let us set
\[ r_j^s = -\min(e_j^s, e_j^a, e_j^s, e_j^at) \quad \text{at each} \quad j \in J_m^v. \quad (4.52) \]

Then we are assured \((4.51)\) is met at each \( j \in J_m^v \). We also know that \( r_j^s \)
at each \( j \) in \( J_m^s - J_m^v \) meets \((4.51)\). However,
\[ \sum_{j \in J_m^s - J_m^v} r_j^s + \sum_{j \in J_m^v} r_j^s = c_m^s - \sum_{j \in J_m^v} (r_j^s + \min(e_j^s, e_j^a, e_j^s, e_j^at)) \]
\[ \sum_{j \in J_m^v} j \in J_m^v \quad (4.53) \]
since the sum of the given \( f_j^s \)'s over \( J_m^s \) is \( c_m^s \). So, in order that we achieve requirement \((4.2)\), we must increase the \( f_j^s \)'s over \( J_m^s - J_m^v \) by a total of \( V \). But we must also ensure that \((4.51)\) is preserved. It turns
out that we can render this increase to \( V \) and preserve (4.51) because \( c^s_m < 0 \). The formula we use is

\[
T^s_j + \min(e_j, e_j^s, e_j^s, e_j, e_j^s, e_j^s) < \frac{1}{\sum_{j \in J_{m}} (T^s_j + \min(e_j, e_j^s, e_j^s, e_j^s, e_j^s, e_j^s))}
\]

at each \( j \in J_{m} \)  

(4.54)

We note that the above change of \( r^s_j \) is along the subgradient \( z \); i.e., the subgradient calls for increasing these \( r^s_j \)’s.

Next, if \( C^s > 0 \), then we require

\[
T^s_j + \min(e_j, e_j^s, e_j^s, e_j^s, e_j^s, e_j^s) > 0 \quad \text{at each } j \in J_{m}
\]

The set of nodes where this requirement is violated is

\[ J^V_{m} = \{ j | j \in J_{m}, T^s_j + \min(e_j, e_j^s, e_j^s, e_j^s, e_j^s, e_j^s) < 0 \} \]

It turns out that (4.52) to (4.54) directly apply here for changing \( r^s_j \), except that \( V \) now refers to a nonpositive quantity.

We can compare \( y_{m}^{d} = 1 \) against \( y_{m}^{d} = y_{m}^{d} = 0 \) similarly. If now \( C^s_m < 0 \) and \( C^d_m > 0 \) or vice versa, then the current \( r^s_j \)’s and \( r^d_j \)’s suffice. However, if \( C^s_m < 0 \) and \( C^d_m < 0 \) or if \( C^s_m > 0 \) and \( C^d_m > 0 \), then we may be required to make additional changes to the \( r^s_j \)’s and the \( r^d_j \)’s. Let us assume one of these cases exist, and in particular assume \( C^s_m < C^d_m \). Also let the current \( r^s_j \)’s and \( r^d_j \)’s be \( r^s_j \)’s and \( r^d_j \)’s respectively, and that we are attempting to find the \( r^s_j \)’s and \( r^d_j \)’s respectively. We now require,

\[
T^s_j + \min(e_j, e_j^s, e_j^s, e_j, e_j^s, e_j^s) \leq r^d_j + \min(e_j, e_j^d, e_j^d, e_j^d, e_j^d)
\]

at each \( j \in J_{m} \)
Let \( J_{^m}^v \) denote the set of nodes where this requirement is violated; i.e.,
\[
J_{^m}^v = \{ j \mid j \in J_{^m}, r_j^s + \min(e_j, e_j, e_j, e_j) > r_j^d + \min(e_j, e_j, e_j, e_j) \}
\]

It turns out that because (4.2) and (4.3) currently hold for the subvector of \( T \) at \( m \), and also because \( C_s^d < C_d^s \), the following formulas stipulate the needed changes:

\[
\begin{align*}
T_j^s &= T_j^m - V_j/2 \quad \text{at each } j \in J_{^m}^v \\
T_j^d &= T_j^s + V_j/2 \quad \text{at each } j \in J_{^m}^v \\
T_j^s &= T_j^s - V_j(V/A) \quad \text{at each } j \in J_{^m} - J_{^m}^v \\
T_j^d &= T_j^d + V_j(V/A) \quad \text{at each } j \in J_{^m} - J_{^m}^v
\end{align*}
\]

where
\[
V_j = r_j^s + \min(e_j, e_j, e_j, e_j) - r_j^d - \min(e_j, e_j, e_j, e_j)
\]

at each \( j \in J_{^m}^v \)

\[
V = \sum_{j \in J_{^m}^v} V_j
\]

\[
A = - \sum_{j \in J_{^m} - J_{^m}^v} V_j
\]

For the case when \( C_s^d < C_d^s \), the above exposition applies but the symbols \( s \) and \( d \) alternate. Finally, we mention a minor point. That is, at the head vertex, we initialize \( r_j^s \) and \( r_j^d \) at each \( j \in J_{^m} \), by simply allocating \( c_s^m \) and \( c_d^m \) equally across the appropriate \( r_j^s \)'s and \( r_j^d \)'s respectively.
The Ascent Method

Like the subgradient method, this method improves $v(L_{u_r})$ over $u$ for a fixed $r$. However, it applies only when the instream treatment decisions are all fixed. Recall that, at vertices where the instream treatment decisions are all fixed, problem $(L_{u_r})$ constitutes a set of independent subproblems. This section addresses the improvement of $(L_{u_r})$ over $u$ for any such subproblem.

The subproblem here will be defined over an entire watershed when the instream processors in the watershed are all inactive. Or it may be defined over only a portion of a watershed when one or more instream processors in the watershed are active. In this latter case, the active instream processors mark the boundaries for the subproblem. Figure 4 illustrates three subproblems in a watershed. Note that the stream network defining a subproblem is conceptually the same as the stream network we represent in any watershed. So, to be general, we will visualize the subproblem's stream network as consisting of the usual level 1, 2 and 3 streams.

We note first of all there are no instream processors to contend with in the subproblem's network. Secondly, since $r$ is fixed, so are the allocations of the mining and abatement costs at multiple effluent mines, in essence, we can view each mine source in the subproblem as a single effluent source. Thirdly, if we do not relax the quality constraints in the subproblem, the subproblem has the form
Min \( c_j = \sum_{j \in J^s} c^s_j \)

s.t. \( x_j^e + x_j + x_j^d + x_j + x_j^t + x_j^r \leq 1 \) at each \( j \in J^s \cap J^e \)

\( x_j^e + x_j^d + x_j^r \leq 1 \) at each \( j \in J^s \cap J^d \)

and \( w_{ip_{ij}}^e + w_{ip_{ij}}^d + w_{ip_{ij}}^r + w_{ip_{ij}}^s + w_{ip_{ij}}^e \leq r_{ij} \) for \( i=1 \) to \( I_{hj} \) and \( j \in J^e \)

\[(8_i)\]

where \( J^e \) is the set of nodes in the subproblems, \( c^s_j \) is given by (2.16) or (2.17) and supplemented by \( c^d_r = r_j \), \( c^d_r = r_j \), \( c^d_r = r_j \), \( c^d_r = r_j \) and \( c^d_r = r_j \) at appropriate multiple effluent mine nodes in \( J^e \), and \( w_{ip_{ij}}^e \) is given by (2.8) or (2.9) as appropriate.
Clearly, if we relax the quality constraints in \((S_j)\), we obtain the subproblem \((LR_{ur})\). This \((LR_{ur})\) we noted has the integrality property, in essence, the maximum of \(v(LR_{ur})\) over \(u\) must equal \(v(\overline{S}_1)\), where \(\overline{S}_1\) is the LP relaxation of \((S_j)\). Thus, the best that the ascent method can achieve is \(v(\overline{S}_1)\). Problem \((\overline{S}_1)\) however is not easy to solve because of the presence of multiple worst-cases in the quality constraints. So, what we do is remove all but one of the worst-cases from the quality constraints in the usual Lagrangian fashion. That is, we obtain the problem

\[
\min C_2 = \sum_{j \in \mathcal{J}^P \cap \mathcal{J}^g} (a_j x_j + a_j s_j + d_j x_j + e_j x_j + e_j x_j + e_j x_j)

+ \sum_{j \in \mathcal{J}^g \cap \mathcal{J}^s} (a_j (1-x_j x_j x_j) + e_j x_j + e_j x_j + e_j x_j)

\text{s.t.} \quad x_j + x_j + x_j + d_j x_j + e_j x_j + e_j x_j \leq 1

\text{at each} \quad j \in \mathcal{J}^P \cap \mathcal{J}^g

x_j x_j x_j \leq 1 \quad \text{at each} \quad j \in \mathcal{J}^g \cap \mathcal{J}^s

\text{and} \quad \sum_{i \in \mathcal{J}^I} \sum_{j \in \mathcal{J}^I} w_{ij} x_{ij} x_{ij} \leq \mathcal{E}_{ij}

\text{for some} \ i \text{ and each} \ j \in \mathcal{J}^s
\]

\((S_2)\)

where the effective costs in the symbol \(e\) are as in (4.16) to (4.27) with the exception that the summations over the worst-cases do not include the particular \(i\) identified in \((S_2)\). It is again the case here that the maximum of \(v(LR_{ur})\) over the \(u_{ij}'s\) is equal to \(v(\overline{S}_2)\), where we fix all the \(u_{kj}'s\), \(k \neq i\).
The ascent method here solves (S_2). We apply this method to each worst-case in turn to improve v(LR_{ur}) over u. To elucidate, suppose there are four worst-cases. We first fix the u_{1j}'s for worst-cases 2, 3 and 4, and optimize the u_{1j}'s by solving (S_2). Then we fix the u_{1j}'s for worst-cases 1, 3 and 4, and optimize the u_{2j}'s. We continue in this fashion for worst-cases 3 and 4. We may then repeat this process over the four worst-cases. We have experimented with one or two such repetitions.

The ascent method can only bring about an improvement to v(LR_{ur}), never a deterioration; hence its name. Secondly, repeated applications of the ascent method across the worst-cases is not guaranteed to converge to v(S_1). We have identified some exceptions to this convergence, in essence, the terminal condition for repeated applications of the ascent method can be different from the optimality conditions for (S_j).

The basic idea in the ascent method is to choose decisions in descending cost-effectiveness, much like the idea used in solving the LP relaxation of a simple knapsack problem. For simplicity, let problem (S_2) be defined on a single stream. Consider now the most upstream node, say j, and let x^*_j be the least cost decision at this node; i.e.,

\[ e_j^* = \min\{0, e_j, e_j^a, e_j^s, e_j^{st}, e_j^{sd}, e_j^{dt}, e_j^{dat}\} \]

Now if x^*_j satisfies the ith worst-case quality constraint at node j, then it implies from the LP viewpoint that the quality slack of this constraint resides in the basis, and in consequence, the optimal value
of \( u_{ij} \) is zero because of the LP complementary slackness theorem. On the other hand, if \( x_j^s \) violates the quality constraint at \( j \), then we would first pick the decision (relative to \( x_j^s \)) that costs the least per unit reduction in the \( i \)th worst-case flow to node \( j \). If this new decision meets the \( i \)th worst-case quality requirement at \( j \), then we regard \( u_{ij}^e \), the effective sum of the \( u_{ij} \)'s over node \( j \) and nodes downstream from \( j \) until the first active instream processor, as being equal to the least cost required for reducing the \( i \)th worst-case flow by one unit. But if the new decision does not meet the quality requirement, we again compare the remaining decisions against the new (incumbent) decision and pick the most cost-effective decision. We continue in this fashion until we eventually satisfy the \( i \)th worst-case quality constraint at \( j \) (if we cannot, we can declare problems \((S_2)\) and \((S_1)\) as infeasible). Also, we take \( u_{ij}^o \) as being equal to the cost needed for reducing the \( i \)th worst-case flow to node \( j \) by one unit, in the final step taken to achieve the quality at \( j \).

Consider next the second node from the top of the stream, and let this node be \( j \). Now, in order to ascertain if the \( i \)th worst-case quality requirement at \( j \) is achieved by the least cost decision at \( j \), we regard

\[
\tilde{w}_{ip_{1j}} = \begin{cases} 
q_{ip_{1j}} & \text{if the quality slack at } p_{1j} \\
\text{is nonbasic from the LP viewpoint} \\
\tilde{w}_{ip_{1j}} + w_{ip_{1j}} & \text{otherwise.}
\end{cases}
\]

Note that the quality slack \( p_{1j} \) would be zero (nonbasic) if we had to incur additional expense over and above the least cost decision at \( p_{1j} \).
Thus, using the above formula, we determine if the least cost decision at \( j \) meets the \( i \)th worst-case quality requirement at \( j \). If so, we regard \( u_{ij}^* = 0 \) and

\[
\nu_{ij}^* = \nu_{p_{ij}}^* + \nu_{ij}^n + \nu_{ij}^e
\]

and simply proceed to the next downstream node. Otherwise, we find the most cost-effective decision at \( j \) and record its cost per unit \( i \)th worst-case flow reduction. Then we check the quality slack at the next upstream node. If the slack is positive, then we find the most cost-effective decision at the next upstream node relative to its incumbent decision, and select the decision between \( j \) and the next upstream node that has the best cost-effectiveness. If the quality slack at the next upstream node is zero, we compare the cost-effectiveness ratio (cost per unit pollutant reduction) of the best decision at \( j \) against \( u_{ip_{ij}}^e \). If the ratio is the smaller, we elect to go with the best decision at \( j \); otherwise, we introduce the quality slack at \( p_{ij} \) into the basis (LP terminology) and regard

\[
\nu_{p_{ij}}^* = \nu_{ip_{ij}}^* + \nu_{p_{ij}}^n \leq q_{p_{ij}}^j
\]

where \( \nu_{ip_{ij}}^e \) is the pollutant output due to the incumbent decision at \( p_{ij} \). Thus, whatever be the case, we end up reducing the \( i \)th worst-case flow at \( j \) in the cheapest possible way. We iterate in the above fashion until the \( i \)th worst-case quality requirement is met. Having satisfied the requirement, we regard \( u_{ij}^e \) as the final cost-effectiveness ratio achieved and \( \nu_{ij}^* = q_{ij} \), implying that the quality slack at \( j \) is zero. Note that if in the final iteration, the pollutant reduction came from introducing the quality slack at \( p_{ij} \) into the basis, then \( u_{ij}^e = u_{ip_{ij}}^e \).
We then take up the third node from the top of the stream. The rules here essentially correspond with those at the second node. We continue in this fashion until we exhaust all nodes. At this time, we will have recorded the nodes where the quality slacks are zero and their associated $u^e_{ij}$'s. So we simply proceed up the stream beginning with the last node and identify the positive $u^e_{ij}$'s.

Our implementation of the ascent method is a little different from the description above. That is, we perform all computations taking the currently available $u^e_{ij}$'s as the origin. This significantly saves on the computations involved in the method exposed above where $u^0$ is the origin. But the principles remain the same.

BRANCHING FROM A LIVE VERTEX

We discuss here the rules for selecting the branching variable at any given live vertex, and the branching process after such a selection has been made. Recall we prioritize the variable types for selection. That is, we give top priority to the instream treatment decision variables, if any are free at the given vertex, second priority to the mining and abatement decisions at multiple effluent mines when these variables are free, and last priority to all remaining variables.

Consider first a vertex where one or more instream treatment decision variables are free. The rule here is to choose the instream variable that has the highest processor cost among all the free instream variables. Once chosen, we take two branches, one branch stipulating
instream treatment and the other its negation. Letting \( j \) be the node hosting the instream processor, we set \( u_{ij} = 0 \) for each \( i = 1 \) to \( I_{h_j} \) at the successor vertex stipulating instream treatment, since the quality standard at \( j \) will always be met.

Consider next a live vertex where all instream treatment decisions are fixed but where one or more mining and(or) abatement decisions at the multiple effluent mines are free. Recall that at such a vertex, there is no gap between the maximum of \( v(LR_{ur}) \) over \( r \) and \( v(LR_u) \), in essence, we have available the mining and abatement decisions at the multiple effluent mines that are optimal in \( (LR_u) \). The idea we deploy to select the branching variable relates to the question: how much deterioration is there in \( v(LR_{ur}) \) if we suppress the optimal mining and(or) abatement decisions at a given multiple effluent mine? In particular, we pick the mine that causes the greatest deterioration in \( v(LR_{ur}) \). The following two paragraphs discuss how the deterioration is computed at a given mine.

Let the given \( r' \)ne be a potential multiple effluent mine where both the mining and abatement decisions are free. Recall that (4.49) and (4.50) give the conditional minimum costs for the restrictions \( y^e_m = 1 \) and \( y^d_m = 1 \) respectively. Clearly, the optimal mining decision is that which yields the minimum among \( 0, C^e_m, C^d_m \). So, if we suppress the optimal decision, then the deterioration in \( v(LR_{ur}) \) arises from having to choose between the remaining two alternatives. In particular, we would choose the smaller cost alternative and thus obtain the deterioration at this mine.
Consider next an abandoned or active multiple effluent mine, or a potential surface or deep mine, where the abatement decision is free. Recall that procedure (R4.2) identifies the optimal abatement level $h$ and the associated cost. If now we suppress abatement at level $h$, then procedure (R4.2) can be used again to find the next best level of abatement and the associated cost. In this way, we can quantify the deterioration of interest at the given mine.

Having selected the multiple effluent mine showing the greatest deterioration in $v(LR_{ur})$, we branch on the mining decision variable or the abatement decision variable depending on the situation. If it is a potential multiple effluent mine where both the mining decision and abatement decision are free, we branch first on the mining decision variable yielding three branches corresponding to the three mining alternatives. If it is an abandoned or active mine, or a potential surface or deep mine, we branch on the abatement decision as follows.

Suppose that the allowable abatement levels are between $b_1$ and $b_2$ (initially, $b_1=0$ and $b_2=B_m$), and that the optimum abatement level is $h$. Then on one of the branches, we restrict the abatement decision to abatement at level $h$. If $h>b_1$, then we restrict the abatement decision to abatement at levels $b_1$ to $h-1$ on a different branch. Also if $h<b_2$, we restrict the abatement decision to abatement at levels $h+1$ to $b_2$ on yet another branch. Thus, we can have at most three or at least two branches.

After we branch on the abatement decision at a multiple effluent mine, we also need to ensure that the $r_j^e$, $r_j^{sa}$, or $r_j^{da}$'s, as the case
may be, reflect accurately the abatement cost allocations they represent at each successor vertex. Regard the \( r_j^a \)'s as typifying all cases. Now, when we restrict the abatement decision to abatement at level \( h \) on the first branch, we know that because of (4.33), the \( r_j^a \)'s reflect truly the cost of abating at level \( h \); so we can keep these \( r_j^a \)'s as they are.

Next, if \( h > b_1 \), recall we restrict the abatement decision to abatement at levels \( b_1 \) to \( h-1 \) on another branch. Here, because of (4.35) at \( h-1 \), the sum of the \( r_j^a \)'s over the levels \( b_1 \) to \( h-1 \) may be less than \( c_{h-1,m}^a \), in which case we are required to update the \( r_j^a \)'s so that they equal \( c_{h-1,m}^a \).

This update can be done using the inductive approach outlined for the initialization of \( r \) at the head vertex. Lastly, when \( h < b_2 \), we restrict the abatement decision to abatement at levels \( h+1 \) to \( b_2 \) on yet another branch as noted. Here, we are required to update the \( r_j^a \)'s so that (4.35) is an equality at \( h+1 \) and that the sum of the \( r_j^a \)'s over levels \( h+2 \) to \( b_2 \) equals \( c_{b_2,m}^a - c_{h+1,m}^a \). This update can be accomplished again by using the inductive procedure described for initializing \( r \) at the head vertex.

Consider now a vertex where the instream decisions and the mining and abatement decisions at multiple effluent mines are all fixed. Recall that at such a vertex, the restricted problem of (P) constitutes several independent subproblems, and the strategy was to solve these subproblems one at a time by branching exhaustively on their free variables. The next section explains this strategy. For now, consider the selection of the branching variable in a given subproblem. Recall that the subproblem \( (LR_{ur}) \) is solved simply by selecting at each node the
decision that gives the least effective cost (see definitions (4.16) to (4.27)). If we now suppress the optimal decision at a given node, then \( v(LR_{ur}) \) will deteriorate by an amount equal to the difference between the minimum of the remaining effective costs and the effective cost of the optimal decision. The selection rule is to select the node having the greatest such deterioration. Having selected the node, we restrict the mine source decision to the optimal decision on one branch and its negation on the other.

SELECTING BETWEEN LIVE VERTICES

We maintain two lists of live vertices. One list contains vertices that are regarded as direct descendants of the head vertex, and the other has vertices that are descendents of a vertex where the instream treatment decisions and the mining and abatement decisions at multiple effluent mines are all fixed.

At the top of the enumeration tree, both lists are empty. At the head vertex, we improve the lower bound \( v(LR_{ur}) \), use a heuristic to obtain a good feasible solution to (P), branch on an instream treatment decision variable, improve the lower bounds at the two successor vertices and add the vertices to the first list. Naturally, if the lower bound of one of the vertices exceeds the feasible solution cost, we fathom the vertex. We then scan the first list of vertices, and use one of the following two strategies to select the next live vertex. In the first strategy, we simply select the vertex with the smallest lower
bound. In the second, we associate with each live vertex a "good" feasible solution which we obtain using the feasible solution heuristic. In doing so, we also update the current best feasible solution if we realized an improved solution, and eliminate any vertex from the first list whose lower bound cost exceeds the cost of the improved feasible solution. In any case, having associated with each live vertex the "good" feasible solution it contains, we estimate that the cost of its best feasible solution is

\[ C = v(LR_{ur}) + (C^f - v(LR_{ur}))F \]

where \( C^f \) is the cost of the good feasible solution and \( F \) is a fraction. We have experimented with values in the range 0.6 to 0.9 for this fraction. We then select the live vertex in the first list that has the smallest cost estimate.

After the above selection, we examine if the selected vertex has any free instream treatment decision or a free mining or abatement decision at any multiple effluent mine. If it is the case, we select the branching variable, improve the lower bounds at the successor vertices, fathom if possible, add the unfathomed vertices to the list, and use the first or second strategy to select the next live vertex from the first list. We continue in this fashion, adding vertices to the first list, until we select a vertex where the instream treatment decisions and the mining and abatement decisions at all multiple effluent mines are all fixed. We then regard this vertex as heading the second list of vertices.
To begin with, we identify all the individual subproblems at the vertex heading the second list of vertices. Let there be $K$ subproblems, and let $k$ denote a subproblem's index. We associate with each subproblem, the lower bound $L_k$ and the upper bound $U_k$; initially, $L_k$ is the subproblem's $v(LR_{ur})$ and $U_k$ is the cost of a feasible solution to the subproblem obtained using the feasible solution heuristic. We then solve the subproblems, taking a subproblem at a time and branching exhaustively on the subproblem's variables. This branching on a subproblem creates the second list of vertices. We use the first or second strategy indicated earlier for selecting between live vertices in the second list. Also we update $U_k$ each time an improved feasible solution is realized, letting that $k$ is the current subproblem we are examining. In addition, we use the following upper bound to fathom live vertices in the second list:

$$\min \{ U_k, C - \sum_{i=1}^{K} L_i \}$$

where $C$ is the cost of the current best feasible solution to $(P)$. Furthermore, we keep a record of the smallest lower bound of all the vertices we have fathomed from the second list. Let $S_k$ be this smallest lower bound. Eventually when the second list of vertices is empty, we compare $S_k$ against $U_k$. If $S_k > U_k$, then $U_k$ is the subproblem's optimal cost and we set $L_k = U_k$. Otherwise, there is no guarantee that $U_k$ is the optimal cost and we set $L_k = S_k$. We then take up the next subproblem and repeat as above. Eventually when we are done with all subproblems, we
update the best feasible solution to (P) if there is an improved overall solution, and return to the first list of vertices.

There is also one special purpose device we use akin to the implementation of the decomposition concept in ALCOT. That is, we maintain a list of solved subproblems, these subproblems referring to those identified at vertices heading the second list of vertices. These subproblems are conditional only on the particular set of fixed mining and abatement decisions at multiple effluent mines, and recur at vertices heading the second list of vertices. We establish these recurrences by scanning the list of stored subproblems, and implement the subproblems' optimal solutions or lower bounds (corresponding to the $S_k$'s).

THE FEASIBLE SOLUTION HEURISTIC

This heuristic finds a feasible solution for the subproblem given by $(S_i)$ in the section on the ascent method. That is, the heuristic applies only after the instream treatment decisions and the mining and abatement decisions at multiple effluent mines are all fixed.

The heuristic is based on the view that the effective costs defined in (4.16) to (4.27) reflect the cost-effectiveness of the source decision alternatives, the higher the effective cost, the lower the cost-effectiveness of the particular alternative. So the idea is to keep the sum of the effective costs as small as possible and achieve a feasible solution to $(S_i)$. Note that this sum is actually the
Lagrangian objective. The decisions that minimize the Lagrangian objective are simply those which have the least effective costs. If these decisions by chance achieve the quality requirements in \((S_j)\), then we have the feasible solution we are looking for. But this does not occur generally.

We had indicated earlier that we may view the stream network of the subproblem \((S_j)\) as comprising the usual level 1, 2 and 3 streams. For simplicity of exposition however, assume that the network consists of a single stream. Nevertheless, the principles in the ensuing exposition apply in general.

Consider the most upstream node. Simply select the decision that has the least effective cost and also meets the quality requirement at the node for each worst-case. Regard this decision as the incumbent at this node.

Consider next the second node from the top of the stream. Let the decision with least effective cost be the incumbent at this node. Now, for the incumbent decisions at the first two nodes, identify the worst-case situations violating the quality standard at the second node. Identify next the decisions at the first and second nodes, each of which, relative to the incumbent decision, renders a reduction in the sum of the pollutant flows across the violating worst-cases and at the same time maintains the currently satisfied quality constraints at the first and second nodes. Call these decisions the candidate decisions. Among the candidate decisions, select the decision that incurs the
smallest increase in effective cost relative to its incumbent, and make this decision the incumbent at its node. Once again, check to see if the incumbent decisions satisfy the quality requirements at the second node. Repeat the above process as many times as is necessary until the quality standard is achieved at the second node for each worst-case.

Proceed to the third node from the top of the stream, and let its least effective cost decision be its incumbent. Once again, identify the violating worst-cases and the candidate decisions at the first three nodes as above. Select the candidate decision that incurs the least increase in effective cost relative to its incumbent, and make it the incumbent. Continue in this fashion until the quality standard is achieved at the third node for each worst-case.

Consider the remaining downstream nodes in the direction of stream flow and achieve the quality standard at each node for each worst-case as above. This basically is the essence of the heuristic procedure. The heuristic is usually rapid because the Lagrangian solution is usually close to feasible solution. Secondly, most decisions render pollution reductions simultaneously across all worst-case situations.

There are exceptional cases when the above heuristic fails to produce feasible solutions. In these instances, we disregard the unachieved quality constraints as we proceed downstream from node to node, and mark up the cost of the final solution by an arbitrary percentage (we have used 20%). Thus, the cost of the heuristic solution, feasible or otherwise, plays its role in the second strategy we described for choosing between live vertices.
The discussion above actually constitutes only the first step of the overall heuristic procedure. In the second and final step, we attempt to reduce the actual cost of the heuristic. Assume that the heuristic solution obtained at the end of the first step is feasible; we defer the discussion for the infeasible case. Note that the $c_j^e$'s in the description of problem $(S_j)$ refer to the actual costs of the source decisions. So the idea now is to regard all decisions having smaller actual costs than their incumbent decisions as the candidate decisions. The idea extends a little further. That is, we also require each candidate decision to meet the quality standard at each downstream node for each worst-case. With this perspective, we associate with each candidate decision the following ratio:

$$\frac{\text{Effective cost of Candidate} - \text{Effective cost of Incumbent}}{\text{Actual cost of Incumbent} - \text{Actual Cost of Candidate}}$$

We select the candidate decision having the smallest such ratio, and make it the incumbent at its node. We then update the list of candidate decisions and once again select the candidate decision that has the smallest above ratio. We repeat this process as many times as is necessary until the candidate list is empty. It is the author's judgement that the second step of the heuristic procedure can cause a sizable reduction to the cost of the feasible solution at the end of the first step.

The above ratio implies the loss in cost-effectiveness of decisions per unit decrease in the actual cost of decisions. By keeping this loss
to a minimum between iterations of the second step, we expect to utilize the available quality constraint slacks as efficiently as possible.

Regarding the case when the heuristic solution at the end of the first step is infeasible, we exclude from the second step all decisions that are upstream of any violated quality constraint. The candidate list of decisions comes only from the remaining decisions. With this rule in effect, we improve the given infeasible solution as above. In any case, we mark up the cost of the eventual infeasible solution as noted earlier.
CHAPTER V
A BRANCH AND BOUND ALGORITHM USING LINEAR PROGRAMMING RELAXATION

Recall that the nonlinearities in the minimum cost problem as formulated in Chapter II arise only from the clipping effect of instream treatment expressed in equations (2.11), (2.15) and (2.22). This chapter shows linear forms for these equations, in effect, the minimum cost problem transforms to an ILP and a branch and bound algorithm using LP relaxation applies. In particular, a code developed by Gonsalves [7] called ENUMCODE implements such an algorithm. This chapter explains briefly this code and the next chapter illustrates potential results.

LINEAR FORMS FOR EQUATIONS (2.11), (2.15) AND (2.22)

Consider equation (2.11). Define $w_{ij}$ as the aggregate ith worst-case flow to the instream processor at $j$; i.e.,

$$w_{ij} = w_{ip_{1j}} + w_{ip_{2j}} + w_{ij} + w_{ij}^r$$ for each $j \in J^f$ \hspace{1cm} (5.1)

Also let the upper bound on the maximum value of $w_{ij}$ over all feasible upstream solutions be $w_{ij}^u$. Clearly now, if $w_{ij}^u > 0$, then $w_{ij}$ as shown below

$$w_{ij} \begin{cases} \leq w_{ij}^u - w_{ij}^x \hfill \\
\geq 0 \end{cases} \hspace{1cm} \text{for each } j \in J^f \hspace{1cm} (5.2)$$
can represent $w^*$ in (2.11) because smaller values of $w^*_j$ benefit the minimum cost problem so that one of the inequalities in (5.2) will always be binding, and in consequence, if $x_j^* = 0$ then $w^*_j = w^*_j$ or if $x_j^* = 1$ then $w^*_j = 0$. Next let, $w^*_j$ be the minimum possible value of $w^*_j$ over all feasible upstream solutions. Clearly if $w^*_j > 0$, then $w^*_j = w^*_j$ accomplishes our objective. However if $w^*_j < 0$, then $w^*_j$ may be less than zero, in which case $w^*_j$ will not represent $w^*_j$. So define

$$w^*_j \begin{cases} 
\geq w^*_j + w^*_j z^*_j \\
\geq w^*_j + w^*_j z^*_j - w^*_j
\end{cases} \text{ for each } j \in \mathcal{J}^t \quad (5.3)$$

where $z^*_j$ is a binary variable. Note that smaller values of $w^*_j$ benefit the minimum cost problem, so one of the inequalities in (5.3) will always be binding. There are now four possibilities: 1) $w^*_j > 0$, $x_j^* = 0$, 2) $w^*_j > 0$, $x_j^* = 1$, 3) $w^*_j < 0$, $x_j^* = 0$ and 4) $w^*_j < 0$ and $x_j^* = 1$. In the first case, $w^*_j = w^*_j$ because of (5.2), and $w^*_j = w^*_j$ whether $z^*_j$ is zero or one. In the second case, $w^*_j = 0$, and $z^*_j = 0$ minimizes the R.H.S. of (5.3), so $w^*_j = 0$. In the third and fourth cases, $w^*_j = 0$, and $z^*_j = 1$ minimizes the R.H.S. of (5.3), so $w^*_j = w^*_j$. Thus (5.3) in conjunction with (5.2) provides the linear form for equation (2.11) when $w^*_j < 0$.

Equation (2.15) can be linearized similarly. Let

$$s^*_j = a^*_p j^* + a^*_p j^* + a^*_p j^* + a^*_p j^* \text{ for each } j \in \mathcal{J}^t$$

$$\bar{s}^*_j = \text{upper bound on the maximum feasible value of } s^*_j \text{ for each } j \in \mathcal{J}^t$$

$$\underline{s}^*_j = \text{minimum feasible value of } s^*_j \text{ for each } j \in \mathcal{J}^t,$$
Then 
\[
\begin{align*}
    a_j^z \left\{ \begin{array}{l}
    a_{j} - \overline{a}_{j} x_{j} \\
    \geq 0
    \end{array} \right. \quad \text{for each } j \in J^c
\end{align*}
\]

and 
\[
\begin{align*}
    a_j^z \left\{ \begin{array}{l}
    a_{j} + \overline{a}_{j} y_{j} \\
    \geq a_{j} - \overline{a}_{j} x_{j} - \overline{a}_{j}
    \end{array} \right. \quad \text{for each } j \in J^c
\end{align*}
\]

give the linear form for (2.15), where \( y^z_{j} \) is a binary variable.

Finally, regarding the criterion function, its linear formulation is

\[
c^T = \sum_{j=1}^{J} c^x_j + \sum_{m=1}^{M} c^m + \sum_{j \in J^c} (c^x_j x^z_j + c^y_j y^z_j)
\]

where

\[
y^z_j \left\{ \begin{array}{l}
    a_{j} + \overline{a}_{j} x_{j} - \overline{a}_{j} \\
    \geq 0
    \end{array} \right.
\]

Note that smaller values of \( y^z_j \) benefit the criterion, so one of the inequalities in (5.8) will always be binding. Secondly, if \( x^z_j = 0 \) then \( y^z_j = 0 \). Thirdly, if \( x^z_j = 1 \), then \( y^z_j = a^z_j \) if \( a^z_j > 0 \) or \( y^z_j = 0 \) if \( a^z_j < 0 \). This sums up the justification for (5.7) and (5.8).

**ENUMCODE**

This code written by Gonsalves [7] applies to any mixed integer linear program and in particular to the mixed binary linear program specifying the minimum cost problem. It is a branch and bound code where the bounds are determined using LP relaxation. The LP part of the code is
called PROFOR written by Martin [11], which is based on the revised simplex method and uses the product form of the inverse so as to handle large problems in core.

Initially, at the vertex heading the enumeration tree, the LP solution provides a lower bound on the minimum cost, and a mechanism for rounding the fractional valued binary variables in the basis provides a feasible solution whose cost is an upper bound on the minimum cost. Then a branching variable is chosen from the set of binary variables having fractional values in the basis. This selection process involves pseudo-costs and an estimate of the degradation in the LP minimum for each fractional valued binary variable when the variable is forced to zero and one respectively. The selection process is explained later. Once a fractional valued binary variable is chosen, the successor vertex having the higher LP minimum is stored and the other successor becomes the current vertex to be expanded from. Naturally, if the lower bound of the worse vertex is not less than the current upper bound, it is fathomed and not stored.

The expansion from the current vertex is similar to that described for the head vertex. If the rounding mechanism applied to the LP solution offers a better feasible solution, the upper bound is updated. The selection process for the branching variable is identical. The successor vertex having the worse LP minimum is stored or fathomed, and the other successor becomes current. In some instances, both vertices may be fathomed in which case the vertex with the best pseudo-cost estimation (discussed later) in tree storage is recovered and taken as
the new current vertex. This overall method of expanding live vertices is called depth first search technique with flexible backtracking.

Eventually, when the tree storage of vertices is empty, the best binary solution recorded is the minimum cost solution.

**Pseudo-costs**

Pseudo-costs are attributed to the binary variables and provide an approximate rate of deterioration in the LP objective when corresponding fractional valued binary variables are forced to their upper or lower bounds. There are thus the "up" and "down" pseudo-costs associated with each binary variable, which are:

$$\text{PCUP}_j = \frac{z^u - z^0}{1 - f_j}$$

$$\text{PCDN}_j = \frac{z^d - z^0}{f_j}$$

where $\text{PCUP}_j$ and $\text{PCDN}_j$ are the up and down pseudo-costs of the $j$th binary variable, $z^0$ is the LP minimum at the current vertex, $z^u$ is the LP minimum when variable $j$ is forced to one, $z^d$ is the LP minimum when variable $j$ is forced to zero, and $f_j$ is the fractional value of variable $j$ in the basis.

Initially, the pseudo-costs are unknown. They are calculated for a binary variable the first time the variable appears with a fractional value in the LP optimal basis at any given vertex. Subsequently, when
the variable reappears with a fractional basic value at a different vertex, the deterioration it causes to the LP minimum is estimated to be

\[ \text{PCUP}_j(1 - f_j) \text{ in the "up" direction} \]

and \[ \text{PCDN}_j(f_j) \text{ in the "down" direction} \]

where \( f_j \) is its current fractional value. This is more efficient than the calculation of actual deteriorations. Also, when a variable with a known pseudo-cost is chosen as the branching variable, the pseudo-costs for the variable are updated using the formulae:

\[
\text{PCUP}_j = \text{PCUP}_j(1 - \text{PCF}) + \frac{z^u - z^0}{1 - f_j} \text{ PCF}
\]

\[
\text{PCDN}_j = \text{PCDN}_j(1 - \text{PCF}) + \frac{z^d - z^0}{f_j} \text{ PCF}
\]

where \( \text{PCF} \) is a weighting factor based on the level or depth in the tree at which the current branching occurred. This weighting factor decreases with increasing depth of the tree since it is assumed that pseudo-costs calculated deep in a tree are likely to be local to that part of the tree whereas those calculated near the top of the tree are likely to be more global.
Choosing the Branching Variable

The variable that is chosen for branching is that variable which maximizes the sum of the degradations in both branches; i.e.,

$$\max_j (PCUP_j (1 - f_j) + PCDN_j (f_j))$$

The idea here is to concentrate on critical variables as defined by large pseudo-costs. This branching criterion also aids to minimize the time spent to prove the optimality of the incumbent integer solution.

The above criterion is applied within each class of a set of priority classes, each class constituting several binary variables. In the present context, the top priority class contains the instream treatment decision variables, the second class has the mining decision variables at potential multiple effluent mines, the third has the abatement decisions at all multiple effluent mines, and the last class holds all remaining variables.

Choosing a Vertex from Tree Storage

For an estimation of a vertex in tree storage, the following expression is used:

$$E = z^0 + \sum_{j=1}^{J} (PCUP_j (1 - f_j) + PCDN_j (f_j))$$
where $J$ is the set of fractional valued binary variables in the optimal LP basis of the vertex. The vertex with the smallest $E$ is chosen and made the "current" vertex. The estimate $E$ provides a relative measure for the best binary solution available at each vertex, in essence, the choice favors the vertex likely to have the best binary solution.
CHAPTER VI
COMPUTATIONAL EXPERIENCE AND CONCLUSIONS

This final chapter presents the computational experience with the three algorithms of interest on an AMDAHL 470 V.8. Computer and concludes with an analysis of the comparative performances of the algorithms. A total of five test problems have been considered, four of which are hypothetical and one a real life case depicting the mine drainage situation near Elkins, West Virginia [3]. Data for the hypothetical problems reflect typically occurring quantities and are therefore valid for conclusive purposes.

Table 2 contains the essential features of the five test problems, where Problem 5 depicts the real life case. Problems 1 and 2 are devoid of multiple effluent mines and are identical except for the three instream treatment sites in Problem 2. Similarly, Problems 3 and 4 are identical but for the 6 instream treatment sites in Problem 4. Thus the effect of instream treatment decisions on computational time can be isolated by comparing Problems 1 and 2 or Problems 3 and 4. Similarly, Problems 1 and 3 or Problems 2 and 4 can be compared to isolate the effect of multiple effluent mines on the computational time. To elucidate, Problems 3 and 4 each has 26 nodes in each watershed. If multiple effluent mines were absent in these problems, then the
Table 2. Test Problem Features

<table>
<thead>
<tr>
<th>Prob No.</th>
<th>Water sheds</th>
<th>No. of Streams</th>
<th>No. of Nodes</th>
<th>No. of Aban sheds</th>
<th>No. of Aban, doned</th>
<th>No. of Aban, tial</th>
<th>Potential stream cases</th>
<th>In- stream cases</th>
<th>Worst cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>9</td>
<td>37</td>
<td>29</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>9</td>
<td>37</td>
<td>29</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>27</td>
<td>78</td>
<td>53</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>27</td>
<td>78</td>
<td>53</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>10</td>
<td>36</td>
<td>20</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

Subproblems in each watershed would be independent, in effect, one would expect the total computational time to be triple the time for solving 26 nodes. Thus, a third of the CPU time taken respectively by problems 3 and 4 should be less than the CPU time taken respectively by problems 1 and 2, if multiple effluent mines do not significantly affect CPU time. Note that problems 1 and 2 each has 11 more nodes than 26.

Table 3 shows the characteristics of the test problems per the linear formulation of Chapter V. The continuous variables here include the quality constraints' slacks. Note the relatively high constraint matrix density (nonzero elements) which could have a detrimental impact on an LP based algorithm. Note also that Problem 2 has three more binary variables than Problem 1 corresponding to the three instream treatment
Table 3. Test Problem’s Characteristics

<table>
<thead>
<tr>
<th>Problem Index</th>
<th>No. of Binary Variables</th>
<th>No. of Continuous Variables</th>
<th>No. of Constraints</th>
<th>Constraint Matrix Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>124</td>
<td>74</td>
<td>111</td>
<td>11.85%</td>
</tr>
<tr>
<td>2</td>
<td>127</td>
<td>86</td>
<td>114</td>
<td>7.86%</td>
</tr>
<tr>
<td>3</td>
<td>324</td>
<td>156</td>
<td>274</td>
<td>4.00%</td>
</tr>
<tr>
<td>4</td>
<td>330</td>
<td>180</td>
<td>280</td>
<td>3.13%</td>
</tr>
<tr>
<td>5</td>
<td>131</td>
<td>128</td>
<td>142</td>
<td>3.63%</td>
</tr>
</tbody>
</table>

variables, and likewise, Problem 4 has 6 more binary variables than Problem 3 corresponding to the six instream treatment variables. In other words, we have used input data for Problems 2 and 4 such that $w^e_{ij}$, which we defined in Chapter V, is greater than or equal to zero for each $i=1$ to $I_h$ and $j \in J^t$. This simplified the generation of the LP constraint matrix.

Table 4 gives the results of ALCOT. ALCOT converges only for the Elkin’s problem and only when the quick and dirty lower bound computation method is deployed, taking 6.91 CPU minutes for convergence. All other runs fail to converge after 30 minutes of CPU time, even when they were initiated with upper bounds slightly above their respective minimum costs. Other results (not reported here) show that the dynamic programming lower bound computation method performs consistently and considerably worse than the quick and dirty method. This poor performance is probably because different mine types (surface versus
Table 4. ALCOT Results

<table>
<thead>
<tr>
<th>Problem Index</th>
<th>Quick and Dirty Bound Method</th>
<th>Dynamic Programming Bound Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Iterations</td>
<td>Iteration at Optimum</td>
</tr>
<tr>
<td>1</td>
<td>216000</td>
<td>none</td>
</tr>
<tr>
<td>2</td>
<td>212000</td>
<td>none</td>
</tr>
<tr>
<td>3</td>
<td>136000</td>
<td>none</td>
</tr>
<tr>
<td>4</td>
<td>113000</td>
<td>none</td>
</tr>
<tr>
<td>5</td>
<td>91000</td>
<td>14384</td>
</tr>
</tbody>
</table>

<sup>a</sup> - CPU time is in seconds

underground) have markedly different effects on different worst-case situations, in consequence, the independent consideration of the worst-cases in the DP method proves to be worse than the simultaneous consideration of the worst-cases in the quick and dirty method. Some type of surrogate constraint has to be constructed for efficient implementation of dynamic programming. Another notable result is ALCOT fails to determine the stopping vector for each test problem even after 5 minutes of CPU time. The bottle-neck here were the stream confluences where it took an inordinately long time to compute the recursive functions, this in turn being due to the many feasible pollution flows to confluence nodes. Hence the stopping vector determination was scrapped from all the runs reported in Table 4.

Table 5 gives the results of the branch and bound algorithm that uses lagrangian relaxation. The results are certainly dramatic in comparison to ALCOT's results. The ascent method proves to be better than the
Table 5. Lagrangian Relaxation Results

<table>
<thead>
<tr>
<th>Problem Index</th>
<th>Subgradient Method</th>
<th>Ascent Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Vertices Expanded</td>
<td>Vertex at Live Optimum VERT (sec)</td>
</tr>
<tr>
<td>1</td>
<td>73</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>121</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>562</td>
<td>108</td>
</tr>
<tr>
<td>4</td>
<td>3267</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>65</td>
<td>3</td>
</tr>
</tbody>
</table>

The subgradient method for problems 2, 3 and 5, the subgradient method performing the better for problem 1. The subgradient method was not applied to problem 4 because of its poor performance for problem 3. Experiments with the subgradient method show that it is best to halve the subgradient step size when the lagrangian falls to increase after 4 steps. Also, smaller sequences of \( u \) prove to be better than larger sequences; for example, using sequence lengths of 12, 8 and 6 for the top, middle and bottom parts of the enumeration tree proves consistently better than similar lengths of 24, 20 and 16. Experiments with the ascent method show it is best to use the method once for each worst-case situation; repetitions over the worst-cases increases computational time. Other experiments show: 1) the lagrangian multiplier vector \( r \) is best changed once only at each relevant vertex implying that the gap between \( v(\text{LR}_{u_r}) \) and \( v(\text{LR}_u) \) is not very significant in general, and 2) computational time is smaller when the selection rule for selecting between live vertices employs the lagrangian functional value as opposed
to the estimate described in Chapter IV, implying that proof of optimality is more important than finding the vertex that contains the optimal solution. Note the power of the feasible solution heuristic which finds the optimal solution relatively early in the enumeration tree for each test problem.

Comparing the CPU times for problems 1 and 3 in Table 5, we see that a third of problem 3's CPU time is greater than problem 1's time, indicating that the presence of multiple effluent mines does slow down the algorithm. The same is true when we compare problems 2 and 4. Instream treatment also slows down the algorithm, quite severely at that when we compare problems 3 and 4.

Table 6. ENUMCODE Results

<table>
<thead>
<tr>
<th>Problem Index</th>
<th>Vertices Expanded</th>
<th>Vertices Fathomed By Obj. Function Infability</th>
<th>Maximum Live Vertices</th>
<th>Vertex at Optimum</th>
<th>CPU Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>924</td>
<td>352</td>
<td>9</td>
<td>45</td>
<td>1220</td>
</tr>
<tr>
<td>2</td>
<td>1504</td>
<td>681</td>
<td>9</td>
<td>135</td>
<td>419</td>
</tr>
<tr>
<td>3</td>
<td>2264</td>
<td>257</td>
<td>120</td>
<td>517</td>
<td>2487</td>
</tr>
<tr>
<td>4</td>
<td>2532</td>
<td>141</td>
<td>16</td>
<td>497</td>
<td>none</td>
</tr>
</tbody>
</table>

a - Number in tree storage at the time of premature termination
Table 6 shows the results of ENUMCODE. We see that the CPU times for problems 1 and 2 are significantly greater than that achieved by the lagrangian code. Secondly, there is no convergence for problems 3 and 4 after CPU times in excess of those taken by the lagrangian code, and further a significant number of live nodes exist in tree storage at termination. Regarding problem 5, we experienced some unrecoverable numerical problems with ENUMCODE and we do not have results at this time.

Compare first the results for problem 1 in Tables 1 and 2. We see that the lagrangian code using subgradient optimization expands a meager 73 vertices whereas ENUMCODE, the theoretically better bound calculator, expands 924. Moreover, the lagrangian code produces a maximum of 18 vertices in tree storage whereas ENUMCODE has a maximum of 45. The likely reason for the better performance of the lagrangian code over ENUMCODE lies in the power of the feasible solution heuristic in the lagrangian code which finds the optimal solution very early in the enumeration tree, whereas the rounding mechanism in ENUMCODE did not have the same success.

Compare next the results of problem 2 in Tables 5 and 6. We see that both algorithms found the optimal solution relatively early in the enumeration tree. However, ENUMCODE expands a considerably greater number of vertices than does the lagrangian code. This is troubling because ENUMCODE works with better bounds. However, the lagrangian code uses decomposition at instream processors and thus works with powerful bounds on occasions. This is likely the reason for the speed of the lagrangian code.
Explanations for the better performance of the lagrangian code over ENUMCODE for problems 3 and 4 are similar. In general, to accurately compare algorithmic performances, we require to gather better statistics. However, due to limitations of time, the author defers further investigation to another effort. Nevertheless, the superior performance of the lagrangian code has been worth this investigation.
LIST OF REFERENCES


