INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book. These are also available as one exposure on a standard 35mm slide or as a 17" x 23" black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600
Application of network flow and zero-one programming to open-pit mine design problems

Cai, Wenlong, Ph.D.

The University of Arizona, 1989
APPLICATION OF NETWORK FLOW AND ZERO-ONE PROGRAMMING TO OPEN-PIT MINE DESIGN PROBLEMS

by

Wenlong Cai

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF MINING AND GEOLOGICAL ENGINEERING
In partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY
WITH A MAJOR IN MINING ENGINEERING
In the Graduate College
THE UNIVERSITY OF ARIZONA

1989
As members of the Final Examination Committee, we certify that we have read the dissertation prepared by Wenlong Cai entitled Application of Network Flow and Zero-One Programming to Open-Pit Mine Design Problems and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

5-4-89 5-4-89 5-6-89 5-9-89 7/31/89

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copy of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Donald E. Myers 7/31/89

Dissertation Director Date
STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: [Signature]
ACKNOWLEDGEMENTS

I would like to thank Dr. Young C. Kim for his guidance and advice in preparing this dissertation. I am especially grateful to the time that Dr. Kim spent in the discussions of dissertation subjects as well as in the evaluation and English corrections of this dissertation.

I also would like to thank Dr. Jaak J.K. Daemen, Dr. John L. Denny, Dr. DeVerle P. Harris and Dr. Donald E. Myers for being my committee members. I appreciate the comments on the dissertation subjects and the corrections on my English from my committee members.

I thank the Ministry of Education of the People's Republic of China, for providing a scholarship that enabled me to study in the United States.

I am grateful to the mining consulting firm - Mintec, Inc., Tucson, Arizona for introducing me to many practical problems. These problems inspired me to carry out the research in this dissertation. Special thanks are to Mr. A. Fred Banfield - president of the Mintec, Inc., his financial support made it possible for me to go for my doctoral degree.

Finally, I would like to thank my wife Sumin and my daughter Xuemei. I can never make up the sacrifices they experienced during the period when I had to study in an abroad country while left them behind. After they join me in the United States, they have been making every moment of my life a happier and a more enjoyable one.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>9</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>12</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>13</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>15</td>
</tr>
<tr>
<td>1.1 General</td>
<td>15</td>
</tr>
<tr>
<td>1.2 Statement of The Problem</td>
<td>20</td>
</tr>
<tr>
<td>1.2.1 Statement of Ultimate Pit Limit Design Problem</td>
<td>21</td>
</tr>
<tr>
<td>1.2.2 Statement of Mining Sequence Development Problem</td>
<td>22</td>
</tr>
<tr>
<td>1.3 Scope of Study and Solution Techniques</td>
<td>25</td>
</tr>
<tr>
<td>2 ULTIMATE PIT LIMIT DESIGN AND LONG RANGE OPEN PIT MINE PLANNING</td>
<td>28</td>
</tr>
<tr>
<td>2.1 Basic Hierarchy of Mine Planning System</td>
<td>28</td>
</tr>
<tr>
<td>2.2 Ultimate Pit Limit Design</td>
<td>34</td>
</tr>
<tr>
<td>2.2.1 Pit Limit Design Objectives and Assumptions</td>
<td>34</td>
</tr>
<tr>
<td>2.2.2 Criteria for Optimal Ultimate Pit Limit Design</td>
<td>35</td>
</tr>
<tr>
<td>2.2.3 Economic Inventory Block Model</td>
<td>35</td>
</tr>
<tr>
<td>2.2.4 Pit Slope Angles and Cone Generation</td>
<td>37</td>
</tr>
<tr>
<td>2.2.5 Pit Limit Design via Moving Cone Heuristic</td>
<td>41</td>
</tr>
<tr>
<td>2.2.6 Review of Current Pit Design Approaches</td>
<td>42</td>
</tr>
<tr>
<td>2.3 Mining Sequence Development</td>
<td>44</td>
</tr>
<tr>
<td>2.3.1 Objectives and Assumptions</td>
<td>45</td>
</tr>
<tr>
<td>2.3.2 Generation of Series of Pushbacks</td>
<td>46</td>
</tr>
<tr>
<td>2.3.3 Mine Sequencing among Pushbacks</td>
<td>46</td>
</tr>
</tbody>
</table>
2.3.4 Review of Current Approaches .................................................. 48
2.3.5 Remarks on Current Approaches .................................................. 49

3 PIT LIMIT DESIGN VIA NETWORK FLOW TECHNIQUES 51
  3.1 The Maximum Network Flow Problem ........................................... 51
  3.2 DMKM Algorithm for Network Flow Problem .................................. 54
  3.3 Network Flow Formulation of Pit Limit Design Problem .................. 59
  3.4 Difficulties in Direct Application of Maximum Flow Algorithms ...... 60

4 DEVELOPMENT OF A SIMULATION ORIENTED NETWORK FLOW ALGORITHM FOR OPEN PIT LIMIT DESIGN PROBLEM 63
  4.1 Hanson’s Modified Network Flow Algorithm .................................... 64
    4.1.1 Concept of Support ............................................................. 64
    4.1.2 Lowest Support First Priority Theorem ................................... 64
    4.1.3 Hanson’s Algorithm ............................................................ 66
    4.1.4 Non-Optimum Proof of Hanson’s Algorithm ................................ 68
  4.2 Non-Optimum Theorem on General Simulation Oriented Pit Design Algorithms .................................................. 71
  4.3 Development of a Simulation Oriented Network Flow Algorithm ...... 77
    4.3.1 Hindrance Concept for Ore Blocks ......................................... 77
    4.3.2 Lowest Hindrance with Highest Value First Priority Theorem ...... 79
    4.3.3 Description of A Simulation Oriented Network Flow Algorithm .... 81
  4.4 Performance of the New Algorithm .............................................. 89
    4.4.1 Selected Comparison Criteria ............................................... 89
    4.4.2 Comparison Results ............................................................ 90
# TABLE OF CONTENTS - (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5 Summary Remarks</td>
<td>92</td>
</tr>
<tr>
<td>5 DEVELOPMENT OF A ZERO-ONE PROGRAMMING MODEL FOR MINE SEQUENCING PROBLEMS</td>
<td>94</td>
</tr>
<tr>
<td>5.1 Mine Sequencing Problem of this Study</td>
<td>94</td>
</tr>
<tr>
<td>5.2 Zero-One Programming and Implicit Enumeration</td>
<td>97</td>
</tr>
<tr>
<td>5.2.1 Zero-One Programming Problem</td>
<td>97</td>
</tr>
<tr>
<td>5.2.2 Implicit Enumeration Algorithm</td>
<td>98</td>
</tr>
<tr>
<td>5.2.3 Remarks on Balas' Implicit Enumeration Algorithm</td>
<td>100</td>
</tr>
<tr>
<td>5.2.4 Precedence Constraint Formulation</td>
<td>101</td>
</tr>
<tr>
<td>5.3 Single Period Zero-One Programming Mine Sequencing Problem Formulation</td>
<td>105</td>
</tr>
<tr>
<td>5.3.1 Definition of Decision Variables and Mine Sequencing Assumptions</td>
<td>105</td>
</tr>
<tr>
<td>5.3.2 Limiting the Number of Decision Variables</td>
<td>106</td>
</tr>
<tr>
<td>5.3.3 Definition of Objective Functions</td>
<td>113</td>
</tr>
<tr>
<td>5.3.4 Formulation of Problem Constraints</td>
<td>115</td>
</tr>
<tr>
<td>5.3.5 Precedence Constraints Formulation of Mine Sequencing Problem</td>
<td>117</td>
</tr>
<tr>
<td>5.3.6 A Complete 0-1 Programming Formulation of Mine Sequencing Problem</td>
<td>118</td>
</tr>
<tr>
<td>5.4 Implementation of the 0-1 Programming Mine Sequencing Model</td>
<td>120</td>
</tr>
<tr>
<td>6 APPLICATION OF THE 0-1 PROGRAMMING MODEL TO THE DIAMOND MINE SEQUENCING PROBLEM</td>
<td>121</td>
</tr>
<tr>
<td>6.1 Diamond Mine Sequencing Project Case Summary</td>
<td>121</td>
</tr>
<tr>
<td>6.2 Definition of Sub-optimization Objectives</td>
<td>121</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS - (Continued)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3 Solution Procedure for the Diamond Project</td>
<td>122</td>
</tr>
<tr>
<td>6.4 Problem Formulation for the 1st Period - Sample Layout</td>
<td>125</td>
</tr>
<tr>
<td>6.5 Mine Sequencing for Period 1 - Sample Computer Run Layout</td>
<td>140</td>
</tr>
<tr>
<td>6.6 Summary of Diamond Pit Sequencing Schedule</td>
<td>146</td>
</tr>
<tr>
<td>6.7 Conclusions on Diamond Pit Sequencing</td>
<td>154</td>
</tr>
<tr>
<td>6.8 Experiences on Application of 0-1 Programming to the Mine Sequenc-</td>
<td>154</td>
</tr>
<tr>
<td>ing Problem in this Study</td>
<td></td>
</tr>
<tr>
<td>7 CONCLUSIONS</td>
<td>157</td>
</tr>
<tr>
<td>7.1 Conclusions</td>
<td>157</td>
</tr>
<tr>
<td>7.1.1 Pit Design</td>
<td>157</td>
</tr>
<tr>
<td>7.1.2 0-1 Programming Mine Sequencing Model</td>
<td>159</td>
</tr>
<tr>
<td>7.2 Future Research Recommendations</td>
<td>163</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>172</td>
</tr>
<tr>
<td>A Numerical Example on the Application of the Simulation Oriented Net-</td>
<td></td>
</tr>
<tr>
<td>work Flow Algorithm</td>
<td>172</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>185</td>
</tr>
<tr>
<td>Diamond Pit Reserve List (Pushback 2 - 6)</td>
<td>185</td>
</tr>
<tr>
<td>APPENDIX C</td>
<td>191</td>
</tr>
<tr>
<td>Mine Sequencing Reports for Diamond Pit from Period 2 - 6</td>
<td>191</td>
</tr>
<tr>
<td>APPENDIX D</td>
<td>206</td>
</tr>
<tr>
<td>FORTRAN Program Listing of 0-1 Programming Routine</td>
<td>206</td>
</tr>
<tr>
<td>SELECTED BIBLIOGRAPHY</td>
<td>220</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Circular Nature of Mine Evaluation Process</td>
<td>16</td>
</tr>
<tr>
<td>1.2 Cross Section View of the Next Best Mineralization</td>
<td>19</td>
</tr>
<tr>
<td>1.3 Sequence Access Concept</td>
<td>19</td>
</tr>
<tr>
<td>1.4 Relationship among Pushback Generation, Mining Sequence Development and Financial Analysis</td>
<td>20</td>
</tr>
<tr>
<td>1.5 The Saw-tooth Like Stripping Ratios</td>
<td>23</td>
</tr>
<tr>
<td>2.1 Seven Main Mine Planning Tasks</td>
<td>29</td>
</tr>
<tr>
<td>2.2 Three Dimensional Representation of Block Model</td>
<td>31</td>
</tr>
<tr>
<td>2.3 Concept of Restricting and Non-Restricting Blocks</td>
<td>36</td>
</tr>
<tr>
<td>2.4 Concept of Shared Contribution</td>
<td>36</td>
</tr>
<tr>
<td>2.5 Defining Pit Limits with a Series of Cones and Cone Segments</td>
<td>39</td>
</tr>
<tr>
<td>2.6 Five Cone Generation Routines</td>
<td>40</td>
</tr>
<tr>
<td>2.7 Example of Moving Cone Method Missing Shared Contribution and Over-Mining at the Same Time</td>
<td>42</td>
</tr>
<tr>
<td>2.8 Example of a Pushback</td>
<td>47</td>
</tr>
<tr>
<td>2.9 A 2-D Hypothetical Pushback Pattern</td>
<td>47</td>
</tr>
<tr>
<td>3.1 An Example of a Maximum Network Flow Problem</td>
<td>53</td>
</tr>
<tr>
<td>3.2 An Initial Layered Network for Figure 3.1 Network</td>
<td>56</td>
</tr>
<tr>
<td>3.3 A 2-D Example of Pit Design Network</td>
<td>61</td>
</tr>
<tr>
<td>4.1 Support and Restricting Block Distinction</td>
<td>65</td>
</tr>
<tr>
<td>4.2 Support Concept</td>
<td>65</td>
</tr>
<tr>
<td>4.3 Hanson’s Theorem does not Eliminate the Re-Allocation Problem</td>
<td>69</td>
</tr>
<tr>
<td>4.4 An Upward Cone and a Downward Cone</td>
<td>72</td>
</tr>
<tr>
<td>4.5 Re-Allocation via Upward and Downward Cone</td>
<td>73</td>
</tr>
</tbody>
</table>
### LIST OF ILLUSTRATIONS - (Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6 An Example of Re-Allocation Problem not Being Completely Over-</td>
<td>75</td>
</tr>
<tr>
<td>come by Multi-Pass Strategy</td>
<td></td>
</tr>
<tr>
<td>4.7 An Example of Hanson's Theorem Avoiding the Re-Allocation Problem</td>
<td>76</td>
</tr>
<tr>
<td>4.8 An Example of Ore Block Hindrance</td>
<td>78</td>
</tr>
<tr>
<td>4.9 An Equivalent Network Representation of Figure 4.8 Problem</td>
<td>78</td>
</tr>
<tr>
<td>4.10 An Example of Defining Ore Allocation Sequence to Avoid Re-</td>
<td>80</td>
</tr>
<tr>
<td>Allocating Problem</td>
<td></td>
</tr>
<tr>
<td>5.1 Cross Section of Pushbacks 1 &amp; 2</td>
<td>109</td>
</tr>
<tr>
<td>5.2 Mining Patterns derived from each Bench according to Precedence</td>
<td></td>
</tr>
<tr>
<td>Constraints (cross section views)</td>
<td>111</td>
</tr>
<tr>
<td>6.1 Reserve List of Pushback 1</td>
<td>127</td>
</tr>
<tr>
<td>6.2 SR among the Benches in first period's Problem Formulation</td>
<td>131</td>
</tr>
<tr>
<td>6.3 Computer Printout of 0-1 Programming Problem Formulation for</td>
<td></td>
</tr>
<tr>
<td>Period 1</td>
<td>134</td>
</tr>
<tr>
<td>6.4 Sample Mine Sequencing for Period 1 - Input for Mine Sequencing</td>
<td>141</td>
</tr>
<tr>
<td>6.5 Sample Mine Sequencing for Period 1 - Input for 0-1 Programming</td>
<td>143</td>
</tr>
<tr>
<td>6.6 Sample Mine Sequencing for Period 1 - Input of Buffer Values</td>
<td>145</td>
</tr>
<tr>
<td>6.7 Sample Mine Sequencing for Period 1 - List of Feasible Solution #1</td>
<td>145</td>
</tr>
<tr>
<td>6.8 Sample Mine Sequencing for Period 1 - Solution Adjustment</td>
<td>147</td>
</tr>
<tr>
<td>6.9 Sample Mine Sequencing for Period 1 - List of Final Schedule</td>
<td>147</td>
</tr>
<tr>
<td>6.10 Mine Sequencing Output for Period 1</td>
<td>150</td>
</tr>
<tr>
<td>6.11 List of Type 1 Input Data File</td>
<td>153</td>
</tr>
<tr>
<td>6.12 List of Type 2 Input Data File</td>
<td>153</td>
</tr>
<tr>
<td>7.1 Basic 2-D Cone Removal Pattern</td>
<td>165</td>
</tr>
<tr>
<td>7.2 Block in Upper Cone but not in Lower Cone</td>
<td>165</td>
</tr>
</tbody>
</table>
### LIST OF ILLUSTRATIONS - (Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3</td>
<td>An Equivalent Network to Figure 7.2</td>
<td>168</td>
</tr>
<tr>
<td>7.4</td>
<td>Coning Inconsistency Resulting from Non-Zero Base Radius</td>
<td>169</td>
</tr>
<tr>
<td>7.5</td>
<td>Coning Inconsistency Resulting from Measuring Blocks at Center while</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Generating Cone at the Mid Point of Toe Line of the Base Block</td>
<td>169</td>
</tr>
<tr>
<td>B.1</td>
<td>Reserve Listing of Pushback 2</td>
<td>186</td>
</tr>
<tr>
<td>B.2</td>
<td>Reserve Listing of Pushback 3</td>
<td>187</td>
</tr>
<tr>
<td>B.3</td>
<td>Reserve Listing of Pushback 4</td>
<td>189</td>
</tr>
<tr>
<td>C.1</td>
<td>Mine Sequencing Report for Period 2</td>
<td>192</td>
</tr>
<tr>
<td>C.2</td>
<td>Mine Sequencing Report for Period 3</td>
<td>196</td>
</tr>
<tr>
<td>C.3</td>
<td>Mine Sequencing Report for Period 4</td>
<td>200</td>
</tr>
<tr>
<td>C.4</td>
<td>Mine Sequencing Report for Period 5</td>
<td>202</td>
</tr>
<tr>
<td>C.5</td>
<td>Mine Sequencing Report for Period 6</td>
<td>204</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Information Contained in a Block of Typical Copper Porphyry Deposit</td>
<td>32</td>
</tr>
<tr>
<td>4.1 Summary of Model Information</td>
<td>90</td>
</tr>
<tr>
<td>4.2 Comparison Results Summary</td>
<td>91</td>
</tr>
<tr>
<td>6.1 Summary of Client’s Tonnage Schedule</td>
<td>121</td>
</tr>
<tr>
<td>6.2 Benches of Pushback #1 entering 1st Period Problem Formulation</td>
<td>129</td>
</tr>
<tr>
<td>6.3 Benches of Pushback #2 entering 1st Period Problem Formulation</td>
<td>130</td>
</tr>
<tr>
<td>6.4 Decision Variable Assignment for the 1st Period</td>
<td>132</td>
</tr>
<tr>
<td>6.5 Mining Schedule Following the Client’s Tonnage Requirement</td>
<td>148</td>
</tr>
<tr>
<td>6.6 Summary of Scheduled Tonnages</td>
<td>149</td>
</tr>
<tr>
<td>A.1 Original Value Matrix</td>
<td>173</td>
</tr>
<tr>
<td>A.2 Initial Support Matrix</td>
<td>174</td>
</tr>
<tr>
<td>A.3 Support Matrix After Setup</td>
<td>174</td>
</tr>
<tr>
<td>A.4 Updated Value Matrix</td>
<td>175</td>
</tr>
<tr>
<td>A.5 Initial Hindrance Matrix</td>
<td>175</td>
</tr>
<tr>
<td>A.6 Hindrance Matrix After Set Up</td>
<td>176</td>
</tr>
<tr>
<td>A.7 Initial Support Matrix for Part Four of the Algorithm</td>
<td>177</td>
</tr>
<tr>
<td>A.8 Support Matrix for Part four of the Algorithm</td>
<td>177</td>
</tr>
<tr>
<td>A.9 Updated Value Matrix</td>
<td>178</td>
</tr>
<tr>
<td>A.10 Updated Value Matrix</td>
<td>179</td>
</tr>
<tr>
<td>A.11 Updated Value Matrix</td>
<td>180</td>
</tr>
<tr>
<td>A.12 Updated Value Matrix</td>
<td>180</td>
</tr>
<tr>
<td>A.13 Updated Support Matrix for Part 4 of the Algorithm</td>
<td>181</td>
</tr>
<tr>
<td>A.14 Updated Value Matrix</td>
<td>182</td>
</tr>
<tr>
<td>A.15 Updated Value Matrix</td>
<td>182</td>
</tr>
<tr>
<td>A.16 Updated Support Matrix</td>
<td>183</td>
</tr>
<tr>
<td>A.17 Updated Value Matrix</td>
<td>184</td>
</tr>
</tbody>
</table>
ABSTRACT

An algorithm which adopts a moving cone approach but is guided by maximal network flow principles is developed. This study argues that from a network flow point of view, the re-allocation problem is a major obstacle to prevent a simulation oriented pit design algorithm from reaching the optimum solution. A simulation oriented pit design algorithm can not resolve the re-allocation problem entirely without explicit definition of predecessors and successors. In order to preserve the advantages of moving cone algorithm and to improve the moving cone algorithm, the new algorithm tries to avoid the re-allocation situations. Theoretical proof indicates that the new algorithm can consistently generate higher profit than the popular moving cone algorithm.

A case study indicates that the new algorithm improved over the moving cone algorithm (1% more profit). Also, the difference between the new algorithm and the rigorous Lerchs-Grossmann algorithm in terms of generated profit is very insignificant (0.015% less). The new algorithm is only 2.08 times slower than the extremely fast moving cone algorithm.

This study also presents a multi-period 0-1 programming mine sequencing model. Once pushbacks are generated and the materials between a series of cutoffs are available for each bench of every pushback, the model can quickly answer, period by period, what is the best (maximum or minimum) that can be expected on any one of these four items: mineral contents, ore tonnages, waste tonnages and stripping ratios. This answer is based on a selected cutoff and considers the production capacity defined by the ore tonnage, the desired stripping ratio and the precedence constraints among benches and pushbacks. The maximization of mineral contents is suggested to be the direct mine sequencing objective when it is
permissible. Suggestions also are provided on how to reduce the number of decision variables and how to reduce the number of precedence constraints.

A case study reveals that the model is fast and operational. The maximization of mineral contents increases the average grades in early planning periods.
1.1 General

Nowadays, about 65 percent of world minerals is mined from surface or open pit mines. Open pit mining has many advantages over underground mining such as: higher safety, higher recovery, higher productivity, etc. With large production scale and reduced production cost, the mineral percentage produced from open pit mining may continue to increase, as is indicated by the fact that in U.S., 83 percent of minerals and 82 percent of metallic ores were mined from open pit mines (Allsman, 1968).

Large capital requirements and low ore grades are two non-negligible factors in today's open pit mining. There is little room for inefficiency in both mine planning and operation. In fact, a carefully developed mine plan could turn an otherwise non-profitable project into a profitable one and vice versa.

In general, or at least in the western world, the objective of a company taking on a project is to increase the wealth of the company and to satisfy the stock holders' earnings expectations. A company investing in a mining project would not be justified in paying more than the net present value to be derived from the project. Therefore, a meaningful mine plan ought to try to maximize the net present value produced by the mining project and an "optimum" mine plan should maximize the net present value produced by the mining project.

Although the objective of mine planning is clear, the way to reach the so called "optimum" mine plan is far from straightforward. In fact, whether this "optimum" mine plan can ever be reached is questionable. This is because a mine
reserve usually can not be turned into cash all at once. A mine reserve must be mined across time. Determination of mine size or production rate requires not only balancing the current investment cost and the future returns involving time value of money but also considering the equipment operating efficiencies. Moreover, many practical situations or constraints are difficult to quantify and there exist many uncontrollable uncertainties or risks such as: geological risk, engineering risk, economic risk, political risk, etc.

The problem in developing an “optimum” mine plan is further complicated by the circular nature of mine evaluation. Figure 1.1 illustrates this circular nature of the mine evaluation process.

In figure 1.1, ore reserves are controlled by break-even cutoff grade, higher cutoff grade resulting in less ore reserve. Mine size depends on the ore reserves. Production costs decrease as mine size becomes larger, and lower production costs can lower the break-even cutoff grade.

![Figure 1.1 Circular Nature of Mine Evaluation Process](after Gentry and O'Neil, 1984)
A viable mine plan should be able to deal with the above difficulties and should have enough flexibility for future modification. One approach toward working out an acceptable mine plan is to: 1) temporarily fix some parameters such as cutoff grade, 2) work out a mine plan based on mining the "next best" philosophy, and develop an alternate mine plan by varying the formerly fixed parameters. The desired mine plan is the one which "maximizes" the net present value (NPV) of project cash flow or DCF-ROR, where the DCF-ROR stands for discounted cash flow rate of return, or any other financial criterion which the company chooses. In this study, NPV is assumed to be the financial criterion.

The "optimum" mine plan in this study is defined as the mine plan which, under current predicted economic conditions, technological capabilities and practical constraints, provides maximum NPV among the alternative mine plans, based on mining the "next best" philosophy.

Mathieson (1982) discusses an overall pushback mine planning methodology with emphasis on the design of an optimal sequence of pit expansions or phases. Mathieson's paper presents an overall picture of today's long range open pit mine planning practice. The list of Mathieson's mine planning procedures is given below:

1) Generate a series of nested pits inside a mineralization model by varying break even cutoff grades. Usually, if the economic parameters such as mining and mineral processing costs are fixed, a series of nested pits can be generated with higher product price (lower break-even cutoff grade) resulting in a larger pit and lower product price (higher break-even cutoff grade) resulting in a smaller pit. A pit based on higher product price necessarily includes the pit based on lower product price. If the distance between any two of these series of nested pits exceeds the minimum allowable mining width, these series of nested pits represent the "next best" mineralization with inner pit being the "best" and the outer pit being the "next best". Therefore, the series of nested pits can serve as pushbacks for mine sequencing with the boundary of the outer most pit serving as ultimate pit limit. Figure 1.2 shows a cross section of series of such "next best" incremental pits on a
hypothetical deposit.

2) Calculate bench by bench inventories for ore tonnage, waste tonnage and average grade values for a series of cutoff grades for each incremental pushback.

3) Develop a mining sequence which satisfies the following constraints: a) advanced stripping in each planning period so that not only the minimum ore tonnage requirement of this period can be met but also the minimum ore tonnage requirements of subsequent planning periods can be met; b) smoothed stripping ratios through all mine planning periods; c) access to the current sequence is provided; and d) if possible, blend ore grades in different pushbacks so that the ore grades from successive planning horizons do not vary too much, although higher grades are desired in early planning periods. Figure 1.3 shows the sequence access concept.

4) Optimize production schedule. This includes the following: a) generate alternative ore and waste schedules based on variable production rate and cutoff grade strategies; b) compare the schedules through NPV analysis; c) select a single or combined schedule which has the highest NPV.

The “pushback” approach is the most popular mine sequencing practice. This is probably because the pushback approach adopts a mining “next best” policy and such a policy seems the most logical one in a mining project having high geologic and economic risks.

The mine planning problems discussed in this study deal with the first three of the four parts of the pushback mine planning approach. The production optimization part is not included because it involves repeated application of part three, based on management policy decision and non-unique cash flow analysis. Thus, this dissertation is concerned with developing tools and guidances in open pit mine design practice.
Figure 1.2 Cross Section View of the Next Best Mineralization

Figure 1.3 Sequence Access Concept (After Mathieson, 1982)
1.2 Statement of The Problem

The open pit mine planning approaches outlined in the previous section involve three modules: 1) pushback generating module; 2) mining sequence development module; and 3) financial analysis module. The relationships among these three modules are shown in figure 1.4.

![Diagram showing relationships among pushback generation, mining sequence development, and financial analysis](image)

Figure 1.4 Relationship among Pushback Generation, Mining Sequence Development and Financial Analysis

Module 1 provides a guideline for the whole mine plan, both financially and operationally. Module 2 ensures operational requirements in addition to the goals set in module 1. Module 3 determines which mining sequence within the main planning frame of module 1 and module 2 is more attractive.

As indicated in the previous section, this study deals only with modules 1 and 2. Module 1 is based on the repeated application of ultimate open pit limit analysis with various break-even cutoff grades. Module 2 involves period by period mining
sequence development subjected to practical operational constraints. Therefore, modules 1 and 2 inevitably deal with the following problems:

1. Ultimate open pit limit analysis;

2. Pushback incremental pit generation and inventory summary for ore, waste and grade values above a series of cutoff grades;

3. Mining sequence development.

Problem 2 can be easily solved when the solution of problem 1 is readily available though some engineering constraints such as mining width and pushback volumes need to be considered. Problems 1 and 3 require tremendous efforts to solve. Therefore, problem 1 (i.e., ultimate pit limit design) and problem 3 (i.e., mining sequence development) are discussed next.

1.2.1 Statement of Ultimate Pit Limit Design Problem

Besides being a useful tool in mining sequence development, the ultimate open pit limit design defines a maximum possible mining boundary which is necessary in locating surface facilities, designing dump locations and estimating capacity of processing plant. All the long range, medium range and short range mine planning will be confined within the ultimate pit limit.

"Briefly, the objective of the ultimate pit limit design (which is also frequently referred to as pit design in short) is to determine the projected final pit limits of an ore body and its associated projected grade and tonnage, which will maximize some pre-specified economic criteria while satisfying practical operational requirements" (Kim, 1978).

Kim (1978) lists some circumstances under which an attempt to select an optimal open pit limit should be made. These circumstances are directly quoted as follows:
"1) The pit is nearing its final stage in mine life and accurate cost as well as slope information can be utilized and/or
2) The deposit is relatively small (i.e., five to six year life) requiring mining to pit limits from the start, and/or
3) Surface is highly irregular and mineralization is highly spotty, thus requiring investigation of all possibilities”.

To quote Barnes and Johnson (1982): “the importance of an accurate ultimate pit limit plan is undeniable”. “It is the basic function of all surface mine planning”. “Though the ultimate pit limit analysis is extremely important, it may also be quite expensive in terms of computer time, memory requirements, and planning personnel time”. “A responsible pit limit analysis consumes a considerable quantity of computational effort”.

Thus, an approach which generates quality results within acceptable execution time would be very useful. The first purpose of this study is to examine the possibility of developing such an approach (algorithm).

1.2.2 Statement of Mining Sequence Development Problem

Although the pushbacks define a general direction for mine planning, i.e., the next best mineralization, practical situations simply may not permit mining the next best reserves. One constraint is stripping ratio. Since higher grade ore can carry more waste stripping than lower grade ore, the next best mining sequence may not provide smooth stripping ratios from one period to the next. Higher grade increments may be associated with higher stripping ratios and lower grade increments with lower stripping ratios. To maintain a constant mill supply, the stripping requirement in one period may be small but can go sky high in the period immediately following. Figure 1.5 shows the saw-tooth like stripping ratios.
Figure 1.5 The Saw-tooth Like Stripping Ratios (After Mathieson, 1982)
To make efficient use of equipment, stripping ratios must be smoothed out even though they may show an increasing trend from earlier to later planning periods. In this case, some advanced stripping may be necessary.

Practical considerations can put mine sequencing in an awkward situation. On the one hand, mine sequencing should follow the next best mining sequence as closely as possible. On the other hand, all those practical constraints such as stripping ratios, grade balancing, constant mill supply, sequencing road access, etc. must be considered as an integral part to provide a practical yet an “optimal” mining sequence.

The primary goal in mining sequence development is to locate the sequence which maximizes the NPV of the project subjected to economic, engineering, technological and geological constraints.

As mentioned in a previous section, optimization of the production schedule requires repeated application of part three (mine sequencing) in the pushback mine sequencing approach, where a mining sequence is developed based on one set of ore tonnage, waste tonnage and average grade for a single economic cutoff grade. Also, the developed mining sequence must satisfy the minimum ore exposure, smoothed stripping ratio and grade blending. For each variation of cutoff grade, one application of part three analysis is required. The more alternatives explored, the more likely the “optimal” mining sequence will be located, where the “optimal” mining sequence is the one which maximizes NPV among alternative mining sequences.

Although many alternative mining sequences are desired, developing even one mining sequence is not easy. For a given set of ore, waste and grade parameters, there is no guarantee that a feasible mining sequence exists even when subjected to only three out of the four constraints listed above. As mentioned by Gershon (1988), “the difficulty in achieving a practical schedule overrides the problem of finding the best schedule”.

So far, in developing one feasible mining sequence, mining engineers still use trial and error approaches. Although there usually is a general direction as to
how to proceed, there is no clear direction as to whether it is possible to achieve the goal until it is tried. Period by period, the work is done by trial and error. If the sequencing cannot proceed any further in any one period due to practical constraints, the whole trial and error process must be re-started. This is very frustrating, especially to those who just enter mine planning. One purpose of this dissertation is to investigate the possibility of developing a somewhat systematic approach to mine sequencing.

1.3 Scope of Study and Solution Techniques

As mentioned in a previous section, the purposes of this dissertation are twofold. The first is to examine the possibility of developing a pit limit design algorithm which generates quality results within an acceptable execution time. In 1986, Hanson developed a modified network flow algorithm for pit limit design. The algorithm follows network flow principles, yet adopts a simulation type approach which is similar to the popular moving cone method and has many attractive features such as being simple and easy to understand and to program. In a comparison study, Kim, et.al (1987) showed that Hanson's algorithm could not guarantee an optimal solution and occasionally generated less profit than a multi-pass moving cone heuristic in pit limit analysis. Nonetheless, Hanson's algorithm has the potential to provide quality results within reasonable execution times. This study intends to evaluate Hanson's modified network flow algorithm, to examine what is the best results that is obtainable based on simulation type pit design approach and to see if it is possible to develop an algorithm which generates higher quality results than Hanson's algorithm without increasing the execution time too much. The second purpose of this dissertation is to examine and develop a somewhat systematic mine sequencing strategy applicable during any particular mining sequence development.

With respect to the first purpose, a simulation oriented network flow algorithm is developed. This algorithm is an improved version of Hanson's algorithm
and employs a moving cone type simulation approach guided by maximum network flow principles. Based on two theorems developed in this study, the new algorithm combines the simplicity of moving cone with the rigor of maximum network flow principles.

The performance of the new algorithm is compared to that of the Lerchs-Grossmann algorithm (1965), moving cone algorithm and Hanson's algorithm (1986), via case studies. The performance comparison includes the generated profits and computer execution time. The above three algorithms are included here for the following reasons: 1) the Lerchs-Grossmann algorithm is a mathematically proven optimization algorithm; 2) the moving cone algorithm is the most popular method; 3) the new method is an improved version of Hanson’s algorithm and a comparison with the original algorithm can provide some idea on how much improvement has been made.

With respect to the second purpose, a zero-one programming model based on the sub-optimization concept is proposed. The proposed model provides some systematic feature in mining sequence development and reduces the current trial-and-error process to some extent. Given a set of pushbacks, with all the materials of each bench in every pushback being classified as ore and waste, the 0-1 programming mine sequencing model provides an optimized answer to questions such as: what is the highest mineral content (average grade) that is achievable for this period? What is the minimum waste tonnage that must be stripped in this period? All these answers are within the joint constraints of minimum ore tonnage and the precedence constraints among benches in the same pushback and the benches of different pushbacks but lying on the same elevation. The model provides the scheduler with enough flexibility by allowing the scheduler to adjust the zero-one program schedule to obtain an exact schedule. Multi-period mining schedules can be worked out by defining a sub-optimization objective for every mine sequencing period, then proceeding scheduling period by period. The discussion will be based on a multi-period mining schedule development for an actual mine sequencing case.
This model can reduce the burden of mining sequence development work.

This dissertation proceeds as follows: In chapter 2, long-range open pit mine planning and its solution methods are reviewed. In chapter 3, some theoretical background on maximum network flow problem and its solution technique are presented. This chapter provides necessary theoretical references for the subsequent chapter. Chapter 4 discusses the development of a simulation oriented network flow algorithm and its implementation performance. Chapter 5 presents the zero-one programming mine sequencing model. Chapter 6 illustrates the application of the 0-1 programming mine sequencing model using Diamond mine sequencing case. Chapter 7 contains the conclusions of this study and future research recommendations.
2.1 Basic Hierarchy of Mine Planning System

The primary objective of mine planning is to develop a timely schedule which tells when and where mining should take place for the depletion of ore reserves. Mine planning can be carried out completely by computers or partially by computers and partially by human-beings. The former is termed "automatic" mine planning, and the latter is termed "manumatic" mine planning. Both are referred to as computerized mine planning. Since the mid 60's, computerized mine planning began to play more and more important roles in mining industry. Today, mine planning work is mostly done by or assisted by computers.

Computerized open pit mine planning involves seven main tasks. They are presented in figure 2.1.

The seven mine planning tasks are briefly discussed below.

The geological survey is the first task in mine planning which includes the coding of surface topography, locations of roads, and surface facilities. Sample data base management includes entering and editing the drill hole survey data such as collar coordinates, azimuth, dip angle, rock type and mineral quality by intervals. Rock type and the mineral quality information are usually composited to bench intervals. The sample data base can be expanded when new information becomes available. The importance of this geological survey and sample data base should not be overlooked, because the quality of the initial data determines the quality of the whole mine planning.
When the geological survey and sample data base are complete, both the rock mechanics study and the mineralization modelling can be initiated.

The rock mechanics study provides information of maximum safety slope angle for relevant rock types in the deposit, ore grindability, rock drilling and fragmentation characteristics. Whereas the final pit slope angle has a tremendous economic impact on the ultimate pit limit analysis, both mining and milling costs are greatly affected by the ore grindability, rock drilling and fragmentation characteristics. Rock mechanics is itself an important branch of mining engineering. This part of study is usually carried out by the people who specialize in the field.

In mineralization modelling, a model first must be selected. One commonly stressed point in model selection is that a computerized model is a "model" and not "reality". Depending on the study objective, the required degree of transformation of reality into the model can be quite different. The more commonly applied models are 3-D fixed block model and Gridded seam model. The 3-D fixed block model
is best suited to massive, metallic deposits. The gridded seam model is best for tabular, coal deposits. Figure 2.2 shows a 3-D fixed block model. Table 2.1 lists the information contained in a block of a typical porphyry copper deposit.

Once the type of mineralization model is decided, sample information, such as rock type and quality of minerals is extended by interpolation. Commonly used interpolating techniques include; 1) polygonal, 2) triangular grouping, 3) inverse distance weighting, and 4) geostatistical, which is a statistical inference technique based on regionalized variable theory. Since its first appearance in the early 60's (Matheron, 1963), geostatistics has become the dominant interpolation technique in the mineral industry. This is because the technique provides not only a more accurate estimate but also a confidence limit on the estimate. With the introduction of indicator kriging and/or probability kriging, geostatistics can produce both an average estimate and a distribution of mineralization in any one block so that averages above series of cutoff grades are readily available. However, even geostatistical methods can not generate very reliable estimates without a "sufficient" number of samples. During long range mine planning, the number of samples based on diamond drill holes may be far from "sufficient". More discussions on mineralization modelling can be found in Ramani and Stanley (1979), Davey and Switzer (1979) and Bideaux (1979).

After the mineralization modelling is completed, minable ore reserves can be calculated by ultimate pit limit analysis based on break-even cutoff grades. Ultimate pit limit analysis defines a maximum possible mining boundary. All the subsequent mine planning works are performed within the pit limit. Section 2.2 contains further discussions on pit limit design.

Long range mine planning involves developing a mining sequence for the depletion of the minable ore reserves. The time span of each planning period in a long range mine plan could range from one year to five years. The objective of the mining sequence development is to maximize the net present value of the
Table 2.1 Information Contained in a Block of Typical Copper Porphyry Deposit. 
(After Crawford, and Davey 1979)

<table>
<thead>
<tr>
<th>Items</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copper ($Cu$)</td>
<td>%</td>
</tr>
<tr>
<td>Molybdenite ($MoS_2$)</td>
<td>%</td>
</tr>
<tr>
<td>Gold ($Au$)</td>
<td>oz/ton</td>
</tr>
<tr>
<td>Silver ($Ag$)</td>
<td>oz/ton</td>
</tr>
<tr>
<td>Copper Concentrate Recovery</td>
<td>%</td>
</tr>
<tr>
<td>Copper Concentrate Grade</td>
<td>%</td>
</tr>
<tr>
<td>Copper Smelting Recovery</td>
<td>%</td>
</tr>
<tr>
<td>Copper in Blister</td>
<td>%</td>
</tr>
<tr>
<td>Copper Refining Recovery</td>
<td>%</td>
</tr>
<tr>
<td>$MoS_2$ Concentrate Recovery</td>
<td>%</td>
</tr>
<tr>
<td>$MoS_2$ Conversion Recovery</td>
<td>%</td>
</tr>
<tr>
<td>Gold Concentrate Recovery</td>
<td>%</td>
</tr>
<tr>
<td>Gold Refining Recovery</td>
<td>%</td>
</tr>
<tr>
<td>Silver Concentrate Recovery</td>
<td>%</td>
</tr>
<tr>
<td>Silver Refining Recovery</td>
<td>%</td>
</tr>
</tbody>
</table>
project subject to practical operational constraints. The present value of a mining operation can often be improved by starting with a higher cutoff grade in the early years of mining and tapering to a lower cutoff near the end of the mine life (Lane, 1964; Blackwell, 1970; Marek and Welhener, 1985). Fundamentally, dynamic definition of cutoff grade is needed because "economic cutoff determination is a vital part of our operation, because we are not in the business to maximize copper production, or mining life, or to minimize unit costs, but rather to maximize profitability for our owners" (Fiore, 1986).

A short range mine plan is usually a subdivision of a long range mine plan with reduced ore and waste target tonnages. The time span of each period in a short range mine plan ranges from months to years. The objective of short range mine planning is to conform to more detailed practical operational constraints such as preventing premature cutoff of access roads, blending mill feed, etc., within the long range mine plan.

Production scheduling is at the bottom of the mine planning hierarchy. The time span of production scheduling ranges from a shift to daily or weekly. Production scheduling involves detailed and complex operational constraints, such as how to distribute shovels, manpower, materials and supplies, etc. The objective of production scheduling is to develop the most detailed mine plan conforming to the short range and long range mine plans.

The difficulty of mine planning increases as planning becomes more detailed. Therefore, the difficulties of mine planning increase from long range through short range to production scheduling.

As stated in section 1.2, this study is limited to ultimate pit limit design and long range mine planning. Therefore, ultimate pit limit design and mine sequencing are discussed in more detail next.
2.2 Ultimate Pit Limit Design

The procedures one has to go through during ultimate pit limit design (or pit design in short) are as follows:

1. Define pit design objectives and clarify pit design assumptions.
2. Set criteria for optimal pit design.
3. Build an economic inventory model.
4. Generate a coning scheme.
5. Carry out pit design.

These procedures are discussed next.

2.2.1 Pit Limit Design Objectives and Assumptions

The ultimate pit limit design gives a maximum possible mining boundary in three dimensions or the mineable ore reserves. One commonly employed objective in ultimate pit limit design is to maximize total cumulative net profit without considering the time value of money, while satisfying pit slope stability and operational constraints.

The assumptions required during ultimate pit limit design are listed below:

1) The selected mineralization model is a good representation of reality for pit design purposes. A pit design based on removal of profitable blocks and their overlying blocks is satisfactory in terms of final pit slope angle requirement.
2) Mining cost is independent of mining sequence.
3) Projected grade information for each block is accurate.
4) Final pit slope angle projections are reliable.
5) Current forecasts of product prices and costs are applicable during the entire projected mine life.
2.2.2 Criteria for Optimal Ultimate Pit Limit Design

To achieve maximum net profit, the design must meet the criteria below:

1. Revenue obtained from the sale of particular ore may be used to offset only the stripping cost that is absolutely necessary to expose the ore, while conforming to the pit slope angle requirement.

2. Profitable combinations of two or more conical mining regions, which are non-profitable when considered individually, must be detected. This condition ensures that no profitable regions will be left out of the design.

3. No waste block should be mined unless it restricts an ore block.

The restricting and non-restricting block concept shown in figure 2.3 is used to satisfy the first criterion. The shared contribution concept presented in figure 2.4 is used for the second criterion.

2.2.3 Economic Inventory Block Model

Before an ultimate pit limit design can be carried out, an economic block inventory model must be set up. In the economic block inventory model, the profit for each block is calculated as follows:

\[
\text{Gain} = \text{[product price ($/unit) \times grade (unit/ton) \times volume (cu-ft) \times tonnage factor (ton/cu-ft) \times recovery factor]}
- \text{mill cost ($/ton-ore)}
- \text{overhead and administration cost ($/ton-ore)}
\]
The ore block

The restricting blocks

The non-restricting blocks

Note: only random blocks are shown here to illustrate the restricting and non-restricting concept.

Figure 2.3 Concept of Restricting and Non-Restricting Blocks
(After Hanson, 1986)

Figure 2.4 Concept of Shared Contribution (After Hanson, 1986)
Stripping cost = \(-[\text{volume (cu-ft)} \times \text{tonnage factor (ton/cu-ft)} \times \text{mining cost ($/ton)}]\)

Net = Gain + Stripping cost
Block Value = \(\text{max}[\text{Net, StrippingCost}]\)

In the economic inventory model, a block is considered as an ore block if the profit value of that block is positive. Otherwise, the block is considered as a waste block. However, a waste block is classified into two categories. The first category contains pure waste blocks as well as low quality blocks whose mining as waste blocks would be less costly than sending them to the mill and processing them. The second category contains the blocks such that sending them to the mill and processing them is less costly than mining them as a pure waste. Therefore, if these second category of waste blocks are sent to the mill, they can pay portion of their own stripping. This classification of waste blocks into two categories is one reason for the above value calculation. Once the economic inventory model is available, one can proceed to cone generation.

2.2.4 Pit Slope Angles and Cone Generation

Final pit slope angles have a great impact on ultimate pit limit design. Slight variation of final pit slope angles can result in tremendous changes on profitability of the pit design. In an open pit operation, mining must proceed in such a way that the removal of blocks, be it profitable or not, must conform to final pit slope angle requirements. Any mining pattern which fails to do so may result in pit slope failures, which can be very costly.

Figure 2.3 shows an example of a mining pattern when only one ore block at the bottom is to be mined. It can be seen that when only one ore block at the bottom is to be mined, the mining pattern is an inverted cone with cone apex sitting on the ore block to be mined. A mining pattern with more than one ore blocks at
the bottom can be approximated by the union of individual cones generated above each ore block. An ultimate pit limit can be approximated by the union of all individual profitable cones. This concept was first reported by Pana (1965). Figure 2.5 shows the joint profitable cone approximation of an ultimate pit limit.

Some sort of cone generation routine can be found virtually in every open pit design program. The speed and flexibility of these routines determine the usefulness of a particular pit limit design program. Several methods of generating the cones have been presented (Pana, 1965; Lerchs and Grossmann, 1965; Johnson, 1968; Lemieux, 1977, Myers, 1980). The disk cone method seems to be the most popular. Several cone generating routines are presented in figure 2.6.

The disk cone method allows virtually unlimited slope variability. A cone can be represented by a series of horizontal planes, each plane having a cone boundary defined on it. Usually, the number of these horizontal planes coincides with the number of benches above the ore block under consideration. Several pit slopes can be specified by partitioning every horizontal planes into sections with each section representing one pit slope. The transformation of one slope to another in two adjacent slope sections is provided by linear interpolation. To test whether or not a block on a certain level is a restricting block to the ore block at the cone bottom, one simply checks whether or not the block is within the cone boundary on the corresponding level.

Generally, the blocks are considered to act at their individual geometric centers to determine whether a block is restricting to the bottom ore block or not. Whether to put the apex of the cone on top, in the middle or at the bottom of an ore block has an effect on the profitability of that cone. The conservative approach is to put the cone apex at the bottom of the ore block. Sometimes, even a cone base radius is enforced.
Figure 2.5
DEFINING PIT LIMITS WITH A SERIES OF CONES AND CONE SEGMENTS
Figure 2.6 Five Cone Generation Routines
(After Hanson, 1986)
2.2.5 Pit Limit Design via Moving Cone Heuristic

Once an economic inventory block model and the cone generating program are available, ultimate pit design can be started. In this section, the pit limit design is carried out via a moving cone method similar to the one reported by Pana (1965). The reason for choosing this method is because of its popularity and simplicity.

Pit design via moving cone method proceeds as follows:

1) Locate the first ore block in a top-down search fashion, place an inverted cone on this ore block and calculate the cumulative value of all the blocks falling in that inverted cone. If the cumulative value is non-negative, the cone is mined out by updating the initial model surface to a secondary surface with all the blocks in the mined out cone lying just above the new surface. Pit design profit is updated. The initial surface is saved throughout the pit design process for later ore reserve calculation. If the cumulative value of the first cone is negative, the cone is left untouched. After the cone of first ore block is checked, the inverted cone is moved to a second ore block. The cumulative value of all the ore and waste blocks falling in the second cone is calculated. If the cumulative value of this second cone is non-negative, this second cone is mined out by updating current pit surface. Again, the pit design profit is updated. Otherwise, this second cone is left untouched. The cone is then moved to a third ore block and the checking process is repeated until the cone of the last ore block is checked. This is the end of the first pass. Note that every time a cone is mined out, the pit surface is updated.

2) The second pass is started by placing an inverted cone on the first remaining ore block in a top-down search fashion. This second pass proceeds exactly like the first pass. This pass is needed because some cones that have been mined out in the first pass may have exposed new profitable cones. Once the second pass is completed, a third pass is started like the second pass and this multi-pass process continues until at any given pass there is no improvement on the pit design profit compared with previous pass. At this stage, the current surface is set to the ultimate pit limit. The blocks contained between the initial surface and the current
surface are mined out blocks. The current pit design profit is the maximum profit from the multi-pass moving cone pit design approach.

It can be seen that the above moving cone method simulates the actual mining process. It is simple, easy to understand and program. However, considering one cone at a time, the moving cone method can not detect shared contribution and it can over mine non-restricting waste blocks, both of which are shortcomings of this method. Figure 2.7 shows an example where the moving cone method fails to detect shared contribution and over mines at the same time.

![Diagram]

Legend: Value(A)=Value(B)=+10.0
Value(C)=+2.0; Value(E)=-9.0
Value(D)=Value(F)=-2.0
Value(G)=-5.0

Value(A)+Value(D)+Value(E)=-1.0
Value(B)+Value(F)+Value(E)=-1.0

But, Value(A)+Value(B)+Value(D)+Value(E) +Value(F) = -7.0; Note that

Value(A)+Value(B)+Value(C)+Value(D)+Value(E)+Value(F)+Value(G)=+4.0

Figure 2.7 Example of Moving Cone Method Missing Shared Contribution and Over-Mining at the Same Time.

2.2.6 Review of Current Pit Design Approaches

Ultimate pit limit design requires heavy computing and man power. This is due to the usual problem size. An open pit limit design problem often involves around 500,000 blocks for a 3-D fixed block model.

Ever since the first published paper by Lerchs and Grossmann in early 1965, the topic of optimum open pit limit design has received a lot of attention in academic
and research institutions. This fact becomes obvious if one examines the number of papers being published on this subject since 1965.

Kim (1978) classified the various pit design techniques into two main categories, i.e., rigorous and heuristic, where the word “rigorous” implies the availability of mathematical proof of optimumity and the word “heuristic” is used to describe an algorithm which works in nearly all cases but which lacks rigorous mathematical proof of optimumity.

The ultimate pit limit design algorithms developed so far can be listed as follows:

1. True optimizing algorithms
   a) Lerchs and Grossmann’s 3-D graph algorithm (Lerchs and Grossmann, 1965; Lipkewich and Borgman, 1969)
   b) Johnson’s Network Flow Algorithm (Johnson, 1968)

2. Moving cone heuristic algorithms
   a) Moving cone of Kennecott (Pana, 1965; Carlson, et al., 1966)
   b) Moving cone by Lemieux (1979)
   c) Moving cone by Mario and Slama (1973)
   d) Heuristic algorithm by Phillips (1972)
   d) Heuristic algorithm by Korobov (1974)
   e) Parameteric function by Bongarcon and Marechal (1976)
   f) Modified network flow algorithm by Hanson (1986)

3. Dynamic Programming Heuristic
   a) Lerchs and Grossmann 2-D dynamic programming (Lerchs and Grossmann, 1965)
b) Johnson's 3-D dynamic programming (Johnson, 1970; Johnson and Sharp, 1971)

c) Barnes' Best-Valued dynamic programming (Barnes, 1980 and 1982)

d) Braticevic 3-D dynamic programming (Braticevic, 1984)

Although there are many rigorous and heuristic algorithms available, the majority of the mining companies in U.S. still uses the moving cone heuristic by Pana (1965) in their pit limit design, despite the fact that the method has obvious shortcomings. The moving cone method is a simulation approach. The method is quite simple, easy to understand and runs fast on computer. Also, the moving cone method provides results that are satisfactory in nearly all practical cases especially in massive deposits.

The computation costs of the other algorithms are usually high. Hanson's algorithm has the merit of combining the simplicity of moving cone with the rigor of network flow. However, Hanson's algorithm can not guarantee an optimum solution and occasionally generates less profit than the multi-pass moving cone method (Kim, et al., 1987). Chapter four will evaluate Hanson's algorithm, examine how far the general simulation type pit design approach can go in generating quality results and try to answer if there exists an approach which lies between the moving cone method and a rigorous algorithm, and which can produce an optimum pit limit.

2.3 Mining Sequence Development

Mining sequence development determines the profitability of a mining venture. Different mining sequences extract ore reserves in different ways.

This study adopts the "pushback" mine sequencing approach (Iles, 1981; Mathieson, 1982; Marek and Welhener, 1985; Fiore, 1986). There are many variations of the pushback mine sequencing approaches. The approach discussed in section 1.1 is considered by this author to be a standard approach, because of its
popularity. Subsequent references of pushback mine sequencing in this text refer to this approach.

The procedures in “pushback” mine sequencing are listed below:

1. Set up objectives and assumptions.
2. Derive series of nested pits (pushbacks).
3. Develop various mining sequences based on different cutoff grades and mining rates and select the sequence that maximizes net present value of the mine.

2.3.1 Objectives and Assumptions

The main objective of mine sequencing is to maximize net present value of mining project. Other objectives are to meet grade-tonnage requirements, to meet precedence requirements among selected sequences, and to guarantee haul road accessibility. The decision variable can be defined as one bench or part of a bench.

The assumptions in mine sequencing are as follows:
1) Projected grade information for each decision variable is accurate.
2) Final pit slope angle projections are reliable.
3) Current forecasting of product prices and costs are applicable during all sequencing periods for the net present value analysis.
4) Each pushback is a minable area, containing all haulroads, conveyor routes, crusher status, and other geometric parameters necessary for field implementation. The pushback pattern can faithfully follow the pit slope angle requirement. Figure 2.8 shows an example of a pushback.
5) The access road to a sequence is inserted after a sequence has been developed. Currently, no pit limit optimization program includes access road as a constraint.
2.3.2 Generation of Series of Pushbacks

A series of nested pits can be generated by applying an ultimate pit limit design algorithm repeatedly with various cutoff grades. In general, one cutoff grade results in one pit. By varying cutoff grades, series of nested pits could be generated with a larger pit associated with lower cutoff grade and a smaller pit associated with higher cutoff grade. The inner most pit is the most profitable pit because it is based on highest cutoff grade. Each outer pit increment is less profitable than its inner pits. Therefore, profitability decreases from the inner most pit to the outer most incremental pit. If the horizontal distance between any two of those nested pits is greater than the minimum mining width, these nested pits can serve as a series of pushbacks. The pit slope angle used in the pushback generating routine is usually the final pit slope angle. Figure 2.9 shows a 2-D hypothetical pushback patterns.

If mining can proceed in exactly way in which the nested pits are generated, maximum net present value will be achieved. This is because each outer pushback represents the "next best" mineralization. Unfortunately, mining may not be able to proceed following an inner pit to outer pit sequence due to practical constraints.

2.3.3 Mine Sequencing among Pushbacks

Obviously, to achieve maximum net present value and satisfy the pit slope stability requirement, mining among pushbacks should proceed in such a way that the benches having the same elevation should be mined in an order from inner pushbacks to outer pushbacks. In other words, there is a precedence requirement or sequential requirement among the benches having the same elevation from different pushbacks. The benches in inner pushbacks have higher priority to be mined than the benches in outer pushbacks if those benches are on the same level.

Also, there is a sequential requirement among benches within the same pushback, i.e., within the same pushback, benches from higher elevation have higher
A pushback is a detailed design that incorporates all knowledge to date on haulage, slope stability, ore zones, etc. Pushbacks are designed in "hours."

Figure 2.8 Example of a Pushback (After Iles and Perry, 1981)

Figure 2.9 A 2-D Hypothetical Pushback Pattern
(After Marek and Welhener, 1985)
priority to be mined than benches from lower elevations.

The problem of sequencing among pushbacks can be defined as follows: to maximize the net present value of a mining sequence subject to sequential requirements among benches, ore, waste tonnage requirement (mining rate), grade blending requirement, and road access requirement.

Alternative mining sequences can be generated by varying cutoff grades. The optimal pit mining sequence is the sequence having the maximum net present value.

2.3.4 Review of Current Approaches

In contrast to ultimate pit limit design problem, there is little literature available on the mine sequencing problem.

Four types of approaches have been reported to handle the mine sequencing problem. These are:

1) Direct “optimum” sequencing. This approach works out an “optimum” mining sequence on individual block basis by directly applying operations research techniques. This type of approach can be seen in Johnson (1968), Davis and Williams (1975) and Dagdelen (1985).

2) “Cone miner” approach. This approach uses an interactive computer program referred to as “cone miner” to mine out material inside one cone. By moving this “cone miner” inside the mineralization model, a mining sequence can be defined. The “cone miner” is activated by defining several base points of a polygon and the desired cone slopes. This type of approach is reported by Pana (1965), Carlson, et al. (1966), Fairfield and Leigh (1969) and Mathieson (1982).

3) “Pushback” sequencing approach. This approach repeatedly uses an existing ultimate pit limit design algorithm on a mineralization block model to obtain series of nested pits. A particular mining sequence can be worked out based on ore reserves in each pushback. Finally, by varying the cutoff grades, production capacities, etc., alternative mining sequences can be obtained. The mining sequence which provides maximum net present value is the desired mining sequence. This
type of pushback approach is reported by Crawford (1976), Mathieson (1982) and Whittle (1988).

4) Suboptimization approach. This approach attempts to build some level of intelligence to guide the scheduling process. For instance, one way to carry out mine sequencing is to identify the most troubling long term problem and then define multi-period objectives to overcome it, or at least to minimize the trouble that it causes. This type of approach is reported by Gershon (1987, 1988). In reporting the suboptimization concept, Gershon (1988) suggested that, for a 3-D fixed block model, each block in the model should be assigned a desirability weight according to the long term objective. These desirability weights can be dollar values of each block or some other factors. The desirability weight is defined in Gershon’s approach as the value that is to be revealed in the future according to the pit slope restrictions by removing the block under consideration. This desirability weight concept is somewhat like Hanson’s support concept except that the direction of the cone expansion is reversed. Hanson’s support concept is discussed in chapter four. Once the desirability weights for every block in the model are available, one may proceed to mine sequencing following the guideline of the desirability weights.

2.3.5 Remarks on Current Approaches

In this author's opinion, direct “optimum” mine sequencing will not prevail in industry using today’s technology, computing facilities and operations research techniques.

Mathieson (1982) provides a number of cogent observations on the direct “optimum” mine sequencing approach. Some of his most important observations are as follows:

“They generate ‘optimum’ sequences and schedule via a network type of block aggregation under very simplistic mining constraints.”

“They may be based on an ‘optimizing’ procedure which operates from the last mining period toward the present. As such, they do not look forward in time to
locate and mine the ‘next best’ ore in sequence. The latter approach is intuitively more likely to yield the maximum return possible on the property since the phasing order was chosen to be the best, irrespective of changes in metal prices, costs, grade estimates, or discount rates.”

“The automatic scheduling systems developed to date have not enjoyed very wide application due to their relatively high computing cost, and their inability to properly simulate real open pit mining situations.”

The “cone miner” sequencing approach requires a lot of computer searches even for an experienced person. The method is based entirely on a trial-and-error.

The “pushback” mine sequencing is so far the most popular approach in mining sequence development. This is because finding the “next best” ore seems to be the most logical approach in a mining project. Unfortunately, in pushback mine sequencing which requires alternative mining sequences based on different cutoff and production capacities, each mining sequence must be worked out completely by a trial-and-error approach.

Gershon’s approach is probably better for short term production scheduling problems. For long term mine sequencing problem, not much information can be gained from this approach in most practical situations because of the averaging effects. For example, when the cone goes deeper and deeper, the cone becomes larger and larger. Once several cones cover the same area, the values of the blocks at the apexs of these cones will not be large enough to convince the scheduler.

However, the idea of a suboptimization has great value. Combining the suboptimization concept with the popular pushback approach, one may be able to develop a systematic and practical mine sequencing approach. Being able to answer quickly what is the best that can be expected based on available resources would be very helpful to a scheduler. Chapter Five will develop a model which introduces systematic features into the mining sequence development and reduces the current trial-and-error process to some extent.
CHAPTER 3

PIT LIMIT DESIGN VIA NETWORK FLOW TECHNIQUES

It is well known that an open pit limit design problem can be re-formulated and solved as a network flow problem (Johnson, 1968; Picard, 1976). Net profit from pit design is maximized when network flow reaches the maximum flow. However, several difficulties have been identified with respect to the direct application of network flow to the pit limit design problem (Johnson, 1973). These are: 1) difficulty with variable wall slopes, 2) difficulty with different block sizes, and 3) difficulty with re-allocation of flow from one negative block to another. These difficulties have precluded the method from even becoming a research topic.

This chapter examines the general maximum network flow problem and its most efficient solution algorithm, the Dinic-Malhotra-Kumar-Maheshwari (DMKM) algorithm (Syslo et al, 1983), so that necessary background for the next chapter is provided. The network flow formulation of the pit limit design problem is also presented.

3.1 The Maximum Network Flow Problem

Sending objects from one place to another place in an optimal fashion has great practical value. For instance, the movement of the maximum amount of water, oil or gas through a network of pipelines to stations, the shipment of a maximum possible quantity of products from a factory to markets, can all be regarded as sending maximum flows through networks. These types of problems have a common feature: how to maximize the flow from a given node \( s \) (source) through intermediate nodes to another given node \( t \) (sink), in a network in which each arc \( a_{ij} \) has a
specified capacity $c_{ij}$ — the maximum flow that can pass through arc $a_{ij}$ ($c_{ij} = \infty$ if the capacity of $a_{ij}$ is unlimited).

Formally, the maximum network flow problem (or maximum flow problem in short) may be stated as follows: Let $G = (V, A)$ be a directed network (or directed graph) in which every arc $(i, j)$ is assigned a capacity $c_{ij}$. Also, $V$ is a set of nodes and $A$ is a set of arcs. A flow pattern (or simply a flow) is an assignment of a non-negative flow $f_{ij}$ to every arc $(i, j)$ such that the following conditions are satisfied:

1. For every arc $a_{ij}$ in $G$,

   \[ 0 \leq f_{ij} \leq c_{ij} \]  

2. There is a specified node $s$ in $G$, called source, for which

   \[ \sum_i f_{si} - \sum_i f_{is} = F \]  

   where $\sum_i f_{si}$ is the flow over all the outgoing arcs from node $s$ and $\sum_i f_{is}$ is the flow over all the incoming arcs into node $s$. Quantity $F$ is called the value of flow or flow value.

3. There is another specified node $t$ in $G$, called the sink, for which

   \[ \sum_i f_{ti} - \sum_i f_{it} = -F \]  

4. For each of the remaining nodes $k$, called intermediate nodes,

   \[ \sum_i f_{ki} - \sum_i f_{ik} = 0 \]  

Equation (3.1) states that the flow through any arc does not exceed its capacity. Equation (3.4) states that the aggregate flow that enters each intermediate node equals the aggregate flow leaving that intermediate node. The other two conditions state that the net flow out of the source $s$ as well as the net flow into the
sink $t$ equals flow $\mathcal{F}$. Therefore, the flow $\mathcal{F}$ is called the value of the flow through the network. The maximum flow problem is that given $G, s, t$ and $c_{ij}$'s, find a flow pattern such that the flow $\mathcal{F}$ from $s$ to $t$ is a maximum, subjected to equations (3.1), (3.2), (3.3) and (3.4).

A simple maximum network flow problem is shown in figure 3.1.

![Figure 3.1 An Example of a Maximum Network Flow Problem](image)

In figure 3.1, there are four nodes and five arcs. Every arc is associated with a bracket. The numbers on the left inside the brackets represent capacities of the corresponding arcs. The numbers on the right are the flows assigned to the corresponding arcs. Maximum flow is obtained by sending 4 units along path $s \rightarrow a \rightarrow t$ and 5 units along path $s \rightarrow b \rightarrow t$. The maximum flow is 9 units.

The criterion of maximum flow is given by an important theorem of network theory known as the max-flow min-cut theorem. The max-flow min-cut theorem states that, for any network with a single source and sink, the maximum feasible flow from source to sink equals the minimum cut value for all the cuts of the network. A cut may be defined as any set of oriented branches containing at least one branch from every path from source to sink. The cut value is the sum of the flow capacities of the branches (in the specified direction) of the cut. This means that as long as
any one oriented branch belonging to the minimum cut is not saturated (the flow assigned to this branch equals the branch capacity), the network flow can not be a maximum flow. Therefore, a maximum flow algorithm must be capable of locating every existing path from source to sink and saturate it. Otherwise, the algorithm can not produce an optimal solution.

One of the minimum cuts of figure 3.1 network is the one which separates the source $s$ and the sink $t$ such that one set contains $s$, $a$, $b$ and the other set contains $t$. The cut value is the sum of $c_{at}$ and $c_{bt}$, i.e, 9 units. This proves that the 9 units is the maximum flow for figure 3.1 network.

3.2 DMKM Algorithm for Network Flow Problem

Several rigorous algorithms have been proposed for the solution of the maximum flow problem (Ford and Fulkerson, 1956; Edmonds and Karp, 1972; Dinic, 1970; Malhotra et al., 1978). The algorithm by Malhotra et al. is the simplest and probably the most efficient (Syslo et al., 1983). Since the algorithm by Malhotra et al. is based on Dinic's basic approach, the algorithm is also called Dinic–Malhotra–Kumar–Maheshwari (DMKM) algorithm.

The main idea behind the DMKM maximum flow algorithm (in fact, in a broad sense, behind all maximum flow algorithms, [Syslo et al., 1983]) is to start pushing a certain amount of material from source $s$ progressively through to sink $t$. Then look at the "useful" or "unsaturated" arcs in the network (arcs that can be used for additional flow from $s$ to $t$) and push some more. Continue pushing until no more material can be pushed through.

This pushing must, however, be done systematically and carefully; otherwise, one may fall into a trap. For example, in figure 3.1, if 2 units are pushed through path $s \rightarrow b \rightarrow a \rightarrow t$ and 3 units are pushed through path $s \rightarrow b \rightarrow t$, then only 2 units can be pushed through path $s \rightarrow a \rightarrow t$. Thus, the network flow is 7 units which are not the maximum flow for the network. However, if the 2 units' flow
on path $s \rightarrow b \rightarrow a \rightarrow t$ are returned to the source and re-allocated through path $s \rightarrow b \rightarrow t$, a maximum flow of 9 units can be obtained. Note that if at the beginning of the pushing process one pushes 5 units through path $s \rightarrow b \rightarrow t$ and then pushes 4 units through path $s \rightarrow a \rightarrow t$, the maximum network flow of 9 units would have been obtained and the need for re-allocation could have been avoided.

Can this re-allocation work be avoided? The answer is generally no. In a complex network with thousands of nodes and arcs, it is almost impossible to find a "perfect" pushing sequence so that the re-allocation work can be avoided completely. Therefore, a systematic pushing scheme and the capability of re-allocation become necessary conditions for any algorithm to the maximum flow problem. When an algorithm is incapable of re-allocating the flows, the algorithm can not achieve an optimal solution in a rigorous sense. This is, in fact, the theoretical basis for the two theorems and one algorithm to be discussed in the subsequent chapter.

The pushing scheme of the DMKM maximum flow algorithm amazingly resembles the allocation process of simulation oriented pit limit design algorithms. Therefore, the DMKM algorithm is discussed in more detail next.

The DMKM algorithm proceeds as follows:

1. Extract a layered acyclic network $G_\ell = (V_\ell, A_\ell)$ out of the current network. In the first iteration, this current network is the initial or original network $G(V, A)$. In this layered acyclic network, all the nodes $V_\ell$ are partitioned into "layers," $V_1, V_2, ..., V_{\ell}$. The first layer, $V_1$, consists of only the source $s$. Layer $V_2$ consists of all nodes that are immediate successors of $s$ (i.e., at a distance 1 from $s$). $V_3$ consists of all nodes that are immediate successors of nodes in $V_2$. And so on. The $i$th layer $V_i$ consists of all nodes at a distance $(i - 1)$ from $s$. Finally, $V_{\ell} = t$. Thus every node in a layered network lies on a path from $s$ to $t$, and all paths from $s$ to $t$ are of the same length $\ell - 1$. Every arc in a layered network is connected from some node in the $i$th layer to some
other node in the \((i+1)\)th layer, and there is no arc between two nodes within the same layer. Note that the connecting arcs between layers are allowed to point toward the source node \(s\). One layered network based on the figure 3.1 network is shown in figure 3.2.

Figure 3.2 An Initial Layered Network for Figure 3.1 Network

2. For each node \(v\) in the given layered network, determine the maximum additional flow that can be pushed through node \(v\). This maximum additional flow is called the potential \((v)\). The potential \((v)\) is determined by first finding the total additional flow that can go into node \(v\) [in-potential \((v)\)] and the total additional flow that can possibly flow out of \(v\) [out-potential \((v)\)] and then taking the smaller of the two as the potential \((v)\). Clearly, node \(v\) can handle no more than potential \((v)\) amount of flow. This potential is calculated for all nodes except \(s\) and \(t\). For the source \(s\), the potential \((s) = \text{out-potential} (s)\) and for the sink \(t\), the potential \((t) = \text{in-potential} (t)\). Next, a reference node \(r\) is located such that the node \(r\) has the smallest potential among all nodes of \(G_t\). In figure 3.2, potential \((s) = 10\), potential \((t) = 9\), potential \((a) = 4\), and potential \((b) = 5\). The reference node is \(a\).
3. Establish a flow of potential \((r)\) from the source \(s\) to the sink \(t\). This is done by first pushing potential \((r)\) units of flow into the outgoing arcs from \(r\) — taking these arcs one by one and saturating them until all units are finished — thus pushing potential \((r)\) units to the next higher layer. The flow now reaching the next layer can be distributed among their outgoing arcs and pushed to the next layer. This process of pushing potential \((r)\) units to the next higher layer is continued until sink \(t\) is reached. Note that the potential \((r)\) is the minimum of all potentials, no supply of pushes should get stuck at any intermediate nodes. All the potential \((r)\) units should arrive at the sink \(t\). Similarly, potential \((r)\) units of flow can be pulled from the immediate predecessor of node \(r\), which in turn can pull the amount from their predecessors, and so on, until this amount is pulled out from the source \(s\).

4. Delete all the saturated arcs from \(G_t\). Also, a node that has either all its incoming or outgoing arcs saturated is deleted from the network. Deletion of a node causes the deletion of all its incoming and outgoing arcs. The effects of these deletions will be reductions of in-potentials and out-potentials of various nodes in the network. This completes one iteration. If there is any \(v\) belonging to \(V_t\) left, go to step 2 to start a new iteration. If there is no \(v\) belonging to \(V_t\) left, the current layered network has been saturated. Update the original network with the current saturated layered network, i.e., update the flow through each arc of the original network according to the flow through the corresponding arc of the current saturated layered network. Then, check if there are any usable arcs (as explained on next page) connecting the source and the sink in the original network. If there are, go to step 1 to extract a new layered network out of those usable arcs. Otherwise, stop. The maximum flow of the network \(G(V, A)\) is obtained by summing all the flows.
Remarks on the DMKM maximum flow algorithm:

1. In every new iteration, a reduced network and thus a new layered network \( G_\ell(V_\ell, A_\ell) \) results. The potentials of every node and a new reference node \( r \) must be recomputed.

2. In extracting a layered network from an existing network, the distance from any one node to the source node equals the number of arcs between this one node and the source node. There is always a connection between some node in the \( i \)th layer and some other node in the \((i + 1)\)th layer. There is no connection between the nodes within the same layer. In the DMKM algorithm, a usable arc \((i, j)\) is said to exist from node \( i \) to \( j \) if (1) either the network has an arc \((i, j)\) whose capacity \( c_{ij} \) is greater than the currently assigned flow \( f_{ij} \) or (2) if there is an arc \((i, j)\) in the network with some non-zero reverse flow \( f_{ji} \). In both cases, the effective flow from \( i \) to \( j \) can be augmented (added): in case (1) by adding more forward flow, and in case (2) by reducing some reverse flow. Therefore, as long as there are surplus supplies at the source and there is an unsaturated path connecting the source and the sink, a layered network \( G_\ell(V_\ell, A_\ell) \) will be extracted on this path sooner or later until all the unsaturated paths are saturated. In this way, the maximum flow is warranted. Being able to locate a usable arc \((i, j)\) is the re-allocation capability of the DMKM algorithm.

3. In any iteration, all incoming or outgoing arcs at the reference node \( r \) will be saturated so that at least one node gets deleted in each iteration. For a \( n \) node network, the number of iterations for one layered network is bounded by \( n \), i.e., the number of iterations could reach \( n \). Several layered networks may have to be extracted from a given network before a maximum flow pattern is achieved.

4. The initial computer storage requirement of \( G(V, A) \) includes: \( n \) nodes, \( m \) arcs, \( m \) capacities and \( m \) assigned flows, where \( n \) is the number of nodes in \( G(V, A) \) and \( m \) is the number of arcs in \( G(V, A) \).
3.3 Network Flow Formulation of Pit Limit Design Problem

The network flow method was first applied to pit limit optimization by Johnson in 1968. A pit limit design problem can be transformed into a network with positive valued (ore) blocks and a dummy source on one side and negative valued (waste) blocks and a dummy sink on the other side. Basically, each block is denoted by one node. Each positive block is connected to all negative blocks that must be removed to mine the positive block according to pit slope requirements. Every positive block is connected to a dummy source and every negative block is connected to a dummy sink.

The direction of the arcs in the network is drawn as follows: all the arcs connecting the source and the positive blocks point from the source to the positive blocks. All the arcs connecting the negative blocks and the sink point from the negative blocks to the sink. Finally, the arcs connecting the positive blocks and the negative blocks point from the positive blocks to the negative blocks. The arc direction determines the flow direction on that arc.

The arc capacities are defined so that the source has an unlimited capacity to send. The sink has unlimited capacity to receive. The arcs connecting the source and the positive blocks have capacities of the corresponding positive block values so that no more flows than the ore block value are allowed. The arcs connecting the negative blocks and the sink have capacities of the corresponding negative block values so that no more flows than the waste block value are allowed. Finally, the arcs connecting the positive blocks and the negative blocks have unlimited capacities because if a waste block is a restricting block to an ore block, it must be mined out at any cost to mine the ore block below.

A 2–dimensional example of transforming a pit design problem into a network flow problem is given in figure 3.3.

In general, pit limit design based on the network flow approach involves the following steps:
1. Transforming a pit limit design problem into a network flow problem such as the one shown in figure 3.3.

2. Using the given network, work out a maximum flow pattern by pushing the unlimited source flows through its successors to the sink until the network flow becomes a maximal.

3. If there are any negative valued nodes whose arcs are connected to the sink and have unused capacities, go to step 4. Otherwise, go to step 5.

4. Delete all the negative valued nodes whose arcs have unused capacities from the pit design network, together with their preceding positive valued nodes.

5. The optimum solution to the pit limit design problem is obtained. Locate the arcs that connect the positive blocks and the source and sum up all the residual capacities. This sum is the net profit from pit design. The nodes in the current network constitute the blocks within the optimum open pit limit.

Note that a rigorous maximum flow algorithm is capable of detecting any unsaturated path between the source and the sink.

3.4 Difficulties in the Direct Application of the Maximum Flow Algorithms

As can be seen in previous sections, the direct network flow approach to pit limit design requires an extremely complex data structure. For each node in the network, one has to keep a record of its successors and predecessors besides keeping records of the arc capacities and flows. A typical pit limit design problem may involve 500,000 blocks which would need 500,000 nodes to represent them. In applying the DMKM algorithm, if the maximum flow problem is bounded by \( n \) iterations and if one iteration can be processed in one second on average, the total time required would be 5.79 days. In addition, several layered networks may have to be extracted. This is a tough job even for mainframe computers. As such, the difficulty of direct application of rigorous network flow algorithms to the pit limit
a: Original Value Matrix

<table>
<thead>
<tr>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.0</td>
<td>-2.0</td>
<td>-1.0</td>
<td>-3.0</td>
<td>-5.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#6</th>
<th>#7</th>
<th>#8</th>
<th>#9</th>
<th>#10</th>
</tr>
</thead>
<tbody>
<tr>
<td>+4.0</td>
<td>-1.0</td>
<td>-1.0</td>
<td>-2.0</td>
<td>+1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#11</th>
<th>#12</th>
<th>#13</th>
<th>#14</th>
<th>#15</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5.0</td>
<td>+4.0</td>
<td>+1.0</td>
<td>+1.0</td>
<td>+1.0</td>
</tr>
</tbody>
</table>

b: Transformed Network (after Johnson, 1968)

Figure 3.3 A 2-D Example of Pit Design Network (After Johnson, 1968)
design problems can be easily appreciated.

The direct application of network theory to pit design faces great hurdles. As established in chapter Two and in the current chapter, the popular moving cone method needs to come closer to an optimum and the rigorous algorithms need to be less complicated. Is there an algorithm that can serve as a compromise between these two? Is the network theory discussed in this chapter enough of a hint to lead to such an algorithm? How far can the simulation oriented pit limit design algorithms be improved in optimizing pit design? The next chapter attempts to answer these questions.
CHAPTER 4

DEVELOPMENT OF A SIMULATION ORIENTED NETWORK FLOW ALGORITHM FOR OPEN PIT LIMIT DESIGN PROBLEM

In 1986, J. Hanson proposed a modified network flow algorithm which has the merit of combining the rigor of network flow with the simplicity of the moving cone method. Although his method touched the concept of network flow, it was mostly a moving cone type simulation method. Because of the simplicity and the popularity of the moving cone method in pit limit design practice, an improved moving cone type algorithm that consistently generates more profit at acceptable computing cost is of great practical interest. Unfortunately, case studies by Kim et al. (1987) show that Hanson’s algorithm missed the optimum solutions in all cases, and the algorithm occasionally generated less profit than the multi-pass moving cone method.

This chapter, based on the maximum flow principles presented in the previous chapter, will show why Hanson’s algorithm can not reach the optimum solution and what Hanson’s algorithm really means in terms of network flow theory. This chapter will also answer the question of how far the moving cone type pit design algorithms can be improved for optimizing the pit limit design. Finally, a simulation oriented network flow algorithm is developed following the guidelines of the theorems developed in this chapter. The performance of the new algorithm will be compared with that of the rigorous Lerchs-Grossmann (LG) algorithm, the multi-pass moving cone method (MC) and Hanson’s modified network flow (HA) algorithm through the use of practical case studies.
4.1 Hanson's Modified Network Flow Algorithm

Hanson's modified network flow algorithm is characterized by two key aspects. One is the support concept and the other is the Lowest-Support-First-Priority theorem.

4.1.1 Concept of Support

According to Hanson, "a support is defined as the number of positive valued blocks which underlie a particular block and whose removal necessitates the removal of the supported block in question" (Hanson, 1986). This particular block which limits the mining of the underlying positive valued blocks is called a restricting block. The support and restricting block distinction is presented in figure 4.1.

Figure 4.2 shows a cross section of a 3-dimensional support pattern. In figure 4.2, 3 positive valued (ore) blocks are represented by A, B, and C. An inverted cone is generated above each ore block. In some regions, these cones overlap. Regions 1-a, 2-a, 3, 2-b, and 1-b are the supported regions. Regions 1-a and 1-b are supported solely by ore block A and B respectively. Region 2-a is supported jointly by ore blocks C and A. Region 2-b is supported jointly by ore blocks A and B. Finally, region 3 is supported jointly by ore blocks A, B, and C. Therefore, the support of region 1-a and region 1-b is 1, the support of region 2-a and 2-b is 2 and the support of region 3 is 3.

4.1.2 Lowest Support First Priority Theorem

Hanson's Lowest Support First Priority theorem states: "Allocation of flow from ore blocks to their overlying waste blocks must take place in such manner that waste with the least support is paid off first, in order of the amount of support each waste block possesses" (Hanson, 1986).

This theorem is based on the fact that if a waste block is supported by fewer ore blocks, that waste block may have a lesser chance of being mined.
The ore block

The restricting blocks

The non-restricting blocks

Note: only random blocks are shown here to illustrate the restricting and non-restricting concept.

Figure 4.1 Support and Restricting Block Distinction (After Hanson, 1986)

Figure 4.2 Support Concept (After Hanson, 1986)
Therefore, its removal should be considered earlier than other waste blocks with higher support.

The Lowest Support First Priority theorem can be applied to the mining situation of figure 4.2 as follows: In order for ore block A to provide any support to regions 2 and 3, its value must be at least sufficient to offset the mining of 1-a. Similarly, before region 3 can receive any support from block A, region 2-a must also be completely supported, either entirely by block A or jointly by both A and C.

4.1.3 Hanson's Algorithm

Based on the support concept and the Lowest Support First Priority theorem, Hanson's modified network flow algorithm proceeds as below:

1. Set up a scheme which generates a cone such that its apex can be placed on any ore block with top intersecting the topography of mineralization model.

2. Generate a support matrix such that given any block in the model, the number of ore blocks that support this block is known. This number is called support of the block. If the block is a waste block, its support value is negative. If the block of concern is an ore block, its support value is positive. An ore block is a support to itself. Therefore, the number of supports of an ore block counts itself. The support matrix for figure 4.2 situation would be like this: the waste blocks in regions 1-a and 1-b have a support value of -1, indicating that each of them is supported by only one ore block, and the blocks in this region are waste blocks. The waste blocks in 2-a and 2-b have a support value of -2, indicating that each of them is supported by two ore blocks, and the blocks in this region are waste blocks. Similarly, the waste blocks in region 3 have a support value of -3. The support for ore blocks A, B, and C are 1, 1, and 2 respectively. The support values of these three blocks are positive because they are ore blocks. The support value of C is 2 because the block C is supported by both A and block C.

3. Locate the first ore block in a top-down search fashion. Place an inverted
cone on this ore block. Allocate the positive value of this ore block up to its overlying waste blocks to offset their negative values in such a way that the blocks with least support in the cone are allocated first, then the blocks with second least support, ..., as long as there is positive value left. The allocation process for this ore block stops when either of the following occurs: a) there is no positive value left to allocate, b) all waste values overlying this ore block are offset to zeros. The values of the allocated blocks are updated to prevent double counting. After the value of the first ore block is allocated, the inverted cone is moved to the second ore block. The positive value of this ore block is allocated up to its overlying waste blocks to offset their negative values, in the same manner as for the first ore block. Again, the value of the allocated blocks is updated to prevent double counting. The cone is then moved to the third ore block and the process is repeated until the value of the last ore block is allocated. This is the end of the first iteration.

4. Check whether any negative value is still left among the restricting blocks overlying any ore block. If there are no negative values, stop. The optimum solution is obtained. Otherwise, any ore block with negative overlying blocks is deleted from the support matrix to avoid further consideration of this ore block. This is done by deleting this ore block and reducing the support of each overlying block by 1. Note that the support matrix contains possible solution set. Therefore, deletion of an ore block from the support matrix is analogous to removing this ore block from the possible solution set.

5. The second iteration, with an updated support matrix, is started after resumming profit values of each block that remain in the support matrix to their original values. This second iteration proceeds exactly like the first one. This iteration is needed since the deleted ore blocks may have helped to offset waste values for those ore blocks remaining in the support matrix. In other words, some non-profitable ore blocks may remain in the possible solution set because of the contribution from the deleted ore blocks. This iteration thus guarantees that no over mining will take place during pit limit design.
At the end of the second iteration, the negative value over any ore block is again checked. If there is no negative value over any ore block, then the "optimum" solution is obtained. Otherwise, the ore block is deleted and the support matrix is updated. Then, a third iteration is started. Iterations continue until there is no negative value over any ore block.

At this point, Hanson's algorithm stops. The union of all remaining ore blocks in the support matrix and their restricting blocks is the ultimate pit. Note that an updated permanent pit surface is obtained only at the end of Hanson's algorithm.

4.1.4 Non-Optimum Proof of Hanson's Algorithm

Like the DMKM maximum flow algorithm, Hanson's modified network flow also pushes ore values to their successors, i.e., waste blocks. This is done by using ore values to offset the waste values.

Locating an ore block and using the value of this ore block to offset the values of all its restricting waste blocks is equivalent to pushing flows on all the forwarding paths from the source, then through the ore block node and through the successors of this ore block node in a network flow problem. A forwarding path is defined here as a path on which every arc is pointing toward the sink. The network flow formulation of the pit limit design problem is such that all the ore blocks and the source are on one side, all the waste blocks and the sink are on the other side, every ore block is connected to the source and every waste block is connected to the sink.

The successors of an ore block are traced by generating an inverted cone on the ore block. If a waste block falls inside the inverted cone, i.e., this waste block is a restricting block for the ore block, this waste block is a successor of the ore block and the ore block is the predecessor of this waste block.

The value of an ore block being exhausted is analogous to the flow through a source-ore arc reaching the capacity limit. Similarly, the value of a waste block being completely offset is analogous to the flow through a waste-sink arc reaching
the capacity limit.

Basically, what the Lowest Support First Priority theorem implies is that, if a waste node has fewer predecessors, a flow should be augmented through this node earlier than other waste nodes having more predecessors. Hanson’s theorem defines a flow augmenting sequence among waste nodes on the forwarding paths. The effect of the Lowest Support First Priority theorem is to reduce re-allocation situations, which will be discussed later in this chapter. The theorem itself is not sufficient to completely eliminate the need for re-allocation. This can be seen

Figure 4.3 Hanson’s Theorem does not Eliminate the Re-Allocation Problem through an example in figure 4.3. In figure 4.3, following a top-down search fashion, the value of ore block A will be allocated first, then ore block B and finally ore block C. For ore block A, 10 units are allocated to its least supported region H; since both region F and region I have support of -2, the value allocated to these two regions is somewhat arbitrary, i.e., region F is allocated 3 units and region I also 3 units.
This used up the 16 units available at ore block A. For ore block B, all the available 10 units are allocated to its least supported region J. For ore block C, 10 units are allocated to its least supported region E, 2 units are allocated to its second least supported region F (note that after 3 units of ore block A are allocated to region F, this region has -2 units left to be paid off), and finally 2 units are allocated to its third least supported region G. This sequence of allocation left -2 units in region I, which resulted in the deletion of ore blocks A and B from the possible solution set. For this example, Hanson's algorithm mines only ore block C, with a net profit of 13, but the optimum solution is to mine all three ore blocks, with net profit 14. The reason for Hanson's algorithm missing the optimum is because the Lowest Support First Priority theorem does not eliminate the re-allocation problem. Again in figure 4.3, if 2 more units of ore block C are used to payoff the region F so that 2 units of ore block A are released from region F to payoff the -2 units in region I (i.e., if 2 units of ore block C are re-allocated back to ore block A to payoff the -2 units in region I), the optimum solution would have resulted.

As mentioned in the previous chapter, systematic pushing and re-allocation capability are necessary conditions for an optimum routine in a network flow problem. The Lowest Support First Priority theorem can not eliminate the re-allocation problem. Hanson's algorithm does not have the re-allocation capability and as a result does not optimally allocate. The problem is that, when an ore block has negative overlying values, this does not necessarily mean that there is no path between this exhausted ore block and other ore blocks that have residual capacities left. A path between the source and the sink includes not only the forwarding path but also the path on which some arcs' direction may be reversed (the arc pointing toward the source).

Deleting ore blocks from the possible solution set in Hanson's algorithm results in deleting potentially profitable blocks from the possible solution set. This is one reason why Hanson's algorithm missed the optimum solution. A more fundamental reason for Hanson’s algorithm missing the optimal solution is the lack of a
re-allocation scheme.

4.2 Non-Optimum Theorem on General Simulation Oriented Pit Design Algorithms

This section introduces the first of the two theorems of this chapter - the Non-Optimum Theorem on the general simulation oriented pit design algorithms. In this study, a simulation oriented pit design algorithm is defined as follows: an algorithm in which a restricting block of an ore block is traced according to whether the block falls within or outside an inverted cone generated above the ore block and in which there is no explicit definition of successors and predecessors.

It has been proven that Hanson's Algorithm lacks the re-allocation process. This re-allocation capability can not be inserted easily into Hanson's algorithm. In fact, it is difficult to insert the re-allocation capability to any type of simulation oriented pit design algorithms which lack explicit definition of successors and predecessors. On the other hand, an explicit definition of successors and predecessors leads to the rigorous graph theory algorithms such as the Lerchs-Grossmann 3-dimensional graph algorithm and the Network Flow algorithm.

It seems that the predecessors of a waste block can be traced by generating a cone whose extending direction is opposite to the normal inverted cone, i.e., if the normal cone is an upward cone, this opposite cone would be a downward cone as shown in figure 4.4. All the blocks that are covered within this cone can not be mined without the removal of the waste block at the apex of this cone. The ore blocks falling within this cone necessarily support this waste block. Figure 4.5 presents one situation in which one augmented path can possibly be traced from a surplus ore block to a waste block needed to be offset in simulation oriented pit design algorithms. The figure 4.5 example is the same as the figure 4.3 example except that waste block K and waste block L are added together with their corresponding downward cones. The purpose of the figure 4.5 example is to re-allocate to block
L the value that has been previously assigned to block K. Starting from ore block C, one finds the waste block K via an upward cone. From waste block K, one finds the ore block A via a downward cone. From ore block A, one finds the waste block L via an upward cone.

Figure 4.4 An Upward Cone and A Downward Cone

In general, there are far more waste blocks than ore blocks in a pit design model. To set up an upward cone on every ore block is already an enormous task. To set up a downward cone on every waste block to trace an ore block could be even a more formidable job. In addition, a re-allocation path between a surplus ore block and a waste block to be paid off may involve many upward and downward cone searches. Systematic re-allocation and bookkeeping is almost impossible to accomplish without clear successor and predecessor definitions. Therefore, if the re-allocation capability is to be installed in a simulation oriented pit design algorithm, the algorithm loses its simplicity. It may even lose computational advantages over the rigorous graph theory algorithms.

The simulation oriented pit design algorithms need to retain their simplicity and computational efficiency yet they need to be improved so that their generated profits are as close to the optimum solution as possible. Following is a theorem which attempts to provide a general guideline on improving the simulation oriented
pit design algorithms.

Figure 4.5 Re-Allocation via Upward and Downward Cone

Non-optimum theorem:

The re-allocation problem is a major and the only obstacle preventing the simulation oriented algorithms from reaching an optimumal solution. The multi-pass strategy is essentially a re-allocation scheme. However, multi-pass in general can not entirely solve the re-allocation problem. A simulation oriented pit design algorithm should, besides adopting a multi-pass strategy, try to avoid the necessity of re-allocation. If the re-allocation problem is solved or avoided completely, the optimum solution can thus be obtained. Otherwise, the optimum can not be
guaranteed. Because of the great difficulty in providing a rigorous re-allocation scheme and the unlikely event of completely avoiding the re-allocation problem, a simulation oriented pit design algorithm can not be an optimum program.

The point, that a multi-pass strategy is essentially a re-allocation scheme, can be seen in this way. Suppose some poorer ore block is used to offset restricting waste blocks, which are also supported by a richer ore block. At the end of one pass, suppose that the richer ore block is mined out and that the poorer ore block is not. The pit surface is updated and all the unmined blocks are re-initialized to their original values. In this instance, the value of the poorer ore block will now offset only the value of left over restricting waste blocks which had not been supported by the richer ore block that was mined during previous pass. This is exactly equivalent to as if the value from the poorer ore block to the richer ore block has been returned, i.e., re-allocated back to offset the restricting waste blocks of the poorer ore block. This is indeed a re-allocation.

However, there are situations in which the contribution from a poorer ore block to a richer ore block will not be re-allocated back because even the richer ore block is not mined. This does not necessarily mean that there is no path from this poorer ore block to the other richer ore block. Figure 4.6 illustrates this point.

In figure 4.6, eight different regions are marked by capital letters. Three regions are ore blocks A, B and C and five are waste regions D, E, F, G and H. The numerical values in each region represent economic values of the region. Suppose the first pass allocation sequence in figure 4.6 was such that only 8 units of ore block A are allocated to region D and the remaining 6 units of ore block A are allocated to region E. Also, 13 units of ore block B are allocated to region F and G whereas 10 units of ore block C are allocated to region H. Under such an allocation, -1 unit is left in region D, +1 unit is left in ore block B and +2 units are left in ore block C. Since ore block A has waste value overlying and the value of ore block A is exhausted, the block A is non-minable. This will, in turn, result in not mining blocks B and C because one ore block is considered at one time without
shared contribution. (None of the ore blocks A, B and C is mineable if considered individually.) Thus, no ore block can be mined. During the second pass, the same allocation process is repeated as the previous one and since no improvement is made on profit, the allocation process stops with none of the ore blocks mined. However, at the end of the first pass, if one unit is re-allocated from ore block B to ore block A, an optimal solution of 2 units will result. Therefore, when there is no mining at the end of one pass, the multi-pass approach as it is currently being used can not fulfill its re-allocation mission. Yet such a situation can surely occur in a complicated 3-dimensional practical case.

Figure 4.6 An Example of Re-Allocation Problem not Being Completely Overcome by Multi-Pass Strategy

In figure 4.6, if the ore value allocation sequence of Hanson’s Lowest Support First Priority theorem is followed, the optimum solution of 2 units is also obtained (see figure 4.7). Therefore, definition of an “optimal” allocation sequence to avoid the re-allocation situation can improve the quality of solution.
Hence, either re-allocation or defining a sequence to avoid re-allocation produced an optimal solution to this example problem. The former is a necessary condition to optimal solution in network theory. The latter is a direct result from the former, i.e., if there is no need for re-allocation, the algorithm must also be an optimum one.

![Diagram](image)

**Legend**: The numerals in each parenthesis represent the economic value of the corresponding region. Other numerals in each region represent the support value of the corresponding region.

Figure 4.7 An Example of Hanson's Theorem Avoiding the Re-Allocation Problem

Unfortunately, fully guaranteed re-allocation capability is difficult to incorporate in a simulation type pit design algorithm due to the lack of explicit definition of predecessors and successors. Also, it is not always possible to define a perfect sequence that avoids re-allocation situations completely. This concludes the statement of the non-optimum theorem.
4.3 Development of a Simulation Oriented Network Flow Algorithm

Hanson’s Lowest Support First Priority theorem defined an ore value allocation sequence among waste blocks. A more elaborate way is to define an ore value allocation sequence among ore blocks as well. In addition, instead of assigning equal support weight to all ore blocks, one can assign relative support weight to each ore block according to the magnitude of its ore values. For instance, if an average waste block value is \( V(w) \) and the ore block value is \( V(o) \), then the support that one ore block can provide to the waste block is maximum \( [V(o)/ | V(w) |, 1] \). In this way, the difference between individual ore block values is taken into account. A theorem that defines an "optimal" allocation sequence to avoid the re-allocation problem will be discussed in section 4.3.2.

4.3.1 Hindrance Concept for Ore Blocks

The hindrance of an ore block is defined as the number of distinct restricting waste regions that lie above this ore block. For example, in figure 4.8, ore block A has four restricting waste regions, i.e., F, G, H, I. Ore block B has three restricting waste regions, i.e., region G, I, J, and ore block C has also three restricting waste regions, i.e., E, F, G. Therefore, the hindrance of ore block A, B and C is 4, 3, and 3 respectively.

Figure 4.9 is an equivalent network representation of figure 4.8.

From figure 4.8 and 4.9, it can be seen that although supported by the same one ore block, different waste regions correspond to different nodes in network representation. Therefore, the more regions are involved in one cone, the more nodes will be derived from that cone and the more connections will be made from the bottom ore block to the restricting nodes.

The concept of the hindrance of an ore block, deals with defining the flow allocation sequence among ore blocks whereas Hanson’s support deals with defining
Legend: The numerals in the base ore blocks represent the hindrance value of the corresponding ore block.

Figure 4.8 An Example of Ore Block Hindrance

Figure 4.9 An Equivalent Network Representation of Figure 4.8 Problem
flow allocation sequence among waste blocks. The hindrance concept will help us in developing an improved Lowest Support First Priority theorem to avoid the re-allocation situations. This improved theorem is discussed next.

4.3.2 Lowest Hindrance with Highest Value First Priority Theorem

The Lowest Hindrance with Highest Value First Priority theorem is as follows: To avoid the re-allocation situations in a network flow problem, the sequence of allocation of flow from ore blocks to their overlying waste blocks should proceed in such a manner that the value of an ore block with the lowest number of waste hindrance regions is assigned first. In case of a tie situation, the ore block with highest value has highest priority. In allocation of flow from a particular ore block to its overlying waste blocks, the waste with the least ore support value is paid off first. The ore support to a waste block is defined according to its ore value as maximum \( \frac{\text{ore block value}}{\text{average waste block value}} \). The average waste block value is taken as the absolute value of the mining cost of a full waste block.

The proof of this theorem is as follows: To reduce the situation of re-allocation, one must avoid using poorer ore blocks to help richer ore blocks. One way of doing this is to define a “best” flow allocation sequence among both ore blocks and waste blocks. If the value of the ore block with more waste hindrance regions is allocated first, this ore value is more likely to be allocated so that the value allocation of ore blocks with lesser waste hindrance will be affected later. If some of those ore blocks with lesser waste hindrance are richer ore blocks and the restricting regions of these richer ore blocks obtain values from the ore block with more waste hindrance, a re-allocation situation will occur. If the value of ore blocks with more waste hindrance is to be allocated to the most needed region, this value should be allocated only after all the value of ore blocks with lesser waste hindrance has been allocated. Figure 4.10 illustrates this point.

In figure 4.10, if the value of ore block is allocated in a top-down search fashion, the sequence is ore block A, B and C. From the value of block A, 10 units
will be assigned to region H, 5 units to region F and 1 unit to region I. 10 units will be assigned to region J from B. Block C will allocate 10 units to region E, 2 units to region G. There remain -4 units of waste in region I. These -4 units can be paid off by re-allocating the 4 units in region F from block A to region I and using 4 units of block C to offset the -4 units in region F.

Figure 4.10 An Example of Defining Ore Value Allocation Sequence to Avoid Re-Allocation Problem

By adhering to the Lowest Hindrance with Highest Value First Priority theorem, such a sequence for the figure 4.10 example problem is obtained. For example, in figure 4.10, both ore block C and B have waste hindrances of three, whereas A has a waste hindrance of 4. Therefore, ore blocks C and B will be allocated prior
to ore block A. Between C and B, the value of ore block C is larger than the value of ore block B. Thus, the ore block C has higher priority than ore block B. Following an allocation sequence of ore block C, B and A, the ore block value allocation proceeds according to Hanson's Lowest Support First Priority theorem as follows: for ore block C, allocate 17 units to regions E, F, and G; for ore block B, allocate 10 units to region J; for ore block A, allocate 10 units to region H and allocate 5 units to region I. In this way, the three ore blocks are mined without re-allocation.

Assigning a different support weight to each individual ore block according to its economic value is just another way of differentiating in support among waste blocks. This is based on common sense. This ends the proof on the theorem.

4.3.3 Description of A Simulation Oriented Network Flow Algorithm

Based on the two theorems presented in previous sections, a simulation oriented network flow algorithm is developed. The algorithm consists of four parts. The first part determines the profitable cones with only one ore block at the base of the cone. The second part of the algorithm deletes any ore block which can not support its own stripping. The third part of the algorithm defines a flow allocation sequence among ore blocks based on the Lowest Hindrance with Highest Value First Priority theorem. The fourth part of the algorithm carries out pit design via a multi-pass simulation oriented pit design algorithm. For the fourth part, the logic of each pass is similar to that of Hanson's modified network flow algorithm except that the support provided by an ore block is defined as maximum \( \frac{\text{value of ore block}}{\text{average value of waste block}} \) instead of 1.

The simulation oriented network flow algorithm proceeds as below:

1. Set up a scheme which generates a cone such that its apex can be placed on any ore block with top intersecting the topography of mineralization model.

2. Locate the first ore block in a top-down search fashion, place an inverted cone on this ore block, compare the positive value of this first ore block with the cumulative value of all the overlying restricting waste blocks of this ore block. If the
former is larger, the cone of this first ore block is individually profitable. The cone is thus mined out. The surface is updated. Otherwise, every block in this first cone is untouched. After the first ore block is examined, the inverted cone is moved to the second ore block. The positive value of this second ore block is compared with the cumulative value of all the overlying restricting waste blocks of the second ore block. Again, if the former is larger, the cone of this second ore block is profitable. This second cone is thus mined out. The surface is updated. Otherwise, every block in this second cone is left untouched. The cone is then moved to a third ore block. The process is repeated until the last ore block is examined. This is the end of the first iteration.

3. Since the mining of some cones may have created some other profitable cones, a second iteration proceeds the same way as the first iteration. After the end of the second iteration, the third iteration is started. The iteration continues until at any given iteration there is no improvement on the pit design profit compared with the previous iteration. This ends the first part of the algorithm. The first part of the algorithm is most effective in dealing with a massive deposit. In dealing with massive deposit, many ore blocks may have been mined before entering the second part of the algorithm. This may drastically reduce the number of blocks in subsequent parts of the algorithm.

4. Generate a support matrix within the remaining blocks in the model such that given any block, the number of ore blocks that support this block is known. This number is called simple support of the block. If the block of concern is a waste block, its support value is negative. If the block of concern is an ore block, its support value is positive. An ore block is a support to itself therefore, the number of simple supports of an ore block counts itself. This simple support matrix is exactly the same as the one in Hanson’s algorithm.

5. Locate the first ore block in a bottom-up search fashion. If this first ore block does not have a simple support value of 1, go to the second ore block in a bottom-up search. Otherwise, place an inverted cone on this ore block, compare the
positive value of this first ore block with the cumulative negative value of its least supported restricting waste blocks (these waste blocks have a simple support value of -1). If the former is larger, go to the second ore block in the bottom-up search without changing anything. Otherwise, permanently delete this first ore block from further consideration. Update the simple support value of all the restricting blocks of the deleted first ore block. This is done by adding +1 to the simple support value of every restricting waste block and -1 to the simple support value of every restricting ore block. After the first ore block is examined, the inverted cone is moved to a second ore block in the bottom-up search. The positive value of this second ore block is compared with the cumulative value of the least supported restricting waste blocks. Again, if the former is larger, go to the third ore block in the bottom-up search without changing anything. Otherwise, permanently delete the second ore block from further consideration. Update the simple support value of all the restricting blocks of the deleted second ore block. The update is done exactly in the same manner as for the first ore block, i.e., adding +1 to the simple support value of every restricting waste block and -1 to the simple support value of every restricting ore block. The cone is then moved to a third ore block and the process is repeated until the last ore block is examined. This is the end of first iteration.

6. A second iteration is started if any ore block was deleted in the first iteration. This second iteration proceeds exactly like the first iteration. The iteration is needed because the deletion of some ore blocks in the previous iteration may allow the deletion of some other ore blocks that do not support their least simple support. If any ore block is deleted in the second iteration, a third iteration begins. Iteration continues until at any iteration no ore block is deleted. This ends the second part of the algorithm. The second part of the algorithm is most effective dealing with sparse mineralized deposits. Many non-profitable ore blocks may have been excluded before entering part three of the algorithm.

7. Generate a support matrix S such that the support values of any waste
blocks supported by the same combinations of ore blocks are the same but the support values of those waste blocks supported by distinct combinations of ore blocks are different. This is done in next two sub-steps.

7.1 Assign a support value to each ore block in the model. This support value equals the sequence number of the ore block in a top-down ore block search in the mineralization model. For example, if an ore block is the fifth ore block found in a top-down search fashion, the support value of this ore block is assigned as 5.

7.2 For each ore block in the model, do the following. Place an inverted cone on the ore block. For every waste block falling in the inverted cone, calculate its support value $S(ID)$ as follows:

$$S(ID) = S(ID) + IOBLK$$

(4.1)

where in equation (4.1), $S(ID)$ is the support value of waste block located at position ID. IOBLK is the sequence number of the ore block in a top-down search. $S(ID)$ is initially set to zero. The ID is calculated according to equation (4.2):

$$(IZ - 1) \times NX \times NY + (IY - 1) \times NX + IX$$

(4.2)

where in equation (4.2), NX is the number of blocks in the X direction in the mineralization model; NY is the number of blocks in the Y direction in the mineralization model. IX, IY, IZ are the coordinates of the waste block in the mineralization model. Finishing the calculation for the last ore block finishes step 7.

8. Go back to the first ore block; for each ore block in the model do the following: Place an inverted cone on the ore block; sort the $S(ID)$'s of all the waste blocks falling in the inverted cone according to the magnitude of $S(ID)$. It is irrelevant whether the sorting is done according to the smallest value first or according to the largest value first. Then, among the sorted $S(ID)$'s, count the number of times that $S(ID)$'s change their values. This number plus 1 is the number of distinct hindrance regions of the ore block.
9. Sort the ore block processing sequence so that the ore block with the lesser number of waste hindrance regions has the higher priority to be picked up.

10. Sort the ore block processing sequence so that the ore blocks with the same number of waste hindrance regions are arranged in a sequence such that the ore block with the higher value has a greater priority to be picked up. This ends the third part of the algorithm. Before going into the fourth part of the algorithm, the matrix $S$ is re-set to all zeros.

11. Generate a support matrix $S$ such that given any block in the model, the value of support of that block is known. This value is defined as the support of the block. The matrix $S$ is calculated as below:

For each ore block in the model, do the following. Place an inverted cone on the ore block. Calculate the support that this ore block can provide as:

$$NSPT = \max \left[ \frac{\text{value of the ore block}}{\text{average value of the waste block}}, 1 \right]$$

then, for each restricting block of the ore block, calculate the support of the restricting block as:

$$S(ID) = |S(ID)| + NSPT$$

for restricting ore blocks,

$$S(ID) = S(ID) - NSPT$$

for restricting waste blocks,

where in equation (4.4) and (4.5), the $S(ID)$ and $ID$ are defined as in equation (4.1).

12. Locate the first ore block in a top-down search fashion. Place an inverted cone on this ore block. Allocate the positive value of this ore block up to its overlying waste blocks to offset their negative values in such a way that the blocks with least support in the cone are allocated first, then the blocks with second least support,
... as long as there is a positive value left. The allocation process for this ore block stops when either of the following occurs: a) there is no positive value left to allocate, b) all waste values overlying this ore block are offset to zeros. The values of the allocated blocks are updated to prevent double counting. After the value of the first ore block is allocated, the inverted cone is moved to the second ore block. The positive value of this ore block is allocated up to its overlying waste blocks to offset their negative values, exactly in the same manner as for the first ore block. That is, the blocks with least support in the cone are allocated first, then the blocks with second least support, etc. The allocation stops when either there is no positive value left or all the waste values are offset. Again, the values of the allocated blocks are updated to prevent double counting. The cone is then moved to the third ore block and the process is repeated until the value of the last ore block is allocated. This is the end of the first iteration.

13. Check whether any negative values are still left among the restricting blocks overlying any ore block. If there are no negative values, stop. The optimum solution has been obtained. Otherwise, any ore block with negative blocks overlying is deleted. This is done by putting this ore block into a temporary non-minable group as if this ore block does not exist.

14. The second iteration is started after restoring the profit values of each block in the mineralization model to their original values. This second iteration proceeds exactly like the first iteration. This iteration is needed since the deleted ore blocks may have helped to offset waste values for those ore blocks remaining in the possible minable set. In other words, some non-profitable ore blocks may remain in the possible solution set because of the contribution from the deleted ore blocks. This iteration thus guarantees that no over mining will take place during pit limit design.

At the end of the second iteration, the negative value over any ore block is again checked. If there is no negative value over any ore block, then the “optimum” solution has been obtained. Otherwise, the ore block is deleted and the value
matrix is resummed to its original value. Then, a third iteration is started and iterations continue until there is no negative value over any ore block. At this stage, a new permanent pit surface is obtained with all the remaining ore blocks and their restricting blocks being mined out. This is the end of the first pass.

15. A second pass is started with all those ore blocks in the temporary non-minable set being brought back and a value matrix resummed to its original value. This second pass repeats steps 12 through 14, i.e., the first pass. At the end of the second pass, the surface is again updated. After bringing back the deleted ore blocks and restoring the value matrix to its original values, a third pass is started. This multi-pass process repeats until at any given pass there is no improvement on the pit design profit compared with the previous pass. At this stage, the final permanent surface obtained is the ultimate pit limit. The cumulative value of blocks falling between the initial pit surface and the pit limit is the profit from pit design.

Remarks: The four parts of the algorithm can be coded into four separate routines. The purpose of the first two parts of the algorithm is to reduce the number of ore blocks in the model so that the computing in the last two parts can be speeded up. Although the algorithm can proceed in the above listed order, all four parts of the algorithm can be applied in mixed orders to achieve better results once the first pass is finished. For instance, the third part and the fourth part can be interchanged. When a permanent surface is obtained, i.e., some ore blocks are mined, the waste hindrance regions above an ore block may have changed. Therefore, to obtain a better solution, part three may have to be executed before part four is run for the second pass.

Due to the nature of each part of the algorithm, the program structures of the four parts are very similar. What is more important is that the new algorithm is a repeated application of the moving cone strategy. The algorithm thus retains the simplicity of the moving cone method.

In addition, since the new algorithm is developed in the context of network flow theory, the algorithm provides a theoretically improved approach over the
moving cone algorithm in terms of generated pit design profit.

The theoretical advantages of the new algorithm over the traditional moving cone algorithm are:

1) Because the algorithm considers one sub-cone at a time instead of one whole cone at a time, the new algorithm is capable of detecting many shared contributions. The sub-cone in the new algorithm is characterized by the blocks having the same support values. In addition, the new algorithm does not over mine non-profitable ore blocks because no ore block at an upper position is allowed to help the ore block at the lower position.

2) The new algorithm is developed to avoid as many re-allocation situations as possible. According to the non-optimum theorem developed in section 4.2, if the re-allocation situation is avoided completely, the optimum solution would be obtained.

Unfortunately, the non-optimum theorem also implies that the new algorithm may not produce a rigorous optimum solution, because there is no guarantee that the re-allocation situations can be completely avoided. How close the result from the new algorithm is to the result from rigorous optimum routines may depend on the nature of mineralization. Nevertheless, the new algorithm should generate more profit than the traditional multi-pass moving cone algorithms on the same basis.

In section 4.4, it will be seen that the new algorithm does generate greater profits than both the multi-pass moving cone method and Hanson's algorithm, well within the acceptable computing time.

Appendix A contains a numerical example of the algorithm.
4.4 Performance of the New Algorithm

In this section, the new algorithm (simulation oriented network flow algorithm (SNF)) will be compared with four other algorithms. These four algorithms are: 1) Lerchs-Grossmann (LG) 3-dimensional graph algorithm; 2) Multi-pass Moving cone algorithm (MC); and 3) Hanson’s algorithm (HA); 4) A multi-pass simulation oriented network flow algorithm without part three of SNF and the waste support defined as \( \text{max} \left( \frac{\text{ore value}}{\text{average waste value}}, 1 \right) \) (MNF). The LG algorithm is taken here as a reference of profit because the algorithm is a known optimum routine. The MC algorithm is included here because of its popularity. Comparing the new algorithm with the MC method in terms of generated profit and execution time could reveal where the new algorithm stands. A comparison between the new algorithm and the Hanson’s algorithm will show how much the Hanson’s algorithm has been improved. Finally, a simple multi-pass simulation oriented network flow algorithm, without part three and the waste support defined as \( \text{max} \left( \frac{\text{ore value}}{\text{average waste value}}, 1 \right) \), will reveal the impact of part three of the new algorithm and of the new waste hindrance definition. Note that part three is only executed once before part four in executing the SNF of this study.

4.4.1 Selected Comparison Criteria

In this study, the selected criteria for comparison of 4 different algorithms are; 1) total profit value of each design and 2) computation time involved. The comparison is based on one actual deposit whose model information is summarized in table 4.1.
Table 4.1 Summary of Model Information

<table>
<thead>
<tr>
<th>Deposit</th>
<th>Massive copper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Type</td>
<td>Fixed 3-D block model</td>
</tr>
<tr>
<td>Model Dimension</td>
<td>40 * 40 * 24 (NY * NX * NZ)</td>
</tr>
<tr>
<td>Block Dimension</td>
<td>100 * 100 * 40 cu-ft (DY * DX * DZ)</td>
</tr>
<tr>
<td>Economics</td>
<td>Current economical condition</td>
</tr>
<tr>
<td>Pit Slope</td>
<td>45 degrees in all directions</td>
</tr>
<tr>
<td>Profit Unit</td>
<td>U.S. dollars</td>
</tr>
</tbody>
</table>

The comparison in this study is based on current economic data. The performance of the algorithms in terms of their sensitivity to different economic parameters is not made in this study. One should be aware that the results given in this section can and will change if different economic parameters are assumed in the model.

Another important aspect which has been ignored in the comparison is the practicality of the generated pit limit design. For example, the effect of deleting small, non-minable pit bottoms or of specifying the minimum size pit bottom is not considered during comparison. Similarly, the effect of inserting haul roads to the generated pit limit is not addressed.

The results do pertain primarily to the 3-dimensional ore body model used and not necessarily to the real deposit that the model represents. Nonetheless, the comparison does provide information as to which algorithm generates greater profits within a reasonable execution time.

4.4.2 Comparison Results

The generated pit design profit and the corresponding computer execution time for the four algorithms are summarized in table 4.2.
Table 4.2 Comparison Results Summary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Generated Profit</th>
<th>Execution Time</th>
<th>Computer Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>LG</td>
<td>555,792,100</td>
<td>189.73 minutes</td>
<td>PC AT (SI:7.0)</td>
</tr>
<tr>
<td>MC</td>
<td>550,211,072</td>
<td>27.97 seconds</td>
<td>VAX 8600</td>
</tr>
<tr>
<td>HA</td>
<td>552,692,224</td>
<td>74.69 seconds</td>
<td>VAX 8600</td>
</tr>
<tr>
<td>MNF</td>
<td>555,293,696</td>
<td>52.61 seconds</td>
<td>VAX 8600</td>
</tr>
<tr>
<td>SNF</td>
<td>555,708,928</td>
<td>58.26 seconds</td>
<td>VAX 8600</td>
</tr>
</tbody>
</table>

From table 4.2, it can be seen that the LG algorithm generated the highest profit and the lowest profit is resulted from the MC method. The SNF algorithm generated the second highest profit which is very close to the optimal solution (within 99.98% of optimum solution). The SNF algorithm generated $5,497,856 (1.00%) more profit than the MC algorithm, $3,016,704 (0.55%) more profit than the Hanson’s modified network flow algorithm, $415,232 (0.07%) more profit than the simple multi-pass simulation oriented network flow and $83,172 (0.015%) less profit than the optimum solution $555,792,100.

With respect to computing time, as expected, the MC algorithm is the fastest. The execution time of the SNF algorithm is approximately 2.08 times slower than the MC algorithm but the execution time is well within the practical acceptable range (in this case 58.26 cpu seconds). The execution time of the MNF is close to that of SNF (52.61 cpu seconds). The execution time of Hanson’s is the longest (74.69 cpu seconds). The reason that the new algorithm takes less time to execute than the Hanson’s algorithm is because part 1 and part 2 have reduced the number of blocks in part 3 and part 4 of the algorithm which, in turn, reduced the allocation time in a single iteration and the overall execution time. Also, the ore block processing sequence developed in part 3 has reduced the number of reallocation situations and the number of iterations in part 4 of the algorithm. This eventually resulted in a reduced execution time.

In summary, based on the case study in this section, the new algorithm (SNF) can generate significantly higher profits than the traditional multi-pass moving cone
method and Hanson's modified network flow algorithm. The profit difference between the SNF and the MNF is not really significant. The profit difference between the SNF and the rigorous LG algorithm is virtually negligible. The execution time of the new algorithm is about 2.08 times slower than the multi-pass moving cone method and well within the practical acceptable range.

4.5 Summary Remarks

Based on network flow theory, this chapter examined how far the simulation oriented pit design algorithms can be improved in optimizing pit design profit. Two theorems introduced in this chapter provided general guidelines for the improvement of the simulation oriented pit design algorithms. The first theorem states that reallocation is a major obstacle to achieve optimum solution by simulation oriented pit design algorithms without losing their simplicity and computing efficiency, and the graph pit design algorithms are virtually the only way to guarantee an optimum solution. The second theorem suggests that one way to avoid the re-allocation problem is to define an "optimum" ore value allocation sequence.

Following the principles of the Lowest Hindrance with Highest Value First Priority theorem, a simulation oriented network flow algorithm is introduced. The new algorithm clearly shows an improvement over both the popular multi-pass moving cone method and the Hanson's modified network flow algorithm in terms of generated pit design profit. The profit difference between the new algorithm and the rigorous LG algorithm is virtually neglectable according to the case study in previous section. What is more important is that the execution time of the new algorithm is well within the practically acceptable time range. The case study of the previous section shows that the new algorithm is only 2.08 times slower than the extremely fast moving cone method.

A fast sorting routine is the key to the success of part three of the new algorithm and thus the success of the new algorithm. Part three of the new algorithm
can reduce the number of passes needed in subsequent runs because of the definition of a better value allocation sequence among ore blocks. In addition, even without part three, the new algorithm can still consistently provide better results than the moving cone method. The reason is that the moving cone method fails to detect the joint contributions and can over mine non-profitable blocks, whereas the new algorithm will not over mine non-profitable blocks. It can also detect more regions of joint contribution even without the part three enhancement.

The new algorithm does not over mine non-profitable blocks and can be used both as a pit design algorithm itself and as a pre-run for the rigorous graph algorithms, depending on the objective of the user.
CHAPTER 5

DEVELOPMENT OF A ZERO-ONE PROGRAMMING MODEL FOR
MINE SEQUENCING PROBLEMS

5.1 The Mine Sequencing Problem of this Study

The mine sequencing problem of this study starts with a series of pushbacks in which bench by bench inventories are available on ore tonnage, waste tonnage and average grade values for a series of cutoff grades for each incremental pushback.

As discussed in section 1.1, one way of generating mine sequencing pushbacks is to generate a series of nested pits by varying product prices corresponding to a constant cost. If the distance between any two of these series of nested pits exceeds the minimum allowable mining width, these series of nested pits represent "next best" mineralization with the inner pit being the "best" and the outer pit being the "next best". Therefore, the series of nested pits can serve as pushbacks for mine sequencing with the boundary of the outer most pit serving as ultimate pit limit. The pushbacks thus generated have an important feature, i.e., the inner most pushback has the best mineralization and the mineralization declines from inner pushbacks to outer pushbacks, with the outer most pushback having the lowest mineralization. This inner pushback to outer pushback sequence provides a general direction for pushback mine sequencing so that the NPV of the mining sequence is maximized.

As also discussed in section 1.1, the optimization of a mining schedule requires generating alternative ore and waste removal schedules based on variable cutoff grade strategies. Corresponding to each cutoff grade strategy, a mining schedule can be developed and a net present value (NPV) analysis can be carried out. A
single or a combined schedule which has the highest NPV is selected as an optimum mining schedule.

In a multi-period mine sequencing problem, once a cutoff grade strategy is chosen for a period, the available average grade above the selected cutoff grade as well as the ore and the waste tonnages are fixed for every bench in every pushback. The NPV of a mining schedule depends, in every period, only on how many tons of ore are to be mined, how many tons of waste are to be stripped and what is the average grade of the ore mined subjected to some practical constraints such as ore tonnage requirement for constant mill feed, etc (assuming constant mining and milling costs).

Although the NPV of a project is maximized if mining proceeds following an inner pushback to outer pushback sequence, mining may have to balance among distinct pushbacks to satisfy practical constraints. In other words, the optimization of a mining schedule is a constrained optimization problem.

It seems reasonable to state that for constrained optimization, the NPV of a mining schedule based on one set of fixed cutoff grade strategies is maximized as long as one follows the inner pushbacks to outer pushbacks as closely as possible while seeking to satisfy practical constraints. As a result, if the mining sequence of inner pushback to outer pushback is formulated as precedence constraint, the optimum mine sequencing problem becomes the search for the feasible solution that not only follows the precedence constraints and meets the practical constraints, i.e., the ore tonnage, the waste tonnage, stripping ratio and balancing grade constraints, but also has the highest possible mineral contents (highest average grade) in early periods.

In balancing among distinct pushbacks to satisfy practical constraints, most current approaches are entirely by trial-and-error. As such, the current approach may find one feasible solution for one period, but it can not guarantee that this feasible solution satisfies the multi-period long term mine sequencing objective, such as to pursue the highest possible mineral contents.
Using the sub-optimization criterion, this chapter develops a zero-one programming mine sequencing model that can quickly generate multi-period mining schedules which meet specified long term objectives.

In this study, 0-1 programming is chosen for mine sequencing for the following four reasons:

1) 0-1 programming is capable of answering what is the best (maximum or minimum) that can be expected on a particular item subjected to a given set of constraints.

2) 0-1 programming is capable of systematically detecting whether or not a feasible solution exists corresponding to a given set of constraints.

3) 0-1 programming is capable of recording all the feasible solutions that have been examined in the path toward fathoming the optimum solution. The objective function values of these feasible solutions are improved from the early solutions to the later solutions in terms of optimization of the objective function. As such, one can stop after any feasible solution has been found, provided that one is satisfied with the attained objective value under the current iteration.

4) 0-1 programming is capable of providing information on which constraint should be relaxed in case of difficulty in finding a feasible solution.

In this study, the 0-1 programming is applied period by period to the multi-period mine sequencing problem. In other words, one application of 0-1 programming is needed for each single period mine sequencing problem. Each single period objective satisfies (or conforms) to the long term multi-period mine sequencing objective. For example, if the long term objective is to maximize the mineral contents – an objective which should be pursued whenever it is possible (under the premise of the same cost, maximization of mineral contents in early periods produces higher NPV), the objective in each single period should be defined as to maximize the mineral contents so that the mineral contents in every period is maximized. Thus, a long term multi-period mine sequencing objective is achieved by defining a series of sub-optimization objectives. A multi-period mining sequence development is
carried out period by period.

This and the next chapter investigate how to apply 0-1 programming as a fast and effective mine sequencing tool such as:

1) how to limit the number of decision variables in the 0-1 program formulation to obtain a feasible solution quickly;

2) how to find an efficient precedence constraint or sequential requirement (SR) formulation so that the vast number of SR constraints can be reduced enormously and the solution time can thus be reduced;

3) what is to be experienced in applying 0-1 programming to the multi-period pushback mine sequencing problem and how to overcome some of the problems associated with applying 0-1 programming.

The discussion in this and in the next chapter are based on a single mining schedule development where the cutoff grade is already chosen for every scheduling period. To better understand the model to be developed in this chapter, the next section discusses the 0-1 programming problem. An alternative SR formulation developed by this author (Cai, 1985) is also presented.

5.2 Zero-One Programming and Implicit Enumeration

5.2.1 Zero-One Programming Problem

Decision problems where only two choices are available, yes or no, are frequently encountered in the practical world. For instance, should this block be mined? Should this bench be mined? What are the effects of these decisions on the objective function value?

With just two choices, these kinds of decisions can be represented by decision variables that are restricted to just two values, i.e., zero or one. These types of problems are often solved by zero-one programming.

Mathematically, the standard zero-one programming problem can be stated in the following form:
\[ \text{Minimize} \quad \sum_{j=1}^{n} c_j x_j \]
\[ \text{Subject to :} \quad \sum_{j=1}^{n} a_{ij} x_j \geq p_i \quad (5.1) \]

Where \( n \) is the number of decision variables, \( m \) is the number of constraints, \( c_j \)'s are the coefficients in the objective function, \( x_j \)'s are the decision variables of the problem, and \( p_i \)'s are the right hand side elements. The \( a_{ij} \)'s are the coefficients of all linear inequalities. A feasible solution is a vector \( X = (x_1, x_2, \ldots, x_n)^T \) which satisfies (5.1); here \( x_1, x_2, \ldots, x_n \) will be either zero or one. An optimum solution is the feasible solution which minimizes the objective function.

### 5.2.2 The Implicit Enumeration Algorithm

Among the solution algorithms for zero-one programming problem, the most popular one is Balas' additive implicit enumeration. Balas' algorithm is chosen in this study for its popularity and simplicity. The principle employed in implicit enumeration is the following. For a \( n \)-variable 0-1 programming problem, there will be \( 2^n \) possible solutions, most of which may be infeasible. By employing a certain strategy for selecting a few solutions to be enumerated explicitly, most of the \( 2^n \) solutions could be enumerated implicitly. The computations can thus be reduced enormously.

The implicit enumeration method is essentially a tree search algorithm that uses information generated in the search to exclude portions of the tree from further considerations. The solution tree is constructed as follows. Every possible solution
is a node in the solution tree. Starting with node 0, meaning no variable has been raised to 1 yet, the solution tree expands itself by raising one variable at a time to 1 until all the variables are raised to 1. If the solutions with the same number of variables being raised to 1 are grouped as one level, there are a total of $n+1$ levels in the solution tree. The node 0 is in level 0 and the solution with all variables being raised to 1 is in level $n$. Each node in a lower level (having higher level value) is a descendant of some node in a higher level (having lower level value). An optimal solution, if the solution exists, is obtained by following some branches of the tree rather than the whole tree.

The implicit enumeration method proceeds as follows:

Starting with node 0, the implicit enumeration method checks at the origin nodes to see if the descendant nodes can be reached, i.e., if the completions of these partial solutions are feasible. Sets of descending solutions are ignored, as the enumeration proceeds, when,

1) “Completions” of partial solutions are found to be infeasible, or
2) “Completions” of partial solutions are found to be less attractive due to a feasible solution found earlier.

This is accomplished by backtracking with some provisions to avoid duplicating the previously examined branches and provisions to exclude all the descendant branches from further consideration.

When the enumeration process can not go any further, the solution enumeration process is finished and all the $2^n$ solutions have been enumerated either explicitly or implicitly. The last feasible solution enumerated is the optimum solution for (5.1), since each successive feasible solution sets a new lower bound for the objective function value during the solution process.

5.2.3 Remarks on Balas' Implicit Enumeration Algorithm

The following are the theoretical foundations of Balas's additive algorithm:

1) The problem (5.1) is first rearranged into (5.2) as follows:

\[
\begin{align*}
\text{Minimize} & \quad \sum_{j=1}^{n} c_j x_j \\
\text{Subjected to:} & \quad -p_i + \sum_{j=1}^{n} a_{ij} x_j \geq 0 \\
& \quad c_j > 0 \\
& \quad x_j = 0, 1 \\
& \quad p_i \geq 0 \\
& \quad 1 \leq j \leq n \\
& \quad 1 \leq i \leq m
\end{align*}
\]

(5.2)

Where every item is defined as in (5.1).

2) An objective function is to be minimized while subjected to the constraints that: a) algebraic sums of all left side items in each constraint must be non-negative, b) all decision variables can only achieve either 0 or 1.

3) The lesser the decision variables involved in a solution are raised to 1, the better the objective value will be because of the form of objective function used. Hence, the algorithm starts the enumeration process with all \(x_j\)’s set to 0 and raises to 1 only those decision variables which are absolutely necessary to meet the constraint requirements.

4) Only those variables having positive coefficients in a constraint could be helpful to make the algebraic sum of all left hand items of that constraint become non-negative. Thus, the coefficient sum of one variable (\(a_{ij}\)’s) may reflect the “helpfulness” to bring about feasibility. This is one way to determine which variable will be introduced to a partial solution.

5) The last enumerated feasible solution sets up a new lower bound for objective function coefficients. This guarantees that the objective value is always being
improved and that the last feasible solution fathomed is optimal.

6) For a $n$-variable 0-1 programming problem, there are a total of $2^n$ possible solutions. No matter how large the number of possible solutions, the number is finite. Therefore, the solution enumeration process of Balas’ additive algorithm would come to an end after a finite number of iterations. Since most of the solutions will be enumerated implicitly and only a small portion of the solutions will be enumerated explicitly, the solution enumeration process would terminate after a reasonable number of iterations. It should be noted here, however, that this “reasonable number” still could be too large to be cost-effective if there are too many variables involved.

7) The solution enumeration process is a process of examination, meaning that when the process is finished, all the solutions (not necessarily feasible) have been examined either explicitly or implicitly. This, naturally, does not guarantee a feasible solution for certain. When Balas’ additive algorithm is applied to solve a zero-one programming problem and no feasible solution is found at the end of the enumeration process, then there is no feasible solution for the problem itself!

5.2.4 Precedence Constraint Formulation

In pushback mine sequencing, there may be a large number of precedence constraints or sequential requirements (SR) among the benches from the same pushbacks and among the benches from different pushbacks that are on the same level. Usually, a pushback will have 15 benches on the average, and often more than 4 pushbacks are generated in a single pit. Therefore, there may well be $4 \times (15-1) + 15 \times (4-1)$ or 101 SR to be considered for a typical open-pit mine.

In 0-1 programming, the SR, or the precedence constraints, are customarily formulated in the following manner (Plane and McMillan, Jr, 1971):

If $x_i$ must precede $x_j$, then $x_i \geq x_j$.

For example, if $x_1$ must precede $x_4$, then $x_1 \geq x_4$. One SR basically needs one inequality constraint.
For 101 SR, 101 inequalities would be needed. The more SR involved, the more SR constraints will be needed. The following five aspects have stimulated a search for a more efficient SR formulation with an attempt to reduce the number of SR constraints and the computer execution time.

1) During a solution process, a solution must be examined by every problem constraint at each iteration.

2) The time spent on a constraint computation has little to do with the number of variables in that constraint, i.e., the time spent on the SR constraints is almost the same as the time spent on the basic problem constraint.

3) The more constraints are involved in a problem formulation, the more time is needed to examine them at each iteration.

4) The more iterations are involved to solve a problem, the more time will be spent on the SR constraints.

5) The computer execution time for the 0-1 programming problem could be too long to be cost-effective for mine sequencing problem when the number of decision variables is more than 40-50 (based on this author's experience). To provide a 0-1 programming solution for a single period mine sequencing problem within 1-10 minutes on a VAX 750 is what the 0-1 programming pushback mine sequencing model in this thesis was intended to achieve.

It is possible to reduce the number of SR constraints and the computer execution time enormously when the SR among the 0-1 decision variables possess a special feature (Cai, 1985). This special feature is that the SR can be grouped into several SR sets, and as a result, the number of SR sets is much smaller than the total number of SR. To do so, each SR set should consist of successive decision variables linked together like a chain. For example, \( x_1 \geq x_2 \geq x_3 \) is one SR set of which the variables can be linked. Similarly, \( x_5 \geq x_6 \geq x_7 \) can form another set (chain). But, these two chains cannot be linked together as one chain because \( x_4 \) is missing. Therefore, they must be viewed as two SR sets.

The SR involved in a pushback mine sequencing problem can have this special
feature, i.e., the SR among benches from the same pushback form one chain, if the
decision variable is defined as a bench. Also, the SR among benches from distinct
pushbacks but on the same level can form one chain.

Cai (1985) developed an alternative SR 0-1 programming formulation. The
formulation was targeted at the above SR situations. The SR formulation is given
as follows: For \( k \) -th SR chain

\[
\sum_{j=1}^{q} a_{kj}x_j \geq \sum_{j=1}^{q} a_{kj} \\
1 \leq q \leq n \\
1 \leq k \leq s
\]  

Where in (5.3),

\( n \) = total number of decision variables of 0-1 problem;
\( q \) = the maximum subscript value of decision variables in a solution
under examination; Note that this \( q \) varies according to
the maximum subscript value of the solution under examination
during implicit enumeration;
\( k \) = subscript for a given SR chain;
\( s \) = total number of SR chains;
\( a_{kj} \) = any convenient constant value assigned by user
at problem formulation stage to the \( k \) -th set of SR chain;
\( a_{kj} > 0 \) for \( x_j \)'s subjected to the \( k \) -th set of SR;
\( a_{kj} = 0 \) otherwise.

Basically, the SR among all the benches from one pushback forms one SR
chain, the SR among all the benches from distinct pushbacks but on the same level
also forms one chain.

As a result of this SR formulation, the original \( 4 \times (15-1) + 15 \times (4-1) = 101 \)
SR constraints (4 pushbacks and 15 benches in each pushback) can be reduced to
\( 4 + 15 = 19 \) SR, i.e., 19 constraints – an 81.2% reduction in number of constraints.
The inequality (5.3) assumes that the variables with smaller subscripts have higher priorities to be raised to 1 than the variables with larger subscripts. Therefore, the variables with higher priorities to be raised to 1 should be assigned smaller subscripts and the variables with lower priorities to be raised to 1 should be assigned larger subscripts. The inequality (5.3) works this way: if somehow a variable of subscript q is raised to one, the right hand side has a value of $\sum_{j=1}^{q} a_{kj}$. To hold the inequality (5.3), the left hand side must also have a value of $\sum_{j=1}^{q} a_{kj}$, this is possible only if all the $x_j$'s with subscript values being smaller or equal to q are raised to 1. Since the $a_{kj}$'s corresponding to the variables not subjected to the k-th chain of SR are assigned value of zeros, to raise the value of all $x_j$'s with subscript values being smaller or equal to q to 1 ensures that the SR is followed for k-th chain.

Combining (5.2) and (5.3), a revised zero-one programming formulation is given as follows:

\[
\begin{align*}
\text{Minimize} & \quad \sum_{j=1}^{n} c_{j}x_{j} \\
\text{Subjected to:} & \quad -p_{i} + \sum_{j=1}^{n} a_{ij}x_{j} \geq 0 \\
& \quad \sum_{j=1}^{q} a_{kj}x_{j} \geq \sum_{j=1}^{q} a_{kj} \\
& \quad c_{j} > 0 \\
& \quad x_{j} = 0, 1 \\
& \quad p_{i} \geq 0 \\
& \quad 1 \leq j \leq n \\
& \quad 1 \leq i \leq m \\
& \quad 1 \leq q \leq n \\
& \quad 1 \leq k \leq s
\end{align*}
\]  

(5.4)

where, in (5.4), all items are defined the same as those defined in (5.1) and (5.3).
5.3 Single Period Zero-One Programming Mine Sequencing Problem Formulation

Single period 0-1 programming mine sequencing problem formulation involves the following: 1) definition of decision variables and mine sequencing assumptions; 2) limiting the number of decision variables in the problem formulation; 3) definition of objective functions; 4) selection of problem constraints; and 5) precedence constraints formulation.

5.3.1 Definition of Decision Variables and Mine Sequencing Assumptions

The Diamond mine sequencing problem requires for bench by bench sequencing. One bench from each pushback is therefore, defined as one decision variable. If the \( j \)th bench of \( i \)th pushback is denoted by \( x_{ij} \), then

\[
x_{ij} = \begin{cases} 
1 & \text{if the bench is scheduled to be mined} \\
0 & \text{otherwise}
\end{cases}
\]

The assumptions for the Diamond mine sequencing problem are basically the same as those listed in section 2.3.1:

1) Projected grade information for each decision variable is accurate.
2) Final pit slope angle projections are reliable.
3) Current forecasting of product prices and costs are applicable during all sequencing periods for the net present value analysis.
4) Each pushback is itself a minable area, containing all haul roads, conveyor routes, crusher status, and other geometric parameters necessary for field implementation. The pushback pattern can faithfully follow the pit slope angle requirement.
5) At the end of a single period 0-1 programming mine sequencing, the mined benches at bottom may have to be adjusted so that a bench is partially mined to meet the client’s tonnage schedule. In case a bench is partially mined, the ore and
waste tons are based on the percentage of mining multiplied by the average available ore and waste tonnages on that bench.

6) The access road to a sequence is inserted after a sequence has been developed.

The first 3 assumptions present the credibility to the mine sequencing problem. Assumption 4 indicates that as long as the benches on the same level are mined in the order of inner pushback to outer pushback, the pit slope constraints will not be violated. Since each bench of each pushback has enough room for its own mining, the decision to mine or not to mine can be made according to a zero-one programming decision.

Assumption 6 implies that one inserts roads after the zero-one programming decision. Nowadays, the interactive graphic planner (IGP) is becoming more and more popular and is available at least from professional software developers. For example, the IGP in MEDSYSTEM™ developed by Mintec Inc., in Tucson, Arizona can summarize average grade, ore and waste tonnages if the user uses a mouse or keyboard to define an area on screen and this planning area can be zoomed large and small. At the end of the mine sequencing, one can plot either on screen or on a plotter to see where, and which bench in which pushback is mined during each planning period. In case a bench is partially mined, other questions such as: 1) what percentage of that bench has been mined, 2) where should this partial mining occur, and 3) how and where the access road should be developed for each period, can best be answered by an engineer with the visual help of IGP.

5.3.2 Limiting the Number of Decision Variables

The number of decision variables should be a minimum. As discussed earlier, implicit enumeration is a solution enumeration method. The addition of one variable doubles the number of possible solutions. When the number of decision variables in a problem formulation reaches a certain range, the addition of one more variable may increase the solution time unacceptably. In general, the number of decision
variables in a 0-1 problem formulation should be limited to 40-50 if Balas' algorithm is to be applied (Cai, 1985).

There are at least three ways to reduce the number of decision variables in a mine sequencing problem: 1) to limit the number of decision variables using the targeted tonnage requirement for each period; 2) to limit the number of decision variables by SR; 3) to limit the number of decision variables by combining benches into bigger composite benches. These three possibilities are discussed next.

For the first approach, there is no reason to include more than the targeted ore tons into the problem formulation for a particular mining expansion, because no more than the targeted ore tons will be scheduled for mining. (This first approach will later be referred to as tonnage limiting rule.) Therefore, in any one pushback, the benches below the bench at which the top-down cumulative ore & waste tonnages exceed the targeted amount of ore and/or waste should not be formulated. Note that the benches in the same pushback must be mined in a top-down manner and that the benches of a succeeding pushback can be mined before the benches of a preceding pushback are completely mined out. The benches of succeeding pushbacks should be included in the problem formulation. Once a pushback is considered for problem formulation, the benches of that pushback are subjected to the tonnage limiting rule. As far as how many pushbacks should be formulated, this decision is a subjective one. In this study, there are only four pushbacks. A pushback is included in a problem formulation as long as there are benches left for scheduling in that pushback.

For the second approach, if the benches of the first pushback entering the problem formulation are at levels 1, 2, 3, ..., k respectively, then, there is no reason to include the second pushback benches at levels lower than k in problem formulation. In other words, no second pushback benches below the k-th level should be formulated because of the SR constraints. Similarly, if the lowest formulated bench of a preceding pushback is at level k, the benches of succeeding pushbacks below the level k should not be formulated. An exception to this rule is: 1) when the kth
bench is the last bench of a preceding pushback (meaning that no more benches are below this last bench in the preceding pushback) and 2) the benches in preceding pushback are all mined out, e.g., there is no precedence constraints between the benches of preceding pushbacks and the benches of succeeding pushbacks. An example for this second approach can be seen through figure 5.1, where the last bench of pushback 1 is bench 2. If the lowest bench of pushback 1 in problem formulation is bench 1, benches of pushback 2 below the bench 1 (i.e., bench 2, 3) should not be formulated. Only when bench 2 of pushback 1 is in the problem formulation and the tonnage limiting rule has not yet become a factor for pushback 2, can the bench 2 and/or 3 of pushback 2 be formulated.

For the third approach, when the first two strategies fail to reduce the number of decision variables to a convenient range, one can combine several top benches in each pushback into one composite bench. This is only when one is sure that all the benches in the composite bench are within the tonnage requirement and the composite bench is an acceptable scheduling unit.

Using the three ways, of reducing the number of decision variables, should drastically shorten the computation time in the pushback mine sequencing problem.

The first two of the above approaches to limiting the number of decision variables do not constitute very tight constraints. In case there is a need, the first two approaches to limiting the number of decision variables can be combined to form a much tighter constraint.

This tighter constraint is based on not including more than the targeted ore tons into the problem formulation from any one mining pattern. Here, a mining pattern refers to any one bench and all its preceding benches according to precedence constraints.

For example, in figure 5.1 and figure 5.2, there are 2 benches in pushback 1 and 3 benches in pushback 2. The top bench is bench 1 in both pushbacks 1 and 2. The precedence constraints are defined as follows: for benches in the same pushback, a lower level bench can not be mined without the removal of all the higher
Figure 5.1 Cross Section of Pushback 1 & 2

(Bold line delineates the boundaries between pushbacks)
level benches. In addition, the benches in pushback 2 can not be mined unless the same level benches in pushback 1 are mined. According to this precedence constraint definition, the benches of pushback 1 have two mining patterns. The first mining pattern consists of only bench 1. The second mining pattern consists of bench 1 and 2. The benches of pushback 2 have 3 mining patterns. The first mining pattern consists of the top benches of both pushback 1 and pushback 2. The second mining pattern consists of bench 1 & 2 of pushback 1 and bench 1 & 2 of pushback 2. The third mining pattern consists of bench 1 & 2 of pushback 1 and bench 1 & 2 & 3 of pushback 2.

Please note that a mining pattern derived from a lower bench of any pushback always includes the mining patterns derived from an upper bench of the same pushback as well as the same and upper level benches of all the inner pushbacks. Please note also that even the top bench of pushback 2 can not be mined without mining the bench 1 of pushback 1. But, bench 1 of pushback 2 can be mined ahead of bench 2 of pushback 1. This is because bench 1 of pushback 1 and bench 1 of pushback 2 form a legitimate mining pattern.

Therefore, whether any particular bench is formulated or not depends on whether the cumulative ore and waste tonnages exceed the targeted amount following a mining pattern to that bench. Once the targeted amount of ore & waste tonnages are exceeded, the benches of lower priority are no longer considered in the problem formulation, i.e., the larger mining patterns that contain the current mining pattern in consideration are all excluded from problem formulation. For instance, in figure 5.2, if bench 1 of pushback 1 exceeds the targeted ore & waste tonnages, no other benches in both pushback 1 and 2 will be formulated. But, if bench 1 of pushback 2 exceeds the targeted ore & waste tonnages, only benches 2 and 3 of pushback 2 will be excluded from problem formulation.

This last approach constitutes a tighter constraint because of limiting the number of decision variables according to mining patterns instead of pushbacks.
pushback 1

bench 1

Mining pattern when only bench 1 of pushback 1 is in consideration

pushback 1  pushback 2

bench 1  bench 1

Mining pattern when bench 1 of pushback 2 is in consideration.

pushback 1

bench 1

bench 2

Mining pattern when bench 2 of pushback 1 is in consideration.

Figure 5.2 Mining patterns derived from each bench according to precedent constraints. (Cross section views)
Figure 5.2 Mining patterns derived from each bench according to precedence constraints. (Cross section views) (cont'd)
5.3.3 Definition of Objective Functions

In this study, the objective of the mine sequencing problem involves any one of the four items: grade, ore tonnage requirements, waste tonnage requirements and stripping ratio.

The objectives of mine sequencing can be defined as the following:
1) Maximize cumulative waste tonnages.

\[ \sum_{i=1}^{n} \sum_{j=1}^{m(i)} w_{ij} x_{ij} \]  \hspace{1cm} (5.5)

2) Minimize cumulative waste tonnages.

\[ \sum_{i=1}^{n} \sum_{j=1}^{m(i)} w_{ij} x_{ij} \]  \hspace{1cm} (5.6)

3) Maximize cumulative ore tonnages.

\[ \sum_{i=1}^{n} \sum_{j=1}^{m(i)} o_{ij} x_{ij} \]  \hspace{1cm} (5.7)

4) Minimize cumulative ore tonnages.

\[ \sum_{i=1}^{n} \sum_{j=1}^{m(i)} o_{ij} x_{ij} \]  \hspace{1cm} (5.8)

5) Maximize cumulative mineral contents.

\[ \sum_{i=1}^{n} \sum_{j=1}^{m(i)} c_{ij} x_{ij} \]  \hspace{1cm} (5.9)

6) Maximize cumulative stripping ratios.

\[ \sum_{i=1}^{n} \sum_{j=1}^{m(i)} \left( \frac{w_{ij}}{o_{ij}} \right) x_{ij} \]  \hspace{1cm} (5.10)

7) Minimize cumulative stripping ratios.
\[ \sum_{i=1}^{n} \sum_{j=1}^{m(i)} (w_{ij}/o_{ij})x_{ij} \]  

Where

- \( n \) = number of pushbacks;
- \( m(i) \) = number of benches in \( i \)-th pushback;
- \( w_{ij} \) = waste tonnage contained in \( j \)-th bench of \( i \)-th pushback;
- \( o_{ij} \) = ore tonnage contained in \( j \)-th bench of \( i \)-th pushback;
- \( c_{ij} \) = mineral contents contained in \( j \)-th bench of \( i \)-th pushback;
- \( c_{ij} = v_{ij} \times sp_{ij} \times g_{ij} \);
- \( v_{ij} \) = volume of \( j \)-th bench in \( i \)-th pushback;
- \( sp_{ij} \) = specific gravity of materials in \( j \)-th bench of \( i \)-th pushback;
- \( g_{ij} \) = average ore grade of \( j \)-th bench of \( i \)-th pushback;
- \( x_{ij} \) = decision variable that represents \( j \)-th bench of \( i \)-th pushback.

Any one of the above objective functions may be applied to the following situations:

1. In some planning period, one may want to strip as much waste as possible to ensure necessary ore tonnage exposure. In some other periods, one may want to delay waste stripping as long as possible to increase NPV. This is where the first and the second objectives can be utilized.

2. In some planning period, one may find that it is difficult to find enough ore. One may want to determine the maximum ore tonnages that are obtainable. In some other occasions, one may have to mine more than the targeted ore tonnages to strip a specified amount of waste. Since one wants to keep constant ore exposure, one may ask what are the minimum ore tonnages that have to be carried with the
targeted waste stripping. This is where the third and the fourth objectives may be helpful.

3. During production periods, higher grades means higher revenues at constant price and cost. One may want to maximize the average grade among all the benches scheduled to be mined. This is where the fifth objective can be utilized.

4. Minimization of waste tonnages does not necessarily ensure minimized stripping ratios. Also, maximization of waste tonnages does not necessarily offer maximized stripping ratios. To work out a smoothed stripping ratio plan, at one period or through out mine sequencing, one may want to know what the minimum possible stripping ratio is as well as the maximum possible stripping ratio. This is where the sixth and seventh objectives are useful.

Of all the objectives, maximization of the mineral contents (average grades) for each planning period should be the most attractive objective. This is due to two factors. Firstly, to promise constant ore supply in subsequent periods, one strips in advance and this stripping may happen in outer pushbacks before the inner pushbacks are completely mined. This “breaks” the “next best” mining sequence indicated by the inner to outer pushbacks. Secondly, higher grades in early periods mean higher value of NPV. Therefore, the desired long term objective of the Diamond mine sequencing problem is to maximize mineral contents in each planning period. However, which of the above seven objectives is selected depends on how difficult it is to work out a mining sequence. In case it is difficult to obtain a feasible schedule for a multi-period sequencing problem, other objectives may have to step in.

5.3.4 Formulation of Problem Constraints

The ore tonnage requirement is set up as follows:

\[
\sum_{i=1}^{n} \sum_{j=1}^{m(i)} a_{ij} x_{ij} \leq t_{ot} + b_{uot} \tag{5.12}
\]
The waste tonnage requirement is set up as follows:

\[
\sum_{i=1}^{n} \sum_{j=1}^{m(i)} o_{ij} x_{ij} \geq t_{ot} - b_{lot} \quad (5.13)
\]

\[
\sum_{i=1}^{n} \sum_{j=1}^{m(i)} w_{ij} x_{ij} \leq t_{wt} + b_{uwt} \quad (5.14)
\]

\[
\sum_{i=1}^{n} \sum_{j=1}^{m(i)} w_{ij} x_{ij} \geq t_{wt} - b_{lwt} \quad (5.15)
\]

Where \(n, m(i), o_{ij}, w_{ij}, x_{ij}\) are defined as previous.

\(t_{ot}\) = ore tonnage requirement in period \(t\),
\(b_{uot}\) = an upper limit buffer ore tonnage for period \(t\), where \(0 \leq b_{uot} \leq \max [o_{ij}, i = 1, \ldots, n; j = 1, \cdots, m(i)]\)
\(b_{lot}\) = a lower limit buffer ore tonnage for period \(t\), where \(0 \leq b_{lot} \leq \max [o_{ij}, i = 1, \ldots, n; j = 1, \cdots, m(i)]\)
\(t_{wt}\) = waste tonnage requirement in period \(t\),
\(b_{uwt}\) = an upper limit buffer waste tonnage for period \(t\), where \(0 \leq b_{uwt} \leq \max [w_{ij}, i = 1, \ldots, n; j = 1, \cdots, m(i)]\)
\(b_{lwt}\) = a lower limit buffer waste tonnage for period \(t\), where \(0 \leq b_{lwt} \leq \max [w_{ij}, i = 1, \ldots, n; j = 1, \cdots, m(i)]\)

The provision of \(b_{uot}, b_{lot}, b_{uwt}\) and \(b_{lwt}\) is to guarantee an integer feasible solution. A set of solutions based on either to mine or not to mine decision may not fit exactly the defined target. With \(b_{uot}, b_{lot}, b_{uwt}\) and \(b_{lwt}\) defined as above, one ensures an integer feasible solution if there is one to the problem. In general, the smaller the values of these \(b_{uot}, b_{lot}, b_{uwt}\) and \(b_{lwt}\), the better the ore \& waste tonnages constraints are enforced. These buffer values can be assigned to the middle point values corresponding to their range. For example, if the \(\max [o_{ij}, i = 1, \ldots, n; j = 1, \cdots, m(i)]\) is 500,000, then, a mid point value for \(b_{uot}\) would be \((0 + 500,000)/2.0 = 250,000\).
5.3.5 Precedence Constraints Formulation of Mine Sequencing Problem

In the pushback mine sequencing problem, a lot of precedence constraints or sequential requirements (SR) must be considered. Numerous benches from the same pushback are subjected to SR, e.g., there are 13 to 29 benches in the Diamond problem. The benches from different pushbacks but on the same level are also subjected to SR. The first SR is because of the nature of geometry. One cannot mine an underlying bench without removing the overlying bench. The second SR is due to both pit slope and mine sequencing optimization requirements. Note that the inner pushbacks generally have better mineralizations.

Based on the result of section 5.2.4, one SR chain requires one constraint. The precedence constraints of mine sequencing problem are formulated as follows:

For the benches of pushback \( i \)

\[
\sum_{i=1}^{n} \sum_{j=1}^{q} a_{ij} x_{ij} \geq \sum_{i=1}^{n} \sum_{j=1}^{q} a_{ij}
\]

(5.16)

where

\( x_{ij} = \) decision variable representing \( j \)-th bench of \( i \)-th pushback

\( a_{ij} = \) coefficient whose value is assigned as:

\( a_{ij} > 0 \) for decision variables subjected to SR chain of pushback \( i \),

\( a_{ij} = 0 \) otherwise,

\( 1 \leq q \leq m(i) \)

\( q \) varies according to the maximum subscript \( j \) value, i.e., the bottom bench of pushback \( i \), of a solution under examination,

\( m(i) = \) number of benches from pushback \( i \) in problem formulation.

\( n \) is the number of pushbacks as before.

For benches on the \( j \)-th level from different pushbacks:

\[
\sum_{i=1}^{n} \sum_{j=1}^{p} a_{ij} x_{ij} \geq \sum_{i=1}^{n} \sum_{j=1}^{p} a_{ij}
\]

(5.17)
where

\[ a_{ij} = \text{coefficient whose value is assigned as:} \]

\[ a_{ij} > 0 \text{ for decision variables subjected to SR chain of level } j, \]

\[ a_{ij} = 0 \text{ otherwise,} \]

\[ 1 \leq p \leq \max[m(i), 1 \leq i \leq n] \]

\( p \) varies according to the maximum subscript \( j \) value, i.e., the bottom bench of a solution under examination,

all the remaining variables are defined as in (5.16).

5.3.6 A Complete 0-1 Programming Formulation of Mine Sequencing Problem

Taking maximization of cumulative mineral contents as an example, a complete 0-1 programming formulation of a mine sequencing problem is given as follows:

\[
\begin{align*}
\text{Maximization} & \quad \sum_{i=1}^{n} \sum_{j=1}^{m(i)} c_{ij} x_{ij} \\
\text{Subject to:} & \quad \sum_{i=1}^{n} \sum_{j=1}^{m(i)} a_{ij} x_{ij} \leq t_{ot} + b_{uot} \\
& \quad \sum_{i=1}^{n} \sum_{j=1}^{m(i)} a_{ij} x_{ij} \geq t_{ot} - b_{lot} \\
& \quad \sum_{i=1}^{n} \sum_{j=1}^{m(i)} w_{ij} x_{ij} \leq t_{wt} + b_{uw} \\
& \quad \sum_{i=1}^{n} \sum_{j=1}^{m(i)} w_{ij} x_{ij} \geq t_{wt} - b_{lw} \\
& \quad \sum_{i=1}^{n} \sum_{j=1}^{p} a_{kij} x_{ij} \geq \sum_{i=1}^{n} \sum_{j=1}^{p} a_{kij} \\
& \quad \sum_{i=1}^{n} \sum_{j=1}^{p} a_{lij} x_{ij} \geq \sum_{i=1}^{n} \sum_{j=1}^{p} a_{lij}
\end{align*}
\]

where

\( n = \text{number of pushbacks,} \)

\( m(i) = \text{number of benches inside } i - \text{th pushback} \)

\( x_{ij} = \text{decision variable represents } j - \text{th bench of } i - \text{th pushback} \)

\( w_{ij} = \text{waste tonnage contained in } j - \text{th bench and } i - \text{th pushback} \)
\( o_{ij} \) = ore tonnage contained in \( j \)-th bench and \( i \)-th pushback;

\( c_{ij} \) = mineral contents contained in \( j \)-th bench of \( i \)-th pushback;

\( c_{ij} = v_{ij} \times sp_{ij} \times g_{ij} \)

\( v_{ij} \) = volume of \( j \)-th bench in \( i \)-th pushback

\( sp_{ij} \) = specific gravity of materials in \( j \)-th bench of \( i \)-th pushback,

\( g_{ij} \) = average ore grade of \( j \)-th bench of \( i \)-th pushback,

\( a_{kij} \) = coefficient whose value is assigned as:

\( a_{kij} > 0 \) for decision variables subjected to the SR of pushback \( k \).

\( a_{kij} = 0 \) otherwise,

\( a_{lij} \) = coefficient whose value is assigned as:

\( a_{lij} > 0 \) for decision variables subjected to the SR of level \( l \).

\( a_{lij} = 0 \) otherwise,

\( t_{ot} \) = ore tonnage requirement in period \( t \),

\( b_{uot} \) = an upper limit buffer ore tonnage for period \( t \), where

\( 0 \leq b_{uot} \leq \max[o_{ij}, i = 1, \ldots, n; j = 1, \ldots, m(i)] \)

\( b_{lot} \) = a lower limit buffer ore tonnage for period \( t \), where

\( 0 \leq b_{lot} \leq \max[o_{ij}, i = 1, \ldots, n; j = 1, \ldots, m(i)] \)

\( t_{wt} \) = waste tonnage requirement in period \( t \),

\( b_{uwot} \) = an upper limit buffer waste tonnage for period \( t \), where

\( 0 \leq b_{uwot} \leq \max[w_{ij}, i = 1, \ldots, n; j = 1, \ldots, m(i)] \)

\( b_{lwot} \) = a lower limit buffer waste tonnage for period \( t \), where

\( 0 \leq b_{lwot} \leq \max[w_{ij}, i = 1, \ldots, n; j = 1, \ldots, m(i)] \)

\( 1 \leq i \leq n \)

\( 1 \leq k \leq n \)

\( 1 \leq j \leq m(i) \)

\( 1 \leq q \leq m(i) \)

\( 1 \leq p \leq \max[m(i), 1 \leq i \leq n] \)
\[1 \leq l \leq \max[m(i), 1 \leq i \leq n]\]

5.4 Implementation of the 0-1 Programming Mine Sequencing Model

The implementation of the 0-1 programming mine sequencing model involves repeatedly applying the single period mine sequencing formulation (eq. 5.18), one for each planning period, until the multi-period mine sequencing is done. In the process of multi-period mine sequencing, each subsequent period mine sequencing is based on the results of mine sequencing of previous periods. The implementation of the 0-1 programming mine sequencing model is illustrated in the next chapter using the Diamond pit mine sequencing case.
CHAPTER 6

APPLICATION OF THE 0-1 PROGRAMMING MODEL TO THE DIAMOND MINE SEQUENCING PROBLEM

6.1 Diamond Mine Sequencing Project Case Summary

After finishing ultimate pit limit design, 4 pushbacks are generated. The number of benches in each pushback is respectively 13, 17, 23 and 29, where bench by bench inventories on ore tonnage, waste tonnage and average grade values for a series of cutoff grades for each incremental pushback are available.

The client of Diamond project hands over an ore & waste tonnage schedule and cutoff grades for six periods and requests the development of a bench by bench mining schedule based on their tonnage requirements. The client’s tonnage schedule is summarized in table 6.1. At present, there is no grade balancing requirement but higher grades in earlier production periods are desired.

### Table 6.1 Summary of client’s tonnage schedule

<table>
<thead>
<tr>
<th>Period #</th>
<th>Ore tonnage</th>
<th>Waste tonnage</th>
<th>Stripping Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>267,000</td>
<td>747,000</td>
<td>2.80</td>
</tr>
<tr>
<td>2</td>
<td>833,000</td>
<td>2,052,000</td>
<td>2.46</td>
</tr>
<tr>
<td>3</td>
<td>2,000,000</td>
<td>4,771,000</td>
<td>2.39</td>
</tr>
<tr>
<td>4</td>
<td>2,000,000</td>
<td>5,652,000</td>
<td>2.83</td>
</tr>
<tr>
<td>5</td>
<td>2,000,000</td>
<td>5,636,000</td>
<td>2.82</td>
</tr>
<tr>
<td>6</td>
<td>1,436,000</td>
<td>538,000</td>
<td>0.37</td>
</tr>
</tbody>
</table>

6.2 Definition of Sub-optimization Objectives

The long term objective for the Diamond project is defined as maximization of mineral contents (total ore tonnage x average grade). This is because higher
average grade in early periods brings higher NPV returns to the project. Thus, the sub-optimization objective for each planning period is defined as the maximization of mineral contents. Occasionally, the stripping ratio is minimized when it is difficult to reduce the stripping ratio for any single period.

6.3 Solution Procedure for the Diamond Project

As mentioned in the previous chapter, multi-period mine sequencing is carried out period by period by defining a series of sub-optimization objectives for each individual planning period. A subsequent period mine sequencing is based on the results of mine sequencing for previous periods. The solution procedures (steps) for a multi-period mine sequencing are as follows:

1) pick up remaining unmined benches from every pushback, classify the materials in each bench into ore & waste categories according to current cutoff grade strategy. The number of benches from each pushback is limited according to the discussion of section 5.3.2 except that there are no composite bench complications. That there are no composite bench complications in the Diamond pit sequencing is due to the fact that the number of decision variables in most 0-1 programming formulations is less than 50 after applying the first two decision variable limiting rules. In general, Balas’ algorithm has no problems in solving a 50-variable 0-1 programming problem.

2) Set up a priority sequence among the selected benches. The priority sequence is set up following these rules:

a) The benches from the same pushback are subjected to precedence constraints such that the benches from a lower position must not be mined unless the benches above them are all removed;

b) The benches from a given pushback are subjected to precedence constraints such that the benches of outer pushbacks must not be mined unless the benches of inner pushbacks are all removed if these benches are on the same level.
Remarks on the priority sequence setup:

Ideally, the priority setup for the benches from a given pushback maybe such that the last bench of an inner pushback has higher priority over the top bench of an outer pushback. This kind of priority setup forces the 0-1 programming to mine all the benches of an inner pushback before the benches of outer pushbacks are mined. If there is no feasible solution, then the next to last bench of an inner pushback can be set up to have higher priority than the top bench of the outer pushback and so on. However, this approach is slow and unnecessary in this study.

The priority setup in this study adopts an indirect approach to insure that the benches of inner pushbacks attain higher priorities over the benches of outer pushbacks. This is achieved by assigning smaller subscripts to the benches of inner pushbacks as compared to outer pushbacks as well as to the benches of higher elevations as compared to lower elevations. In other words, the highest bench in the first pushback (most inner pushback) is assigned the smallest subscript value (1). The second highest bench in the first pushback is assigned the second smallest subscript value (2). If there are 7 benches from pushback 1 in the problem formulation, the 7-th bench of pushback 1 is assigned a subscript 7. Then, the highest bench in second pushback is assigned a subscript value of 8. The second highest bench in second pushback is assigned a subscript of 9 and so on. Note that in precedence constraint formula (5.3), the variables with smaller subscript value assume higher priorities. Also, a lower subscript variable is likely to be evaluated earlier in 0-1 programming. Thus, the partial solutions containing the benches of higher elevations and the benches of inner pushbacks attain higher priorities, to be considered first. If the completions of these partial solutions are feasible and the corresponding objective value is improved, the algorithm will record them.

Finally, it should be noted that the priority setup of this study automatically satisfies the pit slope requirement.

3) Formulate the selected benches into 0-1 programming problem. The decision variable is defined as given in section 5.3.1.
The assignment of decision variables follows an order as below:

a) The benches from the inner pushbacks are assigned smaller subscripts than the benches from outer pushbacks;

b) For benches from the same pushback, the benches at higher positions are assigned smaller subscripts than the benches from the lower positions. Thus, the highest bench of the first pushback is assigned to be the first decision variable. The lowest bench of the last pushback is assigned to be the last decision variable.

The objective for all the planning periods is to maximize mineral contents except for the first period where the sub-optimization objective is chosen as minimization of stripping ratios. This is because it is difficult to reduce the stripping ratio for the first planning period.

The objective functions for minimization of stripping ratios and maximization of mineral contents can be found in section (5.3.3).

The minimization of stripping ratios can directly adopt the objective function of equation (5.4). However, the maximization problem must be transformed into a minimization problem in order to apply the implicit enumeration algorithm. The precedence constraint setup must be changed accordingly. The transformation is done in two steps:

a) Let \( Z = f(x) \) represent the objective function to be maximized. Since the minimization of \((-Z)\) is equal to the maximization of \((Z)\), a maximization problem is transformed into a minimization problem by multiplying every item in the objective function by \((-1)\).

b) Make a substitution of \( x_{ij} = 1 - y_{ij} \) for all the decision variables whose objective function coefficients are negative. This substitution is done for the variables both in the objective and in the problem constraints. Note that the precedence constraint order is reversed after the substitution. This is because if \( x_1 \geq x_2 \geq x_3 \geq \ldots \geq x_n \), then \( 1 - y_1 \geq 1 - y_2 \geq 1 - y_3 \geq \ldots \geq 1 - y_n \) and \( y_1 \leq y_2 \leq y_3 \leq \ldots \leq y_n \).

In this manner, the maximization of mineral contents is transformed into a
standard 0-1 program and is solved via Balas' implicit enumeration algorithm.

At the end of the implicit enumeration, a reverse substitution of \( y_{ij} = 1 - x_{ij} \) is made to obtain the solution to the original maximization problem.

4) The solution from 0-1 programming is adjusted to obtain the exact ore tonnage to guarantee a constant mill feed. Every time an amount of ore is put back or taken out from a bench, a certain waste tonnage is also put back or taken out, in proportion to the current bench stripping ratio. This adjustment is necessary because of the buffer value relaxation of ore and waste tonnages in 0-1 program formulation (5.18).

This solution adjustment should not be viewed as a shortcoming of this scheduling model. Rather, it is an advantage because one can have the final control of the schedule after 0-1 programming provides an optimized guideline.

The solution adjustment procedure is explained in section 6.5 using the sample computer run for the period 1.

If the last period mine sequencing is done, stop. The multi-period mine sequencing is finished. Otherwise, go to step 1 to continue with the next period mine sequencing.

6.4 Problem Formulation for the 1st Period - Sample Layout

The requirements for the 1st period are listed as follows:
Ore: 267,000 tons
Waste: 747,000 tons
Stripping ratio: 2.80
Number of pushbacks remaining: 4

The number of decision variables is determined by first adding the benches of each pushback in a top-down approach so that the cumulative ore and waste tonnages from each pushback do not exceed the targeted ore and waste tonnages, i.e., 267,000 tons of ore and 267,000 \( \times \) 2.80 (747,600) tons of waste. Note that
the waste tonnage constraint is calculated according to the desired stripping ratio multiplied by the required ore tonnage. This is because in most cases, the input data to the mine sequencing problem is to provide mill feed requirement and the desired stripping ratio.

The reserves are classified into 5 classes. Class 5 contains waste materials. Class 1 - 4 contain 'ore' materials with higher class materials having higher grades. Figure 6.1 shows the reserves for pushback #1. The reserves for the other 3 pushbacks are listed in appendix A. In figure 6.1, the first five lines are remark lines. The rest are reserve data. The numerals in the first column are bench numbers. The numerals in the second column are class classifications for the material in those classes. The figures in the third column are the reserve tonnages and the figures in the fourth column are the reserve grades (oz/ton).

Before problem formulation begins, one may find the following remarks helpful. One of the major tasks in mine sequencing is to determine which bench from which pushback should be mined to satisfy certain ore and waste tonnage requirements as well as the stripping ratio requirements. In this study, 0-1 programming has been used to make such a decision.

The best way to carry out this mission is to incorporate all the "potentially mineable benches" into the problem formulation and use 0-1 programming to make the choice on which bench of which pushback should be mined for the current period. Here, the "potentially mineable benches" refer to all the remaining unmined benches of all pushbacks except those benches which are obviously non-mineable:

1) the benches that should be excluded based on the cumulative ore & waste tonnage rule, i.e., the first approach in limiting the number of decision variables.

2) the benches that should be excluded based on precedence constraints among the benches of different pushbacks, but are on the same levels, i.e., the second approach in limiting the number of decision variables.

Thus, the problem formulation proceeds in the following manner for the 1st
| Figure 6.1 Reserve List of Pushback 1 |
period: 1) formulate benches of pushback 1 in a top-down search fashion until the cumulative ore & waste tonnages exceed the targeted ore & waste tonnages; 2) formulate the benches of the second pushback also to the targeted ore & waste tonnages provided that the precedence constraint has not become a limiting factor. The benches of pushback 3 and 4 are determined in the same way as for the benches of pushback 2.

With the benches of all pushbacks thus formulated for this period, a "fair" chance for being mined is ensured every potentially mineable benche. Note that in the Diamond mine sequencing, there are 4 pushbacks and a total of 82 benches. Applying the first two strategies discussed in section 5.3.2, the number of decision variables in the Diamond mine sequencing problem formulation for all 6 planning periods can readily be reduced to less than 50 or so. Therefore, the tighter rule of limiting the number of decision variables discussed at the end of section 5.3.2 is not applied in this case study. If there are many more pushbacks and benches, one may use the tighter limiting rule to formulate benches so that the number of decision variables is limited and the 0-1 program is solvable.

The benches of pushback #1 entering the first period problem formulation are determined as shown in table 6.2.

For pushback #1, at bench 30, the cumulative ore & waste tonnages reach the targeted amounts. Thus, the benches of pushback #1 entering the problem formulation of period 1 are 24, 25, 26, 27, 28, 29 and 30. This in turn limits the bench of any succeeding pushbacks going below bench 30 during the 1st period formulation.

The benches of pushback #2 entering the first period's problem formulation are determined likewise except that the bottom bench must not go below bench 30. This bench 30 limitation is because of the precedence constraints among the same level benches of distinct pushbacks. The benches of pushback #2 entering the problem formulation of period 1 are selected as shown in table 6.3.

In table 6.3, at bench 30, both the cumulative values of ore & waste are less
Table 6.2 Benches of pushback #1 entering 1st period problem formulation

<table>
<thead>
<tr>
<th>Bench</th>
<th>Class</th>
<th>Ore (1000's tons)</th>
<th>Waste (1000's tons)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>By Class</td>
<td>By Bench</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>1.136</td>
<td>1.136</td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>6.439</td>
<td>6.439</td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>1.121</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>57.587</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>3</td>
<td>3.977</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>0.189</td>
<td>62.875</td>
</tr>
<tr>
<td>28</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>11.477</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>105.498</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>3</td>
<td>2.669</td>
<td>119.644</td>
</tr>
<tr>
<td>29</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>14.867</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>122.784</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>13.258</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>0.833</td>
<td>151.742</td>
</tr>
</tbody>
</table>

than the targeted tonnages. The binding factor for choosing benches of pushback #2 is the precedence constraint at bench 30 between pushback 1 and 2. Therefore, benches 24, 25, 26, 27, 28, 29 and 30 are in first period's problem formulation.

The benches of pushback 3 and 4 entering the first period's problem formulation are selected as for pushback 1 and 2. At the end of bench selection, 28 benches are chosen. Since a 28-variable 0-1 programming problem can be solved easily via Balas' algorithm, the composite bench grouping becomes unnecessary. Among the 28 benches in the first period's problem formulation, 7 are from each pushback. The precedence constraints among these 28 benches are shown in figure 6.2.
Table 6.3 Benches of pushback #2 entering 1st period problem formulation

<table>
<thead>
<tr>
<th>Bench</th>
<th>Class</th>
<th>Ore (1000's tons)</th>
<th>Waste (1000's tons)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>By Class</td>
<td>By Bench</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>0.540</td>
<td>0.540</td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>0.583</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>17.225</td>
<td>17.809</td>
</tr>
<tr>
<td>28</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>3.674</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>40.184</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>3</td>
<td>0.835</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>4</td>
<td>1.610</td>
<td>46.303</td>
</tr>
<tr>
<td>29</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>55.625</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>3.788</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>4.848</td>
<td>64.261</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In figure 6.2, the first 4 columns represent pushback 1, 2, 3 and 4 respectively. The first 9 lines represent bench levels. Each entry with a “Bench” and a number such as “Bench 24” represents a bench at a level indicated by the number. There are precedence constraints among the benches on the same level. There are also precedence constraints among the benches in the same pushback. The arrows in figure 6.2 point toward higher priority directions.

The decision variable assignments for the first period are shown in table 6-4. In table 6-4, the tonnage figures in bold emphasize the fact that the cumulative ore or waste tonnages reach or exceed the targeted amounts when considered individually by pushback. A bench number in bold indicates that problem formulation stops at that bench because of the precedence constraint between the benches of different pushbacks.
Figure 6.2 SR among the benches in first period’s problem formulation
Table 6.4 Decision variable assignment for the 1st period

<table>
<thead>
<tr>
<th>Pushback #</th>
<th>Bench #</th>
<th>Decision Variable</th>
<th>Ore (1000 tons)</th>
<th>Waste (1000 tons)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>1</td>
<td>3,428</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>2</td>
<td>18,585</td>
<td></td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>3</td>
<td>1,136</td>
<td>42,409</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>4</td>
<td>6,439</td>
<td>82,325</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>5</td>
<td>62,875</td>
<td>169,486</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>6</td>
<td>119,644</td>
<td>274,256</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>7</td>
<td>151,742</td>
<td>310,403</td>
</tr>
<tr>
<td>Sub-total</td>
<td>7</td>
<td>7</td>
<td><strong>341,836</strong></td>
<td><strong>900,892</strong></td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>8</td>
<td>644</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>9</td>
<td>11,150</td>
<td></td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>10</td>
<td>28,572</td>
<td></td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>11</td>
<td>540</td>
<td>47,052</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>12</td>
<td>17,809</td>
<td>101,036</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>13</td>
<td>46,303</td>
<td>192,428</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>14</td>
<td>64,261</td>
<td>304,684</td>
</tr>
<tr>
<td>Sub-total</td>
<td>7</td>
<td>7</td>
<td>128,913</td>
<td>685,566</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>15</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>16</td>
<td>4,422</td>
<td>4,938</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>17</td>
<td>32,420</td>
<td>38,080</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>18</td>
<td>42,123</td>
<td>103,995</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>19</td>
<td>48,384</td>
<td>185,273</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>20</td>
<td>43,712</td>
<td>266,799</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>21</td>
<td>44,962</td>
<td>342,564</td>
</tr>
<tr>
<td>Sub-total</td>
<td>7</td>
<td>7</td>
<td>216,023</td>
<td><strong>941,733</strong></td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>22</td>
<td>7,005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>23</td>
<td>18,467</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>24</td>
<td>2,429</td>
<td>39,107</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>25</td>
<td>975</td>
<td>74,831</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>26</td>
<td>4,511</td>
<td>139,044</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>27</td>
<td>13,019</td>
<td>223,390</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>28</td>
<td>19,394</td>
<td>349,671</td>
</tr>
<tr>
<td>Sub-total</td>
<td>7</td>
<td>7</td>
<td>40,328</td>
<td><strong>851,515</strong></td>
</tr>
</tbody>
</table>
Based on the decision variable assignments, ore & waste target requirements and the precedence constraints among decision variables, the first period problem formulation is worked out using equation (5.18). Figure 6.3 shows the computer print out for the first period problem formulation.

In figure 6.3, under "COEF. OBJ. FUNC." are the objective coefficients of the problem formulation. The 28 objective coefficients are printed such that 8 coefficients make up one row until the 28 objective coefficients are printed as 3 lines with 8 coefficients and 1 line with 4 coefficients. The first period's objective is assigned as minimization of stripping ratios. Therefore, the figures in the objective function are the stripping ratios associated with the corresponding decision variables (benches).

The reason for choosing minimization of stripping ratio as the first period's objective is because it is difficult to reduce the stripping ratio in the first period. Here, two points should be noted: 1) the stripping ratios have all been transformed into integers. This is done by multiplying every ratio by a factor (10,000 in figure 6.3). For example, the decision variable 4 represents bench 27 of pushback #1. In that bench, the stripping ratio is 82,325/6,439 which equals 12.79 or approximately 13.0. This ratio (13) is then multiplied by the factor 10,000. This can be checked against the coefficient of decision variable 4 in the objective function. The factor 10,000 is chosen here to transform all the stripping ratios into approximately the magnitude of bench tonnages. The reason for doing this is for computation purposes. 2) If a bench does not have any ore, the stripping ratio of that bench is assigned a large integer (100 in figure 6.3). This number (100) is then multiplied by the factor 10,000 giving 1,000,000 as the coefficient of the corresponding decision variable. This can be seen in the objective function coefficients of decision variables 1, 2, 8, 9, 10, 15, 22 and 23.

Under the "COEF. RHS." are the right hand side (RHS) values of 0-1 programming formulation. Note that the one dimensional RHS matrix in the original 0-1 programming formula is extended to a two dimensional matrix. This
Figure 6.3 Computer Printout of 0-1 Programming Problem Formulation for Period 1
<table>
<thead>
<tr>
<th>COEF.</th>
<th>RHS</th>
<th>COEF.</th>
<th>RHS</th>
<th>COEF.</th>
<th>RHS</th>
<th>COEF.</th>
<th>RHS</th>
<th>COEF.</th>
<th>RHS</th>
<th>COEF.</th>
<th>RHS</th>
<th>COEF.</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
<td>191129</td>
</tr>
<tr>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
<td>-922435</td>
</tr>
<tr>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
<td>572765</td>
</tr>
<tr>
<td>444486</td>
<td>816441</td>
<td>1127700</td>
<td>1388168</td>
<td>1606133</td>
<td>1788530</td>
<td>1941164</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>816441</td>
<td>1127700</td>
<td>1388168</td>
<td>1606133</td>
<td>1788530</td>
<td>1941164</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1127700</td>
<td>1388168</td>
<td>1606133</td>
<td>1788530</td>
<td>1941164</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1388168</td>
<td>1606133</td>
<td>1788530</td>
<td>1941164</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 6.3 Computer Printout of 0-1 Programming Problem Formulation for Period 1 (cont’d)
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>24</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.3 Computer Printout of 0-1 Programming Problem Formulation for Period 1 (cont’d)
**Figure 6.3** Computer Printout of 0-1 Programming Problem Formulation for Period 1 (cont’d)
Figure 6.3 Computer Printout of 0-1 Programming
Problem Formulation for Period 1 (cont'd)

<table>
<thead>
<tr>
<th>16</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
extension is to accommodate the precedence constraints of (5.16) and (5.17), where
the RHS values vary according to the maximum subscript value of a solution under
examination. Instead of calculating the RHS values every time they are needed, all
the possible RHS values for each precedence constraint (PC) are saved in an array so
that when the maximum subscript value of a solution is known, the corresponding
RHS value is readily available from the RHS array. It is obvious that the maximum
possible subscript value of all possible solutions is limited by the total number
of decision variables. Since one constraint needs one set of RHS values, a two
dimensional array is needed to hold all the possible RHS values for the PC. The first
dimension indicates the constraint identity and the second dimension the maximum
subscript value. To make program coding easier, the RHS values for the ore & waste
tonnage constraints are also extended into a two dimensional array, with the first
dimension indicating the constraint identity and the second dimension telling the
maximum subscript value. To make sure, a single, identical RHS value is picked up
from the RHS array for one ore & waste constraint, the same RHS value is extended
through out all the subscripts. The drawback of introducing a 2 dimensional RHS
array is that the program occupies larger computer memories. The advantage is
that the computing is faster. Based on the experience of this study, the computer
memory is not a limiting problem.

The RHS values in the first four constraints are production tonnage require-
ments with the first two for ore target tonnages and the second two for waste target
tonnages. The RHS values are worked out in the following manner. The targeted
ore tonnage for the first period is 267,000 tons. The upper and lower buffer ore
tonnages are 75,871 tons (figure 6.6). Therefore, an upper RHS ore value equals
342,871 (267,000 + 75,871). A lower RHS ore value equals 191,129 (267,000 - 75,871).
The upper and lower buffer waste tonnages are 174,835 tons (figure 6.6). Thus, an
upper RHS waste value equals 922,435 (267,000 x 2.80 + 174,835, 2.8 is the targeted
stripping ratio). A lower RHS waste value equals 572,765 (267,000 x 2.80 - 174,835).
The negative RHS values in constraint 1 and 3 are because of the transformation
of less than and equal constraints to greater than and equal ones.

The rest of the RHS values serve for sequential requirement purposes. One can check that the entries of the RHS values and the entries of MATRIX ELEMENTS satisfy equations (5.16) and (5.17). The problem matrix elements are under "MATRIX ELEMENTS". Both the RHS values and the matrix elements are printed in the same way as the objective coefficients. Finally, the constraints of all inequalities are greater than and equal type.

6.5 Mine Sequencing for Period 1 - Sample Computer Run Layout

A computer run for the first period mine sequencing is traced in figures 6.4 - 6.9. The discussion of this section follows the sequence of the computer list.

In figure 6.4, one can see that when the computer program PMAIN is executed, the terminal prompts the user to enter the total number of planning periods. The numeral 6 is entered. The computer then prints the current default period number on screen. The period number is automatically updated whenever a new period begins. In the sample run, 1 is printed on the screen. If this is a re-start, one can enter a period number to restart with. For the sample run, carriage return is entered telling the program to proceed with period 1 mine sequencing. (A carriage return in a program run is echoed as 0 in this chapter.) At this time, items displayed on the screen are: the targeted production tonnage (in 1000's tons), the lower limiting stripping ratio, the upper limiting stripping ratio, the cutoff class (grade) and overriding code for all the six planning periods with one line for one period. This is an echo of the input data which the program is reading in. Thus, for the first period, the targeted production tonnage is 267,000 tons, the lower limiting stripping ratio is 2.80, the upper limiting stripping ratio is also 2.80, the cutoff class is 1 (please refer to figure 6.1, where the reserve is divided into 5 classes of materials) and the overriding code is 0. Here, displaying all the input data for six periods is to remind the user of the overall scheduling requirements.
Enter total # of periods ==> 6

The current time period :  1
Enter a new one assuming change ==>  

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>267</td>
<td>2.80</td>
<td>2.80</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>833</td>
<td>2.46</td>
<td>2.46</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2000</td>
<td>2.39</td>
<td>2.39</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2000</td>
<td>2.83</td>
<td>2.83</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2000</td>
<td>2.82</td>
<td>2.82</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1436</td>
<td>0.37</td>
<td>0.37</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 6.4 Sample Mine sequencing for period 1
- Input for Mine Sequencing
The definition of overriding code is for the first period only. In the pre-production period, one may just want to strip waste without defining any ore requirements. The overriding code of 0 means that the targeted production tonnage is the ore tonnage to be produced. Waste tonnage requirement is to be calculated through the amount of targeted ore multiplied by the designated stripping ratio. The overriding code of 1 means that waste is a major item. The targeted production tonnage refers to the waste tonnage to be produced. There is no requirement on the amount of ore that should be mined. Currently, the overriding code has no effect on other periods.

Figure 6.5 lists the input control parameters to the 0-1 programming problem. This includes: optimization type, item to be optimized, maximum number of iterations allowed and the pivot row designation. It can be seen that, for the first period, the optimization type is minimization, the item to be minimized is s.r (stripping ratio), the maximum allowed number of iterations is 5000 and the pivot row is not defined.

A pivot row designation may speed up the 0-1 program solution process and may change the feasible solution stream. The program has the option of allowing the user to enter a starting pivot row number. If none is entered, the program selects the pivot row by itself. In the sample run, no pivot row is entered. After all the options are entered, the program asks the user to check if there are any options to be altered. One may notice that, by default, the current program version disabled constraint 3, which is grade constraint. Also, the SR flags for both benches in the same pushback and the benches on the same level are set to 1.

The program then lists the default upper and lower buffer values for ore and waste as shown in figure 6.6. The default upper and lower buffer values for ore are determined as one half of the maximum possible single bench ore value among all the formulated benches. The default upper and lower buffer values for waste are determined as one half of the maximum possible single bench waste value among all the formulated benches. The user has the option to change these default values.
Enter optimization type please
[0=minimization, 1=maximization] =>

Enter the variable you want to optimize
[1=ore, 2=waste, 3=grade, 4=s.r., 5=none] =>

Enter the maximum iterations [DEFAULT:99999] =>
5000

Enter pivot row [if no SR exists, hit return] =>

Any parameter you want to change?
Disabled constraint : 3
Current opt type : 0;
Current opt variable : 4;
Current max iteration allowed : 5000;
Current flag for SR in a pushback : 1
Current flag for SR in a level : 1
Pivot row is : 0
[0=no, 1=re-enter disabled constraint
2=re-enter optimization type,
3=re-enter optimization variable,
4=re-enter maximum iterations allowed,
5=re-enter SR in one pit,
6=re-enter SR on the same level.
7=re-enter pivot row]
Enter your choice please =>

Figure 6.5 Sample Mine sequencing for period 1
- Input for 0-1 Programming
so as to either tighten or loosen these constraints. No change is made in the sample run.

Next, the program enters the 0-1 programming solution process. Once a feasible solution is found, the program gives the choice of exiting the 0-1 program solution process and going to the post 0-1 programming solution adjustment. In the sample run, one feasible solution is found during iteration 1. (See figure 6.7.) The sample run continues the 0-1 programming solution process until the optimum solution is obtained. It turns out that the first feasible solution found thus far happens to be the optimum solution as well.

In listing feasible mining sequences (figure 6.7) from 0-1 programming, pushback names are listed along the horizontal direction. The bench numbers are listed vertically. At the cross positions, an entry of 0 means that the corresponding bench has been mined. An entry of 1 means that the corresponding bench has not been mined. (In appendix B, one may see -999's at cross positions. This indicates that the corresponding benches have been mined in previous periods. One may also see -8's which indicate non-existing benches).

After exiting the 0-1 program solution process, one enters the post 0-1 program solution adjustment routine. The post 0-1 program solution adjustment for the sample run is listed in figure 6.8. The program first displays whether there is a shortage or excess of ore as compared to the targeted amount. In the sample run, the 0-1 program solution provided 74,836 tons of excess ore. The program indicates that 151,742 tons of ore can be put back in bench 30 of pushback 1. Note that the mining is proceeding according to the precedence constraints, i.e., the benches of higher priorities get mined earlier. When there is an excess of ore in the schedule, one “unmines” or puts back this excess amount of ore in its original position. This putting back process follows the opposite of the mining priorities, e.g., among the mined benches, the benches of lowest priority get unmined first. Thus, in the sample run, bench 30 of pushback 1 is unmined first. The entered amount was 74,836 to meet the targeted ore tonnage exactly. The program then tells the user that
Current buffer values are:

<table>
<thead>
<tr>
<th>Buffer</th>
<th>Ore upper value</th>
<th>Ore lower value</th>
<th>Waste upper value</th>
<th>Waste lower value</th>
<th>Limiting value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ore upper value</td>
<td>76871</td>
<td>76871</td>
<td>waste upper value</td>
<td>174835</td>
<td>151742</td>
</tr>
<tr>
<td>ore lower value</td>
<td>76871</td>
<td>76871</td>
<td>waste lower value</td>
<td>174835</td>
<td>349671</td>
</tr>
</tbody>
</table>

[<CR> or 0=no, 1=adjust ore upper buffer value
2=adjust ore lower buffer value
3=adjust waste upper buffer value
4=adjust waste lower buffer value]

Enter if adjustment is needed?

0

Figure 6.6 Sample Mine Sequencing for Period 1
- Input of Buffer Values

PUSHBACK #1 #2 #3 #4

BENCH 22 0 0 0 1
BENCH 23 0 0 0 1
BENCH 24 0 1 1 1
BENCH 25 0 1 1 1
BENCH 26 0 1 1 1
BENCH 27 0 1 1 1
BENCH 28 0 1 1 1
BENCH 29 0 1 1 1
BENCH 30 0 1 1 1

ORE TONNAGE GENERATED: 341836.000
WASTE TONS GENERATED: 900892.063
MINERAL CONTENTS [OUNCES]: 7017.640
STRIPPING RATIO: 2.638

HIT <CR> TO CONTINUE, ENTER 999 TO EXIT =>

0

Figure 6.7 Sample Mine Sequencing for Period 1
- List of Feasible Solution #1
153,084 tons of waste will automatically be put back to bench 30 of pushback 1. This amount of waste is calculated according to the bench 30 stripping ratio. The updated ore and waste tonnages and stripping ratio are reported. Note that if one is not satisfied with the revised result, one has the opportunity to re-enter the amount of ore to be put back. Such a measure provides a powerful tool for trying out different options.

The sample run takes the result as satisfactory. A carriage return is entered. Since the program realizes that the targeted ore amount is achieved, the program proceeds (in figure 6.9) printing the summary of the ore and the waste that are scheduled and the stripping ratio attained. A list of bench percentage that has been mined out is provided for every bench of every pushback. Also printed are the amount of ore, waste, mineral content and the accompanying stripping ratio. This mining schedule is the desired final schedule for the sample run. The items in figure 6.9 are defined basically as in figure 6.7 except that this time the entries at cross positions of pushback name and bench numbers indicate the percentages that the corresponding bench have been mined.

6.6 Summary of Diamond Pit Sequencing Schedule

Following the procedure discussed in previous sections, all six period mining schedules are developed for the Diamond pit according to the client’s tonnage requirement. This is accomplished by series of single period sequencing. These schedules and the relevant items are summarized in table 6.5 and table 6.6 respectively.

In table 6.5, the left most column are bench numbers. Pushback numbers are listed in the third row. The table entries are the numbers indicating in which period the benches of the corresponding pushback are mined. In case a bench is mined in two consecutive periods, the bench percentage that is mined in each period is given in parenthesis. Also, an extra line is inserted for that bench. Note that the
The ore is more than target by 74836.
You can put back this amount 151742.

Stripping ratio of the bench 2.80.
Average grade of the bench 3.00.

As you will put back this much ore 74836.
You will put back this amount of waste also 153064.
The targeted waste is approximately 747600.
Your current total waste is 900892.
Your total waste will be 747608.
The ave. stripping ratio will be 2.80.

If ok, hit <CR>, otherwise enter 1 to change.}}}

Figure 6.8 Sample Mine Sequencing for Period 1
- Solution Adjustment

<table>
<thead>
<tr>
<th>PUSHBACK</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
</tr>
</thead>
<tbody>
<tr>
<td>BENCH 24</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>BENCH 25</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>BENCH 26</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>BENCH 27</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>BENCH 28</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>BENCH 29</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>BENCH 30</td>
<td>50.68</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Ore Tonnage generated : 267000.000
Waste Tons generated : 747608.000
Mineral contents [Ounces] : 5382.390
Stripping Ratio : 2.801

Figure 6.9 Sample Mine Sequencing for Period 1
- List of Final Schedule
Table 6.5 Mining schedule following the client’s tonnage requirement

<table>
<thead>
<tr>
<th>Period mined</th>
<th>Pushback #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Bench #22</td>
<td>air</td>
</tr>
<tr>
<td>Bench #23</td>
<td>air</td>
</tr>
<tr>
<td>Bench #24</td>
<td>1</td>
</tr>
<tr>
<td>Bench #25</td>
<td>1</td>
</tr>
<tr>
<td>Bench #26</td>
<td>1</td>
</tr>
<tr>
<td>Bench #27</td>
<td>1</td>
</tr>
<tr>
<td>Bench #28</td>
<td>1</td>
</tr>
<tr>
<td>Bench #29</td>
<td>1</td>
</tr>
<tr>
<td>Bench #29</td>
<td>3</td>
</tr>
<tr>
<td>Bench #30</td>
<td>1 (50.68)</td>
</tr>
<tr>
<td>Bench #30</td>
<td>2 (49.32)</td>
</tr>
<tr>
<td>Bench #31</td>
<td>2</td>
</tr>
<tr>
<td>Bench #32</td>
<td>2</td>
</tr>
<tr>
<td>Bench #33</td>
<td>2</td>
</tr>
<tr>
<td>Bench #34</td>
<td>3</td>
</tr>
<tr>
<td>Bench #35</td>
<td>3</td>
</tr>
<tr>
<td>Bench #36</td>
<td>3</td>
</tr>
<tr>
<td>Bench #37</td>
<td>3</td>
</tr>
<tr>
<td>Bench #38</td>
<td>3 (40.62)</td>
</tr>
<tr>
<td>Bench #38</td>
<td>4 (59.38)</td>
</tr>
<tr>
<td>Bench #39</td>
<td>4</td>
</tr>
<tr>
<td>Bench #40</td>
<td>4</td>
</tr>
<tr>
<td>Bench #41</td>
<td>4</td>
</tr>
<tr>
<td>Bench #42</td>
<td>4 (31.03)</td>
</tr>
<tr>
<td>Bench #42</td>
<td>5 (68.97)</td>
</tr>
<tr>
<td>Bench #43</td>
<td>5</td>
</tr>
<tr>
<td>Bench #43</td>
<td>6 (19.10)</td>
</tr>
<tr>
<td>Bench #44</td>
<td>5</td>
</tr>
<tr>
<td>Bench #45</td>
<td>5</td>
</tr>
<tr>
<td>Bench #46</td>
<td>5</td>
</tr>
<tr>
<td>Bench #47</td>
<td>6</td>
</tr>
<tr>
<td>Bench #48</td>
<td>6</td>
</tr>
<tr>
<td>Bench #49</td>
<td>6</td>
</tr>
<tr>
<td>Bench #50</td>
<td>6 (100.0)</td>
</tr>
</tbody>
</table>
Table 6.6 Summary of scheduled tonnages

<table>
<thead>
<tr>
<th>Period #</th>
<th>Scheduled ore (tons)</th>
<th>Scheduled waste (tons)</th>
<th>Stripping Ratio</th>
<th>Mineral Contents (ounces)</th>
<th>Average Grades (oz/ton)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>267,000</td>
<td>747,808</td>
<td>2.801</td>
<td>5,382.390</td>
<td>0.0202</td>
</tr>
<tr>
<td>2</td>
<td>833,000</td>
<td>2,002,929</td>
<td>2.404</td>
<td>19,407.939</td>
<td>0.0233</td>
</tr>
<tr>
<td>3</td>
<td>2,000,000</td>
<td>4,876,800</td>
<td>2.438</td>
<td>48,250.797</td>
<td>0.0241</td>
</tr>
<tr>
<td>4</td>
<td>2,000,000</td>
<td>5,612,963</td>
<td>2.806</td>
<td>41,674.770</td>
<td>0.0208</td>
</tr>
<tr>
<td>5</td>
<td>2,000,000</td>
<td>5,618,730</td>
<td>2.809</td>
<td>42,382.750</td>
<td>0.0212</td>
</tr>
<tr>
<td>6</td>
<td>1,432,114</td>
<td>535,430</td>
<td>0.374</td>
<td>35,365.672</td>
<td>0.0247</td>
</tr>
<tr>
<td>Total</td>
<td>8,532,114</td>
<td>19,394,660</td>
<td>2.273</td>
<td>192,464.320</td>
<td>0.0226</td>
</tr>
</tbody>
</table>

benches in pushback 1 range from 24 to 36, the benches in pushback 2 range from 24 to 40, the benches in pushback 3 range from 24 to 46 and the benches in pushback 4 range from 22 to 50.

Tables 6.5 and 6.6 are based on series of single period computer mine sequencing reports. As an illustration, the first period mining schedule computer output is presented in figure 6.10. The rest of 5 period mining schedule computer outputs are given in appendix B.

Most items in figure 6.10 are self explanatory. For example, the title line indicates that figure 6.10 is a mine sequencing report for period 1 of 6, etc. A feasible schedule (solution) is found at the first iteration with an objective value of 2,570,000. The feasible solution and the final mining schedule are listed in the same format as the ones in figures 6.7 and 6.9 and these items are defined the same as in figures 6.7 and 6.9. The optimum solution is given in the same format as that for the feasible solutions. The search for the optimum solution is concluded at iteration number 74. It turns out that the first feasible solution happens to be the optimum solution to the 0-1 program problem in period 1 mine sequencing as well.

The input data for the 0-1 programming mine sequencing program consists of three types of files.
Figure 6.10 Mine sequencing report for period 1
Problem run completed and terminated at iteration: 74

Optimum solution obtained:

Pushback $1$ $2$ $3$ $4$

Bench 22 0 0 0 1
Bench 23 0 0 0 1
Bench 24 0 1 1 1
Bench 25 0 1 1 1
Bench 26 0 1 1 1
Bench 27 0 1 1 1
Bench 28 0 1 1 1
Bench 29 0 1 1 1
Bench 30 0 1 1 1

Ore tonnage generated: 341836.000
Waste tons generated: 900892.063
Mineral contents (ounces): 7017.640
Stripping ratio: 2.636

The optimal obj value is: 2570000

Normal end of the execution

Final mine schedule (after adjustment) for period 1 of 6

Pushback $1$ $2$ $3$ $4$

Bench 24 100.00 0.00 0.00 0.00
Bench 25 100.00 0.00 0.00 0.00
Bench 26 100.00 0.00 0.00 0.00
Bench 27 100.00 0.00 0.00 0.00
Bench 28 100.00 0.00 0.00 0.00
Bench 29 100.00 0.00 0.00 0.00
Bench 30 50.68 0.00 0.00 0.00

Ore tonnage generated: 267000.000
Waste tons generated: 747808.000
Mineral contents (ounces): 5382.390
Stripping ratio: 2.801

Figure 6.10 Mine sequencing report for period 1 (cont’d)
The first input file type consists of one file containing the information on the total number of classes (please refer to the reserve list in figure 6.1, where each class contains materials between two distinct cutoff grades), debug printout option, targeted production tonnage, lower and upper limiting stripping ratios, the cutoff class (grade) and the overriding code. Figure 6.11 lists such a file. On the first line, there are two entries. The first entry (5) tells the program that 5 classes are to be considered. The second entry (0) indicates to the program that there will be no debug print out. The second to sixth lines present, in an order from left to right, the targeted production tonnages in 1000 tons, lower and upper limiting stripping ratios, the cutoff classes (grades) and the overriding codes for six periods with one line for each period.

The second input file type also consists of one file. This second type input file contains only the file names. These file names inform the program where to read in the reserve tonnages and grades from each pushback. Note that the reserves are stored by pushbacks. The total number of file names should conform to the number of pushbacks to be used in mine sequencing. The file names are listed in the order of pushback 1 to pushback n. A list of such a file can be seen in figure 5.13 where four file names are arranged so that each occupies one line.

The third input file type is the reserve file, with one file containing bench by bench tonnage and grade reserves from one pushback. The number of third type input files depends on how many file names are in second type input file, i.e., how many pushbacks are to be used in mine sequencing. For example, four file names in figure 6.12 mean that four pushbacks are to be used in mine sequencing. Thus, the number of the third type input files should be four as well. The pushback 1 reserve file listing is given in figure 6.1. The rest of 3 pushback reserve files are listed in appendix A.
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>2.80</td>
<td>1 0</td>
</tr>
<tr>
<td>267</td>
<td>2.80</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>833</td>
<td>2.46</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>2.39</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>2.83</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>2.82</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>1436</td>
<td>0.37</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6.11** List of Type 1 Input Data File

**Figure 6.12** List of Type 2 Input Data File
6.7 Conclusions on Diamond Pit Sequencing

Applying the 0-1 programming pit sequencing model described in this chapter, a mining schedule is developed for diamond pit based on client’s tonnage requirement. The scheduled ore tonnages follow exactly the client’s tonnages. The scheduled waste tonnages do not match the client’s waste tonnage exactly but they are very close. The model has been built to provide the exact ore tonnages while attempting to force the waste tonnages within satisfactory stripping ratios.

For this study, the tonnage schedule was developed based on maximizing mineral contents in five of the six periods. The schedule, therefore, gives a higher NPV of the mine compared to many other alternative mining schedules with the same cutoff definitions and ore and waste tonnages. It should be recognized, however, that the schedule adjustments may have changed this optimality somewhat. From table 6.6, one can see that the average grades in periods 2 and 3 (0.0233 and 0.0241 respectively) are higher whereas the average grades in periods 4 and 5 are below the overall average grade for 6 periods (0.0226). The average grade of period 6 is the highest because of the high mineral content in the last 7 benches of pushback 4 (please refer to table 6.5). Due to the precedence constraints, these benches of pushback 4 could not be mined earlier. The results show that maximization of mineral contents from periods 2 to 6 has an effect on the mineral contents scheduled in each period.

6.8 Experiences on Application of 0-1 Programming to the Mine Sequencing Problem in this Study

In applying a 0-1 programming mine sequencing model, one should not pursue the ‘optimum’ solution blindly. Usually, a 0-1 programming optimum solution is dependent on problem constraints. One optimum solution could be obtained by a group of tighter constraints. Another optimum solution could be obtained by group of looser constraints. Which ‘optimum’ solution best suits the mine sequencing
problem depends on the constraint setup.

Pushback mine sequencing has an important feature that the mining sequence among the benches in the same pushback has already been decided to be from top to bottom. Further, the mining sequence among the benches on the same elevation is defined to be from the inner pushback to the outer pushback (due to both pit slope stability and higher NPV considerations). A multi-period pushback mine sequencing problem is essentially to determine for each pushback where the mining should stop at the end of each planning period.

With such a pre-requisite, a group of tight production tonnage constraints may easily result in no feasible solutions. On the other hand, an 'optimum' solution obtained under much looser ore and waste tonnage constraints may not be satisfactory in practice. Therefore, there is a trade off between the non feasible solution situation and the undesirable 'optimum' solution situation.

As mentioned in section 5.3.4, the smaller values of those buffer ore and waste tonnages in equations 5.12-5.15 result in tighter ore and waste tonnage constraints. Please note that a group of tighter ore and waste tonnage constraints is usually desirable if and when feasible solutions can be obtained. This is because the smaller the buffer values, the closer the obtained ore and waste tonnages are to the required tonnages. A starting buffer ore tonnage is suggested as half of the maximum single bench ore tonnage among all the formulated benches. A starting buffer waste tonnage is suggested as half of the maximum single bench waste tonnage among all the formulated benches.

Therefore, in the situation of a much looser constraint setup, one may not need an optimum solution because this 'optimum' solution may not satisfy the ore and waste tonnage requirement. Since the post 0-1 programming solution adjustment gives one the chance of adjusting an available mining sequence, one can go to solution adjustment once a satisfactory feasible solution is found. This will save a lot of computer execution time. Note that the search for an optimum solution may take a long time when the problem constraints are loose.
Before each 0-1 program run, it is a good practice to enter a maximum number of iterations allowed. Sometimes, obtaining an optimum solution may cost simply too much time. One should analyze the problem to see if the effort is justified. Some feasible solution may be good enough for solving the original mine sequencing problem. In case only feasible solutions are desired, the mine sequencing problem could become very easy.

The FORTRAN subroutine that solves the standardized 0-1 programming problem is presented in appendix C. The subroutine is a modified version of Kuester and Mize's routine (1973).
CHAPTER 7

CONCLUSIONS

7.1 Conclusions

The objectives of this study were twofold. One was to examine and develop an open pit limit design algorithm that is simple and fast yet consistently generates higher profit than the popular moving cone method. The other was to examine and develop a fast 0-1 programming mine sequencing model.

7.1.1 Pit Design

With respect to the first objective, a new algorithm – the Simulation Oriented Network Flow Algorithm – is developed. The new algorithm adopts a moving cone type approach but is guided by the maximum network flow theory to its maximum potential.

Two theorems have been introduced to guide the development of the new algorithm. The first theorem, the non-optimum theorem, states that, according to network flow theory, the re-allocation problem is the major obstacle which prevents the simulation oriented algorithms from reaching the optimum solution. The traditional multi-pass approach of the moving cone is a re-allocation scheme from the network flow point of view. This multi-pass approach can not solve the re-allocation problem entirely. Without an explicit definition of predecessors and/or successors like the graph theory algorithm, a simulation oriented pit limit design algorithm can not produce an optimum result. Therefore, a simulation oriented pit design algorithm must try to avoid the re-allocation situation entirely besides adopting a multi-pass strategy.
The second theorem, the Lowest Hindrance with Highest Value First Priority Theorem, reveals an "optimum" ore value flow allocation sequence to avoid the re-allocation situation. This second theorem suggests that the value of an ore block with the least number of "hindrance" should have the highest priority to be allocated to its upper wastes. In case there is a "hindrance" tie among the ore blocks, the value of the ore block with the highest value is allocated first. This is the first strategy to reduce the number of re-allocation situations. The second strategy for reducing the possible re-allocation situation is an adoption of Hanson's Lowest Support First Priority approach. Hanson's lowest support first priority theorem defines a flow allocation sequence among waste blocks when the value of a particular ore block is to be allocated. The manner of reducing the number of re-allocation situations by both strategies is clearly demonstrated in chapter four.

In addition to defining a value allocation sequence to avoid at least partly the re-allocation situations, the new algorithm adopts a multi-pass strategy to further overcome some of the re-allocation problems, i.e., to partly carry out some of the re-allocation.

Because of the support definition, the new algorithm can detect the shared contributions to the extent of one sub-cone region, i.e., to the region having the same support values. In other words, the new algorithm is capable of combining several sub-cone regions (the blocks in each of these sub-cone regions have the same support value) either from the same cone or from distinct cones into a whole region and of detecting whether the region is profitable or not.

The new algorithm is thus more precise than the moving cone method, but less precise than the rigorous algorithms. The rigorous algorithms are capable of detecting the shared contributions to the extent of one node (block) because of the explicit definition of predecessors and successors. The moving cone method is incapable of detecting shared contributions because of the cone by cone examining approach. Therefore, any profitable region that is detected by the moving cone method can be detected by the Simulation Oriented Network Flow Algorithm. Sim-
Similarly, any profitable region that is detected by the simulation oriented network flow algorithm can be detected by the rigorous algorithm. The reverse is not true. This is the theoretical reason why the new algorithm can consistently generate higher profit than the moving cone method, yet why the new algorithm is not an optimum one.

Since the new algorithm adopts the moving cone type approach, the new algorithm preserves the advantages of the moving cone method, i.e., simple, easy to be understood and to be programmed.

For one case study, present in chapter four, the new algorithm improved over the popular moving cone method in terms of generated profit (about 1%). The profit generated by the new algorithm is within 99.985% of the optimum solution. Furthermore, the new algorithm is only 2.08 times slower than the extremely fast moving cone method.

Although the above case study result may and will change when other cases are studied, it seems reasonable to conclude, based on the analysis of chapter 4 and the case study result, that the algorithm developed in this study is simple yet consistently generates higher profits than the popular moving cone method. Also, the new algorithm appears to be fast. Based on this author's experience, the solution time applying the new algorithm is well within the practically acceptable range. The difference between the new algorithm developed in this study and the rigorous Lerchs-Grossmann algorithm in terms of generated profit is insignificant.

The new algorithm developed in this study does not over-mine. The new algorithm can thus be used as an effective tool to provide initial surfaces (pits) for the much slower rigorous algorithms.

7.1.2 0-1 Programming Mine Sequencing Model

With respect to the second objective of this study, a multi-period 0-1 programming open pit mine sequencing model was developed incorporating a sub-optimization concept.
This study dealt with a pushback open pit mine sequencing. In the pushback mine sequencing, a series of nested pits can be generated by varying product prices corresponding to a constant cost. If the distance between any two of these series of nested pits exceeds the minimum allowable mining width, these series of nested pits represent "next best" mineralization with the inner pit being the "best" and the outer pit being the "next best". Therefore, these series of nested pits can serve as pushbacks for mine sequencing with the boundary of the outer most pit serving as ultimate pit limit. The pushbacks thus generated attain an important feature, e.g., the inner most pushback has the best mineralization and the mineralization is declining from inner pushbacks to outer pushbacks with the outer most pushback having the lowest mineralization. This inner pushback to outer pushback sequence provides a general direction for pushback mine sequencing so that the NPV of the mining sequence is maximized.

Once the materials on each bench of every pushback are classified between series of cutoff grades, the 0-1 programming mine sequencing model can provide period by period optimized mining schedules. A long term objective is realized by pursuing period by period short term objectives in applying the sub-optimization concept.

The inputs to the model are the reserves between series of cutoff grades, the production capacities defined by the targeted ore and the desired stripping ratios for each planning period. The output from the model are the schedules regarding the period in which a particular bench is mined. Also reported are each period's actual attained ore and waste tonnages, the stripping ratio and the average grade.

So far, few applications of 0-1 programming to mine sequencing problem are reported. This is probably because 1) lack of an algorithm that is capable of solving large scale 0-1 programming problems, say, the problems involving hundreds to thousands of decision variables, in a practically acceptable time; 2) potentially large problem size involved in mine production scheduling; 3) a multi-period mine production scheduling problem which is formulated into one integreted large scale
0-1 programming problem is virtually unsolvable.

To use the 0-1 programming model as an effective and rapid mine scheduling tool, the following had to be accomplished during this study:

1) Sub-optimization objectives are introduced. These objectives allow pursuing high mineral contents and/or any other frequently encountered mine sequencing objectives subjected to ore, waste tonnage and the precedence constraints in series of individual planning period. A long term objective is thus achieved by pursuing a series of short term period by period objectives. A large problem is reduced to a series of small but related problems.

2) The approaches of limiting number of decision variables to within 50 in each period problem formulation are suggested. Such a problem formulation is usually within quickly solvable ranges.

3) A new precedence constraint or sequential requirement (SR) formulation is used. As a result, the number of constraints needed for the SR is dramatically reduced (more than 80%). This in turn reduces the solution time.

4) Post 0-1 programming single period schedule adjustment is incorporated as an integral part of the 0-1 programming mine sequencing model. This post 0-1 programming schedule adjustment enables the scheduler to have final control on the mining schedule after 0-1 programming provides scheduling guidelines.

5) Restarting capability after any previously scheduled periods is also incorporated as an integral part of the 0-1 programming mine sequencing model. This can save a lot of frustration when several alternative mining sequences are to be tried out.

With the reserves in each pushback available, and using the 0-1 programming model described here, the desired mining schedule can be worked out almost instantaneously by simply defining the targeted amount of ore, the stripping ratios and the items to be optimized for each period.

A fast 0-1 programming mine sequencing model presents great advantages over simple trial and error as well as the simple spreadsheet approach. This is
because of the capability of systematically searching both feasible and optimum solutions subjected to multi-item joint constraints.

In chapter 5, the process of mine sequencing is demonstrated using the first period Diamond pit mine sequencing case. This provides a perspective on how to apply the 0-1 programming model developed in this study. Applying the 0-1 programming model, the 6-period Diamond pit mine sequencing required 30 minutes on a VAX 8600. Virtually every time one hits the last keystroke of input, a schedule is displayed on the screen. All the single period mining schedules are developed in no more than 10 minutes. The model is thus practical for application.

Even if one is stuck at one period, i.e., one can not find any feasible solution during current period: one can re-start the current period mine sequencing after relaxing the particular constraint which prevented attaining a feasible solution. One also can re-start at any previously scheduled period.

An important observation on the new model based on the Diamond pit sequencing indicates that maximization of mineral contents in five of six period's mine sequencing does increase the average grades in early periods. This is even after the post 0-1 programming solution adjustment.

In this study, the maximization of mineral content has been selected as the sub-optimization objective for most of the planning periods. In this author's opinion, the maximization of mineral contents should always be the sub-optimization objective for long range mine planning whenever it is permissible. This is because higher mineral contents in early periods bring higher NPV to the mining sequence for any given cutoff strategy.

As such, the 0-1 programming model developed in this study provides some systematic feature to mine sequencing. Finally, by defining loose constraints in the 0-1 programming formulation of the mine sequencing problem, one may be able to apply the model to develop multi-period mining sequences automatically to gain a
first look at the possible sequences.

Final Remarks:

Although the decision variable of mine sequencing of this study is defined as a whole bench, the decision variable of pit sequencing is not limited to one bench. One can divide a whole bench into efficient working slices, define the precedence relationship among these working slices, then carry out pit sequencing such as the one discussed in this study. In this sense, the scheduling model presented can be applied to many types of mine production scheduling problems. With increased availability of Interactive Graphic Planner, the 0-1 programming model presented in this study could be very interesting.

The grade balancing constraint is not included in the 0-1 programming model for two reasons. One is that adding the grade balancing constraint may easily deny some feasible solutions. The other is that, in long range mine sequencing, higher mineral contents should be pursued for each planning period. Therefore, the grade balancing constraint should give way to maximization of mineral contents. Nevertheless, the grade balancing constraint can be added to the model whenever it is necessary. Overall, the 0-1 programming model described in this study adds some systematic feature to the mine sequencing problem.

7.2 Future Research Recommendations

While this author was finishing this study, other pit design algorithms either have appeared or are scheduled to appear in publications. One is by P. Huttatosal and R. E. Cameron (1989). The other is by Y. Zhao and Y.C. Kim. Huttatosal and Cameron's algorithm is an alteration of the Lerchs-Grossmann's tree algorithm. Their algorithm includes only the ore blocks in tree transformations while keeping track of waste values in separate arrays. Zhao and Kim's algorithm is also a variation of Lerchs-Grossmann's tree approach. The algorithm generates arcs only from ore blocks (nodes) to waste blocks (nodes). An arc between an ore block and a waste
block is generated only when the ore block belongs to a positive chain (cumulative mass > 0) and when the waste block belongs to a negative chain (cumulative mass < 0). Detailed discussions on both Huttatosol-Cameron's algorithm and Zhao-Kim's algorithm can be found in the listed bibliography.

Since pit design is such an important yet time consuming job in open pit mine planning, researches on this topic are bound to continue until a fast optimum algorithm prevails. So far, among the numerous pit design algorithms, some sacrifices optimum for speed (heuristic algorithms), while others sacrifices speed for optimum (rigorous algorithms). The SNF algorithm developed in this study bridges between the heuristic and the rigorous algorithms.

The recommendation for future research from this study regarding pit design algorithms is to continue the search for a fast, simple and optimum pit design algorithm. It seems that such an algorithm can only be obtained from an ingenious adoption of graph theory results. The recommendation for future research from this study also introduces one practical problem affecting pit design algorithms as follows.

In theory, the simulation oriented network flow (SNF) pit design algorithm generates higher profit than the moving cone method (MC). However, there may be situations when the SNF algorithm does generate less profit than the MC because of both over mining and undermining, in comparison to the MC pit. This phenomenon is due to the coning approximation of a pit boundary.

The fundamental reason is that all the materials within an upper cone are not necessarily within a lower cone even if the ore block at the apex of the upper cone is within the lower cone. In other words, not all the restricting blocks of an upper ore block fall into the restricting blocks of the lower ore block. This is referred to as the coning effect in this study. The coning effect gives the moving cone
1) Basic 4-level cone removal pattern (in cross section):

![Diagram of Basic 4-level cone removal pattern](image)

Figure 7.1 Basic 2-D Cone Removal Pattern

2) Situation:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
</tr>
<tr>
<td>(-2)</td>
<td>(-2)</td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
</tr>
<tr>
<td>-1</td>
<td>+2</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
</tr>
<tr>
<td>0</td>
<td>+2</td>
<td>+1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 7.2 Block in Upper Cone but not in Lower Cone
method a great advantage. If the upper ore block is not minable, a lower ore block may become minable with only the profit portion of the upper non-profitable cone being transferred to the lower ore block.

A (2-D) example showing the situation is as follows.

Figure 7.1 shows a basic 4-level cone removal pattern in cross section. In figure 7.1, the bottom block is the cone base block. Figure 7.2 shows the value matrix. In figure 7.2, the numerals in parenthesis in each block are the block economic values. Other numerals represent the support value of the corresponding blocks. The letters 'A', 'B' and so on represent block identifications. (Both figure 7.1 and 7.2 are shown on the previous page).

Two cones are generated - an upper cone for block L and a lower cone for block R. Block F is in the upper cone but not in the lower cone due to particular slope geometry. (Note that the blocks delineated by bold lines represent the joint portion of the two cones generated above the ore blocks L and R)

3) Proceed with mining as follows (blocks with 0.0 values are excluded from calculations):

a) Simulation Oriented Network Flow:

1. Ore block L is first examined. +2.0 units of value are allocated to waste block F. The values at waste block F and ore block L are cleared as 0.0.
2. Ore block R is next examined. +1.0 units of value are allocated to waste block G. This leaves a 0 value at ore block R and -1.0 value at waste block G.
3. Since there is negative value above both ore blocks L and R, both ore blocks are deleted as non-profitable. The generated profit is 0.0.

b) Moving Cone:

1. Ore block L is first examined. The summation value of the cone above ore block L is:
2. Value(F) + Value(G) + Value(L) = -2.0 - 2.0 + 2.0 = -2.0 The cone is non-profitable. The value of the blocks within the cone above ore block L is left unchanged.

3. Ore block R is next examined. The summation value of the cone above ore block R is:

4. Value(G) + Value(L) + Value(R) = -2.0 + 2.0 + 1.0 = +1.0 The cone is profitable. The blocks within the cone of ore block R are mined with net profit of +1.0. Note that the waste block F is left unmined.

Thus, in the above situation, the simulation oriented network flow algorithm undermines compared to the moving cone result. Now, let the values of ore blocks L and R be increased to +3.0 and +2.0 units respectively. Proceeding with mining in the same sequence as before for both SNF and MC, SNF generates net profit of +1.0 unit with both ore blocks mined. MC generates net profit of +3.0 also with both ore blocks mined. The profit difference between the two methods is because of waste block F which is mined by the SNF but not mined by the MC. This time the SNF over mined compared to the MC.

This author believes that the coning effect has an impact on rigorous algorithms as well. This is because the rigorous algorithms consider one block as one unit, as it is the case with SNF, whereas MC considers one cone as one unit, thereby performing better in particular situations shown above.

Figure 7.3 shows an equivalent network of figure 7.2. Please note that block F is a successor of block L but not a successor of block R in network construction. When the figure 7.3 network reaches maximum flow, the residual capacity between the source (S) and the positive valued nodes (L) and (R) is 1 unit. The generated profit from figure 7.3 network is the same as the SNF result. This type of coning problem becomes more frequent, 1) when a non-zero base radius is used to allow for the minimum mining width (see figure 7.4), 2) when a block is either removed
or left in place based on whether the center is inside or outside the cone while the cone is generated at the mid point of toe line of the cone base block.

In figure 7.4, block B is in the cone of block A. But, mining of block A does not require removing the shaded area whereas mining of block B does require removing the shaded area. In figure 7.5, block B is not a restricting block if the block B is either removed or left in place based on whether the mid point of toe line is inside or outside the cone. But, the block is a restricting block if the block B is either removed or left in place based on whether the center is inside or outside the cone. Both cases could cause a discrepancy in support volume at cone bottom area and could create a problem.

Figure 7.3 An Equivalent Network to Figure 7.2
Figure 7.4 Coning inconsistency resulting from non-zero base radius

Figure 7.5 Coning inconsistency resulting from measuring blocks at center while generating cone at the mid point of the base block
A preliminary solution to the above problem is as follows:

1) Check the coning pattern and make sure that the above situations will not happen. This may prove to be difficult to do.

2) If it is possible, define a zero (or very small) base radius at the toe of cone base block. The main effect of defining a base radius is to essentially lower the cone apex to somewhere below the cone base block causing more blocks fall inside the cone. Since a cone with a base radius can cause the SNF pit design and other rigorous algorithms "malfunction", one can design a pit without a base radius, then adjust the final pit to insure the minimum mining width.

3) Measure the blocks consistently at the toe elevation.

Final remarks:

1) Whether to choose a zero cone base radius or a non-zero cone base radius is still a subjective decision.

2) The non-zero base radius coning pattern makes it tougher for the cone base area to insure a minimum mining width. This results in a smoother ultimate pit than the zero-based coning pit. Fortunately for the moving cone method, this does not create a problem. For the SNF pit design and other rigorous algorithms however, it is a problem as shown in the above example.

3) Since one does not have a prior knowledge of which block would be the pit bottom block, the best way is to relax the cone in bottom area and let the SNF and other rigorous algorithms to serve at their best potentials. Then, one adjusts the pit bottoms.

Different coning techniques can result in quite different outcomes in terms of both the generated profit and the execution time. It would be interesting to examine how different coning patterns affect various pit design algorithms including the rigorous algorithms.

The mine sequencing model developed in this study proposed a systematic way to develop mining sequences. Currently, a mining sequence is developed ac-
cording to one fixed cutoff grade strategy. Rather, an "optimum" mining sequence can be selected among several mining sequences which have been developed based on fixed cutoff grades. It is obvious that if the allowable number of decision variables can be increased significantly, say, to thousands, and the solution time can be reduced to a practically acceptable range, a mine sequencing problem can be solved using a much smaller volume as a decision variable instead of one bench. Also, the classification of "ore" and "waste" under a pre-defined cutoff grade could be avoided. A multi-period mine sequencing problem could even be formulated into an integral 0-1 programming problem and being solved. Thus, the advancement in the state of art of mine sequencing is dependent on the advancement in the state of art of operations research theory, particularly the development of a fast algorithm capable of solving a large scale 0-1 programming problem with high density matrix.
APPENDIX A

A Numerical Example for the Application of
the Simulation Oriented Network Flow Algorithm
The following two dimensional numerical example shows in more detail the calculations and matrix manipulations performed by the simulation oriented network flow algorithm to determine the economic open pit limits. In this case, blocks are equi-dimensional and slope angles constant at 45 degrees. The example is hypothetical and is used here only for illustration purposes.

The following notation is used to identify block subsets and specific matrix entries throughout the example:

1) \( V \) = Value matrix;
2) \( S \) = Support matrix based on support provided by ore blocks;
3) \( H \) = Hindrance matrix based on number of waste regions above ore blocks;
4) \( O \) = Ore block processing sequence index;
5) \( N \) = Temporary non-minable ore block index array; and
6) \( O(k) \) = Ore block in question; and
7) \( (rk) \) = The set of restricting blocks lying "above" block \( O(k) \); and
8) \( (i,j) \) = The block located at level \( i \) and column \( j \).

1. Check and mine out profitable cones with only the cone bottom ore block being considered.

Based on the original value matrix and a top-down search, the ore block processing sequence is defined as \( O(1) = (3,3) \), \( O(2) = (3,5) \) and \( O(3) = (5,5) \).

<table>
<thead>
<tr>
<th>Table A.1 Original Value Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

Beginning with \( O(1) = (3,3) \), it is found that \( V(O(1)) = +6.0 < V(r1) = -8.0 \). Thus, the ore block at \( (3,3) \) is not minable. Similarly, since \( V(O(2)) = +7.0 \)
\(< V(r_2) = -8.0 \text{ and } V(O(3)) = +9.0 < V(r_3) = -22.0, \text{ all the ore blocks are not minable when only the ore block at cone bottom is considered.} \)

2. Create a support matrix.

Based on the value matrix, a support matrix is created as described in section 4.3.3. Note that each support matrix entry holds the number of positive value (support) blocks of the value matrix whose removal necessitates the removal of the restricting block in question. For instance, the removal of restricting block \( V(2,3) \) is necessary if either ore block \( V(3,3) \) or \( V(5,5) \) are to be removed, but not if block \( V(3,5) \) is to be removed, therefore \( S(2,3) = -2 \). Also note that those blocks with positive value matrix entries (\( V(3,3), V(3,5), \) and \( V(5,5) \)) have their support matrix entries stored as positive values to indicate that they were originally positive value blocks. Finally, since the ore block \( (3,3) \) is supported both by itself and ore block \( (5,5) \), the support of block \( (3,3) \) is assigned as 2.

Table A.2 Initial Support Matrix

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>+1</td>
<td></td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Table A.3 Support Matrix After Setup

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>+2</td>
<td>-1</td>
<td>+2</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

3. Search and delete the ore block that does not support their own stripping.
Since the ore blocks with support great than 1 are supported by some other ore blocks or some other profitable combinations, only the ore block with support value of 1 can be checked. In the support matrix, only the ore block \((5,5)\) has \(S(5,5) = 1\). The ore block \((5,5)\) is thus checked, where \(V(O(3)) = +9\) and the \(V(r3)\) in least support, i.e., \(S(rk) = -1\), \(V(S(rk)=-1) = -10\). Therefore, the ore block \((5,5)\) does not support its least support and should be deleted from further consideration. At the end of part 2 of the algorithm, an updated value matrix is obtained.

<table>
<thead>
<tr>
<th>Table A.4 Updated Value Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

To illustrate the solution process, for this particular example problem, the ore block at \((5,5)\) is retained. In actual application of the algorithm, the ore block at \((5,5)\) should be deleted permanently from further consideration, i.e., the updated value matrix (table A.4) should be used instead of the initial value matrix (table A.1).

4. Define ore block processing sequence.

An hindrance matrix is set up such that the hindrance of waste blocks (negative valued blocks) is assigned as zero and the hindrance of ore blocks (positive valued blocks) is assigned as the ore block sequence number in a top-down search.

<table>
<thead>
<tr>
<th>Table A.5 Initial Hindrance Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
Table A.6 Hindrance Matrix After Set Up

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4</td>
<td>-4</td>
<td>-6</td>
<td>-6</td>
<td>-5</td>
<td>-5</td>
<td>-3</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-4</td>
<td>-6</td>
<td>-5</td>
<td>-3</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>+1</td>
<td>-3</td>
<td>+2</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

After setting up the hindrance matrix as discussed in section 4.3.3, the ore block processing sequence is defined as: \( O(1) = (3,5) \), \( O(2) = (3,3) \) and \( O(3) = (5,5) \). This is because \( H(3,3) = H(3,5) = 2 > H(5,5) = 3 \). Note that \( V(3,5) = 7 > V(3,3) = 6 \) therefore, the ore block \((3,5)\) has higher priority to be considered first than the ore block at \((3,3)\) though \( H(3,3) = H(3,5) = 2 \). To illustrate part three of the algorithm, the ore block at \((3,3)\) is considered. The hindrance of ore block \((3,3)\) is found by doing the following:

Sort the \( S(i,j) = S(r_1) \) according to decreasing support values. A sorted sequence is found to be: \( S(1,1) = S(1,2) = S(2,2) = S(2,3) = -4 > S(1,3) = S(1,4) = S(1,5) = S(2,4) = -6 \). The hindrance of the ore block = number of value changes among sorted \( S(r_1) \)'s + 1. Therefore, \( H(3,3) = 1 + 1 = 2 \).

At the end of part three of the algorithm, the support matrix is re-initiated to zeros.

5. Create a support matrix for part four of the algorithm.

Pick \( O(1) = (3,5) \), assign \( S(3,5) = V(3,5)/I \) average waste value \( I = 7.0/1.0 = 7 \). Pick \( O(2) = (3,3) \), assign \( S(3,3) = V(3,3)/I \) average waste value \( I = 6.0/1.0 = 6 \). Pick \( O(3) = (5,5) \), assign \( S(5,5) = V(5,5)/I \) average waste value \( I = 9.0/1.0 = 9 \). Since no more ore block exists, the search of ore blocks stops. The rest entries of matrix \( S \) is defined as:

\[
S(rk) = \sum_{all O(k)} -S(k)
\]  \hspace{1cm} (1)

For example, \( S(1,4) = - S(3,3) - S(3,5) = -6 - 7 = -13 \).
Table A.7 Initial Support Matrix for Part Four of the Algorithm

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>+6</td>
<td>0</td>
<td>+7</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+9</td>
<td>0</td>
</tr>
</tbody>
</table>

1  2  3  4  5  6  7  8  9

Table A.8 Support Matrix for Part Four of the Algorithm

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-15</td>
<td>-15</td>
<td>-22</td>
<td>-22</td>
<td>-16</td>
<td>-16</td>
<td>-9</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>+15</td>
<td>-9</td>
<td>-9</td>
<td>+16</td>
<td>-9</td>
<td>-9</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-9</td>
<td>-9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

1  2  3  4  5  6  7  8  9

6. Allocate the value of the first ore block: O(1) = (3,5).
Block value to be allocated => V(O(1)) = V(3,3) = +7.0.
The restricting block values V(r1) are now summed in subsets according to their support matrix entries.

Restricting Block Totals

\[ \sum_{i=1}^{n} V(i,j) \in (S(i,j) = -16) = -4.0 \]
\[ \sum_{i=1}^{n} V(i,j) \in (S(i,j) = -22) = -4.0 \]

Grand Total = -8.0

Because the value (+7.0) available at (3,3) can not completely support the restricting block grand total of -8.0, 7.0 units are the maximum flow to be allocated. The flow allocation takes place according to the Lowest Support First-Priority principle outlined in 4.3.3.
Lowest-Support-First-Flow Allocation

Allocate +4.0 to $S(r1) = -16$ blocks
Allocate +3.0 to $S(r1) = -22$ blocks

+7.0 = maximum flow

Note that restricting blocks with $S(rk) = -22$ cannot be completely supported. Value is therefore allocated in this region from left to right in a top-down fashion. This will leave -1 in (1,5).

The value matrix is now updated to reflect these allocations while the support matrix remains unchanged.

Table A.9 Updated Value Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>-1</td>
<td>+6</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>+9</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

7. Continue, $O(2) = (3,3)$ is the next positive value block to be processed.
Block value to be allocated $==$ $V(O(2)) = V(3,3) = +6.0$.

Restricting Block Total

$$\sum_{allr2} V(i,j) \in (S(i,j) = -15) = -4.0$$
$$\sum_{allr2} V(i,j) \in (S(i,j) = -22) = -1.0$$

Grand Total -5.0

This time the restricting blocks can be completely supported, therefore the maximum value to be allocated equals the total restricting values, leaving +1.0 unit remain in $V(3,5)$. The value matrix again updated.
Table A.10 Updated Value Matrix

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>+9</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

8. Continue, \( O(3) = (5,5) \) is the next positive value block to be processed. Block value to be allocated \( \Rightarrow V(O(3)) = V(5,5) = +9.0 \).

Restricting Block Total

\[ \sum_{i \in r3} V(i,j) \in (S(i,j) = -9) = -10.0 \]

Grand Total -10.0

Since the grand total of the restricting waste values is -10.0 unit which exceed the ore block value available at (5,5). The maximum flow would be the ore block value which is 9.0 units. The allocation of these 9 units ore value follows the Lowest-Support -First-Flow principle.

Lowest-Support-First-Flow Allocation

Allocate +9.0 to \( S(r3) = -9 \) blocks

+9.0 = maximum flow

Note that restricting blocks with \( S(r3) = -9 \) can not be completely supported. Value is therefore allocated in this region from left to right in a top-down fashion. This will leave -1 in (1,9).

The value matrix is now updated to reflect these allocations while the support matrix remains unchanged.
9. At this stage, all the values of the ore blocks are allocated. The algorithm goes on checking whether there is any negative values overlying above ore blocks. It is found that in table A.11, -1 unit is overlying only above the ore block (5,5). Thus, the ore block (5,5) is put into a temporary non-minable set. \( N(1) = (5,5) \). The value matrix is updated. This is the end of iteration 1 of pass 1.

Note that the ore block at (5,5) has an asterisk meaning that the ore block is in temporary non-minable set. Since the removed ore block (5,5) may have helped to offset the restricting waste values of ore block (3,3) and ore block (3,5), a new iteration is needed to check on the two ore blocks. A new iteration will begin with an updated support matrix.
Table A.13 Updated Support Matrix for Part 4 of the Algorithm

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-6</td>
<td>-6</td>
<td>-13</td>
<td>-13</td>
<td>-7</td>
<td>-7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-6</td>
<td>-6</td>
<td>-13</td>
<td>-7</td>
<td>-7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>+6</td>
<td>0</td>
<td>+7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

10. Allocate the value of the first ore block: \( O(1) = (3,5) \).

Block value to be allocated \( \Rightarrow V(O(1)) = V(3,3) = +7.0 \).

The restricting block values \( V(r1) \) are now summed in subsets according to their support matrix entries.

Restricting Block Totals

\[
\sum_{all r1} V(i,j) \in (S(i,j) = -7) = -4.0
\]

\[
\sum_{all r1} V(i,j) \in (S(i,j) = -13) = -4.0
\]

Grand Total = -8.0

Because the value (+7.0) available at (3,3) cannot completely support the restricting block grand total of -8.0, 7.0 units are the maximum flow to be allocated. The flow allocation takes place according to the lowest support First-Priority principle outlined in 4.3.3.

Lowest-Support-First-Flow Allocation

Allocate +4.0 to \( S(r1) = -7 \) blocks
Allocate +3.0 to \( S(r1) = -13 \) blocks

+7.0 = maximum flow

Note that restricting blocks with \( S(rk) = -13 \) cannot be completely supported. Value is therefore allocated in this region from left to right in a top-down
fashion. This will leave -1 in (1,5).

The value matrix is now updated to reflect these allocations while the support matrix remains unchanged.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>S</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>-1</td>
<td>+6</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Table A.14 Updated Value Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>+9*</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

11. Continue, O(2) = (3,3) is the next positive value block to be processed. Block value to be allocated ==> \( V(O(2)) = V(3,3) = +6.0 \).

Restricting Block Total

\[
\sum_{all \, r2} V(i, j) \in (S(i, j) = -6) = -4.0 \\
\sum_{all \, r2} V(i, j) \in (S(i, j) = -13) = -1.0 \\
\]

Grand Total -5.0

This time the restricting blocks can be completely supported, therefore the maximum value to be allocated equals the total restricting values, leaving +1.0 unit remain in \( V(3, 5) \). The value matrix again updated.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>+9*</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

Table A.15 Updated Value Matrix
12. At this stage, all the ore blocks are checked. Note that ore block (5,5) is in the non-minable set. Since there is no negative values above the positive valued ore blocks, all the ore blocks together with their restricting waste blocks are mined and the surface is updated. This is the end of second iteration and also the end of first pass. One pass is finished whenever a permanent surface is obtained.

13. A second pass is started with all the ore block (5,5) being brought back to the possible solution set. Record the number of ore blocks being brought back, which is 1.

<table>
<thead>
<tr>
<th>Table A.16 Updated Support Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Continue, $O(1) = (5,5)$ is the next positive value block to be processed. Block value to be allocated $\Rightarrow V(O(1)) = V(5,5) = +9.0$.

Restricting Block Total

$$\sum_{all} V(i,j) \in (S(i,j) = -9) = -10.0$$

Grand Total -10.0

Since the grand total of the restricting waste values is -10.0 unit which exceed the ore block value available at (5,5). The maximum flow would be the ore block value which is 9.0 units. The allocation of these 9 units ore value follows the Lowest-Support-First-Flow principle.
Lowest-Support-First-Flow Allocation

Allocate +9.0 to S(r1) = -9 blocks

+9.0 = maximum flow

Note that restricting blocks with S(r1) = -9 can not be completely supported. Value is therefore allocated in this region from left to right in a top-down fashion. This will leave -1 in (1,9).

The value matrix is now updated to reflect these allocations while the support matrix remains unchanged.

Table A.17 Updated Value Matrix

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

14. At this stage, all the ore blocks are checked. Since there is negative value above the positive valued ore block (5,5), the ore block (5,5) is again put back to the non-minable set. Since the only ore block being brought to the possible solution set is put back to the non-minable set, there is no improvement in the second pass over the first pass. The algorithm ends with the ultimate pit limit defined by mining block (3,3) and (3,5) together with their restricting blocks: (1,1...7), (2,2...6), (3,3) and (3,5). The pit design net profit is 1 unit.
APPENDIX B

Diamond Pit Reserve Listing (Pushbacks 2 - 4)
**RESERVES FOR SECOND PUSHBACK (PIT 002)**

<table>
<thead>
<tr>
<th>#</th>
<th>TONS</th>
<th>OZ/TON</th>
<th>TONS</th>
<th>OZ/TON</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>5</td>
<td>0.6435934E+02</td>
<td>0.1537002E-02</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>0.1114922E+02</td>
<td>0.2489723E-02</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td>0.2857195E+02</td>
<td>0.2320528E-02</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>0.5397720E+03</td>
<td>0.1909999E-03</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td>0.4705204E+02</td>
<td>0.1687715E-02</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>0.8333332E+02</td>
<td>0.1464220E-01</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>0.1723598E+03</td>
<td>0.2000000E-02</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>5</td>
<td>0.1013595E+03</td>
<td>0.4059942E-02</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>0.1674243E+03</td>
<td>0.1565533E-01</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>0.4018357E+02</td>
<td>0.1509999E-01</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>3</td>
<td>0.3522755E+03</td>
<td>0.1978295E-01</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>4</td>
<td>0.1699448E+03</td>
<td>0.3311338E-01</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>5</td>
<td>0.1924793E+03</td>
<td>0.4103177E-02</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>0.5562495E+03</td>
<td>0.1926464E-01</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>0.3787879E+03</td>
<td>0.2097511E-01</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>0.4844845E+03</td>
<td>0.3547665E-01</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>5</td>
<td>0.5064884E+03</td>
<td>0.4050000E-02</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>2</td>
<td>0.8368100E+03</td>
<td>0.1952666E-01</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>3</td>
<td>0.3825775E+03</td>
<td>0.2687126E-01</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>4</td>
<td>0.3150981E+03</td>
<td>0.3928746E-02</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>5</td>
<td>0.1183713E+03</td>
<td>0.1148342E-01</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>0.1100599E+03</td>
<td>0.2097296E-02</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>0.4467857E+03</td>
<td>0.2556719E-02</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>3</td>
<td>0.1215910E+03</td>
<td>0.1450212E-01</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>0.1035983E+03</td>
<td>0.1909999E-01</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>0.3422531E+03</td>
<td>0.3648972E-02</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>0.1359484E+03</td>
<td>0.1318446E-01</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>2</td>
<td>0.1247158E+03</td>
<td>0.2071727E-01</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>3</td>
<td>0.2063334E+03</td>
<td>0.2172726E-01</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>4</td>
<td>0.2941265E+03</td>
<td>0.2245519E-02</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>5</td>
<td>0.2714017E+03</td>
<td>0.3571001E-01</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>0.1678922E+03</td>
<td>0.1942006E-02</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>0.1715803E+03</td>
<td>0.2293935E-02</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>3</td>
<td>0.3750000E+03</td>
<td>0.3649265E-02</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>4</td>
<td>0.2418506E+03</td>
<td>0.4095462E-02</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>5</td>
<td>0.3219998E+03</td>
<td>0.4829786E-02</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>0.1961723E+03</td>
<td>0.2196420E-01</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>2</td>
<td>0.3020394E+03</td>
<td>0.2405940E-02</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>3</td>
<td>0.1978407E+03</td>
<td>0.3095905E-02</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>4</td>
<td>0.1321062E+03</td>
<td>0.1694113E-01</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>5</td>
<td>0.2481714E+03</td>
<td>0.3560004E-02</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>2</td>
<td>0.1893994E+03</td>
<td>0.2472289E-01</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>3</td>
<td>0.1850189E+03</td>
<td>0.4084841E-02</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>0.4228410E+03</td>
<td>0.1448396E-01</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>5</td>
<td>0.2213635E+03</td>
<td>0.2250009E-01</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>2</td>
<td>0.7575750E+03</td>
<td>0.2660216E-02</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>3</td>
<td>0.1226704E+03</td>
<td>0.4389983E-02</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>4</td>
<td>0.1918586E+03</td>
<td>0.1147175E-01</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>5</td>
<td>0.1992355E+03</td>
<td>0.2414285E-01</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>0.1325757E+03</td>
<td>0.2700000E-01</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>2</td>
<td>0.1092355E+03</td>
<td>0.3875221E-01</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>3</td>
<td>0.6052855E+03</td>
<td>0.4157411E-02</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>4</td>
<td>0.1890151E+03</td>
<td>0.2178382E-01</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>5</td>
<td>0.1267813E+03</td>
<td>0.2428490E-01</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>0.1147273E+03</td>
<td>0.2802248E-01</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>2</td>
<td>0.6208335E+03</td>
<td>0.4293587E-01</td>
<td></td>
</tr>
</tbody>
</table>

**Figure B.1 Reserve Listing of Pushback 2**
"RESERVES FOR THIRD PUSHBACK (PIT OH4)"

**TOTAL**

- **OF CLASS**: 5
- **OF MINERAL CLASS**: 4

**CLASS 5 IS WASTE**

<table>
<thead>
<tr>
<th>Case</th>
<th>Bench</th>
<th>Total Tons</th>
<th>Au-Oz/Ton</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>1</td>
<td>0.8906213E+02</td>
<td>0.19119107E+01</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>0.4272625E+02</td>
<td>0.47120755E+01</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>0.4938449E+02</td>
<td>0.39359472E+02</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>0.3595462E+02</td>
<td>0.38799472E+02</td>
</tr>
<tr>
<td>26</td>
<td>2</td>
<td>0.3670516E+02</td>
<td>0.36794472E+02</td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td>0.3807948E+02</td>
<td>0.38079472E+02</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>0.3175560E+02</td>
<td>0.13607873E+01</td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>0.3567336E+02</td>
<td>0.36555133E+01</td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td>0.3109551E+02</td>
<td>0.08250005E+02</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>0.5661815E+02</td>
<td>0.18555133E+01</td>
</tr>
<tr>
<td>28</td>
<td>2</td>
<td>0.4702625E+02</td>
<td>0.47120755E+01</td>
</tr>
</tbody>
</table>

Figure B.2 Reserve Listing of Pushback 3
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.14382039E+02</td>
<td>0.10158546E-01</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.13371201E+03</td>
<td>0.19099999E-01</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.18181818E+01</td>
<td>0.25160375E-01</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.26540094E+03</td>
<td>0.28081170E-03</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.14753789E+02</td>
<td>0.90703182E-02</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.16852228E+03</td>
<td>0.22066687E-01</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.36383636E+01</td>
<td>0.29717581E-01</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.20962108E+03</td>
<td>0.38401219E-02</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.35600680E+02</td>
<td>0.12759033E-01</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.28115509E+03</td>
<td>0.24205213E-01</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.15151515E+02</td>
<td>0.38000001E-01</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.18628888E+03</td>
<td>0.23321232E-02</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.20961951E+03</td>
<td>0.15824429E-01</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.21988529E+03</td>
<td>0.10099999E-01</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.36393939E+01</td>
<td>0.20900000E-01</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.18930830E+01</td>
<td>0.23656565E-01</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0.15150900E+02</td>
<td>0.39744848E-02</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0.10090909E+02</td>
<td>0.14692929E-01</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0.19327473E+03</td>
<td>0.24044379E-01</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.24696070E+01</td>
<td>0.25600003E-01</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0.70255152E+02</td>
<td>0.41703732E-02</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0.14507580E+01</td>
<td>0.16521692E-01</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>0.23448994E+03</td>
<td>0.23500003E-01</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0.15151515E+02</td>
<td>0.25345691E-01</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0.15151515E+02</td>
<td>0.39744848E-02</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0.19327473E+03</td>
<td>0.24044379E-01</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>0.24696070E+01</td>
<td>0.25600003E-01</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>0.14507580E+01</td>
<td>0.16521692E-01</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>0.36383636E+01</td>
<td>0.29717581E-01</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0.15151515E+02</td>
<td>0.25345691E-01</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>0.15151515E+02</td>
<td>0.39744848E-02</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>0.19327473E+03</td>
<td>0.24044379E-01</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>0.24696070E+01</td>
<td>0.25600003E-01</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>0.14507580E+01</td>
<td>0.16521692E-01</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>0.23448994E+03</td>
<td>0.23500003E-01</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>0.15151515E+02</td>
<td>0.25345691E-01</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>0.15151515E+02</td>
<td>0.39744848E-02</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>0.19327473E+03</td>
<td>0.24044379E-01</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>0.24696070E+01</td>
<td>0.25600003E-01</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0.14507580E+01</td>
<td>0.16521692E-01</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>0.23448994E+03</td>
<td>0.23500003E-01</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>0.15151515E+02</td>
<td>0.25345691E-01</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>0.15151515E+02</td>
<td>0.39744848E-02</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>0.19327473E+03</td>
<td>0.24044379E-01</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>0.24696070E+01</td>
<td>0.25600003E-01</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>0.14507580E+01</td>
<td>0.16521692E-01</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>0.23448994E+03</td>
<td>0.23500003E-01</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>0.15151515E+02</td>
<td>0.25345691E-01</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>0.15151515E+02</td>
<td>0.39744848E-02</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.19327473E+03</td>
<td>0.24044379E-01</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>0.24696070E+01</td>
<td>0.25600003E-01</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>0.14507580E+01</td>
<td>0.16521692E-01</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>0.23448994E+03</td>
<td>0.23500003E-01</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>0.15151515E+02</td>
<td>0.25345691E-01</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>0.15151515E+02</td>
<td>0.39744848E-02</td>
<td></td>
</tr>
</tbody>
</table>

Figure B.2 Reserve Listing of Pushback 3 (cont'd)
*RESERVES FOR FOURTH PUSHBACK (PIT OX4)*

<table>
<thead>
<tr>
<th></th>
<th>Bench Class</th>
<th>Tons</th>
<th>Au.Oz/Ton</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>5</td>
<td>0.7004735E+00</td>
<td>0.000800E-00</td>
</tr>
<tr>
<td>Class 2</td>
<td>5</td>
<td>0.1846685E+02</td>
<td>0.5537651E-04</td>
</tr>
<tr>
<td>Class 3</td>
<td>2</td>
<td>0.2426875E+01</td>
<td>0.1999999E-01</td>
</tr>
<tr>
<td>Class 4</td>
<td>5</td>
<td>0.3910701E+02</td>
<td>0.2135087E-02</td>
</tr>
<tr>
<td>Class 5</td>
<td>5</td>
<td>0.4513181E+01</td>
<td>0.1909999E-01</td>
</tr>
<tr>
<td>Class 6</td>
<td>5</td>
<td>0.1300435E+03</td>
<td>0.4608817E-02</td>
</tr>
<tr>
<td>Class 7</td>
<td>2</td>
<td>0.1301893E+03</td>
<td>0.1909999E-01</td>
</tr>
<tr>
<td>Class 8</td>
<td>5</td>
<td>0.2233894E+03</td>
<td>0.2350000E-02</td>
</tr>
<tr>
<td>Class 9</td>
<td>2</td>
<td>0.1935313E+04</td>
<td>0.1909999E-01</td>
</tr>
<tr>
<td>Class 10</td>
<td>5</td>
<td>0.3498712E+03</td>
<td>0.4508757E-02</td>
</tr>
<tr>
<td>Class 11</td>
<td>1</td>
<td>0.4734843E+00</td>
<td>0.1370531E-01</td>
</tr>
<tr>
<td>Class 12</td>
<td>2</td>
<td>0.2741041E+02</td>
<td>0.2800000E-01</td>
</tr>
<tr>
<td>Class 13</td>
<td>3</td>
<td>0.2051318E+00</td>
<td>0.2886706E-01</td>
</tr>
<tr>
<td>Class 14</td>
<td>5</td>
<td>0.5005876E+03</td>
<td>0.4500000E-02</td>
</tr>
<tr>
<td>Class 15</td>
<td>1</td>
<td>0.1181186E+02</td>
<td>0.1540235E-01</td>
</tr>
<tr>
<td>Class 16</td>
<td>2</td>
<td>0.2507570E+02</td>
<td>0.3401601E-01</td>
</tr>
<tr>
<td>Class 17</td>
<td>3</td>
<td>0.1837721E+01</td>
<td>0.3709702E-01</td>
</tr>
<tr>
<td>Class 18</td>
<td>5</td>
<td>0.8107024E+03</td>
<td>0.5200000E-02</td>
</tr>
<tr>
<td>Class 19</td>
<td>1</td>
<td>0.1725379E+02</td>
<td>0.1241162E-01</td>
</tr>
<tr>
<td>Class 20</td>
<td>2</td>
<td>0.2511362E+02</td>
<td>0.1909999E-01</td>
</tr>
<tr>
<td>Class 21</td>
<td>3</td>
<td>0.5284901E+00</td>
<td>0.2358522E-01</td>
</tr>
<tr>
<td>Class 22</td>
<td>5</td>
<td>0.6036376E+03</td>
<td>0.3455544E-02</td>
</tr>
<tr>
<td>Class 23</td>
<td>1</td>
<td>0.7800018E+01</td>
<td>0.1858124E-01</td>
</tr>
<tr>
<td>Class 24</td>
<td>2</td>
<td>0.1594693E+02</td>
<td>0.1909999E-01</td>
</tr>
<tr>
<td>Class 25</td>
<td>3</td>
<td>0.3257572E+01</td>
<td>0.2350263E-01</td>
</tr>
<tr>
<td>Class 26</td>
<td>5</td>
<td>0.7343268E+03</td>
<td>0.3574971E-02</td>
</tr>
<tr>
<td>Class 27</td>
<td>1</td>
<td>0.1135837E+02</td>
<td>0.1853292E-01</td>
</tr>
<tr>
<td>Class 28</td>
<td>2</td>
<td>0.5209344E+00</td>
<td>0.2551836E-01</td>
</tr>
<tr>
<td>Class 29</td>
<td>3</td>
<td>0.1287878E+00</td>
<td>0.3172243E-01</td>
</tr>
<tr>
<td>Class 30</td>
<td>5</td>
<td>0.7437110E+03</td>
<td>0.3010591E-02</td>
</tr>
<tr>
<td>Class 31</td>
<td>1</td>
<td>0.1437191E+02</td>
<td>0.1434816E-01</td>
</tr>
<tr>
<td>Class 32</td>
<td>2</td>
<td>0.4715006E+01</td>
<td>0.1909999E-01</td>
</tr>
<tr>
<td>Class 33</td>
<td>3</td>
<td>0.1299242E+00</td>
<td>0.2551113E-01</td>
</tr>
<tr>
<td>Class 34</td>
<td>4</td>
<td>0.1803393E+00</td>
<td>0.3509987E-01</td>
</tr>
<tr>
<td>Class 35</td>
<td>5</td>
<td>0.7005619E+03</td>
<td>0.4272534E-02</td>
</tr>
<tr>
<td>Class 36</td>
<td>1</td>
<td>0.1207044E+00</td>
<td>0.1844519E-01</td>
</tr>
<tr>
<td>Class 37</td>
<td>2</td>
<td>0.2024682E+00</td>
<td>0.2000001E-01</td>
</tr>
<tr>
<td>Class 38</td>
<td>3</td>
<td>0.1446097E+00</td>
<td>0.3310463E-01</td>
</tr>
<tr>
<td>Class 39</td>
<td>4</td>
<td>0.1802093E+00</td>
<td>0.3787713E-01</td>
</tr>
<tr>
<td>Class 40</td>
<td>5</td>
<td>0.5709979E+00</td>
<td>0.8604529E-02</td>
</tr>
<tr>
<td>Class 41</td>
<td>1</td>
<td>0.1070448E+00</td>
<td>0.1701799E-01</td>
</tr>
<tr>
<td>Class 42</td>
<td>2</td>
<td>0.1301134E+00</td>
<td>0.2097512E-01</td>
</tr>
<tr>
<td>Class 43</td>
<td>3</td>
<td>0.1534090E+00</td>
<td>0.3633300E-01</td>
</tr>
<tr>
<td>Class 44</td>
<td>4</td>
<td>0.4051610E+00</td>
<td>0.4051647E-02</td>
</tr>
<tr>
<td>Class 45</td>
<td>5</td>
<td>0.1601280E+00</td>
<td>0.1587497E-01</td>
</tr>
<tr>
<td>Class 46</td>
<td>2</td>
<td>0.2369315E+00</td>
<td>0.2174015E-01</td>
</tr>
<tr>
<td>Class 47</td>
<td>3</td>
<td>0.4810055E+00</td>
<td>0.2467912E-01</td>
</tr>
<tr>
<td>Class 48</td>
<td>5</td>
<td>0.5008618E+00</td>
<td>0.4381956E-02</td>
</tr>
<tr>
<td>Class 49</td>
<td>1</td>
<td>0.1512204E+01</td>
<td>0.1212270E-01</td>
</tr>
<tr>
<td>Class 50</td>
<td>2</td>
<td>0.7460999E+00</td>
<td>0.2264190E-01</td>
</tr>
<tr>
<td>Class 51</td>
<td>3</td>
<td>0.5621212E+00</td>
<td>0.2329610E-01</td>
</tr>
<tr>
<td>Class 52</td>
<td>4</td>
<td>0.1653393E+01</td>
<td>0.3764434E-01</td>
</tr>
<tr>
<td>Class 53</td>
<td>5</td>
<td>0.5217809E+00</td>
<td>0.4100000E-02</td>
</tr>
</tbody>
</table>

Figure B.3 Reserve Listing of Pushback 4
Figure B.3 Reserve Listing of Pushback 4 (cont’d)
APPENDIX C

Mine Sequencing Reports for Diamond Pit from Period 2 - 6
MINE SEQUENCING REPORT FOR PERIOD 2 OF 6:

VARIABLE TO BE OPTIMIZED: GRADE
TYPE OF OPTIMIZATION: MAXIMIZATION
MAXIMUM # OF ITERATIONS ALLOWED: 5000
REQUIRED ORE TONNAGE: 833000
REQUIRED WASTE TONNAGE: 2052000
CORRESPONDING STRIPPING RATIO: 2.46

# OF BENCHES FROM PUSHBACK 1: 6
# OF BENCHES FROM PUSHBACK 2: 12
# OF BENCHES FROM PUSHBACK 3: 12
# OF BENCHES FROM PUSHBACK 4: 12
TOTAL # OF BENCHES IN PROBLEM FORMULATION: 42

LIST OF FEASIBLE SCHEDULES (BEFORE ADJUSTMENT):

ITERATION NO: 31
OBJ. VALUE = 3684917

FEASIBLE SCHEDULE NO. 1

PUSHBACK#1 #2 #3 #4
BENCH 22 0 0 0 1
BENCH 23 0 0 0 1
BENCH 24-999 0 1 1
BENCH 25-999 0 1 1
BENCH 26-999 0 1 1
BENCH 27-999 0 1 1
BENCH 28-999 0 1 1
BENCH 29-999 0 1 1
BENCH 30 0 0 1 1
BENCH 31 0 1 1 1
BENCH 32 0 1 1 1

ORE TONNAGE GENERATED: 902063.000
WASTE TONNAGE GENERATED: 1918048.000
MINERAL CONTENTS [OUNCES]: 18357.930
STRIPPING RATIO: 2.126

Figure C.1 Mine sequencing report for period 2
### Iteration No. 2115

**Obj. Value = 3649273**

#### Feasible Schedule No. 2

<table>
<thead>
<tr>
<th>Pushback</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>BENCH 22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 24-999</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 25-999</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 26-999</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 27-999</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 28-999</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 29-999</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 30</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 31</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 32</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Ore Tonnage Generated**: 906485.000  
**Waste Tonnage Generated**: 1923070.000  
**Mineral Contents [Ounces]**: 18714.371  
**Stripping Ratio**: 2.121

### Iteration No. 2962

**Obj. Value = 3547838**

#### Feasible Schedule No. 3

<table>
<thead>
<tr>
<th>Pushback</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>BENCH 22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 24-999</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 25-999</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 26-999</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 27-999</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 28-999</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 29-999</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 30</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 31</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BENCH 32</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Ore Tonnage Generated**: 938905.000  
**Waste Tonnage Generated**: 1961150.000  
**Mineral Contents [Ounces]**: 19728.721  
**Stripping Ratio**: 2.089

---

Figure C.1 Mine sequencing report for period 2 (cont’d)
Figure C.1 Mine sequencing report for period 2 (cont’d)
Problem run completed and terminated at iteration: 5000
Iteration no 5000   Obj. value = 3364481

Optimum solution obtained:

PUSHBACK #1 #2 #3 #4
BENCH 22 0 0 0 1
BENCH 23 0 0 0 1
BENCH 24-999 0 0 0 1
BENCH 25-999 0 0 0 1
BENCH 26-999 0 0 0 1
BENCH 27-999 0 0 0 1
BENCH 28-999 0 0 0 1
BENCH 29-999 1 1 1
BENCH 30 0 1 1 1
BENCH 31 0 1 1 1
BENCH 32 0 1 1 1

Ore tonnage generated: 918686.000
Waste tons generated: 1753305.000
Mineral contents (ounces): 21562.291
Stripping ratio: 1.908

The optimal Obj value is: 3364481

Normal end of the execution

Final mine schedule (after adjustment) for period 2 of 6

PUSHBACK #1 #2 #3 #4
BENCH 24-999.00 100.00 100.00 0.00
BENCH 25-999.00 100.00 100.00 0.00
BENCH 26-999.00 100.00 100.00 0.00
BENCH 27-999.00 100.00 100.00 0.00
BENCH 28-999.00 100.00 100.00 0.00
BENCH 29-999.00 100.00 82.99 0.00
BENCH 30 100.00 0.00 0.00 0.00
BENCH 31 100.00 0.00 0.00 0.00
BENCH 32 100.00 0.00 0.00 0.00
BENCH 33 100.00 0.00 0.00 0.00

Ore tonnage generated: 833000.000
Waste tons generated: 2002929.000
Mineral contents (ounces): 19407.939
Stripping ratio: 2.404

Figure C.1 Mine sequencing report for period 2 (cont’d)
Figure C.2 Mine sequencing report for period 3
Figure C.2 Mine sequencing report for period 3 (cont'd)
Figure C.2 Mine sequencing report for period 3 (cont’d)
## Optimum Solution Obtained:

<table>
<thead>
<tr>
<th>Pushback</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bench 22</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 23</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 24-999</td>
<td>0.00</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Bench 25-999</td>
<td>0.00</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Bench 26-999</td>
<td>0.00</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Bench 27-999</td>
<td>0.00</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Bench 28-999</td>
<td>0.00</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Bench 29-999</td>
<td>0.00</td>
<td>999.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 30-999</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 31-999</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 32-999</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 33-999</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 34</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Bench 35</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Bench 36</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

- **Ore Tonnage Generated:** 2145819.000
- **Waste Tons Generated:** 4588798.000
- **Mineral Contents [Ounces]:** 81340.289
- **Stripping Ratio:** 2.138

**The optimal obj value is:** 7480880
**Normal End of the Execution**

### Final Mine Schedule (After Adjustment) for Period 3 of 6

<table>
<thead>
<tr>
<th>Pushback</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bench 22</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 23</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 24</td>
<td>999.00</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Bench 25</td>
<td>999.00</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Bench 26</td>
<td>999.00</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Bench 27</td>
<td>999.00</td>
<td>999.00</td>
<td>999.00</td>
</tr>
<tr>
<td>Bench 28</td>
<td>999.00</td>
<td>999.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 29</td>
<td>999.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 30</td>
<td>999.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Bench 31</td>
<td>999.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Bench 32</td>
<td>999.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Bench 33</td>
<td>999.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Bench 34</td>
<td>100.00</td>
<td>100.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 35</td>
<td>100.00</td>
<td>100.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 36</td>
<td>100.00</td>
<td>100.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 37</td>
<td>-8.00</td>
<td>100.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Bench 38</td>
<td>-8.00</td>
<td>40.62</td>
<td>0.00</td>
</tr>
</tbody>
</table>

- **Ore Tonnage Generated:** 2000000.000
- **Waste Tons Generated:** 4876800.000
- **Mineral Contents [Ounces]:** 48250.797
- **Stripping Ratio:** 2.438

Figure C.2 Mine sequencing report for period 3 (cont’d)
MINE SEQUENCING REPORT FOR PERIOD 4 OF 6:

VARIABLE TO BE OPTIMIZED : GRADE
TYPE OF OPTIMIZATION : MAXIMIZATION
MAXIMUM # OF ITERATIONS ALLOWED : 5000
REQUIRED ORE TONNAGE : 2000000
REQUIRED WASTE TONNAGE : 5652000
CORRESPONDING STRIPPING RATIO : 2.83

# OF BENCHES FROM PUSHBACK 1 : 0
# OF BENCHES FROM PUSHBACK 2 : 3
# OF BENCHES FROM PUSHBACK 3 : 13
# OF BENCHES FROM PUSHBACK 4 : 17
TOTAL # OF BENCHES IN PROBLEM FORMULATION : 33

LIST OF FEASIBLE SCHEDULES (BEFORE ADJUSTMENT):

<table>
<thead>
<tr>
<th>ITERATION NO</th>
<th>OBJ. VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>68</td>
<td>5258426</td>
</tr>
</tbody>
</table>

FEASIBLE SCHEDULE NO. 1

PUSHBACK 1 #2 #3 #4

<table>
<thead>
<tr>
<th>BENCH</th>
<th>22</th>
<th>0</th>
<th>0</th>
<th>0-999</th>
</tr>
</thead>
<tbody>
<tr>
<td>BENCH</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0-999</td>
</tr>
<tr>
<td>BENCH</td>
<td>28-999</td>
<td>999</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>BENCH</td>
<td>29-999</td>
<td>999</td>
<td>999</td>
<td></td>
</tr>
<tr>
<td>BENCH</td>
<td>30-999</td>
<td>999</td>
<td>999</td>
<td></td>
</tr>
<tr>
<td>BENCH</td>
<td>31-999</td>
<td>999</td>
<td>999</td>
<td></td>
</tr>
<tr>
<td>BENCH</td>
<td>32-999</td>
<td>999</td>
<td>999</td>
<td></td>
</tr>
<tr>
<td>BENCH</td>
<td>33-999</td>
<td>999</td>
<td>999</td>
<td></td>
</tr>
<tr>
<td>BENCH</td>
<td>34-999</td>
<td>999</td>
<td>999</td>
<td></td>
</tr>
<tr>
<td>BENCH</td>
<td>35-999</td>
<td>999</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BENCH</td>
<td>36-999</td>
<td>999</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BENCH</td>
<td>37-8-999</td>
<td>999</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BENCH</td>
<td>38-8</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>BENCH</td>
<td>39-8</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

ORE TONNAGE GENERATED : 1918798.000
WASTE TONS GENERATED : 5562815.000
MINERAL CONTENTS [OUNCES] : 40132.000
STRIPPING RATIO : 2.899

PROBLEM RUN COMPLETED AND TERMINATED AT ITERATION: 1171

<table>
<thead>
<tr>
<th>ITERATION NO</th>
<th>OBJ. VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1171</td>
<td>5258426</td>
</tr>
</tbody>
</table>

Figure C.3 Mine sequencing report for period 4
Figure C.3 Mine sequencing report for period 4 (cont’d)
MINE SEQUENCING REPORT FOR PERIOD 5 OF 6:

VARIABLE TO BE OPTIMIZED : GRADE
TYPE OF OPTIMIZATION : MAXIMIZATION
MAXIMUM # OF ITERATIONS ALLOWED : 5000
REQUIRED ORE TONNAGE : 2000000
REQUIRED WASTE TONNAGE : 5636000
CORRESPONDING STRIPPING RATIO : 2.82

# OF BENCHES FROM PUSHBACK 1 : 0
# OF BENCHES FROM PUSHBACK 2 : 0
# OF BENCHES FROM PUSHBACK 3 : 5
# OF BENCHES FROM PUSHBACK 4 : 14
TOTAL # OF BENCHES IN PROBLEM FORMULATION 19

LIST OF FEASIBLE SCHEDULES (BEFORE ADJUSTMENT):

<table>
<thead>
<tr>
<th>ITERATION NO</th>
<th>OBJ. VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1936539</td>
</tr>
</tbody>
</table>

FEASIBLE SCHEDULE NO. 1

PUSHBACK#1 #2 #3 #4

BENCH 22 0 0 0-999
BENCH 23 0 0 0-999
BENCH 33-999-999-999 0
BENCH 34-999-999-999 0
BENCH 35-999-999-999 0
BENCH 36-999-999-999 0
BENCH 37 -8-999-999 0
BENCH 38 -8-999-999 0
BENCH 39 -8-999-999 0
BENCH 40 -8-999-999 0
BENCH 41 -8 -8-999 0
BENCH 42 -8 -8 0 0
BENCH 43 -8 -8 0 0
BENCH 44 -8 -8 0 1

ORE TONNAGE GENERATED : 2051300.000
WASTE TONNAGE GENERATED : 5651318.000
MINERAL CONTENTS [G/TON] : 436533.512
STRIPPING RATIO : 2.765

- PROBLEM RUN COMPLETED AND TERMINATED AT ITERATION: 816

ITERATION NO 816 OBJ. VALUE = 1936539

Figure C.4 Mine sequencing report for period 5
OPTIMUM SOLUTION OBTAINED:

PUSHBACK #1 #2 #3 #4
BENCH 22 0 0 0-999
BENCH 23 0 0 0-999
BENCH 24-999-999-999-999
BENCH 32-999-999-999-999
BENCH 33-999-999-999 0
BENCH 34-999-999-999 0
BENCH 35-999-999-999 0
BENCH 36-999-999-999 0
BENCH 37 0-999-999 0
BENCH 38 0-999-999 0
BENCH 39 0-999-999 0
BENCH 40 0-999-999 0
BENCH 41 0 0-999 0
BENCH 42 0 0 0 0
BENCH 43 0 0 0 0
BENCH 44 0 0 0 0

ORE TONNAGE GENERATED: 2051300.000
WASTE TONS GENERATED: 5651318.000
MINERAL CONTENTS [OUNCES]: 43633.512
STRIPPIING RATIO: 2.755

THE OPTIMAL OBJ VALUE IS: 1936539

NORMAL END OF THE EXECUTION

FINAL MINE SCHEDULE (AFTER ADJUSTMENT) FOR PERIOD 5 OF 6

PUSHBACK #1 #2 #3 #4
BENCH 33-999.00-999.00-999.00 100.00
BENCH 34-999.00-999.00-999.00 100.00
BENCH 35-999.00-999.00-999.00 100.00
BENCH 36-999.00-999.00-999.00 100.00
BENCH 37 -8.00-999.00-999.00 100.00
BENCH 38 -8.00-999.00-999.00 100.00
BENCH 39 -8.00-999.00-999.00 100.00
BENCH 40 -8.00-999.00-999.00 100.00
BENCH 41 -8.00-8.00-999.00 100.00
BENCH 42 -8.00-8.00 100.00 100.00
BENCH 43 -8.00-8.00 100.00 80.90
BENCH 44 -8.00-8.00 100.00 0.00
BENCH 45 -8.00-8.00 100.00 0.00
BENCH 46 -8.00-8.00 100.00 0.00

ORE TONNAGE GENERATED: 2000000.000
WASTE TONS GENERATED: 5618730.000
MINERAL CONTENTS [OUNCES]: 42382.750
STRIPPIING RATIO: 2.809

Figure C.4 Mine sequencing report for period 5 (cont'd)
Figure C.5 Mine sequencing report for period 6
OPTIMUM SOLUTION OBTAINED:

PUSHBACK #1 #2 #3 #4
BENCH 22 0 0 0-999
BENCH 23 0 0 0-999
BENCH 24 0-999-999-999-999
BENCH 26 0-999-999-999-999
BENCH 27 0-999-999-999
BENCH 28 0-999-999-999
BENCH 29 0-999-999-999
BENCH 30 0-999-999-999
BENCH 31 0-999-999-999
BENCH 32 0-999-999-999
BENCH 33 0-999-999-999
BENCH 34 0-999-999-999
BENCH 35 0-999-999-999
BENCH 36 0-999-999-999
BENCH 37 0-999-999-999
BENCH 38 0-999-999-999
BENCH 39 0-999-999-999
BENCH 40 0-999-999-999
BENCH 41 0-999-999
BENCH 42 0-999-999
BENCH 43 0-999 0
BENCH 44 0-999 0
BENCH 45 0-999 0
BENCH 46 0-999 0
BENCH 47 0 0 0
BENCH 48 0 0 0
BENCH 49 0 0 0

ORE TONNAGE GENERATED : 1363042.000
WASTE TONS GENERATED : 512116.000
MINERAL CONTENTS [OUNCE]: 32736.670
STRIPPING RATIO : 0.376

THE OPTIMAL OBJ VALUE IS : 262900

NORMAL END OF THE EXECUTION

FINAL MINE SCHEDULE (AFTER ADJUSTMENT) FOR PERIOD 6 OF 6

PUSHBACK #1 #2 #3 #4
BENCH 43 -8.00 -8.00-999.00 100.00
BENCH 44 -8.00 -8.00-999.00 100.00
BENCH 45 -8.00 -8.00-999.00 100.00
BENCH 46 -8.00 -8.00-999.00 100.00
BENCH 47 -8.00 -8.00 -8.00 100.00
BENCH 48 -8.00 -8.00 -8.00 100.00
BENCH 49 -8.00 -8.00 -8.00 100.00
BENCH 50 -8.00 -8.00 -8.00 100.00

ORE TONNAGE GENERATED : 1432114.000
WASTE TONS GENERATED : 535430.000
MINERAL CONTENTS [OUNCE]: 35365.672
STRIPPING RATIO : 0.374

Figure C.5 Mine sequencing report for period 6 (cont’d)
APPENDIX D

FORTRAN Program Listing for 0-1 Programming Routine
SUBROUTINE ZONE
CHARACTER TITLE*80, DATE*20
DIMENSION A(100,640), B(100, 640), C(640), INV (640)
DIMENSION BP (20), D(4,640), ISEQ(100), NBIP(20), BEMPTY(20)
DIMENSION MBMD (20,120), MSBP (20), IOPT(640), MBMD(20,120)
DIMENSION MBMD, MBMD
COMMON/IN1/IPI,T,CLBD, MB, JOPT
COMMON/AD3/MBIP, MBIP, BEMPTY
COMMON/AD4/MBBP(120), MBDB(20), MBAP
COMMON/CLS2/IPICT, PICT, PACT
COMMON/SP1/WT1, WST, A, B, C, INV, NRM
COMMON/SP2/10PT, XEQ, XEQFL, XSEQP, XPIAD, XPIOF
COMMON/SP3/IPI, XPI, D, XPI, XPI, XPI, XPI, XPI, XPI, XPI
COMMON/SP4/TITLE, DATE
COMMON/SYS/ ICON, IPT, IPT, IPT, IPT, IPT, IPT, IPT, IPT, IPT, IPT
COMMON/JS/JS(640), Y(100), JSUL(640), JSUL(640), JSUL(640), JSUL(640)
DIMENSION JS(640), Y(100), JSUL(640), JSUL(640), JSUL(640), JSUL(640)
INTEGER SORT, SORT, A, B, C, D, Y, BEMPTY, E
INTEGER TEMP, XN, XN, ZOPT, ZOPT, ZOPT, ZOPT, ZOPT
DATA ICOI, IYAR/100,640/

C*** INITIALIZATION OF PROGRAM SYSTEM PARAMETERS
C
C  LLL = 0
C
DO 43 I=1,NVAR
  JP(I)=0
  JSUL(I)=0
  BS(I)=0
  MK(I)=0
  Y(I)=0
  MP(I)=0
  V(I)=0
  MB(I)=0
  JOPT(I)=0
C
  JSUL(I)=0
  SORT(I)=0
  IOPT(II)=0
DO 42 J=1,MBM
  42 CONTINUE
C
  DO 44 IPIT=1,MBP
  44 MBP(IPIT)=0
C
C*** CUMULATIVE NO. OF BENCHES IN PITS
C
DO 1115 IPIT=1,MBP
  1115 CONTINUE
        WRITE(*, 1118) (MBP(IPIT), IPIT=1, MBP)
1118 FORMAT(5I, 'THE ARRAY MBP ' ,8IS)
C
C**** SET THE VALUE OF KM TO THE ADJUSTED VALUE IN 'SETUP' & 'ZER01'.
C

IM = NCUL
NM = NMR
IF (IDBG.EQ.2) WRITE(IPT,51) MM
51 FORMAT(115)
C
C**** ECHO TO THE INPUT DATA SO FAR.
C
C WRITE(ICON,1107) TITLE
1107 FORMAT(/,8X,A80)
C WRITE(ICON,1200) DATE
1200 FORMAT(8X,A20)
C
IF(IDBG.EQ.5) GO TO 1130
C WRITE(ICON,1103)
WRITE(IPR,1103)
1103 FORMAT(/,8X,'ECHO TO THE STANDARDIZED PROBLEM FORMULATION :')
C WRITE(IPR,1101)
1101 FORMAT(/,IX,5I,' COEF. OBJ. FUNC. /',/)
WRITE(IPR,1102) (C(I),I=1,MM)
1102 FORMAT(8I15)
C WRITE(IPR,1120)
1120 FORMAT(/,IX,5I,' COEF. RHS . ',/)
DO 1105 L = 1, NM
WRITE(IPR,1110) (B(L,K),K=1,MM)
1105 CONTINUE
C WRITE(IPR,1125)
1125 FORMAT(/,IX,5I,' MATRIX ELEMENTS ',/)
DO 1135 L = 1, NM
WRITE(IPR,1110) (A(L,K),K=1,MM)
1110 FORMAT(8I15)
C
1130 WRITE(IPR, 9001) (I,WRIP(I),I=1,NPIT)
9001 FORMAT(10I,I,' # OF BENCHES FROM PIT ',I2,': ',I5)
WRITE(IPR, 9002) MSRIP(NPIT)
9002 FORMAT(10I,' TOTAL # OF BENCHES IN PROBLEM FORMULATION ',I5/)
C
WRITE(IPR,1109)
1109 FORMAT(/10I,'LIST OF FEASIBLE SCHEDULES (BEFORE ADJUSTMENT): ','/10I,')
C
C**** DETERMINE THE NO. OF NON-ZERO MATRIX ELEMENT. ACCORDING TO
C**** THE RATIO OF OBJ COEF. OVER CONSTRAINT COEF., FOR EACH ROW, FROM
C**** LEFT TO RIGHT, SORT THE COLUMN ID OF POSITIVE COEF. IN AN
C**** INCREASING ORDER.
C
C*** FIRST SETUP UPPER BOUND FOR OBJ. VALUE
C
NRBOUND = 2**30
C
DO 49 I=1,MM
WR(I)=0
C
NR = COUNTER FOR NON-ZERO MATRIX ELEMENT.
C  JR = MODIFIED AND SORTED NON-ZERO MATRIX ELEMENT.
C
DO 46 J=1,NH
   IF(A(I,J).LT.0) GO TO 46
   WR(I)=WR(I)+1
   IABC=WR(I)
   IF(A(I,J).NE.0) GO TO 4401
   SORT(IABC)=MBOUND
   GO TO 4402
4401 SORT(IABC)=C(J)/A(I,J)
4402 JR(I,IABC)=J
45 CONTINUE
C
   IF(WR(I).LE.1) GO TO 49
   IAB=WR(I)-1
   DO 48 K=1,IAB
      KPI=K+1
      IAB=WR(K)
      DO 48 L=KPI,IABC
         IF(SORT(K).LE.SORT(L)) GO TO 48
      TEMP=SORT(K)
      SORT(K)=SORT(L)
      SORT(L)=TEMP
      JR(I,K)=JR(I,L)
      JR(I,L)=TEMP
48 CONTINUE
49 CONTINUE
C
   IF(IDBG.EQ.3) WRITE(IOUT,22) IAB,(WR(L),L=1,IAB)
22 FORMAT(IX,5X,'IAB AND WR ARRAY ',16I6)
   IF(1DDBG.EQ.3) WRITE(IOUT,1122)
      - (((JR(I,J),J=1,NCOL),L=1,NM)
1122 FORMAT(IX,5X,'ARRAY JR(1,J) :'/,1X,8I14) /
C
C****LOCATE THE ROW WHOSE LEFT MOST POSITIVE COEFFICIENTS (HAVING
C**** BEEN SORTED) WILL FIRST BE ADDED INTO STARTING PARTIAL SOLUTION
C
C
ZMIX = MBOUND
MS=1
ICOUNT = 0
300 BMOST=B(NM,1)
IMAX=NM
DO 304 I=1,NM
   IF(BMOST.GE.B(I,1)) GO TO 304
   BMOST=B(I,1)
   IMAX=I
304 CONTINUE
C
   IF (IPRIOF.NE.0) IMAX = IPRIO
   IABC=WR(IMAX)
   IF (IABC.NE.0) GO TO 305
   WRITE(*,50) IABC
   WRITE(IPR,50) IABC
50 FORMAT(/,6X,'ERROR IN VALUE ASSIGNMENT OF "IPRIO" -- THEREFORE',
      --' THE EXECUTION HAS BEEN TERMINATED', IS)
      STOP
C
305 CONTINUE
DO 325 I=1,IABC
C JS(I)=JS(IMAX,I) >>> CHANGE FOR S.R.
JS(J) = I
DO 319 K=1,NM
Y(K)=B(Z,JS(I))
DO 318 L=1,I
IAB=JS(L)
IF (L .EQ. 1) GO TO 318
IA = JS(L-1)
C IF (B(K,IAB).LE.B(K,IA)) GO TO 318
Y(K) = Y(K) + B(K,IA)
Y(K) = Y(K) - B(K,IAB)
318 Y(K) = Y(K) + A(K,IAB)
C 318 WRITE(*,*) 'Y(K),IA,IAB,B(K,IA),B(K,IAB),A(K,IAB)'
C -
Y(K),IA,IAB,B(K,IA),B(K,IAB),A(K,IAB)
IF(IDBG.EQ.3) WRITE(IOUT,23) K, Y(K)
C WRITE(*,23) K, Y(K)
IF(Y(K).LT.0) GO TO 326
319 CONTINUE
C IF(IDBG.EQ.3) WRITE(IOUT,23) (K,Y(K),K=1,NM)
C WRITE(*,23) (K,Y(K),K=1,NM)
23 FORMAT(1X,52,'Y(',I2,')=',10)
C C CJSUM=0
C DO 323 J=1,I
IAB=JS(I))
C CJSUM=CJSUM+C(IAB)
323 CONTINUE
JRM=-1
C IF(IDBG.EQ.3) WRITE(IOUT,24) JRM,EABC,(JS(J),J=1,IABC)
C WRITE(*,24) JRM,EABC,(JS(J),J=1,IABC)
24 FORMAT(1X,62,'PATH TO ST. NO. 72',213,813)
C C *** FIRST FEASIBLE SOLUTION HAS BEEN FOUND. GO TO PATH 72.
C C GO TO 72
325 CONTINUE
JRM=IFR(IMAX)
C IF(IDBG.EQ.3) WRITE(IOUT,25) JRM, EABC, (JS(J),J=1,IABC)
C WRITE(*,25) JRM, EABC, (JS(J),J=1,IABC)
25 FORMAT(1X,62,'PATH TO ST. NO. 100',213,813)
C C *** NO FEASIBLE SOLUTION HAS BEEN FOUND AFTER ALL THE POSITIVE COEF.
C *** IN ROW IMAX HAVE BEEN ADDED TO THE STARTING PARTIAL SOLUTION
C *** GO TO PATH 100 FOR REFORMULATION.
C C GO TO 100
C C------------------------------------------------
C 62 JS=JS(I)
C TIMES MODIFICATION.
C S2 MS=MS+1
C ICOUNT = ICOUNT + 1
C C**** TEMPORARY MODIFICATION.
C IF (ICOUNT.NE.100) GO TO 1111
C ICOUNT = 0
IF(IDBG.EQ.3) WRITE(IOUT,20) MS,ZMIN
WRITE(*,20) MS,ZMIN
20 FORMAT(I0X,'ITERATION NO.',I7,10X,'OBJ. VALUE =',I10)

C
IF (IDBG.EQ.3) WRITE(IOUT,11) (JS(J),J=1,NW)
WRITE(*,11) (JS(J),J=1,NW)
11 FORMAT(I0X,5X,'JS ARRAY VALUES',6I3)

C
IF(IDBG.EQ.3) WRITE(IOUT,27) CJSUM,CFSUM, (JS(K),K=1,NW)
IF(IDBG.EQ.3) WRITE(IOUT,29) CJSUM,CFSUM,(JSUL(K),K=1,NW)
29 FORMAT(I0X,5X,'CJSUK,CFSUK,JJSUL ARRAY',6I3)
C
ITERATION COUNTER, EXIT IF MSMAX IS EXCEEDED
C
1111 IF (MS.EQ.MSMB) GO TO 81
C
5820 IF(MS.LE.MSMB) GO TO 64
WRITE(10X) MS, MSMB
WRITE(IOUT,18) MS,MSMB
18 FORMAT(10X,'ITERATION NO. EXCEEDED, THEREFORE STOP',/,
110X,'MS = ',I10,5X,'MSMB = ',I10,/) STOP
C
DO 57 II=1,NW
II=II
IF(JS(J).EQ.0) GO TO 58
IF(NW.EQ.1) GO TO 57
II=II+1
GO TO 58
57 CONTINUE
58 IF(JS(NW).EQ.0) GO TO 8100
IF (I.EQ.MM) JMSM=MM
C
C*** COMPUTE THE CURRENT OB. F. VALUE. JMSM = COUNTER OF BASIC VARIABLES.
8100 CJSUM=0
DO 64 1=1,JMSM
IF(JS(I).LE.0) GO TO 64
IAB=JS(I)
CJSUM=CJSUM+C(IAB)
64 CONTINUE
C
DO 67 I=1,NM
DO 62 J = 1, NW
IF (JS(J).LE.0) GO TO 62
Y(I) = -B(I,JS(J))
GO TO 63
62 CONTINUE
63 CONTINUE
DO 65 K= 1, NW
IF (JS(K).LE.0) GO TO 65
Y10 = JS(K)
IPP=B(I,Y10)
Y11 = IABS(IPP)
IPP = Y(I)
Y12 = IABS(IPP)
IF (Y11 .LE. Y12) GO TO 65
Y(I) = -B(I,Y10)
55 CONTINUE
   DO 68 J=1, NME
      IF(JS(J).LE.0) GO TO 68
      IAB=JS(J)
      Y(I)=Y(I)+A(I,IAB)
68   CONTINUE
70   CONTINUE
 C
 IF(IDBG.EQ.3) WRITE(IOUT,26) CJSUM,CPSUM, (Y(I),I=1,NM)
 IF(IDBG.EQ.3) WRITE(IOUT,27) CJSUM,CPSUM, (JS(K),K=1,NM)
 26  FORMAT(I6,6X,' CJSUM,CPSUM, AND Y, 2I10, 8I10)
 C
C**** CHECK IF IMPROVEMENT WAS MADE DURING THIS ITERATION. CJSUM = OB.VALUE
   DO 71 J=1,NM
71   IF(Y(I).LT.0) GO TO 100
C ... ADJUST THE STATEMENT BELOW COULD RESULT A LOT OF FEASIBLE SOL.
 IF(2MIN.LE.CJSUM) GO TO 74
 IF(CJSUM.EQ.0) GO TO 100
72  CJSUM=CJSUM
   DO 76 J=1, NME
76   JOPT(J) = JS(J)
   IAB=JME+1
   DO 79 J=IAB,NM
79   JOPT(J)=0
 C
 C ... REVERSE ORDER OF SOLUTION VARIABLES IF JOPT=1 (MAX PROBLEM)
     IF (JOPT .NE.0) THEN
 C
 C ... POSITION VARIABLE TO THEIR CORRESPONDING POSITION IN STORING ARRAY
   DO 684 J=1, NM
684  E(J) = 0
   DO 683 J=1, NM
      IF (JS(J) .NE. 0) THEN
         K = IABS(JS(J))
         E(K) = JS(J)
      ENDIF
683  CONTINUE
 C *** RESUME I FROM ' 1 - X ' IF APPLICABLE.
   DO 1354 K = 1, NM
      IF (E(K).EQ.0) E(K) = K
   1354  CONTINUE
 C ... RESUME ORIGINAL MEANING OF VARIABLES
   DO 680 J=1, NM
      IF (E(J) .GT. 0) THEN
         E(J) = NM - E(J) + 1
      ELSE
         E(J) = 0
      ENDIF
680  CONTINUE
 C ... POSITION THE VARIABLES BACK TO ORIGINAL FORMULATION
   DO 685 J=1, NM
      K = NM - J + 1
   685   E(J) = JS(J)
C
73 CONTINUE
   WRITE(IPR,20) NS,ZMIN
   WRITE(*,20) NS,ZMIN
   LLL = LLL + 1
   IFLAG = 0
55 FORMAT(2I15)
   WRITE(IPR,5520) LLL
   WRITE(*,5520) LLL
5520 FORMAT(/,'X', 'FEASIBLE SCHEDULE NO. ',I6)
5521 FORMAT(10I1,14I8)
C
   DO 2222 IPIT = 1,MPIT
      MBCH = MBIP(IPIT)
      DO 2222 IBENCH=1,MBCH
      IF (NBND(IPIT,IBENCH) .LT. 0) GO TO 2222
      NBND(IPIT,IBENCH) = 1
2222 CONTINUE
   ISMAX=0
   DO 1113 KS=1,NS
      IF(E(KS).LE.0) GO TO 1113
      DO 1117 IP=1,MPIT
      IF(E(KS).GT.NSBIP(IP)) GO TO 1117
      IF (E(KS).GT.NSBIP(IP)) GO TO 1119
      IS=E(KS)
      IS = IS + BEMPTY(IP)
      GO TO 1124
1119 IS = E(KS) - NSBIP(IP-1)
      IS = IS + BEMPTY(IP)
   1124 CONTINUE
   C
   WRITE(IPR,1126) KS,IP,IS,ISMAX,E(KS),NSBIP(IP-1),NSBIP(IP)
   C
   WRITE(*,1126) KS,IP,IS,ISMAX,E(KS),NSBIP(IP-1),NSBIP(IP)
1126 FORMAT(/,'THE VALUE OF KS' = ',I10,/ & ' IPIT = ',I10,/ & ' IS = ',I10,/ & ' ISMAX = ',I10,/ & ' E(' ,I3,' ) = ',I10,/ & ' NSBIP(IP-1) = ',I10,/ & ' NSBIP(IP) = ',I10,/)
   NBND(IP,IS)=0
   IF (ISMAX.LT.IS) ISMAX=IS
   GO TO 1113
   C
1117 CONTINUE
1113 CONTINUE
C
   WRITE(IPR,1116)
   WRITE(*,1116)
1116 FORMAT(/11X,'PUSHBACK #1 #2 #3 #4')
   NH = ISMAX + NGAP
   IF (NH .GT. 120) NH = 120
   DO 1112 IN=1,NH
   C
   IF (INDB(IP) .LE. 0) GO TO 1112
   DO 1202 IPIT=1,MPIT
      HTRB(IPIT) = 0
      JDF = INDB(IPIT)-INDS(1)
      IF (INDB(IPIT).LE.INDS(IPIT)) THEN
ITnP(IPIT) • IBKD(IPIT,IB-JDP)
ENDIF
IF (IMDB(1B),GT,MBIP(IPIT)+INDTB(IPIT)-1) NTMP(IPIT) = -8
1202 CONTINUE
DO 1203 IPIT=1, NPIT
IF (NTMP(IPIT) .GE. 0) GO TO 1204
1203 CONTINUE
GO TO 1112
1204 CONTINUE
WRITE(IPR,1114) IMDB(IB),(NMTIP(IPIT),IPIT=1,NPIT)
WRITE(*, 1114) IMDB(IB),(NMTIP(IPIT),IPIT=1,NPIT)
WRITE('101, 1114) IMDB(IB),(NMTIP(IPIT),IPIT=1,NPIT)
1114 FORMAT(11X,'BENCH ',I2,2014)
C
NRTCT=0
NRVT=0
NRGT=0
DO 8000 J=1,NH
JSJ=JS(J)
IF(JSJ.LE.0) GO TO 8000
NRTCT=NRTCT+1
NRVT=NRVT+2
NRGT=NRGT+3
8000 CONTINUE
STRI = 99999.
IF (NRCT .GT. 0) THEN
STRI = STRI/(NRCT)*IFACTV
ENDIF
C
RCCT = FLOAT(NRTCT)*1000.0/FLOAT(IFACTV)
RNVT = FLOAT(NRVT)*1000.0/FLOAT(IFACTV)
RNGT = FLOAT(NRGT)*1000.0/FLOAT(IFACTV)
WRITE('*.8010) RCCT,RNVT,RNGT,STRI
WRITE('101,8010) RCCT,RNVT,RNGT,STRI
8010 FORMAT(/,10X,'ORE TONNAGE GENERATED : ',F15.3,
       /,10X,'WASTE TONNS GENERATED : ',F15.3,
       /,10X,'MINERAL CONTENTS [Ounces]: ',F15.3,
       /,10X,'STRIPPING RATIO : ',F15.3/)
WRITE('*.8012)
WRITE('101,8012)
8012 FORMAT(/5X,'HIT <CR> TO CONTINUE, ENTER 999 TO EXIT ==> ')
READ('*.8013) IAS
WRITE('101,8013) IAS
8013 FORMAT(I3)
IF (IAS .EQ. 999) RETURN
C
C**** REINITIALIZE OPTIMUM BASIS ARRAY.
C
74 CONTINUE
IF (JSUL(JRKE) .GE. 0) GO TO 92
DO 80 I=1,JRKE
J=JRKE+I-1
IF (JSUL(I)) .GE. 0) GO TO 85
80 CONTINUE
C
81 CONTINUE
C**** PROBLEM COMPLETE.
C
WRITE(ICON,1220) MS
WRITE(IPR,1220) MS
1220 FORMAT(/,'PROBLEM RUN COMPLETED AND TERMINATED AT ITERATION &','/)
C
IF no feasible solution is found, jump to 5504.
C
IF (ILL.LT.0) GO TO 5504
WRITE(ICON,20) MS,ZMIN
WRITE(IPR,20) MS,ZMIN
IF(ISBG.EQ.3) WRITE(IOUT,20) MS,ZMIN
C
IF (IDBG.EQ.3) WRITE(IOUT,5100)
WRITE(ICON,5100)
WRITE(IPR,5100)
WRITE(ICON,1116)
WRITE(IPR,1116)
MS = ISMAX + 1
IF (MS .GT. 120) MS = 120
C
DO 1128 IBEICH=1,MS
IB = IBERCH
DO 1129 IPIT=1,IPIT
MTMP(IPIT) = 0
JDP = INDDB(IPIT)-INDBB(1)
IF (INDDB(IPIT).LE.INDBB(IBERCH)) THEN
MTMP(IPIT) = NMBD(IPIT,IBERCH-JDP)
ENDIF
IF (INDBB(IB).GT.MBIP(IPIT)+INDTB(IPIT)-1) MTMP(IPIT) = -8
1129 CONTINUE
WRITE(IPR,1114) INDDB(IBERCH),(MTMP(IPIT),IPIT=1,IPIT)
WRITE(ISV,1226) INDDB(IBERCH),(MTMP(IPIT),IPIT=1,IPIT)
1128 CONTINUE
C
1226 FORMAT(1,14E18)
WRITE(IPR,8010) BTCW,SWTW,AGTW,STRIPK
C
C**** CHECK IF THE SOLUTION IS BOUNDED. JJP = COUNTER.
C
1480 JJP = 0
DO 405 I=1,MS
IF(JOPT(I).LE.0) GO TO 405
ILOPT(I) = JOPT(I)
JJP = JJP + 1
405 CONTINUE
C
IF (JJP.GT.0) GO TO 5501
C
5504 BTCW = 0
SWTW = 0
IFLAG = 1
C
WRITE(IPR,331)
WRITE(ICON,331)
331 FORMAT(/,'CONTRAIN[0] NEED TO BE RELAXED : '/)
DO 5506 K=1, MM
   IF (Y(K) .LT. 0) THEN
      WRITE (IPB, 5506) K
      WRITE (*, 5506) K
   ENDIF
5506 CONTINUE
5506 FORMAT (10X, 'CONSTRAINTS # ',I2)
C IF(IDBG.EQ.3) WRITE(IOUT,331)
C GO TO 5502
C C
5501 ZMIN = 0
   DO 5070 K=1, MM
      KK=IOUT(K)
      IF (KK.LE.0) GO TO 5070
      ZMIN = ZMIN + C(KK)
5070 CONTINUE
C WRITE (ICON,6080) ZMIN
   WRITE (IPR,5080) ZMIN
   IF (IDBG.EQ.3) WRITE (IOUT,6080) ZMIN
5080 FORMAT (/10X,'THE OPTIMAL OBJ VALUE IS : ',110)
C 5502 WRITE (ICON,19)
   WRITE (IPR,19)
   IF (IDBG.EQ.3) WRITE (IOUT,19)
   19 FORMAT (/,'NORMAL END OF THE EXECUTION')
RETURN
C C**** START BACK TRACKING BY ASSIGNING THE RIGHT MOST POSITIVE VARIABLE
C *** ITS COMPLETENESS VALUE AND DROP THE VARIABLES TO THE RIGHT
C 85 JSUL(J)=-1
     JS(J)=-JS(J)
     IF(J.EQ.MM) GO TO 52
     IBC=J+1
     DO 90 K = IBC, JRNE
        JS(K)=0
90    JSUL(K)=0
   GO TO 52
92 JS(JRME)=-JS(JRME)
     JSUL(JRME)=-1
   GO TO 52
C C *** CHECK TO SEE IF ANY VARIABLE CAN BE ADDED INTO CURRENT PARTIAL
C *** SOLUTION; IF NO, JUMP TO 74 TO START BACKTRACKING. IF YES,
C *** IDENTIFY THEIR COLUMN ID , ADD THEM IN CURRENT PARTIAL SOLUTION
C *** AND START ANOTHER EXAMINATION PROCESS BY GOING TO 52.
C 100 DO 102 J=1, MM
102 JS(J)=1
   IF(IDBG.EQ.3) WRITE(IOUT,26) CJSUM,CFSUM, ((Y(K),K=1,MM)
   IF(IDBG.EQ.3) WRITE(IOUT,27) CJSUM,CFSUM, ((JS(K),K=1,MM)
   IF(IDBG.EQ.3) WRITE(IOUT,29) CJSUM,CFSUM,(JSUL(K),K
1 =1,MM)
27 FORMAT (IX,5I, ' CJSUM,CFSUM,AND JS ',21I0, 8I3)
   DO 106 J=1, JRNE
IF(JS(J).GE.0) GO TO 104
JNS(1)=-JS(J)
NS(JNS(1))=0
GO TO 106
104 IF(JS(J).EQ.0) GO TO 106
IABC(JS(J))
NS(IABC(1))=0
106 CONTINUE
C
DO 120 I=1,WM
IF(NS(I).LE.0) GO TO 120
ICZC(C(I))=CJSUM
IF(IICZC.LT.ZKII) GO TO 120
NS(I)=0
120 CONTINUE
C
DO 122 I=1,WM
122 NS(I)=1
DO 140 I=1,WM
IF(Y(I).GE.0) GO TO 140
DO 136 J=1,WM
IF(A(I,J).LE.0) GO TO 136
NS(J)=0
135 CONTINUE
140 CONTINUE
C
DO 150 J=1,WM
IF(NS(J).LE.0) GO TO 150
IF(NS(J).LE.0) GO TO 150
NS(J)=0
150 CONTINUE
C
DO 160 J=1,WM
160 IF(NS(J).GT.0) GO TO 167
GO TO 74
167 MARKP=0
DO 183 I=1,WM
IF(Y(I).GE.0) GO TO 183
SUNCR=0
APOS=0
IABC=NR(I)
C
DO 180 I=1,IABC
IAB=JR(I,N)
IF(NS(IABC).LE.0) GO TO 180
IAB=JR(I,N)
SUNCR=SUNCR+C(IABC)
APOS=APOS+A(I,IABC)
JU=SUNCR+CJSUM
IAPY=APOS+Y(I)
IF(JU.GE.ZKII) GO TO 177
IF(IDBG.EQ.3) WRITE(IOUT,30) JU,ZKII
30 FORMAT(I1,5X,' JU VALUE ',I15,' DB. F. VALUE',I15)
C
IF(IAPY.GE.0) GO TO 181
GO TO 180
177 IF(IAPY.LE.0) GO TO 74
GO TO 183
180 CONTINUE
C
GO TO 74
C---------------
C 181 IF(IAPIY.GT.0) GO TO 183
IF(K.GE.NA(I)) GO TO 182
IABC=NA(I)
C
NI = I + 1
DO 174 K=NI,IABC
TAB=JR(I,K)
IF(NS(IAB).GT.0) GO TO 183
174 CONTINUE
C 182 MARF=1
183 CONTINUE
IF(MARF.GT.0) GO TO 240
DO 210 J=1,NN
IF(NS(J).LE.0) GO TO 210
YASUN=0
DO 198 I=1,NN
IYAX=Y(I)+A(I,J)
IF(IYAX.GT.0) GO TO 198
YASUN=YASUN+Y(I)+A(I,J)
198 CONTINUE
C 28 FORMAT(1X,5X, 'CJSUJ,CFSUJ,AID V ARRAY ',218, 8I10)
C
V(J)=YASUN
210 CONTINUE
C
DO 230 IY=1,NN
J=IY
IF(NS(J).LE.0) GO TO 230
IF(NN.LE.J) GO TO 235
L=J+1
C
DO 226 K=L,NN
IF(NS(K).LE.0) GO TO 226
IF(V(J).LT.V(K)) GO TO 230
226 CONTINUE
C
DO 229 I=L,NN
IF(NS(I).LE.0) GO TO 229
IF(V(J).LT.V(I)) GO TO 229
IF(C(J).LE.C(I)) GO TO 229
J=I
229 CONTINUE
C
GO TO 235
230 CONTINUE
C 235 JS(JBNE+1)=J
C IF(IDBG.EQ.3) WRITE(IOUT,28) CJSUJ,CFSUJ, (V(K),K=1,NN)
C IF(IDBG.EQ.3) WRITE(IOUT,31) MARF,YASUN, (NS(K),K=1,NN)
31 FORMAT(1X,6X, 'MAREF,YASUN AND NS ARRAY ',217,8I10)
C*****
GO TO 52
C 240 CONTINUE
DO 241 J=1,NH
   MF(J)=0
   C
   DO 250 I=1,NH
   IF(Y(I).GE.0) GO TO 250
   APOS=0
   DO 248 J=1,NH
   IF(NS(J).LE.0) GO TO 248
   IF(A(I,J).LE.0) GO TO 248
   APOS=APOS+A(I,J)
   CONTINUE
   C
   IF(MAP.APOS+Y(I)) GO TO 250
   DO 255 K=1,NH
   IF(NS(K).LE.0) GO TO 255
   IF(A(I,K).LE.0) GO TO 255
   MF(K)=1
   CONTINUE
   250 CONTINUE
   255 CONTINUE
   C
   CFSUM=0
   DO 262 I=1,NH
   IF(MP(I).LE.0) GO TO 262
   CFSUM=CFSUM+C(I)
   CONTINUE
   C
   KCCZ=CJSUM+CFSUM
   IF(IDBG.EQ.3) WRITE(IOUT,26) CJSUM,CFSUM,KCCZ
   IF(KCCZ.GE.ZKII) GO TO 74
   IF(NS.GT.1) GO TO 265
   J=1
   GO TO 267
   265 J=JNH+1
   C
   267 DO 268 I=1,NH
   IF(MF(I).LE.0) GO TO 268
   JS(I)=I
   J=J+1
   CONTINUE
   GO TO 52
END
SELECTED BIBLIOGRAPHY


