INFORMATION TO USERS

This manuscript has been reproduced 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.

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

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700  800/521-0600
A TABU SEARCH METHOD FOR A

TACTICAL FOREST PLANNING PROBLEM

BY

EVELYN WINNIFRED RICHARDS

A Thesis Submitted to the

Faculty of Engineering

In Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Industrial Engineering

Approved:

Dr. Eldon A. Gunn

Dr. Carl-Louis Sandblom

Dr. Tao Yang

Dr. Harvey Millar

Dr. Michel Gendreau

TECHNICAL UNIVERSITY OF NOVA SCOTIA

Halifax, Nova Scotia

1997
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
"AUTHORITY TO DISTRIBUTE MANUSCRIPT THESIS"

A Tabu Search Method for a Tactical Forest Planning Problem

The above library may make available, or authorize another library to make available, individual photo/microfilm copies of this thesis without restrictions.

Full Name of Author: Evelyn Winnifred Richards

Signature of Author: 

Date: 

ii
DEDICATION

This thesis is dedicated to Gerald Bance
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>XIV</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>XVIII</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>XIX</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>XX</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>XXI</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1. THE PLANNING PROBLEM AND MODEL</td>
<td>1</td>
</tr>
<tr>
<td>1.2. THESIS ORGANIZATION</td>
<td>6</td>
</tr>
<tr>
<td>CHAPTER 2 OVERVIEW OF FOREST MANAGEMENT ISSUES</td>
<td>8</td>
</tr>
<tr>
<td>2.1. ECONOMIC AND SOCIAL IMPORTANCE OF FORESTS</td>
<td>8</td>
</tr>
<tr>
<td>2.2. MULTIPLE OWNERS, DECISION-MAKERS AND GOALS</td>
<td>9</td>
</tr>
<tr>
<td>2.3. CHARACTERISTICS OF FOREST MANAGEMENT MODELS</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1. Forest inventories and GIS Systems</td>
<td>13</td>
</tr>
<tr>
<td>2.3.2. Growth Models: Predicting Future Inventory</td>
<td>16</td>
</tr>
<tr>
<td>2.4. HIERARCHICAL PLANNING</td>
<td>18</td>
</tr>
<tr>
<td>2.4.1. Forest Management Hierarchy</td>
<td>19</td>
</tr>
<tr>
<td>2.4.2. Hierarchy of Planning Models</td>
<td>21</td>
</tr>
</tbody>
</table>
2.4.2.1.  Strata-Based Models ................................................. 22

2.4.2.2.  Spatially Explicit Models for Tactical Planning ................. 25

2.5.  Cohesion Between Strategic and Tactical Models ....................... 29

CHAPTER 3  TACTICAL PLANNING MODELS AND SOLUTION METHODS .... 31

3.1.  Solving Tactical Planning Models ..................................... 32

3.2.  Spatial Decision Units ................................................... 34

3.3.  Adjacency Constraints ................................................... 36

3.3.1.  Strong Valid Inequalities and Cutting Planes Algorithms ...... 38

3.3.2.  The Adjacency Problem: Node Packing ........................... 40

3.3.2.1.  Clique Inequalities ................................................. 41

3.3.2.2.  Odd Hole, Odd Anti-Hole Inequalities ......................... 41

3.4.  Generating Clique Constraints .......................................... 42

3.4.1.  Generation Procedure ................................................ 42

3.4.2.  Bloedel Farm Case Study .............................................. 43

CHAPTER 4 ................................................................. 47

TABU SEARCH ........................................................................ 47

4.1.  Basic Tabu Search Algorithm ............................................. 47

4.1.1.  Strict and Fixed TS, Tabu Tenure ..................................... 49

4.1.2.  Aspiration Criteria ....................................................... 50

4.1.3.  Summary of the Basic TS Structure .................................. 51

4.2.  Reduced Neighborhood Search ............................................. 52

4.3.  Move Evaluation, Intensification, Diversification ..................... 52

4.4.  Dynamic Rules for Evaluation and Prohibition ....................... 55

V
4.5. REACTIVE TABU SEARCH .................................................. 57
4.6. STRATEGIC OSCILLATION ................................................. 59

CHAPTER 5 HARVEST SCHEDULING AND
ROAD BUILDING PROBLEM (HSRBP) ........................................ 61

5.1. MODEL ASSUMPTIONS .................................................. 61
5.2. STANDS AND OPENINGS ............................................. 64
5.3. LOST VOLUME PENALTY .............................................. 68
5.4. HARVEST VOLUME RESTRICTIONS ................................. 71

CHAPTER 6 ROADING NETWORKS ............................................ 74

6.1. CREATING THE PROPOSED ROAD NETWORK .................. 74
6.2. GRAPH REPRESENTATION OF THE ROADING NETWORK ...... 76
6.3. INTEGER PROGRAMMING FORMULATION FOR RNP(x) ...... 78
6.4. EQUIVALENCY TO STEINER TREE PROBLEM .................. 80

6.4.1. The Steiner Tree Problem on Graphs ......................... 81
6.4.2. RNP(x) as a Steiner Tree Problem ............................. 82
6.4.3. Solving Steiner Tree Problems .................................. 83
6.4.4. Differences between SRNP(x) and RNP(x) ..................... 85

6.5. SOLVING RNP(x) .......................................................... 86

6.5.1. Path Heuristic ...................................................... 87
6.5.2. Computational Complexity ....................................... 88
6.5.3. Numerical Results ................................................ 89

CHAPTER 7 TABU SEARCH ALGORITHMS FOR HSRBP .................. 92

7.1. SEARCH ALGORITHM STRUCTURE .................................. 93
7.2. **Moves and Neighbourhood Search** ................................................................. 96

7.2.1. *Move Types* ................................................................................................. 96

7.2.2. *Tabu Status and Aspiration Criteria* ......................................................... 97

7.2.3. *Calculating Change in Objective Function* ................................................ 97

7.2.3.1. Loss Penalty ............................................................................................... 98

7.2.3.2. Roading Cost ............................................................................................. 98

7.2.3.3. Deviation from Maximum Opening Size ................................................... 100

7.2.3.4. Add Move .................................................................................................. 100

7.2.3.5. Delete Move .............................................................................................. 101

7.2.3.6. Swap Move ............................................................................................... 103

7.2.3.7. Harvest Target Deviations ......................................................................... 103

7.3. **Special Diversifying Moves** ................................................................. 103

7.4. **Fixed Tabu Search Algorithm (FTS)** .................................................... 106

7.5. **Oscillating Tabu Search Algorithm (OTS)** .............................................. 106

7.6. **Reactive Tabu Search Algorithms (RTS)** ............................................... 107

7.6.1. *RTS Data Structures and Cycle Detection* ............................................. 107

7.6.2. *Hashing Address Computation* ................................................................. 109

7.7. **Oscillating Reactive Tabu Search Algorithm (ORTS)** .......................... 110

**CHAPTER 8** **Empirical Results: Shulkell Study Region** ...................... 113

8.1. **Study Region and Model** .......................................................................... 113

8.1.1. *Characteristics of the Forested Stands* .................................................. 114

8.1.2. *Alternate Datasets* .................................................................................. 116

8.1.3. *Roading Network* .................................................................................... 117

8.1.4. *Model Parameters* .................................................................................. 118
8.2.  THE ALGORITHMS ........................................................................................................... 119
  8.2.1.  Fixed Tabu Search........................................................................................................ 119
  8.2.2.  Fixed Tabu Search with Random Diversification Moves (FTSESC) ..................... 120
  8.2.3.  Reactive Tabu Search (RTS)....................................................................................... 121
  8.2.4.  Oscillating Tabu Search (OTS) ................................................................................ 122
  8.2.5.  Oscillating Reactive Tabu Search ............................................................................. 123

8.3.  TRADEOFF ANALYSIS ..................................................................................................... 127

CHAPTER 9  DISCUSSION......................................................................................................... 129

9.1.  SUMMARY ....................................................................................................................... 129

9.2.  DIRECTIONS FOR FURTHER RESEARCH ................................................................. 130
  9.2.1.  Integer Programming Formulations ......................................................................... 131
  9.2.2.  Stand types and GIS Management .......................................................................... 132
  9.2.3.  Productivity Loss Functions .................................................................................. 133
  9.2.4.  Incorporating Different Goals or Constraints .......................................................... 133
  9.2.5  Other Types Of Intervention and Road Network Extensions .................................. 134

REFERENCES ............................................................................................................................ 136

APPENDIX A  FLOWCHARTS ................................................................................................. 147

APPENDIX B  DATASETS ......................................................................................................... 154

APPENDIX C  FORTRAN CODE ............................................................................................ 162

  COMMON BLOCKS ................................................................................................................ 162
    Parameters and Constants .................................................................................................. 162
    best_move ....................................................................................................................... 162

viii
subroutine merge_opens2(open_list,a_count,stand,o_period) .................................................. 178
subroutine make_new_open(stand,period,o_period) ............................................................... 179
subroutine add_stand2(stand,period) ..................................................................................... 179
subroutine findadj(vlook,o_adj,vadj,val) ............................................................................. 180

MOVE CALCULATIONS ............................................................................................................ 181
subroutine calc_full_nbhood6 .............................................................................................. 181
subroutine calc_obj_change5 ................................................................................................. 183
subroutine calc_ass_penalty(stand,period,ass_penalty) ...................................................... 185
subroutine calc_unass_penalty(unass_penalty) ...................................................................... 185
subroutine calc_volume(stand,period,volume) ...................................................................... 186
subroutine calc_objective ...................................................................................................... 187
subroutine calc_base_loss ..................................................................................................... 188
subroutine calc_dev_fr_maxopen(stand,period,type) ........................................................... 188
subroutine calc_add_stand(stand,period,dev_penalty) ......................................................... 189
subroutine calc_new_open(stand,open_no,dev_value) .......................................................... 189
subroutine calc_rem_stand(stand,opening,dev_penalty) ...................................................... 189
subroutine calc_opens(test_dev_fr_maxopen) ...................................................................... 189
subroutine calc_feas_for_maxopen(stand,period,type) ......................................................... 189
subroutine calc_r_cost_sng(link,cost,l_period) ..................................................................... 189
subroutine calc_road_cost .................................................................................................... 189

MOVE IMPLEMENTATION ..................................................................................................... 189
subroutine implement_best_move2 ....................................................................................... 189
subroutine implement_stand_move2 ...................................................................................... 189
subroutine do_this_move2 ..................................................................................................... 189
subroutine update_roads ...................................................................................................... 189
subroutine re_time .................................................................................. 189
subroutine get_l_p_stands ..................................................................... 189
subroutine get_l_p_prev(link_to_change,link_period_prev) .................. 189
subroutine get_prev_path(link_to_change,p_link.path_period) .......... 189
subroutine get_all_links(link_list,link_len) ....................................... 189
subroutine stand_escape_route ............................................................ 189
subroutine update_best ........................................................................ 189
subroutine rts_escape_route2 ............................................................... 189
subroutine choose_a_link(link,iperiod) ............................................ 189
subroutine new_escape_route ............................................................... 189
subroutine link_escape ......................................................................... 189

UTILITY ROUTINES AND FUNCTIONS ................................................... 189

subroutine re INSTALL ................................................................. 189
subroutine write_sol(icount) ............................................................. 189
subroutine write_data_rts(icount) ..................................................... 189
subroutine save_init_sol ................................................................. 189
subroutine get_init_sol ................................................................. 189
subroutine init_best_values ............................................................. 189
subroutine write_iter ......................................................................... 189
subroutine init_move_counters2 ....................................................... 189
subroutine update_move_counters2 .................................................. 189
subroutine update_obj_coeff ............................................................ 189
subroutine init_x ............................................................................. 189
subroutine init_stands ....................................................................... 189
subroutine init_opens ....................................................................... 189
subroutine get_data(dataset) ......................................................... 189
subroutine calc_parameters .......................................................... 189
subroutine init_move_values ......................................................... 189
subroutine init_search_params ..................................................... 189
subroutine init_objective ............................................................ 189
subroutine save_sol ................................................................. 189
subroutine clear_solution ........................................................... 189
subroutine time_stats(elapsed) ...................................................... 189
subroutine read_growth ............................................................. 189

ROAD SUBROUTINES ..................................................................... 189

subroutine init_roads ................................................................. 189
subroutine build_roads ............................................................... 189
subroutine make_st_list(link_point,stand_list) ............................... 189
subroutine read_road ................................................................. 189
subroutine const_road_heap ......................................................... 189
subroutine pqdownheap(k) ............................................................ 189
subroutine pqremove(k) .............................................................. 189
subroutine pqchange(t,new_pr) ..................................................... 189
subroutine pqupheap(k) .............................................................. 189
subroutine calc_retine_cost(link_to_change) .................................. 189

RTS SUBROUTINES ..................................................................... 189

subroutine check_for_reps(escape) ................................................ 189
subroutine init_reactive ............................................................. 189
subroutine del_one_link ............................................................. 189

TABU SUBROUTINES .................................................................. 189

xii
subroutine init_tabu_rts ................................................................. 189
subroutine get_aspiration2(asp) ...................................................... 189
subroutine check_tabu(tabu)     'move tabu? .................................. 189
subroutine tabu_add(std) ................................................................ 189

APPENDIX D  NUMERICAL RESULTS .................................................. 189
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Hierarchical Context of the Planning Problem</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Low Road Cost Solution</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>High Road Cost Solution</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>Efficient Frontier</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Using GIS in Forestry</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Relational Database Query</td>
<td>15</td>
</tr>
<tr>
<td>2.3</td>
<td>Spatial Analysis Using ArcView™</td>
<td>16</td>
</tr>
<tr>
<td>2.4</td>
<td>Hierarchy of Forest Planning</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Strata-Based Model</td>
<td>22</td>
</tr>
<tr>
<td>2.6</td>
<td>FORPLAN Model</td>
<td>24</td>
</tr>
<tr>
<td>2.7</td>
<td>Model II Formulation</td>
<td>25</td>
</tr>
<tr>
<td>2.8</td>
<td>Tactical Planning Framework</td>
<td>26</td>
</tr>
<tr>
<td>2.9</td>
<td>Adjacency Requirement</td>
<td>28</td>
</tr>
<tr>
<td>3.1</td>
<td>Adjacency Graph</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>Odd Hole and Anti-Hole</td>
<td>42</td>
</tr>
<tr>
<td>3.3</td>
<td>Algorithm to Generate All Cliques in a Sparse Graph</td>
<td>43</td>
</tr>
<tr>
<td>4.1</td>
<td>Tabu Search Algorithm</td>
<td>51</td>
</tr>
<tr>
<td>5.1</td>
<td>Multi-Period Openings</td>
<td>67</td>
</tr>
<tr>
<td>5.2</td>
<td>Mean Annual Increments</td>
<td>70</td>
</tr>
<tr>
<td>5.3</td>
<td>Lost Volume Function, Growing Stands</td>
<td>70</td>
</tr>
<tr>
<td>5.4</td>
<td>Lost Volume Function, Dead Stands</td>
<td>71</td>
</tr>
</tbody>
</table>
LIST OF TABLES

TABLE 3.1. BLOEDEL PROBLEM RESULTS .............................................................................. 46

TABLE 8.1. SHULKELL REGION LAND TYPES ................................................................... 126

TABLE 8.2. DISTRIBUTION OF FORESTED STANDS BY AGE AND MERCHANTABLE VOLUME .............................................................................. 115

TABLE 8.3. STANDS BY COVER TYPE .............................................................................. 116

TABLE 8.4. SHULKELL STANDS BY STAND TYPE .............................................................. 116

TABLE 8.5. MODEL PARAMETERS ................................................................................. 126

TABLE 8.6. FTS SOLUTIONS .......................................................................................... 120

TABLE 8.7. FTSESC SOLUTIONS .................................................................................... 121

TABLE 8.8. PERCENTAGE DIFFERENCE IN FTS OVER FTSESC ........................................ 121

TABLE 8.9. REACTIVE TABU SEARCH RESULTS .............................................................. 122

TABLE 8.10. OTS STATISTICS ....................................................................................... 124

TABLE 8.11. ORTS ALGORITHM FEATURES ..................................................................... 125

TABLE 8.12. ORTS SAMPLE RESULTS ............................................................................. 126

TABLE 8.13. COEFFICIENT OF VARIATION ..................................................................... 126

TABLE D.1. STATISTICS FOR SHULKELL DATASET ................................................................ 189

TABLE D.2. STATISTICS FOR DATASET 1 ................................................................................. 189

TABLE D.3. STATISTICS FOR DATASET 2 ................................................................................. 189

TABLE D.4. STATISTICS FOR DATASET 3 ................................................................................. 189

TABLE D.5. STATISTICS FOR DATASET 4 ................................................................................. 189

TABLE D.6. STATISTICS FOR DATASET 5 ................................................................................. 189
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ac.</td>
<td>acre</td>
</tr>
<tr>
<td>dbh</td>
<td>diameter breast height</td>
</tr>
<tr>
<td>ft.</td>
<td>feet</td>
</tr>
<tr>
<td>ft$^3$</td>
<td>cubic feet</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographic Information System(s)</td>
</tr>
<tr>
<td>ha</td>
<td>hectare</td>
</tr>
<tr>
<td>HSRBP</td>
<td>Harvest Scheduling and Road Building Problem</td>
</tr>
<tr>
<td>IP</td>
<td>Integer Program</td>
</tr>
<tr>
<td>IRPM</td>
<td>Integrated Resource Planning Model</td>
</tr>
<tr>
<td>km</td>
<td>kilometre</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Program</td>
</tr>
<tr>
<td>m.</td>
<td>metre</td>
</tr>
<tr>
<td>m$^3$</td>
<td>cubic metre</td>
</tr>
<tr>
<td>mai</td>
<td>mean annual increment</td>
</tr>
<tr>
<td>MCIP</td>
<td>Monte Carlo Integer Programming</td>
</tr>
<tr>
<td>MIP</td>
<td>Mixed Integer Program</td>
</tr>
<tr>
<td>PH</td>
<td>Path Heuristic</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computer</td>
</tr>
<tr>
<td>RNP</td>
<td>Road Network Problem</td>
</tr>
<tr>
<td>SA</td>
<td>Simulated Annealing</td>
</tr>
<tr>
<td>SAWWS</td>
<td>Strategic Analysis of Wood Supply</td>
</tr>
<tr>
<td>SRNP</td>
<td>Steiner Road Network Problem</td>
</tr>
<tr>
<td>TS</td>
<td>Tabu Search</td>
</tr>
<tr>
<td>USDA</td>
<td>United States Department of Agriculture</td>
</tr>
<tr>
<td>VRP</td>
<td>Vehicle Routing Problem</td>
</tr>
<tr>
<td>WHR</td>
<td>Wildlife Habitat Relationship</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

This work would not have been possible without the contributions of many people. I was fortunate to have abundant sources of inspiration, and significant levels of financial and spiritual support. I would like to acknowledge the Nova Scotia Lands and Forests staff, especially Colleen Bowers (Truro office), for providing the GIS coverage of Cumberland county. Thanks are also due to Eric Robson for his assistance in interpreting the growth models, and to the Bible Hill office staff for helping to create the roading network. Financial support from the National Sciences and Engineering Research Council of Canada and the Province of New Brunswick Women's Doctoral Scholarship program, is gratefully acknowledged.

Thanks are due to my reading committee and external advisor for reading and criticizing the thesis. The faculty and graduate students of the Industrial Engineering department at TUNS have created a spirited, productive and congenial research environment which is very conducive to developing ideas and pursuing academic research, and I am indebted to all of them. I especially thank my supervisor, Dr. Eldon Gunn, for his intellectual inspiration and guidance, as well as his patience and diligence.

Finally, I will never forget that my family and many friends made it possible for me to complete this work by giving me emotional support and constant encouragement throughout the past four years.
ABSTRACT

At the tactical level of forest planning, decision support systems must deal explicitly with spatial and temporal restrictions on clearcuts and the design of road access to the forest. Formulating and solving optimization models which provide useful decision choices in this context is a serious challenge.

This thesis advances a new model formulation for the harvest scheduling and road building optimization problem. The tactical planning problem is treated in the context of a hierarchical planning system, where a strategic planning process has been first executed. The strategic plan goals of sustainable harvest are inputs to the tactical planning process, where the object is to produce a spatially and temporally explicit schedule of harvesting and road building. The model is designed to produce harvest schedules which minimize both biological productivity losses due to sub-optimal timing of harvests, and the costs of road construction. The objective function to be minimized is a weighted sum of these two opposing cost factors. Solving the optimization problem for a range of weightings produces a spectrum of solutions, from which non-dominated solutions are selected to produce an efficient frontier of roading cost versus lost productivity.

The model uses forest stands as spatial decision units for harvest scheduling, thus avoiding the reduction of the solution space which occurs when stands are pre-blocked. The model produces schedules which are compliant with maximum opening size and adjacency delay requirements, without restricting adjacency delay to one planning period. The configuration of the road system is not restricted to be a tree structure, in that multiple access points and cycles in the proposed network are permitted. As a result, this model can address a wider range of practical road network designs than other models.
in the literature. An heuristic algorithm was developed to solve the multi-period road network optimization problem. This algorithm was shown to provide solutions which consistently fall within a small percentage of the optimum solution, and which are optimal in the majority of cases.

Topographical adjacency relationships between stands are represented by a general undirected graph. Openings which are created by harvesting are connected sub-graphs of this forest graph. These sub-graphs are dynamically created throughout the solution process. Spatial feasibility of a proposed schedule is determined by utilizing depth first search and articulation point searches on these sub-graphs.

The combined harvest and road building problem is solved using a tabu search metaheuristic. The search algorithm has a dynamic feedback mechanism to set tabu tenure, and a strategic oscillation strategy to smooth transitions throughout the feasible region. These features of the algorithm essential to ensure solution quality over many problem instances. Computational studies were carried out on a forested region in Cumberland County, Nova Scotia. The study results demonstrate consistent results over the original dataset and five additional datasets. Tradeoff curves which are produced from these solutions provide valuable information, clearly showing the relationship between road building budgets and the range of harvest timing choices available to the decision maker. Thus, this model and solution algorithm represent a significant contribution to the tactical forest management problem.
CHAPTER 1

INTRODUCTION

Forests are a valuable resource. In addition to wood and mineral products which are economically consequential, forests provide habitat for a multitude of animal and plant species, 'facilities' for recreational and sporting activity, and are a carbon sink and a source of oxygen, essential to balance these elements of the earth's atmosphere. Scientific management of our forest resources is an important endeavour, and has in recent years become the focus of operations research activities, aimed at developing better models and solution methods to represent the diverse requirements and concerns of all decision-makers and owners.

1.1. The Planning Problem and Model

This thesis addresses the optimization of harvest and road building schedules in decision support systems for forest management. The problem, a component of the tactical level of the hierarchical planning framework, is to produce an optimal, or near-optimal, and spatially explicit plan that is consistent with management goals derived from a strategic planning process. The optimization model is formulated to determine minimal cost schedules that implement elements of strategic planning (such as harvest levels), and that are in compliance with environmental regulations. Costs include economic factors, such as the expenditures required for road construction. They also include factors which penalize inappropriate timing of harvests, which will negatively impact the future productivity of the forest.

In this hierarchical context, it is assumed that the strategic planning exercise has used a long planning horizon and, amongst other goals, has enforced the requirement that
production levels be consistent with sustained yield. One result of such a process is the recommendation of achievable levels of sustainable wood harvest over some longer planning horizon. The tactical planning problem is to produce a spatially explicit schedule of stand harvests and road building (Figure 1.1). This schedule must meet the strategic goals on volumes. It is further constrained to meet maximum opening size restrictions and adjacency delay constraints.

![Diagram](Image)

**Figure 1.1. Hierarchical Context of the Planning Problem**

In many tactical planning optimization models in the literature, the objective function to be maximized is the net present value accruing from the periodic flows of wood products. A different approach is used in this model. It is known that harvest levels, which meet strategic goals, have already been determined in the aspatial model subject to suitable constraints. Thus, levels of production are already optimized. The objective of this model is then restricted to minimizing the costs of road construction and negative impacts on forest productivity due to timing of harvests, while being
constrained to meet the recommended production levels for each period. This approach provides new information to the decision-maker which is not obtained by other models.

Costs in future productivity are induced by harvesting stands at other than their period of peak biological productivity. Roads must be built to gain access to stands and transport timber to main roads. The objective function is a penalty function, comprised of two weighted penalty terms, lost volume penalty and roading cost. The lost volume penalty assesses a cost for timing the harvest of stands at other than their peak productivity. The roading cost estimates the cost to access stands scheduled for harvest on a pre-designed set of road links. Clearly, roading costs and lost productivity costs are in opposition: reducing lost productivity involves increasing the cost of road building (See Figures 1.2 and 1.3). Moreover, the units of measure are incompatible. This precludes formulating any one cost function that is meaningful across different problem sets and decision makers. The model objective function is a parameterized sum of the two cost factors, and varying the parameter influences the system to produce solutions across a range of road building costs. The non-dominated solutions then are recorded in two-dimensional attribute space to produce an efficient frontier of choices between road building costs and lost productivity costs. This trade-off curve quantifies the real trade-off relationship between road cost and productivity cost (Figure 1.4).
Figure 1.2. Low Road Cost Solution.
RC is $24,840 and lost volume 563,444 ft³.

Figure 1.3. High Road Cost Solution.
RC is $99,132 and lost volume 306,526 ft³.

Both stand adjacency relationships and road network feasibility are modeled using
graphs. The road network model incorporates multiple access points and a general graph which is not restricted to have a tree structure. Maximum opening size and adjacency delay requirements are represented as conditions on certain specially defined subgraphs of the forest adjacency graph. The combined harvesting and road building model is solved using a Tabu Search metaheuristic method. The road network problem is a sub-problem in the algorithm, and is solved at each step using a custom heuristic method.

An ancillary component of this thesis is the clarification of some issues in integer programming model formulations of the block-based harvest scheduling problem. An algorithm for generating constraints to tighten Linear Programming relaxations for this problem is given, and some empirical results on a well known dataset are reported.

![Efficient Frontier](image-url)

Figure 1.4. Efficient Frontier
1.2. Thesis Organization

To set the context for this work and introduce terminology and relevant technologies, Chapter 2 is an overview of modern forest management planning systems. Some of the unique characteristics of modern forest management planning processes are described. The fitness of forest management for hierarchical organization is well accepted, and this context is assumed in this thesis. Strategic and tactical processes, their inputs and outputs and some of the well-known optimization models for these systems are described in section 2.4. The nature of the feedforward and feedback systems between these two levels is a matter of much concern; these issues are covered in the final section.

Chapter 3 concentrates on tactical planning systems, and particularly on modeling issues which arise when the planning problem is both spatially and temporally specified. Optimization models for forest tactical planning problems are notoriously difficult to solve, hence various heuristic strategies and specialized solution procedures have been developed. The choice of spatial decision units is a well-discussed issue amongst planners and modelers. For models which produce harvest block- or polygon-based schedules, a considerable effort has been made by many researchers to simplify the methods used to specify adjacency constraints. Sections 3.3 is a discussion of the implications of adjacency constraint specification on the solution of such models by branch and bound and cutting planes algorithms. A simple method of specifying some strong inequalities is given in section 3.4, and computational results on a well-known dataset are used to emphasize the usefulness of this formulation.

The model which is proposed in this thesis is solved using a Tabu Search procedure. Accordingly, chapter 4 is devoted to a detailed account of the Tabu Search metaheuristic method, its known successes and some of its more sophisticated
implementations.

Chapter 5 describes the harvest scheduling model. Graph structures are used to model adjacency and maximum opening restrictions. The lost volume penalty function which measures the negative effects of some harvest timing choices is described, and the harvest model parameters and constraints are specified.

The forest road network design problem is documented in Chapter 6. Its mathematical model is based on graphs, and it is shown to be NP-hard by demonstrating equivalency to the Steiner problem in graphs. The Path Heuristic which is used to solve the roading problem is then developed. This heuristic is shown to produce near optimal solutions on some specialized graphs.

Chapter 7 describes the tabu search algorithm for solving the combined harvest scheduling and road building problem. Several tabu search procedures were tested and compared. Chapter 8 contains details of the study area and empirical results. Finally, chapter 9 contains a discussion of the implications of this thesis and some recommendations for further work.
CHAPTER 2

OVERVIEW OF FOREST MANAGEMENT ISSUES

The purpose of this chapter is to provide background in forest management issues to set the context for the tactical planning model of this thesis. The significance of forests and forestry for our society is discussed, and some emerging trends in forest management that encompass the needs and goals of multiple decision makers and forest owners are presented. Technologies which are important in their support of the forest management decision making process include Geographic Information Systems (GIS), growth and yield modeling and simulation. The hierarchical planning system context for forest management is introduced, with an overview of the types of modeling issues which arise in strategic and tactical planning. Finally, a summary of some of the connectivity issues relating to these models is given.

2.1. Economic and Social Importance of Forests

The forest products industries are a cornerstone of the Canadian economy. In 1994, forestry generated $48 billion in shipments, $32 billion of which were exports. In fact, Canada is the world's largest exporter of forest products, having an 18% share of world trade. Forest industries account for 14% of Canadian manufacturing gross domestic product, and are the major or sole employer in more than 350 rural communities [Industry Canada 1996a]. In the maritime provinces, forestry is the most economically consequential sector. Forest industries account for one job in twelve in New Brunswick, and one in twenty-four in Nova Scotia [Industry Canada 1996b, 1996c].
Forests are much more than a source of supply for wood and paper products. Canadian forests account for almost 10% of the earth's forests, which are a vital part of our planet's life support system [Murphy et al 1993]. Forests are the habitat of a multitude of animal and plant species. Healthy forests are required in maintaining a clean water supply. They provide "facilities" for recreational and sporting activity and a carbon sink for the earth's atmosphere.

Continued improvements in technology have increased the efficiency with which forest products are produced, and some recent attention to re-cycling paper products will no doubt result in decreased pressure on raw materials. However, demands for paper and wood products are increasing with a world population that is growing in both size and sophistication. It seems unlikely that improvements in manufacturing technologies will be sufficient to meet the future demand for forest products, without seriously depleting the resource or compromising the environment. Effective management of forest resources is critical to avoid making environmentally or economically catastrophic decisions.

New knowledge of the impact of human actions in forests on the global environment is gained daily. Synthesis of this knowledge into management practices, however, is a much slower and more difficult process. The development of intelligent systems to manage forests so that both economic and ecological goals are achieved with any degree of certainty is in its infancy.

2.2. Multiple Owners, Decision-Makers and Goals

A distinguishing and complicating feature in forest management is the existence of multiple "owners" of the forest. In Canada, 91% of the forest area is publicly owned and controlled. The remainder is owned by several hundred thousand private woodlot
owners [Murphy et al 1993]. On public lands, control lies in the jurisdiction of Provincial Governments, although private enterprise may propose and carry out management of these resources [NBOGVT 1994]. In effect, governments act as stewards of the resource for the Canadian people. These governments lose their jurisdiction at national borders; the care and nurturing of the earth does not segregate at these artificial boundaries. Considering the vital importance of forests to the planet's ecological health, one may say that all living species of the earth have title to its forests, and that the management of forest ecosystems is a matter of public trust, regardless of ownership [Davis and Barrett 1992].

Thus, the register of potential participants in forest management ranges from large governmental agencies (national and international) and forest products industries to individual woodlot owners; from communities which depend on the local mill for their economic viability to scientific and environmental advocates for species habitat; from organized industrial lobbies to individual hikers, hunters and campers. The goals and desires of these participants are defined within different frames of reference and are often in conflict. Forest management decision-making processes must be capable of integrating these diverse groups of decision makers with their often conflicting objectives [Weintraub and Davis 1996].

Forest products industries must be economically viable, and thus have a dominant goal which is to maximize profits. Thus, they plan to extract products from the forest to meet their markets and to minimize their operational costs. Governments recognize that local economies may depend on the forest products industry. To maintain a degree of social equilibrium, it is desirable that the industry operate in a sustainable fashion, so that communities are not disrupted by swift depletion of forest resources. This objective has been expressed as the requirement for non-declining yield (harvest in any period must be
at least equal to that from the previous period) and as a reserve margin (a percentage of the final period harvest remaining in inventory). This concept of sustainable forestry is now being expanded or changed to address the sustainability of forest ecosystems [Thompson and Welsh 1993, Booth et al 1993, Dewhurst et al 1995].

Ecological considerations include biodiversity, green cover requirements and water quality and supply. Achieving and maintaining a diverse wildlife population is dependent on, amongst other factors, the availability of forest habitat to support a broad spectrum of plant and animal species. Habitats that are suitable for nesting and breeding, foraging and migration must be available to support wildlife. The choice of interventions (such as clearcutting, shelterwood cutting, thinnings and cleanings), as well as the level and location of such activities, have an obvious impact on habitat. Clearcuts remove mature forest stands, and create edge-type habitat. The spatial arrangement of interventions is important in preserving existing habitat and in creating a diverse environment [Booth et al 1993]. Thus, legislators have initiated restrictions on cutting, such as the maximum opening size and adjacency delay, which are intended to both preserve mature areas and to create new openings which are necessary for foraging and breeding. Appropriate management of waterways and watersheds must be included in planning so as to safeguard the future water supply. Road building causes sedimentation and must therefore be carefully managed so as to preserve the quality of streams and brooks.

Recreational users require hiking and climbing areas, fishing and hunting areas and scenic areas. This is usually achieved by setting aside regions of the forest for recreational use. "Set-asides" are also specified when a particular type of habitat, such as deer wintering yards or old-growth forest, are to be specifically preserved. The choice of what area and how much area to allocate to recreational use and to preservation is a
continuing source of conflict between recreational users, environmental advocates and commercial users.

2.3. Characteristics of Forest Management Models

Scientific management of our forest resources is crucial, and has in recent years become the focus of operations research activities, aimed at developing better models and solution methods to represent the diverse requirements and concerns of all owners of our planet's forests. The forest management problem has many dimensions. Users have diverse and sometimes conflicting goals, ranging from extraction of wood products to preservation of wildlife habitats. Biological scientists strive to understand forest growth and the life patterns of its inhabitants, and to define measures of desirable forest states such as biodiversity. The prediction of forest growth is further complicated by the occurrence of natural events such as fire, pest infestation and disease. Decision makers may be federal, provincial or municipal governments, or individuals who own land. Planning takes place at strategic, tactical and operational levels, requiring telescoping degrees of spatial resolution and detail in planning models.

Early forest management mathematical models concentrated on producing recommendations for harvest levels that would maximize yield while maintaining long-term sustainability of the resource. More comprehensive management models are now needed to address the requirements of all users -- the forest products industry, recreational users, and environmentalists. This presents a significant challenge to researchers to define appropriate model formulations, and to develop new methods of solving these complex models. The availability of a comprehensive and accessible forest inventory is one key to the success of modern management systems.
2.3.1. Forest inventories and GIS Systems

The forest inventory consists of several levels of information. Biological data regarding the forested areas, the location of waterways, roads, and ownership boundaries are all required data for forest management.

The basic spatial unit for forested areas is the stand, which is a contiguous region that is homogeneous in species composition and future expected growth characteristics. Stands may be delineated by age; thus "even-aged" stands are also homogeneous in age-class or diameter-breast-height (dbh). "Uneven-aged" stands, inhomogeneous in age-class, are further specified by calculating the distribution of age-classes (dbh class or basal area class) of trees in the stand area.

A Geographic Information System (GIS) is a spatially referenced database management system. Information is stored as a collection of thematic layers which are geographically referenced [Environmental Systems Research Institute1997]. Forest management has a critical geographic dimension: management choices include not only what action is selected but where (and when) it is to be implemented [Jordan 1993]. Thus, a GIS is a natural choice for gathering, maintaining, updating and displaying inventory for the forest management process. (Figure 2.1) GIS, a relatively new technology, has had a real impact on forest management in improving the speed, flexibility and accuracy with which such data can be stored and manipulated.
A GIS includes a set of generic tools which can be customized for forest management purposes [Jordan 1993]. Relational database functionality allows selection and grouping of spatial features by attributes which are stored in tables. For example, Figure 2.2 shows the distribution of budworm damaged stands in a forested area. This figure was generated by querying a forestry GIS for stands matching this classification. Spatial analysis, such as proximity analysis, allows the user to extract information on stands based on their spatial relationship to other features. As an example, Figure 2.3 shows stands which are adjacent to main roads, and was generated using ARCVIEW™'s spatial analysis capability. Use of the database and analysis tools ranges from simple "point and click" usage to complex programmed algorithms.

GIS has had a large impact on forestry planning because of its visual nature: mapped "what-if" management scenarios or their predicted results provide a common meeting point for decision-makers from widely varying backgrounds. Thus, GIS systems
allow disparate groups of users to communicate effectively and, for that reason alone, are advantageous to the planning process.

Forest management is having to adapt to radically different groups of decision makers, new goals and an evolving science. Moreover, GIS is a relatively new technology. Thus there are of course some challenges in implementing GIS technology. First, the effort in generating the initial database is enormous. Second, the database must be well designed so as to be comprehensive, flexible and extensible, since the nature of future demands on the system for information are not known in advance. Finally, a well managed system of updates is critical, since forest interventions are almost continuously changing the landscape.

Figure 2.2. Relational Database Query. Dark areas are Budworm Damaged Stands (ARCVIEW™ Image Source: Nova Scotia Dept. Lands & Forests GIS Database)
Figure 2.3. Spatial Analysis Using ARCVIEW™
Selection of Stands Adjacent to Main Roads
(Source: Nova Scotia Dept. Lands & Forests GIS Database)

Despite some problems, it is fair to say that GIS has had a major positive impact on forest planning systems and has removed several impediments to modern forestry management endeavours. It has provided the capability to develop a comprehensive spatially referenced database, an excellent communications tool and significantly increased analysis functionality [Davis and Barrett 1992, Ball 1994, Dewhurst et al 1993].

2.3.2. Growth Models: Predicting Future Inventory.

Models which predict the growth of trees are essential to forest management [Brack 1996a]. The goal of growth modeling is to predict the future condition of a stand using the current composition (i.e. the species, age, and stocking or density), any previous or planned silvicultural treatments and a measure of site capability or site quality. Due to the length of crop rotations in forestry, these predictions must be made decades into the future [Province of Nova Scotia 1993]. The value of a growth and yield
model lies in its ability to provide the best possible prediction of the relative outcomes of various management alternatives [Edwards and Christie 1981]. Data for growth and yield models may come from permanent sample plots, temporary sample plots and stem analysis [Brack 1996a]. Growth and yield models are thus location specific, since climate, soil and topology strongly affect forest growth. In Nova Scotia, the Department of Lands and Forests has developed growth and yield models for first and second rotation unmanaged stands and for managed stands. Data have been obtained from hundreds of permanent and temporary sample plots in plantations, pre-commercial thinnings, commercial thinnings, shelterwoods and unmanaged stands [Province of Nova Scotia 1993]. The future inventory is predicted in terms of total, merchantable and sawlog stand averages for each diameter, height and basal area class. The model is limited to eight softwood species: mixedwood and hardwood stand predictions are calculated as for softwood growth and are adjusted by volume factors.

The growth and yield model forms the basis for growth simulation, which is used to evaluate the effect of different management scenarios on the forest. In Nova Scotia, for example, the Strategic Analysis of Wood Supply (SAWS) system is a simulation model of even-aged forest management [Gunn 1994b]. Inputs to the simulation are the current inventory, levels for each potential treatment (including harvesting and silvicultural activities), as well as ending period inventory restrictions, the number of planning periods and the discount rate. The spatially explicit stand-based inventory is aggregated into "macro-stands", thus losing geographic representation in order to reduce the number of decision variables to a manageable size. The outputs from SAWS are the number of acres which were assigned to each treatment, and the volumes of merchantable hardwood and softwood which are thus extracted in each planning period.
Thus, SAWS is a strategic planning tool, intended for use on large areas without being spatially explicit.

2.4. Hierarchical Planning

Forest management has all the classical elements of a hierarchical problem -- differing levels of decision making, elements of uncertainty in forecasts, and a rolling planning horizon [Hax and Golovin 1978, Gunn and Rai 1987]. It has become established practice to adopt a hierarchical planning framework for forest management, and to structure decision support systems or mathematical models to conform to the requirements of each phase [Vertinsky et al 1994, Gunn and Rai 1987, Barros and Weintraub 1982, Weintraub and Cholaky 1991]. The hierarchical framework is appropriate to integrate groups of decision makers, with different and often conflicting objectives, at different hierarchical levels into decision making processes [Weintraub and Davis 1996].
2.4.1. Forest Management Hierarchy

An example of a forest management hierarchy is illustrated in Figure 2.4.

Figure 2.4. Hierarchy of Forest Planning

Hierarchical decision support systems for forest management must resolve multiple and often conflicting objectives of decision makers. Different levels of spatial detail are required at strategic, tactical and operational planning stages, and appropriately designed feedforward and feedback mechanisms must link models at each level. Each planning level demands appropriately designed mathematical models to support the decision-
making process. These models differ in the planning horizon used, the level of decision-maker to be "satisfied", and the refinement of spatial detail in the inputs and output of the model.

Strategic planning addresses the allocation of forest resources amongst multiple and competing demands [Weintraub and Cholaky 1991]. At this stage, the planning horizon is long, encompassing at least one crop rotation. Examples of strategic goals are the creation or preservation of habitat and assurance of a perpetual supply of timber products. Large forests may be divided into zones, stands are aggregated into "macro-stands" or strata and recommendations for interventions are developed. These recommendations include an estimate of the amount of timber product that may be extracted (annual allowable cut) and levels (acreage, volume) of silvicultural activities that are required to sustain the annual allowable cut. The strategic planning horizon usually encompasses more than one crop rotation. The aspatial outputs of the strategic planning phase are input goals or constraints to the tactical planning phase.

The overall goal of the tactical planning process is to produce a spatially and temporally viable schedule of activities. In this phase a shorter horizon, usually less than or equal to one rotation, and shorter time periods are used. For example, in New Brunswick, the tactical planning horizon is thirty-five years, with five year planning periods [NBGOVT 1994]. One critical feature of tactical planning is the requirement to produce spatially explicit recommendations. Decisions become discrete (whether or not to cut a given stand) rather than continuous (the percentage of a given stand-type to be assigned to a particular activity). This introduces non-linearities into the models, and increases the difficulty of solving them to optimality [Weintraub and Davis 1996]. The output of tactical planning is a mapped schedule of interventions (harvesting, silvicultural activities) on stands or blocks of stands, and schedules of other activities
such as road building, setting aside special areas for special habitats (deer wintering, shelterwood, water and stream buffers, wildlife corridors etc.).

In the operational planning phase, a tactical plan for one period is implemented in the best possible way. This phase of "on-the-ground" planning requires fine levels of detail in constructing roads, scheduling personnel and equipment. Operational planning horizons are typically very short -- for example, a one year horizon with bi-weekly planning periods may be suitable for planning the detail of "on-the-ground" operations.

2.4.2. Hierarchy of Planning Models

In forest management, the difficulties in solving models which span the strategic and tactical planning phases has, to some extent, motivated the pursuit of hierarchical planning [Weintraub and Cholaky 1991]. The structure of forest harvest/treatment models in the literature may be broadly categorized into two types, strata-based models and area-based or spatial models, depending on whether the stands are aggregated aspatially by type or not. This delineation coincides with the strategic and tactical phases of the hierarchical planning process.

Strata-based models are amenable to Linear Programming methods, since decision variables are continuous. Area-based or spatial models are not, since decisions to chose from management options or road building projects require binary or integer decision variables. There is considerable debate over the value of Mixed Integer formulations due to the considerable computational burden which occurs when problems of realistic size are dealt with. Some methods of improving formulations by adding inequalities which tighten the LP relaxation of the integer program have been applied to this problem [Barahona, Weintraub and Epstein 1992]. In many cases, however, researchers have turned to heuristic and metaheuristic methods to solve area-based models.
2.4.2.1. **Strata-Based Models**

Strata-based models are used to develop long term plans for forest industries and governments. They are used to determine the potential harvest of forest products from large areas over long time frames, perhaps encompassing several rotations. These are strategic planning models, in which the goal is usually to determine the level of periodic product flows which are sustainable. Constraints on the levels of harvest in these models are designed to ensure the sustainability of the resource. These constraints may include the non-declining yield requirement and an ending reserve margin. In addition to sustainability issues, environmental considerations such as the maintenance of a diverse age composition of the forest, or the creation of a given age-class structure in the forest, are appropriate in these strategic planning models. The strata model will develop a capacity plan subject to corporate strategy, but may not provide spatially feasible solutions [Nelson, Brodie and Sessions 1991].

![Diagram](image)

**Figure 2.5 Strata-Based Model**

Linear Programming is especially suitable for formulating and solving strata-based plans when stochastic effects are ignored. Decision variables such as the volume or area to harvest (or treat) from a stand type are continuous and constraints are easily specified. Timber RAM [Navon 1971] has been used extensively in the 1970s in the United States for strata-based planning. FORPLAN (FORest PLANning), developed for the US Forest Service, is a well-known LP model for forest wide harvest scheduling and has served as the primary analysis system for the USDA Forest Service Land Management Planning.
LP strata-based models may be further classified as Model I [Navon 1971] and Model II [Johnson and Scheurman 1977] formulations. In the Model I formulation, candidate schedules of potential treatments over the planning horizon are enumerated for each stand type. Each sequence differs from the others in the number of harvests over the time horizon and/or the age at time of harvest, and will produce a different pattern of outputs (wood products) over time [Johnson and Scheurman 1977, Gunn and Rai 1987]. The decision variables $x_{tq}$ are the amount (area) of each stand type $l$ to be assigned to each possible schedule of treatments $q$. This formulation removes time variables from the model equations, at the expense of the specification and computation of the coefficients for a large number of treatment schedules. In the Model II formulation, treatment schedules over time are not pre-specified for each stand type. Rather, the decision variables are linked to the periods in the planning horizon. TimberRAM [Navon 1971] is a well-known modeling system that is based on Model I formulation. FORPLAN is a Model I formulation, and both FORPLAN and Musyc systems can accommodate Model II-type decision variables [Field 1984].

In the basic FORPLAN model, land is allocated to general management objectives and treatments and product flows are scheduled. The following description of the FORPLAN model was taken from Field [1984]. In this model, the decision variables $x_{tq}$ are the number of acres of land type $l$ to be assigned to management strategy $q$. Given $T$ land types, $S_l$ possible management strategies for land type $l$, $N$ planning periods and $M$ possible inputs outputs and other responses. $A_l$ is the number of acres in land type $l$, $D_{tq}$ is the discounted net value per acre of land type $l$ under management strategy $q$, $P_{tgik}$ the response level $k$ (per acre) in time period $j$, land type $l$ and management strategy $q$. $V_{tqj}$ the timber harvest volume. $B_{jk}$ is the desired level of input, output or response in period $j$. The LP model is then
Max \( \sum_{l=1}^{T} \sum_{q=1}^{S_l} D_{lq}x_{lq} \)

subject to

\[ \sum_{q=1}^{S_l} x_{lq} = A_l, \quad l = 1, \ldots, T \]

Area constraints

\[ \sum_{l=1}^{T} \sum_{q=1}^{S_l} V_{lqj}x_{lq} - \sum_{l=1}^{T} \sum_{q=1}^{S_l} V_{lqj-1}x_{lq} \leq 0, \quad j = 1, \ldots, N - 1 \]

Nondeclining timber harvest

\[ \sum_{l=1}^{T} \sum_{q=1}^{S_l} P_{lqk}x_{lq} \leq \gamma_k \geq B_{jk}, \quad j = 1, \ldots, N, k = 1, \ldots, M \]

Input, output and response

\[ x_{lq} \geq 0, \quad \forall l, q \]

Non - negativity

Figure 2.6 FORPLAN Model

Recently, the Ecosystem Management Analysis center (Fort Collins, Co.) has developed a more comprehensive modeling system (SPECTRUM) which, building on FORPLAN capability, provides a Graphic User Interface to define decision variables, constraints and objectives. In SPECTRUM, goal programming models allow the user to address conflicting multiple use objectives. SPECTRUM is foremost a linear programming system, although some integer variables are used in defining cost functions [Schuster, Leefers and Thompson 1993].

Figure 2.7 shows the Model II formulation, as taken from [Gunn and Rai 1987]. In this formulation, in contrast to Model I, decision variables \( x_{ij} \) and \( w_{jN} \) represent both the timing and magnitude of harvest choices for each stand type at each period. Gunn and Rai [1987] observed that Model II can be formulated as an acyclic network, where the nodes correspond to initial regeneration periods and planning horizon periods \( i = -M, \ldots, 0 \) and \( i = 1, \ldots, N \) and arcs represent the flow of products from a regeneration period to a harvest period. At each period in the horizon, stand types (i.e. age-classes) are aggregated and the initial delineation of the forest into stand types (by age-class) is lost.
\[
\begin{align*}
\text{Max} & \sum_i \sum_j D_{ij} x_{ij} + \sum_i E_{iN} w_{iN} \\
\text{Subject to} & \quad \sum_j x_{ij} + w_{iN} = A_i, i = -M, 0 \\
& \quad \sum_i x_{jk} + w_{jN} = \sum_j x_{ij}, j = 1, N \\
\text{where} & \quad x_{ij} \quad \text{ha regenerated in period } i \text{ and regeneration harvested in period } j \\
& \quad w_{iN} \quad \text{ha regenerated in period } i \text{ and left as ending inventory in period } N \\
& \quad A_i \quad \text{ha in period } 1 \text{ that were regenerated in period } i \ (i = -M, 0) \\
& \quad M \quad \text{age of the oldest age - class present in period } 1 \\
& \quad D_{ij}, E_{ij} \quad \text{Discounted net revenue coefficients}
\end{align*}
\]

Figure 2.7. Model II Formulation

In summary, strata-based optimization models typically have an objective function which is to maximize the net present value of product flows over time. The models are aspatial, since stands are aggregated into strata or macro-stands. These models are relatively easy to formulate as linear programs, and although there may be many variables, are computationally tractable. The aspatial model output is usually a schedule of harvest levels (in area or volume) and possibly a schedule of silvicultural treatments over the planning horizon. These outputs are the inputs to the tactical planning problem.

2.4.2.2. Spatially Explicit Models for Tactical Planning

Area-based models address the tactical issues in medium range planning horizons, (usually less than one crop rotation), while attempting to achieve harvest levels as specified in the long range plan. Transportation requirements (road networks) and spatial concerns such as access over time, adjacency over time, habitat dispersion and fragmentation, requirements for habitat and riparian corridors are normally addressed in area based models [Brodie and Sessions 1991]. Area-based models provide input to the short term operational planning systems and may be used to provide feedback to the
strata-based strategic model. The output from these models is a mapped schedule of forest treatments, and possibly road-building, for each period in the tactical planning horizon (Figure 2.8).

![Tactical Planning Framework](image)

**Figure 2.8 Tactical Planning Framework**

Decisions regarding the choice of resource projects include both the type of project (silvicultural activity, harvest) and the timing. Transportation issues such as construction, maintenance, upgrading or removing roads must be dealt with in the tactical model, since a large proportion of management costs is spent on road construction or upgrading, especially in natural forests. One strategy of modeling access to forested areas is to pre-select a limited number of routes to be considered. This strategy not only requires preliminary work but results in a model in which there is no guarantee of optimality since the best route may not have been included. In order to determine the best (i.e. cheapest) routes for accessing stands and transporting products to market, road construction decisions must be represented explicitly in the optimization model along with timber resource management options [Weintraub and Navon 1986]. In a mathematical program, this results in the inclusion of integer decision variables for road projects, and can significantly increase solution difficulties.

The Integrated Resource Planning Model (IRPM) is a well known mixed integer model that simultaneously considers land allocation and scheduling with a transportation
network model. Land is divided into areas (polygons) on which fractions of resource projects are allocated. The model was developed by Kirby and others [Kirby et al 1980, Kirby, Hager and Wong 1986], and combines resource project selection with a transportation network subject to road capacity constraints. Many others have used variants of the IRPM model.

Protection of habitat, preservation of or convergence to an age-class structure, insuring of water quality -- these are some of the many expressions of environmental requirements in forest management. The most commonly specified environmental constraints are the maximum opening size and adjacency delay requirements. The maximum opening size constraint limits the area of any clearcut. In New Brunswick at this time this limit is 100 ha. [NBGOVT 1994], in Nova Scotia 50 ha. [Province of Nova Scotia 1990]. Adjacency delay constraints specify that area bordering clearcuts may not be scheduled for harvest for a fixed length of time, say 10 or 20 years. The effect of adjacency delay is to allow a "green-up" period for the stand which has been clearcut. In the case of an even-aged forest, this would prevent the execution of a progressive clearcut, which would cause a forced migration of the large animals out of their habitat with no likely means to return. Adjacency delay means (roughly) that a harvest pattern such as Figure 2.9(a) is appropriate, but Figure 2.9(b) is not. Another regulation which has been imposed to avoid this effect is the requirement for wildlife corridors -- strips of forest of a prescribed minimum width which allow animals to travel "through" cut over areas. These goals may be found in LP, IP, MIP and non-linear mathematical models as constraints, or as terms in objective functions of goal programming and heuristic search formulations.
Special habitat protection is often a requirement. For example, in New Brunswick, designated land areas are set aside from harvesting to provide deer wintering yards. Other pre-specified "set-asides" may occur when a particular species is deemed to be in danger of extinction due to loss of habitat, or when an area is to be reserved for a recreational use such as a park. These restrictions remove the affected stands from consideration for clearcut harvesting, although other methods such as shelterwood cutting may be allowed [Province of Nova Scotia 1990, NBGOVT 1994].

In an effort to directly include the objectives of diversity and habitat availability, Hof and Joyce [1992] present a non-linear model to optimize two types of wildlife habitat, edge-type and dense-type, combined with forest harvest optimization. This is further developed by Hof and Joyce [1993] to model edge effects, fragmentation effects and habitat area threshold with integer variables. In Hof and Rafael [1993], approaches for finding optimum allocation of forest age-classes to meet multi-species conservation objectives are investigated.

Davis and Barrett (1992) have developed a model to examine the long-term effects of forest practice on habitats. The model predicts the effect on habitat of even-aged management by large clear-cuts, small group selection planting, selection cutting, hardwood type conversions to conifer management, shelterwood and reserves. The
California Wildlife Habitat Relationship (WHR) system is used to measure the suitability of habitat type for different species. This "measurement" includes expert scoring of the each suitability of each WHR cover type for reproduction, food and shelter for each vertebrate species (Davis and Barrett 1992). The first phase (a linear program) produces a prediction of economic outputs (revenues, costs, net present worth) based on an aspatial allocation of different forest management practices (an assignment of areas of each WHR type to different management prescriptions). This aggregate plan is then allocated spatially, manually allocating ("by expert judgment") areas using the GIS. The model then "grows" the GIS polygons into the future, and evaluates each future state of the forest in terms wildlife habitat suitability for three vertebrate species.

2.5. Cohesion Between Strategic And Tactical Models.

Some of the technical and philosophical issues with regard to these mathematical models are the effects of aggregation and disaggregation on treatment schedule feasibility, the choice of appropriate measures to feed sensitivity information upwards in the hierarchy, and the choice of spatial decision units. For example, the output of the strategic phase, the aspatial treatment schedule, may not be attainable when constrained by spatial restrictions such as adjacency delay.

Whether or not a "gap" exists between the level of timber harvests produced by the strategic and tactical phase has been discussed in Daust and Nelson [1993], Nelson and Finn [1990], Nelson and Errico [1993], Yoshimoto 1994, Yoshimoto and Brodie [1994]. If there is indeed a gap, it is important to identify the tactical level factors which cause the discrepancy, and to feed this sensitivity information back to the strategic planning process.
At tactical planning, the existence or the perception of the existence of significant setup and capitalization costs has encouraged planners to group stands into harvest blocks of a minimum size [Clements, Dallain and Jamnick 1990, Jamnick and Walters 1993, 1991, Hokans 1984]. This has the effect of reducing the number of decision variables and thus making models more computationally tractable. It has the negative effect of limiting the set of scheduling choices and thus potentially excluding better schedules. (This issue is discussed in more detail in Chapter 3.) This is a relevant factor in considering the so-called gap between the aspatial and spatial forecasts. The significant effects observed by Nelson and Finn [1990], Daust and Nelson [1993], Yoshimoto [1994] and Yoshimoto and Brodie [1994] are estimated using blocked stands, and these may in some measure be due to the blocking as well as to real impacts of spatial constraints. Observe for example, in Lockwood and Moore [1992], that the spatially constrained tactical schedule produced harvest volumes that had less than one percent deviation from the strata-based (aspatial) solution.

An additional complication to the process of determining the cost of spatial constraints is the fact that the magnitude of the effect of adjacency delay and maximum opening size is definitely location dependent -- there will obviously be a larger impact in forests that contain large homogeneous tracts of land. Thus, these case studies should be carefully examined in this respect.
CHAPTER 3

TACTICAL PLANNING MODELS AND SOLUTION METHODS

The framework for forest management tactical planning models was outlined in chapter 2. The desired output of these models is a good or optimal schedule of harvesting and road building. These problems are characterized by the requirement for discrete decision variables, and a complicated constraint structure due to spatial and temporal constraints on harvesting. Hence, the models are difficult to solve exactly using integer programming methods, and, although some efforts in that direction have been made, there has been a considerable direction of energy towards heuristic and metaheuristic methods of solution.

Tactical planning models are constrained spatially and temporally to meet environmental regulations which limit the size of clearcuts and prescribe adjacency conditions on areas near to clearcuts. Road networks are based on a pre-determined set of potential road links which give full access to the forested area. In this thesis, several modeling issues will be addressed. These include the design of an appropriate objective function, the selection of spatial decision units, and the inclusion of a more general road network. The model which is proposed will be solved using a metaheuristic method; however some attention to the solution methods employed by other researchers on other models is appropriate to set the context for the new model.

The first section of this chapter contains a summary of some of the methods which have been used to solve tactical planning models. Section 3.2 deals with the choice of spatial decision units, and section 3.3 the issues in modeling adjacency constraints. An
improved formulation for adjacency constraints is presented, and in section 3.4 results from implementing the improved formulation on a small study area in BC are given.

3.1. Solving Tactical Planning Models

Integer and mixed integer programming (MIP), Monte Carlo Integer Programming (MCIP), dynamic programming, heuristic and metaheuristic methods have been used to solve tactical forest management problems. Adjacency and road building requirements, requiring discrete decision variables, create the main difficulty in solving these polygon-based models using linear programming with branch-and-bound techniques. For example, Murray and Church [1995] solved a 431-block problem (excluding roading) on a 486/50 personal computer, but the solution time exceeded 24 hours. [Also, see Nelson and Brodie 1990.] Although some efforts have been made to solve such models exactly [see Weintraub et al 1994, Barahona, Weintraub and Epstein 1992, Guignard, Zhu and Chajakis 1995], it is not surprising that heuristic and simulation methods have been the tools of choice for large problems. The disadvantage of using heuristics, of course, is that optimality of the solution is not known or expected and only probabilistic measures of deviation from true optimum can be presented. However, given the lack of exactness in forest growth simulation, and the hierarchical nature of the management setting, the ability to provide several good solutions in reasonable time is very useful [Gunn and Rai 1987].

Integer and mixed integer programming models simultaneously consider roading decisions and land use decisions. These models are generally variations on the IRPM model [Kirby et al 1980], where binary decision variables are used to model land use decisions and road construction decisions. Some improvements on the formulation of the traffic capacity constraints in IRPM have been reported in Guignard et al [1995].
Monte Carlo Integer Programming (MCIP) has been used to solve area-based models including adjacency constraints [Nelson and Brodie 1990, Nelson, Brodie and Sessions 1991]. In this method, acceptable schedules are produced by generating harvest patterns over blocks randomly, then discarding those that do not meet adjacency requirements. MCIP has the advantage of simplicity, and the ability to generate a large number of feasible schedules easily for consideration by planners. Its disadvantage is that it is inefficient compared to other random heuristic search methods such as interchange, simulated annealing and Tabu Search [Murray and Church 1995].

Specialized heuristic methods have been developed for this problem. Weintraub, Jones and others [1994] describe a heuristic system for solving the IRPM model of Kirby [Kirby, et al 1980, Kirby et al 1986]. Weintraub et al [1991] used simulation techniques combined with heuristic scheduling rules to solve a truck scheduling and short term timber cutting problem. The PASS system [Tanke 1985] is a tool for analyzing alternative harvest schedules under alternative road networks. This is essentially a data management system where harvest schedules are exogenously supplied.

Lockwood and Moore [1992] used Simulated Annealing (SA) to solve a harvest scheduling problem, based on a stand-based model. They produced spatially explicit harvest schedules on test data of 6148 and 27,548 stands, which conformed to maximum and minimum opening restrictions, 20 year adjacency delays, and met even-flow harvest targets over minimum area. This is the only stand-based model in the literature which addresses all of the above constraints. The authors used a penalty function method to impose constraints and to drive the solution towards the desired harvest characteristics. Harvest blocks were formed at each stage of the procedure using a depth first graph search method. The model did not include any roading considerations. The main difficulty in the solution procedure appears to be in the specification of the penalty
functions. The shape of the functions was easily determined from the nature of the constraint that they represented. However, experimentation with several runs was necessary to adjust coefficients so that no one function dominated over others early in the SA procedure, yet the "hard" adjacency and opening size constraints were satisfied in the final solution.

Murray and Church [1994] developed a Tabu Search (TS) method to solve the operational planning formulation based on that of Nelson and Brodie [1990]. Their TS method achieved near optimal solutions in relatively short computing times. The model included road building and adjacency constraints, with stands pre-blocked. This study was relatively small, consisting of 45 blocks and 52 road links. With an improved IP formulation, this problem can be solved to optimality in reasonable time. This is discussed fully in section 3.4.

### 3.2. Spatial Decision Units

A basic concern in tactical model formulation is the choice of spatial decision units. From a spatial and biological perspective, the stand is the logical decision unit. This is because the stand represents a contiguous area which is homogeneous in terms of its current state and future growth potential. Stands which are too large may be subdivided without losing this homogeneity. Results of decisions based on stands may be mapped using a GIS, and thus present a "picture" of the model results which is meaningful to the decision maker. On the other hand, stands tend to vary widely in size, and there may be many small stands. This makes problems awkward to formulate, and in the case of integer programming, exponentially harder to solve.

Aside from modeling issues, economic reasons may influence the decision to create aggregated blocks of stands for decision units. A minimum opening size (harvest
block) has been imposed in many models [Wightman and Baskent 1994, Baskent and Jordan 1991, Lockwood and Moore 1992]. In cases where large fixed costs of harvest are of concern, (for example, where hillside terrain is being harvested as is common in British Columbia), a large production volume is required to make the operation economically viable. High capital costs may dictate maximizing utilization of harvesting equipment. This may require a minimum block size to reduce time spent in moving and setting up equipment. Other considerations, apart from polygon area, influence the design of blocks. In some situations, an economically suitable block has a minimum value of merchantable timber product. Block shape can have an impact on harvesting costs — long, thin wedge shaped areas are often more expensive to work. The ratio of perimeter to area can be used to evaluate the quality of a block shape.

Most of the models proposed in the literature [Barahona, Weintraub and Epstein 1992, Barros and Weintraub 1982, Jones et al 1991, Nelson et al 1991, Weintraub and Cholaky 1991, Yoshimoto et al 1994] assume pre-defined harvest blocks as spatial decision units. Others [Baskent and Jordan 1991, Jamnick and Walters 1993, Jordan and Baskent 1992] form blocks or neighborhoods by random seed search methods combined with block formation rules. BLOCK [Clements, Dallain and Jamnick 1990] and CRYSTAL [Jamnick and Walters 1993] are examples of such systems for forming harvest blocks. Hokans [1984] proposes the use of artificial intelligence to select stands for harvest by constructing a discriminant function that simulates the spatial criteria used by experienced foresters. The computer support system speeds the process by calculating important measures such as distance to roads and recency of harvest of adjacent areas. Their proposed "expert" system would "learn" the rules that experts use to select cutting units.
There are some distinct disadvantages to using blocks as decision units. First, there is some effort involved in creating these blocks. Pre-blocking implies a reduction in the number of stand-harvest timing choices that are available, and may eliminate many good solutions from consideration. Aggregating stands of different age class or growth potential means that some stands will be scheduled at other than their peak production period. Thus, blocking may be expected to lead to less than optimal solutions since the feasible solution space of the problem has been restricted.

Solving stand-based models is not always too difficult. Lockwood and Moore [1992] successfully solved a problem of significant size using stands as input units without pre-blocking, using simulated annealing (SA). Blocks were dynamically created as the procedure progressed, using a cost function to penalize clusters of stands (that are assigned harvest in the same period) which are either too small or too large. Thus, within the limitations of the simulated annealing procedure, their model addresses all possible configurations of stands into harvest blocks. This thesis presents a similar "dynamic" blocking algorithm using graph structures, and a solution method which produces good solutions in reasonable computation time.

3.3. Adjacency Constraints

This section deals with effective formulation of adjacency constraints for tactical planning models based on pre-aggregated polygons or blocks. Let \( x_s = I(0) \) be the decision variable representing harvest (not harvest) polygon \( s \). The most obvious formulation for adjacency constraints is the dis-aggregated form: \( x_i + x_j \leq 1 \) for each pair of adjacent polygons \( i \) and \( j \). The number of such constraints can be large (one for each pair of adjacent polygons and for each planning period) however, and this has led several researchers to investigate methods of aggregating these constraints.
Various attempts at aggregating these pair-wise adjacencies into a smaller set of constraints have been made. These are generally based on recognizing that different patterns of adjacent polygons can be identified, and that constraints can be derived which cover several adjacent polygon pairs. [See Jones, Meneghin and Kirby 1991, Murray and Church 1996a and 1996b, Torres-Rojo and Brodie 1990] The difficulty with these aggregations is that although a smaller constraint set may be generated, they may not be the best formulations for models which are to be solved using integer programming methods. Murray and Church [1996] noted this problem, and recognize the importance of identifying cliques (a set of mutually adjacent polygons). The Type I constraints of Jones et al [1991] in fact identify such cliques of cardinality 2, 3 and 4, although they did not recognize these as clique constraints in a general graph theoretic framework. Moreover, they stated that cliques of order 5 or more are a physical impossibility. This is not true when adjacency is defined by polygons being a minimum distance apart as opposed to the stricter condition of sharing a common edge. They then went on to aggregate these "Type I" constraints into further aggregated Type II constraints. Torres-Rojo and Brodie [1990] provide heuristic rules to create one adjacency constraint for each polygon (the T-B method) and produce a different set of constraints than do Jones, Meneghin and Kirby [1991]. In Yoshimoto and Brodie [1994], a third method which utilizes matrix algebra produces yet another set of adjacency constraints.

Better formulations, which involve less constraints and which sharpen the bound achieved by the Linear Programming relaxation of integer programs, are possible. Moreover, in the sparse adjacency graphs for these forest planning problems, it is relatively easy to generate the set of clique constraints which are known to represent a tight LP formulation [Padberg 1973, Nemhauser and Wolsey 1988, Nemhauser and Trotter 1974]. To explore this concept, some integer programming results are required.
The next sub-section summarizes the fundamental theory of strong valid inequalities and cutting planes methods for solving integer programs. For a comprehensive description, see Nemhauser and Wolsey [1988].

### 3.3.1. Strong Valid Inequalities and Cutting Planes Algorithms

Given the discrete (binary) integer programming problem

\[
\text{IP: max}\{cx: Ax \leq b, x \in B^*\}, \text{ with feasible region } S = B^* \cap P, \text{ where }
\]

\[
P = \{x \in R^n: Ax \leq b, 0 \leq x_i \leq 1\}. \text{ The linear programming relaxation of IP, ILP, is }
\]

\[
\text{max}\{cx: x \in P\}. \text{ This is the problem obtained by relaxing the integrality constraints. In }
\]

other words, the binary variables \(x_i\) are replaced by continuous bounded real variables

\(0 \leq x_i \leq 1\). The feasible region of ILP, \(P\), contains \(S\), and the optimal solution to ILP is

greater than or equal to that of IP. A valid inequality for ILP, \(\alpha x \leq \pi\), is one which

satisfied at all points in \(S\). In theory, the integer program IP can be reduced to a linear

program wherein a system of valid inequalities completely defines \(\text{conv}(S)\), the convex

hull of \(S\). Thus, the integer programming problem can be replaced by a linear

programming problem. Although in theory this can always be done, the problem of

finding such an inequality system for \(\text{conv}(S)\) is hard.

For an integer problem IP and its relaxation ILP, let \(z_{ip}\) be the optimal solution to

IP and \(z_{ilp}\) the optimal solution to ILP. The cutting planes method is to find valid

inequalities which reduce the gap between \(z_{ip}\) and \(z_{ilp}\), sharpening the bound \(Z_{ilp}\), and,

consequently, forcing variables to attain integral values in ILP. Strong valid inequalities

are those which define facets or high order faces of the polyhedron \(\text{conv}(S)\). The idea is

to start with the inequality system \(\{Ax \leq b, 0 \leq x_i \leq 1\}\) and, if this does not define \(\text{conv}(S)\),

then to progressively construct stronger valid inequalities to create a more complete

description of \(\text{conv}(S)\). If it is practical to complete this process, a linear program which
is equivalent to the integer program is defined, meaning that the optimal solution to the linear program is also the optimal solution to the integer program. Alternately, one can generate enough inequalities so that the "gap" \( z_{ilp} - z_{lp} \) is small, and then use branch and bound from that point forward to find the optimal integer solution.

It is not always necessary to generate the complete inequality system which describes \( \text{conv}(S) \) to solve \( IP \). Also, from a problem-solving practical point of view, at some point the effort involved in locating further inequalities may outweigh the effort in completing the solution using branch and bound. Thus, the cutting planes algorithms involve an iterative procedure as follows:

1. Solve \( ILP \) with minimal formulation. If the solution is integer, the optimum solution to \( \text{conv}(S) \) has been found. Otherwise,

2. Find a set of (strong) valid inequality which is violated by the current fractional solution. (This is called the separation problem). Add these inequalities to the current problem and go to 1.

Many hard integer problems have been successfully approached with this method. Brethauer and Cabot [1994], Clark and Armentano [1995], Corberan and Sanchis [1994], Dijkstrausen and Faigle [1993], Magnanti, Mirchandani and Vachani [1995], Pochet and Wolsey [1991 and 1986] and Weintraub and Vera [1991] are only a few examples. Nemhauser and Sigismondi [1992] have developed a cutting planes algorithm for the node packing problem which is of direct relevance to this work. See also Barahona, Weintraub and Epstein [1992], and Weintraub, Barahona and Epstein [1994], wherein these methods are used to solve the adjacency problem as part of a column generation algorithm for solving forest planning problems.

### 3.3.2. The Adjacency Problem: Node Packing

For simplicity of notation, omit the time dimension. Let \( A(i) \) be the list of
polygons which are adjacent to polygon $i$, and $x_i = 1$ if stand $i$ is scheduled for harvest.

**Definition 1:** The adjacency graph of the forest $AG(V,E)$ is a simple graph (a graph with no loops) with one node $v$ for each polygon and one edge $e=(v,w)$ for each pair of adjacent polygons $v$ and $w$. See figure 3.1.

**Definition 2:** A node packing (stable set, independent set, vertex packing) in a graph $G=(V,E)$ is a subset $S$ of $V$ such that the subgraph induced by $S$ has no edges. (The subgraph induced by $S$ is the graph with node set $S$ and the edges of $E$ which have both their end nodes in $S$.) In a weighted graph, the maximal independent set is one of maximum weight.

![Figure 3.1. Adjacency Graph](image)

A harvest solution is feasible with respect to adjacency rules if $X = \{i : x_i = 1\} \subseteq V$ is an independent set or node packing in the graph $AG$. For example, in figure 3.1, two feasible solutions are $\{1,3,7,10\}$ and $\{2,5,6,9\}$. The problem of finding node packings of maximum weight is an NP-hard problem [Garey and Johnson 1979, Nemhauser and Wolsey 1988]. Nonetheless, recognition of the nature of the adjacency constraints as node packing conditions leads immediately to some known strong valid inequalities which define facets for the node packing problem. These inequalities significantly
strengthen the disaggregated representation of the adjacency constraints. Although a complete description of the node packing polytope is not known, the maximal clique, odd hole, odd anti-hole and web (claw) facet-defining inequalities are very strong in sharpening the LP bound for this problem [Nemhauser and Trotter 1974].

3.3.2.1. Clique Inequalities

A clique in a graph is a complete subgraph -- that is, each node is joined by an edge to every other node. A maximal clique is a clique which is not a proper subset of any other clique, and clique, when unqualified, will mean maximal clique in this discussion. Clearly, the constraint \( \sum_{i \in C} x_i \leq 1 \), where \( C \) is a clique in \( AG \), is a valid inequality for this problem. In fact, this is a facet-defining inequality for the node packing problem [Nemhauser and Trotter 1974]. In figure 3.1, there are several cliques of cardinality 3. (For example \{1,2,3\}, \{1,4,5\}. ) For some graphs, called perfect graphs, clique inequalities give a complete description of the node packing integer polytope. In general, however, other inequalities are needed. These are the (lifted) odd hole constraints, odd anti-hole constraints and web or claw constraints [Nemhauser and Trotter 1974].

3.3.2.2. Odd Hole, Odd Anti-Hole Inequalities

An odd hole in a graph is, by definition, a chordless cycle of odd length. The existence of an odd hole \( H \) gives rise to the valid inequality \( \sum_{i \in H} x_i \leq \frac{1}{2}[|H| - 1] \). This inequality can be lifted to form a stronger inequality \( \sum_{i \in H} x_i + \sum_{j \in N(H)} a_j x_j \leq \frac{1}{2}[|H| - 1] \) by considering all nodes which are adjacent to \( H \) (those in \( N(H) \)) [Nemhauser and Trotter 1974]. Both clique and odd hole inequalities were used by Barahona et al [1992] in their column generation algorithm. In Figure 3.1, \{4,6,9,8,5\} is an odd hole, and its inequality is \( x_4 + x_6 + x_9 + x_8 + x_5 \leq 2 \). This is "lifted" by including node 7 (and finding the largest possible value of \( a_7 \), to be 2) to derive the inequality \( x_4 + x_6 + x_9 + x_8 + x_3 + 2x_7 \leq 2 \).
A anti-hole is the edge complement of a hole. That is, an anti-hole of cardinality $n$ is a graph which is formed by removing the edges of an $n$-hole from $K_n$, the complete graph on $n$ nodes. Odd holes and anti-holes are illustrated in Figure 3.2. If a graph contains an odd anti-hole $G$, then the inequality $\sum_{i \in G} x_i \leq 2$ is a facet defining inequality for the node packing problem.

![Figure 3.2. Odd hole and Anti-Hole](image)

### 3.4. Generating Clique Constraints

The number of cliques in a given graph can be very large, and hence difficult to enumerate completely [Bron and Kerbosch 1973]. However, for these adjacency problems, the graphs are of low density, resulting in a small number of cliques of cardinality greater than three. The following procedure was developed to locate cliques in a graph. It is suitable for sparse graphs.

#### 3.4.1. Generation Procedure

Let $A = \left\{ a_{ij} | (i, j) \in E \rightarrow a_{ij} = 1 \right\}$ be the adjacency matrix of $G=(V,E)$, and calculate the degree of each node, $\text{deg}(i) = \sum_j a_{ij}$. and the maximum degree $M = \max(\text{deg}(i), i \in V)$. Observe that a node $i$ can be an element of a clique of order $n$ only if $\text{deg}(i) \geq n$. Then, the procedure is to build cliques of order $(n+1)$ from cliques of order $n$. Denote by $n$-clique a clique of cardinality $n$. First, the 2-cliques are identified directly from $A$ and stored as ordered pairs $\left\{ a_{ij} | a_{ij} = 1, j > i \right\}$. Then, higher order cliques
are built successively, discarding the lower cardinality dominated cliques as new ones are found. See Figure 3.3.

This algorithm is relatively efficient since the degree of the nodes is used to exclude many of the cliques from consideration. Also, each sub-clique is automatically discarded in step (3) and only the non-dominated or maximal cliques are output at the end in step (5). The computational effort in generating these maximal cliques should be no greater than that spent in aggregating constraints into other, less efficient forms.

<table>
<thead>
<tr>
<th>For $n = 3$ to $M$</th>
<th>(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each clique $k = {k_1, k_2, ..., k_n}$ of order $n$</td>
<td>(2)</td>
</tr>
<tr>
<td>If $\deg(k_1), \deg(k_2), ..., \deg(k_n) \geq (n + 1)$ then:</td>
<td>(3)</td>
</tr>
<tr>
<td>For $b = 1$ to $</td>
<td>V</td>
</tr>
<tr>
<td>If $a_{b,k_1} = a_{b,k_2} = ... = a_{b,k_n} = 1$ then</td>
<td>(5)</td>
</tr>
<tr>
<td>Mark clique $n$ dominated</td>
<td></td>
</tr>
<tr>
<td>Endif</td>
<td></td>
</tr>
<tr>
<td>If $b &gt; k_n$ then</td>
<td></td>
</tr>
<tr>
<td>Store new clique ${k_1, k_2, ..., k_n, b}$</td>
<td></td>
</tr>
<tr>
<td>Endif</td>
<td></td>
</tr>
<tr>
<td>Next $b$</td>
<td></td>
</tr>
<tr>
<td>Endif</td>
<td></td>
</tr>
<tr>
<td>Next clique</td>
<td></td>
</tr>
<tr>
<td>Next $n$</td>
<td></td>
</tr>
<tr>
<td>Output each maximal clique which is not marked dominated.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.3 Algorithm to Generate all Cliques in a Sparse Graph

3.4.2. Bloedel Farm Case Study

The MacMillan Bloedel tree farm in Vancouver, British Columbia consists of 45 forest polygons. A potential road network of 52 links has been defined for the area [Nelson and Brodie 1990]. The planning problem is taken from Murray and Church [1994], who based it on the work of Nelson and Brodie [1990]. There are 3 planning periods, and the planning problem is to schedule harvests on the given polygons which meet upper and lower bounds on harvest in each time period, and undiscounted revenue
bounds for each period. Adjacency delay of one period is imposed on adjoining polygons. The potential road network is a tree structure -- that is, it contains no cycles. Thus, the road network feasibility constraints are relatively simple to specify since each link either requires one predecessor in the tree or none at all. Each stand has a set of road links, one of which is required if that stand is to be harvested. Decision variables $x_{it}$ are to harvest polygon $i$ in period $t$ and $z_{jt}$ to build link $j$ in period $t$. Discounted and undiscounted revenues from harvesting polygon $i$ in period $t$ are $a_{it}$ and $\bar{a}_{it}$, and discounted (respectively undiscounted) costs to build each road link are $c_{rl}$ and $\bar{c}_{rl}$. $M_r$ is the set of links from which one must be built if link $r$ is built, and $S_i$ is the set of road links from which one must be built if stand $i$ is to be harvested. (Note that each $M_r$ has only one element for this road network, since it contains no cycles.)

The IP formulation of the problem used by Murray and Church [1994] is:

<table>
<thead>
<tr>
<th>Objective:</th>
<th>[ \max \sum_i \sum_t a_{it} x_{it} - \sum_i \sum_r c_{rl} z_{rl} ] Discounted revenues - discounted road costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject to:</td>
<td>[ \sum_t x_{it} \leq 1 \quad \text{for each } i ] Harvest each polygon at most once.</td>
</tr>
<tr>
<td></td>
<td>[ \sum_t z_{rl} \leq 1 \quad \text{for each } r ] Build each link at most once.</td>
</tr>
<tr>
<td></td>
<td>[ z_{rl} \leq \sum_t x_{it} \quad \text{for all } r, t, k \in M_r ] Road Connectivity</td>
</tr>
<tr>
<td></td>
<td>[ x_{it} \leq \sum_{r \in S_i} z_{rl} \quad \text{for all } i, t ] Link required to harvest stand $i$</td>
</tr>
<tr>
<td></td>
<td>[ \sum_t v_{it} x_{it} \geq L_t \quad \text{for all } t ] Lower bound on volume</td>
</tr>
<tr>
<td></td>
<td>[ \sum_t v_{it} x_{it} \leq U_t \quad \text{for all } t ] Upper bound on volume</td>
</tr>
<tr>
<td></td>
<td>[ \sum_t \bar{a}<em>{it} x</em>{it} - \sum_r \bar{c}<em>{rl} z</em>{rl} \geq LR_t \quad \text{for all } t ] Bound on undiscounted profit</td>
</tr>
<tr>
<td></td>
<td>[ n_i x_{it} + \sum_{r \in N_i} x_{it} \leq n_i \quad \text{for all } i, t ] Adjacency Requirements</td>
</tr>
<tr>
<td></td>
<td>[ x_{it}, z_{rl} \in {0,1} \quad \text{all } i, t, r ] Binary Variables</td>
</tr>
</tbody>
</table>

The optimal solution to this problem is $5,953.20$ [Murray and Church 1995]. Note
that the adjacency requirements are expressed in aggregated form of Torres and Brodie [1990], where \( N_i \) is the set of polygons adjacent to \( i \). This problem is suitable for pre-generating clique constraints since it is a relatively small study area. To illustrate the effectiveness of generating maximal clique constraints, three formulations are solved. The first (DISAGG) is with the disaggregated form of clique constraints

\[
x_i + x_j \leq 1, \text{for each pair of adjacent polygons } i, j \text{ and } t = 1,2,3.
\]

The second (MURRAY) is the above formulation with aggregated adjacency constraints. The third (MAXCLIQ) is the above problem with the aggregated adjacency constraints replaced by maximal clique constraints. Table 3.1 summarizes results from implementing these three formulations, using LINDO on a 486/33 personal computer.

The maximum cardinality of the cliques was 4, and there were no maximal 2-cliques, 20 3-cliques and 17 4-cliques. Thus, with three time periods, this resulted in 111 clique constraints in MAXCLIQ. There are 112 adjacent pairs which resulted in 336 constraints in DISSAG. MURRAY used 135 constraints to specify adjacency. Thus, MAXCLIQ is efficient in reducing the number of clique constraints. The constraints were generated using a QBASIC program, which took less than 2 seconds (elapsed) time to compute the maximal cliques.

Running the program in LINDO with the disaggregated cliques was unsuccessful. The program ran for more than 24 hours and terminated when the pivot limit of 1210599 was exceeded. This formulation was included to underline the importance of generating tight constraints for the problem.
<table>
<thead>
<tr>
<th></th>
<th>LP OPTIMUM</th>
<th>Number of Branches</th>
<th>Time to Solution (HR:MIN:SEC)</th>
<th>Number of Adjacency Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISSAG</td>
<td>6767.596</td>
<td>&gt;4842</td>
<td>&gt;24 hrs</td>
<td>336</td>
</tr>
<tr>
<td>MURRAY</td>
<td>6394.120</td>
<td>1227</td>
<td>1:16:37.53</td>
<td>135</td>
</tr>
<tr>
<td>MAXCLIQ</td>
<td>6076.102</td>
<td>533</td>
<td>0:17:42.04</td>
<td>111</td>
</tr>
</tbody>
</table>

Although it is dangerous to infer a lot from one case study, this does show that it is easy to generate clique constraints for small problems, and that the clique representation is relatively compact. Since it has been proven that the maximal clique constraints form facets of the node packing problem, they are a logical choice to represent the adjacency requirements in these problems.

For large problems, it may be impractical to generate all the maximal cliques in the adjacency graph. In this case, a cutting planes algorithm which finds violated clique and other (odd hole, odd anti-hole, web) violated strong inequalities [Nemhauser and Sigismondi 1992] for this problem could be used.
CHAPTER 4

TABU SEARCH

A Tabu Search strategy was chosen to solve the tactical forest planning problem because of its record of success on hard optimization problems. Since the algorithms presented in chapter 7 of this thesis are based on this metaheuristic, this chapter is devoted to describing the principle ideas behind the tabu search method, the basic algorithm and some of its more sophisticated variants. Amongst the more advanced implementations of tabu search, those algorithms which use dynamic parameter assignments and the self-tuning, feedback-based Reactive Tabu Search method (R-TABU) [Battiti and Tecchiolli 1994] are especially important in the context of this thesis.

Tabu search (TS) belongs to the metaheuristic class of algorithms, where local optimization (local neighbourhood search) is enhanced with strategies which enable the search process to go beyond local minima. Simulated annealing, genetic algorithms and evolutionary algorithms are also included in this category. (See Reeves [1996] for a comprehensive description of these and other metaheuristic methods.) The seminal ideas behind tabu search were first developed by Glover [1986], and independently by [Hansen 1986]. Tabu search is a history-based or memory-based strategy; information from previous phases of the search is used to direct future phases.

4.1. Basic Tabu Search Algorithm

Consider an optimization problem $P$, which is to find a solution $x$ which minimizes
some function $z(x): \ P: \ \min z(x)|x \in F \subseteq X$. If $x$ is a solution to $P$, and if a set of moves $M$ which perturb $x$ to form new solutions in $X$ can be defined, then the neighbourhood of $x$ is the set of these perturbed solutions. That is, denoting the set of all allowed moves by $M$, $N(x) = \{x' \in X| x' = x \oplus m, \ m \in M \}$. In constrained optimization problems, moves which produce an infeasible solution $x' \notin F$ are called infeasible moves, and neighbourhoods may or may not be restricted to the feasible region $F$.

Moves are relatively simple operators on the current solution, that is, those which perturb the solution slightly. For example, in a job shop scheduling problem, a solution is a sequencing of $n$ jobs $x(j_s) = t_s, \ t_s \in [1, n]$. A move can be defined as a swap of the ordering of two jobs $k$ and $l$, $x(j_s) \leftrightarrow x(j_t)$, thus permuting the previous solution. In this case, the neighbourhood of a solution is the set of all permutations of $x$ obtained by swapping any two elements of $x$. As another example, in the Vehicle Routing Problem [Gendreau, Hertz and Laporte 1994], a commonly used move is the insertion of a city into a route. The neighbourhood of a given route is the set of all new routes formed by all possible insertions into the current route. As a third example, in solving the Maximum Clique problem using tabu search, moves are to add or drop vertices from the current clique solution [Battiti and Protasi 1995, see also Gendreau, Soriano and Salvail 1993].

Tabu search is based on neighbourhood or local search, where, starting with an initial solution, each possible move is evaluated and the best improving move chosen. Local search ceases when no improving move can be found; that is, local search stops when a local optimum is found. In tabu search, when no improving moves can be found, a best non-improving move is selected. This serves to lead the search away from the current local optimum, to where a different neighbourhood has new possibilities for improving on the current best solution.
4.1.1. Strict and Fixed TS, Tabu Tenure

Used without restriction, the inclusion of non-improving moves in local search could result in endless cycling back to the same local optima. Thus, TS incorporates a prohibition strategy to avoid returning to previously encountered solutions. The prohibition scheme can be to explicitly forbid moves which lead to previously encountered solutions, the so-called strict tabu search. An indirect method, the tabu tenure or move prohibition approach, is to forbid the reversal of recently taken moves. Strict tabu search explicitly rejects the repetition of previous solutions at all phases of the search, while move prohibition is temporal in nature, and may allow the search to progress back to a previous solution in time.

Strict tabu search requires the ability to determine whether a move would in fact give rise to repetition of a previous solution. The difficulty with testing for this condition directly is that storage and indexing to all solutions which have been encountered in the search is, computationally, both time and memory intensive. A common strategy is to store only a list of the moves which have been taken, the reverse elimination method [Dammeyer and Voss 1993, Glover 1990a and 1990b]. As each candidate move is evaluated in the neighbourhood search, the sequence of previous moves is traced in reverse order to determine whether the candidate move should be prohibited. This method avoids the large storage and indexing costs to access previous solutions, but incurs an increasing computational cost as the search progresses. Recently, data structures for storing past solutions which are efficient in terms of both space and computational effort have been investigated [Battiti 1996, Woodruff 1993]. However, most work in the literature has focused on the second prohibition strategy, the move prohibition method, since it is simple in concept and is relatively trivial in computational effort to implement.
The major issue in developing algorithms which use move prohibition to direct the search process is to define appropriate rules to restrict or forbid a small sub-set of moves, the so-called tabu moves. The length of time (number of iterations of neighbourhood search) for which a move is forbidden is called the tabu tenure or prohibition period. Tabu tenure is often referred to as the tabu list size; this expression is derived from a common implementation of tabu status as a list of forbidden moves. The most basic of the TS rules is to exogenously determine the tabu tenure. This is called fixed tabu search, since, once a move is taken, its reversal is forbidden for a fixed number of iterations. There is no universal "good" value for tabu tenure, although some empirically determined guidelines for certain problem types have been published [Glover 1989]. Thus, if a fixed tabu search algorithm is to be attempted on a given problem, an experimentation phase to determine an effective setting for tabu tenure is usually required. This empirical investigation will often involve "expert" feedback from the algorithm designer. Thus, this investigation, in addition to being costly, may be difficult to reproduce, and may be instance-specific. That is, the choice of tabu tenure which is arrived at through experimentation on one problem instance may not be a good choice for other cases of the same problem [Battiti 1995].

4.1.2. Aspiration Criteria

In addition to tabu considerations, moves are accepted or rejected with regard to aspiration criteria. An aspiration can over-ride the tabu status of a move -- when this happens, the move is said to be aspirated. While a suite of aspirations may be defined for any given problem, most TS algorithms include the aspiration by default, which is to choose the "least tabu" move when all possible moves are tabu. This ensures that the process does not terminate arbitrarily. Aspiration by objective is always included. This
aspiration over-rides tabu status for any move which yields a solution better than any encountered thus far in the search, ensuring that optimum solution, if it is encountered, is not lost.

4.1.3. Summary of the Basic TS Structure

Denote by \( z(x, H) \) the move evaluation function. (The notation \( (x, H) \) means that this evaluation function may change throughout the course of the search algorithm as a function of the recorded history of the search, \( H \). See section 4.3.) The TS algorithm may be outlined as follows:

<table>
<thead>
<tr>
<th>1. Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
</tr>
<tr>
<td>1.2</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. Neighbourhood Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
</tr>
<tr>
<td>2.2</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3. Update and save best solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
</tr>
<tr>
<td>3.2</td>
</tr>
<tr>
<td>3.3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>If stopping condition is not met, go to Step 1. Otherwise, terminate and report results.</td>
</tr>
</tbody>
</table>

Figure 4.3. Tabu Search Algorithm

The cost function \( z \) and constraints defining the feasible region can be non-linear, discontinuous and even stochastic. Thus, tabu search provides a framework which is flexible in that it can be applied to a wide variety of problems. For many problems, the method has an inherent computational efficiency since only the incremental change to

4.2. Reduced Neighborhood Search

For large problems, evaluation of every possible move in the neighbourhood may be computationally prohibitive. In this case, a suitable subset of the neighbourhood must be identified for move evaluation. Candidate list strategies [Glover et al 1974, Frenkewey 1983] are used to create the reduced neighbourhood.

The reduced neighbourhood may be chosen by drawing a random sample of moves from $M$. If the result, i.e. the quality or composition of new solutions attained by executing these moves, is unsatisfactory, the process can be repeated. Another candidate list strategy is to decompose the neighbourhood into "critical" subsets. This method requires a system of "remembering" which subsets have been used in the past, so as to systematically cover all different critical sub-regions during the search process. Another random sampling method is to draw a small random sample of moves for most iterations, yet periodically evaluating a much larger sample of moves.

4.3. Move Evaluation, Intensification, Diversification

In a simplistic algorithm, tabu moves are those which would reverse recently taken moves, while the evaluation function is the original objective function $z$. The essence of Tabu search is to add principles of intelligent problem solving to the basic search
mechanism. TS enhances the subsequent stages of the search by acting on the information gleaned from the results of the search so far. Tabu schemes are based on knowledge of the history of the search, $H$, and a set of rules or principles which act on that knowledge to determine tabu status, aspiration conditions and move evaluation functions. Thus, the move evaluation function is denoted by $z(x,H)$. Differing evaluations of moves may be done at different phases of the search, depending on whether intensification or diversification is required.

*Intensification* means to search for better solutions near a known good solution. To effect intensification in the neighbourhood of a solution, the best improving moves are chosen until a local optimum is found. Intensification is also achieved by searching about solutions with known good attributes. In this case, the evaluation function $z(x,H)$ is changed to favour solutions with these characteristics over those which lack them.

*Diversification* means to move the search trajectory away from the current solution, that is, to change the structure of the current solution so that the new neighbourhood search leads to exploration of a different region of the search space. Diversification is necessary when an intensification phase fails to further improve the solution, that is, when intensification finds a local optimum. In memoryless search algorithms such as pure local search, diversification is effected by re-starting the process with a new randomly generated initial solution. In contrast, tabu search uses its memory of previous solutions and moves to create diversification in a systematic and largely deterministic manner.

Thus, diversifying moves are (usually non-improving) moves that lead away from a current local minimizer, or moves which specifically change the structure of the current solution. For example, in the packing problem, the goal is to place unequally weighted objects into boxes so that the total weight is distributed evenly across the boxes. In this
problem, a diversifying move is to swap two objects which are very different in weight, thus significantly changing the composition of two of the boxes' contents. [Dowsland and Dowsland 1992, Dowsland 1993] In the harvesting and road building problem for forest planning, (Chapter 7 of this thesis), choosing moves which reduce the number of road links required effects one type of diversification.

The basic method for achieving diversification in tabu search is the enforcement of tabu tenure, creating a restricted neighbourhood derived from the set of allowed moves. This forces the process to look for moves which do not reverse recently taken moves, thus leading the search to new feasible regions. Because tabu tenure is typically small, the imposition of tabu status is often referred to the use of short-term memory, or recency-based memory. In the medium to long-term phases of the search, diversification can be implemented by discouraging or penalizing moves which have occurred frequently in the past. This is often accomplished by augmenting the cost function with a penalty term which is proportional to the frequency of the move, for example:

\[ z(x \oplus m, H) = z(x) + \frac{aF_m}{t}, \text{ where} \]

\[ t = \text{current iteration} \]

\[ F_m = \text{number of times move } m \text{ has been chosen} \]

\[ a = \text{a suitable constant} \]

Stochastic diversification schemes may be designed by selecting a series of random moves -- this is akin to a random re-start method. The choice of the number of random moves may be a free parameter (i.e. fixed but determined prior to search) or may be determined dynamically by examining the history of the search [Battiti 1995]. Stochastic diversification adds robustness to the algorithm, and is thus desirable, especially when the solution space is highly non-convex or discontinuous.
Intensification and diversification strategies interact and oppose each other to form the fundamental cornerstones of longer term memory in tabu search. [Glover 1991a] At each stage of the search, an appropriate choice of intensification or diversification is made by examining the state of the search. For example, at the initial or startup phase, intensification is required. Once a local minimizer is found, diversification by tabu status (short term memory) is used to force the search into new areas, hopefully finding new minimizers. In the mid-range and later stages of the search, diversification which encourages moving to new local minima by changing the solution structure is appropriate. These are long-term memory processes, and often involve penalizing frequently taken moves or solutions with key elements that have occurred frequently in the past. Thus, the design of tabu search algorithms that effectively balance intensification and diversification is a major challenge to researchers. The most effective schemes use dynamic rules to modify tabu tenure and $z(x,F)$ throughout the search duration. [Glover, Taillard and de Werra 1993, Soriano and Gendreau 1996

4.4. Dynamic Rules for Evaluation and Prohibition

Tabu search algorithms in which the evaluation function is not changed and the tabu tenure is fixed have been classified as fixed tabu search in Battiti [1995]. Fixed TS has been found to be quite successful in getting good solutions to many hard problems. The difficulty with fixed TS is in determining a good value for tabu tenure. Its magnitude determines the effectiveness of the algorithm in escaping from the "attraction basin" of a local minimizer [Battiti 1995]. If tenure is too small, not enough moves are prohibited and the search will return again and again to the same minimum point. If the tenure is too large, the search is over-constrained and is likely to miss other good minimizers.
Often, an extensive period of experimentation with problem instances is required to find a good value for tabu tenure. Since this is an empirical determination, the value found may be dependent on problem instance -- that is, it may depend on the magnitude (or relative magnitude) of cost function coefficients, or on the constraint structure. This experimentation period can constitute a lengthy trial and error process, sometimes with intervention by the designer of the algorithm. Moreover, if the statistical properties of the search space vary widely in different regions, a tabu tenure that is suitable for one region may be unsuitable for another [Battiti and Protasi 1995].

In the quest for more robust and more generally applicable TS procedures, researchers have developed methodologies and data structures to incorporate dynamic rules for move evaluation. These rules include modifying parameters which define the extent to which the search history \( H \) is used in move evaluation, and the introduction of different move selection rules at different stages in the search progression. In addition to deterministic rules, randomness is often incorporated into the algorithm, in a controlled way, to increase the robustness of the algorithm. For example, researchers have found that a relatively straightforward modification of fixed TS which is to vary tabu tenure randomly within a range improves the search performance for some problems [Dell'Amico and Trubian 1993, Gendreau et al 1994].

Intelligent dynamic choice rules have been used by Gendreau et al [1994, 1992] to solve vehicle routing and traveling salesman problems. They chose a tabu tenure \( \theta \), drawn randomly from the interval [5,10]. Others [for example, Dell'Amico and Trubian 1993] allow tabu tenure to vary on an interval \([t_{min}, t_{max}]\), and then invoke diversification and intensification strategies by increasing (respectively, decreasing) \( t_{min} \) and \( t_{max} \). Woodruff and Spearman [1992] studied TS formulations with varying levels of diversity, and found that better results were always found when some level of diversity is
incorporated. Mooney and Rardin (1992) postulated that relatively high diversification is a necessary condition for finding good solutions using TS for complex problems. Battiti and Tecchiolli [1995] have gone even farther in determining the appropriate tabu tenure dynamically as the tendency of the search process to cycle back to known solutions is detected. Their self-tuning framework for tabu search is called the Reactive Tabu Search.

4.5. Reactive Tabu Search

A group of researchers at Trento University in Italy, led by Dr. Roberto Battiti, have developed the Reactive Tabu Search framework. The system is described in detail in [Battiti and Tecchiolli 1994, 1995]. This section summarizes the main ideas of reactive tabu search.

Reactive Tabu Search (R-TABU) is a self-tuning heuristic, integrating simple feedback schemes to determine the value of tabu tenure at each iteration. R-TABU is aimed at providing a framework which is applicable to a wide range of discrete optimization problems, while avoiding the trial and error adjustment of tabu tenure. The R-TABU process uses the past history of the search to dynamically tune parameters (tabu tenure), and to automate the balance of diversification and intensification in the search process. Let \( L \) be the number of possible moves, and denote by \( T^{(t)} \) the tabu tenure at iteration \( t \). The search trajectory at iteration \( t \) is the set of visited solutions \( \{x^{(1)}, x^{(2)}, \ldots, x^{(t)}\} \). R-TABU is based on detecting the repetition of previously visited solutions in the search trajectory.

In R-TABU, tabu tenure \( T \) is always less than \( (L-2) \), ensuring that at least two moves are allowed at each iteration. The level of diversification is controlled by \( T \): the larger \( T \) is, the longer the distance (i.e. the length of the search trajectory) must be before
the trajectory will cycle back to a previously visited solution, and hence the greater the diversity. In some applications, the minimum cycle length can be determined from \( L \) and \( T \). For example, in the case where the solution space is the set of binary strings of length \( L \), and moves are to complement one entry in the string, the minimum repetition interval is \( 2(T + 1) \). That is, \( x^{(r+1)} = x^{(r)} \Rightarrow r \geq 2(T + 1) \).

In R-TABU, \( T \) is set equal to one at the beginning of the search. If solutions are repeated, \( T \) is increased to an amount proportional to the moving average of the length of short cycles. After an increase has been made, if no repetitions occur for a sufficiently long period, \( T \) is decreased by a constant factor. This scheme is designed to "break" short cycles. Long cycles, for example those longer than \( 2L \) in the binary string problem, are not prevented by this mechanism. Battiti [1995] also recognizes that the search trajectory may be trapped in a restricted region of the solution space even though no limit cycle exists. In analogy with the chaotic attractors of dynamical systems, this type of behaviour is called chaotic cycling, and also is not prevented by the above scheme. To break long cycles, exit from a "chaotic attractor" and to increase robustness, R-TABU uses an "escape" mechanism, which is to execute a random number of randomly selected moves. The escape system is triggered when too many solutions are repeated too often, indicating that chaotic cycling has occurred.

The ability to detect the repetition of previous solutions easily is essential to the R-TABU framework. Cycling is checked at each iteration, and thus it is critical that a time and space efficient method be available to include this information in the search history. The reverse elimination method, used in strict tabu search, is space efficient but excessively taxing computationally. In R-TABU [Battiti and Tecchioilli 1995], hashing vectors and digital tree structures are used to store the incidence of solutions. In Battiti [1996], persistent dynamic sets implemented on red-black trees are proposed to store
previous solutions, using hashing techniques [Knuth 1973, Woodruff and Zemel 1993] to provide search indexes. This method is efficient with respect to computation time ($O(L)$ in the worst case) and space ($O(I)$).

The authors have shown that R-TABU outperforms other algorithms for the Quadratic Assignment problem [Battiti and Tecchiolli 1995] and the Maximal Clique problem [Battiti and Protasi 1995]. R-TABU, augmented with strategic oscillation, forms the basis of the algorithm presented in this thesis to solve the forest harvest scheduling problem.

4.6. Strategic Oscillation

A substantial proportion of the reported TS algorithms restrict the neighbourhood of a solution to the feasible region. However, complex constraint structures may tend to trap the search trajectory in a subspace of the feasible region. That is, it is not obvious how to select a sequence of feasible moves that cause the search to transition from one region to another. Glover [1990a] proposed the idea of "strategic oscillation", where the search trajectory is first moved away from a certain solution, then drawn back towards it. This concept is applied to oscillating through the boundaries of the feasible region. The concept is that allowing infeasible moves may result in the search process more quickly transitioning to new local minima. The infeasible move or moves become a "stepping stone" to a previously unexplored feasible region. [Woodruff and Spearman 1992, Glover 1990a, Mooney and Rardin 1992, Gendreau et al 1991].

If the moves and the structure of the problem $P$ are such that we can categorize moves that lead into and out of the infeasible region, oscillation can be achieved by executing a series of such moves. For example, in the knapsack problem, the constraint is an upper bound on the total weight or cost of the items placed in the knapsack. Moves
which add items "push" the solution into the infeasible region, and moves which delete items will lead the solution back to the feasible region.

For more complex problems with more complicated constraints, specifying oscillating by defining move type sequences may not be possible. One way to achieve oscillating behaviour (on any problem) is to add penalty terms to the evaluation function, with coefficients that are dynamically updated depending on the feasibility of a recent set of accepted solutions. Specifically, suppose that there are $m$ constraint sets in the problem to define the feasible region $X$. We add $m$ penalty terms $F_j(x)$ to $z(x)$, each of which measures the deviation of a solution from the feasible region with respect to that constraint set. Each penalty term is multiplied by a positive coefficient $\rho_j$, and each $F_j(x)$ is non-negative.

$$z(x, H) = z(x) + \sum_{j=1}^{m} \rho_j F_j(x), \rho_j > 0, \text{and}$$

$$F_j(x) = 0 \quad \text{for } x \in X$$

$$F_j(x) > 0 \quad \text{for } x \notin X$$

The coefficients $\rho_j$ are modified throughout the search to encourage the acceptance of a mixture of feasible and infeasible solutions. If feasible solutions are to be encouraged, the coefficients are increased. Otherwise, if infeasible solutions are required, the coefficients are decreased. In this way, a systematic oscillation through boundaries of the feasible region is effected. Gendreau et al [1991] used this approach in solving the vehicle routing problem. They showed that, for the VRP, their penalty function approach was far superior to tabu search methods which maintained feasibility at each step. This method is used in the TS algorithm for solving the forest harvesting problem (Chapter 7).
CHAPTER 5

HARVEST SCHEDULING AND ROAD BUILDING PROBLEM
(HSRBP)

In this chapter, definitions and data structures to formulate the tactical harvest scheduling and road building problem HSRBP are developed. This model can be used to analyze the tradeoffs between lost biological productivity and the costs of road building, while meeting timber volume goals and conforming to environmental restrictions.

Section 5.1 is a general description of the purpose of the model and the assumptions on which it is based. In section 5.2 the spatial decision units, stands and openings, are defined. In section 5.3, the lost volume function which is used to penalize inappropriate timing of harvests is described. Section 5.4 describes the harvest volume constraints which originate in the strategic planning process.

5.1. Model Assumptions

Assume a hierarchical planning structure such as the one described in chapter 1, where a strategic planning process has been completed. The model is intended for tactical planning horizons of approximately fifteen to twenty-five years, with planning periods of length in the order of five years. Although many different tactical level goals are worth considering, this work will be restricted to minimizing negative impacts on forest productivity caused by timing of harvests, and minimizing the costs of road construction. The model is then constrained to meet harvest volume bounds for each planning period and for the total harvest, these target bounds having been derived from the strategic planning process. These bounds represent a harvesting goal which is
consistent with long-term sustained yield requirements, where the objective was to maximize the economic benefits of timber harvesting. Thus, unlike most of the tactical models in the literature, this model does not have an objective to maximize volumes or net present value of timber products. The model is further constrained to meet spatial and temporal environmental restrictions for green-up periods and maximum clearcut size.

Harvest decision units for this model are the forest stands, that is, contiguous areas of the forest which are homogeneous in cover type, age and expected future growth potential. Only one method of harvest is employed (clearcutting), and other silvicultural activities are not considered. Negative effects of the timing of harvests on forest productivity are measured by the Lost Volume Function as described in Section 5.2.

Both stand scheduling and road building are required elements of tactical harvest planning problems. A network of road links that covers the area has been designed and represents the input to the road network selection problem, discussed in detail in Chapter 6, section 6.1. For the purpose of defining the model generally, it is sufficient to note that the road building costs are assumed to constitute the major costs of accessing stands in the forest. Other costs, such as skidding/hauling costs are omitted, as are any restrictions on transportation flows over the road network. The management of the forest stands is assumed to be under a single "owner", such as a provincial government, for which road maintenance is an on-going activity. Thus, road maintenance costs are not a cost factor in the model and the roads are permanent for the duration of the tactical planning horizon. No upper limit was placed on the amount of road construction which may be scheduled for any period, and any effects on wood volumes or opening sizes due to road construction are ignored.

The primary decision variables in this model are the stand harvest decisions. The model treats road building as a consequence of a harvest schedule. There are (at least)
two other ways to look at this problem. First, it is conceivable that roading and harvest decisions be considered simultaneously. The implication of using such a model is that, when a search algorithm is used, moves could be taken which render the road network infeasible with respect to its own connectivity and with respect to covering access to all stands scheduled. Thus this strategy was discarded, and it was decided that a model which separated roading decisions from harvest decisions was more likely to be successful. Since the roading sub-problem can easily be made feasible for any harvest, it was decided to have roading be a consequence of harvest decisions.

There is no commensurable unit of measure for the lost volume penalty and the costs of road construction. Lost volume represents a loss of productivity, the difference between the volume which could be obtained under best timing for each stand and that which is expected under the proposed schedule. Road link construction costs are estimated in dollars, depending on the length of the road link and, possibly, topological factors. This cost is discounted across the planning horizon to include a "hedge" against uncertainty. It is not at all clear that a sensible method of assigning dollar values to lost volume could be derived. Even if a conversion factor or method was clearly indicated, the question of the appropriate way to discount the "lost volume dollars" arises. Yet, clearly, the cost of roading and the cost of harvest timing are opposing factors in HSRBP. For these reasons, a tradeoff analysis to compare the effects of different weightings of the two cost factors is recommended.

To summarize, the HSRBP is to generate a schedule of stand harvests and road construction that:
**Minimizes:** Lost Volume Costs + Road Building Costs

**Subject to:**
- Harvest volumes being within bounds for each period;
- Total harvest volume being within bounds for the planning horizon;
- Maximum clearcut sizes and green-up period requirements being met;
- Road Network feasibility (operational feasibility).

### 5.2. Stands and Openings

Stand characteristics include the species composition, age of trees, stocking and site capability classification. Geographically, stands have defined boundaries and known area.

**Definition 5.1** A *stand* is a contiguous area of the forest which is homogeneous in cover type, age of trees and site capability.

Suppose that the forest area under consideration is composed of $S$ stands, and that the planning horizon consists of $J$ periods.

**Definition 5.2** A *harvest schedule* is a vector $x$ with elements $x_s = j, s = 1,...,S, j \in [0,J+1]$, where the value assigned to each element of $x$ is the harvest period for that stand.

Stands which begin period 1 in a clearcut state are assigned $x_s = 0$; stands which are not scheduled for harvest within the planning horizon are assigned period $j=J+1$.

**Definition 5.3** The *forest adjacency graph* or forest graph $FG=(S,E)$ is a set of nodes $S$, one for each stand, and an edge $e$ in $E$ for each pair of stands which share a common boundary.

The forest graph may consist of a set of connected subgraphs. This happens when the region under consideration includes non-forested areas which then "disconnect" the forest graph. It is also worth noting that adjacency may, in some jurisdictions, be defined by the proximity of one stand to another. That is, stands are deemed to be adjacent if
they are within a minimum distance of each other. The adjacency graph has the same
definition regardless of which method is used to define stand to stand adjacency, but the
"proximity" definition of adjacency results in a denser adjacency graph.

An opening in the forest is a contiguous area which has been clearcut and not
regenerated. Most jurisdictions impose a maximum acreage on the size of openings
(\textit{maxopen}), and require that areas adjacent to openings be left forested for a period
known as the "green-up" period. Openings are determined by the harvest schedule \(x\), the
adjacency delay time, and the forest adjacency graph \(FG\).

\textbf{Definition 5.4} The \textit{adjacency delay time} \(ADT\) is the number of planning periods for
which green-up period is required on adjacent openings.

In many jurisdictions, \(ADT\) is set at 10 years, or 2 planning periods in the context
of this tactical model. The NS Forest/Wildlife Guidelines specify adjacency delay and
maximum opening restrictions as "Where total area would exceed 50 hectares (125 acres)
avoid clearcutting stands adjoining a clearcut area, until the regeneration in the original
clearcut is at least two metres (six feet) or else provide appropriate wildlife corridors."
[Province of Nova Scotia 1990] For most stands in Nova Scotia, this translates to an
adjacency delay of approximately ten years.

Now, if stands are pre-blocked so that the combined acreage of any two stands is
greater than \(\text{maxopen}\), the imposition of the maximum opening and adjacency delay
requirements is effected by prohibiting harvest of adjacent blocks in consecutive periods.
Since this model is stand based, requiring adjacency delay on neighbouring stands
imposes a more restrictive condition than that specified in the guidelines, since most
stands are significantly smaller than \(\text{maxopen}\). These guidelines can be interpreted to
mean that openings can be increased to the maximum size over the number of periods in
the adjacency delay time. Since openings are not pre-blocked, a different method of
specifying these conditions is needed. This is done by first defining multi-period openings, and then imposing conditions on these openings.

**Definition 5.5** The set of stands which, under schedule $x$, would be clearcut within one green-up interval ending at period $p$ is $S_p(x) = \{ s \mid x_s \in \{ p - ADT + 1, ..., p \} \} .

**Definition 5.6** $E_p(x)$ is the set of edges of $E$ with both end-nodes in $S_p(x) .

**Definition 5.7** $F_p(x) = (S_p(x), E_p(x))$ is the subgraph of $FG$ induced by $S_p(x)$ and $E_p(x)$.

**Definition 5.8** The opening period $p$ is the maximum of the periods assigned to nodes in an opening.

**Definition 5.9** An opening with opening period $p$ $O^p_k(x)$ is a connected subgraph of $F_p(x)$.

Thus, openings are induced by a harvest schedule and the adjacency delay time, and each stand in an opening of period $p$ is assigned to a harvest period within the adjacency delay time window terminating at period $p$. For example, if the number of planning periods is four and the adjacency delay time is two, openings of period one include stands already clearcut and those scheduled for period one, period two those scheduled for periods one or two, and so on. Thus, each stand which is scheduled for harvest may be a member of as many as two openings. Figure 5.1 illustrates multi-period openings for an $ADT$ of two periods.

With this definition of openings, both the adjacency delay and maximum opening size constraints are imposed by the condition that every opening (as defined above) be of size not exceeding $maxopen$. Thus the spatial constraints on the harvest schedule are given by:

$$\sum_{s \in O^p_k(x)} a_s \leq maxopen, k = 1, ..., K^p(x), p = 1, ..., j \quad (1)$$
Since openings are derived from a given harvest schedule $x$, there is no convenient method of specifying these constraints in an integer programming formulation. Enumeration of all potential openings which could cause a $\text{maxopen}$ violation is possible, but not practical since the number of possible openings is exponential in the number of stands. However, the computational complexity of graph searches to calculate these openings is rather trivial in this application, since the forest graph will be sparse. Thus, although this is not a useful model for integer programming methods, it will prove to be workable for our heuristic search methods.

![Figure 5.1. Multi-period Openings](image)
5.3. Lost Volume Penalty

Appropriate timing of stand harvests is of major importance in realizing long-term sustainablility of the forest resource and in maximizing its productivity. In this model, a lost volume penalty function is used to differentiate between effective and ineffective timing of harvests. This penalty function, as a factor in the objective function to be minimized, also favours solutions that attempt to recover from existing inefficient conditions (over abundance of old stands and dead stands). It also discourages those solutions which would incur losses in the future, i.e. outside the planning horizon for which the model is solved.

Definition 5.10 The mean annual increment, $ma_i(g)$, of a stand at age $g$ is the average wood volume increase per unit area per year at full stocking.

Mai depends on the existing covertype, age and site capability of the stand [Gunn 1994b]. Figure 5.2 shows $ma_i$ curves for stands of site index 3 - 8 for Nova Scotia softwoods (a higher numbered site index indicates better growth potential). If the biological production potential of the forest is to be maximized, then stands should be scheduled for harvest when they have reached their maximum $ma_i$. The lost volume penalty function was developed to form a proxy measure for negative effects on potential productivity which can be attributed to decisions made within the planning horizon. The lost volume penalty induced from harvesting stand $s$ in period $j$ is a function of the stand size, age, stocking, $ma_i$ and its current growth state (growing or moribund). For growing stands the contribution to lost volume from assigning stand $s$ to harvest in period $j$ is:

$$LV_{growing}(s, j) = \left[ma\_max(s) - ma_i(s, j)\right] \times a_i \times age(j) \times stocking$$  \hspace{1cm} (2)

If stand $s$ is harvested at its period of maximum productivity, $ma\_max\_age(s)$, the penalty is zero. Suppose that a stand is harvested at period $j$, before its peak productivity period. Then, the incurred penalty is equal to the difference between the volume which
is achieved at that harvest and the maximum volume which could be achieved by harvesting at \( \text{mai}_\text{max}_\text{age}(s) \). If the stand has reached its peak productivity at some period \( k \) in the planning horizon or prior to the inception of the first planning period, then a decision to delay harvest of that stand results in the land being kept in a declining state of productivity. Suppose that the stand is eventually harvested at period \( j \). Then, the penalty is derived from observing that, for \( \text{age}(j) \) years, the land has been occupied by a stand producing at the rate \( \text{mai}(s,j) \), which is less than \( \text{mai}_\text{max}(s) \). Assuming that the next rotation harvest for the stand occurs at maximum productivity, the appropriate penalty is the volume corresponding to the difference in \( \text{mai}_\text{max}(s) \) and \( \text{mai}(s,\text{min}[j,J+1]) \). The lost volume penalty, being the sum of all stand penalties, will evaluate to zero only if the scheduling of all stands is such that productivity has not been harmed. Therefore, minimizing the lost volume penalty is a sensible proxy for maximizing forest productivity.

In figure 5.3, the stand is at ageclass 7 in period 1 of the planning horizon, and reaches maximum \( \text{mai} \) at ageclass 10. The figure illustrates the penalties for harvest in period 1 (early) and in period 5 (late).

For stands which have a significant amount of dead timber, the lost volume function is defined differently. The presumption is that the stand will resume normal growth after the inhibiting deadwood is removed and some remedial silvicultural action taken. Thus, for \( \text{dead} \) stands, the penalty is based on the number of periods into the planning horizon that no intervention is taken.

\[
LV_{\text{dead}}(s,j) = \begin{cases} 
0 & \text{if } j = 1 \\
\text{mai}_\text{max}(s) \times (j-1) \times a_s \times \text{stocking} & \text{if } j \in [2,J+1]
\end{cases}
\]  

Thus, if the stand is cleared in the first planning period, no penalty is applied, since this is the best action that can be taken. Otherwise, the delay to future periods incurs a
penalty calculated from the stand's expected future growth at maximum \textit{mai} (Figure 5.4 illustrates this penalty for waiting until period 4 of the planning horizon.).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5_2.png}
\caption{Mean Annual Increments}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5_3.png}
\caption{Lost Volume Function, Growing Stands}
\end{figure}
5.4. **Harvest Volume Restrictions**

Target harvest volumes for each period and for the overall harvest are derived from a strategic planning process. For the tactical model, periodic volumes and total volume are allowed to deviate from target values by a percentage of the target. A larger percentage deviation is allowed on periodic harvests than on total.

Unlike many other tactical harvest scheduling models, this model does not attempt to maximize harvest volumes or net present value of timber. The premise is that the harvest targets have been derived from a strategic model, where maximization of timber product value is done under sustainability constraints over long planning horizons. Actual harvest targets (often called annual allowable cut) produced by a strategic optimization model over this longer horizon with a non-declining yield constraint, will be attained by "harvesting" stands at ages near their peak biological productivity in most cases. Thus, this tactical scheduling model will maintain consistency with the strategic plan by also scheduling the stands at the "best" times where possible. Re-optimizing for
volume in the tactical planning phase is not useful in this hierarchical framework, and may tend to favour schedules which are inconsistent with the upper level strategic planning model assumptions. Thus, within the tactical horizon, the goal is to implement the strategic plan while minimizing costs and meeting spatial requirements.

Define parameters:

\[ s_j \quad \text{volume from harvesting stand } s \text{ in period } j \ (\text{ft}^3) \]
\[ TV \quad \text{Total volume target \ (ft}^3) \]
\[ dTV \quad \text{Percentage Deviation allowed from Total volume target} \]
\[ PV_j \quad \text{Periodic Volume Target, period } j \ (\text{ft}^3) \]
\[ dPV \quad \text{Percentage Deviation allowed from Periodic harvest target} \]

and indicator variables

\[ x_{sj} = \begin{cases} 
1 & \text{if stand } s \text{ is scheduled for harvest in period } j \\
0 & \text{otherwise} 
\end{cases} \]

Then, the harvest constraints are:

\[ TV(1-dTV) \leq \sum_j \sum_s V_{sj} x_{sj} \leq TV(1+dTV) \quad \text{Total Target} \quad (4) \]
\[ PV_j(1-dPV) \leq \sum_s V_{sj} x_{sj} \leq PV_j(1+dPV) \quad j = 1, \ldots, J \quad \text{Periodic Targets} \]
\[ \sum_j x_{sj} \leq 1 \quad s = 1, \ldots, S \quad \text{One harvest/stand} \]

The model is then summarized:

\[ \min \sum_{j=1}^{J} \sum_{c=1}^{C} L(s,j)x_{sj} + \rho \text{ RC} (x) \]

Subject to:

Spatial Constraints \quad (1)

Harvest Constraints \quad (4)

and operational feasibility for the road network.

\textbf{RC}(x) is the cost of constructing road links so that all stands are accessed. These are determined by solving \textbf{RNP}(x), the road building problem which is induced by a
harvest schedule $x$. The parameter $\rho$ is increased (respectively decreased) to drive the search algorithm to produce solutions with lower (respectively higher) roading costs. The road network problem and the heuristic procedure for solving $RNP(x)$ are fully described in Chapter 6.
CHAPTER 6

ROADING NETWORKS

In the previous chapter, the harvest scheduling component of HSRBP was defined. This chapter deals with the road building optimization problem. Each harvest schedule $x$ induces a road network design problem $RNP(x)$. This problem is to select from a pre-specified set of road links a minimal cost set which is operationally feasible for $x$. Multiple access points and cycles in the network are allowed, and in this sense this model deals with more general road networks than those which have been treated to date in the literature.

In this chapter, a graph structure and an integer programming formulation for $RNP(x)$ are first developed. A solution to the discounted, multi-period road building problem is a directed spanning graph of the road network. It is shown that $RNP(x)$ is NP-complete by transforming the single-period, undiscounted road design problem to a Steiner problem on graphs. A heuristic solution procedure, the Path Heuristic, $PH$, is developed, and is shown to produce near-optimal solutions to $RNP(x)$ with optimal solutions in many cases. In addition, $PH$ is especially suitable for use in the tabu search algorithms described in the succeeding chapter. This is because the initial solution to the roading problem, a set of shortest directed paths, is easily used to compute incremental costs of road building when moves are implemented in the tabu search algorithm.

6.1. Creating the Proposed Road Network

A roading system is required to analyze the cost of gaining access to stands
assigned to be harvested in any schedule. If no road network, or only a partial network, exists for the area under study, a proposed system must be created. Its development occurs in two phases. They are the spatial definition of suitable road links and the mathematical definition of the graph representing the road network and the stand/road requirements.

The essential requirement for the proposed roading system is that it defines a connection from every stand in the area to a main road. That is, the network model of the roads must contain at least one path from each stand to one of the existing main roads. Apart from conforming to this connectivity requirement, the network must be designed with sufficient care that it provide a fair representation of the problem at hand. The role of the road model is to approximate the real costs of road building which would be incurred in accessing stands in the forest interior. Thus, the design requirements are that the network be connected, that it make topological sense, and that there be links defined within a minimum distance of each stand.

The proposed links are sketched on a map of the forested area, using the major topological features (e.g., rivers, streams, existing main roads) to determine link placement. If a GIS system and mapped coverage of the area is available, the road design process is considerably enhanced by utilizing the system's spatial analysis capabilities to calculate link lengths, verify connectivity and coverage of the area and assign link identifiers. The effort expended in adding the road network to the GIS will be returned "with interest" when proposed solutions are represented graphically. Once all road links have been created, the stand to road link relationships are defined by assigning the nearest link to each stand. (Stands which are already accessible by main road are assigned a null link value.) Again, this process is less awkward when a GIS system is available. The example network in Figure 6.1 has three existing main roads and sixteen
proposed links.

![Diagram of stands and roading network]

Figure 6.1. Stands and Roading Network

6.2. Graph Representation of the Roading Network

The network is represented as a weighted graph \( RG = (V,R,W) \), where the edges \( R \) are road links and each weight \( w_r \) is the cost to build link \( r \). The cost, \( w_r \), may be the length of link \( r \), although other factors such as the nature of the terrain, road elevation and proximity to water buffers may be included in determining the link construction cost. The graph \( RG \) may be non-euclidean when factors other than length are chosen.

The vertices \( V \) are end nodes of the links, which are indexed in any unique fashion, with the proviso that only one node index is used to indicate that a link accesses a main road. That is, all existing main roads are "collapsed" into one node \( \alpha \). The network in figure 6.1 has five links which are connected to access roads (1,8,11,12,16).
Its graph is $V = \{\alpha, a, b, c, d, e, f, g, h, i, j, k, l\}$, $R = \{(\alpha, a), (a, b), (b, c), (a, f), (f, g), (g, e), (e, d), (h, \alpha), (g, h), (c, d), (d, \alpha), (c, \alpha), (j, i), (i, k), (k, l), (i, \alpha)\}$, as shown in figure 6.2.

![Figure 6.2. Graph Representation of Road Network](image)

Some comments on the nature of these graphs are in order. First, several links may be constructed beginning at a main road, (depending on the length of the road which is incident to the forest area), and the node $\alpha$ may represent more than one of these roads. Thus, the degree of $\alpha$ is expected to be large compared to that of other nodes in the graph. For nodes other than $\alpha$, the graph is likely to be relatively sparse. To illustrate this, consider the design which is an $N$ by $N$ gridding of the forest area, shown in figure 6.3. Each interior node has degree 4, the number of nodes is $(N-1)^2 + 4(N-1)$, the number of edges is $(N-1)N + 2(N-1)$, and thus the average node degree is

\[
\frac{(N-1)N + 2(N-1)}{(N-1)^2 + 4(N-1)} = \frac{N + 2}{N + 3}.
\]

This is arguably the most dense pattern of potential road links that would be designed, and thus the highest vertex degree we would expect to find in RG (for nodes other than $\alpha$) is 4.
6.3. Integer Programming Formulation for $\text{RNP}(x)$

Denote by $\text{RNP}(x)$ the problem of finding a minimum cost road plan for a given harvest schedule $x$. To specify $\text{RNP}(x)$ as an integer program, some parameters, sets and variables are required. As well, for each link in RG which is a member of a cycle, paths from that link to the main roads must be enumerated. This enumeration is practicable on graphs which are sparse as described above, since only a few possibilities exist for each link, and not all links will be in a cycle. This is not a generally applicable method, since too many cycles or a denser graph would render the enumeration unworkable. See Magnanti and Wolsey [1995] for a description of network design models. Nevertheless, this enumeration was done for this problem to produce an integer program which was then used to generate optimal solutions for the road building problem. This section describes the $IP$ formulation for $\text{RNP}(x)$. 
Parameters
\[ w_n \] discounted cost to build link \( r \) in period \( t \)
\[ S \] Set of stands
\[ J \] number of planning periods
\[ R \] Set of road links

Sets
\[ CC = \{ s \in S | x(s) = 0 \} \] Existing Clearcuts
\[ \delta (r) = \{ l \in R | (r,l) \text{ is connected to } r \} \] The cutset of \( r \).
\[ D(r) = \{ l \in \delta (r) | l \text{ is connected to } t \text{ in } RG(R \setminus \{ r \}) \} \]
\[ S_r = \{ s \in S | CC \text{ harvest of stand } s \text{ requires link } r \} \]
\[ P_r = \{ \text{Paths which connect link } r \text{ to } \alpha \} \] Paths from \( r \) to \( \alpha \)
\[ = \{ P_r \} \] All Paths

Decision Variables
\[ z_{nj} = \begin{cases} 
1 & \text{if link } r \text{ is built in period } j \\
0 & \text{otherwise} 
\end{cases} \]

Indicator Variables
\[ Y_{pk} = \begin{cases} 
1 & \text{if path } p \text{ is built in or before period } k \\
0 & \text{otherwise} 
\end{cases} \]

Timing and connectivity constraints for this problem must be included to ensure that, for each link \( r \) which is built in period \( j \), some path which links \( r \) to \( \alpha \) is also built in period \( j \) (or earlier). For links which are members of a sub-tree of \( RG \) containing \( \alpha \), the timing and connectivity requirements are specified by examining the cutset of \( r, \delta (r) \) and constructing the set \( D(r) \). For these links, at least one of the links in \( D(r) \) must be built in the same period or in a prior period.

\[ z_{nj} \leq \sum_{k=1}^{l \in D(r)} z_{ik} \] \hspace{1cm} (6.1)

For links which are members of cycles in \( RG \), indicator variables \( Y_{pk} \) and an enumeration of all paths to \( \alpha \) are required. For each link \( r \) which lies in a cycle of \( RG \) containing \( \alpha \), the paths \( P_r \) are enumerated and indicator variables \( Y_{pk} \) used to form constraints (6.2).
\[ |p_i| Y_{ik} \leq \sum_{j \in 1}^{t} \sum_{r \in p_i} z_{rj} \]  \hspace{1cm} (6.2)

\[ z_{rk} \leq \sum_{j=1}^{t} \sum_{p \in p_r} Y_{ij} \]  \hspace{1cm} (6.3)

where \(|p_i|\) is the cardinality (number of edges) of the path \(p_i\). Equations 6.1, 6.2 and 6.3 then ensure connectivity of the road network.

For any harvest solution \(x\), the elements of \(x\) which are member of \(S_r\) provide a bound on the period at which link \(r\) must be built. Define

\[ j_{\text{min}}(x) = \{ \min_{1 \leq s \leq K} \} \mod (J+1). \]

Then the constraints

\[ \sum_{j=1}^{j_{\text{min}}(x)} z_{rj} \geq 1, r = 1, |R| \]  \hspace{1cm} (6.4)

ensure that the required link is built prior to harvest.

The optimization problem \(RNP(x)\) can be expressed as the following (binary) integer programming problem:

\[ \text{RNP1: } \min \sum_{j=1}^{J} \sum_{r=1}^{R} w_{rj} z_{rj} \]

Subject to constraints 6.1, 6.2, 6.3, 6.4 and

\[ \sum_{j=1}^{J} z_{rj} \leq 1, r = 1, |R| \hspace{1cm} \text{Construct each link at most once} \]

\[ z_{rj} \in \{0,1\} \hspace{1cm} \text{Binary variables} \]

### 6.4. Equivalency to Steiner Tree Problem

The set of links which are required to be built during the planning horizon, \(RL\), is the set \(\{r | j_{\text{min}}(x) > 0\} \). A feasible solution to \(RNP(x)\) is a directed connected subgraph \(S (V', R')\) of \(RG\) such that, for each \(r\) in \(RL\), there is a directed path in \(S\),

\[ P_r = \{r, r_n, r_{n-1}, \ldots, r_1\} \]

from the required link \(r\) in \(RL\) to \(\alpha\). That is, the ending link in this
path, \( r_n \), has one node equal to \( \alpha \). Each link in each path is assigned a construction period, a mapping \( \Omega : R \rightarrow \{1, J + 1\mid \Omega(r_n) \leq \Omega(r_{n-1}) \leq \ldots \Omega(r_1) \leq \Omega(r)\} \). An optimal solution is one of minimum weight, i.e. one that minimizes the discounted costs of road building \( \sum_{r_{\Omega(r) \leq j}} \frac{w_r}{(1+d)^{\Omega(r)}} \) (\( d \) is the discount rate).

This network design problem has two special features: the requirement for timing the road link construction and the existence of required links. The second observation leads to a comparison of \( RNP(\alpha) \) with the classic Steiner Tree problem.

If timing of road construction is dropped from the problem, (i.e., let \( J=1 \)), \( RNP(\alpha) \) is reminiscent of the Steiner tree problem on graphs.

### 6.4.1. The Steiner Tree Problem on Graphs

The Steiner tree problem (STP) on a weighted undirected graph \( G=(V,E) \) is to find a minimum cost connected subgraph of \( G \) which contains some subset \( T \) of \( V \). The nodes \( T \) are called "Terminal" or "Special" nodes. The optimal solution to STP is always a tree of \( G \) (a connected subgraph of \( G \) with no cycles), which contains all of the terminal nodes \( T \) and possibly some other nodes in \( V \setminus T \). These "extra" nodes are called Steiner nodes. If \( T \) is equal to \( V \), the problem reduces to the minimum spanning tree problem and is solved in polynomial time using Dijkstra's labeling algorithm, or some variant such as Prim's or Kruskal's algorithm. Other special cases are also solvable in polynomial time, but in general the Steiner tree problem is NP-complete.[Garey and Johnson 1979]

STP arises in many network design problems. For example, suppose that the problem at hand is to install a minimum cost fibre optic network that connects a set of customers (nodes). Any optimal solution must be a minimum spanning tree on the graph of customers (nodes) and possible connections (edges). If, however, not all customers
must be connected, and other nodes can be used to complete the network, the problem becomes a Steiner tree problem, where the terminal nodes \( T \) are the "essential" customers. In VLSI design, components (chips) must be connected along horizontal and vertical channels using the least amount of wire. Terminal nodes are those locating the chips, and intermediate nodes are the intersections of the horizontal and vertical channels. This problem is referred to as the rectilinear Steiner tree problem since edge costs are measured using the rectilinear norm. [Magnanti and Wolsey 1995, Hwang and Richards 1992].

6.4.2. \( RNP(x) \) as a Steiner Tree Problem

Denote by \( SRNP(x) \) the single period road design problem induced by the harvest schedule \( x \). In this road network design problem, instead of a subset of required nodes, a set of links are required. Also, node \( \alpha \) must be in the solution. In contrast to solutions to STP which are always trees, the solution to this network problem may contain cycles. An obvious example is where a set of required links forms a cycle. (In figure 6.2, if a given harvest requires links 1, 2, 3 and 12, this cycle will be in the solution.) Closer investigation of \( SRNP(x) \) reveals that it is in fact equivalent to the following STP, and therefore also NP complete.
Given $RG = (V, R, W)$, the road graph, with $RL \subseteq R$, the set of required links

Define the Steiner Road Graph $SRG(SV, SL, W)$ as follows:

Set $RLV$ (Required Link Vertices) = $\{\alpha\}$

For each edge $(v, w)$ in $RL$

- Replace $(v, w)$ with a vertex $vw$. Set $RLV = RLV \cup \{vw\}$.
- If either $v$ or $w$ is $\alpha$, add a zero-length edge between $vw$ and $\alpha$.
- Re-label all edges in the cutset of $(v, w)$: $(u, v) = (u, vw)$ and $(u, w) = (u, vw)$

Set $V = V - \{(v), (w)\}$

End processing $RL$

$SV = V \cup RLV$

and $SL = R - RL$.

Then, $SRNP(\alpha)$ is equivalent to the Steiner Tree Problem on $SRG$ with Terminal nodes $RLV$. The cost of the solution is the total cost of building the required links, plus the cost of the additional links required to connect all Terminal nodes in $SRG$ to $\alpha$.

As an example, consider the network shown in Figure 6.4a. Links 4 and 9 are required. The transformed graph is shown in Figure 6.4b, where the Terminal nodes are $cd$ and $eh$. Edge 2 is now $(b, cd)$, edge 3 $(b, cd)$, edge 6 is $(f, eh)$ and so on. Note that in this case the "Terminal Link" $(c, d)$ originally formed a cycle with edges 2 and 3, with only one possibility (link 1) to complete the tree. Thus, the minimum cost Steiner tree will include the shortest of links 2 and 3. This path is shown in boldface in Figure 6.4b in the case that link 2 is of lesser cost than link 3. Figure 6.4c shows the actual road path on the original graph.

### 6.4.3. Solving Steiner Tree Problems

A number of exponential time exact algorithms have been proposed to solve STP. Aneja [1980] formulated the STP as a set covering problem in a 0-1 integer program, and uses a row generation procedure in the solution. Beasley [1989] and Balakrishnan and Patel [1987] identified reduction schemes which exploit the special structure of Steiner
trees and improve the basic integer formulation. Beasley's algorithm is a branch and bound scheme which uses Lagrangean relaxation techniques to sharpen bounds; Balakrishnan and Patel generate a series of spanning trees of increasing weight. Although these efforts have provided encouraging improvements in solution time (see Beasley 1989 where graphs of thousands of vertices are solved), there is of course still the possibility that that, in solving any instance of STP, the solution time will be exponential in the number of variables.

![Figure 6.4a](image1.png)   ![Figure 6.4b](image2.png)

Figure 6.4a  Figure 6.4b

![Figure 6.4c](image3.png)

Figure 6.4c

A class of heuristic algorithms for STP based on shortest path graphs has been shown to provide solutions close to optimal with reasonable computational times. [Wu et al 1986, Widmayer 1986 and Melhorn 1988, Takahashi and Matsuyam 1980, Kou 1990,
Kou and Maki 1987]. The most computationally efficient of these is Melhorn's algorithm [Melhorn 1988], which is of order \( (E + V \log V) \), where \( E \) is the number of edges and \( V \) the number of vertices) in \( G \). Other heuristics for Steiner tree problems and their variants include simulated annealing methods, edge insertion methods and tabu search methods. For a comprehensive discussion of the Steiner tree problem, see Hwang and Richards [1992].

6.4.4. Differences between \( SRNP(x) \) and \( RNP(x) \)

Consider \( RNP(x) \), the multi-period problem, which is NP-complete since \( SRNP(x) \) is NP-complete. An optimal solution to \( SRNP(x) \) is not necessarily optimal for \( RNP(x) \). To illustrate this, consider the problem in figure 6.5. Links \( l \) and \( RL \) are required, \( z \) is the length of the shortest path from \( a \) to \( RL \) and \( x \) is the length of the shortest path from \( \alpha \) to \( RL \). If \( x \) is greater than \( z \), then the optimal solution for the undiscounted (single - period ) problem is to build path \( P1 \) and link \( l \), with cost equal to \( (y + z) \). For the multiple-period problem, suppose that \( RL \) is required in period \( 1 \) and \( l \) is required in period \( p > 1 \). Then, if \( x + \frac{y}{(1 + d)^{p-1}} < y + z \), the minimum cost solution is to build link \( l \) and path \( P2 \). For example, if \( p = 2 \) and the discount rate is .10, the longer path \( P2 \) will be optimal whenever \( x \) is less than \( (z + .0909y) \). Note also that, as the discount rate is increased, the likelihood of finding a cheaper but longer path is increased.

Figure 6.5. Multi-Period vs Single-Period Solution
6.5. Solving $RNP(x)$

An optimal solution to $RNP(x)$ can be obtained by formulating and solving the integer program RNP1. Developing RNP1 requires a considerable effort in enumeration of connectivity constraints, and may result in a large number of indicator variables and connectivity constraints. The solution of the integer program is computationally prohibitive (Solution times for the Shulkell problem exceeded 30 minutes). Since $RNP(x)$ must be solved a great many times in the tabu search algorithm, an effective heuristic to solve $RNP(x)$ was developed.

The path heuristic ($PH$) is based on the well-known class of "label-correcting" algorithms which include Dijkstra's algorithm for shortest paths, Prim's algorithm for minimum spanning tree and others [Ahuja et al 1993]. For convenience, Prim's algorithm [Tarjan 1983, Ahuja et al 1993] for constructing the shortest path spanning tree of a graph is outlined in figure 6.6.

<table>
<thead>
<tr>
<th>Prim's Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Assign weight $W(v)=\infty$ for all vertices $v$.</td>
</tr>
<tr>
<td>2. Begin with an arbitrary vertex $k$. Set $V=V\setminus{k}$. Set $dad(k)=0$.</td>
</tr>
<tr>
<td>3. Process all nodes $t$ which are adjacent to $k$</td>
</tr>
<tr>
<td>For each $t$;</td>
</tr>
<tr>
<td>if $W(t)&gt;(W(k) + \text{length of } (t,k))$</td>
</tr>
<tr>
<td>Assign $W(t) = W(k) + \text{length of } (t,k)$</td>
</tr>
<tr>
<td>Assign $dad(t) = k$</td>
</tr>
<tr>
<td>4. If $V$ is not empty, choose the vertex $k$ in $V$ with smallest weight $W(k)$, set $V=V\setminus k$ and go to step 3.</td>
</tr>
</tbody>
</table>

Figure 6.6 Prim's Algorithm for Shortest Path Network

The weight $W(t)$ is a tentative distance label for vertex $t$. This label is updated whenever a better (i.e., shorter) path is found between $t$ and $k$. The algorithm begins with a "singleton" tree, and builds on that tree by examining adjacent nodes in a breadth first search. When complete, a directed spanning tree rooted at $k$ is created, in which each
node is contained in a shortest path to the root. The directed tree is stored using the
"dad" structure, in which each node is assigned a predecessor ("father") node in the tree.

6.5.1. Path Heuristic

If the first vertex chosen is the main road vertex $\alpha$, Prim's algorithm will produce a
shortest path spanning tree between all nodes in $RG$ and $\alpha$. The difficulty is that these
paths will not necessarily include all of (or even any of) the links which are required for
the harvest schedule $x$. This motivates changing Prim's algorithm in the following way:
attempt to include the required links by reducing the weighting of paths which contain
any element of $RL$. The Path Heuristic ($PH$) is as follows:

Path Heuristic
1. Assign weight $W(v)=\infty$ for all vertices $v$ in $V$.
2. Begin with $\alpha$. Set $V=V\setminus\alpha$. Label $k=\alpha$
3. Process all nodes $t$ in $V$ that are adjacent to $k$
   For each $t$;
   If $(k,t)$ is a required link,
   assign $W(t)=0$
   Otherwise,
   if $W(t)>(W(k)+\text{length of } (t,k))$
   assign $W(t)=W(k)+\text{length of } (t,k)$
   assign dad($t$)=$k$
   End processing edges $(k,t)$
4. If $V$ is not empty, choose the vertex $k$ in $V$ with smallest weight $W(k)$, set $V=V\setminus k$, and
   go to step 3.
5. If any required links is not a member of some path, it must form a cycle with two
   paths. Assign the link to the shortest of these.
6. For each path, assign the earliest required period to each link, and calculate the cost
   of the road project.

Figure 6.7. Path Heuristic

Clearly, the search begins with node $\alpha$, since the object is to build paths to the main
road access. The difference between $PH$ and Prim's algorithm is that, when a required
link adjacent to $t$ is encountered in step 2, a weight of 0 (highest priority) is assigned to $t$.
This forces required links into the solution. After completion of steps 1-3, a tree has
been created which connects each node in RG to α. If any required links are missing from this tree, they must then form a cycle with two paths in the tree. Step 4 then adds these links to the shortest of the paths in which the cycle occurs. Finally, step 5 assigns the earliest required harvest period to each link in each path.

One might observe that this heuristic in fact calculates more paths than are necessary to solve $RNP(\alpha)$. This is true, since it is only strictly necessary to calculate paths until all required links have been included. The extra iterations of Step 3 could be removed by stopping the algorithm when $V$ contains no more vertices incident to required links. The complete set of paths produced by $PH$ will be used to quickly calculate the (approximate) incremental roading costs for moves in the tabu search algorithm. So, the algorithm will be used in this form.

6.5.2. Computational Complexity

The running time of Prim's algorithm is $O(|V||R|)$. When implemented with a priority queue effected by a binary heap data structure, this complexity is reduced to $O(|V| + |R| \log |V|)$ [Ahuja et al 1993, Tarjan 1983, Sedgewick 1983]. In the Path Heuristic, some extra computational cost is incurred by adding any missing required links (Step 4) and re-timing the links in each path (Step 5). Step 4 computation effort is $O(|RL|)$, typically $<< O(|R|)$, since only those required links which form a cycle in the graph will be left missing from the path network. Step 5 computations require at most $O(|R|)$ comparisons to set the timing of the links. This is easily seen by examining the search method used to assign the timing of the links. (Figure 6.8). The re-timing algorithm makes use of two variables: $period(r)$ and $c_period(r)$. $Period(r)$ is the period $jmin_\alpha(x)$ determined by the stand assignments and $c_period(r)$ is the period assigned for construction of link $r$. 
Set \( period(r) = \min_{x} \), \( c\_period(r) = (J+1) \) for \( r = 1..R \).

For each link \( r \):

If \( r \) is a leaf (i.e. it has no successors):

Set \( c\_period(r) = period(r) \)

do while \( \{(dad(r) <= 0) \, \text{and} \, c\_period(dad(r)) > c\_period(r)\} \)

\( c\_period(dad(r)) = c\_period(r) \)

\( r = dad(r) \)

done

End if

End

Figure 6.8. Link Re-Timing Procedure

The re-timing loop is executed once for each \( r \) which is a leaf in the shortest path spanning tree. The first time a path is encountered in this procedure, it is processed entirely (since \( c\_period \) is set to \( (J+1) \) initially). If the leaves are processed in an unordered way, path segments would be re-processed each time a new leaf incident to that path is encountered. This inefficiency is avoided by first sorting the leaves in increasing order of period. The timing of all road links requires \( 2R \) steps to set initial values, \( L \) steps to begin processing each path (\( L \) is the number of leaves), \( R \) steps to make the assignments and \( O(L \log L) \) steps to sort the leaves (for example, using a binary heap sorting algorithm [Sedgewick 1983]). Since the number of leaves is dominated by the number of vertices, the running time of \( PH \) is \( O(|V| + R(1 + \log |V|)) \).

6.5.3. Numerical Results

A sample of 300 harvest solutions \( x \) was generated to evaluate the performance of the Path Heuristic on the Shulkeil study area. These solutions were generated by randomly assigning each stand a harvest period in \([1,20]\). The first four periods were used for the tactical planning horizon, with the remaining stands categorized as unassigned. Optimal solutions to \( SRNP(x) \) were obtained by implementing the integer formulation \( RNP1 \) in LINDO\textsuperscript{TM} [Schrage 1988]. 41.41\% of the \( PH \) solutions were
optimal, the worst case deviation from optimal was 2.81% and the average deviation was 0.43%. (Figure 6.9)

Figure 6.10 compares $PH$ results to optimal solutions for samples (size 100) which vary the number of required road links. For smaller numbers of required links, more heuristic solutions are optimal, but the maximum percentage deviation from optimal on the sample tends to be higher. Nevertheless, the $PH$ performance is again very good, with average deviation from optimal of less than 1.5% in all cases. These experiments show that the path heuristic performed very well on the single-period problem.

![Graph](image)

Figure 6.9. PH Performance for Single-Period Problem, Randomly Generated Stands

<table>
<thead>
<tr>
<th># Required Links</th>
<th>Maximum Deviation</th>
<th>Average Deviation</th>
<th>% Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11.94%</td>
<td>1.26%</td>
<td>79%</td>
</tr>
<tr>
<td>20</td>
<td>14.69%</td>
<td>0.99%</td>
<td>82%</td>
</tr>
<tr>
<td>30</td>
<td>9.64%</td>
<td>1.40%</td>
<td>56%</td>
</tr>
<tr>
<td>40</td>
<td>6.66%</td>
<td>1.02%</td>
<td>54%</td>
</tr>
<tr>
<td>50</td>
<td>5.19%</td>
<td>0.63%</td>
<td>67%</td>
</tr>
</tbody>
</table>

Figure 6.10. SRNP(x) Deviation from Optimal by Number of Required Links
Sixty-four instances of the multiperiod problem (with discount rate 10%) were solved to optimality using LINDO™. The left half column in Figure 6.11 compares path heuristic results with the optimal solution. In seven of the sixty-four runs, the Path Heuristic found an optimal solution. The worst case deviation from optimal was 10.48%, with average deviation of 1.95%. Thus, the heuristic performed very well for the multi-period problem, at this discount rate. The right hand side of Figure 6.11 documents the differences between the discounted multi-period solution and its optimal single-period solution. In twenty-five percent of the cases (16), the optimal single-period solution and the optimal multi-period solution were identical. There was a maximum observed increase in the total length of road links of 3.21%, and an average difference of 0.61%.

<table>
<thead>
<tr>
<th>Percentage Deviation from Optimal</th>
<th>Length Increase over Single Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.95%</td>
</tr>
<tr>
<td>Minimum</td>
<td>0.00%</td>
</tr>
<tr>
<td>Maximum</td>
<td>10.48%</td>
</tr>
<tr>
<td>Sample Size</td>
<td>64</td>
</tr>
<tr>
<td># OPTIMAL Solutions</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 6.11. Path Heuristic Multi-Period Results
CHAPTER 7
TABU SEARCH ALGORITHMS FOR HSRBP

In chapter 4, the tabu search method and some of its important variants was described. It was shown that there is a wide range of sophistication in TS algorithms, and that researchers have spent considerable effort in choosing appropriate TS structures that work well on given problems. The problem here is to choose a solution methodology for the harvest scheduling and road building problem HSRBP.

First, a heuristic search method is evidently required. HSRBP has a constraint structure that is complex, its decision variables are integer and practical problem instances have a large number of variables. Moreover, exact formulations of the adjacency constraints are not possible. The neighbourhood structure for this problem can be designed so that moves are easily evaluated; thus a search algorithm is appropriate. Amongst the well known metaheuristic search strategies, TS was chosen because of its considerable record of success in solving hard optimization problems.

A number of TS methods were tested, which varied in complexity from the simple fixed tabu search to algorithms with complicated dynamic search strategies. Given the difficult nature of the problem, it is not surprising that straightforward implementations of fixed tabu search were found to be ineffective in consistently obtaining good solutions to HSRBP. The best results were obtained from using an algorithm which combines reactive tabu search self-tuning principles, a strategic oscillation through constraint barriers effected by means of a modified objective function, and stochastic diversification moves.

This chapter describes the details of the TS algorithms that were designed to solve
HSRBP. The following chapter documents the empirical studies which were done to compare the algorithm variants.

7.1. Search Algorithm Structure

Figure 7.1 is a top-level flowchart of the search algorithm structure. Each TS algorithm implements the same neighbourhood search and move. The main differences lie in the methods employed to determine tabu tenure, the inclusion/exclusion and choice of stochastic diversification strategies and the inclusion or exclusion of infeasible moves. Inputs to the model are the discount rate, harvest targets, allowed deviations, the maximum opening size and adjacency delay time, the planning periods to be considered and the stand and road data. The road data consists of an adjacency list (nodes) and a cost for each link in the proposed road network. Stand characteristics are the cover type, site capability, current age, stocking, a list of adjacent stands, and the road link required to access the stand.

The first phase is pre-processing of the input data. Mean annual increments for the planning periods are determined (using the Nova Scotia Growth and Yield model) and projected harvest volumes for the planning periods are calculated. For each stand, the lost volume penalties accruing from each possible harvest assignment are calculated and stored in the stand data structure. A linked list of stands which require each road link is calculated and stored. Finally, the openings (graphs) of period 1 are constructed for all stands which have been clearcut in the period immediately preceding the first planning period.
The second stage of the algorithm is to generate a starting solution. This is done using a Monte Carlo procedure. Stands which border on a main road are randomly selected and assigned to each period in turn until the periodic harvest target is achieved. For algorithms which do not allow infeasible solutions (FTS, RTS), assignments which violate opening size constraints are rejected, and the solution is also required to be feasible with respect to total harvest target. The result of the process is an initial solution which requires no roading. If it is not possible to meet total harvest without building roads, the initial solution either may be made feasible by constructing some small set of roads, or an infeasible starting solution may be accepted. The algorithm A road network is then constructed, using the Path Heuristic. This initial network is in fact a shortest path network on the nodes of the road graph. At this stage, the solution incurs a large
lost volume penalty and no roading costs. See Appendix A, Figure A.1.

The third stage is the tabu search algorithm. Moves, which perturb the current solution, change the harvest assignment of one stand. The evaluation function for neighbourhood search is the sum of lost volume and roading costs. This function is augmented with terms which penalize infeasibilities with respect to each constraint set for the oscillating search algorithms (OTS, ORTS). In all cases, the full neighbourhood of the current solution is searched, and the best move is chosen.

This best move is implemented by making the stand assignment, updating the opening graph structure and updating the roading network if necessary. If the move improves on the best solution to date, the solution (i.e. all stand assignments and the roading network assignments) is saved. The tabu status of the stand chosen is set by storing the current move count in the stand data.

Once the move has been implemented, search status counters are updated. These include counters for the number of moves taken (move_count) and the number of moves since improvement (moves_since_improve) in the objective function. A diversification strategy may be invoked if moves_since_improve exceeds a pre-set limit. In RTS and ORTS, data structures which store the hashed value of the current solution are updated, and search status counters which measure cycling to previous solutions are updated. Based on these counters, tabu tenure may be increased or decreased, and a chaotic cycling indicator may be set to true, which then will cause a diversification routine to be executed.

Finally, the stopping rule is tested. The algorithm is terminated when the number of moves taken is greater than a specified maximum (maxiters) and no feasible improving move has been found for a specified number of iterations (max_non_improving). This second condition ensures that the neighbourhood of the last
improving move is thoroughly investigated.

7.2. Moves and Neighbourhood Search

Recall that a harvest solution \( x \) is an \( S \)-vector whose elements are the harvest period assigned to each stand, where harvest periods are integers in the range \([0, J+1]\). Assignment of \((J+1)\) indicates that the stand is not scheduled to be harvested, and stands which have been clearcut previously are assigned period 0. Neighbourhood search details are shown in figure A.2.

7.2.1. Move Types

Given a solution \( x \), the moves which lead to a new solution are a re-assignment of any component of \( x \). That is, moves change the harvest assignment of one stand. These moves can be classified as add, swap and delete moves, in that the stand is added to the harvest schedule, has its harvest period swapped from one period to another in \([1,J]\), or is removed from the harvest schedule. Note that existing clearcuts are of course not candidates for any move.

\(\text{Definition 7.1} \) The moves \( m=(s,k) \) which define the neighborhood \( N(x) \) of a solution are \( x_i : j \to k, k \in [1, J+1], k \neq j, s \in [1, S], x_i \neq 0. \)

<table>
<thead>
<tr>
<th>Move Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>( j = J+1 )</td>
</tr>
<tr>
<td>Delete</td>
<td>( k = J+1 )</td>
</tr>
<tr>
<td>Swap</td>
<td>( j, k \leq J )</td>
</tr>
</tbody>
</table>

Figure 7.2. Move Types

All algorithms employ a full neighbourhood search. For larger datasets (more stands), implementation of a candidate list strategy (see section 4.2) would be appropriate.
7.2.2. **Tabu Status and Aspiration Criteria**

Once a move $x_{j}: j \rightarrow k$ is taken, the stand $s$ is made *tabu* for *tabu_tenure* moves. This is effected by storing the current move counter in the stand's data. Checking tabu status is done by comparing the current move counter to this value and *tabu_tenure*. The duration of *tabu_tenure* is fixed throughout the algorithm in the FTS and OTS methods, and is dynamically varying in RTS and ORTS. Note that the tabu status forbids all non-aspirated moves involving stand $s$. It is a strong condition that may exclude a total of $J$ moves.

Only one aspiration criterion, *aspiration by objective*, was used. The tabu status of a move is disregarded if executing the move would result in a feasible solution which is better than any solution encountered thus far in the search.

7.2.3. **Calculating Change in Objective Function**

In neighbourhood search, each potential move is evaluated by calculating the change in objective function which would result from implementing the move. The evaluation function for these algorithms is

$$z'(x) = LV + \rho \ RNP(x) + \alpha \ Dev_{total} + \beta \ Dev_{per} + \gamma \ Dev_{maxopen}$$

This function consists of two cost factors, $LV$ and $RC$, and three penalty terms $Dev_{total}$, $Dev_{per}$ and $Dev_{maxopen}$, which measure the amount of deviation of the solution from the feasible region for each corresponding constraint boundary. (Parameters $\alpha$, $\beta$ and $\gamma$ are used in the oscillating search method to guide the solution into and out of feasible regions, while $\rho$ is used to examine the tradeoff between road cost and lost volume.) Using indicator variables $x_{sj}$ and $z_{rj}$ which are 1 (true) if stand $s$ is to be harvested (respectively, link $r$ to be built) in period $j$, each term is defined as follows:
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$LV(x)$</td>
<td>Lost Volume</td>
<td>$\sum_{j=1}^{J} \sum_{s=1}^{S} LV(s, j) x_{sj}$</td>
</tr>
<tr>
<td>$RC(x)$</td>
<td>Road Cost</td>
<td>$\sum_{j=1}^{J} \sum_{r=1}^{R} w_{jr} z_{jr}$</td>
</tr>
<tr>
<td>$Dev_{total}(x)$</td>
<td>Deviation from total harvest target</td>
<td>$\left[ \sum_{j=1}^{J} \sum_{s=1}^{S} \left[ V_{sj} x_{sj} - TV \right] - dTV \times TV \right]^+$</td>
</tr>
<tr>
<td>$Dev_{periodic}(x)$</td>
<td>Sum of deviations from periodic harvest targets</td>
<td>$\sum_{j=1}^{J} \left[ \sum_{s=1}^{S} V_{sj} x_{sj} - PV \right] - dPV \times PV_j \right]^+$</td>
</tr>
<tr>
<td>$Dev_{maxopen}(x)$</td>
<td>Sum of opening size deviations from maxopen.</td>
<td>$\sum_{p=1}^{P} \sum_{t \in O_p} \left[ \sum_{s=1}^{S} a_s \right] - \text{maxopen} \right]^+$</td>
</tr>
</tbody>
</table>

Figure 7.3. Definition of Objective Function Terms

In full neighbourhood search, all possible moves $m$ are evaluated by calculating $z(x \oplus m)$. In the following sub-sections, the calculations required for each term in $z'$ are detailed.

7.2.3.1. **Loss Penalty**

The incremental change in lost volume from a move $(s,k)$ is the lost volume induced by the proposed assignment less the lost volume from the current stand assignment. Thus, the new loss penalty is easily computed

$$LV(x \oplus m) = LV(x) + LV(s, k) - LV(s, j)$$

7.2.3.2. **Roading Cost**

Recall that, in the generation of the initial solution, the roading network is calculated by the Path Heuristic, resulting in a directed shortest path network on the nodes of the road network, with each link in the network assigned a period $r_t \in [1..J+1]$. Initially, the network consists of a minimum cost directed spanning tree with all links assigned to $(J+1)$. As stands are added to the harvest solution, the roading network is
made feasible by re-timing (i.e. assigning an earlier construction period to) each link that is required to access the stands (See Figure 7.4). If the stand to be added requires a link which is not in the current path network, the incremental cost to add this link is estimated by calculating the cost to insert that link into the shortest adjacent path.

![Diagram showing re-timing of links in a path](image)

**Figure 7.4. Re-Timing Links in a Path**

Each stand $s$ is either assigned a link ($link(s)$) in the road network (if road building is required to access that stand), or is assigned a "dummy" link 0 if the stand is accessible by main road. For each road link $l$, the set $S_l$ is the set of stands requiring link $l$. Figure 7.5 shows the calculations which determine the incremental roading cost for a move. Note that in the case that a link is re-assigned to a later period, the incremental costs will be negative, thus decreasing the total roading cost $RC(x)$. 
Calculation:
1. If \( \text{link}(s) = 0 \),
   road cost change is 0. END
   Otherwise, set \( l = \text{link}(s) \)
2. If \( l \) is not included in a path, add \( l \) to the cheapest of the two possible paths.
3. Set \( x_r = k \)
   Calculate \( \text{smin} = \min_{m \in \mathcal{S}_r} (x_m) \), the earliest period required by all stands, and
   \[ \text{rmin} = \min_{m \in \mathcal{S}_\text{prev}(l)} (r_m) \], the earliest period required by preceding links. Then
   \[ p = \min(\text{smin}, \text{rmin}) \] is the period required for this link.
   If \( r_i = p \), road cost change is 0.0 END
4. Otherwise, calculate the re-timing cost
5. Cost = \( w_p - w_{l,r} \) !cost of re-timing the link
6. Do until \( l = 0 \)
   \[ l = \text{father}(l) \] !next link in path
   \[ p = \min(r_i, p) \] !new assignment
   Cost(cost + \( w_p - w_{l,r} \)) !cost to re-time
   End do
7. Re-set \( x_r = j \)

END of Calculation

Figure 7.5. Calculating Incremental Road Cost

7.2.3.3. Deviation from Maximum Opening Size

For simplicity in describing these calculations, assume an adjacency delay time of two planning periods. Thus, each stand may belong to at most two openings. Further notation for these calculations is as follows:

- \( a_s \) size in acres of stand \( s \)
- \( \text{size}_k \) Size of opening \( k \)
- \( \text{open}_\text{dev}_k \) \( ([\text{size}_k - \text{maxopen}]^* \)
- \( \text{move}_\text{dev}_\text{open} \) change in \( \text{Dev}_\text{maxopen} \) if the move is implemented

7.2.3.4. Add Move.

In this case the stand may add to the acreage of existing openings if there are any adjacent to the stand. More specifically, the stand adds to any adjacent openings which are of opening period \( k \) or \( (k+1) \), since \( ADT \) is two periods. If the stand is adjacent to
more than one opening, the move results in a merging of those openings. Thus, to calculate \( \text{Dev\_maxopen} \), the adjacency list of stand \( s \) is searched for adjacent openings. If there are no adjacent openings of period \( k \) or \((k+1)\), then the move incurs no additional \( \text{maxopen} \) penalty. Otherwise, the size of the merged openings is calculated, compared with \( \text{maxopen} \), and any additional penalty incurred by the move is calculated. Figure 7.6 shows these calculations.

Let \( \Omega = \{o_1^p, o_2^p, ..., o_n^p\}, p = k, k+1 \) be the set of \( n \) adjacent openings for each opening period. If \( \Omega \) is empty, the penalty is 0.0. Otherwise,

1. Set \( \text{move\_dev\_open} = 0.0 \)
2. For each opening \( O_j^p \) in \( \Omega \), subtract the existing deviation, if any.
   \( \text{move\_dev\_open} = \text{move\_dev\_open} - \text{open\_dev}_{o_j} \)
3. For each opening period \( p \),
   \( O_{\text{size}}_p = a_s + \sum_{j=1}^{n} O_{\text{size}}_{o_j} \) \quad \text{size of merged open for period } p
   \( \text{Move\_dev\_open} = \text{Move\_dev\_open} + \left[ O_{\text{size}}_p - \text{maxopen} \right] \)

Figure 7.6. Calculating Maxopen Penalty from Add Move

![Figure 7.7. Merging Adjacent Openings](image)

7.2.3.5. **Delete Move.**

If the openings in which the stand is a member are smaller than \( \text{maxopen} \), deleting the stand induces no change in \( \text{Dev\_maxopen} \). Otherwise, the delete move will either reduce the current deviation penalty accruing from these openings, or remove it entirely.

To determine the size of affected openings, we observe that removing stand \( s \) will
disconnect the opening graph if \( s \) is an articulation point of the graph [Sedgewick 1983].

Thus, the effect of deleting stand \( s \) is calculated as shown in figure 7.8.

```
Set \( \text{move\_dev\_open} = 0.0 \)
For each opening \( o \) in which \( s \) is a member:
1. If \( \text{Osize}_o - a_s \leq \text{maxopen} \), then
   \[ \text{move\_dev\_open} = \text{move\_dev\_open} - \text{Open\_dev}_o \]
2. Otherwise, if \( s \) is an articulation point of the opening, (Figure 7.8)
   Using depth first search, find the set \( V \) of connected subgraphs of
   \( O'_o = O_o \setminus s \). For each \( v \) in \( V \), calculate the deviation penalty. Then,
   \[ \text{move\_dev\_open} = \sum_{v \in V} \text{Open\_dev}_v - \text{Open\_dev}_o \]
3. If \( s \) is not an articulation point of the opening \( o \), (Figure 7.6)
   \[ \text{move\_dev\_open} = \left( \text{Osize}_o - a_s - \text{maxopen} \right) - \text{Open\_dev}_o \]
```

Figure 7.8. Calculating Maxopen Penalty from Delete Move

Figure 7.9. Splitting an Opening

Figure 7.10. Reducing an Opening
7.2.3.6. **Swap Move.**

In this case, the total change in deviation from maxopen is the change incurred by deleting the stand from period \( j \) and that of adding it to period \( k \). In the case where the move is to shift the stand assignment by one period, some efficiency can be gained by not calculating openings that do not change. For example, if a stand is to be moved from period 1 to period 2, it will be removed from openings of period 1 and added to openings of period 3.

7.2.3.7. **Harvest Target Deviations**

Deviations from periodic and total harvest targets are determined from the current harvest and the volume which would be added or deleted to or from each period by the move. Denote by \( H(j) \) the total volume harvested in period \( j \), and by \( TH \) the total harvest for the current solution. Then, for the move which changes the assignment of stand \( s \) from period \( j \) to period \( k \),

\[
H(j) = H(j) - V_{sj} \quad j \in [1, J] \\
H(k) = H(k) + V_{sk} \quad k \in [1, J] \\
TH = TH - V_{sj} + V_{sk} \quad j, k \in [1, J]
\]

and the resulting deviations are

\[
\text{dev\_total} = \left[ |TH - TV| - dTV \times TV \right]^t
\]

\[
\text{dev\_periodic} = \sum_{j=1}^{J} \left[ |H(j) - PV_j| - dPV \times PV(j) \right]^t
\]

7.3. **Special Diversifying Moves**

In the RTS and ORTS algorithms, a set of special diversifying moves (called ESCAPES) is invoked when the search history indicates that the trajectory is cycling "chaotically " [Battiti and Tecchiolli 1995]. These "escapes" are also used to diversify the search when no improvement has been observed for a sufficiently long time.
These diversification schemes serve two purposes. First, they increase the flexibility of the algorithm in "escaping" from a local optimum or in "jumping" to a new area of the solution space. Secondly, the stochastic nature of the moves increases the robustness of the algorithm. This effect was indicated in empirical tests by a reduced variance in samples from algorithms which included ESCAPES. (See Chapter 8).

Two types of diversification moves were tested. The first, ESCAPE1, selects a random sample of stands and either deletes them from or adds them to the harvest, according to whether or not a stand is currently in or not in the harvest solution (figure A.3). As in the R-Tabu [Battiti and Tecchiolli 1995] algorithm, the size of the sample is chosen to be proportional to the moving average of detected cycle length: (For OTS and FTS, a constant number of random moves, proportional to the number of stands, was chosen.)

\[
\{ \text{sample}_\text{-size} = \begin{vmatrix} r \times \text{moving}_\text{-average} \end{vmatrix}, r \in U[0,1] \}\).
\]

The second strategy, ESCAPE2, is to randomly select a road link in the solution and remove it, and every preceding link in its path, from the solution. This is done by executing delete moves for each stand that are in the current solution and which require any element of the path. The road network is then re-calculated using PH. In the example in Figure 7.11, if link \(i\) is selected, stands which require links \(i,j,k,l\) and \(m\) must all be deleted.

![Figure 7.11. ESCAPE2 Link Selection](image-url)
The motivation for this type of escape routine comes from considering the relationship between stands and road links. One road link may serve many stands, but one stand requires at most one link. Thus, evaluation of single stand moves is not effective in directing the search towards solutions that have different roading structure. Add moves favour adding road links when the loss penalty would be reduced by more than the road cost incurred. Once a link is in the solution, subsequent add moves incur no further roading costs. Thus, it is unlikely that a sequence of moves that would cause a link to be deleted (assigned to period \(J+I\)) or re-timed (moved from current assignment to a later one) will be chosen. This is because there is no direct mechanism in the stand-based move evaluation to predict the objective function change would occur by removing or re-timing road links. For this reason, the diversifying escape moves are based on choosing a link and removing it, and all stands requiring it, from the solution. As in \(ESCAPE1\), the tabu status of each stand is set after it is deleted. \(ESCAPE2\) is outlined in figure A.4.

This type of diversification was not used in algorithms \(FTS\) and \(OTS\) since it is difficult to choose link deletion moves that maintain feasibility. For \(ORTS\) and \(RTS\), the \(ESCAPE2\) strategy worked less well than expected. Further investigation of the algorithm performance led to changing the method. Instead of selecting any link in the road network, the procedure was changed to select the link that is a leaf of the road network graph, and which contributes the greatest reduction in roading costs if it is re-timed. This leaf link was then re-timed one period forward from \(k\) to period \((k+1)\). That is, for each stand associated with the link, its harvest assignment was set to the larger of its current assignment and \((k+1)\). This diversification produced better results on the datasets which were tested.

The final diversification strategy, \(ESCAPE3\), is a combination of \(ESCAPE1\) and
ESCAPE2. If the number of links in the current road network is less than 10% of total, the stand escape routine ESCAPE1 is chosen, otherwise ESCAPE2 is chosen. See figure A.5.

7.4. Fixed Tabu Search Algorithm (FTS)

As described in Chapter 3, the fixed tabu search method has a fixed tabu tenure tabu_tenure, and moves are restricted to those which produce feasible outcomes. The objective function is the sum of the lost volume penalty and the roading cost, \( z = LV + RC \). This method was tested for various settings of tabu_tenure. The trend was that larger values of tabu_tenure gave better results, but of course as tabu_tenure grows, the system eventually runs out of feasible moves. Although FTS was not expected to perform well here due to the complex nature of the maxopen constraints, and the restrictive nature of the harvest target constraints, experimentation with this relatively straightforward TS method was included to provide a baseline against which to compare the more sophisticated algorithms [Barr et al 1995].

7.5. Oscillating Tabu Search Algorithm (OTS)

This is an implementation of the principle of strategic oscillation [Glover 1990a, Kelly, Golden and Assad 1992]. The method of dynamic penalty parameters, as reported by Gendreau et al [1991] for the vehicle routing problem, was used to control the search trajectory.

Recall that the objective function with penalty terms is

\[
z'(x) = LV + \rho \cdot RC(x) + \alpha \cdot Dev\_total + \beta \cdot Dev\_per + \gamma \cdot Dev\_maxopen
\]

The best move chosen in neighbourhood search is that with the lowest value of \( z' \). For feasible solutions \( xfeas \), \( z'(xfeas) \equiv z(xfeas) \), while for infeasible solutions, one or more of the penalty terms is positive. Initially, \( \alpha, \beta \) and \( \gamma \) are set to a large value,
upper_bound. After each move \( m \), a counter is incremented for each penalty term that is zero in \( z'(x \oplus m) \). After check_interval steps have been taken, each penalty parameter is increased (doubled) if its counter is equal check_interval, or decreased (halved) if its counter is zero. This has the effect of increasing the weighting on constraints which have not been satisfied for the last check_interval iterations and decreasing the weighting on constraints which have been violated for all steps. Thus, the penalty parameters pull the solution in and out of the feasible region systematically.

The interaction between the special diversification moves and the best setting for the penalty parameters was studied, and it was determined empirically that best results were achieved when \( \alpha, \beta, \) and \( \gamma \) were set to 1.0 after a diversification required by chaotic cycling, and set to maximum value after a diversification implemented due to no improvement. This is an experimental result only; however it is logical. Setting penalty parameters to 1.0 gently pulls the solution back to the feasible region. Setting parameters to upper_bound radically (more quickly) yanks the solution back to the feasible region.

7.6. Reactive Tabu Search Algorithms (RTS)

The flowchart in figure 7.12 outlines the methodology used to track cycling to previously found solutions, and to use this search history information to dynamically set tabu_tenure. This procedure is the same basic method as in Battiti and Tecchiolli [1995], except that it was extended to use the solution value \( z = LV + RC \) to detect collisions, and use linear probing to resolve the hashing address in the case of collisions.

7.6.1. RTS Data Structures and Cycle Detection

The RTS structures are a hashing table of fixed size table_size, a configuration table with entries \( z = LV + RC \); the solution value, num_reps; the number of times that solution has been repeated and time_last_found; the time of the last repetition. At each
iteration of the algorithm, a hash address for the solution $x$ is calculated. If this address has been used previously, the current solution $z(x)$ is compared with the solution value stored for that hash address. If they are equal, then this is assumed to be a repeated solution, and the cycle length is calculated. Cycle length is the number of steps in the solution trajectory since the solution was last encountered. If this cycle length is less than the parameter $cycle\_max$, the moving average of cycle length is updated, and tabu tenure is increased by a factor of 1.1. The effect of the exponential increase in $tabu\_tenure$ is to "break" the cycling to previous solutions, thus promoting search diversification.

When a new solution is found, its hashing table entry is updated (with a pointer) and the solution value and $move\_count$ stored. If the number of iterations since $tabu\_tenure$ has been changed (counted in variable $steps\_since\_size\_change$) is greater than the moving average of cycle length, the tabu tenure is decreased by a constant factor. Thus, the search trajectory is less constrained when a suitable number of steps have been taken without returning to a previous solution, and thus may be intensified in the new search region.

If the neighbourhood search finds no non-tabu moves, tabu tenure is decreased. Thus, the RTS algorithms do not terminate on finding that all moves are tabu.
7.6.2. Hashing Address Computation

The hashing function is a mapping from a solution \( x \) to an integer in the range \([0, \text{table_size}]\). It is determined by the following computation:
Denote by $ls(l)$ a left cyclic bit shift of the integer $l$

$$H_1 = x_1$$
$$H_n = ls(H_{n-1}) \times x_n; \quad n = 2, S$$
$$H(x) = H_S \mod \text{table\_size}$$

That is, the hash function iteratively computes the product modulo $w$ of each $x_i$, first left shifting the previous product. ($w$ is the word size of the integer $H$) Then, the hash address is the result of these products taken modulo $\text{table\_size}$. This is a recommended method for dealing with hashing of vectors with multiple identical entries [Knuth 1973]. In this problem, the solution vector $x$ will have entries in the range $[0, J+1]$, thus only $J+2$ possibilities. There are likely to be a preponderance of entries equal to $(J+1)$. Thus, the hashing table size was chosen so that it is a prime number which does not divide $(J+2)^k \pm a$ for small values of $k$ and $a$. That is, $\text{table\_size}$ is chosen to be prime so that $(J+2)^k$ is not congruent to $a$ modulo $\text{table\_size}$. For the Shulkell study area, $\text{table\_size}$ was chosen to be 50027. A small study was done to estimate the number of collisions (cases where the hashed address for a new solution is equal to that of a previously encountered solution) that would be expected with this size of table. 5000 randomly generated solutions yielded 234 collisions. Although this number is small (approx. 4.68%), it was decided to include collision resolution by linear probing. The key used for collision resolution is the solution value. Thus, if $H(x)$ is already used, and its associated $z$ does not match that of the solution $x$, the table $H(x)$ is linearly searched for unused slots. The expected cost of collision resolution is small (See Knuth 1973 for a comprehensive description of hashing.).

7.7. Oscillating Reactive Tabu Search Algorithm (ORTS)

This algorithm combines the feedback parameter tuning of RTS and strategic oscillation through boundaries of the feasible region of OTS. The ORTS method
produced solutions that were better or comparable to OTS, but with a reduced variance in solution quality in samples obtained from a set of random initial solutions. Moreover, no experimentation is required to determine an appropriate duration for tabu tenure. Thus, this is the recommended algorithm and the one which was used to produce tradeoff curves for the datasets. The ORTS algorithm in its final form is outlined in figure 7.13.

This algorithm uses the \texttt{ESCAPE3} diversification strategy in two places. The first when either the \texttt{chaotic} variable indicates that the solution trajectory has returned too many times to the same solutions. When the diversification moves have been taken, the penalty parameters are each set to 1.0. This causes the search trajectory to be drawn smoothly to the feasible region. The second instance where diversification is invoked is when no improvement in the solution has been found for 300 moves. After effecting the diversification, penalty parameters are set to their maximum value \texttt{upper\_bound}. 
1. Preprocessing and Initial Solution
2. Initialize Data Structures
3. Repeat RTS for check_interval iterations

   RTS:
   3.1 Implement move and update best solution found.
   3.2 Compute Hashing address $H(x)$ for current solution.
   
   3.2.1 If $H(x)$ is used, then this is a repeated solution.
   Update number of repetitions for this move, and calculate
   the cycle length.
   If the cycle length is less than cycle max:
   Calculate the moving average of cycle length.
   Increase tabu tenure.
   If the number of repetitions for this solution has reached
   3, add one to the chaotic counter.
   If the chaotic counter has reached 3, return escape =
   .true.
   3.2.2 Otherwise, this is a new solution:
   Store new solution $H(x)$ and $z$ in hash_table.
   If the number of moves since tabu tenure has been
   changed is greater than moving_average: decrease
   tabu_tenure
   Return escape=false.

3.3 If escape is true:
   Call Diversification Routine
   Set penalty parameters to 1.0
3.4 Neighbourhood Search.
3.5 If no move is found:
   Decrease tabu tenure
   Goto Neighbourhood Search

END RTS
4. Update penalty coefficients.
5. If diversification is required
   Call diversification routine
   Set penalty parameters to upper_bound.
6. If Stopping rule is true
   Report solution and Stop
7. Go to Step 3.

Figure 7.13. ORTS Algorithm
CHAPTER 8

EMPIRICAL RESULTS: SHULKELL STUDY REGION

This chapter documents numerical results of several experiments which were carried out to compare the tabu search algorithms that were described in Chapter 7. The data for these tests is based on the GIS coverage of a region of Cumberland County in Nova Scotia, Canada. This data was then manipulated to produce five additional datasets. Baselines were produced using FTS, OTS and RTS algorithms. The ORTS method, which combines the dynamic penalty parameter evaluation function of OTS and the reactive search methods of RTS, was then compared to the baseline results. With ORTS, the best form of the starting solution and the best diversification strategy was determined. This algorithm was then used to produce tradeoff curves for all datasets.

This chapter is organized as follows. First, the study region characteristics and datasets used to validate the algorithms are described. Then, a comparison of different algorithms based on solution quality and range of solutions is given. The ORTS algorithm variants are described and the results of comparative analysis presented. Tradeoff curves for each dataset are then shown, and discussed.

8.1. Study Region and Model

A section of the Chignecto Management Unit, in Cumberland County, Nova Scotia, consisting of crown lands on map sheets D21H10T2 and D21H10T4, was chosen to test the algorithms. (Figure 8.1) The Shulie and Kelly rivers are the major waterways in the area, and hence the name SHULKELL was coined for this portion of the
management unit. There are several main roads (the Aub Brown, Bucktogen, Goodwin, Tipping and Meadow Brook roads) which crisscross an area of 5239 hectares. 4493 hectares are in 1039 forested stands, the remainder being barren, swamp or bog (Table 8.1).

![Figure 8.1 Image of the Shulkell Study Area](image)

**8.1.1. Characteristics of the Forested Stands**

Nearly half of the SHULKELL area is in stands of age 41-60 years, accounting for 65% of the merchantable volume (Table 8.2). Stands of age 61-80 years represent 9.6% of the area and contribute 23.5% of the merchantable volume, while only three stands are of age greater than 80 years.

61% of the area is in softwood stands, 36.5% in hardwood stands and 2.5% in
mixedwood stands (Table 8.3). The majority of the area (59.4%) is comprised of natural, untreated and unmanaged stands. 4% of the area has been recently clearcut and 4% is in managed plantations. The remaining 32% is classified as "dead", meaning that evidence of dead material (standing or fallen) was found. Sub-classifications dead-1 and dead-2 further categorize dead stands as to the amount of live residual material on the stand (less than 25%, 26-50% and 51-75% of crown closure respectively) [Province of Nova Scotia 1992]. These dead stands are the shaded areas in figure 8.1.

<table>
<thead>
<tr>
<th>Land Type</th>
<th>Non-Forested Hectares</th>
<th>Forested Hectares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bog</td>
<td>341.38</td>
<td>2,669.52</td>
</tr>
<tr>
<td>Brush</td>
<td>111.81</td>
<td>1,466.46</td>
</tr>
<tr>
<td>Barren</td>
<td>70.77</td>
<td>182.00</td>
</tr>
<tr>
<td>Alders</td>
<td>199.51</td>
<td>175.28</td>
</tr>
<tr>
<td>Marsh/Swamps</td>
<td>20.37</td>
<td></td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td><strong>Total Non-Forested</strong></td>
<td><strong>745.84</strong></td>
<td><strong>Total Forested</strong></td>
</tr>
</tbody>
</table>

Table 8.2. Distribution of Forested Stands by Age and Merchantable Volume

<table>
<thead>
<tr>
<th>Age Range</th>
<th>Stand Count</th>
<th>Area (Hectares)</th>
<th>% of Total Area</th>
<th>Merch. Vol. (Cubic Metres)</th>
<th>% of Total Merch. Vol</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-20</td>
<td>162</td>
<td>1,138.0</td>
<td>25.33%</td>
<td>104.3</td>
<td>0.03%</td>
</tr>
<tr>
<td>21-40</td>
<td>198</td>
<td>749.8</td>
<td>16.69%</td>
<td>22,471.6</td>
<td>7.28%</td>
</tr>
<tr>
<td>41-60</td>
<td>540</td>
<td>2,044.9</td>
<td>45.51%</td>
<td>201,248.8</td>
<td>65.20%</td>
</tr>
<tr>
<td>61-80</td>
<td>103</td>
<td>429.6</td>
<td>9.56%</td>
<td>72,396.4</td>
<td>23.46%</td>
</tr>
<tr>
<td>81-100</td>
<td>3</td>
<td>18.5</td>
<td>0.41%</td>
<td>2,411.4</td>
<td>0.78%</td>
</tr>
<tr>
<td>All Aged</td>
<td>33</td>
<td>112.5</td>
<td>2.50%</td>
<td>10,020.1</td>
<td>3.25%</td>
</tr>
</tbody>
</table>
Table 8.3. Stands by Cover Type

<table>
<thead>
<tr>
<th>Covertype</th>
<th>Merch. Vol. (m³)</th>
<th>Merch Vol. (% of Total)</th>
<th>Area (Hectares)</th>
<th>Area (% of Total)</th>
<th>Stand Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unclassified</td>
<td>0</td>
<td>0.00</td>
<td>764</td>
<td>17.00</td>
<td>94</td>
</tr>
<tr>
<td>Softwood</td>
<td>188,411</td>
<td>61.00</td>
<td>2,546</td>
<td>56.70</td>
<td>651</td>
</tr>
<tr>
<td>Mixedwood</td>
<td>7,629</td>
<td>2.50</td>
<td>144</td>
<td>3.20</td>
<td>38</td>
</tr>
<tr>
<td>Hardwood</td>
<td>112,312</td>
<td>36.50</td>
<td>1,039</td>
<td>23.10</td>
<td>256</td>
</tr>
</tbody>
</table>

Table 8.4. Shulkell Stands by Stand Type

<table>
<thead>
<tr>
<th>Stand Class</th>
<th>Merch. Vol. (m³)</th>
<th>% of Total Merch. Vol.</th>
<th>Hectares</th>
<th>% of Total Hectares</th>
<th>Number of Stands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural</td>
<td>256,429</td>
<td>83.1%</td>
<td>2,670</td>
<td>59.4%</td>
<td>722</td>
</tr>
<tr>
<td>Dead</td>
<td>79</td>
<td>0.0%</td>
<td>594</td>
<td>13.2%</td>
<td>73</td>
</tr>
<tr>
<td>Dead-1</td>
<td>31,764</td>
<td>10.3%</td>
<td>716</td>
<td>15.9%</td>
<td>153</td>
</tr>
<tr>
<td>Dead-2</td>
<td>20,380</td>
<td>6.6%</td>
<td>157</td>
<td>3.5%</td>
<td>44</td>
</tr>
<tr>
<td>Plantation</td>
<td>0</td>
<td>0.0%</td>
<td>182</td>
<td>4.1%</td>
<td>23</td>
</tr>
<tr>
<td>Clearcut</td>
<td>0</td>
<td>0.0%</td>
<td>175</td>
<td>3.9%</td>
<td>24</td>
</tr>
<tr>
<td>Totals</td>
<td>308,653</td>
<td>100.0%</td>
<td>4,493</td>
<td>100.0%</td>
<td>1,039</td>
</tr>
</tbody>
</table>

8.1.2. Alternate Datasets

Alternate datasets were generated from the Shulkell data to test robustness of the algorithms. The datasets were produced by manipulating the age of all stands except those classified as dead or clearcut. To maintain the distribution of stands across age-classes, a discrete probability distribution for ageclass was created from the Shulkell dataset. Then, the new ages were generated by drawing a random sample of ages from the empirically determined distribution. Five new datasets were generated. Images of all datasets are in Appendix B, Figures B.1 to B.6. Figures B.7 and B.8 show the relative distribution of stands by ageclass in these alternate datasets.
8.1.3. Roading Network

Existing main roads and the proposed road network are shown in Figure 8.2. The proposed road network has 135 links (edges) and 125 nodes representing a total of 68.7 kilometres of potential road building. The roading network was developed with some assistance from Nova Scotia Dept. Lands and Forests personnel. This was then mapped onto the GIS coverage of the forest area using ARCVIEW™. Each stand which extended further than .5 km from a main road was assigned the road link which was nearest. This stand to road link assignment was performed manually, with the assistance of the ARCVIEW™ software. 274 of the stands are accessible by main road. The remainder (765 stands) require the construction of at least one road link if they are to be harvested.

Figure 8.2. Shulkell Area Road Network
8.1.4. Model Parameters

The planning horizon is a total of 20 years in four 5-year periods. Adjacency delay and maximum opening sizes were determined from the Province of Nova Scotia [1990] Forest/Wildlife Guidelines and Standards. The maximum opening size is 50 hectares, except for openings which include dead stands. In this case, the acreage contributed by a dead stand is weighted by a factor of 0.5. This is to represent the decreased insistence on a maximum opening size in the spirit of the guidelines [Province of Nova Scotia 1990]. The total harvest volume target of two million (solid) cubic feet was determined by using the SAWS simulation and optimization program [Gunn 1994b]. Model parameters are shown in Table 8.5.

<table>
<thead>
<tr>
<th>Table 8.5. Model Parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning Period Length</td>
<td>5 years</td>
</tr>
<tr>
<td>Planning Horizon</td>
<td>4 periods</td>
</tr>
<tr>
<td>Total Volume Target</td>
<td>2,000,000 cubic feet</td>
</tr>
<tr>
<td>Periodic Volume Target</td>
<td>500,000 cubic feet</td>
</tr>
<tr>
<td>Percentage Deviation from Total</td>
<td>2%</td>
</tr>
<tr>
<td>Percentage Deviation from Periodic</td>
<td>4%</td>
</tr>
<tr>
<td>Adjacency Delay Time</td>
<td>2 periods</td>
</tr>
<tr>
<td>Maximum Opening Size</td>
<td>50 hectares</td>
</tr>
<tr>
<td>Maxopen factor on Dead stands</td>
<td>50%</td>
</tr>
<tr>
<td>Cost/Km of Road Construction</td>
<td>$5,000</td>
</tr>
<tr>
<td>Annual Discount Rate</td>
<td>10%</td>
</tr>
</tbody>
</table>

The Revised Growth and yield tables for Nova Scotia Softwoods [Province of Nova Scotia 1993] were used to provide the mai curves for the datasets. For this study, the softwood covertype was arbitrarily assigned to all stands, and the second-rotation, natural stand growth model was used. Thus, the dataset represents the area in the spatial configuration of stands, but growth and volume data are not necessarily true values.
8.2. The Algorithms

Fixed Tabu Search, Reactive Tabu Search with and without random diversification, Oscillating Tabu Search and Oscillating Reactive Tabu search algorithms were tested. All results reported in this section are based on an equal weighting of road cost and lost volume factors in the objective function, and were produced using the SHULKELL dataset.

8.2.1. Fixed Tabu Search

The Fixed Tabu Search (FTS) algorithm (as described in the previous chapter) allows only feasible moves, and its only free parameter is tabu_tenure. Samples (size 30) of solutions were generated for nine settings of tabu_tenure. The search was terminated after 5000 iterations (neighbourhood searches) or if no feasible non-tabu move was found. Table 8.6 shows the sample mean, best (minimum) solution, standard deviation, average number of steps taken (Mean T), number of steps to reach the best solution and exit status for the run which achieved the best solution.

First, observe that as tenure is increased to 360 and 410, the program exits with no feasible moves found. Secondly, as tabu tenure is increased, a general decrease in standard deviation occurs until tenure reaches 310. The search appears to be inefficient for larger values of tabu tenure, since the best move is found very early in the search. This indicates either an over-constrained search trajectory or a lack of diversification in the search, with the trajectory being "stuck" at the local optimum found. By observation, the quality of the solution obtained and the standard deviation of the sample varies significantly with the value of tabu_tenure which is chosen. (See table 8.6) The best observed solution for this instance is 494,293; it is found for tabu_tenure equal to 310.
Table 8.6. FTS Solutions

<table>
<thead>
<tr>
<th>Tenure</th>
<th>Mean Z</th>
<th>Best Z</th>
<th>Std. Dev.</th>
<th>Mean T</th>
<th>Time of Best</th>
<th>Exit Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>589,963</td>
<td>562,467</td>
<td>17,212</td>
<td>1,345</td>
<td>4107</td>
<td>Max iterations reached</td>
</tr>
<tr>
<td>60</td>
<td>571,098</td>
<td>542,240</td>
<td>17,017</td>
<td>2,862</td>
<td>2102</td>
<td>Max iterations reached</td>
</tr>
<tr>
<td>110</td>
<td>561,668</td>
<td>538,072</td>
<td>14,305</td>
<td>2,046</td>
<td>1109</td>
<td>Max iterations reached</td>
</tr>
<tr>
<td>160</td>
<td>556,369</td>
<td>536,441</td>
<td>13,286</td>
<td>1,921</td>
<td>1010</td>
<td>Max iterations reached</td>
</tr>
<tr>
<td>210</td>
<td>544,288</td>
<td>519,543</td>
<td>13,257</td>
<td>1,587</td>
<td>1429</td>
<td>Max iterations reached</td>
</tr>
<tr>
<td>260</td>
<td>533,619</td>
<td>512,293</td>
<td>12,907</td>
<td>1,553</td>
<td>1042</td>
<td>Max iterations reached</td>
</tr>
<tr>
<td>310</td>
<td>530,876</td>
<td>494,293</td>
<td>16,006</td>
<td>1,015</td>
<td>1179</td>
<td>Max iterations reached</td>
</tr>
<tr>
<td>360</td>
<td>536,122</td>
<td>509,467</td>
<td>16,461</td>
<td>646</td>
<td>932</td>
<td>No Move Found</td>
</tr>
<tr>
<td>410</td>
<td>539,866</td>
<td>524,106</td>
<td>12,572</td>
<td>530</td>
<td>521</td>
<td>No Move Found</td>
</tr>
</tbody>
</table>

8.2.2. Fixed Tabu Search with Random Diversification Moves (FTSESC)

Next, the FTS algorithm was augmented with random diversification moves. As described in chapter 7, these moves are a random selection of stands which are then added or deleted from the harvest. All moves, including the diversification moves, must be feasible. These random moves were invoked whenever the search had executed 200 iterations without finding an improved best solution. The search was terminated when 5000 iterations were completed and at least 500 iterations since the last execution of diversification moves had been taken. This enhancement was tested on the same thirty initial solutions as was done for FTS (Table 8.7).

To compare FTSESC and FTS, percentage difference in solution value found and standard deviation for each tabu tenure were calculated (Table 8.8). Average solution value was better or nearly the same for all selections of tabu tenure. The standard deviation for FTSESC was significantly lower for some cases (tenure = 10, 60, 160, 260, 310) The best solution obtained by FTSESC was less than that of FTS in all but the case where tabu tenure = 310, yet the percentage difference was at most 5.14%. Thus, although these diversification moves improved the algorithm in some cases, the
dependency on *tabu_tenure* still affects results.

Table 8.7. FTSESC Solutions

<table>
<thead>
<tr>
<th>Tenure</th>
<th>Mean z</th>
<th>Best z</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>573927.3</td>
<td>533580.4</td>
<td>15797.32</td>
</tr>
<tr>
<td>60</td>
<td>562187.8</td>
<td>541226.3</td>
<td>13891.18</td>
</tr>
<tr>
<td>110</td>
<td>552213.8</td>
<td>525635.7</td>
<td>15053.7</td>
</tr>
<tr>
<td>160</td>
<td>547635.6</td>
<td>527543.6</td>
<td>10042.51</td>
</tr>
<tr>
<td>210</td>
<td>540197</td>
<td>513337.4</td>
<td>2542.267</td>
</tr>
<tr>
<td>260</td>
<td>535001.5</td>
<td>511371.9</td>
<td>9270.136</td>
</tr>
<tr>
<td>310</td>
<td>534384.3</td>
<td>502723.8</td>
<td>14480.9</td>
</tr>
</tbody>
</table>

Table 8.8. Percentage difference in FTS over FTSESC

<table>
<thead>
<tr>
<th>Tenure</th>
<th>Mean</th>
<th>Best</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2.79%</td>
<td>5.41%</td>
<td>8.96%</td>
</tr>
<tr>
<td>60</td>
<td>1.58%</td>
<td>0.19%</td>
<td>22.50%</td>
</tr>
<tr>
<td>110</td>
<td>1.71%</td>
<td>2.37%</td>
<td>-4.97%</td>
</tr>
<tr>
<td>160</td>
<td>1.59%</td>
<td>1.69%</td>
<td>32.30%</td>
</tr>
<tr>
<td>210</td>
<td>0.76%</td>
<td>1.21%</td>
<td>4.28%</td>
</tr>
<tr>
<td>260</td>
<td>-0.26%</td>
<td>0.18%</td>
<td>39.23%</td>
</tr>
<tr>
<td>310</td>
<td>-0.66%</td>
<td>-1.68%</td>
<td>10.53%</td>
</tr>
</tbody>
</table>

8.2.3. Reactive Tabu Search (RTS)

This algorithm is a straightforward implementation of *R-Tabu* [Battiti and Tecchiolli 1995]. The search is confined to the feasible region, and *tabu_tenure* is varied as cycles are detected. The hashing function \( H(x) \) is as described in section 7.6.2 with linear probing to resolve collisions. Cycles of length less than or equal to fifty moves are used in calculating the moving average of cycle length, and tabu tenure is increased upon detection of a repeated solution. Tabu tenure is decreased by a factor of 0.9 after a number of moves (equal to *moving_average*) have been executed without repeating a solution. If three solutions have been repeated three times, the system is assumed to be
chaotically cycling and the ESCAPE1 diversification sequence is executed.

Two versions of RTS were tested. The first, RTS1, follows the R-Tabu algorithm as described above. The second, RTS2, is augmented by diversification moves (as in FIXEDESC) when a large number of steps have been taken without improvement in the best solution. RTS2 produced slightly better results than RTS1, and standard deviation of the sample for RTS2 was significantly lower (15.96%). Surprisingly, RTS did not find better solutions than FTS. The reasons for this were not made apparent by the experiments.

<table>
<thead>
<tr>
<th></th>
<th>RTS1</th>
<th>RTS2</th>
<th>100(RTS2-RTS1)/RTS1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>586,292</td>
<td>582,893</td>
<td>-0.58%</td>
</tr>
<tr>
<td>Best</td>
<td>552,214</td>
<td>551,724</td>
<td>-0.09%</td>
</tr>
<tr>
<td>Worst</td>
<td>618,664</td>
<td>614,001</td>
<td>-0.75%</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>18,062</td>
<td>15,182</td>
<td>-15.95%</td>
</tr>
</tbody>
</table>

Table 8.9. Reactive Tabu Search Results

8.2.4. Oscillating Tabu Search (OTS)

Oscillating tabu search, as described in Chapter 7, guides the search trajectory in and out of the feasible region. There are three constraint sets, (restrictions on total and periodic harvests and maximum opening size constraints) for which penalty parameters $\alpha$, $\beta$ and $\gamma$ are defined. Thus, (recall that $\rho$ is used only to examine tradeoffs), the objective function $z'(x) = LV(x) + \rho RC(x)$ is replaced with

$$z'(x) = LV + \rho \ CD(x) + \alpha \ Dev\_total + \beta \ Dev\_per + \gamma \ Dev\_maxopen$$

Parameters $\alpha$, $\beta$ and $\gamma$ are initially set to their upper bound, and are increased (doubled) or decreased (halved) according to the feasibility of the previous set of ten solutions with respect to each penalty term. Tabu_tenure is the only free parameter. The
stopping rule terminates the algorithm at a maximum number of iterations (maxiters) with no improvement in the last 300 iterations. Figure 8.3 shows a sample OTS solution trajectory.

![OTS Solution Trajectory Diagram](image)

Figure 8.3. OTS Solution Trajectory

The OTS algorithm out-performs FTS, FTSESC, RTS1 and RTS2 with consistently better average and best solutions (See table 8.10). As for determining a value of tabu tenure that is "best", these results are somewhat inconclusive. The best solution was found with tabu_tenure at 40, the lowest sample average solution was for tabu_tenure equal to 70, and the tightest range of solutions was found with tenure equal to 50.

8.2.5. Oscillating Reactive Tabu Search

Oscillating Reactive Tabu Search (ORTS) is a hybrid of the reactive tabu search and strategic oscillation methods. ORTS includes the expanded neighbourhood of OTS, and eliminates the need to pre-set tabu_tenure by implementing the R-Tabu feedback mechanisms.

Several variant algorithms were examined. Some minor algorithm design issues are setting RTS parameters for cycle_max, increase, decrease, and determining the best
method for modifying penalty coefficients \( \alpha \), \( \beta \), and \( \gamma \). No significant differences were found by modifying the factors \( \textit{increase, decrease and cycle_max} \). Thus these parameters were set to 1.1, 0.9 and 50 respectively.

<table>
<thead>
<tr>
<th>Tenure</th>
<th>Maxiter</th>
<th>Mean</th>
<th>Best</th>
<th>Worst</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5000</td>
<td>466,461</td>
<td>426,962</td>
<td>494,487</td>
<td>18,093</td>
</tr>
<tr>
<td>10</td>
<td>8000</td>
<td>461,499</td>
<td>426,962</td>
<td>494,487</td>
<td>19,075</td>
</tr>
<tr>
<td>20</td>
<td>5000</td>
<td>457,806</td>
<td>424,731</td>
<td>492,361</td>
<td>14,895</td>
</tr>
<tr>
<td>20</td>
<td>8000</td>
<td>449,631</td>
<td>415,225</td>
<td>478,257</td>
<td>16,402</td>
</tr>
<tr>
<td>30</td>
<td>5000</td>
<td>450,222</td>
<td>423,815</td>
<td>474,724</td>
<td>13,367</td>
</tr>
<tr>
<td>30</td>
<td>8000</td>
<td>444,960</td>
<td>416,515</td>
<td>474,274</td>
<td>14,707</td>
</tr>
<tr>
<td>40</td>
<td>5000</td>
<td>449,042</td>
<td>415,347</td>
<td>486,525</td>
<td>15,843</td>
</tr>
<tr>
<td>40</td>
<td>8000</td>
<td>439,590</td>
<td>407,128</td>
<td>476,982</td>
<td>16,704</td>
</tr>
<tr>
<td>50</td>
<td>5000</td>
<td>440,029</td>
<td>413,195</td>
<td>472,798</td>
<td>15,220</td>
</tr>
<tr>
<td>50</td>
<td>8000</td>
<td>430,477</td>
<td>413,195</td>
<td>455,075</td>
<td>10,692</td>
</tr>
<tr>
<td>60</td>
<td>5000</td>
<td>444,664</td>
<td>414,692</td>
<td>494,833</td>
<td>18,431</td>
</tr>
<tr>
<td>60</td>
<td>8000</td>
<td>436,171</td>
<td>414,019</td>
<td>482,642</td>
<td>15,858</td>
</tr>
<tr>
<td>70</td>
<td>5000</td>
<td>436,507</td>
<td>413,103</td>
<td>468,319</td>
<td>15,951</td>
</tr>
<tr>
<td>70</td>
<td>8000</td>
<td>428,906</td>
<td>410,887</td>
<td>467,525</td>
<td>12,588</td>
</tr>
<tr>
<td>80</td>
<td>5000</td>
<td>437,960</td>
<td>414,460</td>
<td>475,529</td>
<td>17,258</td>
</tr>
<tr>
<td>80</td>
<td>8000</td>
<td>432,034</td>
<td>410,721</td>
<td>475,529</td>
<td>17,409</td>
</tr>
<tr>
<td>90</td>
<td>5000</td>
<td>438,284</td>
<td>411,317</td>
<td>484,410</td>
<td>17,296</td>
</tr>
<tr>
<td>90</td>
<td>8000</td>
<td>429,619</td>
<td>411,317</td>
<td>453,426</td>
<td>10,945</td>
</tr>
<tr>
<td>100</td>
<td>5000</td>
<td>435,571</td>
<td>416,943</td>
<td>481,530</td>
<td>15,569</td>
</tr>
<tr>
<td>100</td>
<td>8000</td>
<td>431,492</td>
<td>412,454</td>
<td>480,716</td>
<td>14,531</td>
</tr>
</tbody>
</table>

The major design issues are to produce an effective diversification strategy and to integrate this strategy with strategic oscillation. Amongst diversification strategies, the \( \textit{ESCAPE3} \) method, which uses a combined stand and link based strategy (section 7.3) was found to be the best. This section documents results from comparisons of five ORTS algorithms (Table 8.11). It was found that, for this problem, it was important to start with a feasible solution, and that the \( \textit{ESCAPE3} \) diversification mechanism was the most effective. Table 8.12 contains statistics from a sample of thirty different starting
solutions on the SHULKELL dataset. ORTS outperformed the other schemes with the best solution and lowest sample average. The sample standard deviation was higher than ORTSB1 and ORTSB2, but not significantly so. Thus it was concluded that this ORTS algorithm was the best found so far. It remained to determine robustness of the algorithm across different datasets and the differing cost coefficients which occur when $\rho$ is varied.

Evidence that ORTS is a robust algorithm is presented in Table 8.13. ORTS was run on each dataset for values of $\rho$ ranging from 0.1 to 20. Thus, it was tested on 114 problem instances. Tables D.1 to D.6 contain sample statistics for each dataset and each value of $\rho$. Each instance was repeated for thirty different starting solutions. Table 8.13 shows the coefficient of variation for each sample. The worst case is on SHULKELL for $\rho$ equal to 0.1 and 0.2 where standard deviation is less than five percent of the mean.

<table>
<thead>
<tr>
<th>ORTSA1</th>
<th>Starting Solution may not be feasible to maxopen.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ESCAPE1 diversification.</td>
</tr>
<tr>
<td>ORTSB1</td>
<td>Starting Solution may not be feasible to maxopen.</td>
</tr>
<tr>
<td></td>
<td>ESCAPE2 diversification</td>
</tr>
<tr>
<td>ORTSA2</td>
<td>Starting Solution feasible to maxopen.</td>
</tr>
<tr>
<td></td>
<td>ESCAPE1 diversification</td>
</tr>
<tr>
<td>ORTSB2</td>
<td>Starting Solution feasible to maxopen.</td>
</tr>
<tr>
<td></td>
<td>ESCAPE2 diversification</td>
</tr>
<tr>
<td>ORTS</td>
<td>Starting Solution feasible to maxopen.</td>
</tr>
<tr>
<td></td>
<td>ESCAPE3 diversification</td>
</tr>
</tbody>
</table>
Table 8.12. ORTS Sample Results

<table>
<thead>
<tr>
<th></th>
<th>ORTSA1</th>
<th>ORTSB1</th>
<th>ORTSA2</th>
<th>ORTSB2</th>
<th>ORTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>427,647</td>
<td>427,031</td>
<td>425,306</td>
<td>425,011</td>
<td>416,191</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>5,277</td>
<td>4,627</td>
<td>5,429</td>
<td>4,195</td>
<td>4,907</td>
</tr>
<tr>
<td>Variance</td>
<td>27,849,815</td>
<td>21,406,897</td>
<td>29,469,198</td>
<td>17,602,096</td>
<td>24,078,967</td>
</tr>
<tr>
<td>Range</td>
<td>18,776</td>
<td>16,610</td>
<td>22,067</td>
<td>18,161</td>
<td>16,656</td>
</tr>
<tr>
<td>Minimum</td>
<td>419,463</td>
<td>419,050</td>
<td>415,831</td>
<td>414,865</td>
<td>406,966</td>
</tr>
<tr>
<td>Maximum</td>
<td>438,239</td>
<td>435,661</td>
<td>437,898</td>
<td>433,026</td>
<td>423,622</td>
</tr>
</tbody>
</table>

Table 8.13. Coefficient of Variation

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>SHULKELL</th>
<th>SET 1</th>
<th>SET 2</th>
<th>SET 3</th>
<th>SET 4</th>
<th>SET 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>4.94%</td>
<td>2.88%</td>
<td>2.39%</td>
<td>2.41%</td>
<td>1.68%</td>
<td>1.62%</td>
</tr>
<tr>
<td>0.20</td>
<td>4.61%</td>
<td>3.65%</td>
<td>2.35%</td>
<td>3.00%</td>
<td>2.32%</td>
<td>2.53%</td>
</tr>
<tr>
<td>0.30</td>
<td>2.45%</td>
<td>2.46%</td>
<td>2.13%</td>
<td>2.32%</td>
<td>2.12%</td>
<td>1.61%</td>
</tr>
<tr>
<td>0.40</td>
<td>2.51%</td>
<td>3.23%</td>
<td>2.95%</td>
<td>2.96%</td>
<td>1.77%</td>
<td>2.20%</td>
</tr>
<tr>
<td>0.50</td>
<td>1.70%</td>
<td>2.30%</td>
<td>1.97%</td>
<td>1.90%</td>
<td>1.35%</td>
<td>1.48%</td>
</tr>
<tr>
<td>0.60</td>
<td>1.92%</td>
<td>2.45%</td>
<td>2.98%</td>
<td>1.85%</td>
<td>1.57%</td>
<td>2.24%</td>
</tr>
<tr>
<td>0.70</td>
<td>2.22%</td>
<td>2.04%</td>
<td>1.97%</td>
<td>1.48%</td>
<td>1.21%</td>
<td>1.32%</td>
</tr>
<tr>
<td>0.80</td>
<td>1.48%</td>
<td>2.20%</td>
<td>1.49%</td>
<td>2.06%</td>
<td>1.08%</td>
<td>1.42%</td>
</tr>
<tr>
<td>0.90</td>
<td>2.39%</td>
<td>1.71%</td>
<td>1.73%</td>
<td>2.12%</td>
<td>0.83%</td>
<td>1.51%</td>
</tr>
<tr>
<td>1.00</td>
<td>1.17%</td>
<td>3.00%</td>
<td>3.08%</td>
<td>2.57%</td>
<td>1.08%</td>
<td>1.19%</td>
</tr>
<tr>
<td>2.00</td>
<td>0.92%</td>
<td>1.80%</td>
<td>1.91%</td>
<td>1.09%</td>
<td>1.02%</td>
<td>1.13%</td>
</tr>
<tr>
<td>3.00</td>
<td>0.78%</td>
<td>0.93%</td>
<td>1.17%</td>
<td>1.11%</td>
<td>0.97%</td>
<td>0.98%</td>
</tr>
<tr>
<td>4.00</td>
<td>1.25%</td>
<td>1.03%</td>
<td>1.48%</td>
<td>0.96%</td>
<td>0.55%</td>
<td>1.65%</td>
</tr>
<tr>
<td>5.00</td>
<td>1.13%</td>
<td>1.29%</td>
<td>0.64%</td>
<td>1.26%</td>
<td>0.51%</td>
<td>1.51%</td>
</tr>
<tr>
<td>6.00</td>
<td>1.20%</td>
<td>1.98%</td>
<td>1.12%</td>
<td>1.27%</td>
<td>0.78%</td>
<td>1.34%</td>
</tr>
<tr>
<td>7.00</td>
<td>0.86%</td>
<td>1.13%</td>
<td>0.98%</td>
<td>0.81%</td>
<td>0.66%</td>
<td>0.99%</td>
</tr>
<tr>
<td>8.00</td>
<td>1.14%</td>
<td>0.94%</td>
<td>0.87%</td>
<td>0.41%</td>
<td>0.75%</td>
<td>0.84%</td>
</tr>
<tr>
<td>9.00</td>
<td>1.31%</td>
<td>0.97%</td>
<td>1.09%</td>
<td>0.48%</td>
<td>1.82%</td>
<td>2.32%</td>
</tr>
<tr>
<td>10.00</td>
<td>2.21%</td>
<td>0.87%</td>
<td>2.09%</td>
<td>1.94%</td>
<td>1.78%</td>
<td>1.74%</td>
</tr>
<tr>
<td>11.00</td>
<td>3.36%</td>
<td>3.52%</td>
<td>2.07%</td>
<td>2.57%</td>
<td>1.99%</td>
<td>2.52%</td>
</tr>
<tr>
<td>12.00</td>
<td>2.78%</td>
<td>2.69%</td>
<td>1.87%</td>
<td>2.80%</td>
<td>1.97%</td>
<td>2.62%</td>
</tr>
<tr>
<td>13.00</td>
<td>2.44%</td>
<td>3.07%</td>
<td>1.82%</td>
<td>2.73%</td>
<td>1.58%</td>
<td>2.17%</td>
</tr>
<tr>
<td>14.00</td>
<td>2.53%</td>
<td>3.09%</td>
<td>1.78%</td>
<td>2.66%</td>
<td>1.60%</td>
<td>2.00%</td>
</tr>
<tr>
<td>15.00</td>
<td>2.17%</td>
<td>2.93%</td>
<td>1.75%</td>
<td>1.96%</td>
<td>1.62%</td>
<td>1.84%</td>
</tr>
<tr>
<td>16.00</td>
<td>1.77%</td>
<td>2.75%</td>
<td>2.02%</td>
<td>1.60%</td>
<td>1.75%</td>
<td>1.87%</td>
</tr>
<tr>
<td>17.00</td>
<td>1.45%</td>
<td>2.11%</td>
<td>1.49%</td>
<td>2.17%</td>
<td>1.99%</td>
<td>1.66%</td>
</tr>
<tr>
<td>18.00</td>
<td>1.00%</td>
<td>1.93%</td>
<td>1.89%</td>
<td>1.73%</td>
<td>2.07%</td>
<td>1.75%</td>
</tr>
<tr>
<td>19.00</td>
<td>1.25%</td>
<td>1.53%</td>
<td>1.56%</td>
<td>1.89%</td>
<td>1.96%</td>
<td>1.67%</td>
</tr>
<tr>
<td>20.00</td>
<td>1.25%</td>
<td>1.29%</td>
<td>1.62%</td>
<td>1.36%</td>
<td>1.67%</td>
<td>1.67%</td>
</tr>
</tbody>
</table>
8.3. Tradeoff Analysis

Recall the objective function

\[ z'(x) = LV + \rho \cdot RC(x) + \alpha \cdot Dev\_total + \beta \cdot Dev\_per + \gamma \cdot Dev\_maxopen \]

The coefficient of road cost, \( \rho \), is a weighting factor. The unweighted roading costs are in thousands of dollars, and are discounted at 10% per year. When \( \rho \) is increased, solutions with lower roading costs are favoured by the algorithm. Similarly, decreasing \( \rho \) leads to solutions which allow more road building costs so as to reduce the lost volume penalty. The tradeoff curve is the efficient frontier for the two attributes, minimum road cost and minimum lost productivity.

Tradeoff curves were produced by varying \( \rho \) from 0.1 to 20, and running ORTS for 30 different starting solutions. Figures D.1 - D.6 are charts of these solutions. The tradeoff curve for the Shulkell dataset is in Figure 8.4. At the extremes, it "costs" 950,000 cubic feet of lost volume to spend nothing on road building. At the other extreme, spending $175,000 on road building (at $5000 per km) drives the lost volume to approx. 300,000. More importantly, increasing investment in road building to $40,000 can reduce the lost volume to approximately 425,000 ft\(^3\), a reduction of nearly 55%. Furthermore, the curve shows that there is very little point in increasing road spending to more than $80,000 since the returns on reducing lost volume are small.
Figure 8.4. Efficient frontier for Shulkell Dataset.

Figures D.7 to D.11 show several harvest schedules, with road building costs ranging from no cost to those which incur $105,419 in road construction costs. Figures D.12 and D.13 show opening configurations for one of these solutions.
CHAPTER 9

DISCUSSION

9.1. Summary

This model and tabu search solution methodology provide a flexible framework for posing and solving forest planning problems. The model is the first to successfully deal with defining adjacency constraints on stands, with no restrictions on the number of planning periods in the adjacency delay time. The model also allows for general roading networks with multiple access points and cycles. The objective function lost volume penalty function is a very different concept from any in the forest management literature. This work has shown that it is possible to design and solve mathematical models to support the analysis of tradeoffs between competing factors such as lost production and capital cost of road building. This capability adds significantly to the decision making process -- no assumptions are made regarding the manager's tolerance for lost productivity nor his/her roading budget. Thus, the information can be used to freely analyze the substitution effects without bias.

It has been shown that tabu search methods can be used to solve this problem. The most important feature of the TS algorithm is the strategic oscillation through the feasible region boundaries. This feature is essential in obtaining good solutions -- without it all methods failed. The algorithm is enhanced with a reactive feedback mechanism to automatically set tabu tenure. This eliminates the need for a pre-solution
experiment to determine a good value of tabu tenure. It was also shown that
diversification with stochastic elements is useful to narrow the range of solutions
produced, and to improve overall solutions. This problem has two very clear sub-
structures. The harvest scheduling portion is like a bin-packing problem, and the roading
network is like a fixed-charge problem. These make for a complicated solution space and
difficulties in designing a suitable move evaluation function. These barriers were
addressed in the diversification strategies, by choosing moves which diversify while
selectively increasing or decreasing the road network.

Successful tabu search algorithms require efficient methods to evaluate moves.
This problem, which has a complex constraint structure, was successfully attacked by
using some basic graph methods to deal with openings and an efficient heuristic method
for calculating near-optimal road networks. The algorithm was implemented in
FORTRAN, and no claims are made as to the efficiency of the coding. In fact, the
program is not especially fast. It requires approximately one minute of CPU time per
thousand neighbourhood searches (using an IBM R6000 Model 43P-132 computer). It is
workable, however, and the importance of the information produced offsets the
magnitude of computing costs.

A major design objective for the model and algorithm is portability to different
problem instances. Hence, the emphasis on self-tuning for penalty parameters, tabu
tenure and history-based detection of the need for search diversification. This algorithm
is a framework which can be adapted to other problems.

9.2. Directions for Further Research

Some areas where further research is indicated are integer programming
formulations, managing GIS systems and stand characteristics, the form of productivity
loss functions and the incorporation of different goals and objectives into the model. Extensions of the model to allow for more intervention types and for an expanded definition of stand access could prove to be useful.

9.2.1. Integer Programming Formulations

In Integer Programming formulations of adjacency, the clique constraints are the most important inequalities. For small problems, it has been shown (Chapter 3) that it is feasible to produce all of the maximal clique constraints and significantly reduce the running time of the branch and bound solution method. The number of variables in a problem formulation increases with the number of polygons, the number of intervention types, and the number of periods. As problem size increases, generating all the maximal clique constraints becomes unworkable. In addition, it can be shown that there are a large number of lifted odd hole inequalities for these problems, yet experience has been that few of them are violated in LP solutions [Weintraub, Barahona and Epstein 1994]. Thus, a cutting planes algorithm, in which violated inequalities are identified and added systematically to the simplex tableau, is needed. Significant work has already been done in this area, but it has not been applied, in any great extent, to forest planning problems. Nemhauser and Sigismondi [1992] have developed a modular LP/branch-and-bound system for the node packing problem. They augment the inequality system by identifying violated maximal clique and lifted odd hole inequalities. Weintraub, Barahona and Epstein [1994] used a similar strategy in solving the node packing subproblem to generate columns for the master problem in their algorithm. (They also used some other facet-defining inequalities of the node packing problem.)

The node packing problem is only one component of the harvest and road construction scheduling problem, however. What is not known is what other strong
inequalities, which relate the road network to adjacency constraints, are appropriate to further sharpen the LP relaxation of these models.

Thus it is suggested that sharpening the LP formulations for forest planning problems be approached from the existing foundations in cutting planes algorithms.

9.2.2. Stand types and GIS Management

This thesis proposes the use of the forest stand for the basic spatial decision unit. The reasoning is that the stand is the smallest unit which is uniform in age and future growth, and therefore must be the best choice from a biological productivity perspective. There are some problems with this choice, and some thought as to other implications of the spatial unit choice would be helpful. The first problem which was observed through the course of this work is the adjacency definitions. The practice in New Brunswick and Nova Scotia has been to define stands to be adjacent when they share a common boundary, which in the GIS is a shared arc. The GIS coverage data has, in both provinces, been "buffered". This has resulted in stands being split by road buffers, creating artificial boundaries between stands which are, in reality, adjacent or the same stand. Thus, using the GIS data in this way to define adjacency is somewhat suspect. An obvious and simple solution to this difficulty is to use a distance criterion to determine adjacency. The second problem with stands is that they are often too large, and sometimes are too small to be meaningful management choices for harvest or treatment. For example, one stand may be larger than the maximum opening size. This size difficulty can be alleviated by pre-processing the GIS data, splitting any large stands, and aggregating small and artificially fragmented stands. The third problem with stands is their shape. Areas of very irregular perimeter may be unsuitable for operational planning. (For examples of some extremely irregular shapes, see the solutions shown in
Appendix D for the Shulkell area.) This suggests that the area should be "gridded", producing regular shapes that are of a reasonable area and which are, within a tolerance level, uniform in coverage and growth. The drawback is that gridding could create more decision units (although it might in some cases reduce the set).

One point which should be noted is that this model ignores any effects on wood flow and openings resulting from road building. A possible enhancement to the model is to treat road links as special stands, which contribute volume to harvests and which also contribute to openings. This would imply that road links be analyzed, using the GIS, for forest cover and for acreage in order to calculate the wood flows from clearing a road link.

All of these considerations are issues that can be solved with appropriate management of GIS systems and appropriate attention to the impact of spatial unit choices on models, their solution, and the quality of the information they can produce.

9.2.3. Productivity Loss Functions

In this work, a lost volume penalty function has been proposed. The form of this function merits some study. For example, the function should reflect the desire of the decision-maker to access different areas in the forest, as well as deal with lost potential productivity. Thus, different forms of the lost volume function for dead stands may be designed to reflect lower or higher priority in accessing these stands.

The function proposed in this thesis uses an unweighted estimate of total lost volume to penalized less than optimal harvest scheduling. Forms which are differently weighted, or which are non-linear may be appropriate.

9.2.4. Incorporating Different Goals or Constraints.

The model framework, where all constraints are dealt with as penalty terms, is
amenable to modification to include other goals or restrictions. For example, if an end period age-class distribution for the planning area is required, this could be incorporated as another penalty term in the objective. The effect of adding more penalty terms is to increase the complexity of the oscillation procedure; although it appears that this is easily incorporated, the solution algorithm may need to be modified to effect appropriate parameter changes and to include appropriate memory structures to detect the need for diversification.

9.2.5 Other Types Of Intervention and Road Network Extensions

The model considers only one type of intervention in the forest, the clearcut. A useful extension would be to include other types of harvests, such as selective cutting or shelterwood cutting and silvicultural treatments, such as thinnings. These activities generate the same sort of requirements for roading access. No stand would be eligible for both thinning and harvesting within the tactical planning horizon, and thus only one decision variable per stand need be considered. The model would need to be adjusted to deal with differences in contributions to volume and conditions for evaluating eligibility of a stand for thinning.

Contributions to volume from thinnings and selection cuttings can be estimated in the same manner that is done in the aspatial simulation [Gunn1994b]. It can be expected that the strategic plan specify a desired (aspatial) level of silvicultural activity. Also, suitability of a stand for a thinning activity could be measured with a penalty function. The objective function of this extended model would then have another penalty term, and there would be additional constraints (or penalty terms) to ensure feasibility with respect to the silviculture level targets.

Road maintenance costs could be added to this model. Each road link that is
constructed on one period and required in later periods would require maintenance.
These costs would be added to the roading cost portion of the objective function, and
they could be calculated in much the same way that the link timing is done in this model.

The road network model is quite general in that it allows for multiple main road
access points and for cycles in the graph representing the proposed set of links. It does
not allow the condition that one stand may be equally well accessed by more than one
road link. This limitation should be addressed in future work.
REFERENCES


Heuristic Search Methods*, V. J. Rayward-Smith, I. H. Osman, C. Reeves and G. D.


Chronicle. Vol. 69, No. 2, pp. 141-146.


algorithm for concave minimization over a polyhedron. Computers and Operations

harvest scheduling. Proceedings of the 1991 Symposium on Systems Analysis in


pp. 669-680.

constrained harvest scheduling model. Canadian Journal of Forest Research. Vol. 20,
pp. 1438-1447.


lagrangian method applied to forest management. The Wharton School of the

programming - Decision Analytic Perspective. Proceedings of a Workshop on
Hierarchical Approaches to Forest Management in Public and Private Organizations,

57. Gunn, Eldon A. (1994a). Harvest re-allocation to attain maximum, non-declining
sustainable yield. Working paper, Dept. Industrial Engineering, Technical University
of Nova Scotia.

Document, Nova Scotia Department of Natural Resources.

long-term forest harvesting in an integrated industry structure. Canadian Journal of

60. Hansen, Pierre (1986). The steepest ascent mildest descent heuristic for combinatorial
programming. Congress on Numerical Methods in Combinatorial Optimization,
Capri, Italy.


capacitated arc routing problem. Centre de recherche sur les transports, Universite de
Montreal - CRT-97-03.


spatially optimizing wildlife and timber in managed forest ecosystems. Forest


for optimizing timber age-class distributions to meet multispecies wildlife population

implementation. INTERFACES. Vol. 14, No. 5, pp. 77-84.


APPENDIX A

FLOWCHARTS
BEGIN
Seed Random Number Generator
Set Harvest\_j = 0, j=1, J

Begin Period \( j \)

Generate \( s \) in ST-LIST(0)

\[ x_s < (J+1) ? \]

Not Feasible

Feasibility Required?

Calculate Dev\_maxopen

Feasible

\[ x_s = j \]

\[ H_j = H_j + V_{sj} \]

Update Openings

\[ H_j \text{ Feas?} \]

Next Period \( j \)

Total Harvest Feasible ?

Build Road Network

EXIT

Figure A.1. Generating an Initial Solution
best_z = -\infty 
best_move = none 
link = 0

Calculate Re-Timing costs for this link

k=1
s=next stand for this link

x_s = k?

Feasibility Required?

move feasible?

Calculate z(x,m)

Move Aspirated?

Move Tabu?

z(x,m)<best_z

best_z = z(x,m)
best_move = (s,k)

k = k+1

k<= J+1

List for this link done?

More Links?

Figure A.2. Neighbourhood Search
Figure A.3. ESCAPE1 Flowchart
Figure A.4. ESCAPE2 Flowchart
Figure A.5. ESCAPE3 Flowchart
Figure A.6 OTS Algorithm
APPENDIX B

DATASETS
Figure B.1. Alternate Dataset 1
Figure B.2. Alternate Dataset 2
Figure B.3. Alternate Dataset 3
Figure B.5. Alternate Dataset 5
Figure B.6. Shulkell Dataset
Figure B.7. Stand Counts by Ageclass

Figure B.8. Fully Stocked Acres by Ageclass
APPENDIX C

FORTRAN CODE

common blocks

Parameters and Constants

integer*4 numstands,numperiods,nunodes,numroads,ifuture
integer*4 num_opens,maxadj,maxvert
parameter (numstands=1039)
PARAMETER (nperdays=4)
parameter(num_opens=1039,maxvert=440,maxadj=600)
real*4 rmaxopen
PARAMETER (RMAXOPEN=126.5)
parameter(numroads=135)
parameter(nunodes=124)
parameter(ifuture = nperdays+1)
real*8 gamma_upper,lower_bound,upper_bound
parameter(upper_bound=32767.0)
parameter(lower_bound=0.0001)

common/sol_param/htarget,ptarget,periodic_all,total_all,cur_alpha,
>cur_beta,cur_gamma,d_factor,r_factor
real*8 htarget,ptarget,periodic_all, total_all,cur_alpha,cur_beta,cur_gamma,d_factor(ifuture)
real*8 r_factor

best_move

common/best_move/best_r_cost,
> least_objective,
> m_best_loss_penalty,
> m_best_dev_fr_total,
> m_best_dev_fr_periodic,
> m_best_dev_fr_maxopen,
> m_best_harvest,
> best_move_asp,
> best_stand,best_period,best_type,
> best_link_period,
> best_link_period_stands
real*8 best_r_cost,least_objective,
> m_best_loss_penalty,
> m_best_dev_fr_total,
> m_best_dev_fr_periodic(numperiods),
> m_best_harvest,
> m_best_dev_fr_maxopen
> integer*2 best_stand,
> best_period,
> best_type,
> best_link_period,
> best_link_period_stands
> logical best_move_asp

**best_sol**

common/best_sol/x_save,road_save
integer*2 x_save(numstands)
integer*2 road_save(numroads)

**stand_information**

common/data/stand_info,numread
structure /stands/
 real*4 unass_penalty
 real*4 ass_penalty(numperiods)
 real*4 volume(numperiods)
 real*4 acres
 real*4 stocking
 real*4 ivol
 integer*2 adj(maxvert)
 integer*2 link_req
 integer*2 opening(2)
 integer*2 stand_no
 integer*2 dead
 integer*2 for_non
 integer*2 ageclass
 integer*2 site
 integer*2 cover_type
 integer*2 numaadj
end structure
record/stands/stand_info(numstands)
integer*2 numread

**mai_curves**

common/growth/mai,mai_max_age
integer*2 mai_max_age(6)
real*4 mai(6,22)
Hashing Table

common/hash/config_info, hash_table, config_point
parameter(table_size=50021)
integer*2 hash_table(0:table_size)
integer*2 config_point
structure /sol_stats/
real*8 sol_value
integer*2 num_rep
integer*2 time_last_found
end structure
record/sol_stats/config_info(0:10000)

Move Data

common/move/ move_harvest, move_loss_penalty, move_road_cost,
>move_dev_fr_periodic, move_dev_fr_total, move_dev_fr_maxopen,
>move_infeasible, r_cost, move_feas, move_asp, move_stand, move_period, move_type,
>link_period, link_period_stands
integer*2 move_type, move_stand, move_period, link_period, link_period_stands
real*8 move_harvest
real*8 move_loss_penalty, move_dev_fr_periodic(numperperiods),
>move_dev_fr_total, move_infeasible, move_road_cost,
>r_cost(ifuture), move_dev_fr_maxopen
logical move_feas, move_asp

Opening Data

common/openings/opens
common/open_pl/o_avail, o_last, av_list, o_top
structure/op_info/
real*4 acres
integer*2 operiod
integer*2 numvert
integer*2 numedges
integer*2 v(maxvert) !store stand list
integer*2 epointer(maxvert) !used to point to adjacent stands
integer*2 e(maxadj) !adjacency list
integer*2 pointer_fwd ! points to next available opening num
integer*2 pointer_back ! points to the previous opening
end structure
record/op_info/opens(0:num_opens)
integer*2 av_list(num_opens) !availability list for openings
integer*2 o_avail !points to top of availability list
integer*2 o_last !points to last opening added
integer*2 o_top !points to the top of the opening list

RTS Data

common/reactive/moving_average, increase, decrease, rep, chaotic, chaos, cycle_max
integer*2 chaotic, chaos, cycle_max
integer*2 rep
real increase, decrease, moving_average

**Tabu List**

common/tabu/tabu_tenure, tabu_list, steps_since_size_change, link_tenure
integer itabu_len
parameter (itabu_len=numstands)
integer*2 tabu_list(itabu_len)
real tabu_tenure, link_tenure
integer*2 steps_since_size_change

**Road Network**

common/roads/priority, val, roads, road_heap, heap_size, adj_point, inv, adj, st_list, st_point, link, dad
structure/road_net/
real*4 lghth
integer*2 x
integer*2 y
integer*2 period
integer*2 period_assigned
integer*2 nhops
integer*2 dad_link
integer*2 numsons
integer*2 sons(5)
integer*2 time !move when link is last selected for diversify
logical*1 required_by_harvest
logical*1 required_by_connect
end structure
record/road_net/roads(0:numroads)
c heap data structure NOTE ARRAY LOWER BOUND OF 0 IS REQUIRED
integer*2 road_heap(0:numnodes+1), heap_size
integer*2 inv(0:numnodes+1)
integer*2 adj_point(numnodes+2), adj(3*numnodes, 2)
real*4 priority(0:numnodes+1), val(0:numnodes+1)
c list of stands that are accessed by each road
integer*2 st_list(numstands), st_point(0:numroads+1)
integer*2 link(0:numnodes+1)
integer*2 dad(0:numnodes+1)
integer*2 st_list_all(numstands*3), st_all_point(numroads)

**Current Solution**

common/solution/min_loss_penalty, hvol, best_sol, cur_sol, cur_loss_penalty,
> best_loss_penalty, cur_dev_fr_periodic, cur_dev_fr_total, cur_dev_fr_maxopen,
> cur_infeasible, best_infeasible, cur_road_cost, best_road_cost,
> cur_feasible, best_feasible, hash_add, x
integer*2 x(numstands)
real*8 hvol(numperiods),best_sol,cur_sol,cur_loss_penalty,best_loss_penalty,
>cur_dev_fr_periodic(numperiods),cur_dev_fr_total,cur_dev_fr_maxopen
real*8
cur_infeasible,best_infeasible,cur_road_cost,best_road_cost,cur_feasible,best_feasible
real*8 min_loss_penalty
integer*4 hash_add

**Initial Solution**

common/init_sol/i_sol
integer*2 i_sol(numstands)

**Search Status**

common/search_status/ message,reset_value, at_end, num_esc_steps, total_time,
> s_time, etime_struct, maxiters, move_count, moves_since_improve, imp_time,
> feas_count_total, feas_count_per, feas_count_open, infeas_count_total, infeas_count_per,
> infeas_count_open, check_interval, escape_time, no_improve_tol, after_escape_moves
logical at_end
integer*2 maxiters, move_count, moves_since_improve
integer num_esc_steps
integer*2 no_improve_tol, after_escape_moves, feas_count_total, feas_count_per
integer*2 infeas_count_total, infeas_count_per, infeas_count_open, feas_count_open
real total_time, s_time, reset_value
integer*2 imp_time, escape_time, check_interval
TYPE TB_TYPE
    SEQUENCE
    REAL*4 USRTIME
    REAL*4 SYSTIME
END TYPE
TYPE (TB_TYPE) ETIME_STRUCT
character*32 message

**Orts Main Program**

include 'params.f'
include 'sol_param2.f'
include 'com_open.f'
include 'com_data.f'
include 'road_com_sol.f'
include 'com_tabu2.f'
include 'com_move.f'
include 'search_status.f'
include 'road_com.f'
include 'best_move.f'
include 'com_reactive.f'
character*8 mapin
real etime
logical escape
integer icount, istart, numruns
character*12 ofile
character*12 dataset
real seedval

c init

c get data file and operating parameters
   write(,,) 'Reactive tabu search'
   write(,,) 'Dynamic penalty Coeff'
   mapin = 'shulkell'
   write(,,) 'Enter Dataset name'
   read(,,(a12)) dataset

40 format(a8)

c setup data
   call read_growth !read the main values
   call init_x    !set all x(i)=future
   call init_stands !initialize stand structure
   call init_opens !init opening structure
   call init_roads !init road structures
   call get_data(dataset) !read road, stand and adj data
   call calc_base_loss !get min loss penalty

c set up parameters
   call init_reactive !init reactive search parameters
   call init_search_params !maxiters etc
   call calc_parameters !alpha,beta,bounds for deviations etc
   call make_opens2 !openings for clearcuts

c begin the search
   write(,,) 'Enter start stream,end stream'
   read(,,) istart,numruns
   write(,,) 'Enter max iters'
   read(,,) maxiters
   write(,,) 'Enter increase,decrease,cylemax'
   read(,,) increase,decrease,cycle_max
   write(,,) 'Output file'
   read(,,(a12)) ofile
   write(,,) 'First factor'
   read(,,),ifirst
   write(,,) 'Last factor'
   read(,,),ilast
   write(,,) 'Interval'
   read(,,) interval
   write(,,) 'Enter value to reset parameters to after escape'
   read(,,) reset_value
   write(,,) 'No improve tolerance,after escape_moves'
   read(,,) no_improve_tol,after_escape_moves
   link_tenure = 5.0*float(after_escape_moves)

   c open files
   open (unit=11,status='NEW',file=ofile)
   open (unit=21,file=ofile'/.'tr')
   write(11,*) dataset
   call write_heading_rts
do i = 1,numruns !number of runs
write(*,*) 'random stream',icount
call mcarlo_rand feas(icount)
call save_init_sol
!
get an initial solution
!
and save it
!
for the range of r_factor
do k=1,ifirst,ilast,interval
!
do this so the solution can be repeated
!
 converting to real
!
move_count=0 etc
!
set best to big numbers
!
initialize the hashtable
!
init the opening list
!
make the opening graphs
!
build the path network
!
calculate the current solution
!
get start time
!
saves the best values
!
init the tabu list and tenure
!
save initial solution
!
loop control
!
Next solution
!
Do a small number of moves
!
steps_since_size_change=steps_since_size_change+1 !bump counter
call check_for_reps(escape)  !RTS structures for the current solution
!
If not chaotic cycling
!
search the neighborhood for best move
!
found a move
!
implement it
!
update best values and save if necessary
!
bump the move counters
!
no move found
!
max(1.0,int(tabu_tenure*decrease))! decrease tabu tenure
endif
else
!
escape required due to chaotic cycling
!
set the counter
!
implement ESCAPE3 strategy
!
!
endif
!
next icheck
if((move_count.gt.maxiters).and.>

(moves_since_improve.ge.no_improve_tol))then !check stop condition
!
message = 'Max iterations reached'
!
exit
endif
!
next icheck
if(.not.escape)then
!
update alpha, beta, gamma
!
else
!
cur_alpha = reset_value
!
escape was done, set them to 1.0
cur_beta = reset_value
cur_gamma = reset_value
endif
if((moves_since_improve.ge.no_improve_tol).and. ! long time since improve
> (move_count-escape_time.ge.after_escape_moves))then !and intensification phase over
    escape_time=move_count
call rts_escape_routine2                          !do a diversifying strategy
    cur_alpha = upper_bound
    cur_beta = upper_bound
    cur_gamma = upper_bound
    moving_average = 0.0
    !and max out coefficients
endif
call time_stats(etime)                            !get end time
    total_time = etime-s_time
    call write_data_rts(icount)
call get_init_sol
end do
    call clear_solution
    call init_roads
    call init_opens
    call make_opens2
end do
    close(unit=11)
close(unit=21)
stop
end
Initial Solution

```fortran
subroutine mcarlo_rand_feas(icount)
    include 'params.f'
    include 'sol_param2.f'
    include 'search_status.f'
    include 'com_data.f'
    include 'road_com_sol.f'
    include 'com_open.f'
    include 'com_move.f'
    include 'com_growth.f'
    include 'road_com.f'
    real*4 open_size
    integer icount
    integer isize
    real seedval
    seedval = icount
    call srand(seedval)
    isize = st_point(1)-1
    move_type = 1 ! used to update solution
    c make up an initial solution by randomly assigning
c stands that require no roads initially
    move_type = 1
    do move_period = 1,numperiods
        hvol(move_period)=0.0
        do while ( hvol(move_period).lt.(ptarget-periodic_all))
            move_stand = st_list(int(rand())*isize)+1
            if(x(move_stand).eq.ifuture)then ! not in
                if(calc_feas_for_maxopen(move_stand(move_period),move_period,mov
                    e_type)
                    if(trunc(move_dev_fr_maxopen).eq.0.0) then ! feasible
                        if(hvol(move_period)+stand_info(move_stand).volume(move_period)
                        > .le.(ptarget+periodic_all))then
                            if(stand_info(move_stand).link_req.ne.0)then
                                roads(stand_info(move_stand).link_req).period =
                                min(roads(stand_info(move_stand).link_req).period,
                                roads(stand_info(move_stand).link_req).required_by_harvest=)
                                > .true.
                            endif
                        endif
                    endif
                endif
            endif
            call add_stand2(move_stand,move_period)
            x(move_stand)=move_period
            hvol(move_period)=hvol(move_period)+
            > stand_info(move_stand).volume(move_period)
        endif
    enddo
endf
endf
endf
endf
end do
```
end do
return
end

subroutine mcarlo_rand7(icount)
!
This routine generates an initial solution which may be infeasible
!
include 'params.f'
include 'sol_param2.f'
include 'search_status.f'
include 'com_data.f'
include 'road_com_sol.f'
include 'com_open.f'
include 'com_move.f'
include 'com_growth.f'
include 'road_com.f'
real*4 open_size
integer icount
integer isize
!
call srand(s_time) !use start time for the seed
real seedval
seedval = icount
call srand(seedval)
isize = st_point(1)-1
!
c write(*,*) 'isize is ',isize
move_type = 1 !used to update solution
!
c make up an initial solution by randomly assigning
!
c stands that require no roads initially
!
do move_period = 1,numperiods
!
vol(move_period)=0.0
!
do while ( vol(move_period).lt.ptarget)
!
move_stand = st_list(int(rand()*isize)+1)
!
if(x(move_stand).eq.ifuture)then !not in
!
if(stand_info(move_stand).volume(move_period).ne.0)then
!
if(stand_info(move_stand).link_req.ne.0)then
!
roads(stand_info(move_stand).link_req).period =
!
min0(roads(stand_info(move_stand).link_req).period)
!
roads(stand_info(move_stand).link_req).required_by_harvest=
!
true.
!
endif
!
x(move_stand)=move_period
!
vol(move_period)=vol(move_period)+
!
stand_info(move_stand).volume(move_period)
!
endif
!
end do
!
end do

return
end
Hashing Subroutines

---

**subroutine hashfn(numread)**

```fortran
include 'params.f'
include 'sol_param2.f'
include 'road_com_sol.f'
include 'com_hash.f'
include 'search_status.f'
integer*2 numread
integer*4 temp
hash_add=x(1)
do i=2, numread
  temp=x(i)
  hash_add=xor(ishftc(hash_add,1,32),temp)
end do
if (hash_add .lt. 0) then
  hash_add=1*hash_add
endif
hash_add = mod(hash_add,itable_size)
return
end
```

---

**subroutine clean_hash_table**

```fortran
include 'com_hash.f'
include 'search_status.f'
do i=0,config_point
  config_info(i).num_rep=0
end do
return
end
```

---

**subroutine init_hash_table**

```fortran
include 'params.f'
include 'com_hash.f'
config_point = 0
do i=0,itable_size
  hash_table(i)=-1
end do
do i=0,10000
  config_info(i).num_rep=0
```
config_info(i).time_last_found = 32766
config_info(i).sol_value = -1.0
end do
return
end

Openings and Graphs

subroutine del_one_open(op_no)
c This routine deletes the opening op_no
  include 'params.f'
  include 'com_open.f'
  include 'com_data.f'
  integer*2 op_no, period
  integer*2 iprev, inext
c get data
    period = opens(op_no).period
  c get pointers
    iprev = opens(op_no).pointer_back
    inext = opens(op_no).pointer_fwd
  c update stand info
    do k=1, opens(op_no).numvert
      do j=1,2
        if (stand_info(opens(op_no).v(k)).opening(j).eq.op_no) then
          stand_info(opens(op_no).v(k)).opening(j) = 0
        endif
      end do
    end do
  c update pointers
    if ((op_no).eq.0_top) then
      o_top = (inext)
    endif
    if (iprev.ne.0) then
      opens(iprev).pointer_fwd = inext
    endif
    if (inext.ne.0) then
      opens(inext).pointer_back = iprev
    endif
  c zero opening data
    opens(op_no).period = 0
    opens(op_no).acres = 0.0
    opens(op_no).numvert = 0
    opens(op_no).numedges = 0
  do ii = 1, maxvert
    opens(op_no).v(ii) = 0
    opens(op_no).epointer(ii) = 0
end do
do ii=1,maxvert
  opens(op_no).e(ii)=0
end do
opens(op_no).pointer_fwd=0
opens(op_no).pointer_back=0

! update o_avail
av_list(op_no)=o_avail
o_avail = op_no
  if (op_no.eq.o_last) then
    o_last = iprev
  endif
return
end

**subroutine make_opens2**

c this routine makes up all openings from the current settings of x
  include 'params.f'
  include 'search_status.f'
  include 'com_data.f'
  include 'com_open.f'
  include 'road_com_sol.f'
  integer*2 o_period
  integer*2 numon
  integer*2 vert(maxvert)

c given a solution in X(i,j), build all the openings
c do for each opening type
c type 1 = harvest 0,1
c type 2 = harvest 1,2
c type 3 = harvest 2,3
c type 4 = harvest 3,4
do o_period = 1,numperiods

c init vertex list
do i=1,maxvert
  vert(i)=0
end do
numon=0
do i=numon
  if ((x(i).eq.o_period).or.(x(i).eq.o_period-1)) then !add the stand to the list
    numon=numon+1
    vert(numon)=i
  endif
end do

c now, build the openings for this list of stands
if(numon.gt.0)then
call dfs_build2 (vert,numon,o_period)
endif

c next opening period
end do
return
end

--- subroutine del_stand2(stand) ---
c This routine removes stand from harvest
  include 'params.f'
  include 'corn_data.f'
  include 'com_open.f'
  include 'road_com_sol.f'
  integer*2 stand,period,open_no,o_period
  integer*2 vert(maxvert),numon
c check the openings
do ipos = 1,2
  open_no = stand_info(stand).opening(ipos)
  o_period=opens(open_no),period
  stand_info(stand),opening(ipos)=0
  if(open_no.ne.0)then
    if(opens(open_no).numvert .eq. 1) then
      call del_one_open(open_no)
    else
      c build the vertex list
      numon = 0
      do i=1,opens(open_no).numvert
        if(opens(open_no).v(i).ne.stand)then
          numon=numon+1
          vert(numon)=opens(open_no).v(i)
        endif
      enddo
      call del_one_open(open_no)
      call dfs_build2(vert,numon,o_period)
    endif
  endif
  c do the next opening period
do i=1,open_no.numvert
    if(open_no.v(i).ne.stand)then
      numon=numon+1
      vert(numon)=open_no.v(i)
    endif
  enddo
end

--- subroutine dfs_build2(vert,numon,o_period) ---
c This is a depth first search routine to build all connected components of VERT
  include 'params.f'
  include 'com_data.f'
  include 'com_open.f'
  include 'road_com_sol.f'
  include 'search_status.f'
  integer*2 numon,now,vlook,o_period
integer*2 vert(maxvert),val(maxvert),o_adj(maxvert,maxadj)
integer*2 numadj(maxvert)
integer*2 adj_point(maxvert)
logical endchain
integer*2 vlist(maxvert),vadj,epoint,ij,i,j,k
integer*2 rstack(maxvert) !stack to make the recursion work
integer stack_end

c now, the appropriate stands are in array VERT and the adjacencies in o_adj
c make up an OPENING adjacency list for each vertex by intersecting
c VERT with the stand adjacencies.
c Delete the original
c and rebuild
c init vlist, val
   do ii =1,numon
      val(ii)=0
   end do
c init adjacency pointers
   do ii=1,numon
      adj_point(ii) = 1
      numadj(ii)=0
   end do
c build adjacency lists
   do i=1,numon
      numa=stand_info(vert(i)).numadj
      do j=1,numa
         do k=1,numon
            if(stand_info(vert(i)).adj(j).eq.vert(k))then
               numadj(j)=numadj(j)+1
               o_adj(i,numadj(j))=k
               exit
            endif
         end do
      end do
   end do
endchain = .FALSE.
c start filling up openings, first available is o_avail
do i=1,numon !numon is the number of $X_{ij}=1$
   now=0 !visit pointer
do ii=1,numon
   vlist(ii)=0 ! now used to store pointers to found nodes
end do
stack_end = 0
if (val(i).eq.0) then ! node not visited
   now=now+1 !increment visit pointer
   val(i)=now !store visit number
   vlist(now)=i !store node pointer
   vlook = i
   stack_end=stack_end+1
end if
rstack(stack_end)=i !add node to recursion stack
adj_point(vlook),numadj(vlook),endchain ! find a node adjacent to vlook
if (.not.endchain) then ! found one
now=now+1 ! update visit pointer
val(vadj)=now ! store visit number
vlist(now)= vadj ! store found node
vlook = vadj ! next vertex to search in chain
endf

 otherwise, ! pointer = 0 thus, done for this connected component

c create the new the opening structure
opens(o_avail),o_period=o_period
opens(o_avail),numvert=now
opens(o_avail),pointer_fwd = 0 ! forward pointer
opens(o_avail),pointer_back = o_last ! backward pointer
if (o_last.ne.0) then
opens(o_last),o_avail=o_avail ! forward pointer,previous opening
endif
do kk=1,now ! add the vertex information
opens(o_avail),v(kk)=vert(vlist(kk))
if(x(vert(vlist(kk))).eq.o_period)then
stand_info(vert(vlist(kk))).opening(1)=o_avail
else
stand_info(vert(vlist(kk))).opening(2)=o_avail
endif
opens(o_avail),acres=opens(o_avail).acres +
(1.0 - 0.5*stand_info(vert(vlist(kk))).dead)*
> stand_info(vert(vlist(kk))).acres
if(x(vert(vlist(kk))).eq.o_period)then
stand_info(vert(vlist(kk))).opening(1)=o_avail
else
stand_info(vert(vlist(kk))).opening(2)=o_avail
endif
end do

c now, add the edge lists
epoint=0
do  kk=1,now
  do  jj=1,numadj(vlist(kk))
    opens(o_avail).e(epoint+jj) =
      (vert( o_adj(vlist(kk),jj) ))
  end do
  opens(o_avail).epointer(kk) = epoint+1
  epoint=epoint+numadj(vlist(kk))
end do

; add another epointer so that other procedures can find the end of the list
  opens(o_avail).epointer(now+1) =
    opens(o_avail).epointer(now)+numadj(vlist(now))
  opens(o_avail).numedges = opens(o_avail).epointer(now+1)
  o_last=o_avail   ; save last spot
  o_avail = av_list(o_avail)   ; next available spot
end ; updating structures
end

 subroutine merge_opens2(open_list,a_count,stand,o_period)

c  This routine merges the openings in open_list
  include 'params.f'
  include 'com_open.f'
  include 'com_data.f'
  integer*2 stand,o_period
  integer*2 open_list(maxvert),a_count
  integer*2 vert(maxvert),numon

c  build the vertex list
  numon = 0
  do i=1,a_count
    do j = 1, opens(open_list(i)).numvert
      numon=numon+1
      vert(numon)=opens(open_list(i)).v(j)
    end do
  end do
  numon = numon+1
  vert(numon)=stand
  do i=1,a_count
    call del_one_open(open_list(i))
  end do
  call dfs_build2(vert,numon,o_period)
return
end
subroutine make_new_open2(stand,period,o_period)
  c make new opening for this stand
        include 'params.f'
        include 'search_status.f'
        include 'com_open.f'
        include 'com_data.f'
        integer*2 stand,period,o_period
        if(period.eq.o_period)then
          stand_info(stand).opening(1)=o_avail
        else
          stand_info(stand).opening(2) = o_avail !second position
        endif
        opens(o_avail).v(1)=stand
        opens(o_avail).operiod=o_period
        opens(o_avail).numvert=1
        opens(o_avail).numedges = 0
        opens(o_avail).epointer(1)=0
        opens(o_avail).pointer_fwd = 0 !forward pointer
        opens(o_avail).pointer_back = o_last !backward pointer
        if (o_last.ne.0) then
          opens(o_last).pointer_fwd=o_avail     !forward pointer,previous opening
        endif
        opens(o_avail).acres=(1.0 - 0.5*stand_info(stand).dead)*
          stand_info(stand).acres
        o_last=o_avail                !save last spot
        o_avail = av_list(o_avail)    !next available spot
        return
        end

 subroutine add_stand2(stand,period)
  c This routine adds stand in period/
        include 'params.f'
        include 'com_open.f'
        include 'com_data.f'
        include 'road_com_sol.f'
        integer*2 stand,period,a_count,open_list(maxvert),o_period
        integer*2 istart_period,iend_period
        logical found
  c find adjacent openings
        if(period.eq.0)then
          istart_period=1
          iend_period=1
        else if(period.eq.numperiods)then
          istart_period=period
          iend_period=period
        else
          istart_period=period
iend_period=period+1
endif

do o_period=istart_period,iend_period ! one or two possibilities
  a_count=0
  do i=1,stand_info(stand).numadj
    ilook = stand_info(stand).adj(i)
    if(x(ilook).eq.o_period) then ! in first
      j=1
    else ! in second
      j=2
    endif
    if(opens(stand_info(ilook).opening(j)).operateq.o_period) then
      found = .false.
      do kk=1,a_count
        if(stand_info(ilook).opening(j).eq.open_list(kk)) then
          ! already found
            endif
        end do
      if (.not.found) then
        a_count=a_count+1
        open_list(a_count)=stand_info(ilook).opening(j)
      endif
    end do
  if(a_count.gt.0) then
    call merge_opens2(open_list,a_count,stand,o_period)
  else
    call make_new_open2(stand,period,o_period)
  endif
  end do
return
end

--- subroutine findadj(vlook,o_adj,vadj,val, ---

> adj_point,numadj,endpoint) !find a node adjacent to vlook
include 'params.f'
integer*2 vlook,vadj,val(maxvert),o_adj(maxvert,maxadj)
integer*2 adj_point ,numadj,k
logical endchain
logical done
if (adj_point.gt.numadj) then
  endchain = .true.
  return
endif
done = .false.
c adj_point points to the next adjacency point to look at for this vertex
  do k=adj_point,numadj ! search through opening adjacency list for stand ilook
vadj=0_adj(vlook,k)

c add edge (this is a back-edge if val(vadj).ne.0
  if (val(vadj).eq.0) then !stand not visited before
done = .true.
  exit
else !stand visited before, store the back-edge

c insert store back edge here
  endif
  end do
endchain = .not. done
if (done) then
  adj_point=k+1
else
  adj_point = k
endif
return
end

Move Calculations

subroutine calc_full_nbhood6

  include 'params.f'
  include 'com_data.f'
  include 'road_com_sol.f'
  include 'com_move.f'
  include 'com_growth.f'
  include 'road_com.f'
  include 'best_move.f'
  logical asp,tabu
  real*4 big_number
  integer*4 tabu_reject_count

c variables for road links
  integer*2 link_to_change
  integer*2 link_period_prev
  logical update
  update = .false.
  big_number = 2.0E32
  least_objective = 2.0E32
  tabu_rej_count = 0
  best_type = 0

c check the entire neighborhood
c but do it by road link to minimize # calcs required
  do link_to_change = 0,numroads !each link
    link_period_stands = ifuture
    link_period_prev = ifuture
    if(link_to_change.ne.0) then
      call get_l_p_prev(link_to_change,link_period_prev)
      call calc_retime_cost(link_to_change) !calc re-time costs
c process each stand that requires this link

c i points into the list of stands for this link
    do i = st_point(link_to_change), st_point(link_to_change+1)-1
        move_stand = st_list(i) ! stand to consider
    c calculate the min stand assignment for other stands
        call get_l_p_stands
        do move_period = 1, x(st_list(i))-1 ! all reductions
            if(stand_info(move_stand).volume(move_period).ne.0.) then
                link_period =
                > min0(move_period, link_period_stands, link_period_prev)
                call calc_obj_change5
                call get_aspiration2(asp)
                if(.not.asp) then ! If not aspirated
                    call check_tabu(tabu)
                else
                    tabu = .false.
                endif
            if (.not.tabu) then ! not tabu, check values
                if (move_infeasible .lt. least_objective) then
                    best_move_asp = asp
                    best_stand = move_stand
                    best_period = move_period
                    best_type = move_type
                    least_objective = move_infeasible
                    best_r_cost = r_cost(link_period)
                    best_link_period = link_period
                    best_link_period_stands = link_period_stands
                    m_best_loss_penalty = move_loss_penalty
                    m_best_dev_fr_total = move_dev_fr_total
                    do j = 1, num_periods
                        m_best_dev_fr_periodic(j) = move_dev_fr_periodic(j)
                    end do
                    m_best_harvest = move_harvest
                    m_best_dev_fr_maxopen = move_dev_fr_maxopen
                endif
                else ! tabu rejection
                    tabu_rej_count = tabu_rej_count + 1
                endif
            endif
        end do ! end reductions
    c begin increases in stand assignment
        do move_period = x(st_list(i))+1, ifuture ! all increases
            link_period = min0(link_period_prev, link_period_stands,
            > move_period)
            call calc_obj_change5
            call get_aspiration2(asp)
            if(.not.asp) then ! if not aspirated
                call check_tabu(tabu) ! move tabu?
            else
            endif
tabu = .false.
endif
if (.not.tabu) then !not tabu, check values
  if (move_infeasible .lt. least_objective) then
    best_move_asp = asp
    best_stand = move_stand
    best_period = move_period
    best_type = move_type
    least_objective = move_infeasible
    best_r_cost = r_cost(link_period)
    best_link_period = link_period
    best_link_period_stands = link_period_stands
    m_best_loss_penalty = move_loss_penalty
    m_best_dev_fr_total = move_dev_fr_total
    do j = 1, numperiods
      m_best_dev_fr_periodic(j) = move_dev_fr_periodic(j)
    end do
    m_best_harvest = move_harvest
    m_best_dev_fr_maxopen = move_dev_fr_maxopen
    endif
  else !tabu rejection
    tabu_rej_count = tabu_rej_count + 1
  endif
end do
end do
!next increase
end do
!next stand in list
end do
!next link
end return
end

---

**subroutine calc_obj_change5**

c this routine calculates the objective function change that would
occur if the move of stand, period was done
c objective function has four terms
c penalty due to lost volume due to harvest period
c penalty due to deviations from periodic harvest target
c penalty due to deviations from total harvest target
c cost of road building
c when this routine is called, the proposed move is in com_move.f
c existing assignments are in the stand info
  include 'params.f'
  include 'sol_param2.f'
  include 'com_data.f'
  include 'com_move.f'
  include 'road_com_sol.f'
  include 'com_growth.f'
real*4 volume, ass_penalty
logical test_move_feas
move_infeasible = 0.
do j=1,numperiods
  move_dev_fr_periodic(j)=cur_dev_fr_periodic(j)
end do

c calculate volume
  if(move_period.eq.ifuture)then
    move_type = 3
    move_harvest =
      stand_info(move_stand).volume(x(move_stand))
    move_dev_fr_periodic(x(move_stand))=
      ddim(dabs(hvol(x(move_stand)))
    -move_harvest-ptarget),
    periodic_all)
  move_dev_fr_total = ddim(dabs(cur_sol-move_harvest-htarget),
    total_all)
  move_loss_penalty = stand_info(move_stand).unass_penalty
  - stand_info(move_stand).ass_penalty(x(move_stand))
  else if (x(move_stand).eq.ifuture)then
    move_type = 1
    move_harvest = stand_info(move_stand).volume(move_period)
    move_dev_fr_periodic(move_period)=
      ddim(dabs(hvol(move_period))
    +move_harvest-ptarget),
    periodic_all)
  move_dev_fr_total = ddim(dabs(cur_sol+move_harvest-htarget),
    total_all)
  move_loss_penalty =
    stand_info(move_stand).ass_penalty(move_period) -
    stand_info(move_stand).unass_penalty
  else
    move_type = 2
    move_dev_fr_periodic(x(move_stand))=
      ddim(dabs(hvol(x(move_stand)))
    +stand_info(move_stand).volume(x(move_stand))-ptarget),
    periodic_all)
  move_dev_fr_periodic(move_period)=
    ddim(dabs(hvol(move_period))
  +stand_info(move_stand).volume(move_period)-ptarget),
    periodic_all)
  move_harvest =stand_info(move_stand).volume(move_period)-
    stand_info(move_stand).volume(x(move_stand))
  move_loss_penalty =
    stand_info(move_stand).ass_penalty(move_period)
  - stand_info(move_stand).ass_penalty(x(move_stand))
  move_dev_fr_total = ddim(dabs(cur_sol+move_harvest-htarget),
    total_all)
endif

call calc_road_cost
  call calc_dev_fr_maxopen(move_stand,move_period,move_type)
  move_dev_fr_maxopen=move_dev_fr_maxopen+cur_dev_fr_maxopen

c store the total penalty for periodic deviation and total deviation
move_loss_penalty = move_loss_penalty + cur_loss_penalty
move_infeasible = cur_beta * move_dev_fr_total +
> cur_gamma * move_dev_fr_maxopen
do j = 1, numperiods
  move_infeasible = move_infeasible + move_dev_fr_periodic(j) * cur_alpha
end do
move_infeasible = move_infeasible +
> move_loss_penalty + move_road_cost
move_feas = test_move_feas()
return
end

*** subroutine calc_ass_penalty(stand, period, ass_penalty) ***
c this routine calculates the objective function change that would
c occur if the move of stand, period was done
  include 'params.f'
  include 'sol_param2.f'
  include 'com_data.f'
  include 'road_com_sol.f'
  include 'com_growth.f'
  integer*2 stand, period
  integer*2 site
  real*4 ass_penalty
  integer*2 cur_ageclass
  real*4 cur_mai, max_mai, stock
  get the stand parameters
  site = stand_info(stand).site - 2 ! have to offset by 2 for the index
  cur_ageclass = min(0, 22, stand_info(stand).ageclass + period - 1)
  stock = stand_info(stand).stocking
  cur_mai = mai(site, cur_ageclass)
  max_mai = mai(site,
  > mai_max_age(site))
c calculate penalty to assign stand to harvest in this period
if (stand_info(stand).dead.eq.0) then ! not dead
  ass_penalty = abs(5.0 *
  > float(cur_ageclass)) ! current age
  > (max_mai - cur_mai) ! diff in mai
  > stand_info(stand).acres * stock ! stand area * stock
  else ! stand is dead
  ass_penalty = 5.0 * (float(period - 1))! # periods to wait
  > stand_info(stand).acres ! stand area
  > * max_mai * stock ! max loss
endif
return
end

*** subroutine calc_unass_penalty(unass_penalty) ***
  include 'params.f'
include 'sol_param2.f'
include 'com_data.f'
include 'com_move.f'
include 'road_com_sol.f'
include 'com_growth.f'
real*4 volume
integer*2 site
real*4 ass_penalty,unass_penalty
integer*2 cur_ageclass,age_at_end,age_before
real*4 cur_mai,max_mai,stock

C get the stand parameters
site = stand_info(move_stand).site - 2
! have to offset by 2 for the index
cur_ageclass = min(0,(22,stand_info(move_stand).ageclass)
stock = stand_info(move_stand).stocking
cur_mai = mai(site,cur_ageclass)
age_at_end = min(0
> ((stand_info(move_stand).ageclass+numperiods-1),22)
max_mai = mai(site,
> mai_max_age(site))
age_before = stand_info(move_stand).ageclass -1
if(stand_info(move_stand).dead.eq.0) then ! not dead
if(mai_max_age(site).le.age_at_end) then ! max in period or before
unass_penalty = 5.0*
> max(age_before,mai_max_age(site))*
> abs(max_mai - mai(site,age_before))* ! diff in mai
> stock*stand_info(move_stand).acres
! stock * area
else
unass_penalty = 0.0
endif
else ! stand is dead
unass_penalty = 5.0*max_mai*stand_info(move_stand).acres*numperiods*stock
endif
return
end

__subroutine calc_volume(stand,period,volume)___

include 'params.f'
include 'com_data.f'
include 'com_growth.f'
include 'road_com_sol.f'
integer*2 stand,period
real*4 volume
integer*2 ageclass
real*4 acres,cur_mai
if(stand_info(stand).for_non.eq.9960) then ! existing clearcut
volume = 0.
else if (stand_info(stand).dead.eq.1) then
    volume = stand_info(stand).ivol*stand_info(stand).acres
else
    ageclass = min(0,(stand_info(stand).ageclass+period-1))
    cur_mai=mai(stand_info(stand).site-2,ageclass)
    acres = stand_info(stand).acres
    volume = 5.0*float(ageclass)^
>    cur_mai*acres*stand_info(stand).stocking/1000.
end if
return
end

__subroutine calc_objective

include 'params.f'
include 'sol_param2.f'
include 'com_data.f'
include 'road_com.f'
include 'com_open.f'
include 'road_com_sol.f'
integer*2 i

c init to zero
    cur_loss_penalty = 0.0
    cur_sol = 0.0
    do j=1,numperiods
        cur_dev_fr_periodic(j)=0.0
        hvol(j) = 0.0
    end do
    cur_dev_fr_maxopen = 0.0
    cur_dev_fr_total = 0.0

c calculate volumes and loss penalty for this solution
    do i=1,numread
        if(x(i).gt.0)then
            if(x(i).eq.future)then
                cur_loss_penalty = cur_loss_penalty + stand_info(i).unass_penalty
            else
                cur_loss_penalty = cur_loss_penalty + stand_info(i).ass_penalty(x(i))
            endif
        endif
    end do

c subtract the unavoidable loss
    cur_loss_penalty = (cur_loss_penalty - min_loss_penalty)

c add up the volumes
    do i=1,numperiods
        cur_sol=cur_sol+hvol(i)
    end do

c calculate dev from maxopen
    i = 0_top
    do while (i.ne.0)
        cur_dev_fr_maxopen=cur_dev_fr_maxopen+
> dim(opens(i).acres,rmaxopen)
i=opens(i).pointer_fwd
end do

c calculate deviations from total and periodic
do i=1,numperiods
cur_dev_fr_periodic(i)= ddim(dabs(hvol(i)-ptarget),periodic_all)
end do
cur_dev_fr_total= ddim(dabs(cur_sol-htarget),total_all)
cur_infeasible = cur_loss_penalty+
> cur-road_cost+
> cur-dev_fr_total+
> cur-dev_fr_maxopen
do i=1,numperiods
cur_infeasible=cur_infeasible+cur_dev_fr_periodic(i)
end do
cur_feasible=cur_loss_penalty+cur_road_cost
return
end

subroutine calc_base_loss
include 'params.f'
include 'sol_param2.f'
include 'com_growth.f'
include 'com_data.f'
include 'road_com_sol.f'
integer*2 stand,ageclass,site,max_age,pread
real*4 ass_penalty
min_loss_penalty = 0.0
do stand=1,numread
   !for each stand
   if(x(stand).gt.0)then
      site = stand_info(stand).site - 2
      max_age = mai_max_age(site)
      if(max_age.lt.stand_info(stand).ageclass) then
         !before, put in period 1
         period = 1
         min_loss_penalty = min_loss_penalty +
      > stand_info(stand).ass_penalty(period)
      endif
   endif
   enddo
   return
end

subroutine calc_dev_fr_maxopen(stand,period,type)
c Calculate deviation from maxopen if move is done
include 'params.f'
include 'com_open.f'
include 'com_data.f'
include 'com_move.f'
include 'search_status.f'
include 'road_com_sol.f'
integer*2 stand,period,type
real*4 open_size
integer*2 open_list(20),start_period,j,i
logical found
integer numfound
real*4 dev_penalty
integer*2 next_period

calculate the penalty for of openings that would be created by adding this
stand in this period

c this is an incremental penalty, for this move only

c set penalty to zero

next_period = period + 1
move_dev_fr_maxopen = 0.0
if (type.eq.3) then
  call calc_rem_stand
    > (stand,stand_info(stand).opening(1),dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty
  call calc_rem_stand
    > (stand,stand_info(stand).opening(2),dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty
else if (type.eq.2) then
  swap move
  idir = period - x(stand)
if(idir.eq.1) then
  increase by one
  call calc_rem_stand
    > (stand,stand_info(stand).opening(1),dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty
  call calc_add_stand(stand,next_period,dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty
else if (idir.eq.-1) then
  call calc_rem_stand
    > (stand,stand_info(stand).opening(2),dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty
  call calc_add_stand(stand,period,dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty
else !non-consecutive moves
  call calc_rem_stand
    > (stand,stand_info(stand).opening(1),dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty
  call calc_rem_stand
    > (stand,stand_info(stand).opening(2),dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty
  call calc_add_stand(stand,period,dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty
  call calc_add_stand(stand,next_period,dev_penalty)
    move_dev_fr_maxopen = move_dev_fr_maxopen + dev_penalty

endif !swap move finished
else if (type .eq. 1) then !add move
call calc_add_stand(stand,period,dev_penalty)
move_dev_fr_maxopen=move_dev_fr_maxopen+dev_penalty
call calc_add_stand(stand,next_period,dev_penalty)
move_dev_fr_maxopen=move_dev_fr_maxopen+dev_penalty
else !bad move
stop
endif
move_dev_fr_maxopen=(move_dev_fr_maxopen)
return
end

 subroutine calc_add_stand(stand,period,dev_penalty)
 include 'params.f'
 include 'com_data.f'
 include 'com_open.f'
 include 'com_move.f'
 include 'search_status.f'
 integer*2 stand,period
 real*4 dev_penalty,open_size
 real*4 dev_now
 integer*2 open_list(20)
 integer*2 open_no
 logical found
 numfound=0
 dev_penalty=0.0
 dev_now = 0.0
 if(period .gt. numperiods) return
 open_size = (1.0 - 0.5*stand_info(stand).dead)*
 > stand_info(stand).acres
 do k=1,stand_info(stand).numadj !for each adjacent stand
 ilook=stand_info(stand).adj(k)
 do i=1,2
 open_no=stand_info(iloop).opening(i)
 if(open_no.ne.0) then
 if (opens(open_no).operiod.eq.period) then
 found = .false.
 do m=1,numfound
 if(open_list(m).eq.open_no) then
 found=.true.
 exit
 endif
 end do
 if(.not.found) then
 numfound=numfound+1
 open_list(numfound)=open_no
 open_size = open_size+opens(open_no).acres
dev_now = dev_now + dim(opens(open_no).acres,rmaxopen)
exit ! only one to find
endif
endif
endif
end do ! next opening number (i)
end do ! next adjacency
dev_penalty = dim(open_size,rmaxopen)-dev_now
return
end

---

**subroutine calc_new_open(stand,open_no,dev_value)**

include 'params.f'
include 'com_open.f'
include 'com_data.f'
integer*2 stand,open_no
real*4 dev_value,opensize
integer*2 numon,now,vlook,o_period,opening
integer*2 vert(maxvert),val(maxvert),o_adj(maxvert,maxadj)
integer*2 numadj(maxvert)
integer*2 adj_point(maxvert)
logical endchain
integer*2 vlist(maxvert),vadj,ephoint,jj,i,j,k
integer*2 rstack(maxvert),stack_end ! stack to make the recursion work

c this routine determines whether or not deletion of stand could split the

c opening and the resultant opening sizes

c returns value = 0.0 if resultant openings less than rmaxopen

c otherwise returns dev_value = rmaxopen - opening size - stand acres

c o_period points to the opening in stand_inf that is being considered

c The opening is currently larger than rmaxopen

dev_value = 0.0
c build the vertex list
numon = 0
do i=1,opens(open_no).numvert
   if(opens(open_no).v(i).ne.stand) then
      numon = numon+1
      vert(numon) = opens(open_no).v(i)
   endif
end do
c now, the appropriate stands are in array VERT

c make up an OPENING adjacency list for each vertex by intersecting
c VERT with the stand adjacencies.
c init vlist, val
   do ii =1,numon
      val(ii) = 0
   end do
c init adjacency pointers
   do ii=1,numon

adj_point(ii) = 1
numadj(ii)=0
end do

c build adjacency lists
do i=1,numon
numfound=0
numa=stand_info(vert(i)).numadj
do j=1,numa
do k=1,numon
if(stand_info(vert(i)).adj(j).eq.vert(k))then
   numadj(i)=numadj(i)+1
   o_adj(i,numadj(i))=k
   exit
endif
end do
end do
end do
endchain = .FALSE.
c start filling up openings, first available is o_avail
do  i=1,numon  !numon is the number of Xij=1
   now=0  !visit pointer
   do ii=1,numon
      vlist(ii)=0  ! now used to store pointers to found nodes
   end do
stack_end=0  !empty stack to begin
if (val(i).eq.0) then  ! node not visited
   now=now+1  !increment visit pointer
   val(i)=now  !store visit number
   vlist(now)=i  !store node pointer
   vlook = i
   stack_end=stack_end+1
rstack(stack_end)=i  !add node to recursion stack
end if
99 call findadj(vlook,o_adj,vadj,val,
   adj_point(vlook),numadj(vlook),endchain) !find a node adjacent to vlook
   if (.not.endchain) then found one
      now=now+1  !update visit pointer
      val(vadj)=now  !store visit number
      vlist(now)= vadj !store found node
      vlook = vadj !next vertex to search in chain
      if(adj_point(vadj).le.numadj(vadj))then
         stack_end=stack_end+1
         rstack (stack_end)=vadj !update stack
      endif
      goto 99
   else
end if
99

\textbf{c end of downward chain}
\textbf{c take entry off stack}
vlook = -1
\do jj=stack_end,1,-1
   if(adj_point(rstack(jj)).le.numadj(rstack(jj)))then
   end do
vlook = rstack(jj)
exit
dendif
dend do
if (vlook.ne.-1) then
goto 99
else ! thus, done for this connected component
c calculate the size of the opening
opensize = 0.0
do kk=1,now ! add the stands
    opensize=opensize+stand_info(vert(viist(kk))).acres * (1.0-0.5*stand_info(vert(viist(kk))).dead)
end do
dev_value = dev_value + dim(opensize,rmaxopen)
dendif
dendif
dendif
end do
! next connected component
return
dend

subroutine calc_rem_stand(stand,opening,dev_penalty)
include 'params.f'
include 'corn_open.f'
include 'corn_data.f'
integer*2 stand,opening
real*4 dev_penalty,dev_now,contrib
dev_penalty=0.0
if (opening.eq.0)return
dev_now = opens(opening).acres - rmaxopen
if(dev_now .le.0.0)return
contrib=dev_now -
> (1.0 - 0.5*stand_info(stand).dead)*
> stand_info(stand).acres
if(contrib.le. 0.0) then
    dev_penalty = -1*dev_now ! simple delete ok, return
else ! check opening structure
    call calc_new_open(stand,opening,contrib)
dev_penalty=contrib-dev_now
end if
return
dend

subroutine calc_opens(test dev fr maxopen)
include 'params.f'
include 'corn_open.f'
include 'com_data.f'
include 'road_com_sol.f'
real*4 test_dev_fr_maxopen
integer*2 op_no

test_dev_fr_maxopen = 0.0
op_no = 0_top
do while(op_no.ne.0)
  test_dev_fr_maxopen = test_dev_fr_maxopen +
>     dim(opens(op_no).acres,rmaxopen)
  op_no = opens(op_no).pointer_fwd
end do
return
end

__subroutine calc_feas_for_maxopen(stand,period,type) __
include 'params.f'
include 'com_open.f'
include 'com_data.f'
include 'com_move.f'
include 'search_status.f'
include 'road_com_sol.f'
integer*2 stand,period,type
real*4 open_size
integer*2 open_list(20),start_period,j,i
logical found
integer numfound
real*4 dev_penalty
integer*2 next_period

c This routine is to be used for the feasible moves only; assume all moves
c taken prior to this one are feasible.
c Calculate the opening size that would be created by adding this stand
c or swapping the stand
c If the move is to delete, return
c if (type.eq.3) return

c set variables
move_dev_fr_maxopen = 0.0
next_period = period + 1
if (type.eq.2) then    ; swap move
call calc_add_stand(stand,period,dev_penalty) ; check first type
if(trunc(dev_penalty).gt.0.0) then
  move_dev_fr_maxopen = dev_penalty
return
endif
if(period.lt.numperiods) then
  call calc_add_stand(stand,next_period,dev_penalty) ; check second type
  move_dev_fr_maxopen = dev_penalty
endif
else if (type .eq.1) then
    call calc_add_stand(stand,period,dev_penalty)!first type
    if(trunc(dev_penalty).gt.0.0) then
        move_dev_fr_maxopen = dev_penalty
        return
    endif
    endif
    return
end

__subroutine calc_r_cost_sngl(ilink,cost,l_period)__
include 'params.f'
include 'sol_param2.f'
include 'road_com.f'
include 'con_move.f'
real*4 cost
integer*2 ilink,l_period,imin,p_link,thelink
logical debug
imin = l_period
p_link = -1
default = false.
theilink = ilink

c first check sons of the link to change
    do j=1,roads(ilink).numsons
        if(roads(ilink).sons(j).ne.p_link)
            > imin = min(imin,roads(roads(ilink).sons(j)).period_assigned)
        end do
    endif
    if(imin.eq.roads(ilink).period_assigned) then
        cost = 0.0
        return
    else
        cost = (roads(ilink).lgth)*
        > (d_factor(imin)-d_factor(roads(ilink).period_assigned))
        p_link = ilink
        ilink = roads(ilink).dad_link
    do while ((imin.lt.roads(ilink).period).and.(ilink.ne.0))
    c first check sons of the link to change
    do j=1,roads(ilink).numsons
        if(roads(ilink).sons(j).ne.p_link)
            > imin = min(imin,roads(roads(ilink).sons(j)).period_assigned)
        end do
        imin = min(imin,roads(ilink).period) !and now min(sons,link)
        cost = cost + roads(ilink).lgth*
        > (d_factor(imin)-d_factor(roads(ilink).period_assigned))
        p_link = ilink
i_link = roads(i_link).dad_link
end do
endif
i_link = the_link
return
der end

__subroutine calc_road_cost
 include 'params.f'
 include 'sol_param2.f'
 include 'com_move.f'
 include 'road_com.f'
 include 'road_com_sol.f'
 include 'com_data.f'
 integer*2 link_to_change
if (stand_info(move_stand).link_req.eq.0) then
 move_road_cost = cur_road_cost
else
 move_road_cost = cur_road_cost + r_cost(link_period)
endif
return
der end

Move Implementation

__subroutine implement_best_move2
 include 'params.f'
 include 'search_status.f'
 include 'sol_param2.f'
 include 'best_move.f'
 include 'road_com_sol.f'
 include 'com_data.f'
 include 'com_move.f'
 real*4 volume
 call update_links
 c calc periodic harvests
 if (best_type.eq.1) then !add
 hvol(best_period) = hvol(best_period) + m_best_harvest
 cur_sol = cur_sol + m_best_harvest
 else if (best_type.eq.3) then !delete
 hvol(x(best_stand)) = > hvol(x(best_stand)) - m_best_harvest
 cur_sol = cur_sol - m_best_harvest
 else !swap move
 cur_sol = cur_sol + m_best_harvest
 hvol(best_period) = hvol(best_period) +
> stand_info(best_stand).volume(best_period)
  hvol(x(best_stand))=hvol(x(best_stand))- 
> stand_info(best_stand).volume(x(best_stand))
endif

c update_roads does current road cost
  cur_loss_penalty = m_best_loss_penalty
  cur_dev_fr_total = m_best_dev_fr_total
  cur_infeasible = least_objective
  cur_feasible = cur_loss_penalty + cur_road_cost
  do j = 1, numperiods
    cur_dev_fr_periodic(j)=m_best_dev_fr_periodic(j)
  end do
  cur_dev_fr_maxopen=m_best_dev_fr_maxopen
  x(best_stand)=best_period
  if(best_type .eq. 1) then                                       !add move
    call add_stand2(best_stand,best_period)
  else if (best_type .eq. 2) then                               !swap move
    call del_stand2(best_stand)
    call ADD_STAND2(best_stand,best_period)
  else if (best_type .eq.3) then                                !delete move
    call DEL_STAND2(best_stand)
  endif
  call tabu_add(best_stand)
  return
end

---

**subroutine implement_stand_move2**

  include 'params.f'
  include 'search_status.f'
  include 'sol_param2.f'
  include 'road_com_sol.f'
  include 'com_move.f'
  include 'com_data.f'
  real*4 volume

c this routine implements the move
  if(move_type.eq.1)then                                    !add
    hvol(move_period)=hvol(move_period)+move_harvest
    cur_sol = cur_sol + move_harvest
  else if (move_type.eq.3) then                            !delete
    hvol(x(move_stand))=
>  hvol(x(move_stand))-move_harvest
    cur_sol = cur_sol - move_harvest
  else                                                     !swap move
    hvol(move_period)=hvol(move_period)+
>  stand_info(move_stand).volume(move_period)
    cur_sol = cur_sol + stand_info(move_stand).volume(move_period)
    hvol(x(move_stand))=
>  hvol(x(move_stand))-
stand_info(x(move_stand)).volume(x(move_stand))
cur_sol = cur_sol -
> stand_info(x(move_stand)).volume(x(move_stand))
endif

c update the solution variables
if((move_type.eq.1).or.(move_type.eq.2)) then
  add or swap
  x(move_stand)=move_period
else if (move_type .eq.3) then
  delete
  x(move_stand)=ifuture
endif
cur_loss_penalty = move_loss_penalty
cur_dev_fr_total = move_dev_fr_total
do j= 1, numperiods
  cur_dev_fr_periodic(j)=move_dev_fr_periodic(j)
  end do
cur_dev_fr_maxopen = move_dev_fr_maxopen
cur_infeasible = move_infeasible
cur_feasible = cur_loss_penalty
if(move_type.eq.1) then
  add move
  call add_stand2(move_stand,move_period)
else if (move_type .eq. 2) then
  swap move
  call del_stand2(move_stand)
  call ADD_STAND2 (move_stand,move_period)
else if (move_type.eq.3) then
  delete move
  call DEL_STAND2 (move_stand)
endif
return
end

__ subroutine do_this_move2 __

include 'params.f'
include 'com_data.f'
include 'road_com.sol.f'
include 'com.move.f'
include 'com_growth.f'
include 'road_com.f'
include 'best_move.f'
c this routine calculates and implements the move in com_move.f
c move_stand,move_period,move_type
  logical asp,tabu
c variables for road links
  integer*2 link_to_change
  integer*2 link_period_prev
  link_to_change = stand_info(move_stand).link_req
  link_period_stands = ifuture
  link_period_prev = ifuture
  if(link_to_change.ne.0)then
    call get_l_p_prev(link_to_change,link_period_prev)!prev link period
call calc_retime_cost(link_to_change) !calc re-time costs
c calculate the min stand assignment for other stands
call get_l_p_stands
link_period = > min0(move_period,link_period_stands,link_period_prev)
endif
call calc_obj_change5
best_stand = move_stand
best_period = move_period
best_type = move_type
least_objective = move_infeasible
best_r_cost = r_cost(link_period)
best_link_period = link_period
best_link_period_stands=link_period_stands
m_best_loss_penalty = move_loss_penalty
m_best_dev_fr_total = move_dev_fr_total
do j=1,numperiods
  m_best_dev_fr_periodic(j)=move_dev_fr_periodic(j)
end do
m_best_harvest = move_harvest
m_best_dev_fr_maxopen=move_dev_fr_maxopen
call implement_best_move2
return
dend

--- subroutine update_roads ---
c this routine updates road for best_move.f
c called by implement_best_move
include 'params.f'
include 'road_com.f'
include 'com_data.f'
include 'road_com_sol.f'
include 'best_move.f'
include 'search_status.f'
include 'sol_param2.f'
integer*2 link_to_change,old_period
link_to_change = stand_info(best_stand).link_req
if (link_to_change.eq.0) return !nothing to do
roads(link_to_change).period= > min(best_link_period_stands,best_period)
if(best_period.ne.ifuture)then
roads(link_to_change).required_by_harvest=true.
else
roads(link_to_change).required_by_harvest = .false.
endif
call build_roads
return
dend
**subroutine re_time**

c this routine retimes the links in the path as defined by best_mvoe.f
    include 'params.f'
    include 'sol_param2.f'
    include 'road_com.f'
    include 'road_com_sol.f'
    include 'com_data.f'
    include 'best_move.f'

    integer*2 link_to_change
    integer*2 inmin,p_link,prev_path

    link_to_change = stand_info(best_stand).link_req
    if((best_link_period.lt.1).or.(best_link_period.gt.1future))then
        write(21,*)'error in link_period','best_link_period
        write(21,*)'best_stand,best_period,best_type,link_to_change
        write(21,*)'stand currently assigned to','x(best_stand)
        stop
    endif

    roads(link_to_change).period=
        > min(best_link_period_stands,best_period)
    roads(link_to_change).required_by_harvest = .true.

c if the same, return
    if (best_link_period .eq.
        > roads(link_to_change).period_assigned) return
    roads(link_to_change).period_assigned = best_link_period

   c and retime the path
    inmin = best_link_period

    c save this link
    p_link = link_to_change

    c get next link
    link_to_change = roads(link_to_change).dad_link
    do while(link_to_change.ne.0)

    c if there is another path adjoining at this link,
    c find min of other path
        call get_prev_path(link_to_change,p_link,prev_path)
        inmin = min(inmin,roads(link_to_change).period,prev_path) !min calculated
        roads(link_to_change).period_assigned = inmin !make change
        p_link = link_to_change
        link_to_change = roads(link_to_change).dad_link
    end do
    return

end
subroutine get_l_p_stands

include 'params.f'
include 'road_com_sol.f'
include 'road_com.f'
include 'com_move.f'
include 'com_data.f'
link_period_stands = ifuture
if (stand_info(move_stand).link_req.eq.0) return
do k = st_point(stand_info(move_stand).link_req),
> st_point(stand_info(move_stand).link_req+1)-1
if(st_list(k).ne.move_stand) then
  link_period_stands = min0(x(st_list(k)),link_period_stands)
endif
end do
return
end

subroutine get_l_p_prev(link_to_change,link_period_prev)

include 'params.f'
include 'road_com.f'
include 'com_move.f'
include 'com_data.f'
include 'road_com_sol.f'
integer*2 link_period_prev
integer*2 link_to_change
link_period_prev = ifuture
if(link_to_change.eq.0) return
do j = 1, roads(link_to_change).numsons
  link_period_prev = min0(link_period_prev,
> roads(roads(link_to_change).sons(j)).period_assigned)
end do
return
end

subroutine get_prev_path(link_to_change,p_link,path_period)

include 'params.f'
include 'road_com.f'
include 'com_move.f'
integer*2 p_link,path_period
integer*2 link_to_change
if(link_to_change.eq.0) return
path_period = ifuture
do j = 1, roads(link_to_change).numsons
if (roads(link_to_change).sons(j).ne.p_link)then
path_period = min0(path_period,
>   roads(roads(link_to_change).sons(j)).period_assigned)
endif
end do
return
end

subroutine get_all_links(ilink, link_list, link_len)
  include 'params.f'
  include 'road_com.f'
  integer*2 ilink,link_list(numroads),link_len
  integer*2 now,k,stack_end,stack(numroads)
  c simplified search since only backward links are included in sons
  c and the structure is a tree
  now=1
  stack_end=0
  link_list(now)=ilink
  10 continue
    do k=1,roads(ilink).numsons
       now=now+1
       link_list(now)=roads(ilink).sons(k)
       stack_end=stack_end+1
       stack(stack_end)=link_list(now)
    end do
    if(stack_end.ne.0)then
      ilink=stack(stack_end)
      stack_end=stack_end-1
      goto 10
    endif
  link_len=now
  return
end

subroutine stand_escape_routine
  include 'params.f'
  include 'sol_param2.f'
  include 'search_status.f'
  include 'road_com_sol.f'
  include 'com_data.f'
  include 'com_move.f'
  include 'road_com.f'
  include 'com_reactive.f'
  logical feas
num_esc_steps = min(60, int((1+rand())*moving_average))
do i=1,num_esc_steps
10     move_stand = int(rand() * numstands) + 1
     if (x(move_stand, eq, ifuture) then
         move_type = 1
         move_period = int(rand() * numperiods) + 1
     else if (x(move_stand, ne, 0) then
         move_period = ifuture
         move_type = 3
     else
         goto 10
     endif
     x(move_stand) = move_period
     if (move_type, eq, 1) then
         ! add move
         call add_stand2(move_stand, move_period)
     else
         call DEL_STAND2(move_stand)
     endif
call tabu_add(move_stand)
end do
call init_roads
do i=1,numread
    if ((x(i, ne, 0) .and. (stand_info(i).link_req.ne, 0)) then
        roads(stand_info(i).link_req).period = min0(
            roads(stand_info(i).link_req).period, x(i))
    endif
end do
do i=1,numroads
    if (roads(i).period.lt.ifuture)
        roads(i).required_by.harvest = .true.
    end do
call build_roads
cur_road_cost = cur_road_cost * r_factor
call calc_objective
call update_best
return
end
c update best solutions
  if (.not. move_feas) then
    linfeasible_move
    best_infeasible = dmin1(cur_infeasible, best_infeasible)
    moves_since_improve = moves_since_improve + 1
  else
    feasible_move
    if(cur_feasible.lt.best_feasible) then
      improving
      imp_time = move_count
      best_feasible = cur_feasible
      best_loss_penalty = cur_loss_penalty
      best_road_cost = cur_road_cost
      best_sol = cur_sol
      moves_since_improve = 0
    end if
    call save_sol
    else
      feasible_non-improving
      moves_since_improve = moves_since_improve + 1
    endif
  endif
  return
end

__routine rts_escape_routine2__
c This is the final routine called ESCAPE3
  include 'params.f'
  include 'sol_param2,f'
  include 'search_status.f'
  include 'road_com_sol.f'
  include 'com_reactive,f'
  include 'com_hash.f'
  include 'com_data.f'
  include 'road_com.f'
  logical done, escape
  escape=.true.
c calc number of links in road system
  num_links = 0
  do k=1,numroads
    if(roads(k).period_assigned.ne.future) num_links=num_links+1
  end do
  if ((float(num_links)/float(numroads)).gt. .10) then
    call new_escape_routine
  else
    call stand_escape_routine
    return
  endif
c check the final solution and add it to the hash_table
  done = .false.
  call hashfn(numread)
  do while (.not.done)
    if((hash_table(hash_add).eq.-1).or. !empty slot
      > (config_info(hash_table(hash_add))).sol_value -
> cur_feasible.lt.0.0001)) then
    done = .true.
    else
        hash_add = mod((hash_add+1),ttable_size)
        !linear probe
    endif
    end do
  c if it is a new configuration, add it
  c otherwise, do nothing
  if(hash_table(hash_add).eq.-1) then
      !existing configuration
      config_point=config_point+1
      if(config_point.gt.10000) then
          write(21,*) "*** error *** config_point > 10000'
          write(*,*) "*** error *** config_point > 10000'
          stop
      endif
      hash_table(hash_add)=config_point
      config_info(config_point).time_last_found = move_count
      config_info(config_point).sol_value = cur_feasible
  endif
  return
end

 subroutine choose_a_link(llink,iperiod)
c  Choose link from the leaves in the current road structure
  c and return the link id and the period to shift the link to.
    include 'params.f'
    include 'com_tabu2.f'
    include 'road_com.f'
    include 'search_status.f'
    integer*2 ll,ip,period,l_period
    real*4 cost,least_cost
    least_cost = 999999.0
    do l=1,numroads
      if((roads(l).required_by_harvest)
         > .and.(roads(l).nummons.eq.0)) then !link in road net and is a leaf
        if ((move_count-roads(l).time).gt. link_tenure) then !if not tabu
            l_period = roads(l).period_assigned + 1
            cost = calc_r_cost_snl(l,cost,l_period)
            if(cost.lt.least_cost) then
                least_cost=cost
                l = l
                ip = 1
            endif
        endif
      endif
    enddo
  return
end
__subroutine new_escape_routine

    include 'params.f'
    include 'search_status.f'
    include 'road_com_sol.f'
    include 'com_tabu2.f'
    include 'road_com.f'
    integer*2 ilink, l_period
    if (cur_road_cost.eq.0.0) then
        call stand_escape_routine
    else
        call choose_a_link(ilink, l_period)
        if (ilink.eq.-1) then
            write(21, *) 'no link found amongst leaves'
            message = 'no link found amongst leaves'
            at_end = .true.
        return
    endif
    do i = st_point(ilink), st_point(ilink+1)-1
        tabu_tenure = tabu_tenure + st_point(ilink+1)-st_point(ilink)
        x(st_list(i)).l.l_period = l_period
        if (l_period.eq. ifuture) then
            call del_stand2(st_list(i))
        else
            call del_stand2(st_list(i))
        call ADD_STAND2 (st_list(i), l_period)
        endif
        call tabu_add(st_list(i))
    enddo
    endif
    roads(ilink).period = l_period
    roads(ilink).time = move_count
    if (l_period.eq. ifuture) roads(ilink).required_by_harvest = .false.
    call build_roads
    call calc_objective
    return
end

subroutine link_escape

    c This routine deletes links until a large enough set of stands have been deleted
    include 'params.f'
    include 'sol_param2.f'
    include 'road_com.f'
    include 'com_data.f'
include 'road_com_sol.f'
include 'com_move.f'
include 'com_tabu2.f'
include 'search_status.f'
include 'com_reactive.f'
integer*2 link
integer*2 link_list(numroads)
integer*2 link_len
logical test_sol_feas
integer stand_count
num_esc_steps = min(60,int((1+rand())*moving_average)/5.0)
stand_count=0
num_tries=0
10 num_tries=num_tries+1
   ilink = int(rand()*numroads)+1
   if (roads(ilink).period.eq.ifuture) goto 10
   call get_all_links(ilink,link_list,link_len)
c now, re-assign all the stands that are involved to ifuture
c for each link in the list
do k = 1, link_len
   ilink=link_list(k)
   do i=st_point(ilink), st_point(ilink+1)-1
      if (x(st_list(i)).gt.0.and.x(st_list(i)).lt.ifuture) then
         stand_count=stand_count+1
         x(st_list(i))=ifuture
         call tabu_add(st_list(i))
      endif
   end do
   roads(ilink).period = ifuture ! assign the road link
   roads(ilink).required_by_harvest = .false.
end do
if not enough stands have been deleted, get another link
   if (stand_count.lt.num_esc_steps.and.num_tries.lt.10) goto 10
   call init_opens! re-do the openings
   call make_opens2
   call build_roads ! re-build the roads
cur_road_cost = cur_road_cost*r_factor
   call calc_objective ! calculate the objective
   move_feas = test_sol_feas()
call update_best
tabu_tenure = tabu_tenure + float(stand_count)lincrease tabu_tenure
return
end

Utility Routines and Functions

real*4 function rtrunc(real_number)
real*4 epsilon
real*4 real_number
parameter(epsilon=.01)
if (abs(real_number).lt.epsilon) then
  rtrunc=0.0
else
  rtrunc=real_number
endif
return
end function rtrunc

real*8 function dtrunc(dreal_number)
real*8 epsilon
real*8 dreal_number
parameter(epsilon=.01)
if (dabs(dreal_number).lt.epsilon) then
dtrunc=0.0
else
dtrunc=dreal_number
endif
return
end function dtrunc

* subroutine reinstall

  !reinstall the best solution found

  include 'params.f'
  include 'road_com_sol.f'
  include 'best_sol.f'
  include 'road_com.f'
  include 'com_data.f'
  call init_roads
do i=1,numread
    x(i)=x_save(i)
  if(x(i).gt.0.and.x(i).lt.future)then
    if (stand_info(i).link_req.ne.0) then
      roads(stand_info(i).link_req).period = min(x(i),
      > roads(stand_info(i).link_req).period)
      roads(stand_info(i).link_req).required_by_harvest = .true.
    endif
  endif
  end do
call build_roads
call init_opens
call make_opens2
call calc_objective
return
end

* subroutine write_sol(icount)
include 'params.f'
include 'road_com_sol.f'
include 'road_com.f'
include 'sol_param2.f'
include 'com_data.f'
include 'com_open.f'
integer icount
c use best solution
call reinstall
c write out parameters
  write(61,'(*)') 'r_factor = ',r_factor,'stream = ',icount
  write(71,'(*)') 'r_factor = ',r_factor,'stream = ',icount
  write(81,'(*)') 'r_factor = ',r_factor,'stream = ',icount
c solution now in current
c write out totals to each file
  write(61,800)cur_loss_penalty,cur_road_cost,cur_feasible,
    > hvol
  write(71,800)cur_loss_penalty,cur_road_cost,cur_feasible,
    > hvol
  write(81,800)cur_loss_penalty,cur_road_cost,cur_feasible,
    > hvol
800  format(10(f12.2,','))
c write harvest and openings
do i=1,numread
  write(81,1000)stand_info(i).stand_no,
    > x(i),stand_info(i).opening(1),
    > stand_info(i).opening(2)
end do
do i=1,numread
  write(61,1000)stand_info(i).stand_no,x(i)
end do
c write roads
do i=1,numroads
  write(71,1100) i,roads(i).period_assigned
end do
1000  format(4(i6,','))
1100  format(i4,',';i2)
return
end

subroutine write_datarts(icount)
  include 'params.f'
  include 'search_status.f'
  include 'road_com_sol.f'
  include 'com_reactive.f'
  include 'sol_param2.f'
  integer icount
  write(11,9000)r_factor,best_loss_penalty,
>best_road_cost,
>best_loss_penalty+best_road_cost),
>total_time,
>imp_time,
>move_count,icount,
>message
9000 format (5(f20.2,','),3(i6,','),a30)
return
end

subroutine save_init_sol

include 'params.f'
include 'road_com_sol.f'
include 'init_sol.f'
include 'com_data.f'
do i=1,numread
i_sol(i)=x(i)
end do
return
end

subroutine get_init_sol

include 'params.f'
include 'road_com_sol.f'
include 'init_sol.f'
include 'com_data.f'
include 'road_com.f'
call init_roads
do i=1,numread
x(i)=i_sol(i)
if(stand_info(i).link_req.ne.0)then
  if(x(i).gt.0)then
    roads(stand_info(i).link_req).period=min0(x(i),
    roads(stand_info(i).link_req).period)
  endif
endif
end do
call init_opens
call make_opens2
return
end
**subroutine init_best_values**

include 'params.f'
include 'road_com_sol.f'
best_feasible=999999999.
best_infeasible=999999999.
best_loss_penalty=999999999.
best_road_cost=999999999.
return
end

**subroutine write_iter**

include 'params.f'
include 'road_com_sol.f'
include 'search_status.f'
include 'com_move.f'
write(91,1000)move_count,move_feas,
  > cur_loss_penalty,cur_road_cost,
  >drunc(cur_dev_fr_total),
  >(drunc(cur_dev_fr_periodic(j)),j=1,numperiods),
  >best_feasible
1000  format(i5,','!i4,10((f12.2,','))
return
end

**subroutine init_move_counts2**

include 'params.f'
include 'search_status.f'
include 'sol_param2.f'
cur_alpha=upper_bound
cur_beta=upper_bound
cur_gamma=upper_bound
move_count = 1
moves_since_improve = 0
escape_time=0
message = 'No Message'
feas_count_total = 0
feas_count_per = 0
feas_count_open = 0
infeas_count_total = 0
infeas_count_per = 0
infeas_count_open = 0
return
end
__subroutine update_move_counters2

include 'params.f'
include 'sol_param2.f'
include 'search_status.f'
include 'com_move.f'
include 'road_com_sol.f'
move_count = move_count + 1
if(move_feas)then
  feas_count_per=feas_count_per+1
  feas_count_total=feas_count_total+1
  feas_count_open=feas_count_open+1
else
  if(dtrunc(cur_dev_fr_total).ne.0.0)then
    infeas_count_total=infeas_count_total+1
  else
    feas_count_total=feas_count_total+1
  endif
if(dtrunc(cur_dev_fr_maxopen).ne.0.0)then
  infeas_count_open=infeas_count_open+1
else
  feas_count_open=feas_count_open+1
endif
if(dtrunc(dabs(cur_infeasible - cur_beta*cur_dev_fr_total -
  cur_alpha*cur_dev_fr_maxopen -
  cur_loss_penalty - cur_road_cost)).ne.0.0)then
  feas_count_per=infeas_count_per+1
else !
  feas_count_per=feas_count_per+1
endif
endif
return
end

__subroutine update_obj_coeff2

include 'params.f'
include 'search_status.f'
include 'sol_param2.f'
if (infeas_count_total.eq.check interval )then !last 10 infeasible
  cur_beta = dmin1(upper_bound,cur_beta*2.0)
else if (feas_count_total.eq.check_interval) then !last 10 feasible
  cur_beta = dmax1(lower_bound,cur_beta/2.0)
endif
if (infeas_count_per.eq.check_interval )then !last 10 infeasible
  cur_alpha = dmin1(upper_bound,cur_alpha*2.0)
else if (feas_count_per.eq.check_interval) then !last 10 feasible

cur_alpha = dmax1(lower_bound,cur_alpha/2.0)
endif
if (infeas_count_open .eq. check_interval )then !last 10 infeasible
cur_gamma = dmin1(upper_bound,cur_gamma*2.0)
else if (feas_count_open.eq.check_interval) then !last 10 feasible
cur_gamma = dmax1(lower_bound,cur_gamma/2.0)
endif
feas_count_total=0
feas_count_per=0
feas_count_open=0
infeas_count_total=0
infeas_count_per=0
infeas_count_open=0
return
der

__subroutine init_x
include 'params.f'
include 'road_com_sol.f'
do i=1,numstands
x(i)=ifuture
end do
return
der

__subroutine init_stands
include 'params.f'
include 'com_data.f'
do i=1,numstands
stand_info(i).stand_no=0
do j=1,2
stand_info(i).opening(j)=0
end do
stand_info(i).dead=0
stand_info(i).for_non=0
stand_info(i).ageclass=0
stand_info(i).site=0
stand_info(i).numadj=0
stand_info(i).cover_type=0
do j=1,maxvert
stand_info(i).adj(j)=0
end do
stand_info(i).acres=0.0
stand_info(i).stocking=0.0
stand_info(i).ivol=0.0
stand_info(i).unass_penalty = 0.0
subroutine init_opens

include 'params.f'
include 'com_open.f'
include 'com_data.f'
c this routine initializes the opening structures and
c clears all opening numbers from stand info.
do i=1,numread
   stand_info(i).opening(2)=0
   stand_info(i).opening(1)=0
end do
c clear the openings
   o_avail=1
   o_last=0
   o_top=1
do j=1,maxvert
   opens(i).v(k)=0
   opens(i).epointer(k)=0
   opens(i).e(k)=0
end do
doi=1,num_opens-1
   av_list(i)=i+1
end do
av_list(num_opens)=0
return
end

subroutine get_data(dataset)

include 'params.f'
include 'com_data.f'
include 'road_com_sol.f'
include 'com_move.f'
character*12 dataset
integer*2 maxstand
real*4 tot_vol
integer*2 i,j,k,
> INDX(230000),ADJ(NUMSTANDS,MAXADJ),numadj
real*4 tot_acres
character*16 species
integer cover_type,for_non,ios
real*4 unass_penalty
character*8 mapin
integer*2 stand_list(0:numroads,300)
integer*2 link_point(0:numroads)
logical good_one
mapin = 'shulkell'
do i=0,numroads
   link_point(i)=0
do j=1,300
   stand_list(i,j)=0
end do
tot_acres=0.0
tot_vol=0.0
maxstand=0
numread=0
do i=1,23000
   indx(i)=0
end do
do i=1,numstands
   do j=1,maxadj
      adj(i,j)=0
   end do
write(*,'(a)') 'reading road data ...'
call read_road
write(*,'(a)') 'opening file',dataset
open(unit=20,file=dataset)
istart = 1
do 200, =1,numstands
   read (20,io,=100,iostat=ios) stand_info(i).stand_no,
   > stand_info(i).ACRES,
   > stand_info(i).ageclass,
   > stand_info(i).site,
   > stand_info(i).stocking,
   > stand_info(i).ivol,
   > stand_info(i).cover_type,
   > stand_info(i).for_non,
   > stand_info(i).link_req
   format(i5,f10.2,i4,i4,f6.2,f12.6,i3,i5,i5)
  INDX(STAND_info(i).stand_no)=1
tot_acres=tot_acres+stand_info(i).acres
tot_vol=tot_vol+stand_info(i).ivol*stand_info(i).acres
if((stand_info(i).for_non.ge.9907).and.
> (stand_info(i).for_non.le.9909))then
stand_info(i).dead=1
else
   stand_info(i).dead=0
endif
if(stand_info(i).for_non.eq.9960) then !existing clearcut
   x(i)=0
   stand_info(i).unass_penalty = 0.0
else
   x(i)=ifuture !unassigned
endc

calculate unass penalty and volumes for the planning periods
c and determine whether to add to st_list
   good_one = .false.
move_stand = i
   call calc_unass_penalty(unass_penalty)
   stand_info(i).unass_penalty = unass_penalty
   do k=1,numperiods
      call calc_volume(i,k,stand_info(i).volume(k))
      stand_info(i).volume(k)=trunc(stand_info(i).volume(k))
      if(stand_info(i).volume(k).gt.0.)good_one=.true.
   end do
   call calc_ass_penalty(i,k,stand_info(i).ass_penalty(k))
   end if
   if(good_one)then
      link_point(stand_info(i).link_req)=
         > link_point(stand_info(i).link_req)+1
      stand_list(stand_info(i).link_req).
         > link_point(stand_info(i).link_req))=i
   endif
endc
200 continue
100 if(ios.eq.-1) then
   numread=i-1
else
   numread=numstands
endif
close(20)
write(*,'(a)') 'number of stands read',numread
maxstand=stand_info(numread).stand_no !since they are in order
c make up the stand to road lists
   call make_st_list(link_point,stand_list)
c Read the adjacency file and update stand adjacency list
   open (unit=10, file='myhome3/richarew/thesis/data/!mapin/
      > /mapin/*.adj')
do while (iSTAND.le.MAXstand)
   read (10,*,end=300) inum,istand,(ADJ(indx(istand),J),j=1,INUM)
   numadj=0 !pointer to adjacency list members
   do k=1,inum
      if (adj(indx(istand),k).le.maxstand) then
         numadj=numadj+1
   end do
   link_point(istand).link_req=
      > link_point(istand).link_req)+1
   stand_list(istand).link_req=istand
      > stand_list(istand).link_req)
   end do
endc
subroutine calc_parameters

include 'params.f'
include 'sol_param2.f'
real*4 periodic_per,total_per

set cost to $5000/km (road lengths stored in meers)
discount rate is .10
deriod is 5 years

d_factor(1)=.7879*5.
d_factor(2)=.4893*5.
d_factor(3)=.3038*5.
d_factor(4)=.1886*5.
d_factor(5)=0.0

ptarget = 500.0  !!!thousands of cubic feet
htarget = 2000.0  !!!thousands of cubic feet
periodic_per = 0.05
total_per = 0.02
periodic_all = periodic_per*ptarget
total_all = total_per*htarget
r_factor = 1.0
return
end

subroutine init_move_values

include 'params.f'
include 'sol_param2.f'
include 'com_move.f'
move_road_cost = 0.0
move_feas = .false.
return
end

subroutine init_search_params

include 'params.f'
include 'search_status.f'
c call this routine after data file has been read so that
c numread is known
    maxiters = 5000
    check_interval = 10
    return
end

subroutine init_objective
    include 'params.f'
    include 'sol_param2.f'
    include 'com_data.f'
    include 'road_com_sol.f'
    cur_sol = 0.0      !no harvest
    cur_infeasible = 0.0  !
    cur_feasible = 0.0    !
    cur_loss_penalty = 0.0   !none yet
    cur_road_cost = 0.0
    cur_dev_fr_maxopen=0.0
    c since there is no volume at this point
    do j=1,numperiods
        cur_dev_fr_periodic(j)=ptarget-periodic_all
        cur_infeasible=cur_infeasible+cur_dev_fr_periodic(j)
    end do
    cur_dev_fr_total=htarget-total_all
    cur_infeasible=cur_beta*cur_infeasible+cur_alpha*cur_dev_fr_total
    c calculate the unassigned penalty for all stands
    do stand=1,numread
        cur_loss_penalty=cur_loss_penalty+stand_info(stand).unass_penalty
    end do
    c update the total penalties
    cur_infeasible = cur_infeasible+cur_loss_penalty
    c reduce the loss penalty by the unavoidable penalty
    call calc_base_loss
    cur_loss_penalty = cur_loss_penalty - min_loss_penalty
    return
end

subroutine save_sol
    include 'params.f'
    include 'road_com_sol.f'
    include 'com_data.f'
    include 'search_status.f'
    include 'best_sol.f'
    include 'road_com.f'
    do i=1,numread
        x_save(i)=x(i)
end do
  do i = 1,numroads
    road_save(i) = roads(i).period_assigned
  end do
  return
end

subroutine clear_solution
  include 'params.f'
  include 'com_data.f'
  include 'road_com_sol.f'
  do i=1,numread
    if (x(i).ne.0)then
      x(i)=future
    endif
  end do
  cur_sol = 0.0
  do j=1,numperiods
    hvol(j) = 0.0
  end do
  call init_roads
  return
end

subroutine time_stats(elapsed)
  include 'search_status.f'
  real*4 elapsed
  elapsed = etime_(etime_struct)
  return
end

subroutine read_growth
  include 'com_growth.f'
  open (unit=51, file='~/home3/richarew/thesis/data/mai.dat')
  do isite = 1,6 !corresponds to index 3-8
    read(51, '(i4)') mai_max_age(isite)
    read(51,100) (mai(isite,iage),iage = 1,22) !22 age classes
  end do
100  format (22f7.2)
  close(unit=51)
  return
end
logical function test_sol_feas()
  include 'params.f'
  include 'road_com_sol.f'
  test_sol_feas= .false.
  if(dtrunc(cur_dev_fr_maxopen).ne.0.0) then
    return
  else if(dtrunc(cur_dev_fr_total).ne.0.0) then
    return
  else
    do j=1,numperiods
      if(dtrunc(cur_dev_fr_periodic(j)).ne.0.0) return
    end do
  endif
  test_sol_feas= .true.
  return
end function

logical function test_move_feas()
  include 'params.f'
  include 'com_move.f'
  test_move_feas= .false.
  if(dtrunc(move_dev_fr_maxopen).ne.0.0) then
    return
  else if(dtrunc(move_dev_fr_total).ne.0.0) then
    return
  else
    do j=1,numperiods
      if(dtrunc(move_dev_fr_periodic(j)).ne.0.0) return
    end do
  endif
  test_move_feas= .true.
  return
end function

Road Subroutines

__subroutine init_roads
  include 'params.f'
  include 'road_com.f'
  do i=1, numroads
    roads(i).period = ifuture
    roads(i).required_by_harvest = .false.
    roads(i).required_by_connect = .false.
    roads(i).time = -32766
  enddo
end do
return
end

subroutine build_roads
include 'params.f'
include 'sol_param2.f'
include 'road_com.f'
include 'road_com_sol.f'
include 'search_status.f'
include 'best_move.f'
integer*2 i,j,k,t,imin     !local variables
integer*2 ilink
real*4 new_pr             !used to put closest node or
real*4 cost_to_add(2)     ! to calc min cost path to add link in cycle.
integer*2 knode(2)
do =1,numnodes+1           !init the val array
val(i) = 9999.0            !indicates whether nodes have been parsed
dad(i) = 0
link(i) = -1
end do

call const_road_heap      !init the heap
do =1,numroads            !clear the previous solution
roads(i).period_assigned = ifuture
roads(i).nhops=0
roads(i).dad_link=-1
roads(i).numsons = 0
end do

! compute the shortest path network for the roads
! val(t) stores the length of the shortest path from t to the root
! priority(t) is used to arrange the heap
k = inv(numnodes+1)       !remove the root node
priority(k) = 0.0         !root of tree
val(k)=0.0
link(numnodes+1)=0         !k will point to the element removed
10 do ipoint = adj_point(k),adj_point(k+1)-1
   t = adj(ipoint,1)      !adjacent node
   ilink = adj(ipoint,2)   !link #
   if(inv(t,ie.heap_size)then !still on heap
      if(roads(ilink).required_by_harvest)then !required
         new_pr=0.0
      else
         new_pr= priority(k) + roads(ilink).lgh     !distance to k
      endif
      if(priority(t).gt.new_pr)then         !shorter?
         call pqchange(t,new_pr)      !update priority
      endif
   else
      !logically visits
   endif
end do

val(t)=val(k)+roads(ilink).lgt
    ! update length of path
dad(t)=k
    ! father
link(t)=ilink
    ! simpler
roads(ilink).dad_link = link(k)
roads(ilink).period_assigned=roads(ilink).period
    ! assign period
roads(ilink).required_by_connect=not.roads(ilink).required_by_harvest
    ! true if a steiner link
endif
endif
end do
    ! next adjacent node
if(heap_size.gt.0.)then
    ! not finished
k=1
    ! get top element
goto 10
    ! and continue
else
endif
c fix up the period required
do i=1,numnodes
    ! all but root node
if(val(i).ne.9999.0.)then
    ! lin, parse the path
j = i
    ! save node index
imin = roads(link(j)).period_assigned
    ! period for the last link
do while (dad(j).ne.0.)
    ! parse all connections
j = dad(j)
    ! father
imin = min(imin,roads(link(j)).period_assigned)
    ! min period required
roads(link(j)).period_assigned=imin
    ! update
end do
endif
di end do
    ! next node
c find any missing links
do i=1,numroads
if((roads(i).dad_link.eq.-1))
    > .and.(roads(i).required_by_harvest))then
roads(i).period_assigned=roads(i).period
    ! assign it to its period
knode(1)=roads(i).x
knode(2)=roads(i).y
    ! two directions
do ii=1,2
    ! find the cheapest path
if ( roads(ii).period .lt.
    > roads(link(knode(ii))).period_assigned)then
    call calc_r_cost_sngl(link(knode(ii)),
    > cost_to_add(ii),roads(i).period Assigned)
else
    cost_to_add(ii)=val(knode(ii))
endif
end do
costdiff = cost_to_add(1)-cost_to_add(2)
if ( costdiff.lt.0.) then
    k = 1
else if (costdiff.gt.0.)then
    k=2
else ! tie
    rval=rand()
k=int(rval)+1
endif
    k = knode(k) !finally got it!
c path chosen, implement
roads(i).dad_link = link(k) ! Assign dad link
imin = roads(i).period_assigned
k = dad(k)
do while( (k.ne.0).and.(imin.lt.roads(link(k)).period_assigned) )
    roads(link(k)).period_assigned = imin
    k=dad(k)
    imin = min(roads(link(k)).period_assigned,imin)
end do
endif
enddo
cur_road_cost = 0.0
!
ext link
do i=1,numroads
if(roads(i).period_assigned.ne.ifuture)then
    cur_road_cost = cur_road_cost +
    roads(i).lgth * d_factor(roads(i).period_assigned)
endif
enddo
cur_road_cost = cur_road_cost*r_factor
c find and store the sons for each link
do i = 1, numroads
    k = roads(i).dad_link
    if( (k.ne.0) ) then
        roads(k).numsons = roads(k).numsons + 1
        roads(k).sons(roads(k).numsons)=i
    endif
end do
return
end

---

subroutine make_st_list(link_point,stand_list)
include 'params.f'
include 'road_com.f'
include 'corn_data.f'
integer*2 link_point(0:numroads)
integer*2 stand_list(0:numroads,300)
integer*2 ipoint
ipoint=0
st_point(0)=1
do i=0,numroads
    do j=1,link_point(i)
        st_list(ipoint+j)=stand_list(i,j)
    end do
    ipoint=link_point(i)+ipoint
st_point(i+1)=ipoint+1
end do
return
end

subroutine read_road

include 'params.f'
include 'road_com.f'
integer*2 adjlist(numnodes+1,300,2),adj_pt(numnodes+1)
do i=1,numnodes+1
  !init the adjacency lists
  adj_pt(i)=0
end do
open (unit=10,>
  file=:'/home3/richarew/thesis/data/shulkell/roads.txt')!open road data file
10 read(10,*,end=99)i,roads(i),x,roads(i).y,roads(i).length
  adj_pt(roads(i).x)=adj_pt(roads(i).x)+1
  adj_pt(roads(i).y)=adj_pt(roads(i).y)+1
  adjlist(i,1,1)=roads(i).x
  adjlist(i,1,2)=roads(i).y
  adjlist(i,2,1)=roads(i).x
  adjlist(i,2,2)=roads(i).y
  roads(i).period = ifuture
  roads(i).required_by_harvest = .false.
  roads(i).required_by_connect = .false.
  roads(i).period_assigned = ifuture
i=i+1
goto 10
99 close(unit=10)
c build the adjacency list (pack it)
  istart = 0
  do i=1,numnodes+1
    adj_point(i)=istart+1
    do j=1,adj_pt(i)
      adj(istart+j,1)=adjlist(i,j,1)
      adj(istart+j,2)=adjlist(i,j,2)
    end do
    istart=adj_pt(i)+istart
  end do
  adj_point(numnodes+2)=adj_point(numnodes+1)+adj_pt(numnodes+1)
return
end

subroutine const_road_heap
c Since all priorities are the same, simply construct the array
  include 'params.f'
include 'road_com.f'
integer*2 i
heap_size = numnodes+1
do i=0,numnodes+1
road_heap(i)=i
inv(i)=i
priority(i)=32767.0
end do
priority(numnodes+1)=0.0 !ROOT NODE represents mainroad access
priority(0)=-10 !sentinel value for node 0
return
dend

subroutine pqdownheap(k)
include 'params.f'
integer*2 k,j
integer*2 vert
include 'road_com.f'
vert=road_heap(k)
do while (k.le.heap_size/2)
j=k+k
if(j.lt.heap_size)then
   if(priority(road_heap(j)).gt.priority(road_heap(j+1))) j=j+1
endif
if(priority(vert).le.priority(road_heap(j))) goto 10
road_heap(k)=road_heap(j)
inv(road_heap(j))=k
k=j
end do
10 road_heap(k)=vert
inv(vert)=k
return
end

subroutine pqremove(k)
c remove the kth element of the heap
include 'params.f'
integer*2 k,ksave
include 'road_com.f'
ksave=road_heap(k)
inv(road_heap(k))=heap_size
road_heap(k)=road_heap(heap_size)
inv(road_heap(heap_size))=k
heap_size=heap_size-1
call pqdownheap(k)
k=ksave
return
end

___subroutine pqchange(t,new_pr)
c change element t to have priority new_pr
   include 'params.f'
   integer*4 t,k
   real*4 new_pr,old_pr
   include 'road_com.f'
   old_pr=priority(t)
   priority(t)=new_pr
   k = inv(t)
   if(new_pr.lt.old_pr)then
      call pqupheap(k)
   else
      call pqdownheap(k)
   endif
return
end

___subroutine pqupheap(k)

   include 'params.f'
   integer*4 vert,k
   include 'road_com.f'
   vert=road_heap(k)
do while (priority(road_heap(int(k/2))).ge.priority(vert))
   road_heap(k)=road_heap(int(k/2))
   inv(road_heap(k))=k
   k=int(k/2)
   road_heap(k)=vert
   inv(road_heap(k))=k
end do
return
end

___subroutine calc_retime_cost(link_to_change)

   include 'params.f'
   include 'sol_param2.f'
   include 'road_com.f'
   include 'com_move.f'
integer*2 link_to_change,ilink,l_period,imin,p_link,path_period
integer*2 knode(2)
real*4 cost_to_add(2)
calculate the re-timing costs for link_to_change, each l_period
calculate the re-timing of the link_to_change
if ((link_to_change.le.0).or.(link_to_change.gt.numroads)) then
  write("**") 'error in link ',link_to_change
  stop
endif
do l_period = 1, ifuture
  check to see if the link is in a path
  if (roads(link_to_change),dadj,dadj,eq,-1) then !link is not in a path
    r_cost(l_period)=roads(link_to_change),lgth*
    d_factor(l_period)=cost_to_build_new_link
    knode(1)=roads(link_to_change),x
    knode(2)=roads(link_to_change),y
    do kk=1,2
      k=knode(kk)
      ilink = link(k)
      if (roads(link_to_change),period_assigned,l,l_period) then
        call calc_r_cost_sngl(ilink,cost_to_add(kk),l_period)
      else
        cost_to_add(kk)=0.0
      endif
    end do
    r_cost(l_period)=r_factor*(r_cost(l_period)+
    > min(cost_to_add(1),cost_to_add(2)))
  else ! just do it
    ilink = link_to_change
    call calc_r_cost_sngl(ilink,cost_to_add(1),l_period)
    r_cost(l_period)=cost_to_add(1)*r_factor
  endif
end do
return

end

RTS Subroutines

subroutine check_for_reps(escape)
  include 'params.f'
  include 'road_com_sol.f'
  include 'com_hash.f'
  include 'com_reactive.f'
  include 'search_status.f'
  include 'com_tabu2.f'
  include 'com_data.f'
logical escape, done
integer*2 length
escape = .false.
done = .false.
call hashfn(numread)
do while (.not.done)
  if((hash_table(hash_add).eq.-1).or. !empty slot
     > (config_info(hash_table(hash_add)).sol_value -
     > cur_feasible.lt. 0.0001)) then !match
     done = .true.
  else
     hash_add = mod((hash_add+1),itable_size) !linear probe
  endif
  end do
if(hash_table(hash_add).ne.-1) then !existing configuration
  length = move_count - 
  > config_info(hash_table(hash_add)).time_last_found
  config_info(hash_table(hash_add)).num_rep = 
  > config_info(hash_table(hash_add)).num_rep + 1
  config_info(hash_table(hash_add)).time_last_found = 
  > move_count
  if (config_info(hash_table(hash_add)).num_rep
  > .gt. rep)then
    chaotic=chaotic+1
  if(chaotic.gt.chaos)then
    chaotic=0
    escape = .true.
    return
  endif
  endif
if (length .lt. cycle_max) then
  if(moving_average .eq. 0.0)then
    moving_average = length
  else
    moving_average = 0.1*length + 0.9*moving_average
  endif
  tabu_tenure =min(600.0,increase*tabu_tenure)
  steps_since_size_change = 0
endif
else !new configuration, instal it
  config_point=config_point+1
if(config_point.gt.10000)then
  write(21,*!) "*** error *** config_point > 10000"
  message='config_point > 10000'
  at_end = .true.
endif
hash_table(hash_add)=config_point
config_info(config_point).time_last_found = move_count
config_info(config_point).sol_value = cur_feasible
endif
if (steps_since_size_change .gt. moving_average) then
  tabu_tenure = max(tabu_tenure*decrease, 1.0)
  steps_since_size_change = 0
endif
return
end

subroutine init_reactive
  include 'params.f'
  include 'com_reactive.f'
  chaotic = 0
  chaos = 3
  rep = 3
  cycle_max = 50
  increase = 1.1
  decrease = 0.9
  return
end

subroutine del_one_link
  include 'params.f'
  include 'sol_param2.f'
  include 'road_com.f'
  include 'corn_data.f'
  include 'road_com_sol.f'
  include 'com_move.f'
  include 'com_tabu2.f'
  include 'search_status.f'
  include 'com_reactive.f'
  integer*2 ilink
  integer*2 link_list(numroads)
  integer*2 link_len
  logical test_sol_feas
  integer stand_count
  num_esc_steps = 1
  stand_count = 0
  num_tries = 0
  10  ilink = int(rand() * numroads) + 1
      num_tries = num_tries + 1
      if (roads(ilink).period.ne. ifuture) then
          call get_all_links(ilink, link_list, link_len)
      endif
  endif
  now, re-assign all the stands that are involved to ifuture
  do k = 1, link_len
      ilink = link_list(k)
      do i = st_point(ilink), st_point(ilink + 1) - 1
        if (x(st_list(i)).gt. 0. and. x(st_list(i)).lt. ifuture) then
          if (x(st_list(i)).gt. 0. and. x(st_list(i)).lt. ifuture) then
            (i.e., some code here)
        endif
      enddo
    enddo
  enddo
  return
end
stand_count=stand_count+1
x(st_list(i))=ifuture
call tabu_add(st_list(i))
endif
c end

c assign the road link
roads(ilink).period = ifuture
roads(ilink).required_by_harvest = .false.
end do
cendif

c re-do the openings
call init_opens
call make_opens2

c re-build the roads
call build_roads
cur_road_cost=cur_road_cost*r_factor
calculate the objective
call calc_objective
c update best
move_feas = test_sol_feas()
call update_best

c now set tabu_tenure to be equal to tabu_tenure plus number of stands deleted
tabu_tenure = tabu_tenure+ float(stand_count)
return
cend

Tabu Subroutines

__subroutine init_tabu_rts

include 'params.f'
include 'com_tabu2.f'
include 'com_reactive.f'
tabu_point = 0  !points to tabu list
do =1,tabu_len
tabu_list(i)= -20000.0  !zero tabu list
end do
steps_since_size_change = 0
tabu_tenure = 1.0
moving_average=0.0
return
cend

__subroutine get_aspiration2(asp)
include 'params.f'
include 'sol_param2.f'
include 'com_move.f'
include 'road_com_sol.f'
logical asp, tabu
asp = .false.

! check feasibility
if (move_feas) then  ! feasible
  if((move_loss_penalty+move_road_cost).lt.
    (best_feasible))then
    asp = .true.
  endif
else                 ! infeasible
endif
return
end

------------------------------------------------------------------------
subroutine check_tabu(tabu) ! move tabu?
------------------------------------------------------------------------
include 'params.f'
include 'com_tabu2.f'
include 'com_move.f'
include 'search_status.f'
logical tabu
if((tabu_list(move_stand).ge.(move_count-tabu_tenure))then
  tabu = .true.
else
  tabu = .false.
endif
return
end

------------------------------------------------------------------------
subroutine tabu_add(stand)
------------------------------------------------------------------------
include 'params.f'
include 'com_tabu2.f'
include 'search_status.f'
integer*2 stand
tabu_list(stand)=move_count
return
end
APPENDIX D

NUMERICAL RESULTS
<table>
<thead>
<tr>
<th>$\rho$</th>
<th>Average</th>
<th>Best</th>
<th>Worst</th>
<th>Av. # Moves</th>
<th>Std. Dev</th>
<th>Coeff. Var</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>310,945</td>
<td>292,843</td>
<td>372,456</td>
<td>6,678</td>
<td>15,345</td>
<td>4.94%</td>
</tr>
<tr>
<td>0.20</td>
<td>326,495</td>
<td>307,979</td>
<td>385,927</td>
<td>6,978</td>
<td>15,055</td>
<td>4.61%</td>
</tr>
<tr>
<td>0.30</td>
<td>336,019</td>
<td>325,392</td>
<td>361,142</td>
<td>6,464</td>
<td>8,245</td>
<td>2.45%</td>
</tr>
<tr>
<td>0.40</td>
<td>349,632</td>
<td>337,224</td>
<td>373,660</td>
<td>7,233</td>
<td>8,786</td>
<td>2.51%</td>
</tr>
<tr>
<td>0.50</td>
<td>360,851</td>
<td>352,301</td>
<td>374,346</td>
<td>6,793</td>
<td>6,125</td>
<td>1.70%</td>
</tr>
<tr>
<td>0.60</td>
<td>374,729</td>
<td>361,901</td>
<td>388,906</td>
<td>6,758</td>
<td>7,188</td>
<td>1.92%</td>
</tr>
<tr>
<td>0.70</td>
<td>385,847</td>
<td>373,681</td>
<td>410,644</td>
<td>6,881</td>
<td>8,551</td>
<td>2.22%</td>
</tr>
<tr>
<td>0.80</td>
<td>398,380</td>
<td>386,656</td>
<td>410,775</td>
<td>7,191</td>
<td>5,914</td>
<td>1.48%</td>
</tr>
<tr>
<td>0.90</td>
<td>409,719</td>
<td>399,135</td>
<td>436,416</td>
<td>6,451</td>
<td>9,781</td>
<td>2.39%</td>
</tr>
<tr>
<td>1.00</td>
<td>415,975</td>
<td>406,966</td>
<td>423,622</td>
<td>6,804</td>
<td>4,847</td>
<td>1.17%</td>
</tr>
<tr>
<td>2.00</td>
<td>498,386</td>
<td>488,556</td>
<td>508,032</td>
<td>5,214</td>
<td>4,585</td>
<td>0.92%</td>
</tr>
<tr>
<td>3.00</td>
<td>563,860</td>
<td>556,021</td>
<td>572,178</td>
<td>4,771</td>
<td>4,400</td>
<td>0.78%</td>
</tr>
<tr>
<td>4.00</td>
<td>624,297</td>
<td>606,125</td>
<td>646,247</td>
<td>5,053</td>
<td>7,828</td>
<td>1.25%</td>
</tr>
<tr>
<td>5.00</td>
<td>664,834</td>
<td>655,268</td>
<td>686,107</td>
<td>4,048</td>
<td>7,527</td>
<td>1.13%</td>
</tr>
<tr>
<td>6.00</td>
<td>715,875</td>
<td>695,961</td>
<td>743,025</td>
<td>6,055</td>
<td>8,583</td>
<td>1.20%</td>
</tr>
<tr>
<td>7.00</td>
<td>760,205</td>
<td>744,499</td>
<td>775,101</td>
<td>5,970</td>
<td>6,551</td>
<td>0.86%</td>
</tr>
<tr>
<td>8.00</td>
<td>795,396</td>
<td>773,852</td>
<td>809,759</td>
<td>6,758</td>
<td>9,077</td>
<td>1.14%</td>
</tr>
<tr>
<td>9.00</td>
<td>819,104</td>
<td>800,514</td>
<td>846,428</td>
<td>6,662</td>
<td>10,718</td>
<td>1.31%</td>
</tr>
<tr>
<td>10.00</td>
<td>853,648</td>
<td>818,368</td>
<td>889,636</td>
<td>3,681</td>
<td>18,839</td>
<td>2.21%</td>
</tr>
<tr>
<td>11.00</td>
<td>885,892</td>
<td>840,961</td>
<td>948,920</td>
<td>3,052</td>
<td>29,769</td>
<td>3.36%</td>
</tr>
<tr>
<td>12.00</td>
<td>897,555</td>
<td>854,537</td>
<td>944,427</td>
<td>3,518</td>
<td>24,916</td>
<td>2.78%</td>
</tr>
<tr>
<td>13.00</td>
<td>917,688</td>
<td>881,089</td>
<td>963,015</td>
<td>2,621</td>
<td>22,390</td>
<td>2.44%</td>
</tr>
<tr>
<td>14.00</td>
<td>932,470</td>
<td>895,660</td>
<td>976,755</td>
<td>2,492</td>
<td>23,588</td>
<td>2.53%</td>
</tr>
<tr>
<td>15.00</td>
<td>942,137</td>
<td>890,176</td>
<td>976,755</td>
<td>2,265</td>
<td>20,408</td>
<td>2.17%</td>
</tr>
<tr>
<td>16.00</td>
<td>947,501</td>
<td>908,036</td>
<td>972,957</td>
<td>1,987</td>
<td>16,742</td>
<td>1.77%</td>
</tr>
<tr>
<td>17.00</td>
<td>955,875</td>
<td>917,468</td>
<td>976,167</td>
<td>1,144</td>
<td>13,854</td>
<td>1.45%</td>
</tr>
<tr>
<td>18.00</td>
<td>960,366</td>
<td>939,916</td>
<td>976,755</td>
<td>1,159</td>
<td>9,577</td>
<td>1.00%</td>
</tr>
<tr>
<td>19.00</td>
<td>960,306</td>
<td>928,798</td>
<td>977,427</td>
<td>775</td>
<td>12,006</td>
<td>1.25%</td>
</tr>
<tr>
<td>20.00</td>
<td>957,839</td>
<td>927,437</td>
<td>976,755</td>
<td>998</td>
<td>11,938</td>
<td>1.25%</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Average</td>
<td>Best</td>
<td>Worst</td>
<td>Av. # Moves</td>
<td>Std. Dev</td>
<td>Coeff. Var</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>------</td>
<td>-------</td>
<td>-------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>0.10</td>
<td>360,857</td>
<td>338,866</td>
<td>384,548</td>
<td>6,168</td>
<td>10,382</td>
<td>2.88%</td>
</tr>
<tr>
<td>0.20</td>
<td>379,005</td>
<td>360,060</td>
<td>410,944</td>
<td>5,841</td>
<td>13,818</td>
<td>3.65%</td>
</tr>
<tr>
<td>0.30</td>
<td>393,293</td>
<td>374,518</td>
<td>410,082</td>
<td>6,814</td>
<td>9,674</td>
<td>2.46%</td>
</tr>
<tr>
<td>0.40</td>
<td>404,577</td>
<td>381,821</td>
<td>438,954</td>
<td>6,440</td>
<td>13,050</td>
<td>3.23%</td>
</tr>
<tr>
<td>0.50</td>
<td>415,884</td>
<td>398,342</td>
<td>434,981</td>
<td>5,497</td>
<td>9,555</td>
<td>2.30%</td>
</tr>
<tr>
<td>0.60</td>
<td>428,350</td>
<td>409,974</td>
<td>454,439</td>
<td>6,566</td>
<td>10,479</td>
<td>2.45%</td>
</tr>
<tr>
<td>0.70</td>
<td>441,935</td>
<td>426,785</td>
<td>466,691</td>
<td>6,604</td>
<td>9,026</td>
<td>2.04%</td>
</tr>
<tr>
<td>0.80</td>
<td>452,602</td>
<td>432,697</td>
<td>480,064</td>
<td>6,561</td>
<td>9,970</td>
<td>2.20%</td>
</tr>
<tr>
<td>0.90</td>
<td>461,269</td>
<td>446,691</td>
<td>480,184</td>
<td>5,782</td>
<td>7,904</td>
<td>1.71%</td>
</tr>
<tr>
<td>1.00</td>
<td>475,363</td>
<td>459,349</td>
<td>539,950</td>
<td>5,539</td>
<td>14,242</td>
<td>3.00%</td>
</tr>
<tr>
<td>2.00</td>
<td>562,926</td>
<td>549,637</td>
<td>597,428</td>
<td>2,520</td>
<td>10,132</td>
<td>1.80%</td>
</tr>
<tr>
<td>3.00</td>
<td>614,589</td>
<td>606,351</td>
<td>629,500</td>
<td>2,152</td>
<td>5,706</td>
<td>0.93%</td>
</tr>
<tr>
<td>4.00</td>
<td>659,084</td>
<td>645,503</td>
<td>671,910</td>
<td>2,768</td>
<td>6,768</td>
<td>1.03%</td>
</tr>
<tr>
<td>5.00</td>
<td>699,763</td>
<td>683,391</td>
<td>719,396</td>
<td>2,463</td>
<td>9,038</td>
<td>1.29%</td>
</tr>
<tr>
<td>6.00</td>
<td>746,484</td>
<td>720,503</td>
<td>773,359</td>
<td>2,147</td>
<td>14,786</td>
<td>1.98%</td>
</tr>
<tr>
<td>7.00</td>
<td>755,519</td>
<td>741,558</td>
<td>780,129</td>
<td>2,757</td>
<td>8,566</td>
<td>1.13%</td>
</tr>
<tr>
<td>8.00</td>
<td>782,239</td>
<td>766,783</td>
<td>797,354</td>
<td>3,011</td>
<td>7,355</td>
<td>0.94%</td>
</tr>
<tr>
<td>9.00</td>
<td>808,379</td>
<td>797,860</td>
<td>837,692</td>
<td>3,179</td>
<td>7,878</td>
<td>0.97%</td>
</tr>
<tr>
<td>10.00</td>
<td>831,165</td>
<td>823,341</td>
<td>864,308</td>
<td>3,043</td>
<td>7,250</td>
<td>0.87%</td>
</tr>
<tr>
<td>11.00</td>
<td>873,470</td>
<td>836,711</td>
<td>946,006</td>
<td>3,362</td>
<td>30,789</td>
<td>3.52%</td>
</tr>
<tr>
<td>12.00</td>
<td>896,640</td>
<td>856,632</td>
<td>943,514</td>
<td>3,529</td>
<td>24,124</td>
<td>2.69%</td>
</tr>
<tr>
<td>13.00</td>
<td>915,713</td>
<td>868,072</td>
<td>968,544</td>
<td>3,183</td>
<td>28,094</td>
<td>3.07%</td>
</tr>
<tr>
<td>14.00</td>
<td>927,446</td>
<td>878,286</td>
<td>996,905</td>
<td>3,154</td>
<td>28,616</td>
<td>3.09%</td>
</tr>
<tr>
<td>15.00</td>
<td>952,331</td>
<td>886,250</td>
<td>993,704</td>
<td>2,864</td>
<td>27,924</td>
<td>2.93%</td>
</tr>
<tr>
<td>16.00</td>
<td>958,513</td>
<td>902,042</td>
<td>1,005,713</td>
<td>2,255</td>
<td>26,337</td>
<td>2.75%</td>
</tr>
<tr>
<td>17.00</td>
<td>972,134</td>
<td>914,686</td>
<td>995,357</td>
<td>1,681</td>
<td>20,507</td>
<td>2.11%</td>
</tr>
<tr>
<td>18.00</td>
<td>976,229</td>
<td>924,221</td>
<td>1,012,702</td>
<td>1,838</td>
<td>18,815</td>
<td>1.93%</td>
</tr>
<tr>
<td>19.00</td>
<td>982,175</td>
<td>942,033</td>
<td>1,001,490</td>
<td>1,359</td>
<td>15,009</td>
<td>1.53%</td>
</tr>
<tr>
<td>20.00</td>
<td>986,732</td>
<td>957,224</td>
<td>1,005,713</td>
<td>1,025</td>
<td>12,744</td>
<td>1.29%</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Average</td>
<td>Best</td>
<td>Worst</td>
<td>Av. # Moves</td>
<td>Std. Dev</td>
<td>Coeff. Var</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
<td>----------</td>
<td>----------</td>
<td>-------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>0.10</td>
<td>360,857</td>
<td>338,866</td>
<td>384,548</td>
<td>6,168</td>
<td>10,382</td>
<td>2.88%</td>
</tr>
<tr>
<td>0.20</td>
<td>379,005</td>
<td>360,060</td>
<td>410,944</td>
<td>5,841</td>
<td>13,818</td>
<td>3.65%</td>
</tr>
<tr>
<td>0.30</td>
<td>393,293</td>
<td>374,518</td>
<td>410,082</td>
<td>6,814</td>
<td>9,674</td>
<td>2.46%</td>
</tr>
<tr>
<td>0.40</td>
<td>404,577</td>
<td>381,821</td>
<td>438,954</td>
<td>6,440</td>
<td>13,050</td>
<td>3.23%</td>
</tr>
<tr>
<td>0.50</td>
<td>415,884</td>
<td>398,342</td>
<td>434,981</td>
<td>5,497</td>
<td>9,555</td>
<td>2.30%</td>
</tr>
<tr>
<td>0.60</td>
<td>428,350</td>
<td>409,974</td>
<td>454,439</td>
<td>6,566</td>
<td>10,479</td>
<td>2.45%</td>
</tr>
<tr>
<td>0.70</td>
<td>441,935</td>
<td>426,785</td>
<td>466,691</td>
<td>6,604</td>
<td>9,026</td>
<td>2.04%</td>
</tr>
<tr>
<td>0.80</td>
<td>452,602</td>
<td>432,697</td>
<td>480,064</td>
<td>6,561</td>
<td>9,970</td>
<td>2.20%</td>
</tr>
<tr>
<td>0.90</td>
<td>461,269</td>
<td>446,691</td>
<td>480,184</td>
<td>5,782</td>
<td>7,904</td>
<td>1.71%</td>
</tr>
<tr>
<td>1.00</td>
<td>475,363</td>
<td>459,349</td>
<td>539,950</td>
<td>5,539</td>
<td>14,242</td>
<td>3.00%</td>
</tr>
<tr>
<td>2.00</td>
<td>562,926</td>
<td>549,637</td>
<td>597,428</td>
<td>2,520</td>
<td>10,132</td>
<td>1.80%</td>
</tr>
<tr>
<td>3.00</td>
<td>614,589</td>
<td>606,351</td>
<td>629,500</td>
<td>2,152</td>
<td>5,706</td>
<td>0.93%</td>
</tr>
<tr>
<td>4.00</td>
<td>659,084</td>
<td>645,503</td>
<td>671,910</td>
<td>2,768</td>
<td>6,768</td>
<td>1.03%</td>
</tr>
<tr>
<td>5.00</td>
<td>699,763</td>
<td>683,391</td>
<td>719,396</td>
<td>2,463</td>
<td>9,038</td>
<td>1.29%</td>
</tr>
<tr>
<td>6.00</td>
<td>746,484</td>
<td>720,503</td>
<td>773,359</td>
<td>2,147</td>
<td>14,786</td>
<td>1.98%</td>
</tr>
<tr>
<td>7.00</td>
<td>755,519</td>
<td>741,558</td>
<td>780,129</td>
<td>2,757</td>
<td>8,566</td>
<td>1.13%</td>
</tr>
<tr>
<td>8.00</td>
<td>782,239</td>
<td>766,783</td>
<td>797,354</td>
<td>3,011</td>
<td>7,355</td>
<td>0.94%</td>
</tr>
<tr>
<td>9.00</td>
<td>808,379</td>
<td>797,860</td>
<td>834,692</td>
<td>3,179</td>
<td>7,878</td>
<td>0.97%</td>
</tr>
<tr>
<td>10.00</td>
<td>831,165</td>
<td>823,341</td>
<td>864,308</td>
<td>3,043</td>
<td>7,250</td>
<td>0.87%</td>
</tr>
<tr>
<td>11.00</td>
<td>873,470</td>
<td>836,711</td>
<td>946,006</td>
<td>3,362</td>
<td>30,789</td>
<td>3.52%</td>
</tr>
<tr>
<td>12.00</td>
<td>896,640</td>
<td>856,632</td>
<td>943,514</td>
<td>3,529</td>
<td>24,124</td>
<td>2.69%</td>
</tr>
<tr>
<td>13.00</td>
<td>915,713</td>
<td>868,072</td>
<td>968,544</td>
<td>3,183</td>
<td>28,094</td>
<td>3.07%</td>
</tr>
<tr>
<td>14.00</td>
<td>927,446</td>
<td>878,286</td>
<td>996,905</td>
<td>3,154</td>
<td>28,616</td>
<td>3.09%</td>
</tr>
<tr>
<td>15.00</td>
<td>952,331</td>
<td>886,250</td>
<td>993,704</td>
<td>2,864</td>
<td>27,924</td>
<td>2.93%</td>
</tr>
<tr>
<td>16.00</td>
<td>958,513</td>
<td>902,042</td>
<td>1,005,713</td>
<td>2,255</td>
<td>26,337</td>
<td>2.75%</td>
</tr>
<tr>
<td>17.00</td>
<td>972,134</td>
<td>914,686</td>
<td>995,357</td>
<td>1,681</td>
<td>20,507</td>
<td>2.11%</td>
</tr>
<tr>
<td>18.00</td>
<td>976,229</td>
<td>924,221</td>
<td>1,012,702</td>
<td>1,838</td>
<td>18,815</td>
<td>1.93%</td>
</tr>
<tr>
<td>19.00</td>
<td>982,175</td>
<td>942,033</td>
<td>1,001,490</td>
<td>1,359</td>
<td>15,009</td>
<td>1.53%</td>
</tr>
<tr>
<td>20.00</td>
<td>986,732</td>
<td>957,224</td>
<td>1,005,713</td>
<td>1,025</td>
<td>12,744</td>
<td>1.29%</td>
</tr>
<tr>
<td>$p$</td>
<td>Average</td>
<td>Best</td>
<td>Worst</td>
<td>Av. # Moves</td>
<td>Std. Dev</td>
<td>Coeff. Var</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
<td>-------</td>
<td>--------</td>
<td>-------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>0.10</td>
<td>355,194</td>
<td>335,891</td>
<td>374,366</td>
<td>5,982</td>
<td>8,544</td>
<td>2.41%</td>
</tr>
<tr>
<td>0.20</td>
<td>371,896</td>
<td>353,456</td>
<td>398,604</td>
<td>6,247</td>
<td>11,175</td>
<td>3.00%</td>
</tr>
<tr>
<td>0.30</td>
<td>388,706</td>
<td>372,072</td>
<td>407,381</td>
<td>6,827</td>
<td>9,012</td>
<td>2.32%</td>
</tr>
<tr>
<td>0.40</td>
<td>404,450</td>
<td>378,911</td>
<td>432,390</td>
<td>6,976</td>
<td>11,955</td>
<td>2.96%</td>
</tr>
<tr>
<td>0.50</td>
<td>410,688</td>
<td>392,100</td>
<td>422,212</td>
<td>5,826</td>
<td>7,823</td>
<td>1.90%</td>
</tr>
<tr>
<td>0.60</td>
<td>422,519</td>
<td>410,298</td>
<td>448,384</td>
<td>6,342</td>
<td>7,810</td>
<td>1.85%</td>
</tr>
<tr>
<td>0.70</td>
<td>434,268</td>
<td>423,831</td>
<td>450,199</td>
<td>5,938</td>
<td>6,406</td>
<td>1.48%</td>
</tr>
<tr>
<td>0.80</td>
<td>446,268</td>
<td>432,693</td>
<td>473,403</td>
<td>6,060</td>
<td>9,199</td>
<td>2.06%</td>
</tr>
<tr>
<td>0.90</td>
<td>463,583</td>
<td>448,590</td>
<td>485,551</td>
<td>6,565</td>
<td>9,850</td>
<td>2.12%</td>
</tr>
<tr>
<td>1.00</td>
<td>469,649</td>
<td>449,190</td>
<td>495,752</td>
<td>6,269</td>
<td>12,073</td>
<td>2.57%</td>
</tr>
<tr>
<td>2.00</td>
<td>557,068</td>
<td>543,643</td>
<td>567,350</td>
<td>4,674</td>
<td>6,081</td>
<td>1.09%</td>
</tr>
<tr>
<td>3.00</td>
<td>615,291</td>
<td>602,303</td>
<td>631,406</td>
<td>4,512</td>
<td>6,831</td>
<td>1.11%</td>
</tr>
<tr>
<td>4.00</td>
<td>680,258</td>
<td>658,974</td>
<td>689,415</td>
<td>4,723</td>
<td>6,538</td>
<td>0.96%</td>
</tr>
<tr>
<td>5.00</td>
<td>718,303</td>
<td>702,290</td>
<td>739,007</td>
<td>4,302</td>
<td>9,040</td>
<td>1.26%</td>
</tr>
<tr>
<td>6.00</td>
<td>753,642</td>
<td>737,045</td>
<td>768,181</td>
<td>3,662</td>
<td>9,562</td>
<td>1.27%</td>
</tr>
<tr>
<td>7.00</td>
<td>764,900</td>
<td>757,288</td>
<td>784,556</td>
<td>5,114</td>
<td>6,206</td>
<td>0.81%</td>
</tr>
<tr>
<td>8.00</td>
<td>778,061</td>
<td>771,395</td>
<td>785,279</td>
<td>5,796</td>
<td>3,181</td>
<td>0.41%</td>
</tr>
<tr>
<td>9.00</td>
<td>793,832</td>
<td>787,248</td>
<td>804,584</td>
<td>6,292</td>
<td>3,781</td>
<td>0.48%</td>
</tr>
<tr>
<td>10.00</td>
<td>822,095</td>
<td>803,078</td>
<td>862,365</td>
<td>3,674</td>
<td>15,961</td>
<td>1.94%</td>
</tr>
<tr>
<td>11.00</td>
<td>848,080</td>
<td>821,686</td>
<td>909,755</td>
<td>4,233</td>
<td>21,753</td>
<td>2.57%</td>
</tr>
<tr>
<td>12.00</td>
<td>867,672</td>
<td>835,343</td>
<td>935,164</td>
<td>3,905</td>
<td>24,324</td>
<td>2.80%</td>
</tr>
<tr>
<td>13.00</td>
<td>891,471</td>
<td>858,850</td>
<td>946,548</td>
<td>3,510</td>
<td>24,294</td>
<td>2.73%</td>
</tr>
<tr>
<td>14.00</td>
<td>903,398</td>
<td>847,807</td>
<td>953,705</td>
<td>3,595</td>
<td>24,035</td>
<td>2.66%</td>
</tr>
<tr>
<td>15.00</td>
<td>927,238</td>
<td>889,962</td>
<td>953,558</td>
<td>2,847</td>
<td>18,175</td>
<td>1.96%</td>
</tr>
<tr>
<td>16.00</td>
<td>937,320</td>
<td>897,694</td>
<td>955,927</td>
<td>2,248</td>
<td>15,002</td>
<td>1.60%</td>
</tr>
<tr>
<td>17.00</td>
<td>937,979</td>
<td>894,679</td>
<td>982,767</td>
<td>1,603</td>
<td>20,388</td>
<td>2.17%</td>
</tr>
<tr>
<td>18.00</td>
<td>944,940</td>
<td>905,066</td>
<td>989,687</td>
<td>1,062</td>
<td>16,307</td>
<td>1.73%</td>
</tr>
<tr>
<td>19.00</td>
<td>939,611</td>
<td>900,463</td>
<td>960,953</td>
<td>959</td>
<td>17,717</td>
<td>1.89%</td>
</tr>
<tr>
<td>20.00</td>
<td>942,897</td>
<td>911,864</td>
<td>961,315</td>
<td>1,018</td>
<td>12,829</td>
<td>1.36%</td>
</tr>
<tr>
<td>ρ</td>
<td>Average</td>
<td>Best</td>
<td>Worst</td>
<td>Av. # Moves</td>
<td>Std. Dev</td>
<td>Coeff. Var</td>
</tr>
<tr>
<td>----</td>
<td>---------</td>
<td>----------</td>
<td>---------</td>
<td>-------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>0.10</td>
<td>366,261</td>
<td>354,864</td>
<td>381,534</td>
<td>6,922</td>
<td>6,160</td>
<td>1.68%</td>
</tr>
<tr>
<td>0.20</td>
<td>384,998</td>
<td>370,591</td>
<td>414,147</td>
<td>6,313</td>
<td>8,929</td>
<td>2.32%</td>
</tr>
<tr>
<td>0.30</td>
<td>399,396</td>
<td>381,835</td>
<td>423,952</td>
<td>6,359</td>
<td>8,463</td>
<td>2.12%</td>
</tr>
<tr>
<td>0.40</td>
<td>409,673</td>
<td>396,506</td>
<td>429,372</td>
<td>6,247</td>
<td>7,255</td>
<td>1.77%</td>
</tr>
<tr>
<td>0.50</td>
<td>422,068</td>
<td>410,248</td>
<td>437,666</td>
<td>6,846</td>
<td>5,678</td>
<td>1.35%</td>
</tr>
<tr>
<td>0.60</td>
<td>435,168</td>
<td>420,995</td>
<td>455,677</td>
<td>6,908</td>
<td>6,825</td>
<td>1.57%</td>
</tr>
<tr>
<td>0.70</td>
<td>449,218</td>
<td>440,076</td>
<td>462,712</td>
<td>6,621</td>
<td>5,446</td>
<td>1.21%</td>
</tr>
<tr>
<td>0.80</td>
<td>460,658</td>
<td>452,428</td>
<td>470,878</td>
<td>6,602</td>
<td>4,959</td>
<td>1.08%</td>
</tr>
<tr>
<td>0.90</td>
<td>472,336</td>
<td>464,832</td>
<td>482,635</td>
<td>6,826</td>
<td>3,900</td>
<td>0.83%</td>
</tr>
<tr>
<td>1.00</td>
<td>482,379</td>
<td>472,945</td>
<td>491,034</td>
<td>5,957</td>
<td>5,205</td>
<td>1.08%</td>
</tr>
<tr>
<td>2.00</td>
<td>567,462</td>
<td>555,134</td>
<td>579,355</td>
<td>4,044</td>
<td>5,768</td>
<td>1.02%</td>
</tr>
<tr>
<td>3.00</td>
<td>622,883</td>
<td>615,186</td>
<td>636,696</td>
<td>5,197</td>
<td>6,044</td>
<td>0.97%</td>
</tr>
<tr>
<td>4.00</td>
<td>690,574</td>
<td>682,609</td>
<td>697,660</td>
<td>5,338</td>
<td>3,807</td>
<td>0.55%</td>
</tr>
<tr>
<td>5.00</td>
<td>728,796</td>
<td>715,991</td>
<td>737,008</td>
<td>4,998</td>
<td>3,727</td>
<td>0.51%</td>
</tr>
<tr>
<td>6.00</td>
<td>763,545</td>
<td>751,860</td>
<td>774,774</td>
<td>5,018</td>
<td>5,965</td>
<td>0.78%</td>
</tr>
<tr>
<td>7.00</td>
<td>781,114</td>
<td>773,313</td>
<td>798,890</td>
<td>4,738</td>
<td>5,171</td>
<td>0.66%</td>
</tr>
<tr>
<td>8.00</td>
<td>802,559</td>
<td>794,073</td>
<td>819,313</td>
<td>5,760</td>
<td>5,989</td>
<td>0.75%</td>
</tr>
<tr>
<td>9.00</td>
<td>828,755</td>
<td>814,255</td>
<td>888,197</td>
<td>5,422</td>
<td>15,123</td>
<td>1.82%</td>
</tr>
<tr>
<td>10.00</td>
<td>852,096</td>
<td>831,461</td>
<td>901,572</td>
<td>3,410</td>
<td>15,142</td>
<td>1.78%</td>
</tr>
<tr>
<td>11.00</td>
<td>880,564</td>
<td>854,286</td>
<td>917,273</td>
<td>4,140</td>
<td>17,559</td>
<td>1.99%</td>
</tr>
<tr>
<td>12.00</td>
<td>905,166</td>
<td>872,018</td>
<td>934,516</td>
<td>3,170</td>
<td>17,875</td>
<td>1.97%</td>
</tr>
<tr>
<td>13.00</td>
<td>919,062</td>
<td>881,542</td>
<td>946,337</td>
<td>2,566</td>
<td>14,560</td>
<td>1.58%</td>
</tr>
<tr>
<td>14.00</td>
<td>930,489</td>
<td>908,317</td>
<td>961,527</td>
<td>2,182</td>
<td>14,856</td>
<td>1.60%</td>
</tr>
<tr>
<td>15.00</td>
<td>937,316</td>
<td>915,160</td>
<td>989,872</td>
<td>2,337</td>
<td>15,224</td>
<td>1.62%</td>
</tr>
<tr>
<td>16.00</td>
<td>953,857</td>
<td>925,140</td>
<td>989,872</td>
<td>1,328</td>
<td>16,711</td>
<td>1.75%</td>
</tr>
<tr>
<td>17.00</td>
<td>959,202</td>
<td>920,233</td>
<td>991,829</td>
<td>1,340</td>
<td>19,110</td>
<td>1.99%</td>
</tr>
<tr>
<td>18.00</td>
<td>958,573</td>
<td>924,642</td>
<td>1,000,944</td>
<td>999</td>
<td>19,877</td>
<td>2.07%</td>
</tr>
<tr>
<td>19.00</td>
<td>959,642</td>
<td>937,699</td>
<td>993,785</td>
<td>1,065</td>
<td>18,771</td>
<td>1.96%</td>
</tr>
<tr>
<td>20.00</td>
<td>967,168</td>
<td>938,089</td>
<td>989,872</td>
<td>1,104</td>
<td>16,119</td>
<td>1.67%</td>
</tr>
<tr>
<td>$p$</td>
<td>Average</td>
<td>Best</td>
<td>Worst</td>
<td>Av. # Moves</td>
<td>Std. Dev</td>
<td>Coeff. Var</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-------</td>
<td>--------</td>
<td>-------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>0.10</td>
<td>344,628</td>
<td>330,192</td>
<td>357,913</td>
<td>6,118</td>
<td>5,595</td>
<td>1.62%</td>
</tr>
<tr>
<td>0.20</td>
<td>361,575</td>
<td>341,177</td>
<td>377,259</td>
<td>6,580</td>
<td>9,134</td>
<td>2.53%</td>
</tr>
<tr>
<td>0.30</td>
<td>375,309</td>
<td>364,907</td>
<td>390,505</td>
<td>5,644</td>
<td>6,060</td>
<td>1.61%</td>
</tr>
<tr>
<td>0.40</td>
<td>391,333</td>
<td>370,908</td>
<td>411,908</td>
<td>6,625</td>
<td>8,628</td>
<td>2.20%</td>
</tr>
<tr>
<td>0.50</td>
<td>399,814</td>
<td>389,354</td>
<td>415,610</td>
<td>6,706</td>
<td>5,920</td>
<td>1.48%</td>
</tr>
<tr>
<td>0.60</td>
<td>414,245</td>
<td>399,905</td>
<td>448,469</td>
<td>6,864</td>
<td>9,267</td>
<td>2.24%</td>
</tr>
<tr>
<td>0.70</td>
<td>425,136</td>
<td>417,442</td>
<td>439,919</td>
<td>7,086</td>
<td>5,627</td>
<td>1.32%</td>
</tr>
<tr>
<td>0.80</td>
<td>436,861</td>
<td>426,332</td>
<td>458,270</td>
<td>6,678</td>
<td>6,199</td>
<td>1.42%</td>
</tr>
<tr>
<td>0.90</td>
<td>447,766</td>
<td>434,661</td>
<td>464,597</td>
<td>5,509</td>
<td>6,745</td>
<td>1.51%</td>
</tr>
<tr>
<td>1.00</td>
<td>454,232</td>
<td>447,463</td>
<td>467,514</td>
<td>6,444</td>
<td>5,388</td>
<td>1.19%</td>
</tr>
<tr>
<td>2.00</td>
<td>542,253</td>
<td>528,390</td>
<td>554,351</td>
<td>4,448</td>
<td>6,135</td>
<td>1.13%</td>
</tr>
<tr>
<td>3.00</td>
<td>601,378</td>
<td>591,256</td>
<td>615,316</td>
<td>4,491</td>
<td>5,904</td>
<td>0.98%</td>
</tr>
<tr>
<td>4.00</td>
<td>664,709</td>
<td>644,091</td>
<td>679,496</td>
<td>5,129</td>
<td>10,942</td>
<td>1.65%</td>
</tr>
<tr>
<td>5.00</td>
<td>706,097</td>
<td>685,099</td>
<td>717,748</td>
<td>4,418</td>
<td>10,646</td>
<td>1.51%</td>
</tr>
<tr>
<td>6.00</td>
<td>743,660</td>
<td>726,986</td>
<td>771,410</td>
<td>5,198</td>
<td>9,934</td>
<td>1.34%</td>
</tr>
<tr>
<td>7.00</td>
<td>772,190</td>
<td>759,618</td>
<td>795,489</td>
<td>6,262</td>
<td>7,672</td>
<td>0.99%</td>
</tr>
<tr>
<td>8.00</td>
<td>791,913</td>
<td>778,900</td>
<td>808,442</td>
<td>5,142</td>
<td>6,660</td>
<td>0.84%</td>
</tr>
<tr>
<td>9.00</td>
<td>816,544</td>
<td>792,232</td>
<td>901,418</td>
<td>5,812</td>
<td>18,948</td>
<td>2.32%</td>
</tr>
<tr>
<td>10.00</td>
<td>837,555</td>
<td>822,173</td>
<td>886,514</td>
<td>3,806</td>
<td>14,584</td>
<td>1.74%</td>
</tr>
<tr>
<td>11.00</td>
<td>865,040</td>
<td>830,367</td>
<td>911,879</td>
<td>3,235</td>
<td>21,831</td>
<td>2.52%</td>
</tr>
<tr>
<td>12.00</td>
<td>886,153</td>
<td>856,992</td>
<td>943,031</td>
<td>3,669</td>
<td>23,247</td>
<td>2.62%</td>
</tr>
<tr>
<td>13.00</td>
<td>894,590</td>
<td>846,698</td>
<td>925,418</td>
<td>3,282</td>
<td>19,390</td>
<td>2.17%</td>
</tr>
<tr>
<td>14.00</td>
<td>917,359</td>
<td>879,364</td>
<td>947,761</td>
<td>2,859</td>
<td>18,376</td>
<td>2.00%</td>
</tr>
<tr>
<td>15.00</td>
<td>923,271</td>
<td>891,688</td>
<td>948,356</td>
<td>2,728</td>
<td>17,017</td>
<td>1.84%</td>
</tr>
<tr>
<td>16.00</td>
<td>932,717</td>
<td>893,634</td>
<td>993,459</td>
<td>2,254</td>
<td>17,470</td>
<td>1.87%</td>
</tr>
<tr>
<td>17.00</td>
<td>945,827</td>
<td>921,361</td>
<td>993,459</td>
<td>954</td>
<td>15,662</td>
<td>1.66%</td>
</tr>
<tr>
<td>18.00</td>
<td>942,210</td>
<td>907,107</td>
<td>976,215</td>
<td>1,647</td>
<td>16,492</td>
<td>1.75%</td>
</tr>
<tr>
<td>19.00</td>
<td>946,440</td>
<td>923,327</td>
<td>993,459</td>
<td>1,199</td>
<td>15,843</td>
<td>1.67%</td>
</tr>
<tr>
<td>20.00</td>
<td>949,299</td>
<td>923,648</td>
<td>993,459</td>
<td>1,081</td>
<td>15,900</td>
<td>1.67%</td>
</tr>
</tbody>
</table>
Figure D.1. Solutions for Shulkell Dataset

Figure D.2. Solutions for Dataset 1.
Figure D.3. Solutions for Dataset 2.

Figure D.4. Solutions for Dataset 3.
Figure D.5. Solutions for Dataset 4.

Figure D.6. Solutions for Dataset 5.
Figure D.8. Low Roading Cost Solution
Road Cost $15,062, Lost Volume 674,410.4.
Figure D.9 Medium-low Roading Cost Solution
Roading cost $30,030, Lost Volume 561,148
Figure D.11. High Roading Cost Solution
Road Cost $105,419, Lost volume 315,782
Figure D.12. Openings, Periods 1 and 3, Solution D.10
Figure D.13. Openings, Periods 2 and 4, Solution D.10