DESIGN AND IMPLEMENTATION OF COLUMN GENERATION TO SOLVE LARGE-SCALE PROFIT-MAXIMIZATION FOREST MANAGEMENT PROBLEMS

A Thesis in
Industrial Engineering
by
Krunal Atul Mehta

© 2008 Krunal Atul Mehta

Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

August 2008
The thesis of Krunal Atul Mehta was reviewed and approved* by the following:

Tom M. Cavalier
Professor of Industrial Engineering
Thesis Adviser

Marc E. McDill
Associate Professor of Forest Resource Management

Richard J. Koubek
Professor of Industrial Engineering
Head of the Department of Industrial and Manufacturing Engineering

*Signatures are on file in the Graduate School
Spatially-explicit timber harvest scheduling models have increased in size and complexity as planners address a growing array of entities which translates to constraints of varying form in the forest management planning. After a certain extent of forest sizes, it becomes virtually impossible for commercial solvers to solve the problems under tolerable computational times. Hierarchical planning models provide a better option of managing these large and complex models through decomposition techniques. A Lagrangian relaxation and a column generation algorithm – a modified Dantzig-Wolfe decomposition technique can be used for solving large-scale problems by decomposing them into a set of smaller linked models. Most of the real-world forests can be divided into spatially independent compartments because of the streams, roads, trails, and pipelines running through it. In this paper, a column generation algorithm is developed and implemented to exploit spatial nature of the forest. The price directed approach is pursued to create the randomly generated sub-problems in the initial step and then during the iterative creation of master problems. An integrated software was developed that created original, sub-problem and master formulation files and solved the formulations using commercially available software that uses a Branch and Bound technique. The time required to solve the problem with and without the application of decomposition is compared and the parameters that influence the solution quality in the column generation algorithm are investigated.
# TABLE OF CONTENTS

List of Figures ................................................................................................................. vi

List of Tables .................................................................................................................. viii

Acknowledgements ......................................................................................................... ix

Chapter 1  Introduction to forest management and column generation ....................... 1
  Forest management and Harvest scheduling ............................................................... 1
  Linear programming .................................................................................................... 2
  Need of spatially explicit integer programming model .............................................. 4
  Branch-and-Bound algorithm ..................................................................................... 9
  Introduction to column generation ......................................................................... 11
  Research purpose and objectives ............................................................................. 14

Chapter 2  Literature Review ......................................................................................... 19
  Organization of the chapter ....................................................................................... 19
  Harvest scheduling .................................................................................................... 20
  Dantzig-Wolfe decomposition and Lagrangian relaxation .................................... 25
  Application of decomposition techniques in forest management ....................... 29

Chapter 3  Methodology ................................................................................................. 33
  NPV Maximization model ......................................................................................... 34
  Dantzig-Wolfe decomposition and column generation ......................................... 38
  Decomposing forest into spatially independent compartments ............................. 41
  Constraint structure in the formulation ................................................................... 42
  Exploiting constraint structure and Lagrangian relaxation .................................... 43
  Initialization and generation of columns ................................................................. 50
  Initial restricted master problem ............................................................................. 53
  Iterative subproblem formulation ........................................................................... 56
  Stopping rule ............................................................................................................ 59
  Example forest case study ....................................................................................... 60
  Treatment unit delineations ..................................................................................... 63
  Forest Data .............................................................................................................. 66
  Column generation and harvest scheduling software system ............................... 68
  Factors influencing the column generation algorithm ......................................... 70
    Number of compartments in the forest ................................................................. 71
    Number of management units in the compartment .............................................. 73
    Number of seeds created for initial RLMP .......................................................... 73
    Variation $\eta_i$ in perturbing the shadow prices of original linear problems . 74
LIST OF FIGURES

Figure 1-1: Initial age-class distribution for District-13 forest ..................................... 5

Figure 1-2: Desired ending age-class distribution for District-13 forest ...................... 6

Figure 1-3: District-8 pre-delineated forest structure.................................................... 15

Figure 1-4: District-8 delineated treatment unit level forest structure ....................... 17

Figure 3-1: Column generation algorithm..................................................................... 40

Figure 3-2: PA DCNR BOF managed District-8 Forest in Kittanning county of Pennsylvania......................................................................................................................... 41

Figure 3-3: DCNR Bureau of Forestry District-8 M&C-zoned stand boundaries ....... 61

Figure 3-4: Initial age-class distribution for the M&C zone of District-8 forest specified by forest type................................................................................................................. 62

Figure 3-5: District-8 delineated treatment unit layer..................................................... 65

Figure 4-1: Average objective function values plotted against number of initial seeds for each shadow price perturbation interval ...................................................... 79

Figure 4-2: Graph of objective function values for all 200 replications against the number of initial seeds ........................................................................................................ 80

Figure 4-3: Average solution time plotted against the number of initial seeds for different shadow price perturbation intervals......................................................... 83

Figure 4-4: Graph of solution time against number of initial seeds for all 200 replications ......................................................................................................................... 85

Figure 4-5: Hypothetical graph of how the optimality gap determined by CPLEX evolves over time as the algorithm solves a problem......................................................... 86

Figure 4-6: Average number of iterations plotted against the number of initial seeds for different shadow price perturbations............................................................. 89

Figure 4-7: Average number of subproblems used in RIMP plotted against the number of initial seeds for each shadow price perturbation interval........ 94
Figure 4-8: Average number of identical subproblem seeds plotted against the target number of initial seeds for each shadow price perturbation.............96

Figure 4-9: Graph of number of identical subproblem seeds in the initialization step versus the target number of initial seeds for all 200 replications .......96

Figure 4-10: Average RIMP solution time plotted against the number of initial seeds for each shadow price perturbation interval .................................99
LIST OF TABLES

Table 3-1: District-8 compartment composition .......................................................... 66
Table 3-2: Stumpage prices for wood products ........................................................... 68
Table 4-1: Comparison of column generation objective function values with the original IP solution objective function value ............................................. 78
Table 4-2: Estimate of the confidence interval for a population mean of 200 replications ........................................................................................................ 80
Table 4-3: Comparison of average solution times taken by column generation and branch-and-bound algorithm ............................................................... 82
Table 4-4: Time taken by CPLEX to achieve just better solution compared to column generation ................................................................................................ 87
Table 4-5: Comparison of average number of iterations taken by the master problem to converge for different combinations of initial seeds and shadow price perturbations ................................................................................ 89
Table 4-6: Comparison of average number of subproblem seeds used in RIMP for different combinations of initial seeds and shadow price perturbations ........................................................................... 91
Table 4-7: Comparison of average number of identical subproblem seeds in the initialization step for different combinations of initial seeds and shadow price perturbations ......................................................................... 95
Table 4-8: Comparison of average solution times taken by the RIMP for different combinations of seeding levels and shadow price perturbations ...................................................................................... 98
ACKNOWLEDGEMENTS

This research would not have been possible without the support of many people. First and foremost, I would like to thank my adviser, Dr. Marc E. McDill who was abundantly helpful and offered invaluable assistance, support and guidance. I will be forever grateful for the opportunities he provided. He was always there when I needed him. I have gained so much knowledge studying under him which I am sure is going to help me throughout my life. I am also thankful to Dr. McDill and the Pennsylvania DCNR Bureau of Forestry for the financial support provided to me during entire masters’ program.

I would also like to thank Dr. Tom Cavalier for his invaluable insights and his willingness to serve in my committee. He made me look at my work from a different perspective and understand it even better.

Special thanks to all my lab mates at the Pennsylvania State University for their constant support and encouragement. I would also like to thank all my friends who made graduate school the most memorable experience of my life. I could not have completed this research without the unwavering support of my fiancé Dipal, who was the greatest source of support, in my academic as well as personal life.

Finally, no words can suffice the support my parents have extended to me during the entire masters’ program. They were always proud of my academic achievements and were a constant source of inspiration in my academic pursuits. Without their prayers and support, I wouldn’t have made it.
Chapter 1

INTRODUCTION TO FOREST MANAGEMENT AND COLUMN GENERATION

FOREST MANAGEMENT AND HARVEST SCHEDULING

Forests cover roughly 31% of the earth’s land surface (Sedjo and Lyon, 1990). Forests provide us with several natural resources that are beneficial to individuals, corporations, and governments. Forest management was once the sole domain of professional foresters whose primary objective was to regulate forests in order to maximize the timber value and other natural products extracted from the forests. Forest management has evolved from a relatively simple decision making process as to when to cut stands in order to maximize the net present value of the timber to the current era wherein environmental concerns play an equally important role in natural resource exploitation in many forested areas. Therefore, forestry has remained a rich source for optimization problems and a field of attraction for operational researchers who seek new challenges, in many cases beyond their capabilities. Forest management problems are complex because of the diversity of the landscape, the unpredictability of the natural processes that occur in forests, the inter-relationships between the different components of the forest, and, most important, because of the diversity of values associated with natural resources.
In order to deal with these complexities, optimization in forest planning and management has been widely and successfully used for several decades. Many forest management decision problems can be formulated as mathematical programming problems with various objective functions subject to constraints faced by foresters in achieving those objectives. Mathematical programming allows foresters to solve complex problems involving potentially thousands of management areas and a wide range of objectives. Mathematical programming techniques like linear programming (LP) and mixed integer programming (MIP) have been developed to deal with management objectives to determine which stands will be harvested in which period to meet the projected demands or increase the net present value of timber or to minimize the cost of harvests. While achieving these objectives, the model also satisfies silvicultural, environmental, sustainability and several other planning constraints.

**LINEAR PROGRAMMING MODELS**

The Pennsylvania DCNR Bureau of Forestry (BOF) manages 2.1 million acres of forest owned by the State of Pennsylvania. The 2.1 million acres of forest are divided into 20 districts and are managed for the health of the ecosystem, sustainable timber production, minerals, oil and gas production, transport and storage, recreational values for current and future generations, and other goals. These goals tend to conflict with each other and therefore various planning activities over a lengthy planning horizon are required to balance these goals. Out of all these, timber harvesting is the most important one, and it affects almost all the goals mentioned above. Many activities are planned around timber harvesting. In order to produce a balanced age-class distribution and a
sustainable flow of quality wood products, forest management often focuses on efficient scheduling of timber harvesting activities. These models are often referred as harvest scheduling models (HSM).

Optimization is extensively used to schedule forest management activities under a variety of objectives and constraints. Harvest scheduling problems can be solved by either Linear Programming (LP) or Integer Programming (IP) techniques. The former technique is used more often in practice. BOF also uses harvest scheduling LP models to develop forest plans. The objective function of linear programming models may involve minimizing the cost of achieving production targets or maximizing the net present value of the forest. There are many other forest management goals which are categorized in the constraint structure of a planning model. Some typical constraints are harvest targets or fluctuation constraints, minimum average ending age or volume constraints and extended rotation constraints.

Harvest target constraints specify the amount of timber to produce in each period. Harvest fluctuation constraints maintain the even flow of timber as they specify that the maximum percentage by which the amount of timber harvested is allowed to fluctuate from one period of the planning horizon to another. Constraints can be applied to adjacent periods or to a combination of all the periods. In addition to maintaining the steady flow of timber these constraints also help to ensure that the labor required to carry out harvesting activities in each period does not fluctuate significantly. Minimum average forest ending volume or age constraints ensure that the forest is not over-harvested at the end of the planning horizon. These are necessary, as LP models typically do not account
for concerns that extend beyond the end of the planning horizon. Extended rotation constraints ensure that there is a sufficient amount of mature habitat in the forest.

However, the solutions to most LP HSMs do not tell the specific location where the harvesting should be done. This decision has to be made outside the model. Hence, LP models are called non-spatial timber harvesting models. The lack of site-specific information is a crucial limitation for using LP forest management formulations. Certain types of constraint structures, such as when and where a road should be built to access a particular management unit or fence to cover management units, cannot be addressed by non-spatial LP models

**SPATIALLY EXPLICIT INTEGER PROGRAMMING MODELS**

Throughout the late 1800s and early 1900s, extensive clear-cutting occurred across most of Pennsylvania’s forests. Most of the clear-cutting was followed by regeneration at about the same time. These activities resulted in an unregulated age-class distribution and hence most of today’s second growth forest consists of mature timber, often found in large homogeneous areas. These forest structures have posed several problems in harvest scheduling as large portions of Pennsylvania forests are in a span of 2-3 age classes.
Figure 1-1 shows the current initial age-class distribution for one of the districts (District – 13) managed by the BOF. It illustrates an example of an unregulated forest as a majority of the forest is in a mature state. The reason behind this forest structure is the near-total harvest of the forest carried out approximately one hundred years ago. The entire forest was cut over during a relatively short period of time (much less than one rotation). One of the Bureau’s objectives is to slowly bring the forest to a balanced age-class distribution in order to provide a more diverse mix of age-classes for future generations. The graph shown in Figure 1-2 shows the Bureau’s desired age-class distribution at the end of the planning horizon.
Because of its ability to handle a much wider range of management concerns and to model management costs more accurately, in the future, forest planning will most likely be spatially explicit, which means that a problem formulation will explicitly convey location information. Such models are generally formulated as IP models. These models, unlike linear programming models, use binary variables representing specific management units in the forest and specific prescriptions to apply to them. Binary variables can assume only two values, 1 and 0, indicating whether or not to apply the specific prescription to the specific management unit. The most important advantage of this type of model is that it provides the means of incorporating new constraints such as adjacency constraints, road and fence constraints, which are difficult to formulate without having spatially explicit decision variables.
Adjacency constraints are used to limit the maximum size of harvest openings. These constraints either prevent the simultaneous harvest of adjacent management units (unit restriction model, or URM) or prevent the simultaneous harvest of contiguous forest areas that exceed a certain prescribed area limit (area restriction model, or ARM). Such limits are legally required or voluntarily applied in many forests in the US and elsewhere. The BOF has mandatory guidelines on the maximum size of harvest opening allowed. The BOF generally prohibits harvesting more than 80 acres of contiguous forestland. Fenced areas are typically not allowed to exceed 50 acres (which effectively limits the size of most harvest areas to 50 acres or less).

Pennsylvania’s hardwood forests usually regenerate naturally after a harvest. Deer prefer to browse seedlings, saplings, and sprouts of the most valuable timber species making it the single most important factor limiting regeneration across the Commonwealth. Excessive deer browsing can create a future forest with little or no valuable timber, devoid of aesthetically pleasing and ecologically important shrubs, herbs, and wildflowers normally found in Pennsylvania’s diverse forests. The most effective and economical way to control and prevent deer damage is hunting. The next best option for regenerating desirable tree and plant species is fencing. The Bureau has been using the fencing option quite extensively. Over 800 miles of fence surrounding more than 27,000 acres of Pennsylvania’s forests was constructed between 1995 and 2003. The Bureau has clearly defined spatial constraints that must be considered. Since smaller fenced areas more effectively exclude deer than larger ones, fences should be kept to 50 acres or less whenever possible (Beacom 2000). If a larger area must be
fenced, Bureau guidelines call for splitting it into two smaller areas with a travel corridor one to two chains wide between fences (Beacom 2000). Under this scenario, deer will more likely travel around the fence boundary than attempt to penetrate it (Beacom 2000).

IP models have been used to address these types of concerns. The forest is first divided into a number of management units that are generally based on same forest-type or age-class distribution. While formulating a model, the decision whether the activity should be applied to the entire management unit in a given time period or not is represented with binary variables.

Clearly, spatially-explicit forest planning done through IP modeling has advantages over LP models. But, mathematically, IP management problems are significantly more difficult to solve than LP management problems. This is due to the presence of the integer variables that require a branch-and-bound solution algorithm to solve, which is far more complex and computationally intensive than using the simplex algorithm for solving LP problems. Planning models corresponding to a forest with 300 management units can take anywhere between two minutes and a month to solve on a computer. A forest with 100,000 acres, which is the typical size of a BOF forest district, could potentially have around 3000 management units. The corresponding management models would be very hard to solve. Hence in order to solve these large-scale real world problems in a reasonable amount of time, it is crucial to develop a solution methodology to solve these IP models.
BRANCH-AND-BOUND ALGORITHM

Although a number of algorithms have been proposed for IP problems, the Branch-and-Bound (B&B) technique is used in almost all off-the-shelf computer optimization software available in the industry. A popular solver known as ILOG CPLEX© was used in this research which typically employ B&B or a combination of heuristic and B&B to solve Mixed Integer Programming (MIP) and IP problems. The technique has proven to be reasonably efficient on practical problems, and it has the added advantage that it solves continuous linear programs as subproblems, that is, linear programming problems without integer restriction.

B&B algorithm is an implicit enumeration process in which a series of linear programming relaxations provide bounds (Wolsey 1998). The essence of the algorithm is as follows:

1. Solve the linear relaxation of the problem. If the solution is integer, then it is the optimal solution. Otherwise create two new subproblems by branching on a fractional variable.

2. A subproblem is not active when any of the following occurs:

   1. The subproblem is used to branch on,
   2. All variables in the solution are integer,
   3. The subproblem is infeasible,
   4. The subproblem can be fathomed by a bounding argument.
3. Choose an active subproblem and branch on a fractional variable. Repeat until there are no active subproblems.

The most critical decision is choosing which variable to branch on. In addition to branching on fractional variables, a variety of strategies are available, such as, strong branching (Wolsey, 1998), maximum infeasibility (Ravindran et al., 1987; ILOG S. A., 2002), minimum infeasibility, pseudo cost and pseudo reduced cost (ILOG S. A., 2002). The next big decision is which node to branch from. Several strategies are available, including depth-first search, best node first (Wolsey, 1998; ILOG S. A., 2002), last-in-first-out (Ravindran et al., 1987) and breadth first (ILOG S. A., 2002).

The most important limitation of B&B techniques is that it becomes computationally more intensive with an increase in problem size. The spatially explicit models are often measured by solution times, meaning the most computationally efficient formulations are viewed as best. It is important to take into account that although some models may not be the most efficient to solve, they enable managers to make better decisions in terms of costs, profits and other activities. If management objectives are given precedence over computational efficiency, then the limitation of B&B to solve larger problems is exposed. Hence, it becomes important to find a way to solve large integer problems in a relatively short time and that is where techniques like column generation enter into picture.
INTRODUCTION TO COLUMN GENERATION

Foresters these days work with models that help produce management plans over extremely large geographical areas composed of a large number of treatment units. The models involve multiple time periods and address a growing array of economic, ecological and societal issues. Hierarchical planning models are a better way of managing these large and complex models. Hierarchical planning models decompose large models into a set of smaller linked models. In this thesis, a Lagrangian relaxation and a modified Dantzig-Wolfe decomposition – a column generation technique – is used to solve a large forest planning model that maximizes net present value obtained from harvests.

Column generation is well-suited for these forestry problems because of the non-contiguous nature of forests. Column generation is considered to be most efficient when the problem can be decomposed into perfectly independent subproblems. In other words, the problem’s constraint structure should comprise of two distinct sets of constraints: one set that either limits the values of individual variables or limits the values of a subset of variables, and a second set that affects all the variables in the problem. In this way the problem can be decomposed into a number of independent disconnected subproblems. Forests often consist of many noncontiguous compartments where each compartment is composed of several units that are mapped according to geographic information system boundaries. As discussed earlier, the constraint structure in most real-world forest management problems is so diverse that the problems can be easily decomposed into several disjoint models.
Because of the sheer size of forest-level problems, owing to multiple periods, multiple forest types, multiple site classes and multiple age-classes, hierarchical planning models were developed and solved as compartment-level problems. Forest-level problems were formulated and solved separately. Weintraub and Bare (1996) provide an extensive review of work defining the hierarchical model used in forest planning. Lately, the addition of adjacency constraints in spatially explicit models has made it difficult to formulate and solve even a single forest-level problem. The landscape pattern that results from cutting plans has evolved into a critical issue in forest management. With an increasing focus on spatial harvesting and clearcut opening limitations, incorporation of adjacency constraints within the framework of forest management models has almost become mandatory.

Instead of developing separate planning models, this thesis demonstrates that the forest-level models can be formulated as a single model and then decomposed into a number of independent disjoint subproblems. In order to decompose the forest into different compartment-level subproblems, a decomposition technique called Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960) is used. Enroute, concepts of Lagrangian relaxation (Geoffrion, 1974) and price-directed decomposition will also be used. Lagrangian relaxation is a popular approach wherein complicating constraints like forest-wide constraints are relaxed and penalized in the objective function via Lagrangian multipliers. On the other hand, price-directed decomposition in a hierarchical production planning model coordinates the planning of several noncontiguous, spatially-defined management compartment problems linked by forest wide constraints.
In the past, many researchers have observed that column generation is a powerful decomposition technique for solving a wide range of industrial problems to optimality or to near-optimality. Column generation is a mathematical programming technique that can be used to solve large-scale optimization problems. In colloquial terms, column generation is a way of beginning with a small, manageable part of a problem (specifically, a few of the variables), solving that part, analyzing that partial solution to determine the next part of the problem (specifically, one or more variables) to add to the model, and then re-solving the enlarged model. Column-wise modeling is an iterative procedure that keeps iterating until it achieves a satisfactory solution to the whole problem.

In formal terms, column generation is a way of solving a linear programming problem that adds columns (corresponding to constrained variables) during the pricing phase of the simplex method of solving the problem. Generating a column in the primal simplex formulation of a linear programming problem corresponds to adding a constraint in its dual formulation. Its effectiveness depends on the extent to which a problem can be decomposed into relatively independent and easy-to-solve subproblems.

In applications, constraint matrices of (integer) linear programs are typically sparse and well structured. Subsystems of variables and constraints appear in independent groups, linked by a distinct set of constraints and/or variables. The general idea behind the decomposition paradigm is to treat the linking structure at a superior, coordinating level and to independently address the subproblem(s) at a subordinated level, exploiting their special structure algorithmically.
RESAHRCE PURPOSE AND OBJECTIVES

The goal of this research is to develop a solution methodology that will help in solving large-scale forest management IP models. There are heuristic techniques such as genetic algorithms and simulated annealing for solving large optimization problems. However, the solutions obtained with heuristics are generally sub-optimal and often bounds indicating the quality of these solutions are not provided by the algorithm. Also heuristic solution methodologies often are not easily scaleable for any new constraints that may be added in the future as the requirements of the forest management planning change.

The key goal of this research will be to develop an algorithm to implement a column generation procedure for forest management problems. Column generation is a much more complicated technique than branch-and-bound because of its iterative approach. In order to implement column generation, it is virtually essential to have the software that can do all the tedious steps during the iteration stage. Column generation involves two main steps: 1) initiate the solution process with an initial starting problem, and 2) iteratively modify the initial problem based on the solution characteristics of the problem at the previous step. The integrated column generation software is developed to implement a column generation algorithm for forest management problems which consolidates LP, MIP and solver operations on a unified platform. The initial subproblem creation and iterative master problem procedures will be directed by dual prices in previous iterations.
This study focuses on a real-world forest in District-8 (Kittanning) in the state of Pennsylvania which is managed and maintained by the Pennsylvania DCNR Bureau of Forestry. As discussed earlier, an objective of achieving an uneven but regulated age class distribution in the forest provides considerable flexibility for treatment unit design and a great opportunity to plan and engineer the spatial structure of the future forest. Realistic treatment unit delineation is a critical issue to address in order to implement a spatially explicit planning model.

Figure 1-3: District-8 pre-delineated forest structure
Hence before starting with column generation, processing of forest data in terms of treatment unit delineation is of utmost importance. There are several elements or objectives that can be used to locate treatments and how foresters define treatment unit boundaries. Petroski (2006) did an extensive survey in a forestry community regarding the treatment unit parameters. A series of interviews, both qualitative and quantitative, were conducted with BOF foresters in order to judge the key factors influencing the foresters to decide treatment unit boundaries. Petroski then developed a treatment unit delineation model (TUDM) which has the ability to target hard boundaries such as abrupt age differences, major roads, trails, rivers, pipelines, and existing deer fences. This software was used to delineate the District-8 forest on the basis of the hard boundaries mentioned above. In addition to buffer zones, hard boundaries can also be used to further decompose the forests into more compartments. Figure 1-4 shows a few of the highlighted hard boundaries used during the delineation process. Hence, it can be clearly seen that the structure of the forest provides us with different combinations of compartment sizes which extend important flexibility in formulating subproblems while implementing column generation. Since hard boundaries are mostly created from roads, trails, rivers, pipelines etc., adjacency constraints are also automatically taken care of in the subproblem formulation as units on either side of the road or river cannot be necessarily termed as adjacent units. On comparing figure 1-3 and 1-4, it can also be seen that the delineated treatment unit layer is much more uniform and free of discrepancies.
Goals of this research work can be summarized as below:

- Combine Lagrangian Relaxation and Dantzig-Wolfe Decomposition to decompose the forest.
- Develop an algorithm to implement the iterative column generation procedure for forest management problems.
- Identify the ways of generating an initial set of columns for the initial restricted master problem.
- Develop seamless integrated software in .NET that consolidates LP, IP, Solver and column generation on a unified platform.
- Compare the quality of the solution created by the column generation and branch-and-bound.
- Analyze the influence of input parameters like type and number of initial seeds on output parameters like Restricted Integer Master Problem (RIMP) objective function, total solution time taken by column generation, iterations to converge, number of subproblems in RIMP, number of identical subproblem during initialization and solution time taken by RIMP.
Chapter 2

LITERATURE REVIEW

ORGANIZATION OF THE CHAPTER

Operations research techniques have been applied successfully to solve forest resource management problems for over four decades, and the evolution of these applications during this period has been tremendous (Rajasekaran, 2005). Over this time, forest management has shifted emphasis from the production of timber resources and other products to ecosystem management (Weintraub and Bare, 1996). Forest management plans designed to maximize sustained and preferably constant harvest volume flows have given way to sustainable forest management that is compatible with the Burndtland Commission’s view of sustainable development (World Commission on Environment Development, 1987). Operations research has unquestionably influenced forest management. For example, Kent et al. (1991) describes the inclusion of the FORPLAN system in government legislation in the United States. Forest management is slowly evolving and broadening into ecosystem management, and that has become a significant challenge to operational researchers as problems have become more and more complex with the inclusion of non-timber objectives and goals. This chapter highlights some of the key developments during this evolution. When it comes to research in the fields of forest management, harvest scheduling and operations research, there is a huge
pool of literature to explore. In this chapter, the review of literature is done in a step by step manner, touching the most important topics relevant to the research.

**HARVEST SCHEDULING**

Forest management, once the sole domain of professional foresters, is increasingly influenced by diverse groups, aggressively competing to have management decisions satisfy often conflicting objectives (Martell et al., 1998). In the past, most forest management planning focused primarily on harvesting timber. Most of the models ignored long-term sustainability issues beyond timber sustainability and hence were unable to guarantee maintenance of recreation opportunities, water quality, and other objectives (Murray et al., 2004). Harvest Scheduling Methods (HSMs) are an important forest planning tools because of their ability to maintain productivity without compromising non-timber concerns and to provide data necessary for important management decisions (Murray and Weintraub, 2002).

The United States Department of Agriculture (USDA) Forest Service developed one of the most widely used mathematical planning models. It is, in fact, a series of programs, with the first one, called the Timber Resource Allocation Model (Timber RAM), developed in 1971 (Navon, 1971). Timber RAM was based on the Model 1 formulation described by Johnson and Scheurman (1977). Over the years, Timber RAM evolved into a new program called FORPLAN, which integrated both timber and non-timber objectives (Johnson et al, 1986). As forest management slowly evolved into
ecosystem management, FORPLAN evolved into the latest planning model in the series, SPECTRUM (USDA Forest Service, 1995).

Johnson and Scheurman (1977) formulated LP models to solve forest harvest management problems that satisfy the objective of achieving maximum profit. They categorized the formulations into two types, called “Model I” and “Model II.” Analysis areas, the basic unit of land in these models, are based on aggregate areas of forest land that have similar characteristics, such as forest type, site class, initial age class, etc. Analysis areas are not typically contiguous. Model II, as described by Johnson and Scheurman (1977), dealt with analysis areas by assigning them to a variable until they are cut. Once regenerated, the model re-aggregates all the areas with common forest type and site class that were regenerated in the same period into new analysis areas in order to simplify the modeling of a forest management problem. Model I does not re-aggregate analysis areas, and each area is assigned to a single variable throughout the planning horizon. The Model I approach can be simpler in comparison to Model II but can also result in larger problem formulations which might be difficult to solve. Timber RAM used a Model I approach wherein harvest fluctuation constraints and average ending age constraints were added to meet planning requirements.

With foresters becoming more aware of the need to address non-timber objectives in addition to obtaining maximum profits from timber harvesting, many non-timber issues were modeled and included in the LP problems. Thompson et al. (1973) formulated models that allow timber harvesting subject to constraints that provide certain
types of wildlife habitat conditions. Dykstra (1984) explains how Model I can be modified to incorporate wildlife management, land use planning, and soil loss concerns.

LP models are powerful, simple and easy to solve. But one of the major drawbacks of these models is their limited ability to recognize the spatial characteristics of forests. As a result, it is difficult to incorporate wildlife habitat objectives and road-fence construction constraints in LP formulations of HSMs. Also since government and other management agencies started emphasizing the aesthetic aspect of forest management; very strict spatial restrictions have been imposed on the cutting of contiguous treatment units. In order to address all the above mentioned issues, site-specific IP models have come into existence. Most of the IP models use binary (0-1) variables that indicate site-specific prescriptions that are implemented on an entire unit or not implemented at all.

Due to aesthetic concerns related to large harvest openings, regulations have been imposed to limit the maximum allowed contiguous area harvested within a certain amount of time, also known as a green-up period (Thompson et al., 1973; Jones et al., 1991; Barrett and Gilles, 2000; Boston and Bettinger, 2001; McDill et al., 2002). Such regulations include those developed by the Sustainable Forestry Initiative® (2004), which imposes a 120-acre maximum average harvest opening size limit, and the Pennsylvania Bureau of Forestry’s (2000) limit for maximum harvest opening sizes of 50 acres for fenced harvests and 70 acres for un-fenced harvests. This gave rise to the adjacency constraints in the formulations that restrict contiguous harvest areas to be
below a maximum harvest opening size (Torres-Rojo and Brodie, 1990; Jones et al., 1991; Gunn and Richards, 2005).

IP models tend to be larger than LP models, and they are generally much harder to solve. HSMs that include adjacency constraints are often very large IPs. Due to their difficulty, forest planners have contributed various algorithms to solve these IPs using heuristic methodologies such as TABU search (Boston and Bettinger, 1999; Clark et al., 2000; Richards and Gunn, 2000), artificial intelligence (Hokans, 1983), simulated annealing (Lockwood and Moore, 1993), Monte Carlo integer programming (Barret et al., 1998), Random Search (Nelson and Finn, 1991), genetic algorithms (Lu and Eriksson, 2000; Falcao and Borges, 2001), simulation (Yoshimoto et al., 1994), and dynamic programming (Hoganson and Borges, 1998). Heuristic procedures can be good methods to solve tactical forest planning problems and are often faster than exact approaches. But they have the very significant drawback of being generally unable to measure the quality solutions by providing upper and lower bounds on the objective function value (ILOG S. A. 2006), unlike exact approaches, such as branch-and-bound. Exact solution approaches also have the advantage of using off-the-shelf computer optimization software such as ILOG CPLEX. Most commercial solvers, like LINDO/LINGO, GAMS and CPLEX, use variations of the branch-and-bound algorithm to solve MIP and IP problems.

The adjacency constraints mentioned earlier play a critical role in HSMs, as they directly affect the size of the models and their solution times. They can also affect other constraints present in the constraint structure and harvest unit delineations. Two classes of adjacency formulations are Unit Restricted Models (URM) and Area Restricted
Models (ARM; Murray, 1999). URMs prevent the harvest of any two contiguous treatment units, while ARMs allow groups of contiguous units to be harvested unless the combined area of the units is larger than the maximum harvest opening size. The ARM was initially thought to be very difficult; Murray (1999) stated that the non-linear characteristics of the ARM would make it difficult to solve by exact solution approaches. A variety of linear ARM models have since been formulated (e.g., McDill et al., 2002, Goycoolea et al., 2004) but formulations tend to be quite large and solving them remains a challenge.

Goycoolea et al. (2004) noted that the addition of binary variables and adjacency constraints in the IP models makes them NP-hard. A problem is NP-hard if an algorithm for solving it can be translated into one for solving any NP-problem (nondeterministic polynomial time problem). NP-hard therefore means “at least as hard as any NP-problem,” although it might, in fact, be harder. In order to overcome the difficulty of solving HSMs with adjacency constraints, McDill and Braze (2000), tested numerous URM constraint formulations. They classified formulations into five different categories: Pairwise, Type I, Aggregated, Tightened Coefficient and Minimized constraints. McDill et al. (2002) showed that it is possible to formulate ARM constraints as linear constraints. They presented two exact, linear formulations of ARM constraints, the “PATH ARM” and the “Generalized Management Units (GMU)” approach. The PATH algorithm identifies groups (“paths”) of contiguous management units that just exceed the maximum harvest area and then creates constraints that prevent all the units in the entire set from being harvested concurrently. GMUs are groups of contiguous management
units that have a combined area less than the maximum harvest size. The approach creates new variables representing the treatment of units in a GMU set. URM constraints are then applied to the GMU variables.

**DANTZIG-WOLFE DECOMPOSITION AND LAGRANGIAN RELAXATION**

According to Wolsey (1998), the constraints can be categorized into either a *Convex Constraints* or *Joint Constraints*. *Convex Constraints* are those constraints that only apply to a subset of the variables in the original problem. Another set of constraints called *Joint Constraints* are the constraints that link all or nearly all the variables in the original problem.

Dantzig-Wolfe (D-W) decomposition (Dantzig and Wolfe, 1961) and Lagrangian relaxation (Geoffrion, 1974) are two alternative methods for obtaining tighter lower bounds on the optimal objective function value. In D-W decomposition, the original problem is reformulated by substituting the original variables with a convex combination of the extreme points of a subproblem formulation. The resulting formulation is called a master problem. Column generation is used to solve this master problem which contains a small subset of columns initially. Solving a relaxed master problem provides values of the dual variables corresponding to joint constraints that are further used to generate new columns with negative reduced cost. If one is found, then that column is added to the relaxed master. Lagrangian relaxation dualizes the complicating/linking constraints in the objective function by penalizing their violations through Lagrangian multipliers. It is an iterative procedure where the Lagrangian multipliers are updated in each iteration using
different methods such as the subgradient algorithm (Fisher, 1985), the bundle method (Lemarechal, Nemirovskii and Nesterov, 1995), or the volume algorithm (Barahona and Anbil, 2000).

There exists a strong relationship between D-W decomposition and Lagrangian relaxation. It is well known that when the Lagrangian relaxation is obtained by dualizing exactly those constraints that are the linking constraints in the D-W reformulation, the optimal values of the Lagrangian Dual and the LP relaxation of the D-W reformulation are the same (Huisman et al., 2003). In fact, one formulation is the dual of the other (Geoffrion, 1974; Fisher 1981). Furthermore, the optimal dual variables for the linking constraints in the master problem correspond to optimal multipliers for the dualized constraints in the Lagrangian relaxation (Magnanti, Shapiro and Wagner, 1976). Moreover, the subproblems that we need to solve in the column generation procedure are the same as the ones we would have to solve for the Lagrangian relaxation except for a constant in the objective function. The difference between column generation and Lagrangian relaxation is that in column generation, the values for the dual variables are obtained by solving the LP relaxation of the restricted master and used as the multipliers for the next iteration whereas, in Lagrangian relaxation, the Lagrangian multipliers are updated by subgradient optimization.

Depending on whether the problem is a minimization or maximization problem, Lagrangian relaxation provides a lower bound or upper bound respectively on the optimal IP value of the original problem, whereas the D-W reformulation provides a primal solution at each iteration. Fisher (1985) indicated convergence problems with the
subgradient algorithm to estimate Lagrangian multipliers as it is stopped after a fixed number of iterations without guaranteeing the optimal solution to Lagrangian dual. Lagrangian relaxation is not a computationally intensive approach, unlike D-W where simplex optimization of the subproblems to formulate the master problem can be expensive. Also slow convergence towards the end of the column generation algorithm has been observed (Barnhart et al., (1998); Vanderbeck and Wolsey, (1996)).

Vielma et al. (2006) considered a problem with area restricted adjacency and harvest flow constraints in a multi-period planning horizon. Vielma et al. (2006) found that using harvest flow constraints considerably slows down the problem solution time because of excessive fractioning during the integer programming branch and bound procedure. To overcome this they apply a penalty approach to the harvest volume constraints, updating the penalties applied to the volume constraints by adding hyperplane cuts to the constraint set during the solution of the integer programming problem. The authors were able to find close approximate solutions to problems which they otherwise were unable to solve with strict harvest volume constraints.

Wolsey (1998) also noted that if one drops the “complicating constraints” in a formulation then the resulting relaxation is easier to solve than the original problem. Many problems have such a structure; for example, if one drops the connectivity constraints in the traveling salesman problem, and if one drops the client demand constraints in the uncapacitated facility location problem. However, the resulting bound obtained from the relaxation may be weak, as some important constraints are totally ignored. This can be avoided by using a popular approach called Lagrangian relaxation
which relaxes complicating constraints and penalizes their violation in the objective function via Lagrangian multipliers.

The most popular – because it is very easy to implement – and well-documented way to obtain (near-) optimal multipliers is the subgradient algorithm (Wolsey, 1998). An alternative way is by linear programming. Another way of penalizing the objective function is by solving the linear version of the same problem, obtaining the shadow prices of a complicating constraint to be dualized and then again bringing back integrality constraints in the formulation and using obtained shadow prices as Lagrangian multipliers.

Huisman et al. (2003) noticed that there exists a strong relationship between Dantzig-Wolfe decomposition and Lagrangian relaxation. They advocate dualizing the joint constraints in the D-W reformulation. The optimal value of this formulation is the same as that of the Lagrangian dual. One can then use the optimal dual variables of the relaxed master problem as multipliers to penalize joint constraints in the objective function of the subproblem. Huisman et al. (2003) show two ways to exploit strengths of both the methods by combining them efficiently. Lagrangian relaxation can be applied to the master problem to approximate optimal values of the dual variables or it can be used on the original formulation of the problem to generate good columns.

Cattrysse et al. (1993) used Lagrangian relaxation to approximate the optimal dual variables of the restricted master problem instead of using the simplex algorithm. They apply this technique to solving a variant of the Capacitated Lot-Sizing Problem. A similar
integration of D-W decomposition and Lagrangian relaxation is also used for the
generalized assignment problem (Cattrysse, Salomon and Van Wassenhove, 1994). Jans
and Degraeve (2004) indicate that approximating Lagrangian multipliers from optimal
dual variables indeed speeds up convergence and decreases the problem of degeneracy.

Barahona and Jenessn (1998) first proposed the approach of formulating
subproblems by dualizing the joint constraints in the original problem and solving the
subproblems to obtain the columns that can be added to the master problem. They used a
plant location problem to illustrate the technique. Degraeve and Peeters (2003) apply the
same technique to the cutting stock problem. It has also been applied successfully to the
capacitated lot-sizing problem (Degraeve and Jans, 2003; Huisman et al., 2003). Further,
Lobel (1998) and Fischetti and Toth (1997) refer to the combination of column
generation and Lagrangian relaxation.

APPLICATION OF DECOMPOSITION TECHNIQUES IN FOREST MANAGEMENT

The spatial nature of forests and the buffer zones created between different groups
of stands by roads, streams, and trails give the forest management problem an ideal
structure for applying decomposition techniques. Several authors have developed
different techniques to link various planning models in an organized fashion. Weintraub
and Cholaky (1991) developed an iterative model that solves compartment level
problems to meet harvest volume targets specified by a forest-level problem. Volume
targets are varied to solve the forest-level problems until feasible solutions at the
compartment level are found to be within prespecified tolerance. Bare and Liermann
(1994) present a similar model structure, which is spatially decomposed and utilizes similar aggregation procedures. Hof and Pickens (1987) present a two-tiered model in which several compartment level forest plans are proposed for each compartment.

Hoganson et al. (1984) developed an approach similar to D-W decomposition (1960) to solve large-scale HSMs. Bazaara and Jarvis (1997) suggested that in certain cases the structure of the problem can allow it to be decomposed to make the solution methodology easier. Column generation can be an effective tool for decomposing and solving large scale IP problems (Wolsey, 1998). It is also quite similar to the D-W decomposition algorithm (Dantzig and Wolfe, 1960). The pioneering work of Gilmore and Gomory (1961) on the cutting stock problem demonstrates the strength of the LP relaxation of the master formulation involving a large number of columns. Minoux (1987) showed how several important combinatorial optimization problems can be reformulated and tackled by column generation. Johnson et al. (1993) used a column generation approach to solve a graph partitioning problem. Several other authors have applied D-W decomposition to forest management problems. Nazareth et al. (1980) used D-W decomposition to solve large-scale LP forest management problems where the approach was quite similar to a cutting stock problem. Problems involving set partitioning, set covering and set packaging have a structure that can be easily decomposed, and decomposition techniques have been applied to these problems (Etchberry, 1977; Bramel and Simchi-Levi, 1998; Hoffman, 2005).

Perhaps the airline industry is the one that uses column generation most extensively. Large airline staffing problems, also known as scheduling problems and
multi-commodity network flow problems, have been tackled by column generation (Barnhart et. al., 1991; Desrochers and Soumis, 1989; Anbil et al., 1991 and Desauliniers et al., 1998). Crew recovery problems are another type of problem in the airline industry that can be solved using the column generation approach (Teodorovic, 1990). The set covering structure of delivery and routing problems, scheduling problems and location problems have found wide application for column generation (Hoffman, 2005). Numerous applications in routing and scheduling are covered in a survey by Desrosiers et al. (1994). Barnahart et al. (1994)’s survey presents a unified review of general IP column generation ideas that have appeared in different contexts.

Berck and Bible (1984) demonstrate that the formulations proposed in Johnson and Scheurman (1977) can be solved using D-W decomposition. They showed that the decomposition method took less than half the time taken by the revised simplex method. Hoganson and Rose (1984) also demonstrate decomposition of the forest planning problem. Paredes (1995) provides an excellent conceptual interpretation of D-W decomposition applied to a forest management problem from an economic viewpoint.

Weintraub et al. (1994) applied a column generation algorithm to the HSM. They propose a decomposition method to solve forest management problems with joint constraints and compartment level constraints that include adjacency constraints. After the first iteration, the restricted master problem is solved, and the solution is improved by generating more columns that can be added to the master problem in next iteration. However, they do not discuss the initialization process for the initial restricted master problem. Bazaraa and Jarvis (1977) explain in detail the method to generate an initial
basis, using either the Big M method or the Phase-I method (Chavatal, 1983). In the case of IP problems, a good starting set of columns is very important for the algorithm to converge quickly. This research describes an algorithm to generate the basis for the master problem. Details on initialization, especially for mixed integer programs, are given by Vanderbeck (1994; 2005).
Chapter 3

METHODOLOGY

Dantzig-Wolfe (D-W) decomposition and column generation, devised for LPs, is a success story in large-scale integer programming (IP). In applications, constraint matrices of IPs are typically sparse and well structured. Subsets of variables and constraints can often be recognized as independent disjointed groups linked by a distinct set of constraints and variables. Multi-commodity flow formulations for vehicle routing and crew scheduling problems are well-known examples (Desaulniers et al., 1998; Desrosiers et al., 1995). The general idea behind the decomposition paradigm is to treat the linking structure at a superior, coordinating level and to independently address the subsystem(s) at a subordinated level, exploiting their special structure algorithmically (Lubbecke et al., 2003).

In the real world, foresters desire to achieve a variety of different objectives, but the two most common objective functions used in the forest harvest scheduling models (HSMs) are, (1) minimizing the cost of operations involved in achieving a certain level of timber production, and (2) maximizing the Net Present Value (NPV) or Profit of the forest. This research primarily focuses on models that maximize the discounted profits from the forest. Typically, several other objectives or goals need to be achieved during a planning process, but most of these are included in the constraint structure of the mathematical model. The profit maximization formulation of the problem gives much more flexibility in specifying harvest targets. Unlike cost minimization formulations, it is
not even necessary to designate specific harvest targets with the profit-maximizing formulation. Hence a solution methodology based on column generation is developed here for models with a profit maximization objective function.

**NPV MAXIMIZATION MODEL**

The objective of this model is to maximize the net present value of the forest. The model yields the optimally maximized NPV of the forest within three sets of constraints. This model can also be used to establish a reasonable target volume of wood that can be produced, which can be later used in specifying harvest targets for a cost minimization model. The model constraints consist of logical constraints, adjacency constraints, harvest accounting constraints, and volume flow constraints. The basic model formulation is as follows:

\[
\text{Max } Z = \sum_{m=1}^{M} A_m \left[ c_{m,0} X_{m,0} + \sum_{t=h_m}^{T} c_{m,t} X_{m,t} \right]
\]  

subject to:

\[
X_{m,0} + \sum_{t=h_m}^{T} X_{m,t} \leq 1 \quad \text{for } m = 1, 2, \ldots, M
\]  

\[
\sum_{m \in P} X_{m,t} \leq n_p - 1 \quad \text{for all } p \in P \text{ and } t = h_p, \ldots, T
\]  

\[
\sum_{m \in M_h} v_{m,t} A_{m,t} X_{m,t} - H_t = 0 \quad \text{for } t = 1, 2, \ldots, T
\]  

\[
b_{t,h} H_t - H_{t+1} \leq 0 \quad \text{for } t = 1, 2, \ldots, T - 1
\]  

\[-b_{h,t} H_t + H_{t+1} \leq 0 \quad \text{for } t = 1, 2, \ldots, T - 1
\]
The decision variables are:

\[ X_{m,t} \geq 0 \quad \text{for } m = 1, 2, \ldots, M \text{ and } t = h_m, \ldots, T \]  

**Formulation A-1: Original Problem**

The decision variables are:

\[ X_{m,t} = \text{A binary variable whose value is 1 if prescription } t \text{ is to be applied to management unit } m; \text{ for } t = 1, 2, \ldots, T, \text{ prescription } t \text{ means that the management unit will be harvested in period } t; \text{ when } t = 0, \text{ the prescription is to not harvest the management unit at all during the planning horizon (i.e., } X_{m,0} \text{ represents the “do-nothing” alternative for management unit } m), \]

The problem data are:

\[ M = \text{the number of management units in the forest that are available for harvest during the planning horizon,} \]

\[ T = \text{the number of periods in the planning horizon,} \]

\[ c_{m,t} = \text{the discounted net revenue per hectare if management unit } m \text{ is harvested in period } t \text{ plus the discounted residual forest value based on the state of the management unit at the end of the planning horizon,} \]

\[ h_m = \text{the first period in which management unit } m \text{ is old enough to be harvested,} \]

\[ A_m = \text{the area of management unit } m \text{ in hectares,} \]

\[ v_{m,t} = \text{the volume of sawtimber in m}^3/\text{hectare harvested from management unit } m \text{ if it is harvested in period } t, \]

\[ H_t = \text{a continuous accounting variable giving the total volume of saw timber in m}^3 \text{ harvested in period } t, \]
\[ M_h = \text{the set of management units that are old enough to be harvested in period } t, \]
\[ P = \text{the set of indexes corresponding to the management units in path } i, \]
\[ n_p = \text{the number of management units in path } i, \]
\[ h_i = \text{the first period in which the youngest management unit in path } i \text{ is old enough to be harvested,} \]
\[ b_{l, t} = \text{a lower bound on decreases in the harvest level between periods } t \text{ and } t + 1, \]
\[ b_{h, t} = \text{an upper bound on increases in the harvest level between periods } t \text{ and } t + 1. \]

Equation (1) specifies the objective function of the problem, which is to find the maximum possible discounted net revenue from the forest. Constraint set (2) consists of logical constraints that allow only one prescription to be assigned to a management unit during the planning horizon. Constraint set (3) comprises the maximum harvest opening size constraints generated with the path algorithm (McDill et al. 2002). These constraints prohibit the concurrent harvest of adjacent management units only if the combined area of contiguous harvest units exceeds the maximum harvest opening size. A “path” is a group of contiguous management units whose area just exceeds the maximum harvest opening size. The set of all such paths in a forest is generated with the Path Algorithm of McDill et al. (2002). Constraints are written for each path for each period in which all of the management units in the path are old enough to be harvested.

Constraint set (4) comprises harvest accounting constraints that assign the harvest volume for each period to the harvest accounting variables \( H_t \). In the case of cost-minimization problem, harvest constraints are absolutely necessary. Without them, the
cost-minimizing solution would be to harvest nothing. With a profit maximization objective, we could exclude the harvest constraints completely and the model would determine the harvest levels that produce the maximum profit. Hence we have much more flexibility in specifying harvest constraints with the profit maximization formulation. However, if the model is given too much latitude in setting harvest levels, the harvest level may fluctuate wildly from one period to the next. So a new set of harvest constraints that do not limit the level of the harvest is typically included in profit-maximization models. These constraints, known as the harvest fluctuation or volume flow constraints, limit the amount that the harvest level is allowed to fluctuate from one period to the next. Constraint set (5) and (6) are volume flow constraints.

The profit maximization formulation of the problem gives more flexibility in how the harvest targets must be specified in the problem. On the other hand, the disadvantage of the profit maximization formulation is that it requires you to project the price of wood for the duration of the planning horizon. In practice, it is often easier to predict future wood needs or the productive capacity of the forest than to predict future wood prices. Also, in addition to the constraints used for this research a variety of other constraints could be used in the formulation that could qualify as either compartment-level or forest-wide constraints. Ending age, extended rotations, wildlife opening, road and fence constraints are few examples of additional constraints that can be used. Nevertheless, for this research, it was sensible to keep the original formulation as simple as possible. And hence the constraint structure comprise of three distinct sets of constraint – logical constraints (limit the treatments that can be assigned to each individual treatment units),
adjacency constraints (limit harvests to small groups of contiguous treatment units), and
volume flow constraints (limit all the treatment units in the forest at one time by limiting
fluctuations in the total harvest volume). Hence, the distinctive properties of each
constraint set provide the constraint structure that is needed for this research.

**DANTZIG-WOLFE DECOMPOSITION AND COLUMN GENERATION**

Column generation (Dantzig, 1961) provides a mechanism for iterative resource
price determination in response to problems generated for each separate compartment.
The prices are normally determined by solving a master problem, except in the initial
iteration where the master problem does not exist. In this case either one can take
shadow prices to be zero and ignore the forest-wide constraints during the first iteration
or one can start the iterations with warm-up artificial variables.

The column generation is an iterative algorithm and comprised of the following
basic steps:

- Relax the joint constraints and penalize their violations in the
  compartment level subproblems with the help of shadow prices obtained
  from solving the original LP problem or just ignore the joint constraints
  and solve the compartment level subproblems with convex constraints.

- In the case of using shadow prices from the original LP, find a set of
  feasible solutions for each compartment by perturbing the shadow prices.
  These solutions will satisfy the logical and adjacency constraints for the
treatment units in the compartment.
- Substitute the adjusted variable coefficients of the compartment-level problems in the original objective function, solve the compartment-level problem, and use each compartment-level solution in the form of variables coefficients in a master problem.

- Check the convergence of shadow prices of the joint constraints in the current iteration with those from the previous iteration.

- If the shadow prices have not converged then develop additional feasible solutions at the compartment level using the shadow prices from the last iteration.

- Continue the iterative procedure until the shadow prices converge. If shadow prices don’t converge then an alternate stopping rule will be to stop the algorithm after pre-specified number of iterations.

This algorithm for solving the forest management problem mimics the column generation routine and is shown in the form of a flow chart in Figure 3-1.
Initialization Step: Solve LP Version of Original Problem & record the shadow prices of joint constraints

Seeding Step: Perturb the shadow prices and create a set of columns for each compartment

Initial Column Generation Step: Use the set of shadow prices to generate the Columns. Columns are obtained by solving compartment-level problems that satisfy only the convex constraints.

Master Problem Formulation Step: Combine adjusted variables and unadjusted original objective function coefficient to get the solution which will be added as a variables in Master Problem

Master Solution Step: Solve the LP version of Master Problem

Check whether shadow prices in Master Problem have converged

Record the shadow prices of the joint constraints in Master Problem

Solve IP version of the Master Problem

STOP

Figure 3-1: Column generation algorithm
The forest used in this study (Figure 3-2) is Pennsylvania’s Kittanning State Forest (Pennsylvania Department of Conservation and Natural Resources (DCNR), Bureau of Forestry (BOF), District-8). A typical Pennsylvania forest has many roads, streams, and trails running through it. The areas adjacent to these features will not be cut so this creates natural compartments that are spatially independent of each other. This
spatial independence of the compartments can be exploited to implement the column generation algorithm efficiently for large-scale forest management problems. This forest structure can be seen in Figure 3-2, where buffer zones separate different segments of forestland. This structure provides a convenient way to decompose forest into a number of spatially independent compartments. Solving a problem at the compartment level is relatively easy as there are fewer management units, so the number of variables and constraints involved is much smaller and the joint constraints are absent in the formulation. In the District-8 case study used here, the forest is divided into 9 compartments. Petroski (2006) showed that forests can be delineated more uniformly on the basis of roads, streams, trails, pipelines, fencing etc. The boundaries created were called “Hard Boundaries.” The forest can also be further divided into more compartments on the basis of these hard boundaries.

**CONSTRAINT STRUCTURE IN THE FORMULATION**

The constraint structure in any formulation can be categorized into two types of constraints – convex constraints and joint constraints (Wolsey, 1998). Convex Constraints are those constraints that only apply to a subset of the variables in the original problem. There are two such constraint sets in the HSM used here – logical constraints and adjacency constraints. Logical constraints are written for each management unit in the forest. The variables in these constraints pertain only to individual management unit. Adjacency constraints are written for groups of contiguous management units that just exceed the maximum allowable harvest size. Hence the variables in these constraints also pertain only to relatively small sets of contiguous units.
Hence convex constraints restrict how a particular management zone or compartment is managed without affecting other compartments. Another set of constraints called Joint Constraints are the constraints that link all or nearly all the variables in the original problem. In this case both harvest accounting constraints and volume flow constraints link all the compartment and units in the forest together to satisfy requirements over the entire forest. Hence they can be treated as the Joint Constraints.

For example, in Figure 3-2, it can be seen that harvesting a treatment unit in compartment 1 has no effect in terms of logical or adjacency constraint violations on treatment units in compartment 2, since compartment 1 does not share a border with compartment 2. However, volume flow constraints consider harvests from all compartments and units and hence link the entire forest.

**EXPLOITING CONSTRAINT STRUCTURE AND LAGRANGIAN RELAXATION**

The column generation technique involves an iterative stage of representing solutions to the subproblems that satisfy the convex constraints as variables in another problem called the master problem. The master problem needs to satisfy the joint constraints which are formed using the variables that represent solutions of the subproblems. The first step in the algorithm is to create the initial subproblems which will be later modified using the shadow prices from the original problem.

There are many ways to create the initial seeds from the subproblems. A seed here is an array of optimal variable values of each subproblem that will be substituted in the master problem in the form of subproblem solutions. Column generation algorithm will
then generate master problems which mimics the problem structure of original problem.

In a cost minimization problem there are harvest target constraints and the harvest targets for these constraints can be found by solving the NPV maximization problem for the same forest and determining the volume produced by each compartment in each period. In order to get different subproblem solutions for seeding the master problem, these harvests can be randomly varied. However, this approach is not possible in profit maximization problems, as harvest target constraints in cost minimization problems are replaced with volume flow constraints that limit the fluctuation of harvest volumes between periods.

In the profit maximization formulation of the HSM, volume flow constraints are the joint constraints that affect all units and compartments in the forest and they are also the complicating ones in terms of solution times. Hence it is best to relax these constraints and penalize them in the objective function for the subproblem formulation. This seems to be the best approach since this way the subproblem formulations are free of volume flow constraints pertaining to the individual compartments and at the same time the violation of these constraints are suitably penalized in the objective function. Hence, the resulting compartment level problems will be easier to solve in terms of computational times, which in turn will also provide us with more flexibility in generating more subproblem seeds.

The volume flow constraints \((Joint\, Constraints)\) are taken as the basis to dualize the compartment level problems. In order to dualize the problems, the harvest accounting variable \(H_t\) is substituted in the volume flow constraint to make it just one constraint set.
These constraint sets (8) and (9) are now dualized in the objective functions of each compartment. These constraints are dualized using the corresponding shadow prices obtained by solving the LP version of the original problem. An important thing to notice here is that while solving the original problem, only one of the two volume flow constraints will be binding. That is the constraint that limits the amount of decline in the harvest and the constraint that limits the amount of increment cannot both be binding at the same time. Hence the formulation will always yield at least one non-binding constraint with a positive slack. However each binding constraint will have a non-zero dual price.

Let us denote the shadow prices related with the declining constraint by $\lambda^d_t$ and those related with the rising constraint by $\lambda^u_t$. The shadow prices associated with binding harvest fluctuation constraints indicate how much additional profit could be earned if one more m$^3$ could be shifted from one period to other. Hence if the value of harvest accounting variable $H_t$ is greater than $H_{t+1}$, then the binding constraint will be the one that says that the harvest cannot decline by more than certain percentage. Shadow prices on the binding constraint say that many additional dollars of profit could be made if one more m$^3$ of wood could be produced in period 1 instead of period 2. The dualization of
the volume flow constraints and the corresponding modification of the objective function
is as follows:

$$\text{Max } Z_{\text{adjusted}} = \sum_{m=1}^{M} A_{m} \left[ c_{m,0} X_{m,0} + \sum_{t=h_m}^{T} c_{m,t} X_{m,t} \right]$$

$$= \sum_{t=1}^{T-1} \lambda_{i,t} \left\{ b_{i,t} \sum_{m=1}^{M_{i,h}} v_{m,t} X_{m,t} - \sum_{m=1}^{M_{i,h+1}} v_{m,t+1} X_{m,t+1} \right\}$$

$$- \sum_{t=1}^{T-1} \lambda_{u,t} \left\{ -b_{u,t} \sum_{m=1}^{M_{u,h}} v_{m,t} X_{m,t} + \sum_{m=1}^{M_{u,h+1}} v_{m,t+1} X_{m,t+1} \right\}$$

(10)

The objective function in equation (10) is decomposed into a separate objective function for each compartments $k = 1$ to 9. A slight amount of algebraic manipulation transforms equation (10) into equation (11) which is the decomposed form for each compartment.

$$\text{Max } Z_{\text{adjusted}} = \sum_{m=1}^{M} A_{m} c_{m,0} X_{m,0}$$

$$+ \sum_{m=1}^{M} \left\{ A_{m} c_{m,1} + \left( -b_{i,1} \left( \eta_{i,1_i,0} \right) + b_{u,1} \left( \eta_{u,1_u,0} \right) \right) v_{m,1} \right\} X_{m,1}$$

$$+ \sum_{t=2}^{T-1} \left( \sum_{m=1}^{M} A_{m} c_{m,t} + \left( -b_{i,t} \left( \eta_{i,1_i,0} \right) + b_{u,t} \left( \eta_{u,1_u,0} \right) \right) v_{m,t} \right) X_{m,t}$$

$$+ \sum_{m=1}^{M} \left( A_{m} c_{m,T} + \left( \eta_{i,1_i,0} \right) v_{m,T} \right) X_{m,T}$$

............. (11)
The shadow prices will be varied to generate new columns that can be added to
the master problem. Perturbing the objective function of the subproblem this way creates
different seeds which will in turn be used to formulate the master problem. As a result of
dualizing the volume flow constraints, subproblems will be formulated to satisfy only the
convex constraints and not the joint constraints. The entire compartment level
formulation for initial set of seeds is shown below:

\[
\text{Max } Z_{\text{adjusted}} = \sum_{m=1}^{M_k} A_m c_{m,0} X_{m,0} + \sum_{m=1}^{M_k} \left\{ A_m c_{m,1} + \left( -b_{t,j} \left( \eta, \lambda^d_{t,0} \right) + b_{t,j} \left( \eta, \lambda^u_{t,0} \right) \right) v_{m,1} \right\} X_{m,1} \\
+ \sum_{t=2}^{T-1} \sum_{m=1}^{M_k} \left\{ A_m c_{m,t} + \left( -b_{t,j} \left( \eta, \lambda^d_{t,0} \right) + b_{h,t} \left( \eta, \lambda^u_{t,0} \right) + \left( \eta, \lambda^d_{t-1,0} \right) - \left( \eta, \lambda^u_{t-1,0} \right) \right) v_{m,t} \right\} X_{m,t} \\
+ \sum_{m=1}^{M_k} \left\{ A_m c_{m,T} + \left( \eta, \lambda^d_{T-1,0} \right) - \left( \eta, \lambda^u_{T-1,0} \right) \right\} v_{m,T} \right\} X_{m,T} \\
\]............ (12)

subject to:

\[
X_{m,0} + \sum_{t=h_n}^{T} X_{m,t} \leq 1 \quad \text{for } m \in M_k \\
\] (13)

\[
\sum_{m \in P} X_{m,t} \leq n_p - 1 \quad \text{for } p \in P \text{ and } t = h_p, ..., T \quad \] (14)

\[
\sum_{m=1}^{M_k} \sum_{t=0}^{T} A_m c_{m,t} X_{m,t} - Z_{k,j} = 0 \\
\] (15)
\[
\sum_{m \in M_{kh}} v_{m,t} X_{m,t} - H_{k,i,t} = 0 \quad \text{for } t = 1, 2, \ldots, T \tag{16}
\]

\[
X_{m,t} \in \{0,1\} \quad \text{for } m \in M_k \text{ and } t = h_m, \ldots, T \tag{17}
\]

**Formulation A-2: Initial Lagrangian Subproblem**

Where,

\( Z_{k,i} \) = unadjusted objective function coefficient before dualizing volume flow constraints,

\( M_k \) = set of management units in compartment \( k \),

\( M_{k,h} \) = set of management units in compartment \( k \) that are old enough to be harvested in period \( t \),

\( H_{k,i,t} \) = timber harvest contributions of compartment \( k \) for seed \( i \) in period \( t \),

\( \eta_t \) = a real, randomly generated number used vary the shadow prices of the volume flow constraints in the linear solution of original problem,

\( \lambda^r_{t,0} \) = shadow prices related to rising volume flow constraints in relaxed original problem, and

\( \lambda^d_{t,0} \) = shadow prices related to declining volume flow constraints in relaxed original problem.

Constraints (13) are the logical prescription constraints for the units in compartment \( k \). Constraints (14) are the path adjacency constraints for the unit adjacencies prevalent in compartment \( k \). Constraint (15) is the accounting constraint to
calculate the unadjusted objective function value for the subproblem solution based on
the objective function coefficients of the original problem. Note that the new objective
function for a subproblem has a subscript “adjusted” \((Z_{\text{adjusted}})\) which means that the
objective function values of these formulations are the adjusted ones. At this point, we
are only interested in the variable values assigned to each variable after the adjustments,
and those variable values (either 0 or 1) will be multiplied with the unadjusted objective
function coefficients of the original problem in the net revenue accounting constraints
(equation (15)). Hence \(Z_{k,i}\) will be the optimal subproblem solution for a compartment \(k\),
and initial seed \(i\) which will then be used in the first iteration restricted master problem
(we will call it the initial restricted linear master problem (IRLMP)). In this way, the
IRLMP objective function value will exactly mimic the original problem’s objective
function value for the corresponding variable values.

Constraints in set (16) are volume accounting constraints similar to those in the
original problem. These constraints accrue the timber harvests carried out in each
compartment every period. \(H_{k,i,t}\) tracks the harvest contributions from compartment \(k\) for
seed \(i\) in period \(t\). Since it is a non-restrictive constraint, it is not going to affect the
subproblem solution in any way, but the harvest contributions from each compartment
stored in a variable \(H_{k,i,t}\) will be used to formulate the volume flow constraints in the
IRMP. Constraints (17) are the integrality constraints.
INITIALIZATION AND GENERATION OF COLUMNS

The column generation procedure must be initialized with a feasible formulation of the initial restricted linear master problem (IRLMP). This issue can be addressed by providing an initial set of columns for the restricted master problem. This section discusses the construction of initial columns. The primary purpose of this initialization step is to provide a feasible IRLMP formulation. It should be remember that the final optimal subproblem seeds will resemble the original optimal IP problem. Hence, the aim should be to create initial seeds that are as close to the original optimal IP seed as possible. Since the original IP solution is not known, it is important to find a good way to generate the initial column generation seeds that mimics the original IP seed. Poorly chosen initial columns may lead the algorithm astray if they do not resemble the structure of a possible optimal solution at all.

The initial formulation of the IRLMP is obtained by solving the compartment-level problems shown in A-2 – i.e., the subproblems are formulated with appropriate adjusted variables and an adjusted objective function \( Z_{\text{adjusted}} \) that penalizes joint constraint violations. However, the adjusted objective function value cannot be used to formulate the master problem because of the random intervals used to generate them. It is possible that, in some cases, because of the randomness involved, the algorithm will generate columns whose objective function coefficients will be too far away from the desired optimal objective function coefficients. Instead, the original net revenue accounting constraint (equation 15) in A-2 is used to calculate the original unadjusted objective function value corresponding to a subproblem solution.
No matter what method we use, we have to initialize the Restricted Master Problem (RMP). There are many ways to do this. One way is to initially set the Lagrangian multipliers to zero in formulation A-2, essentially ignoring the flow constraints during the first iteration (Pittman, 2007). The $Z_{k,i}$ and $H_{k,i}$ variable values obtained from formulation A-2 are used to formulate the initial restricted linear master problem. The solution to the IRLMP provides Lagrangian multipliers from the shadow prices of its flow constraints that are passed back to each subproblem so that they may be solved again using these new penalties for the joint constraints. Formulation A-2 is again solved for each compartment, which provides us with new seeds to enter into the master problem. Since the number of possible feasible solutions for the original problem is huge, the number of potential variables in the master problem is also huge. As a result, this approach might require numerous iterations.

Instead, it seems to be a good idea to make use of Lagrangian multipliers in the first step itself by solving the linear relaxation of the original problem and substituting shadow prices of the joint constraints from this problem as multipliers in formulation A-2. Doing this yields a Lagrangian dual problem since $Z_{\text{adjusted}}$ is a relaxation of $Z$, $Z_{\text{adjusted}} \geq Z$. On solving $Z_{\text{adjusted}}$, we obtain an upper bound on the optimal value of $Z$. It is relatively easy to solve but does not give an optimal value for $Z$.

The adjusted function $Z_{\text{adjusted}}$, $\lambda^u$, $\lambda^d \geq 0$, is the lower envelope of a family of functions linear in either $\lambda^u$ or $\lambda^d$, and therefore is concave (Lubbecke et al., 2005). It is piecewise linear and only sub-differentiable at its breakpoints. Hence there may be some
value in exploring the optimal dual solution that exists possibly close to the upper bound. The Lagrangian multipliers or shadow prices used in formulation A-2 can be interpreted as a cost to the objective function or the amount of profit that is given up per m³ to even out the harvest flows between periods. There are two types of volume flow constraints – rising and declining ones – in the original problem and only one of them can be binding at one time. Whichever is the binding constraint, shadow prices have as much impact on the adjusted objective function coefficient as the volume flows. The shadow prices/Lagrangian multipliers can also be seen as the weights used to penalize the flow constraint in the objective function. By randomly varying the shadow prices on either side of their original values we can perturb the adjusted objective function on either side of the upper bound.

Using this approach, we can perturb the shadow prices in different ways. One of the objectives of this research is to experiment with different types of perturbing techniques so that we can find the best way to generate a set of initial seeds. The best possible result will be to find a perturbing technique that will provide near optimal seeds. If that is achieved, then the number of iterations required in column generation will be reduced and, it might not take as long to converge to the optimal solution in the iterative stage. However, it will be also interesting to compare results with the approach discussed earlier of ignoring the joint constraints during initial seeding. The total number of seeds solved the overall process – initial seed generation and column generation iterations – using both approaches will give us some understanding about the best way to minimize solution times.
INITIAL RESTRICTED MASTER PROBLEM

As discussed before, the column generation technique starts with a small number of variables for the master problem and then adds more variables as the algorithm iterates. Because it does not contain all possible columns, the master problem is called the restricted master problem (RMP). The initial set of columns for each compartment is generated by varying the shadow prices and solving the formulation in A-2. Each subproblem solution is represented by a variable (column) in the IRLMP, with the unadjusted objective function value as the objective function coefficient for the corresponding variable. The initial restricted master problem is then solved, which provides new shadow prices for the joint constraints, which are then used to generate more columns for the next iteration. The initial restricted master problem formulation is as follows:

\[
\text{Max } Z_{\text{RLMP}} = \sum_{k=1}^{K} \sum_{i \in I_0} Z_{k,i} R_{k,i} \\
\text{subject to:} \\
\sum_{i \in I_0} R_{k,i} = 1 \quad \text{for } k = 1, 2, \ldots, K \\
\sum_{k=1}^{K} \sum_{i \in I_0} H_{k,i,t} R_{k,i} - H_t = 0 \quad \text{for } t = 1, 2, \ldots, T \\
b_{l,t} H_t - H_{t+1} \leq 0 \quad \text{for } t = 1, 2, \ldots, T - 1
\]
\[-b_{k,t}H_t + H_{t+1} \leq 0\] for \(t = 1, 2, \ldots, T - 1\) \hspace{1cm} (22)

\[0 \leq R_{k,i} \leq 1\] for \(\forall k, i\) \hspace{1cm} (23)

**Formulation A-3: Initial Restricted Master Linear Programming Problems**

Where,

\(R_{k,i} =\) solution \(i\) for compartment \(k\) generated by solving a subproblem, either in the initialization stage or during the iterative process of column generation. In the final master problem the value of this variable will be either 0 or 1, but in order to obtain shadow prices for the joint constraints, the problem is first solved by relaxing the integer constraints on these variables. When the shadow prices converge, in the final iteration, the problem is solved with integrality constraints,

\(I_0 =\) set of solutions for compartment \(k\) created in the initialization step, \(I_0 = 1, \ldots, I_0\),

\(K =\) set of compartments in the forest,

\(Z_{k,i} =\) the unadjusted discounted net revenue obtained by solving the Lagrangian dual problem for compartment \(k\), if option \(i\) is chosen,

\(H_{k,i,t} =\) timber harvest contribution in \(m^3\) of compartment \(k\) for initial seed \(i\) in period \(t\),

\(H_t =\) the amount of harvest produced by the all the compartments in period \(t\),

\(b_{l,t} =\) a lower bound on decreases in the harvest level between periods \(t\) and \(t+1\),

\(b_{h,t} =\) an upper bound on increases in the harvest level between periods \(t\) and \(t+1\).
Equation (18) specifies the objective function of the problem, which is to find the management option for each compartment that minimizes the net discounted revenue obtained from the forest. The coefficients $Z_{k,i}$ in the objective function represent the initial sets of columns obtained by solving the compartment-level problems.

Constraint set (19) are logical constraints that allow only one solution to be chosen for each compartment. Constraint set (20) are harvest accounting constraints that tally the total amount of wood harvested in each period given the solutions that are selected for each compartment. Constraint sets (21) and (22) are the volume flow constraints that limit the fluctuation of the volume harvested from the entire forest between two adjacent periods. These are the joint constraints.

Constraint set (23) are integer restrictions on the variables of the master problem. Ideally, the master problem must have integer restrictions on the variables in order to enforce the integer restrictions of the original IP. However, the purpose of an initial restricted problem is to obtain the shadow prices of the joint constraints to recalculate the penalties in the subproblem formulations. These iterative procedures eventually may result in the convergence of shadow prices, at which time, the integer restrictions are re-imposed on the master problem and it is solved one last time. Thus, the iterative master problems are relaxed in the sense that the variables are allowed to take fractional values.
ITERATIVE SUB PROBLEM FORMULATION

In the next step, the shadow prices obtained from the initial restricted master LP are used to generate a column for each compartment which are then added to the restricted master LP for the next iteration. The iterative subproblem formulation is very similar to the initial subproblem formulation. The only difference is that in the former the shadow prices were perturbed to create sets of columns while later the exact shadow prices from the previous iteration’s restricted master LP are used to penalize the joint constraint violations for each compartment. Hence, in the second iteration only \( k \) columns will be added to the master problem. While solving these iterative subproblems, the updated shadow prices from the previous iteration’s master problem will hopefully help us to obtain a better set of subproblem solutions, and hence better variables to add to the master problem in the next iteration. This iterative procedure continues until the stopping criterion is met. The entire structure of the iterative subproblem formulation is otherwise similar to the initial subproblem formulation. The iterative subproblem formulation is shown in Formulation A-4, on a next page:
Max $Z_{_{\text{adjusted}}}$ = \( \sum_{m=1}^{M_k} A_m c_{m,0} X_{m,0} \)

\[
+ \sum_{m=1}^{M_k} \left\{ A_m c_{m,1} + \left( -b_{m,t} \left( \lambda^d_{t,r} \right) + b_{m,t} \left( \lambda^u_{t,r} \right) \right) v_{m,1} \right\} X_{m,1} \\
+ \sum_{t=2}^{T-1} \sum_{m=1}^{M_k} \left\{ A_m c_{m,t} + \left( -b_{m,t} \left( \lambda^d_{t,r} \right) + b_{m,t} \left( \lambda^u_{t,r} \right) \right) v_{m,t} \right\} X_{m,t} \\
+ \sum_{m=1}^{M_k} \left\{ A_m c_{m,T} + \left( \lambda^d_{T-1,r} \right) \right\} X_{m,T}
\]

.............. (24)

Subject to:

\[
X_{m,0} + \sum_{t=h_n}^{T} X_{m,t} \leq 1 \\text{for } m \in M_k
\]

............. (25)

\[
\sum_{m \in P} X_{m,T} \leq n_p - 1 \\text{for } p \in P \text{ and } t = h_p, ..., T
\]

............. (26)

\[
\sum_{m=1}^{M_k} \sum_{t=0}^{T} A_m c_{m,t} X_{m,t} - Z_{k,r} = 0
\]

............. (27)

\[
\sum_{m \in M_{k,h_n}} v_{m,t} X_{m,t} - H_{k,r,t} = 0 \\text{for } t = 1, 2, ..., T
\]

............. (28)

\[
X_{m,t} \in \{0, 1\} \\text{for } m \in M_k \text{ and } t = h_m, ..., T
\]

............. (29)

*Formulation A-4: Iterative Lagrangian Subproblem*
Where,

\[ \lambda_{d,r} = \text{Shadow prices related to declining volume flow constraints (joint constraints) from the iteration } r \text{ master problem,} \]

\[ \lambda_{u,r} = \text{Shadow prices related to rising volume flow constraints (joint constraints) from the iteration } r \text{ master problem,} \]

\[ r = \text{iteration number for the master problem,} \]

\[ c_{m,t} = \text{the discounted net revenue coefficient for management unit } m \text{ if it is harvested in period } t \text{ (objective function coefficients from the original problem),} \]

\[ X_{m,t} = \text{treatment variables for compartment } k \text{ (from the original problem),} \]

\[ Z_{k,r} = \text{the unadjusted discounted net revenue obtained from the compartment } k \text{ for iteration } r, \]

\[ H_{k,r,t} = \text{timber harvest contributions in m}^3 \text{ of compartment } k \text{ for iteration } r \text{ in period } t. \]

The solutions generated by solving formulation A-4 are represented as variables in the iterative relaxed master LP. With these new variables, the objective function for the iterative master problem will become:

\[
\text{Max } Z^r_{RLMP} = \sum_{k=1}^{K} \sum_{i \in I_r} Z_{k,i} R_{k,i} \quad (30)
\]

where \( Z^r_{RLMP} \) represents the objective function of restricted linear master problem for iteration \( r \) and \( I_r \) is the set of solutions for compartments from initialization step \( (i \in I_0) \) and from iterations 1 through \( r \). This new master problem is now solved, and the
shadow prices from this new solution are again used to reformulate the subproblems in the next iteration. The new subproblems are solved and new variables are again added to the master problem for the next iteration. This procedure continues until the stopping criterion is met. Once the stopping criterion is met, the integrality constraints \( R_{ki} \in \{0,1\} \) are re-imposed to solve the final integer master problem.

**STOPPING RULE**

Column generation is an iterative algorithm that continues creating new master problems until a termination criterion for the algorithm is reached. Different stopping rules could be used to terminate the algorithm. In this case, the iterative procedure is continued until the shadow prices for the master problem do not change from one iteration to the next, within a given tolerance. It is possible for the D-W decomposition and column generation algorithm to quickly produce solutions that are close to optimal, but then make only infinitesimal improvements as it nears convergence in subsequent iterations. Hence, we impose a prespecified tolerance, \( \epsilon \), on the difference between the shadow prices of master problems in successive iterations. When the difference between the shadow prices from one iteration to the next are all less than this tolerance the algorithm is terminated. Once the shadow prices stabilize, the integer constraints on the variables in the final master LP are re-imposed and problem is solved one more time with the variables forced to take binary values. The restricted linear master problem (RLMP) thus becomes the restricted integer master problem (RIMP). The solution of RIMP is a feasible solution to the original problem since the convex constraints are imposed in each
compartment subproblem and the joint constraints are imposed in the master problem. The solution is not necessarily optimal to the original problem, but it should be near-optimal if the process has produced good values for the shadow prices.

**EXAMPLE FOREST CASE STUDY**

In order to carry out the research, a real-world Pennsylvania Department of Conservation and Natural Resources Bureau of Forestry (BOF) example was chosen. The forest shown in the Figure 3-3 is BOF District-8, also known as the Kittanning State Forest, and comprised of many small non-contiguous forest landscapes located in one region. The landscapes’ names are Phyllis Run, Leeper Run, English Run, Beartown Rocks, Stalhman Roundtop, Callen Run, Dutch Hill, Green Brair, Dice Run – West and East, and Work Run. The forest is located in the northern part of Jefferson County. As shown in the figure, the northern half of the forest falls in the High Plateau ecoregion and the southern half falls in the Pittsburgh lower plateau ecoregion. The forest has a total of 7,340 acres zoned as multiple resource and commercial (M&C) zones, where timber, water, recreation, fauna, flora and minerals are major important. There are seven forest types - Northern Hardwoods, Allegheny Hardwoods, Red Maple, Red Oaks, Other Oaks, Other Hardwoods and Conifers, and three site classes. Seventy-three percent of the forest is covered by oaks (49% - Red Oaks, 24% Other Oaks). Site class 1 is dominant, comprising 75% of the M&C land base.
Figure 3-3: DCNR Bureau of forestry District-8 (M&C-zoned stand boundaries are indicated with black lines).

Figure 3-4 show the initial age class distribution by forest type for the forest. Unorganized and widespread harvest at the turn of 20th century resulted in the unregulated age class distribution, with a majority of the acres distributed between ages 70 and 110. The small number of acres in younger stands shown in the figure reflects the limited harvests that have occurred since the area was largely cut over 100 year ago. This
mature age-class distribution tends to result in the harvest scheduling problems that are difficult to solve (McDill and Braze, 2000; 2001).

Figure 3-4: Initial age-class distribution for the M&C zone of District-8 forest specified by forest type
TREATMENT UNIT DELINEATIONS

The forest landscape shown in Figure 3-3 indicates the non-uniform nature of stands. The term “treatment unit” describes a discrete land unit represented by a 0-1 variable. In most harvest scheduling research, management units are created by subdividing stands (Barrett, 1997; Borges and Hoganson, 1999; 2000; Lu and Eriksson, 2000; Dozic, 2004). Recent work by Petroski (2006) showed that on Pennsylvania state forests, treatment units are not delineated according to stand boundaries but the boundaries for units are defined on the basis of roads, streams, trails, pipelines, fencing, topography and competing vegetation (Petroski, 2006).

A software tool called Treatment Unit Delineation Model (TUDM) developed by Petroski (2006) was used to delineate the stand layer of District-8 shown in Figure 3-2. TUDM uses a simulated annealing algorithm to aggregate ¼ - ½ acre triangles into management units based on criteria including hard boundaries, unit shape, unit size and diversity. Hard boundaries are lines that management unit boundaries cannot cross, for example, significant topological or vegetative cover changes, roads, existing deer exclusion fences, etc. Shape index, a patch shape measure (Baskent and Jordan, 1995), is used to create the roundest units possible. A diversity measure is used to aggregate similar triangles, i.e., one objective is to minimize diversity within management units. Stand boundaries are generally ignored unless they represent significant cover type changes. Petroski (2006) provides a complete explanation of TUDM, including an in-depth explanation of the criteria outlined above.
Stand data are necessary to determine yields of forest products and to write forest type-based constraints. As a result, forest cover attributes must be assigned to management units. Since units often cross stand boundaries, a single set of forest cover characteristics cannot be assigned. Instead, cutting components are created by overlaying management unit and stand spatial data; thus, a cutting component belongs to only one management unit and only one stand, but a management unit may contain multiple cutting components and a stand may contain multiple cutting components. An ESRI® ArcGIS™ script carries out this function, generating cutting component spatial data and assigning stand attributes to each.

The District-8 delineated treatment unit layer is shown in figure 3-5. The delineated layer has 368 management units and 1297 management unit components. On average, there are 3.52 components per unit, and the average management unit area is 19.67 acres.
Figure 3-5: District-8 delineated treatment unit layer
FOREST DATA

The base forest model was developed and a range of initial seed numbers (3, 5, 10, 25, 50) and random intervals between which the shadow prices of the original LP were perturbed ((0.75, 1.25), (0.5, 1.5), (0, 2), (-1, 3)) were tested. The objective function of the model was to maximize the discounted net revenue from the forest, including the discounted forest residual value based on the state of the management unit at the end of the planning horizon. The residual forest value assumes that after the end of the planning horizon each stand will be harvested at the financially optimal rotation. A 50-year planning horizon was used with five 10-year planning periods. The allowable harvest volume fluctuation between two consecutive periods was constrained to be within ±10%. The forest was decomposed into 9 compartments, and compartments had different numbers of management units, ranging from 19 to 55 (Table 3-1).

Table 3-1: District-8 compartment composition

<table>
<thead>
<tr>
<th>Compartment</th>
<th># of Units</th>
<th># of Components</th>
<th>M&amp;C Area (Acres)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>34</td>
<td>88</td>
<td>628.8</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>58</td>
<td>353.0</td>
</tr>
<tr>
<td>3</td>
<td>49</td>
<td>179</td>
<td>747.9</td>
</tr>
<tr>
<td>4</td>
<td>55</td>
<td>177</td>
<td>753.4</td>
</tr>
<tr>
<td>5</td>
<td>55</td>
<td>212</td>
<td>1193.4</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
<td>141</td>
<td>909.9</td>
</tr>
<tr>
<td>7</td>
<td>43</td>
<td>163</td>
<td>943.8</td>
</tr>
<tr>
<td>8</td>
<td>34</td>
<td>138</td>
<td>913.7</td>
</tr>
<tr>
<td>9</td>
<td>39</td>
<td>141</td>
<td>794.8</td>
</tr>
<tr>
<td>Total</td>
<td>368</td>
<td>1297</td>
<td>7238.7</td>
</tr>
</tbody>
</table>
The forest is composed of 7 forest types and 3 site classes. The total area of the forest shown in Table 3-1 is the commercially operable area (zoned multiple resource and commercial (M&C)). The remaining acres in the forest are referred to as ‘Other Zones’ which are either too steep to log, or are in streamside management zones, visual buffers along roads, or ecologically sensitive areas. The forest is divided into three site classes: Site Class 1, Site Class 2 and Site Class 3. Site Class 1 is a typical of moist, well-drained, and fairly deep soils that are typically found in bottomlands, along streams or on north slopes. Site Class 2 is characterized by soil intermediate in moisture, depth, drainage, and fertility and is usually found on slopes between ridge tops and bottomlands. Site Class 3 has shallow, rather dry and stony or compact soils which occur on ridges or broad flat plateaus.

In order to model the problem, economic data such as stumpage prices, regeneration costs and the real interest rate were used as input parameters. The real interest rate used for discounting was 4% and a regeneration cost of $200 was used. Table 3-2 shows the assumed stumpage prices for each product (species group). Price data were obtained from the Pennsylvania Woodlands’ Timber Market Report (http://www.sfr.cas.psu.edu/TMR/TMR.htm), which provides prices paid in the Commonwealth of Pennsylvania for standing timber (stumpage) and logs delivered to sawmills. Data used were for the fourth quarter of 2007.
Table 3-2: Stumpage prices for wood products

<table>
<thead>
<tr>
<th>Products</th>
<th>Stumpage Price ($/MBF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Oaks</td>
<td>360</td>
</tr>
<tr>
<td>White Oaks</td>
<td>295</td>
</tr>
<tr>
<td>Red Maple</td>
<td>228</td>
</tr>
<tr>
<td>Hard Maple</td>
<td>420</td>
</tr>
<tr>
<td>Black Cherry</td>
<td>1085</td>
</tr>
<tr>
<td>Other Hrdwds</td>
<td>118</td>
</tr>
</tbody>
</table>

COLUMNS GENERATION AND HARVEST SCHEDULING SOFTWARE SYSTEM

Real world forestry problems are very large, and creating the formulation is a large and daunting task. Formulating models is time consuming, even for small forests with as few as ten management units. Formulation by hand becomes impossible on problems with a realistic planning horizon and number of management units, where variables and constraints can number in tens of thousands. Analyzing solutions and displaying results also becomes difficult as model size increases. Computer programs are indispensable in such cases. The District-8 forest is a medium-sized forest in terms of number of units and variables involved, but the number of variables and constraints for a spatially-explicit HSM for the forest is very large. Hence, in order to deal with such large problems, a suite of Microsoft® Visual Basic™ computer programs has been developed to formulate and analyze mixed integer HSMs.

MakeMIP (Braze, 1999; McDill and Braze, 2000) formulates mixed integer mathematical programs, and MakeMIP ReportWriter analyzes model solutions and produces maps, tables and graphs that interpret the solutions. The formulation file created
by MakeMIP is input to a commercial solver such as CPLEX or LINDO where it is solved. The solution files are then input to the MakeMIP ReportWriter which analyzes solutions and displays the results in the form of maps, tables and graphs. Thus, the process is carried out in three independent steps: 1) creating formulations (MakeMIP); 2) solving formulations (CPLEX), and 3) analyzing solutions (MakeMIP ReportWriter). However, implementing the column generation algorithm is much more complex as it involves an iterative procedure that integrates MakeMIP, ReportWriter and CPLEX operations. In order to facilitate the column generation procedure, a program called Column Generation Software (CGS) was developed in .NET® that integrates all the necessary operations to implement the column generation algorithm.

Many of MakeMIP problem-formulation functions were mimicked in CGS. The MakeMIP program has been developed over several years through the work of Dr. Marc McDill and his research group at the Pennsylvania State University School of Forest Resources. The program provides users with an interface to input the model information to formulate HSMs. It also provides user with the ability to incorporate a wide variety of constraints in the formulation. Once formulated, the problem parameters are stored in a text file so that it can be read by the software in the future for reformulations, and formulations are written to the text files in .lp and .ltx formats so that they can be solved using solvers such as ILOG CPLEX© and LINDO©.

Since column generation is an iterative procedure that involves solving many of subproblems and a master problem, CGS uses CPLEX’s callable library functions to interact with CPLEX to solve problems. A user form was created to enable users to input
CPLEX parameters such as solution precision tolerances, stopping criteria, node file creation, work memory, parallel processing, etc. Hence during the column generation algorithm, CGS works as the interface for CPLEX so users do not have to manually go into CPLEX to input solver parameters. In each iteration, CGS reads the restricted master problem solution, obtains the dual variables and then uses them in formulating subproblems in the next iteration.

**FACTORS INFLUENCING THE COLUMN GENERATION ALGORITHM**

One of the objectives of this research is to investigate the parameters that influence the quality of the final solution generated by the column generation algorithm. Quality refers to the deviation of the RIMP solution from the optimal integer solution obtained by solving the original problem. The solution time taken by the algorithm is also important, as the main purpose of applying column generation on large problems is to make them easier to solve in terms of computational time. The District-8 forest used in this study is a mid-size forest managed by the Bureau. In the case of larger forests, it would be nearly impossible to solve the original problems in commercial solvers that use the branch and bound technique. Many of the typical district forests managed by the Bureau might have as many as 3,000 management units. Modeling these forests would involve a very large number of variables and constraints, which in turn, would make the original problem impossible to solve. In order to analyze the results of the column generation algorithm, comparisons are done with the relatively small District-8 forest so that an exact solution (or at least one with very tight bounds) to the original problem is available.
The column generation algorithm involves several important procedures, from creating the subproblems to getting the final RIMP solution. D-W decomposition, Lagrangian relaxation, initialization of RLMP and creating iterative subproblem seeds are a few of the procedures whose importance cannot be neglected. All involve a lot of critical parameters that might influence the quality and solution time of the algorithm. A few of the potentially important parameters are:

- The number of compartments in the forest,
- The number of management units in the compartment,
- The number of seeds created for the initial RLMP, and
- The variation in the parameter $\eta$, used to perturb the shadow prices of the original linear problem.

The next sections discuss these parameters and how they might affect the quality of the final integer master problem.

**Number of compartments in the forest**

As discussed earlier, D-W decomposition is used to decompose the original problem into a number of smaller subproblems in such a way that each subproblem has a relatively independent set of constraints. In order to apply this technique to forestry problems with adjacency constraints, one has to make sure that the forest can be divided into spatially independent compartments. In real world forests, it is often easy to find spatially independent compartments because of the streams, roads, and trails running through them.
The number of subproblems to be solved at the initialization step is the product of the number of compartments and the number of initial seed variables desired per compartment for the master problem. After the initialization step, one subproblem per compartment is solved at each iteration to produce a new variable for the master problem. Hence, increasing the number of compartments will increase the number of subproblems to be solved and might influence the final solution time. However, dividing the forests into more compartments also means that subproblems will have fewer variables and constraints, making them easier to solve.

The time taken to solve the subproblems is an important issue. The majority of the solution time taken by the column generation algorithm is for solving the subproblems and the final RIMP because of the binary constraints in their problem structures. The number of compartments has an influence on both the subproblems and RIMP. The problem with fewer compartments will have fewer, relatively large subproblems. On the other hand, it will have fewer variables in the final RIMP and hence the time taken to solve RIMP will likely be less. A problem with more compartments will have more, smaller subproblems but a bigger RIMP to solve. Hence, the number of compartments in the forest is a very important parameter that can affect the efficiency of the column generation algorithm.
**Number of management units in the compartment**

While decomposing the forests into spatially independent compartments, it is advisable to have more compartments whose sizes are proportional to the number of management units in them. However, in reality, this is not always possible, as forests have to be decomposed on the basis of natural boundaries and buffer zones. As discussed in the previous section, the number of compartments in the forest directly affects the number of management units in each compartment. The number of adjacency constraints in the subproblems increases drastically with an increase in number of management units in the corresponding compartment. That is, as the number of management units in the compartment increases, the number of adjacency relationships increases at a drastic rate. Hence, the number of management units in the compartment plays an important role in the solution time taken by column generation.

**Number of seeds created for the initial RLMP**

Rajasekaran (2005) found that quality of the RIMP solutions improve with more variables in the master problem, as the chances of finding a solution close to optimality for the original problem increases with a more diverse set of “seeds”. However, it also depends on the way you create the initial set of seeds. Rajasekaran (2005) solved a cost minimization problem and randomly perturbed the targets in a harvest target constraints to get the initial set of seeds. The algorithm randomly created the initial seeds and expected to select the best seed while solving the iterative master problems.
It is very difficult to arrive at a solid conclusion unless one finds a good method to perturb the subproblem seeds so as to get the seeds that are close to the optimal seeds in the RIMP. In this research, a different approach was used to perturb the initial subproblem seeds. The shadow prices obtained by solving linear version of the original problem were used as Lagrangian multipliers to relax joint constraints in subproblems. This approach allows us to get seeds closer to an optimal seed and hence might reduce the necessity of generating a large number of seeds in the first step as proposed by Rajasekaran (2005). Creating more variables at the initialization step means spending more computer time on this step, as each variable requires solving a subproblem. It would be useful to get an idea of how the number of initial variables in the master problem influences the solution time and the quality of the final solution in the column generation process. To address this question, the case study problem was solved with 3, 5, 10, 25 and 50 initial seeds per compartment.

**Variation $\eta_t$ in perturbing the shadow prices of original linear problem**

In the algorithm used in this research, the values of $\eta_t$ (see equation 11) are generated randomly for each period from a uniform distribution within a certain interval. These random values are used to perturb the shadow prices obtained by solving the original LP. The extent to which $\eta_t$ is allowed to vary is an important factor. If $\eta_t$ is allowed to vary in a narrower interval, like (0.95, 1.05), then the variation in the solutions generated for the compartments at the initialization step will be minimal. In fact, there is a good chance that with a (0.95, 1.05) interval, we might end up generating identical seeds and wasting time. This might not pose a problem in cases where a small number of
initial seeds is used, but can be a huge drawback in cases where a large number of initial seeds is desired as it might prove to be very difficult to generate a large number of diverse seeds using a narrow interval. A host of scenarios like this can be imagined. Allowing \( \eta \) to vary to a larger extent will produce a greater variety of solutions for compartments at the initialization step, however, they may be farther from the optimum. For this research, four intervals were tested: (0.75, 1.75), (0.5, 1.5), (0, 2) and (-1, 3).
Chapter 4

COMPUTATIONAL TESTING AND RESULTS

Several problems for the DCNR Bureau of Forestry District-8 forest structure were modeled and solved with the column generation algorithm. As discussed in Chapter 3, there are several different input parameters which may play a vital role in the performance of column generation. This research focused on the number of initial seeds generated and the way they were generated. The number of compartments and management units could also be analyzed, but this research was restricted to the influence of initial seeds in the column generation algorithm.

A computer with a 3.73 GHz processor and 3 GB RAM was used for all the computations. The original problem for the forest was modeled and solved using CPLEX to a 0.1% optimality gap. The 99.9% optimal solution for the original problem obtained from CPLEX was used for assessing the quality of solutions generated by column generation. Twenty different models were created for different combinations of number of initial seeds and amount of perturbation. Since the seeds for the master problem were randomly generated, 10 replications for each problem were created to account for this source of variation. Hence a total of 200 problems were generated and solved using column generation.
The chapter is organized into six sections. The influence of the two input parameters is tested by analyzing six different output results: solution quality, solution time, number of iterations, number of subproblems in RIMP, number of identical subproblems generated by the algorithm, and solution time taken by the RIMP. In order to compare the different output parameters, the average of all the replications for a particular combination of the parameters was used.

**RIMP OBJECTIVE FUNCTION VALUE**

The optimal objective function value found by the price-directed decomposition and column generation algorithm may be quite different from that of the original IP solution. The optimal value of the final iteration linear master problem is an upper bound to the original IP (Martin, 1999; Pittman, 2007). However, constraining the problem with binary restrictions will yield a solution that is not necessarily the same as the original IP solution. A measure of the algorithm’s performance relative to the desired solution of the original IP can be obtained by comparing its RIMP solution and the optimal value of the original IP.
Table 4-1: Comparison of column generation objective function values with the original IP solution objective function value

<table>
<thead>
<tr>
<th>Initial Seed</th>
<th>Interval 1 (0.75,1.25)</th>
<th>Interval 2 (0.5,1.5)</th>
<th>Interval 3 (0,2)</th>
<th>Interval 4 (-1,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CG RIMP Soln ($)</td>
<td>% Deviation From Org IP</td>
<td>CG RIMP Soln ($)</td>
<td>% Deviation From Org IP</td>
</tr>
<tr>
<td>3</td>
<td>15,992,313</td>
<td>0.0692</td>
<td>15,997,732</td>
<td>0.0728</td>
</tr>
<tr>
<td>5</td>
<td>15,992,733</td>
<td>0.0666</td>
<td>15,992,524</td>
<td>0.0679</td>
</tr>
<tr>
<td>10</td>
<td>15,993,850</td>
<td>0.0596</td>
<td>15,991,399</td>
<td>0.0749</td>
</tr>
<tr>
<td>25</td>
<td>15,993,430</td>
<td>0.0622</td>
<td>15,993,595</td>
<td>0.0612</td>
</tr>
<tr>
<td>50</td>
<td>15,996,724*</td>
<td>0.0417</td>
<td>15,993,258</td>
<td>0.0633</td>
</tr>
</tbody>
</table>

Table 4-1 shows the average objective function value of 10 replications of the column generation algorithm for the different combinations of shadow price perturbation intervals and numbers of initial seeds. The table also shows, in percentage deviation terms, how close the average objective function values were to the optimal objective function value obtained by CPLEX. It is difficult to find a trend in the data shown in Table 4-1 in terms of influence of perturbation interval and initial seeding on solution quality. These results illustrate the randomness involved in the initial seed generation step of column generation. The best average solution for the model was obtained for perturbation interval 1 and 50 initial seeds. The objective function deviation should not necessarily be the only parameter to estimate performance of column generation. The solution time is also an important factor, and 50 seeds with a narrower interval will typically require a longer solution time. Figure 4-1 plots the average objective function value obtained from each parameter combination and suggests that the objective function values of all 200 problems are more-or-less randomly distributed and there are no trends.
Figure 4-1: Average objective function values plotted against number of initial seeds for each shadow price perturbation interval.

However, it is important to also consider the overall fluctuation of the objective function value. Figure 4-2 plots the objective function values of all 200 replications against the number of seeds. The figure shows that the variation of the objective function values of the final RIMP is fairly constant with at least 10 seeds per compartment. In order to find the extent of fluctuation, a confidence interval for the net discounted revenue of the sample population of 200 was estimated.
Figure 4-2: Graph of objective function values for all 200 replications against number of initial seeds.

Table 4-2: Confidence interval estimation for a population mean of 200 replications.

<table>
<thead>
<tr>
<th>Confidence Interval</th>
<th>MoE ($),</th>
<th>Upper Bound ($),</th>
<th>Lower Bound ($),</th>
<th>Sample Mean ($),</th>
<th>Std Dev. ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>99%</td>
<td>677.00</td>
<td>15,993,372</td>
<td>15,992,018</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95%</td>
<td>515.13</td>
<td>15,993,210</td>
<td>15,992,180</td>
<td>15,992,695.37</td>
<td>3,716.96</td>
</tr>
<tr>
<td>90%</td>
<td>432.31</td>
<td>15,993,127</td>
<td>15,992,263</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4-2 shows the confidence interval estimates for a population mean of all 200 replications. The margin of error for the 99% confidence interval is as low as $677 which is a very small fraction of the original IP solution. As the column generation procedure involves random generation of initial seeds, there is a certain level of unpredictability associated with the final output as it eventually boils down to the quality of seeds generated in the initial step. Table 4-2 shows that irrespective of the number and
quality of initial seeds generated, we can be 99% confident that the true mean of the population of objective functions that we might get from the column generation algorithm will be between $15,993,372.37 and $ 15,992,018.37. The 99% confidence interval for the average deviation of the RIMP solution from the solution obtained for the original IP is (0.063%, 0.071%). The percentage deviation itself is very small and can be neglected.

In general, it can be said that the column generation procedure implemented in this research consistently generated high-quality near-optimal integer solution. It can thus be concluded that the price-directed decomposition and column generation algorithm implemented with randomly generated initial set of seeds results in solutions that are very close to the original IP solution. There are no observed trends associated with the number of initial seeds or their different random generation intervals in terms of the RIMP solution.

**TOTAL SOLUTION TIME TAKEN BY COLUMN GENERATION**

Solution time is probably the second most important criterion after solution quality in the field of mixed integer programming. Mixed integer programs are for modeling a variety of decision making problems, but the binary restrictions on the variables make them hard to solve, often requiring considerable amounts of computational time. Also, increasing the problem size generally increases the number of branches in the branch-and-bound tree, which makes problems increasingly difficult to solve within a realistic time frame. This is why methods like column generation and other heuristics have gained significant attention lately.
Table 4-3: Comparison of average solution times taken by column generation and the branch-and-bound algorithm

<table>
<thead>
<tr>
<th>Initial Seed</th>
<th>Solution Time (Sec)</th>
<th></th>
<th></th>
<th>Org IP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Interval 1</td>
<td>Interval 2</td>
<td>Interval 3</td>
<td>(-1,3)</td>
</tr>
<tr>
<td></td>
<td>(0.75,1.25)</td>
<td>(0.5,1.5)</td>
<td>(0,2)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>54.5</td>
<td>55.4</td>
<td>60.9</td>
<td>61.7</td>
</tr>
<tr>
<td>5</td>
<td>64.9</td>
<td>64.8</td>
<td>65.5</td>
<td>95.1</td>
</tr>
<tr>
<td>10</td>
<td>166.8</td>
<td>172.5</td>
<td>99.3</td>
<td>106.3</td>
</tr>
<tr>
<td>25</td>
<td>483.6</td>
<td>256.6</td>
<td>237.5</td>
<td>210.9</td>
</tr>
<tr>
<td>50</td>
<td>1650.0</td>
<td>1531.</td>
<td>733.71</td>
<td>714.0</td>
</tr>
</tbody>
</table>

The original IP formulation was solved by CPLEX to a 0.01% optimality gap in 246.9 sec. Figure 4-5 shows the average solution time taken by the column generation algorithm for each combination of number of initial seeds and perturbation interval. Since column generation is an iterative procedure, the column generation software does take extra time to generate subproblem formulation text files. There are many different ways to write the code efficiently to save time in generating these files. In this study, only the pure solution time taken to solve the problems is used to compare the results. One clear trend that can be noticed from the table is that increasing the number of initial seeds increases the solution time taken by algorithm. This is to be expected, as more initial seeds require the algorithm to spend more time in solving subproblems. Rajasekaran (2005) showed that increasing the number of initial seed increases the solution time for different target levels. However, in that study the solution quality improved as the number of initial seeds was increased. He suggested that increasing the number of variables in the master problem will provide a better opportunity for the variables to mix and hence improve the quality of solution. However, as discussed in earlier chapters, the
quality of the initial seeds generated plays a very important role in determining the optimal number of initial seeds. If the algorithm is able to generate seeds that are close to optimal then unnecessarily generating additional seeds is not required. Figure 4-3 below shows average solution times taken by the column generation algorithm for combinations of numbers of initial seeds and shadow price perturbation intervals. In this research, because of the price-directed approach, the algorithm was able to get a very good mix of initial subproblem seeds with a small number of seeds. Hence, it does not seem to be necessary to generate a large pool of initial columns to get a better mix and hence we were able to get a good set of solutions in less time.

Figure 4-3: Average solution time plotted against the number of initial seeds for different shadow price perturbation intervals.
The results in Table 4-2 and Figure 4-3 show that the algorithm took the smallest average solution times for sets of three initial seeds and that average solution quality was not significantly improved with additional seeds. This shows that the algorithm created near optimal subproblem seeds in the initial step and hence the first iteration RLMP solutions were close to the final RIMP solution. This suggests that the majority of the time spent by the problems with more initial seeds is due to the larger number of subproblems. More seeds also mean more variables in the final RIMP formulation. The time required to solve the RIMP may play an important role in the solution time required by the column generation algorithm. Comparing the subproblem solution time with the RIMP solution time might be useful.

Figure 4-3 also shows that for smaller numbers of initial seeds, the perturbation interval does not affect the solution time. However, with 25 and 50 initial seeds, the solution time decreases with broader perturbation intervals. For instance, perturbation interval 1 is smaller than perturbation interval 4; that is, the shadow prices of the joint constraints are perturbed within a smaller interval in the case of perturbation interval 1. Hence, there is a higher probability of creating identical initial seeds with perturbation interval 1. If the same seed is created then the algorithm rejects that seed and goes on to generate another, hopefully different seed. This results in unused identical initial seeds which are solved but not used in the final RIMP formulation.
The time taken by the seeds that go unused has to be accounted for in the final solution time taken by column generation. For smaller numbers of initial seeds the perturbation intervals don’t have a significant influence as there is less possibility of generating similar seeds in a pool of 3, 5 or 10 than with 25 or 50. Hence for larger numbers of initial seeds, it is advisable to use broader perturbation intervals. On the other hand, perturbation intervals do not have substantial impact on the solution time for a smaller number of seeds. Figure 4-4 graphs solution times against the number of initial seeds for all 200 replications.

Figure 4-4: Graph of solution time against number of initial seeds for all 200 replications.
The above discussion showed that column generation took very short times to achieve solutions for which we can be 99% confident that the true mean of the population of objective functions will be within 0.071% of the original IP solution. This analysis compares the time taken to obtain an inferior solution with the time taken to obtain a superior solution. Hence, in order to know whether column generation actually outperformed CPLEX, it is necessary to find the time taken by CPLEX to create the solution just better than that achieved by column generation.

Figure 4-5: Hypothetical graph of how the optimality gap determined by CPLEX evolves over time as the algorithm solves a problem.

In its search for the optimal solution, CPLEX uses a branch-and-bound approach that gradually tightens upper and lower bounds on the optimal objective function value.
The range between these bounds is called the optimality gap, expressed as a percentage of the best current known objective function value. As the branch-and-bound algorithm progresses it typically follows a trend similar to that shown in Figure 4-5. The figure shows how CPLEX can quickly find a solution very close to optimal but then takes a long time to reduce the optimality gap to less than the target: 0.1% in the figure. Hence it is possible that CPLEX could have found a solution that was as good as that found by column generation in less time than required by column generation. So it is important to see whether column generation really beat the time it took CPLEX to find an equivalent solution. Table 4-4 shows the time taken by CPLEX to find a solution that is just better than the solution found by column generation.

Table 4-4: Time taken by CPLEX to achieve just better solution compared to column generation

<table>
<thead>
<tr>
<th>Initial Seed</th>
<th>Interval 1 (0.75,1.25)</th>
<th>Interval 2 (0.5,1.5)</th>
<th>Interval 3 (0,2)</th>
<th>Interval 4 (-1,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time taken by CG to reach optimal IP solution (Sec)</td>
<td>Time taken by CPLEX to reach a CG Soln (Sec)</td>
<td>Time taken by CG to reach optimal IP solution (Sec)</td>
<td>Time taken by CPLEX to reach a CG Soln (Sec)</td>
</tr>
<tr>
<td>3</td>
<td>54.5</td>
<td>83.0</td>
<td>55.4</td>
<td>74.9</td>
</tr>
<tr>
<td>5</td>
<td>64.9</td>
<td>83.0</td>
<td>64.8</td>
<td>83.0</td>
</tr>
<tr>
<td>10</td>
<td>166.8</td>
<td>83.0</td>
<td>172.5</td>
<td>74.9</td>
</tr>
<tr>
<td>25</td>
<td>483.6</td>
<td>83.0</td>
<td>256.6</td>
<td>83.0</td>
</tr>
<tr>
<td>50</td>
<td>1650.0</td>
<td>119.5</td>
<td>1531.0</td>
<td>83.0</td>
</tr>
</tbody>
</table>

Table 4-4 shows that the time taken by CPLEX to reach a solution that is equivalent to the column generation solution is in all cases less than the total time taken by CPLEX to reach the target optimality gap of 0.1%. It can be seen that column
generation beats CPLEX in most cases with five or fewer seeds. In the case of three initial seeds column generation outperformed CPLEX for all perturbation intervals. Recall that these are average solution times of 10 replications. The minimum solution time taken by column generation for any replication was as low as 22 seconds, which was still better than 74.95 seconds, the lowest CPLEX solution time shown in Table 4-4. However, problem size is the crucial factor in this case. CPLEX will start having difficulties with large problems, so the difference in the solution time will only get larger with an increase in problem sizes.

**NUMBER OF ITERATIONS TAKEN BY MASTER PROBLEM TO CONVERGE**

Table 4-5 and Figure 4-6 below show the relationship between the average number of iterations required by the master problem and the number of initial seeds in the master problem for different shadow price perturbation intervals. The number of iterations taken by column generation to converge to an optimal IP solution is important as it is closely related to the solution time taken by the algorithm. If the number of steps taken by an algorithm to converge is less, then the number of total subproblems solved and the number of variables in RIMP are less. This will further help in improving the total solution time taken by algorithm as the time taken to solve subproblems is less and the smaller size of the RIMP will tend to reduce the time taken to solve the RIMP. Hence, it is generally desirable for an algorithm to converge in the least number of iterations possible.
Table 4-5: Comparison of average number of iterations taken by the master problem to converge for different combinations of initial seeds and shadow price perturbations.

<table>
<thead>
<tr>
<th>Initial Seed</th>
<th>Number of Steps to Converge</th>
<th>Interval 1 (0.75,1.25)</th>
<th>Interval 2 (0.5,1.5)</th>
<th>Interval 3 (0.2)</th>
<th>Interval 4 (-1,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>15.9</td>
<td>16.5</td>
<td>17.8</td>
<td>18.5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>14.1</td>
<td>15.8</td>
<td>16.9</td>
<td>17.4</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>12.6</td>
<td>15.0</td>
<td>15.8</td>
<td>16.6</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>12.2</td>
<td>13.3</td>
<td>15.0</td>
<td>16.4</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>12.2</td>
<td>13.8</td>
<td>14.1</td>
<td>14.5</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4-6: Average number of iterations plotted against the number of initial seeds for different shadow price perturbations.
Figure 4-6 confirms that the number of iterations required by the algorithm to converge decreases with an increasing number of initial seeds. This is because with a smaller pool of initial columns, the algorithm generates more seeds during the iterative procedure in order to get the best set of seeds for the final iteration integer master problem. A larger set of initial seeds improves the likelihood of obtaining seeds closer to an optimal seed in the first step itself. And as a result, the number of iterations required to converge after the initial step will tend to be lower.

Another interesting trend observed in Table 4-5 and Figure 4-6 is that for a particular number of initial seeds, the number of iterations increased with the size of the perturbation intervals. The narrower perturbation intervals ensure a more compact set of seeds whereas broader perturbation intervals create seeds that are too far from the optimal seeds. The table shows that the best combination of initial seed and perturbation interval in terms of number of iterations is 50 seeds and perturbation interval 1. But also recall that the same combination draws the poorest computational time and hence it can be concluded that number of iterations required by column generation to converge is really insignificant.

**NUMBER OF SUBPROBLEMS IN RIMP**

Table 4-5 below shows the influence of the number of initial seeds and the perturbation interval of the shadow price perturbations on the number of subproblems used in the final iteration RIMP. The number of subproblems generated by column
generation is a very important factor and is closely related to the total solution time taken by the algorithm.

Table 4-6: Comparison of average number of subproblem seeds used in RIMP for different combinations of initial seeds and shadow price perturbations.

<table>
<thead>
<tr>
<th>Initial Seed</th>
<th>Interval 1 (0.75,1.25)</th>
<th>Interval 2 (0.5,1.5)</th>
<th>Interval 3 (0,2)</th>
<th>Interval 4 (-1,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of SP in RIMP</td>
<td>Total SP Solution Time (Sec)</td>
<td>Proportion of Total Solution time</td>
<td># of SP in RIMP</td>
</tr>
<tr>
<td>3</td>
<td>170.1</td>
<td>19.85</td>
<td>36.45%</td>
<td>175.5</td>
</tr>
<tr>
<td>5</td>
<td>171.9</td>
<td>17.47</td>
<td>26.91%</td>
<td>187.2</td>
</tr>
<tr>
<td>10</td>
<td>203.4</td>
<td>61.70</td>
<td>36.99%</td>
<td>225</td>
</tr>
<tr>
<td>25</td>
<td>334.8</td>
<td>91.33</td>
<td>18.89%</td>
<td>344.7</td>
</tr>
<tr>
<td>50</td>
<td>559.8</td>
<td>204.26</td>
<td>12.38%</td>
<td>574.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interval 3 (0,2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td># of SP in RIMP</td>
<td>Total SP Solution Time (Sec)</td>
<td>Proportion of Total Solution time</td>
<td># of SP in RIMP</td>
</tr>
<tr>
<td>3</td>
<td>187</td>
<td>15.33</td>
<td>25.19%</td>
<td>193.5</td>
</tr>
<tr>
<td>5</td>
<td>197.1</td>
<td>26.91</td>
<td>41.06%</td>
<td>201.6</td>
</tr>
<tr>
<td>10</td>
<td>232.2</td>
<td>51.93</td>
<td>52.28%</td>
<td>239.4</td>
</tr>
<tr>
<td>25</td>
<td>360</td>
<td>128.15</td>
<td>53.95%</td>
<td>372.6</td>
</tr>
<tr>
<td>50</td>
<td>576.9</td>
<td>259.68</td>
<td>35.39%</td>
<td>580.5</td>
</tr>
</tbody>
</table>

Table 4-5 shows the number of subproblems that were used in the RIMP, the total solution time taken for solving subproblems and the proportion of the total solution time taken by entire the column generation procedure. In most cases, the time taken to solve the subproblems is a relatively small proportion of the total solution time as it should be kept in mind that the remaining time taken by the algorithm is just to solve one RIMP. For instance, in the case of three initial seeds and perturbation interval 1, the
average solution time taken to solve an average of 170 subproblems was 19.85 seconds, which accounts for 36.45% of the total solution time. This means that the remaining 63.55% of the total solution time taken by column generation algorithm was used to solve the RIMP.

Hence, it seems that the number of subproblems generated by the algorithm has a smaller impact on total solution time compared to the time taken to solve the RIMP. However, an interesting thing to notice is that the number of subproblems generated by the algorithm is also the number of variables in the RIMP. Thus, while the time taken to solve subproblems might be a concern, the number of subproblems generated by the algorithm definitely has an impact on the time taken to solve the RIMP and in turn on the total solution time taken by column generation process.

The size of subproblems also influences the total solution time. In this research the forest was decomposed into 9 compartments. If the same forest was decomposed into fewer compartments then that would have increased the sizes of each compartment and eventually the size of the subproblems. There will also be fewer variables in the RIMP in this case. The algorithm will then take more time to solve the subproblems and hence the proportions of time used to solve subproblems versus solving the RIMP will change. Conversely, increasing the number of compartments will make the subproblems smaller but also increase the number of variables in the RIMP. It would be a good exercise to estimate the influence of the number of compartments on the time taken to solve the subproblems versus the RIMP. However, this was beyond the scope of this research.
Figure 4-7 shows the relationship between the number of subproblems in the RIMP and both the number of initial seeds and perturbation intervals. The increase in subproblems with number of initial seeds is mainly due to the increase in the number of initial seeds used in the first step. For instance, in the case of 3 initial seeds, the algorithm generates only 27 (3 seeds per compartment) initial columns whereas in the case of 50 seeds, the algorithm generates 450 initial columns. The increase in the number of subproblems with the increase in interval size of shadow price perturbations is mainly because of the number of steps taken by the algorithm to converge. Table 4-4 shows that increasing the interval size increases the number of iterations. The addition of a single iteration will generate subproblems equal to the number of compartments in the forest. Hence, increasing the number of subproblems has less impact with a wider perturbation interval size than increasing the number of initial seeds.
Figure 4-7: Average number of subproblems used in RIMP plotted against the number of initial seeds for each shadow price perturbation interval.

**NUMBER OF IDENTICAL SEEDS DURING INITIALIZATION**

The initial subproblem seeds are generated by perturbing the shadow prices from the solution of the original LP problem and penalizing joint constraint violations through the objective functions of the subproblems. However, the corresponding objective function coefficients in the master problem are not the Lagrangian dual solutions but the unadjusted objective function values obtained by substituting dual optimal variable values in the original, unadjusted subproblem objective function. Hence, there is a very good chance that the algorithm will generate identical seeds during the initialization step despite shadow price perturbations. In order to avoid this, during the initialization step, if the algorithm encounters a subproblem seed that is identical to a previously generated seed, it will reject that seed and generate another seed until a unique seed is generated.
While these identical subproblems seeds are not used in the RIMP, they do take up solution time and hence play an important role in the total solution time taken by the algorithm. The aim should therefore be to avoid unnecessary generation of identical seeds during the initialization step. Table 4-7 and Figure 4-8 below compare the average number of identical subproblem seeds generated by the column generation algorithm in the initialization step for each combination of number of initial seeds and shadow price perturbation interval.

Table 4-7: Comparison of average number of identical subproblem seeds in the initialization step for different combinations of initial seeds and shadow price perturbations.

<table>
<thead>
<tr>
<th>No. of Initial Seeds</th>
<th>Number of Identical Initial Subproblem Seeds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Interval 1 (0.75,1.25)</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1.3</td>
</tr>
<tr>
<td>10</td>
<td>4.2</td>
</tr>
<tr>
<td>25</td>
<td>37.7</td>
</tr>
<tr>
<td>50</td>
<td>222.8</td>
</tr>
</tbody>
</table>
Figure 4-8: Average number of identical initial subproblem seeds plotted against the target number of initial seeds for each shadow price perturbation

Figure 4-9: Graph of number of identical subproblem seeds in the initialization step versus the target number of initial seeds for all 200 replications
Table 4-7 and Figure 4-8 show that the number of identical seeds generated by the algorithm increases with the number of initial seeds and decreases with an increase in the interval size of the shadow price perturbations. The change in both directions is drastic. For a particular perturbation interval, increasing the number of initial seeds means that the algorithm will have to generate more diverse seeds. For instance, for perturbation interval 1, it is easiest to generate three unique seeds so the number of identical seeds generated in that case is 0. But when 50 unique initial seeds are required per compartment, the algorithm has to generate 222 extra seeds to find 450 (50 initial seeds $\times$ 9 Compartments) unique initial seeds. Solving these 222 subproblems takes time, even though they are not used in the final RIMP, and hence contribute to the total solution time.

Similarly, for a particular initial seeding, the number of identical subproblem seeds decreases with an increase in the size of the shadow price perturbation interval. This is because increasing the interval size allows the algorithm to generate more diverse and independent seeds and hence there is less chance of generating identical subproblem seeds. The increase in identical subproblem seeds is very drastic for a combination of narrower perturbation interval and larger number of initial seeds. It is possible that the algorithm will reach a saturation point beyond which it will never find enough unique initial seeds. From Table 4-6 it can be concluded that 3 is the optimal level of seeding for any perturbation interval and 50 initial seeds with perturbation interval 1 proved to be the worst case scenario for the number of identical subproblem seeds.
RIMP SOLUTION TIME

Once shadow prices in the restricted linear master problems (RILMPs) converge, binary restrictions are introduced and an integer version of the final iteration’s master problem is solved. The time taken by the RIMP (final iteration restricted integer master problem) constitutes a majority of the time taken by the entire column generation process in most cases. Hence the size of the RIMP is a key factor in column generation. Table 4-8 compares average solution times taken by the RIMP for different combinations of seeding levels and perturbation intervals. Figure 4-10 shows the average RIMP solution time plotted against the seeding levels for each perturbation interval.

Table 4-8: Comparison of average solution times taken by the RIMP for different combinations of seeding levels and perturbation intervals.

<table>
<thead>
<tr>
<th>Initial Seed</th>
<th>RIMP Solution Time (Sec)</th>
<th>Interval 1 (0.75,1.25)</th>
<th>Interval 2 (0.5,1.5)</th>
<th>Interval 3 (0,2)</th>
<th>Interval 4 (-1,3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (Sec)</td>
<td>Proportion of total CG Soln Time (%)</td>
<td>Time (Sec)</td>
<td>Proportion of total CG Soln Time (%)</td>
<td>Time (Sec)</td>
</tr>
<tr>
<td>3</td>
<td>34.61</td>
<td>63.55</td>
<td>42.65</td>
<td>77.05</td>
<td>45.53</td>
</tr>
<tr>
<td>5</td>
<td>47.43</td>
<td>73.09</td>
<td>44.28</td>
<td>68.31</td>
<td>38.63</td>
</tr>
<tr>
<td>10</td>
<td>104.72</td>
<td>63.01</td>
<td>101.91</td>
<td>59.92</td>
<td>47.37</td>
</tr>
<tr>
<td>25</td>
<td>388.75</td>
<td>81.11</td>
<td>145.28</td>
<td>57.39</td>
<td>106.95</td>
</tr>
<tr>
<td>50</td>
<td>1433.26</td>
<td>87.62</td>
<td>1165.15</td>
<td>77.27</td>
<td>463.84</td>
</tr>
</tbody>
</table>
Figure 4-10: Average RIMP solution time plotted against the number of initial seeds for each shadow price perturbation interval.

Table 4-8 and Figure 4-10 show that average RIMP solution time generally increases with the number of initial seeds; deviations from this trend for small numbers of initial seeds likely reflect the randomness in the algorithm. It is also interesting to see that the graphs in Figures 4-3 and 4-10 are very similar, i.e. the total solution time taken by column generation algorithm tends to be proportional to the time taken to solve the RIMP. Also for higher seeding levels like 10, 25 and 50, the RIMP solution time decreases with increases in the perturbation interval size. This is less clear at lower seeding levels like 3 and 5 due to the randomness in the algorithm.
Applications of integer programming to solve forest management problems are largely still academic. Because of the binary variables in their formulation, integer programs provide more flexibility in decision making, allowing us to model aspects of problems that would be difficult to handle without them. However, the binary variables also make the formulations difficult to solve. Real-world forest management problems have only rarely been solved with exact integer programming techniques, although heuristic solution algorithms are sometimes used to solve real-world forest management problems formulated as integer programming problems. This is mainly due to the size of the problems. Column generation is clearly a success story in large-scale integer programming. Today generic integer programming column generation codes solve many large-scale problems of “industrial difficulty,” that no standard commercial MIP solver could cope with (Lubbecke et al, 2005).

Beyond a certain size, integer problems become increasingly difficult to solve within a realistic time frame. In these cases, a decomposition technique that splits the problem of interest into a number of smaller, more easily solved problems in order to solve the larger problem seems promising. This research formulated and solved a realistic forest management planning problem using a price-directed approach derived from
Lagrangian relaxation, referred to as price-directed decomposition. This allowed the problem to be solved using an iterative column generation algorithm.

The algorithm begins by solving the LP relaxation of the original problem. The shadow prices of the joint constraints are then randomly perturbed and used to penalize joint constraint violations through the objective functions of the sub-problems resulting from a Durban-Watson decomposition of the original problem to get initial sub-problem seeds for a master problem where binary variables represent whether or not to use a particular sub-problem solution (seed) in the solution for the overall problem. In order to assess the influence of various parameters on this algorithm, a real world case study based on forest inventory data from the Pennsylvania Kittanning State Forest (DCNR Bureau of Forestry District-8) was used.

The main objectives of the research were to demonstrate the application of the approach to a moderately-sized real-world problem and to evaluate the influence of different methods of generating sets of random seeds for the initial step of formulating the column generation master problem. The number and quality of the initial seeds used to initialize the column generation process has an impact on the solution quality and time. The number of initial seeds was set at five levels: 3, 5, 10, 25, and 50. The seeds were produced by randomly perturbing the shadow prices of the joint constraints in the original LP solution within various intervals. Four intervals of shadow price variations were tested and found to influence different output parameters. The four target levels tested were: Interval 1 (0.75, 1.25), Interval 2 (0.5, 1.5), Interval 3 (0, 2) and Interval 4 (-1, 3).
The forest inventory data were obtained from the Bureau of Forestry District-8. The forest was delineated into management units using the TUDM software developed by Petroski (2005). The forest was also decomposed into 9 spatially-independent compartments of uneven sizes. The column generation algorithm was implemented on this decomposed forest structure, and 200 test problems were created and solved to analyze the influence of number and quality of initial seeds and size of the shadow price perturbation interval. Different output parameters can be used to assess the performance of the algorithm. Hence, after solving the 200 problems, different output variables were compared to assess the influence of the number of initial seeds and the size of the shadow price perturbation interval. The output parameters analyzed were: RIMP objective function value, total solution time taken by algorithm, number of iterations required for the the shadow prices of the master problem to converge, the number of sub-problems used in the RIMP, the number of identical initial sub-problem seeds, and the RIMP solution time.

One of the primary objectives of applying column generation to large-scale MIP problems is to overcome long solution times. But optimal solution quality of the master problem is also important. The results of this study show that the price-directed decomposition approach produces solutions that are very close to the original IP solution. The variation in the master problem solutions for all 200 replication was relatively small. The 99% confidence interval for a sample mean of $15,992,695.37 was calculated and the margin of error found was mere $677. It means that we can be 99% confident that true mean of the population of objective functions that we might get from column
generation will be between $15,992,018.37 and $15,993,372.37. An optimal IP solution of the same problem being $16,003,391.13, it can also be said that we can be 99% confident that the true mean of the population of objective function will be better than $15,992,018.37 or the true mean of population of optimality gap will be as less as 0.071% from the original optimal IP solution. This shows that the algorithm maintains its solution quality in spite of randomness involved. Hence, the concern of optimality gap being taken care of, the next important thing to achieve was the shorter solution times.

The influence of the number of initial seeds and shadow price perturbations on the column generation solution time was analyzed. The solution time taken by CPLEX to solve the District-8 case study problem to an optimality gap of 0.01% was 246.88 seconds. The best average column generation solution time obtained from all the combinations of initial seeds and shadow price variations was 54.56 seconds in case of 3 initial seeds and the (0.75, 1.25) shadow price interval. From this, one might conclude that even fewer seeds might be better. To explore that possibility, an algorithm was developed with one and two initial seeds respectively but that resulted in an infeasible master problem.

Initially, the column generation algorithm was expected to perform better with more seeds. The logic for this hypothesis was that more seeds should result in a better approximation of the convex hull of the original problem and hence the algorithm should converge in fewer iterations. Also, Rajasekaran (2005) showed that a similar algorithm performed better with more initial seeds. However, Rajasekaran (2005) used a different approach to generate the random seeds. He randomly perturbed the targets in the joint
constraints whereas in this research a price-directed Lagrangian relaxation is used to develop the random set of initial seeds. Rajasekaran’s (2005) approach did achieve shorter solution times than CPLEX, but with fewer initial seeds the optimality gap between the column generation solution and the original IP solution increased. Rajasekaran (2005) also observed that the increase in the final solution time was directly proportional to the solution time taken by the sub-problems. Hence, in order to decrease the optimality gap between solutions, more seeds were required to initialize the solution. But that resulted in an increase in the solution time as more sub-problems were solved to obtain more initial seeds. Hence, in that case it was important to evaluate the trade-off between solution quality and solution time. However, the price-directed approach used in this research showed that it is possible to achieve both shorter solution times and a consistently low optimality gap.

In this study, solution times increased with the number of initial seeds while changing the shadow price interval size had less impact. Out of all 200 replications, the minimum solution time obtained was as low as 22.43 seconds, which is a considerable savings from the 246.88 seconds taken to solve the original IP problem in CPLEX. It is important, however, to compare the time taken by CPLEX to achieve a solution that is similar to that achieved by column generation. Even with this criterion, column generation performed better than CPLEX in terms of solution time. CPLEX took 83.05 seconds to achieve the solution just better than that achieved by column generation with 3 initial seeds and interval (0.75, 1.25). Hence, even for an equivalent solution, column generation performed faster than CPLEX. It can be concluded that the minimum number
of seeds that should be used to initialize the column generation procedure is 3. However, in order to confirm this finding, the same approach should be applied to additional forest case studies with different numbers of compartments and compartment sizes.

In addition to solution quality and times, a few other output parameters were analyzed. These parameters were significant because of their possible indirect impact on either solution quality or solution time. The number of iterations taken by the algorithm to converge might be an important factor as less iteration means fewer sub-problems to solve and also a smaller RIMP. Fewer sub-problems and a smaller RIMP can definitely affect the final solution time of the algorithm. The number of iterations decreased with an increase in the number of seeds. This is in contrast to the relationship between the total solution time and the number of initial seeds. One less iteration reduces the number of sub-problems by the number of compartments. For instance, in the case of 3 versus 50 initial seeds, the difference of 4 iterations means 36 (4 x 9) more sub-problems added to problem with 50 initial seeds. But the number of initial sub-problems solved for 3 seeds was 27 (3 x 9) is considerably less than 450 (50 x 9). Hence the number of initial seeds dominates the number of iterations in contributing to total solution time so the number of iterations taken by the algorithm to converge to an optimal solution is less critical.

Another output parameter analyzed was the number of sub-problems solved and used in the RIMP. The algorithm tends to create identical seeds during the initialization process which are rejected and not used in the RIMP formulation since they would be redundant. While these identical seeds are not used by the algorithm, they do increase the total solution time. The number of sub-problems that were actually used as variables in
RIMP increased with an increase in the number of initial seeds and in the shadow price perturbation interval size. An increase is natural with an increase in number of seeds and the increase with the shadow price interval size is mainly due to the increase in the number of iterations.

An interesting result considered here was the proportion of the total solution time taken by entire algorithm used in solving sub-problems. The average proportion for all 200 replications was 37.83%. The remaining column generation solution time is used to solve linear master problems and the final iteration integer master problem. The time taken by initial and iterative linear master problem was negligible. Hence, it can be said the RIMP took about 62% of the total solution time on average. This observation contributes to the domination of number of initial seeds in final solution time as discussed earlier. Increasing the number of initial seeds increases the number of variables in the RIMP and hence the RIMP size.

From the above discussion it seems that the number of identical seeds created during the initialization step should not be a point of concern. But the number of identical seeds generated increased drastically for 50 initial seeds and the (0.75, 1.25) interval size. Hence with a substantial number of initial seeds and a narrow shadow price interval size, it is likely that the algorithm will find it difficult to generate the unique seeds and in order to encounter the problem a stopping rule was built in the algorithm which stops the algorithm if it was not possible to generate a unique seed in 100 trials. Finally the solution time taken by the RIMP was analyzed and it matched with the trend followed by total solution time and number of sub-problems used in RIMP. Hence we can conclude
that the number of sub-problems in RIMP does influence the RIMP size and its solution time, which in turn affects the total solution time taken by the algorithm.

The solutions produced by the column generation algorithm were never better than the 99.9% solutions produced by CPLEX for the original problem. This was expected, as the problem size is not large enough to be difficult for CPLEX to handle. In fact, this problem was chosen because we expected to be able to solve it to a relatively tight gap with CPLEX so we would have a good idea of what the true optimal solution was. However, the time taken for CPLEX to solve the original problem tends to increase at least exponentially with increasing in forest size (Rajasekaran, 2005). This rate of increase will likely make many larger MIP problems intractable. Thus, CPLEX probably will not be able to solve larger problems to an acceptably tight gap within a reasonable amount of time. On the other hand, it is expected that column generation solution times will increase more-or-less linearly with an increase in the problem size. Additional convex constraints like road and fencing building constraints will likely make it even more difficult for CPLEX to handle the problems, so the addition of these types of constraints will make decomposition approaches like the one presented here even more important.
FUTURE OPPORTUNITIES

This research has developed an iterative column generation algorithm that uses shadow prices on the joint constraints of the original problem to dualize these constraints in the objective functions of the compartment-level and solutions. This approach conveys information from the strategic-level master problem to the sub-problems. A different initialization approach has been explored to get good seeds for the initial master problem formulation. The shadow prices of the LP relaxation of the original problem were randomly perturbed within a pre-determined interval and used to generate different seeds. There are many other ways to perturb the shadow prices, such as using normal distribution. Also this process of generating set of seeds could be extended to the first few iterations. The shadow prices of the joint constraints in the restricted linear master problem can be obtained and perturbed in a similar fashion to generate another set of seeds to be added to the master problem in the next iteration. A gradual decrease in the number of seeds generated in subsequent iterations could also be tested. This might help the algorithm converge sooner and hence reduce solution times.

It would be useful to generalize these results to different forest case studies. The approach can be implemented on forests with different compositions of management units and compartments. Rajasekaran (2005) showed that the number of compartments and their size do play a very important role in the performance of column generation. Hence, it would be interesting to see how the price-directed decomposition approach presented here works with different forest compositions. The size of the District-8 case
The study is also small. It would be good to see if column generation performs better with larger forest sizes.

The algorithm can be applied with a similar ease to cost minimization models as well. The algorithm is scaleable, i.e., can be extended to models with more joint and convex constraints. One interesting area of further research would be to integrate the column generation approach into problems which deal simultaneously with adjacency restrictions and road building.

The main complication in implementing the column generation algorithm in forestry problem is applying the Dantzig-Wolfe decomposition. When forests have a structure like the District-8 case study, i.e., with many buffer zones between management units, they can be easily decomposed into compartments. However, it is not always possible to find buffer zones in the forest and the presence of adjacency constraints can make the decomposition step more difficult. Hence, it would be interesting to find techniques to decompose forest on the basis of boundaries other than buffers zones. If there is no other way to decompose the forest into truly spatially independent compartments, one can look to find ways to handle joint constraints such as adjacency constraints in the master problem.

The column generation method breaks down when the sub-problems are intractable (i.e., if one cannot solve the sub-problem within reasonable time). In many applications, the sub-problems are difficult and cannot be solved with standard techniques. It is then necessary to develop a special purpose algorithm to solve the sub-
problems. There is definitely more work to be done to study the robustness of this methodology across different applications and to see how this approach compares with others.

*****
BIBLIOGRAPHY


Berck, P., and T. Bible. 1984. Solving large-scale harvest scheduling problems by

programming, simulated annealing, and tabu search heuristics for solving spatial harvest
scheduling problems. Forest Science 45:292-301.

heuristic techniques to solve spatial harvest scheduling problems. Forest Science 48:204-
218.

formulations for the vehicle routing problem with time windows. Dept of Industrial
Engineering and Management Science, Northwestern University.


ascent and column generation heuristic for the discrete lot sizing and scheduling problem

harvest scheduling with access road network development. Forest Science 46:204-218.


15(1):58-81


