Optimisation Models for a Financial Portfolio Problem

by

Calvin Bernard Lim

A thesis submitted in partial fulfillment for the degree of Honours in Mathematics

in the

FACULTY OF SCIENCE
DEPARTMENT OF MATHEMATICS AND STATISTICS

November, 2010
Declaration of Authorship

I, CALVIN BERNARD LIM, declare that this thesis titled, ‘Optimisation Models for a Financial Portfolio Problem’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 


Date:
"You’ve got to shoot for something. If you shoot for the stars and hit the moon, it’s OK. A lot of people don’t even shoot."

Robert Townsend
Abstract

FACULTY OF SCIENCE
DEPARTMENT OF MATHEMATICS AND STATISTICS

Honours in Mathematics

by Calvin Bernard Lim

This paper explores the asset allocation problem based on multi-stage stochastic programming optimisation under uncertainty. Two modeling frameworks namely the Non-Anticipativity (NA) Model and the Parent-Child (PC) model are presented. The PC model is an original formulation that takes advantage of the network structure of scenario trees, while the NA model is a popular framework in the field. A Benders reformulation is also introduced as a potential candidate for an efficient solution algorithm for this problem. Moreover, heuristic solvers are also explored as a means to tackle large scale problems. These models were then tested to measure performance and functionality.
Acknowledgements

Writing this thesis has been quite the memorable endeavor, with its own abundance of challenging tasks and interesting discoveries. It would not have been the same without the great insight, patient supervision and thoughtful mentoring of my supervisor, Dr. Heng Soon Gan to whom I am greatly thankful and indebted.

To my family, I owe this much. Thank you for your support and guidance, without which this journey would have been impossible. Your care, concern and understanding was a rich source of inspiration and strength in the toughest of times. Thank you for your prayers and your continuous encouragement.

To the kids of G69, thanks for a wonderful year of great friendship, laughter and camaraderie. We have scars from battles we fought together, and these I will cherish for a lifetime.

To Liv, Em, Sophie and Cayley, thanks for allowing me to use Room 143 as my base camp for several weeks, your gracious hospitality will never be forgotten.

And to Karla, Charlene and Jenna, a great many thanks for proofreading my thesis. You guys are pretty awesome.
## Contents

Declaration of Authorship iii

Abstract v

Acknowledgements vi

1 Introduction 1

2 Background Information 3
   2.1 Linear and Mixed Integer Programming .................................. 3
   2.2 Stochastic Programming .................................................. 4
   2.3 Stochastic Programming Models ....................................... 6
      2.3.1 Anticipative Models ............................................ 6
      2.3.2 Adaptive Models .............................................. 7
      2.3.3 Recourse Models ............................................. 8
   2.4 Solution Methodology .................................................. 9
      2.4.1 Benders Decomposition ......................................... 9
   2.5 Scenario Generation ................................................... 11
      2.5.1 Scenario Tree .................................................. 11
      2.5.2 Scenario Quality ............................................... 13
         2.5.2.1 Correctness ............................................. 13
         2.5.2.2 Accuracy ............................................... 13
         2.5.2.3 Consistency ............................................. 13
      2.5.3 Scenario Generation Methods ................................. 13
         2.5.3.1 Bootstrapping Data .................................. 14
         2.5.3.2 Statistical Analysis .................................. 14
         2.5.3.3 Moment Matching Methods ......................... 14
         2.5.3.4 Conditional Sampling .................................. 15

3 Stochastic Programming Model Formulation 17
   3.1 Non-Anticipativity Model ........................................... 18
      3.1.1 Model Definition ............................................. 18
      3.1.2 Parameters .................................................... 19
      3.1.3 Constraints .................................................... 19
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.4</td>
<td>Objective Function</td>
<td>22</td>
</tr>
<tr>
<td>3.2</td>
<td>Parent-Child Network Model</td>
<td>23</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Model Definition</td>
<td>23</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Parameters and Variables</td>
<td>24</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Constraints</td>
<td>25</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Objective Function</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>Benders Decomposition</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Motivation</td>
<td>27</td>
</tr>
<tr>
<td>4.2</td>
<td>Introduction and General Concept</td>
<td>27</td>
</tr>
<tr>
<td>4.3</td>
<td>Information Pass and Optimality Cuts</td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>Sub Problems</td>
<td>31</td>
</tr>
<tr>
<td>4.4.1</td>
<td>First Sub Problem</td>
<td>31</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Middle Sub Problem</td>
<td>32</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Final Sub Problem</td>
<td>34</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Infeasibility Problem</td>
<td>36</td>
</tr>
<tr>
<td>4.5</td>
<td>Xpress-MP Implementation</td>
<td>38</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Master Program</td>
<td>39</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Sub Routine 1 - SolveSubProblem</td>
<td>40</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Sub Routine 2 - SolveStartProblem</td>
<td>41</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Sub Routine 3 - SolveMiddleProblem</td>
<td>42</td>
</tr>
<tr>
<td>4.5.5</td>
<td>Sub Routine 4 - SolveFinalProblem</td>
<td>43</td>
</tr>
<tr>
<td>4.5.6</td>
<td>Sub Routine 5 - SolveInfeasibilityProblem</td>
<td>44</td>
</tr>
<tr>
<td>5</td>
<td>Heuristics</td>
<td>47</td>
</tr>
<tr>
<td>5.1</td>
<td>Simple Heuristic</td>
<td>49</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Description</td>
<td>49</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Example</td>
<td>49</td>
</tr>
<tr>
<td>5.2</td>
<td>Smart Heuristic 1</td>
<td>50</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Description</td>
<td>50</td>
</tr>
<tr>
<td>5.2.2</td>
<td>LP Formulation</td>
<td>51</td>
</tr>
<tr>
<td>5.3</td>
<td>Smart Heuristic 2</td>
<td>53</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Description</td>
<td>53</td>
</tr>
<tr>
<td>5.3.2</td>
<td>LP Formulation</td>
<td>54</td>
</tr>
<tr>
<td>5.4</td>
<td>Integration into Benders Decomposition Problem</td>
<td>55</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Simple Heuristic Integration</td>
<td>55</td>
</tr>
<tr>
<td>5.4.1.1</td>
<td>Start Sub Problem</td>
<td>55</td>
</tr>
<tr>
<td>5.4.1.2</td>
<td>Middle Sub Problem</td>
<td>56</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Smart Heuristic Integration</td>
<td>57</td>
</tr>
<tr>
<td>5.4.2.1</td>
<td>Description</td>
<td>57</td>
</tr>
<tr>
<td>5.4.2.2</td>
<td>Start Sub Problem</td>
<td>57</td>
</tr>
<tr>
<td>5.4.2.3</td>
<td>Middle sub problem</td>
<td>58</td>
</tr>
<tr>
<td>6</td>
<td>Discussion of Results</td>
<td>61</td>
</tr>
<tr>
<td>6.1</td>
<td>Data Set</td>
<td>61</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Generation Method</td>
<td>61</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Information and Properties</td>
<td>62</td>
</tr>
</tbody>
</table>
6.1.3 Scenario Trees ......................................................... 62
6.2 Experimental Results .................................................. 64
  6.2.1 PC vs NA Model ....................................................... 65
      6.2.1.1 Solve Time .................................................... 66
      6.2.1.2 Number of Variables and Constraints ..................... 66
      6.2.1.3 Summary ....................................................... 67
  6.2.2 Benders Algorithm Results ......................................... 67
      6.2.2.1 Solve Times ................................................... 68
      6.2.2.2 9ST to 128ST ................................................. 69
      6.2.2.3 144ST to 512ST .............................................. 70
      6.2.2.4 768ST to 2304ST ............................................ 71
      6.2.2.5 Summary ....................................................... 72
  6.2.3 Heuristics Results .................................................. 73

7 Conclusion and Further Research ..................................... 77
  7.1 Conclusion ............................................................ 77
  7.2 Further Research ..................................................... 79

A Computational Results .................................................. 81

Bibliography ................................................................. 87
For my parents, David and Belen,
and in the memory of my grandmother, Wong Yuk Yip,
Chapter 1

Introduction

Financial optimisation remains a very attractive area in the field of decision making under uncertainty. Recent noteworthy studies include the development of asset-liability management models for Japanese and American insurance companies [1], [2], asset allocation strategies for equity and fixed income portfolio managers [3], [4], [5], currency hedging options for multinational companies, and risk management for financial institutions [6].

In this thesis, we consider the problem facing an investor who seeks to construct a portfolio of varying financial instruments while facing an uncertain future. From all possible compositions, the investor picks the portfolio that allows him to maximise his returns according to his risk aversion preferences. This investor faces many uncertainties such as unknown asset returns, interest rate fluctuations and economic factors. Moreover, the financial rewards for good decisions and penalties for bad decisions are enormous. Investors are highly compensated for identifying successful investment strategy decisions as compared to their benchmark counterparts. As such, there is a great need for financial models that accurately capture the uncertainties and provide superior returns [7], [8].

The use of multi stage stochastic linear programming is ideal in addressing these issues for several reasons. First, these models are able to account for portfolio rebalancing at future time periods when responses to new information become necessary. Second, these models can formulate specific and detailed strategies that allow the investor to hedge his returns across all plausible scenarios rather than simply provide a broad or static recommendation. Lastly, these models easily take into account realistic market imperfections such as transaction costs, borrowing restrictions, taxes or other regulatory requirements.

Furthermore, uncertainty is discretized and captured in these models by the use of a scenario event tree. The main drawback is that the size of the problem grows exponentially as more time periods or scenarios are included in the problem. This forces many researchers to limit
the number of scenarios and time periods in their scenario event trees. And although the continued development of computing capabilities have significantly increased the size of the models that can be solved, the level of uncertainty that is accounted for in these problems is still relatively small. Thus, much of the research effort today is focused on the design and creation of efficient solution algorithms that provide computational ease and simplicity.

Moreover in this thesis, the three main elements of the financial portfolio problem are considered. This includes a mathematical model based on stochastic linear programming, a generation method for a scenario tree of possible outcomes, and a solution algorithm. And thus, the rest of the thesis is organized as follows:

Chapter 2 explores basic linear and stochastic programming theory. Different stochastic programming models are also introduced, as well as solution methodologies and scenario generation techniques and guidelines. Chapter 3 explores two formulations for the portfolio optimisation problem. The first model is based on previous work by Mulvey [9], while the other model is an original formulation based on the framework on tree networks.

Chapter 4 explores the use of Benders Decomposition method as a solution algorithm to the stochastic programming model. Computer implementation of this method is also discussed in much detail. On the other hand, Chapter 5 introduces three heuristics for solving this problem. These heuristics come in handy when large scale problems become impossible to solve analytically. Chapter 6 presents the computational results from the implementation of the different basic models, solution techniques and heuristics. Lastly, in Chapter 7, conclusions and directions for further research are laid out.
Chapter 2

Background Information

2.1 Linear and Mixed Integer Programming

Linear Programming (LP) is a standard method of solving optimisation problems where the feasible solution space is a convex polytope defined by linear equations. All LP’s include an objective function that we wish to either maximise or minimise. The LP method also requires another set of linear equations and linear inequalities commonly known as constraints that define the solution space polytope.

The standard form for an LP where we seek to maximise the objective function is presented as follows. As we present a maximisation problem, the constraints are defined as less than or equal to type inequalities. Otherwise, greater than or equal to constraints are used. The standard form then is as follows,

\[
\begin{align*}
\text{max} & \quad z = c^T x \\
\text{s.t.} & \quad Ax \leq b
\end{align*}
\]

(2.1) (2.2)

where \( x \) represents a vector of decision variables, \( c \) a vector of cost coefficients, \( A \) a matrix of coefficients in the constraint set, and \( b \) a vector of bounding values.

It is important to note here that the term ‘s.t.’ stands for ‘subject to’, as the objective function \( c^T x \) is maximised subject to the set of constraints \( Ax \leq b \).

For the remainder of this thesis, we impose that all decision variables are required to be non-negative. That is, the constraint, \( x \geq 0 \), is implied for all such decision variables \( x \) in future models.

LP problems are commonly solved using the Simplex Algorithm developed by George Dantzig [10]. This was first proposed by Dantzig after his work with the military during World War II. Implementing the Simplex Algorithm to solve an LP can result in an
exponential number of steps in the worst case scenario. Nonetheless, in practice it is generally very efficient.

Essentially, the Simplex Algorithm solves an LP by searching along the edges of the convex polytope or the space of feasible solutions. From basic LP theory, it is known that the optimal solution is either an extreme point or an extreme ray of the convex polytope. As such, it is enough to only search the boundary of the LP’s feasible region.

Moreover, mixed Integer Programming (MIP) is a method that finds its roots in LP, where MIP is simply an extension of the LP problem that requires integer restricted variables. This was first introduced by R.E. Gomory in 1958, when he developed a new method for generating constraints called “cutting planes”. These constraints prohibit non-integer solutions for the integer restricted variables. The new convex polytope with only integer feasible solutions is also known as the convex hull of the integer feasible region.

However, a MIP solution is not found by immediately solving the corresponding LP with the integer constraints removed. Often, it is not desirable to merely round the LP relaxation variable values to an integer. A new method such as cutting planes to impose integer solutions is required. However, this may not be straightforward, as working from an LP relaxation solution towards a MIP solution using any method is NP-hard.

2.2 Stochastic Programming

In this thesis, I have adopted the work of Sen and Higle [11], who explore the introductory theory of Stochastic Programming (SP) in extensive detail.

SP is a modeling framework for optimisation problems which deal with uncertainty. While all parameters are known in a deterministic optimisation problem, modeling real world problems almost always include some level of uncertainty. As such, SP models use the fact that probability distributions for the given data in a problem are known or can be estimated using statistical methods. Essentially, the objective is to find a course of action or a set of decisions that is feasible for all possible scenarios or outcomes. Similar to LP’s and MIP’s , SP models maximise or minimise the expectation of an objective function made up of decision variables and other random variables. These models are formulated and solved either numerically or analytically in order to provide an a priori strategy for the decision maker faced with uncertainty.

The most commonly found and studied SP models are simple two stage linear programs. In these problems, the decision maker makes certain decisions in the first stage, after which a random event plays out that affects the outcome of the first stage decisions. Then, recourse
decisions can be made in the second stage in order to fix any unfavourable outcomes that might have resulted from making the first stage decision. The optimal strategy in this example is a set of decisions that need to be made both in the first stage and in the second stage. This idea can be extended to more than two stages into what is commonly known as multi-stage SP modeling. A formulation of a SP problem is introduced as follows with the definition of Birge and Louveaux [12].

Recall that a deterministic LP is defined as,

\[ \max \quad z = c^T x \]
\[ \text{s.t.} \quad A x \leq b \]

where \( x \) is an \((n \times 1)\) vector of decisions and \( c, A \) and \( b \) are known data of the sizes \((n \times 1),(m \times n)\) and \((m \times 1)\). Here, all decisions are captured by the variable \( x \). Now, let us proceed to a two stage problem with fixed recourse by Dantzig [10] and Beale [13],

\[ \max \quad z = c^T x + E_{\zeta}[\min q(\omega)^T y(\omega)] \]
\[ \text{s.t.} \quad A x \leq b \]
\[ T(\omega)x + W y(\omega) \leq h(\omega) \]
\[ y(\zeta) \geq 0 \]

The vector \( x \) represents the decision variables in the first stage. However, this model considers a representation of a number of random events \( \omega \in \Omega \). Thus, for a given realisation \( \omega \), the previously unknown \( q(\omega), h(\omega) \) and \( T(\omega) \), which are data in the second stage, are revealed. The dimensions of these vectors are \( q(\omega) \) is \( n_2 \times 1 \), \( h(\omega) \) is \( m_2 \times 1 \) and \( T(\omega) \) is \( m_2 \times n_1 \), where each possible component is a random variable. Combining all the stochastic elements in the second stage produces the vector \( \zeta^T(\omega) = (q(\omega)^T, h(\omega^T), T_1(\omega), \ldots, T_{m_2}(\omega)) \). As a result, we have now transformed the optimisation model to also consider future scenarios dependent on the different values of \( \zeta \) when the first stage decision \( x \) is made.

According to Kaut and Wallace [14], SP modeling has been gaining increased popularity within the mathematical modeling community. This is because modern computing capabilities allow users to add a stochastic element to models that have previously been difficult to solve as SP models due to being too large or complex. As such, SP models can be regarded as mathematical programming whilst some parameters or data are uncertain. Instead of single, non-probabilistic values, these random parameters or data are then described by probability distributions or by stochastic processes, where \( \zeta \) is a random vector, whose distribution must be independent of the decision vector \( x \).
SP models have a special structure where different sets or groups of constraints are considered in different scenarios. Thus, this can be taken advantage of when solving problems. Heuristics that exploit this structure can perform better and faster than just by solving the SP problem itself. This is done by implementing several decomposition algorithms including the Benders Decomposition Method used in this thesis.

As described by Hochreiter [15], the SP problem usually cannot be solved with continuous distributions. Thus, in order to solve such problems numerically, the stochastic problem has to be discretized, most commonly by calculating a useful discretization to generate a scenario tree from the underlying stochastic process. Furthermore, a scenario generator is employed to generate different values for $\omega \in \Omega$. This set of different $\omega$ values are then used as scenarios in a SP problem.

### 2.3 Stochastic Programming Models

The anticipative and adaptive models are special cases of stochastic programs the combination of which produces a recourse model commonly used in the financial industry. For this section, I have adopted the definition prescribed by Yu, Ji and Wang [16].

#### 2.3.1 Anticipative Models

Anticipative models are also referred to as static models. These models are static because the decisions made by the decision maker do not depend on future observations of the environment. Thus, while making a decision the decision maker has to take into account all possible future outcomes, as no opportunities arise to reverse these decisions later on.

With anticipative models, constraints that are probabilistic in nature, also known as chance constraints, are used to govern feasibility. For example, if we set a reliability level $\alpha$, where $0 \leq \alpha \leq 1$, then the constraints can be expressed in the form,

$$P\{\omega | f_j(x, \omega) = 0, j = 1, 2, \ldots, n \} \geq \alpha \quad (2.9)$$

where $f_j : \mathbb{R}^m \times \Omega \rightarrow \mathbb{R}, j = 1, 2, \ldots, n$, and $x$ is the m-dimensional vector of decision variables.

Moreover, the objective function can also be expressed as

$$P\{\omega | f_0(x, \omega) \leq \eta \} \quad (2.10)$$

where $f_0 : \mathbb{R}^m \times \Omega \rightarrow \mathbb{R} \cup \{+\infty\}$ and $\gamma$ is a constant.
In essence, an anticipative model will select a strategy or a decision policy that would meet the desirable characteristics of the constraints and the objective function. Desirability, as in the example above, is defined as the probability of a constraint violation being less than \((1-\alpha)\). The choice for the value of \(\alpha\) is dependent on the current model at hand, the cost of the constraint violations and other similar considerations.

### 2.3.2 Adaptive Models

The essential difference with an adaptive model is that information related to the random outcomes becomes partially available before the decision making process is undertaken.

Now, let \(\mathcal{A}\) be a subfield of all possible events, which is the set composed of the collection of all the relevant information available through observation. Observe that the decision variable \(x\) depends on the observable events, and hence we call \(x, \mathcal{A} -\text{adapted}\) or \(\mathcal{A} -\text{measurable}\). Thus, an adaptive stochastic program is formulated as follows,

\[
\min E[f_0(x(\omega), \omega)|\mathcal{A}] \tag{2.11}
\]

\[s.t. \ E[f_j(x(\omega), \omega)|\mathcal{A}] = 0 \quad j=1, \ldots, n \tag{2.12}\]

\[x(\omega) \in X \quad \text{almost surely} \tag{2.13}\]

where the mapping \(x : \Omega \to X\) is such that \(x(\omega)\) is \(\mathcal{A} -\text{measurable}\). This problem is solved by solving for every possible \(\omega\) in the following deterministic program,

\[
\min E[f_0(x(\cdot)|\mathcal{A})(\omega) \tag{2.14}\]

\[s.t. \ E[f_j(x(\cdot)|\mathcal{A})(\omega) = 0 \quad j=1, 2, \ldots, n \tag{2.15}\]

\[x \in X \tag{2.16}\]

It is interesting to note to the reader that the two extreme cases either when no information or complete information is available are worth examining in more detail. The case where there is no information simply reduces to the anticipative form, while complete information is simply just a distribution model that characterizes the distribution of the optimal objective value. Nonetheless, we are more interested in the case when only partial information is available to the decision maker. This is introduced in the next subsection.

### 2.3.3 Recourse Models

Recourse models seek to merge the anticipative and adaptive models into one mathematical framework. In this framework, the decision maker seeks a strategy or a decision policy that
will not only anticipate future observations, but also exploit partially available information to make recourse decisions. For example, a stock portfolio manager would consider both the future movements of stock prices as well as the rebalancing of portfolios when these prices change. The former is an anticipative, while the latter is an adaptive course of action.

Hence, the two stage stochastic programming problem with recourse can then be formulated as follows,

\[
\begin{align*}
\min & \quad f(x) + \mathbb{E}[Q(x, \omega)] \\
\text{s.t.} & \quad Ax = b \\
& \quad x \in \mathbb{R}^{m_0} + \end{align*}
\]  

(2.17)

(2.18)

(2.19)

where \(x\) are the first stage anticipative decisions made before the random events occur, and \(Q(x, \omega)\) is the optimal value, for any given \(\Omega\), of the following non-linear program,

\[
\begin{align*}
\min & \quad q(y, \omega) + \mathbb{E}[Q(x, \omega)] \\
\text{s.t.} & \quad W(\omega)y = h(\omega) - T(\omega)x \\
& \quad y \in \mathbb{R}^{m_1} + \end{align*}
\]  

(2.20)

(2.21)

(2.22)

Here, \(y\) are the second stage adaptive decisions made after the random event has occurred. The value of \(y\) is dependent on the realization of the first stage random vector. Also, \(q(y, \omega)\) denotes the second stage cost function, and \(\{T(\omega), W(\omega), h(\omega) | \omega \in \Omega\}\) are model parameters. Those parameters are random parameters as these are functions of the random vector \(\omega\). Likewise, \(T\) is the matrix that transforms the first stage decision \(x\), into resources for the second stage problem. An example from a portfolio problem would be that \(T\) is the matrix containing the interest rates, and would thus convert capital in the first period to capital with interest earnings in the second stage. Lastly, \(W\) is the recourse matrix and \(h\) is the second stage resource vector.

Generally, the two stage recourse model can be written as follows:

\[
\begin{align*}
\min & \quad f(x) + \mathbb{E}[\min_{y \in \mathbb{R}^{m_1}} \{q(y, \omega)|T(\omega)x + W(\omega)y = h(\omega)\}] \\
\text{s.t.} & \quad Ax = b \\
& \quad x \in \mathbb{R}^{m_0} + \end{align*}
\]  

(2.23)

(2.24)

(2.25)

Of course, the recourse problem is not restricted to the two stage formulation. This can be extended into a multi-stage program, which we introduce in Chapter 3.
2.4 Solution Methodology

2.4.1 Benders Decomposition

Benders Decomposition [17] is a classical method for solving optimisation problems based on delayed constraint generation and problem partitioning. The idea behind this method is that if we can fix certain complicating variables, the initial problem can be partitioned into two simpler problems, the master problem and the sub problem, that can be solved separately. The master problem is a relaxed version of the initial problem where a smaller subset of constraints and variables are considered while the sub problem is the initial problem with the variables in the master problem fixed.

In this method, the master problem and the sub problem are solved iteratively. After the master problem is solved, solutions are passed on the sub problem. With the complicating variables fixed with the values obtained from the master problem, the sub problem is then solved to optimality. Afterwards, a cut is produced, and this is added to the master problem where it is again solved to optimality.

Observe that during the first iteration, an arbitrary value for the complicating decision variables is used. In the case of maximising (minimising) an objective, the sub problem gives a valid lower (upper) bound as a restriction of the initial problem, while the master problem gives a valid upper (lower) bound as a relaxation of the initial problem. This is because the convergence condition is such that if difference between the current lower bound and minimum upper bound is less than or equal to an arbitrary value, then the algorithm terminates.

A Benders Decomposition of a minimisation problem based on the work of Saharidis and Ierapetritou [18] is presented as follows.

Consider the following LP where vector $y$ contains a number of decision variables considered as complicating variables.

\begin{align}
\min \quad & c^T x + d^T y \\
\text{s.t.} \quad & Ax + By \leq b \\
& Fy \leq p
\end{align}

Applying Benders Decomposition partitions the decision variables into two sets $x$ and $y$. The problem is decomposed into a master problem and a series of sub problems. All the constraints with $x$ involved are included in the sub problem and all the constraints with $y$ involved belong to the master problem. Thus, if we fix the complicating variables $y = \bar{y}$ in
the initial problem, the resulting sub problem has the following formulation.

\[
\begin{align*}
\min & \quad c^T x + d^T \bar{y} \\
\text{s.t.} & \quad Ax \leq b - + B \bar{y}
\end{align*}
\]  
(2.29)  

And associating dual variables \( u \) to the constraints, we can write the dual version of this problem as,

\[
\begin{align*}
\max & \quad (b - B \bar{y})u \\
\text{s.t.} & \quad A^T u \leq c
\end{align*}
\]  
(2.31)  

This is the Benders sub problem. Note that the feasible space of the subproblem is independent of the variable \( y \). Thus, only the objective function changes with respect to the value of \( y \).

Furthermore, if the value obtained from the master problem is feasible, then the dual sub problem produces a bounded solution corresponding to an extreme point in its solution space. On the contrary, if the value obtained from the master problem is infeasible, then the dual sub problem produces an unbounded solution corresponding to an extreme ray in the solution space.

This allows for two types of Benders cuts can be made based on whether the dual sub problem solution is bounded or not. First, if the solution is bounded and the optimality condition is not satisfied, then the following optimality cut is added to the master problem.

\[
z \geq (b - B y) \bar{u}_i^T + d^T y, \quad i = 1, 2, \ldots
\]  
(2.33)

where \( \bar{u}_i \) is the vector that corresponds to the extreme point \( i \). Otherwise, if the solution is unbounded, then the following feasibility cut is added to the master problem.

\[
(b - B y) \bar{v}_j^T \leq 0, \quad j = 1, 2, \ldots
\]  
(2.34)

where \( \bar{v}_j \) is the vector that corresponds to the extreme ray \( j \). After generation of the Benders cuts, the master problem can now be reformulated as follows,

\[
\begin{align*}
\text{Min} & \quad z \\
\text{s.t.} & \quad F y \leq p \\
z & \geq (b - B y) \bar{u}_i^T + d^T y, \quad i = 1, 2, \ldots \\
(b - B y) \bar{v}_j^T & \leq 0, \quad j = 1, 2, \ldots
\end{align*}
\]  
(2.35)  

(2.36)  

(2.37)  

(2.38)


Chapter 2. Background Information

2.5 Scenario Generation

The representation of uncertainty for stochastic elements such as input parameters or environmental data for the portfolio optimisation problem is a very crucial component in the modeling process. The key random input parameters in a financial portfolio management problem are the asset prices and their equivalent returns, as well as cash interest rates within the planning horizon. Possible evolutions of these random parameters during the time horizon are accounted for in terms of discrete joint outcomes that are mapped to the nodes of a scenario tree, such as the one shown in Fig 2.1. Each branch in the tree is simply characterized by a realisation of asset prices and the interest rate.

A major benefit of using SP is that these models are not tied down to any assumptions for the random variables. Hence, SP’s can accommodate arbitrary discrete distributions captured by the use of scenario trees. This can be done so long as the scenarios generated conform to fundamental financial principles, specially being free from arbitrage opportunities. An extensive discussion of alternative scenario generation methods are presented by Dupacova [19] and Kaut [20].

2.5.1 Scenario Tree

The investment horizon is divided into time periods \( t = 0, 1, \ldots, \tau \), where each time period indicates a time at which decisions for portfolio rebalancing can be made. Each scenario tree will have a depth that is equal to the number of investment periods or decisions stages. The parent node at \( t = 1 \) corresponds to the current or initial state at \( t = 1 \). As such, all the input parameters and environmental data at this parent node are known and deterministic.

From the parent node, branches extend out to capture progressive evolutions in the values of the random variables at each time period. Thus, branches coming from the parent node reflect the possible outcomes during the second time period at \( t = 2 \). Then, each outcome is further associated with an immediate successor node in the next time period at \( t = 3 \).

Each scenario completes a sequence of joint realizations of the random variables during the planning horizon. Thus, each scenario is then characterized by a terminal node of the tree. The realizations of the random variables at every time period are captured by traversing from one node to another. Thus, this forms a unique path from the parent node to a terminal node, where each unique path is associated with a scenario. An example of a scenario tree is presented in Fig 2.1. We proceed to introduce the following notation:
The probability, \( \pi_n \), of a certain node \( n \in N \) is determined by multiplying the successive probabilities of each branch along the path as follows,

\[
p_1^n p_2^n \cdots = \pi_n \quad (2.39)
\]

Also, probabilities of all distinct nodes at each time period must sum to one,

\[
\sum_n p_t^n = 1, \quad t \in T \quad (2.40)
\]

We define distinct recourse variables, and corresponding constraints, to model the portfolio rebalancing decisions at each intermediate node of the scenario tree. That is, each node in the tree will have its own set of decision variables. However, at the leaf nodes, \( N_\tau \), we only compute the final value of the portfolio under each scenario. The size of the resulting multi-stage stochastic program grows substantially with the number of tree nodes. In this thesis, we pay close attention to the branching factor from each node, as well as to the total number of scenarios in the tree. These variations in tree size provide a good experimental data set, upon which SP models can be tested.
2.5.2 Scenario Quality

Zenios [21] defined three important criteria in the evaluation of scenario generation quality. These are correctness, accuracy and consistency.

2.5.2.1 Correctness

Scenarios should contain properties that adhere to fundamental financial theory. For example, the term structure of interest rates should exhibit mean revision and changes. This term structure should also consist of changes in level and slope, as well as curvature. Scenarios should also cover all relevant past history. Lastly, scenarios should account for events that were not observed, but are plausible under current market conditions. This is a very important criteria, especially in the wake of the global financial crisis where mathematical models failed to account for black swan events.

2.5.2.2 Accuracy

Since scenarios represent a discretization of a continuous phenomenon or process, accumulation of errors in the discretization is common. Several approaches can be used to make sure that the generated scenario will maintain the same theoretical continuous distribution. For example, accuracy is ensured when we are able to match the moments from the sample scenario set to its theoretical distribution. However, imposing this criteria may lead to a large number of scenarios generated. Hence, there will always be a trade off between ensuring accuracy, and maintaining modest computational complexity.

2.5.2.3 Consistency

When the scenario generation process requires the generation of various different asset classes, it is important to make sure that the scenarios are internally consistent. For example, scenarios in which there is an increase in the interest rate as well as an increase in the bond prices are inconsistent. A good way to ensure consistency is to take into account the correlation between different financial instruments.

2.5.3 Scenario Generation Methods

Ross [22] and Mitra [23] explore the different scenario generation methodologies developed today in much detail. They defined three broad classes of scenario generation techniques. The first approach, bootstrapping, is the simplest approach used and is only performed by
sampling on the already observed data set. A second approach models historical data using
statistical analysis. Here, a probability distribution is fitted to the data and sample sce-
narios are then drawn from that distribution. Lastly, a third approach develops continuous
time theoretical models with parameters estimated to fit the historical data. Finally, all
three models are then discretized and simulated to generate scenarios.

2.5.3.1 Bootstrapping Data

The simplest approach for generating scenarios, without having to use any mathematical
modeling, is to bootstrap available historical data. In this method, each scenario is simply
composed of a sample of returns of the assets obtained by sampling from the historical
data. For example, to generate a nine scenario tree, one has to simply sample nine sets
of historical data returns. This process can be repeated to generate as many scenarios
as needed. This approach preserves the observed correlation between the different asset
classes as well. However, this method will not satisfy the correctness demand as it will
never suggest a monthly return for a scenario that was never observed. Nonetheless, when
sampled correctly, the scenarios satisfy accuracy and consistency as these scenarios model
real life observations.

2.5.3.2 Statistical Analysis

Time series models relate the value of variables at given points in time to the value of these
variables at previous time periods. This method is suitable for solving large scale portfolio
optimisation problems when correlation among the different asset classes is very important.
Time series analysis can also be extended to model correlations among variables such as
yield curves and short rates.

2.5.3.3 Moment Matching Methods

A method of moments method is usually used then the marginal distributions for the
scenario generation process is not known. In practice, this method would usually entail
pairing up the first three or four moments - the mean, variance, skewness and kurtosis -
of the scenario generation process. These methods can also be extended to other statisti-
cal properties such as percentiles or higher co-moments as well. Afterwards, the moment
matching scenario generator will then construct a discrete distribution satisfying the se-
lected statistical properties. These approaches are widely used in the industry, as they are
very easily implemented [24].
Moreover, moment matching methods ensure accuracy by definition as they matches statistical moments. However, correctness is not always ensured as the approach is general and does not automatically reflect accepted financial theories.

### 2.5.3.4 Conditional Sampling

Conditional sampling methods are the most common methods used today. For every scenario tree node, we sample several values from the stochastic process \( \{\xi_1\} \). This is carried out either by sampling directly from the distribution of \( \{\xi_1\} \), or by evolving the process according to an explicit formula,

\[
\xi_{t+1} = z(\xi_t, \epsilon_t)
\]  

(2.41)

Traditional sampling methods are limited to sampling from univariate random variables. Thus, when sampling random vectors is required, marginals need to be sampled separately, and then combined afterwards. Usually, the resulting vector is a vector of independent random variables.

However, the obvious problem with this method is that the size of the tree grows exponentially with the dimension of the random vector. That is, if we sample \( s \) scenarios for \( k \) marginals, we will end up with \( s^k \) scenarios.

Lastly, when examining the quality of a sampling method, the most common problem is a lack of scenarios, as an increasing number of scenarios will result in the discrete distribution converging to the true distribution. By increasing the number of scenarios then, the trees will get closer and closer to the true distribution. As a result, both the instability and the optimality gap decreases, ensuring that the accuracy criteria is met.
Chapter 3

Stochastic Programming Model
Formulation

In this chapter, we examine the asset allocation problem as various formulations of a multi-stage decision problem with recourse, where transactions take place at discrete time periods. Although it is already intuitive that multi-stage models provide superior performance over single period models, extensive work has also been done by Berger and Mulvey [25], Dempster [26], Dert [27] and Klaassen [28], to validate this fact. Yu et al. [16] in their survey paper presents a more comprehensive record of the research done in this area.

Many applications can be made as special cases of this problem. One such application is in the insurance industry, where Carino et al. [1] and Carino and Zeimba [2] developed an asset/liability management (ALM) model for the Japanese insurance company Yasuda Kasai. In addition, Worzel et al [29], Holmer [30], and Sweeney [31] explore different ALM models for U.S. insurance companies Metropolitan Life Insurance, Federal Mortgage Association, and Falcon Asset Management.

Another application is the banking and pension fund industry. Mulvey [32] explores an approach for generating scenarios for pension funds Towers Perrin, while Kusy and Ziemba [6] developed SP models used in banks, and Hiller and Eckstein[3], Zenios [4], and Golub [5] describe SP models for fixed-income securities.

This problem is most commonly represented by a scenario tree as shown in fig 2.1. Whilst some real world processes and restrictions prove difficult to model using this network framework, these issues can be handled by using general linear constraints. Thus, the network model also provides an excellent visual reference for the asset allocation problem.
Chapter 3. Stochastic Programming Model Formulation

3.1 Non-Anticipativity Model

The first model that we introduce is the Non-Anticipativity Model. This model has been extensively studied and for this thesis, I have adopted Mulvey’s [9] formulation of the problem.

3.1.1 Model Definition

Firstly, we split the investment horizon into $\tau$ discrete time periods represented by $T = \{1, 2, \ldots, \tau\}$, where $t = 1$ represents the current time, and $t = \tau$ represent the final stage of the investment horizon. In this model, we are particularly interested in the investor’s position at each of the discrete time periods. Also, we assume that portfolio rebalancing decisions occur at the first instance of each decision stage.

Time intervals are dependent on the specific problem being modeled. A day trader might consider his time interval as short as seconds or minutes, while a pension fund manager will be more interested in longer intervals between decision stages. Furthermore, it is also possible for the time intervals to vary or change over time. For example, we can have short time intervals at the beginning time periods, and successively longer intervals towards the end of the investment horizon. For this thesis, each time interval spans one month, and three, four, and five stage problems are considered.

We also define asset investment categories as the set $A = \{1, 2, \ldots, I\}$. This set must represent a broad category of investment options including stocks, bonds, real estate, foreign currency or cash. For this thesis, five asset categories are considered.

Category 1 is cash, represented by the U.S. 13 Week Treasury Bill rate. Category 2 is stock, represented by the S&P 500 index. Category 3 and 4 are bonds, represented by the U.S. 5-Year and 30-Year Treasury Bond, respectively. Lastly, Category 5 is real estate, represented by the U.S. Housing Price index. These categories all track major financial markets and market indices. We ideally wish for the categories to be mutually independent, meaning that ideally the co-movements between pairs of asset categories are low, such that we can carry out a diversification strategy across all asset categories.

As discussed in Chapter 2, uncertainty is captured by discretizing it into a very large number of scenarios. Here, each scenario is simply the realization of a single set of outcomes across the investment horizon from $t = 1$ to $t = \tau$. Thus, we let $S$ be the set of scenarios, where $s \in S$ is a distinct realization of the nodes. A scenario can also be thought of as a path starting from the parent node to a terminal node in the tree. According to Maranas et
al. [33], typically at least five hundred to a thousand scenarios are required to adequately capture uncertainty in future outcomes.

3.1.2 Parameters

For each \( i \in A, t \in T \), and \( s \in S \), we define the following parameters and decision variables.

**Parameters:**

\[ r^s_{i,t} = 1 + \rho^s_{i,t}, \]  
where \( \rho^s_{i,t} \) is the percent return for asset \( i \), at time period \( t \), and under scenario \( s \).

\( \pi_s \) Probability that scenario \( s \) occurs, where \( \sum_{s=1}^{S} \pi_s = 1 \)

\( \omega_0 \) Wealth in the beginning of time period 1.

\( \sigma_{i,t} \) Transaction costs incurred in rebalancing asset \( i \) at the beginning of time period \( t \)

\( \beta^s_t \) Borrowing rate in period \( t \) under scenario \( s \).

**Decision Variables:**

\( x^s_{i,t} \) Amount of money for asset category \( i \), in time period \( t \), under scenario \( s \), after rebalancing.

\( v^s_{i,t} \) Amount of money for asset category \( i \), in time period \( t \), under scenario \( s \), before rebalancing.

\( \omega^s_t \) Wealth at the beginning of time period \( t \), under scenario \( s \).

\( p^s_{i,t} \) Amount of asset purchased for rebalancing in period \( t \), under scenario \( s \).

\( d^s_{i,t} \) Amount of asset sold for rebalancing in period \( t \), under scenario \( s \).

\( b^s_t \) Amount of money borrowed in period \( t \), under scenario \( s \).

3.1.3 Constraints

The primary decision variable \( x^s_{i,t} \) represent the amount of capital invested in asset \( i \) at the end of time period \( t \), under scenario \( s \). In this model all assets are denominated in U.S. dollars. Nonetheless, this basic model can be readily extended into a multi currency model at the expense of model size and complexity.

Furthermore, we assume that portfolio rebalancing decisions occur at the beginning of each time period. Alternatively, another possible decision policy is to simply make no rebalancing
transactions except to reinvest any interest earned from the previous time period. For simplicity, we impose that interest earning are reinvested in the respective asset categories, unless otherwise reinvested in another asset category. We also impose that borrowing be done on a single time period basis. As such, borrowing from \( t - 1 \) must be settled at time \( t \). Hence, the \( x \) vector depicts the state of each decision stage after rebalancing decisions have been carried out.

At the first time period \( t = 1 \) the investor’s total assets are equal to the initial investment capital plus any borrowings made,

\[
\sum_{i} x_{t,1}^s = \omega_0 + b_{1}^s, \quad s \in S
\]  

(3.1)

and equivalently, the investor’s total assets at \( t = \tau \) are equal to,

\[
\sum_{i} x_{t,\tau}^s = \omega_\tau^s, \quad s \in S
\]  

(3.2)

A borrowing limit is also imposed at every time period. In general, the investor is only allowed to leverage up to \( \phi \) times his total capital at that time period. For this thesis, we have set \( \phi = 2 \), meaning that the investor is only allowed to borrow up to twice his capital at that time period. The borrow limit constraint is thus defined as,

\[
b_{t}^s \leq \phi \sum_{i \in A} v_{t,t}^s, \quad s \in S, t \in T \setminus \{1\}
\]  

(3.3)

and at \( t = 1 \),

\[
b_{1}^s \leq \phi \omega_0, \quad s \in S
\]  

(3.4)

Moreover, the wealth accumulated from interest earnings at the beginning of the \( t^{th} \) period is accounted for using the variable \( v_{t,t}^s \). Thus, the interest earning constraint is given by the amount of capital after rebalancing at \( t - 1 \) multiplied to 1 plus the interest rate,

\[
x_{i,t-1}^s (1 + r_{i,t-1}^s) = v_{i,t}^s, \quad s \in S, t \in T \setminus \{1\} \ i \in A \setminus \{1\}
\]  

(3.5)

Also recall that rebalancing decisions are carried out at the very start of each time period. Hence, the rebalancing constraint, except for cash, is defined as the amount of capital before rebalancing plus any additional investment, and minus any reduced investment,

\[
x_{i,t}^s = v_{i,t}^s + p_{i,t}^s - d_{i,t}^s, \quad s \in S, t \in T \setminus \{1\} \ i \in A \setminus \{1\}
\]  

(3.6)

Cash flows are also restricted at each period to be consistent with the purchasing and selling of assets. The amount of cash after rebalancing is then equal to the amount of cash before rebalancing minus the purchase costs of other assets and settlements of borrowings and plus
the proceeds from sale of assets and new borrowings. Transaction costs in the purchase and selling of assets are also accounted for by using the \((1 - \sigma_{i,t})\) and \((1 + \sigma_{i,t})\) terms. Thus, the cash flow constraint is defined as,

\[ x_{s,t}^s = v_{s,t}^s + \sum_{i \neq 1}^n d_{i,t}^s(1 - \sigma_{i,t}) - \sum_{i \neq 1}^n p_{i,t}^s(1 + \sigma_{i,t}) - b_{t-1}^s(1 + \beta_{t-1}) + b_t^s, \quad s \in S, t \neq 1 \quad (3.7) \]

It is important to note here that no rebalancing transactions or borrowings are made at time period \(\tau\). That is, we also require that at period \(\tau\), purchasing and selling of any assets, as well as borrowing is zero. This is because further adjustments at the terminal nodes of the tree are no longer necessary. These rebalancing constraints at the last time period is defined as,

\[ p_{i,\tau}^s = 0, \quad s \in S, i \in A \setminus \{1\} \quad (3.8) \]

\[ d_{i,\tau}^s = 0, \quad s \in S, i \in A \setminus \{1\} \quad (3.9) \]

\[ b_\tau^s = 0, \quad s \in S \quad (3.10) \]

Observe that constraints 3.1, to 3.10 above are all dependent on the scenario \(s \in S\). We can then separate these constraints one for each scenario. However, for the model to work, additional non-anticipativity constraints need to be included as well. These constraints require that all scenarios with identical pasts up to a certain period must have the same decision policy up to that period. In other words, if two scenarios share an identical past until a certain time period, then we require that the allocation, rebalancing and borrowing decisions for these two scenarios until that time period be the same. Mathematically, we represent these conditions as,

\[ x_{i,t}^s = x_{i,t}^{s'}, \quad t \in T, (s, s') \in S, i \in A \quad (3.11) \]

\[ v_{s,t}^s = v_{s,t}^{s'}, \quad t \in T, (s, s') \in S, i \in A \quad (3.12) \]

\[ p_{i,t}^s = p_{i,t}^{s'}, \quad t \in T, (s, s') \in S, i \in A \quad (3.13) \]

\[ d_{i,t}^s = d_{i,t}^{s'}, \quad t \in T, (s, s') \in S, i \in A \quad (3.14) \]

\[ b_t^s = b_t^{s'}, \quad t \in T, s, s' \in S, \quad (3.15) \]

for scenarios \(s\) and \(s'\) with identical past up to time period \(t\). These constraints above imply that decision variables must be equal to each other for so long as they share a common historical past until some time \(t\) in the investment horizon \(\{1, 2, \ldots, \tau - 1\}\).
3.1.4 Objective Function

A critical component in a financial modeling problem is the choice of the objective function used. The investor’s total wealth at the final time period \( \tau \) under scenario \( s \) is simply equal to,

\[
\omega^s_{\tau} = \sum_i x^i_{s,\tau}
\]  

(3.16)

Thus, the expected final wealth at the last time period \( \tau \) is given by,

\[
\mu = \text{Mean}(\omega_{\tau}) = \sum_s \pi_s \omega^s_{\tau}
\]  

(3.17)

where \( \pi_s \) is the probability that scenario \( s \) occurs. For this thesis, we use an objective function that seeks to maximise the investor’s expected final wealth in the last time period whilst also minimising the risk as represented by the variance of the final wealth variables. A first option is given by the objective function,

\[
\max (1 - \eta) \mu - \eta \text{Var}(\omega_{\tau})
\]  

(3.18)

where \( \text{Var}(\omega_{\tau}) \) is the variance of the final wealth across all scenarios at the final time period \( \tau \). The constant \( \eta \) indicates the investor’s risk aversion preferences. A value close to 1 indicates more sensitivity to risk, while a value close to 0 signifies more risk taking appetite. Moreover, by varying the value of \( \eta \) between its range of 0 and 1, this objective function leads to an efficient frontier of wealth.

However, observe that the objective function requires the calculation of variances, thus making it a non-linear function. This prevents us from using efficient LP solvers that can only be implemented for problems with linear objective functions. Thus, instead of 3.18, we select a linear objective function, introduced by Konno and Yamazaki [34], and Consiglio and Zenios [35], similar to the function introduced above. This is readily done by substituting an absolute term into the variance term of equation 3.18 above as follows,

\[
\max (1 - \eta) \mu - \eta \sum_s \pi_s |\omega^s_{\tau} - \mu|
\]  

(3.19)

Several alternative object functions are also available for use and these viable alternatives are explored below.

The first alternative to the mean-variance objective function is to use the von Neumann-Morgenstern (VM) expected utility of wealth at period \( \tau \). Here, the objective becomes

\[
\max \sum_s \pi_s \text{Utility}(\omega^s_{\tau})
\]  

(3.20)
where $\pi_s$ is still the probability of scenario $s$, $\omega^\tau_s$ is the final wealth in period $\tau$ under scenario $s$, and $\text{Utility}(\omega^\tau_s)$ is the VM utility function introduced by Von Neumann and Morgenstern [36].

Another alternative objective function simply extends the expected utility model above. This takes advantage of the fact that most investors prefer one set of investment strategies over another, even when both strategies yield the same objective value. For example, some investors might prefer constant accumulation of wealth across the entire investment horizon, while some others would prefer an aggressive investment strategy in the beginning, and gradually shift to a more conservative strategy once wealth has already been accumulated.

Hence, this requires us to capture intermediate processes and decisions as well. This is carried out by using a multi objective function rather than the single objective function in 3.20. A general objective function for this problem is,

$$
\max \sum_{k=1}^{K} \omega_k z_k \quad (3.21)
$$

where $\omega_k \geq 0$, $\sum_{k=1}^{K} \omega_k = 1$ and $z_k$ equals the objective function value for the $k$th intermediate objective function.

Other alternatives include Markowitz’s [37] semi variance model, as investors are sometimes more interested in asset declines rather than asset gains. This objective function then is,

$$
\max (1 - \eta)\mu - \eta \sum_s \pi_s (\min[(\omega_s - \mu, 0)]^2 \quad (3.22)
$$

Lastly, yet another option is the downside formula introduced by Lucas and Klasseen [38]. In this case, the investor determines his target wealth level $\omega^*$. Then, risk is measured as the deviance from this target value. The objective function is then given as follows,

$$
\max (1 - \eta)\mu - \eta \sum_s \pi_s (\min[(\omega_s - \omega^*, 0)]^2 \quad (3.23)
$$

### 3.2 Parent-Child Network Model

#### 3.2.1 Model Definition

The parent-child network model is conceptually similar to the non-anticipativity model presented earlier. An investment horizon is divided into $\tau$ discrete time periods represented by $T = \{1, 2, \ldots, \tau\}$. Asset investment categories are also defined as set $A = \{1, 2, \ldots, I\}$ representing broad investment categories such as stocks, bonds, real estate, or cash.
Chapter 3. Stochastic Programming Model Formulation

The notable difference in this model is that the scenario tree is more elegantly presented using nodes. That is, a realisation of an outcome is no longer a scenario but is instead a set of nodes represented by \( N = \{1, 2, \ldots, n\} \). More specifically, a realisation of an outcome is a collection of nodes - one from each time period - such that between time periods, edges connect the nodes. Instead of having decision variables for each scenario, now each node has its own corresponding set of decision variables. This automatically eliminates the need for non-anticipativity constraints and provides a more elegant and less complex problem.

### 3.2.2 Parameters and Variables

In this model, the scenario index for parameters and variables is replaced by an index with the time period and the corresponding node from that time period. For example, the borrowing variable time period 3 node 3, is associated with the third node in the third time period. Thus, for each \( i \in A, t \in T, \text{ and } n \in N \), we define the following parameters and decision variables.

**Parameters:**

\[
\rho^{t',t}_{i,n,n'} = 1 + \rho^{t',t}_{i,n,n'}, \quad \text{where } \rho^{t',t}_{i,n,n'} \text{ is the percent return for asset } i, \text{ during time period } t', \text{ and at node } n'.
\]

\( \pi_n \) Probability that node \( n \) occurs, where \( \sum_{n=1}^{N} \pi_n = 1 \)

\( \omega_0 \) Wealth in the beginning of time period 1.

\( \sigma_{i,t} \) Transaction costs incurred in rebalancing asset \( i \) at the beginning of time period \( t \)

\( \beta^n_t \) Borrowing rate in period \( t \) at node \( n \).

**Decision Variables:**

\( x_{i,t}^n \) Amount of money for asset category \( i \), in time period \( t \), at node \( n \), after rebalancing.

\( v_{i,t}^n \) Amount of money for asset category \( i \), in time period \( t \), at node \( n \), before rebalancing.

\( \omega_t^n \) Wealth at the beginning of time period \( t \), at node \( n \).

\( p_{i,t}^n \) Amount of asset purchased for rebalancing in period \( t \), at node \( n \).

\( d_{i,t}^n \) Amount of asset sold for rebalancing in period \( t \), at node \( n \).

\( b_t^n \) Amount of money borrowed in period \( t \), at node \( n \).
3.2.3 Constraints

The constraints for this model are similar to the constraints for the NA model, where only slight modifications are made. These constraints are as follows.

Since there is only one node at $t = 1$, the wealth at $t = 1$ is given by,

$$\sum_{i \in A} x^1_{i,1} = \omega_0 + b^1_1$$  \hspace{1cm} (3.24)

Likewise, the investor’s total assets at the last time period $t = \tau$ are equal to,

$$\sum_{i \in A} x^n_{i,\tau} = \omega^n, \hspace{1cm} n \in N$$  \hspace{1cm} (3.25)

Similar to the NA model, borrowing limits are also imposed at every time period. The borrow limit constraint is defined as,

$$b^n_t \leq \phi \sum_{i \in A} v^n_{i,t}, \hspace{1cm} t \in T \setminus \{1\}, (n, n_1) \in N, \hspace{1cm} e^{t-1}_{n_1, n_1} = 1$$  \hspace{1cm} (3.26)

and at $t = 1$,

$$b^1_1 \leq \phi \omega_0$$  \hspace{1cm} (3.27)

The accumulation of interest for each asset category at the beginning of the $t^{th}$ period before rebalancing is given by,

$$v^{n_2}_{i,t} = (1 + r^{t-1}_{i,n_1,n_2}) x^{n_1}_{i,t-1}, \hspace{1cm} t \in T \setminus \{1\}, (n_1, n_2) \in N, i \in A \setminus \{1\} \hspace{1cm} e^{t-1}_{n_1, n_1} = 1, \hspace{1cm} (3.28)$$

Rebalancing decisions are similarly carried out at the start of each time period. The rebalancing constraint for each asset category, except for cash, at time period $t$ is defined as,

$$x^n_{i,t} = v^n_{i,t} + p^n_{s,t} - d^n_{s,t}, \hspace{1cm} (n, n_1) \in N, t \in T \setminus \{1\}, i \in A \setminus \{1\} \hspace{1cm} e^{t-1}_{n_1, n_1} = 1$$  \hspace{1cm} (3.29)

Cash flows at each time period are also restricted to be consistent with the purchasing and selling of assets where transaction costs for these activities are also accounted for. The cash rebalancing constraint is therefore,

$$x^n_{1,t} = v^n_{1,t} + \sum_{i \neq 1} d^n_{s,t}(1 - \sigma_{i,t}) - \sum_{i \neq 1} p^n_{s,t}(1 + \sigma_{i,t}) - \sum_{n_1} e^{t-1}_{n_1, n_1} b^n_{l-1}(1 + \beta^n_l) + b^n_t, \hspace{1cm} n \in N, t \neq 1$$  \hspace{1cm} (3.30)

We also assume that no rebalancing transactions take place at time period $t = \tau$. Thus,

$$p^n_{i,\tau} = 0, \hspace{1cm} n \in N, i \in A \setminus \{1\}$$  \hspace{1cm} (3.31)
Chapter 3. Stochastic Programming Model Formulation

\[ d^n_{i,\tau} = 0, \quad n \in N, i \in A \setminus \{1\} \]  
(3.32)

\[ b^n_{\tau} = 0, \quad n \in N \]  
(3.33)

### 3.2.4 Objective Function

For the PC model, we adopt the same objective function used in the NA model. Instead of scenarios, we simply change the indices to nodes. Now recall that this is given by equation 3.19,

\[ \max (1 - \eta)\mu - \eta \sum_{n \in N} \pi_n \omega^n_{\tau} \]  
(3.34)

where \( \mu = \sum_{n \in N} \pi_n \omega^n_{\tau} \). Furthermore, we seek to eliminate the use of the absolute value function in the objective function. We do this simply by introducing two new variables \( \nu^+_n \), \( \nu^-_n \), such that

\[ \nu^+_n - \nu^-_n = \omega^n_{\tau} - \mu, \]  
(3.35)

where \((\nu^+_n, \nu^-_n) \geq 0\).

Hence, if \( \omega^n_{\tau} - \mu \) is negative, \( \nu^-_n \) will take on the positive value of the difference, and \( \nu^+_n \) will be zero. Otherwise if \( \omega^n_{\tau} - \mu \) is positive, \( \nu^+_n \) takes on the value of the difference, and \( \nu^-_n \) will be zero. Rewriting 3.34 and 3.35, we get

\[ \max (1 - \eta)\mu - \eta \sum_{n \in N} \pi_n (\nu^+_n + \nu^-_n) \]  
(3.36)

such that,

\[ \nu^+_n - \nu^-_n = \omega^n_{\tau} - \mu, \]  
(3.37)

\[ (\nu^+_n, \nu^-_n) \geq 0 \]  
(3.38)
Chapter 4

Benders Decomposition

4.1 Motivation

In order to capture the uncertainty associated with the randomness of asset returns, large scenario trees are required. However, a potential tradeoff for a large tree is that the size and complexity of the problem increases as the number of time periods or nodes grow.

As such, modeling realistic investment scenarios requires the use of large trees which computing resources might not be able to handle. An example of this is when the machine runs out of memory space, as the problem has too many variables and constraints for the machine to store and read at once. Thus, in order for these models to be useful in industry, efficient solution algorithms are required to ease the computational burden of the problem by reducing it into something that can be solved readily.

4.2 Introduction and General Concept

One such method is through the use of Benders Decomposition. We seek to implement this solution algorithm on the multi-period parent child model. This is done by breaking up the multi-stage model, which might be a relatively large and complex problem, into separate smaller sub problems classified according to time period. This can be done, as we are merely exploiting the special structure of SP models where constraints and variables are associated with scenarios. An important note to the reader here is that as the objective function requires information on all terminal nodes at once, a further decomposition from time periods to nodes is not possible.

Thus, each sub problem is simply a local version of the multi-period model, where instead of accounting for the decision variables and constraints in all stages, we are only interested
in solving for the decision variables and constraints of a particular time period. This allows for more flexibility as instead of solving a problem with a big scenario tree that might be too computationally intensive, we are left to solve smaller-scaled localised problems. Fig 4.1 shows a four period parent child model decomposed into four sub problems, one each for $t = 1, \ldots, 4$.

Recall that a formulation for the Benders Decomposition method was introduced in chapter 2.4, where the initial problem was split into the master problem and the sub problem. The sub problem, with the fixed complicating variables, generates the Benders cuts for the master problem while the master problem, with the appended cuts, solves for the values of the complicating variables. On the contrary, in our decomposition method, instead of splitting the initial problem into just two smaller sub problems, we split it into several Benders sub problems where, besides the first and last sub problems, each sub problem acts as both a master problem and a sub problem. See figure 4.2 below for an example.

We proceed by first solving the first sub problem at $t = 1$ (SUB 1). This sub problem is always a master problem, as it solves for values of the complicating variables. When this is completed, we continue by iteratively solving the subsequent sub problems, going to and from one sub problem to another. These intermediate sub problems are both master problems and Benders sup problems, as they both generate cuts and receive cuts from other
sub problems. An important note to the reader here is that we do not only necessarily solve each sub problem once. That is, we do not only solve SUB 1, then the sub problem at \( t = 2 \) (SUB 2), one by one until we reach the final sub problem at \( t = \tau \) (SUB TAU) and finish. Rather, the sub problems are solved back and forth, creating new cuts each time. That is, after SUB 2, the next sub problem to be solved can either be at SUB 1 or SUB 3. Figure 4.3 gives a visual idea of this.

![Figure 4.3: Benders Iteration Process](image)

For any intermediate \( t \) sub problem, this sub problem acts as the master problem for the \( t + 1 \) sub problem, as well as the Benders sub problem for the \( t - 1 \) sub problem. Thus, if the \( t \) sub problem generates a good objective value, then a “forward step” is made and the next sub problem at \( t + 1 \) is solved. An important note to the reader here is that a formal definition of a “good” objective value is defined later on, after the LP formulation is introduced. Otherwise, if the objective value is not good, then a ”backward step” is made by appending a cut, and the problem at \( t - 1 \) is again solved. This is a continuous iterative process that starts by solving SUB 1, and then the middle sub problems, until an optimal solution is found in the final sub problem and the algorithm terminates.

### 4.3 Information Pass and Optimality Cuts

In the original model, where decision variables and constraints for all time periods are accounted for at the same time, constraints capture multi-time period processes in a straightforward manner. This includes the interest earning process at the beginning of each time period that requires the amount of capital allocation from the previous time period, as well as the current asset return rates. This process was easily captured by the interest earning constraint in equation 3.28.

However by breaking up the problem into separate time periods, a critical problem arises. Decision variables and constraints for \( t - 1 \) will only be present in the \( t - 1 \) sub problem, and decision variables at \( t \) are only going to be present in the sub problem for \( t \). The
rebalancing constraint for cash, equation 3.30, is similarly affected. Recall that, the cash rebalancing constraint is

This issue is resolved by allowing information to be transmitted from one sub problem to another. Since the investment allocation variables and borrowing variables, $x$ and $b$, are required in the succeeding time period, these values are then passed on to the next sub problem. Thus, if we let $\bar{x}$ and $\bar{b}$ be the values of $x$ and $b$ that are passed on from the previous time period, then we can rewrite equation 3.28 as follows,

$$v_{i,t}^{n_2} = \left(1 + r_{i,n_1,n_2}^{t-1,t}\right) \bar{x}_{i,t-1}^{n_1}$$

(4.1)

and equivalently for equation 3.30,

$$x_{i,t}^{n_1} - v_{i,t}^{n_1} - \sum_{i \in A \setminus \{1\}} d_{i,t}^{n_1} (1 - \sigma_{i,t}) + \sum_{i \in A \setminus \{1\}} p_{i,t}^{n_1} (1 + \sigma_{i,t}) - b_{i,t}^{n_1} = - \sum_{m \in N(t-1)} \bar{b}_{m,t-1}^{t-1} (1 + \beta_{t}^{n})$$

However, if the objective value from the sub problem can still be improved, then instead of a forward pass, the algorithm performs a backward pass where cuts are generated for the previous sub problem at $t - 1$.

These backward passes are formally defined as Benders Optimality Cuts. Observe that the sub problem at time $t$ uses information propagated from the sub problem at $t - 1$. Thus, if the objective value at time $t$ can still be improved, then we require the information propagated from $t - 1$ to be better as well. As such, these Benders Optimality cuts obtained from the sub problem $t$, constrain or bind the solution space for the sub problem at $t - 1$, thereby altering the solution that is passed on to the sub problem at $t$.

These are essentially the dynamics between sub problems. Initially after solving SUB 1, the values of $b_{t}^{m}$ and $x_{1,t}^{n}$, are then passed on to SUB 2. SUB 2 is then solved, and either a forward or backward pass is made. If the optimal solution is valid, then a forward pass is made to time period $t = 3$. However, if the optimal solution is not valid, then a optimality cut is generated and this is back propagated to the sub problem at $t = 1$. This process continues, forward passing variable values and back propagating optimality cuts, until the final sub problem generates the optimal solution and the algorithm terminates.
4.4 Sub Problems

In this section we introduce the sub problem formulations in more detail. Firstly, recall that the original PC model is formulated as follows,

$$\max \left[ (1 - \eta) \mu - \eta \left( \sum_{n \in \mathbb{N}_r} \pi_n |w^n - \mu| \right) \right]$$  \hspace{1cm} (4.2)

where $\mu = \sum_{n \in \mathbb{N}_r} \pi_n w^n_r$, and such that,

$$\sum_{i \in \mathbb{A}} x_{i,1} = w_0 + b_1$$  \hspace{1cm} (4.3)

$$\sum_{i \in \mathbb{A}} x_{i,n} = w_n^n, \quad n \in \mathbb{N}_r$$  \hspace{1cm} (4.4)

$$v_{i,t}^{n_1} = \left( 1 + r_{i,n_1}^{t-1} \right) x_{i,t-1}^{n_1}, \quad t \in \mathbb{P} \setminus \{1\}, (n_1, n_2) \in \mathbb{E}, i \in \mathbb{A} \setminus \{1\}$$  \hspace{1cm} (4.5)

$$x_{i,t}^n = v_{i,t}^n + p_{i,t}^n - d_{i,t}$$  \hspace{1cm} (4.6)

$$x_{i,t}^{n_1} = v_{i,t}^{n_1} + \sum_{i \in \mathbb{A} \setminus \{1\}} d_{i,t}^n (1 - \sigma_{i,t})$$

$$- \sum_{i \in \mathbb{A} \setminus \{1\}} p_{i,t}^n (1 + \sigma_{i,t})$$

$$+ b_t^n - \sum_{m \in \mathbb{N}_{t-1}} b_m^{n-1} (1 + \beta_m^n), \quad t \in \mathbb{P} \setminus \{1\}, n \in \mathbb{N}_t$$  \hspace{1cm} (4.7)

A Benders Decomposition by time periods to this problem is then introduced below.

4.4.1 First Sub Problem

First, the LP formulation for SUB 1 is as follows,

$$\max z_1$$  \hspace{1cm} (4.8)

s.t.

$$\sum_{i \in \mathbb{A}} x_{i,1} = w_0 + b_1$$  \hspace{1cm} (4.9)

$$b_1^1 \leq \phi w_0$$  \hspace{1cm} (4.10)

$$z_1 \leq G_2(x_{11}, b_{11}), \quad n \in \mathbb{N}_2$$  \hspace{1cm} (4.11)

Recall that the idea behind implementing a Benders Decomposition is to break up a multi-stage problem into sub problems consisting of one specific time period with only local
constraints and variables. As such in SUB 1, we are only interested to account for constraints and variables that are present at this stage of the problem. These are given by equations 4.9 and 4.10 respectively.

Observe that equation 4.10 is simply the constraint that captures the initial wealth allocation where the total amount to be invested in all asset classes (LHS) must be equal to the sum of the initial investment capital and any borrowings in this time period (RHS). Also recall that equation 4.9 is the constraint that captures the borrow limit at each time period where the maximum amount that can be borrowed is the initial investment capital multiplied by a constant factor $\phi$.

On the other hand, the objective function, as well as equation 4.11, are introduced as part of the Benders Decomposition algorithm. The objective function $z_1$, is defined to be the upper bound of the wealth at time period 1. This variable is constrained by the Benders Optimality cuts that are generated from SUB 2.

Also, observe that this constraint is a less than constraint, implying that the cut provides an upper bound for the total wealth, $z_1$. This constraint cuts the solution space smaller and smaller, thus providing better solutions after each new cut is generated.

After solving SUB 1, the algorithm always proceeds to solve SUB 2 next. The $x_{1,t}^i$’s and $b_{1,t}$ are then passed forward to the next sub problem, as these values are required to determine interest earnings and cash rebalancing.

Initially, as SUB 2 has not yet been solved, no optimality cuts exists. The objective value is thus unbounded and can go as big as possible. Consequently, as this first LP is unbounded at the first iteration, the LP is not solved, and the $x_{1,t}^i$ and $b_{1,t}$ do not take on any values.

### 4.4.2 Middle Sub Problem

In this section we introduce the LP formulation for sub problems corresponding to all time periods $1 < t < \tau$. The LP in time period $t \in \mathbf{P} \setminus \{1\}$ is as follows,

\[
\max z_t
\]

s.t.

\[
v_{i,t}^n = \left(1 + r_{i,m,n}^{t-1,t}\right)x_{i,t-1}^m, \quad (m,n) \in \mathbf{E}, i \in \mathbf{A}, n \in \mathbf{N}_t \quad (\text{dual: } D_{t,n,i}^1) \tag{4.13}
\]

\[
x_{i,t}^n - v_{i,t}^n - p_{i,t}^n + d_{i,t}^n = 0, \quad i \in \mathbf{A}, n \in \mathbf{N}_t \tag{4.14}
\]

\[
b_t^n \leq \phi \sum_{i \in \mathbf{A}} v_{i,t}^n, \quad n \in \mathbf{N}_t \tag{4.15}
\]
\[ x_{n,t} - v_{n,t} = \sum_{i \in A \setminus \{1\}} d_{n,t} (1 - \sigma_{i,t}) + \sum_{i \in A \setminus \{1\}} p_{n,t} (1 + \sigma_{i,t}) - b_n^t \]
\[ = - \sum_{m \in N_{(t-1)}} \bar{b}^{-1}_m (1 + \beta^m_t), \quad n \in N_t \quad \text{(dual : } D_{t,n}^2) \] (4.16)
\[ z_t \leq G_{t+1}(x_{tn}, b_{tn}), \quad m \in N_{(t+1)} \] (4.17)

If the problem is unbounded, then \( z_t = K \), where \( K \) is an arbitrarily large constant.

Observe that in this sub problem, more constraints and variables are present than in the first sub problem. This is because in these intermediate time periods more processes such as the cash rebalancing or asset rebalancing process need to be captured and explained.

Firstly, equations 4.13 and 4.16 are the interest earning and cash rebalancing constraints, respectively. Both these constraints require the use of information passed on from the previous time period. Equation 4.13 requires information about the asset allocation in time period \( t-1 \), denoted by \( \bar{x}_{m, t-1} \), to be able to compute interest earnings, while equation 4.16 require information about the borrowings, denoted by \( \bar{b}^{-1}_m \), to compute the necessary settlement process. The duals for 4.13 and 4.16 are also stored and will be used to generate the optimality cuts if necessary.

Next, equations 4.14 and 4.15 are constraints also found in the original PC model formulation. Equation 4.14 captures the rebalancing process for all assets except cash, while 4.15 sets an upper bound for the total amount of borrowing allowed at this time period.

Similar to the formulation for SUB 1, the objective function, as well as equation 4.17 are introduced as part of Benders Decomposition Algorithm. Equivalently, \( z_t \) is also defined as the upper bound on the total wealth at time period \( t \). This variable is constrained by the optimality cuts, represented by equation 4.17, that limit the value that \( z_t \) can take.

Recall from section 4.2 that a “pass forward” or “pass backward” is determined by the validity of the objective solution. That is, if the objective solution is valid, then information is passed forward and the next sub problem is solved. Otherwise, an optimality cut is generated and back propagated to the previous sub problem that is then solved.

Hence, after introducing the objective for the middle sub problems, we can then formally define when an objective value is good. The objective solution at time \( t \) is good if, \( z_t > \bar{z}_{t-1} \). That is, if the upper bound for wealth at \( t-1 \) is less than the wealth at time period \( t \), then the objective solution is good. Otherwise, if the upper bound for wealth is smaller at a later timer period, then the objective value can still be improved. Indeed, this should never happen as interest is compounded and wealth should increase after each time period.
Consequently, this implies that the upper bound at \( t - 1 \), \( \bar{z}_{t-1} \), is too large, and needs to be reduced further. This is done by generating optimality cuts that reduce the solution space so that a more reasonable upper bound for wealth is produced at \( t - 1 \).

More formally, if \( z_t < \bar{z}_{(t-1)} \), then we generate an optimality cut for \( t - 1 \) to further reduce its objective value as follows,

\[
\bar{z}_{(t-1)} \leq G_t (x^m_{t-1}, b^m_{t-1}) \quad (4.18)
\]

where,

\[
G_t (x^m_{t-1}, b^m_{t-1}) = \sum_{i \in A} \sum_{m \in N_{t-1}} \sum_{n \in N_t} c^{t-1,t}_{m,n} D^1_{t,n,i} \left( 1 + r^{t-1,t}_{i,m,n} \right) x^t_{i,m} - \sum_{m \in N_{t-1}} \sum_{n \in N_t} c^{t-1,t}_{m,n} D^2_{t,n} b^m_{(t-1)} (1 + \beta^n_t) \quad (4.19)
\]

Observe that the optimality cut is a function of the variables \( x^m_{t-1}, b^m_{t-1} \) from the sub problem \( t - 1 \) instead of \( t \). Indeed, this is to be expected, as this constraint is backward passed to sub problem \( t - 1 \). Moreover, the duals at time \( t \), \( D^1_{t,n,i} \) and \( D^2_{t,n} \), are then attached as coefficients to the decision variables in the constraint.

An important note to the reader here is that at the start of the problem no optimality cuts have yet been generated. As such, the objective functions will have no upper bound and the LP’s will be unbounded. As a consequence, the LP is not solved and no solution is passed on to the next time period. This is why when a problem is unbounded, we let \( z_t = K \), where \( K \) is an arbitrarily large constant.

### 4.4.3 Final Sub Problem

In this section we introduce the LP formulation for the sub problem at \( t = \tau \) as follows,

\[
\max \left( (1 - \eta) \mu - \eta \sum_{n \in N_{\tau}} \pi_n (\nu^+_{n} + \nu^-_{n}) \right) \quad (4.20)
\]

\[

\nu^+_{n} - \nu^-_{n} = \omega^0_n - \mu \quad n \in N_{\tau},
\]

\[
\sum_{i \in A} x^n_{i,\tau} = w^n_{\tau}, \quad n \in N_{\tau} \quad (4.22)
\]

\[
v^n_{i,\tau} = \left( 1 + r^{\tau-1,\tau}_{i,m,n} \right) x^n_{i,\tau-1}, \quad (m,n) \in E, i \in A, n \in N_{\tau} \quad (\text{dual} : D^1_{\tau,n,i}) \quad (4.23)
\]
\[ x_{i,\tau}^n - v_{i,\tau}^n - p_{i,\tau}^n + d_{i,\tau}^n = 0, \quad i \in A, \, n \in N_{\tau} \]  
\[ (4.24) \]

\[ x_{1,\tau}^n - v_{1,\tau}^n - \sum_{i \in A \setminus \{1\}} d_{i,\tau}^n (1 - \sigma_{i,\tau}) + \sum_{i \in A \setminus \{1\}} p_{i,\tau}^n (1 + \sigma_{i,\tau}) - b_{\tau}^n \]
\[ = - \sum_{m \in N_{\tau - 1}} \tilde{b}_{m}^{\tau-1} (1 + \beta_{\tau}^m), \quad n \in N_{\tau} \quad (\text{dual: } D_{\tau,n}^2) \]  
\[ (4.25) \]

\[ p_{i,\tau}^n = 0, \quad i \in A \setminus \{1\}, \, n \in N_{\tau} \]  
\[ (4.26) \]

\[ d_{i,\tau}^n = 0, \quad i \in A \setminus \{1\}, \, n \in N_{\tau} \]  
\[ (4.27) \]

\[ b_{\tau}^n = 0, \quad n \in N_{\tau} \]  
\[ (4.28) \]

Some constraints in this problem are also found in the middle sub problem LP. These constraints include interest earning, asset, and cash rebalancing constraints, denoted by equations 4.23 - 4.25, respectively. However, constraints that capture final time period processes are also additionally included. These include the final wealth constraint, equation 4.22, as well as the no rebalancing constraints, denoted by equations 4.26 - 4.28.

Also observe that the objective function is different to the objective function found in the middle and start sub problems. Here, the objective function is simply the objective function of the original PC Model introduced in Chapter 3, where we seek to maximise the expected returns and minimise the risk or the variance of wealth in the final time period.

Just as in 3.34 - 3.38, we eliminate the absolute value function by introducing two new variables \( \nu_{i,\tau}^+ \) and \( \nu_{i,\tau}^- \) that take on the positive difference between the values. This is captured by equation 4.21.

Furthermore, equations 4.20 and 4.26 - 4.28 are the final wealth constraint and the no rebalancing constraints, respectively. These constraints are relevant only in the final time period as they capture important aspects of the model at this stage. The final wealth constraint, equation 4.20, is used for the formulation of the objective function where the wealth at each terminal node is required. Likewise, the no rebalancing constraints simply prevent any further borrowing or rebalancing of the assets at time \( \tau \), as this is the final stage of the investment horizon.

Equations 4.23 and 4.25 simply capture the interest earning and cash rebalancing processes. As in the middle sub problems, we store the duals of 4.23 and 4.25, as these are later used to generate the Benders optimality cuts, if necessary. It is important to note to the reader here that an alternative to the no purchasing or selling condition in this LP, equation 4.26-27, is to simply set \( x_{i,\tau}^n = \nu_{i,\tau}^n \).
Likewise, just as in the middle sub problems, we generate a Benders optimality cut for the previous time period if the objective value in the final time period is less than the objective value in the previous time period. Recall that, in this case, we wish to further constrain the objective value of the LP for the previous time period. This is carried out by generating an upper bound constraint, consisting of the duals from equations 4.23 and 4.25. That is, if $z_\tau < \tilde{z}_{(\tau-1)}$, we append

$$z_{(\tau-1)} \leq G_\tau \left(x^m_{\tau-1}, b^m_{\tau-1}\right)$$

(4.29)

to LP in time period $(\tau-1)$, where

$$G_\tau \left(x^m_{\tau-1}, b^m_{\tau-1}\right) =$$

$$\sum_{i \in A} \sum_{m \in N_{\tau-1}} \sum_{n \in N_\tau} e^{\tau-1,\tau}_{m,n} \tilde{D}_{\tau,n,i}^1 \left(1 + r_{i,m,n}^{\tau-1,\tau}\right) x^{\tau-1}_{m,i}$$

$$- \sum_{m \in N_{\tau-1}} \sum_{n \in N_\tau} e^{\tau-1,\tau}_{m,n} \tilde{D}_{\tau,n}^2 b^m_{(\tau-1)} (1 + \beta^m_n)$$

(4.30)

### 4.4.4 Infeasibility Problem

Infeasibility occurs when no solution satisfies all the constraints and objective function at once. This happens when no possible capital allocation to the $x^t_{n,i}$’s are feasible due to the total capital being inadequate. In other words, this means the solution space is too constrained or too small to allow a feasible solution to exist inside.

In the start and middle sub problems, this issue is easily resolved, as borrowing capital to twice as much to the original capital available is possible. This allows for more flexibility, and an expanded solution space, such that a feasible solution exists.

However, this is not the case for the final time period LP. While borrowing can be carried over from one time period to the next in the middle sub problems, no borrowing can occur in the final time period as imposed by equation 4.28. Thus, the total capital available can no longer be increased by further borrowings. As a consequence, this often leads to issues as no feasible solutions are available in the final sub problem.

In the event that this arises, we then proceed to solve the Benders Infeasibility Problem for the final time period. This LP seeks to solve the final problem with slack variables for each constraint, and then generates a feasibility cut for the sub problem at $\tau - 1$. The LP formulation is given below.
First, we introduce slack variables $\zeta_1, \zeta_2, \ldots, \zeta_8$, one for each constraint in the final sub problem LP. Then, the LP formulation for the infeasibility problem is,

$$\max - \sum_{n \in N_\tau} (\zeta_{3n}^3 + \zeta_{4n}^4 + \zeta_{7n}^7 + \zeta_{8n}^8) - \left( \sum_{n \in N_\tau} \sum_{i \in A_n} \zeta_{n,i}^1 + \zeta_{n,i}^2 + \zeta_{n,i}^5 + \zeta_{n,i}^6 \right) \quad (4.31)$$

such that,

$$v_{i,n}^n - \zeta_{n,i}^1 = \left( 1 + \tau_{i,m,n}^\tau \right) x_{i,n}^m - \left( 1 + \tau_{i,m,n}^\tau \right) x_{i,n}^{m-1}, \quad (m,n) \in E, i \in A, n \in N_\tau \quad (dual: D_{1,n,i}^\tau) \quad (4.32)$$

$$x_{i,n}^n - v_{i,n}^n - p_{n,i}^n + d_{i,n}^n - \zeta_{n,i}^2 = 0, \quad i \in A, n \in N_\tau \quad (4.33)$$

$$x_{1,n}^n - v_{1,n}^n - \sum_{i \in A \setminus \{1\}} d_{i,n}^n (1 - \sigma_{i,n}) + \sum_{i \in A \setminus \{1\}} p_{n,i}^n (1 + \sigma_{i,n}) - b_n^\tau - \zeta_{n}^3 = 0 \quad (dual: D_{2,n}^\tau) \quad (4.34)$$

$$p_{i,n}^n - \zeta_{n,i}^4 = 0, \quad i \in A \setminus \{1\}, n \in N_\tau \quad (4.35)$$

$$d_{i,n}^n - \zeta_{n,i}^5 = 0, \quad i \in A \setminus \{1\}, n \in N_\tau \quad (4.36)$$

$$b_n^\tau - \zeta_{n}^6 = 0, \quad n \in N_\tau \quad (4.37)$$

$$\sum_{i \in A} x_{i,n}^n - v_{i,n}^n - \zeta_{n}^7 = 0, \quad n \in N \tau \quad (4.38)$$

$$\nu_{n}^\tau - \nu_{n}^- - \omega_{n}^\tau + \mu - \zeta_{n}^8 = 0, \quad n \in N \tau \quad (4.39)$$

This LP is simply Benders Final sub problem with slack variables included in all the constraints. On the other hand, the objective function is updated to now minimise the sum of all the slack variables $\zeta_1, \zeta_2, \ldots, \zeta_8$.

Furthermore, recall that the final sub problem becomes infeasible when no possible values to the decision variables $x_{n,i}$’s can be found. Thus by introducing the slack variables, in essence, we attempt to find the smallest values in excess of the original problem, such that the final sub problem becomes feasible.

A Benders Feasibility Cut is automatically generated for the sub problem at $\tau - 1$, using the value of the duals from equation 4.32 and 4.34 as follows,

$$G_{\tau} \left( x_{\tau-1}^m, b_{\tau-1}^n \right) \geq 0 \quad (4.40)$$
where

\[ G_{\tau}(x_{m,\tau-1}, b_{m,\tau-1}) = \sum_{i \in \mathcal{A}} \sum_{m \in \mathcal{N}_{\tau-1}} \sum_{n \in \mathcal{N}_{\tau}} e_{m,n,\tau-1} D_{i,m,n,i}^{\tau-1} \left( 1 + r_{i,m,n,\tau-1} \right) - \sum_{m \in \mathcal{N}_{\tau-1}} \sum_{n \in \mathcal{N}_{\tau}} e_{m,n,\tau-1} D_{\tau,n,i}^{\tau-1} (1 + \beta_{\tau}) \]  

(4.41)

### 4.5 Xpress-MP Implementation

For this thesis, Benders Decomposition Algorithm is implemented using the Xpress Optimisation Suite. According to the product web page [39]. This is a software package that uses the Xpress-Mosel language, a high level modeling and programming language, to allow users to formulate problems, solve these using solver engines and analyse solutions afterwards. The Benders Decomposition code was created using the Xpress-IVE visual development environment. These programs were then run on the Xpress-Optimizer, a solver engine that is composed of sophisticated, multi-threaded optimisation algorithms for solving LP, MIP, quadratic programming (QP) and other types of problems.

The Benders Decomposition Algorithm is implemented as a nested computer program with programs embedded within one another. This provides an elegant and straightforward implementation of the algorithm. The five major components of the program, subdivided into different levels/tiers, are the master program (MP), the SolveSubProblem routine (SR1), the SolveStartProblem routine (SR2), SolveMiddleProblem routine (SR3), SolveFinalProblem routine (SR4), and SolveInfeasibilityProblem routine (SR5). The sub routines SR2, SR3, and SR4 are simply the LP formulations for the start, middle, and final sub problems of the Benders Decomposition method as discussed in section 4.4.1 - 4.4.3. Likewise, SR5 is simply the Benders infeasibility problem discussed in section 4.4.4. On the other hand,
the MP sets the solve direction of the algorithm based on whether the optimal solution at
the current sub problem is valid. The MP then calls SR1 which determines which of SR2,
SR3 or SR4 to call.

Information is also passed on and returned between the 5 components as can be observed
from the flowchart in Fig 4.4. Firstly, the MP transmits the updated time period to SR1.
This is captured by the variable \textbf{Nextimeperiod} in the flowchart.

With this information, SR1 then determines which of SR2, SR3 or SR4 to call. If either
of SR2 or SR4 are called, no further information are propagated forward. However, if SR3
is called, then SR1 transmits \textbf{Nextimeperiod} again to indicate which of the intermediate
time periods the algorithm is currently on.

Furthermore, information about whether the generation of cuts is also returned to SR1 after
the implementation of SR3 or SR4. If a new optimality or feasibility cuts are generated,
then this is captured by the variable \textbf{CutCount}. This information is then returned from
SR1 to MP so that the time period can be updated again.

\subsection*{4.5.1 Master Program}

The MP is responsible for setting the solve direction of the algorithm. That is, it determines
whether the succeeding sub problem to be solved is the sub problem before or after it. This
is done by checking if the current sub problem generates an optimality cut or not. If so, the
MP reduces the time period by 1, resulting in the previous sub problem to be solved next.
Otherwise, the MP increments the time period by 1, resulting in the forward sub problem
to be solved next.

The MP is also responsible for termination of the algorithm. Two termination conditions
exist. First, if the optimal solution is found, then the MP terminates automatically. Oth-
ernwise, if the run time exceeds the allowed limit, then the MP terminates as well.

The MP flowchart is presented below,

1. While Loop terminates when run time exceeds the allowed time
2. Check whether new cut was generated by SR3 or SR4.
3. \textbf{Nextimeperiod} is updated accordingly
4. SR1 is called, transmitting to it \textbf{Nextimeperiod}
5. Information, the value of NewCuts, is received from SR1
6. If no new cuts are generated while SR4, then solution is optimal
Main Program Code

Figure 4.5: Master Program Flowchart

7. Otherwise, return to step 1.

4.5.2 Sub Routine 1 - SolveSubProblem

SR1 is simply an intermediate routine that determines which of SR2, SR3 or SR4 to call. In order to do this, SR1 receives information about the current time period in the algorithm. If currently at $t = 1$ then SR1 calls SR2, and if currently at $t = \tau$, then SR1 calls on SR4.

Otherwise, for intermediate time periods, SR3 is called. When this happens, SR1 transmits Nextimeperiod to SR3 to indicate which of the intermediate time periods the algorithm is currently on. SR1 also receives information from SR3 and SR4 that indicate whether a cut has been generated or not. Lastly, CutCount is then passed back to the MP. The flowchart for SR1 is presented below.

1. Check value of Nextimeperiod received from MP

2. If Nextimeperiod = 1 , call SR2
Solve Subproblem (Pass, Nextimeperiod)

3. Else if \textbf{Nextimeperiod} = \tau, call SR4

4. Otherwise, call SR3

5. Receive \textbf{CutCount} from SR3 or SR4

6. Return \textbf{CutCount} to MP

\subsection*{4.5.3 Sub Routine 2 - SolveStartProblem}

When SR2 is called by SR1, this routine implements the LP for the first sub problem presented in section 4.4.1. No information is passed into this sub routine as SR1 only solves the start sub problem for SR1. This is carried out first by creating or unhiding constraints, then by solving the LP, and finally by storing the objective solution and the decision variables. Also no information are passed back, as SR1 does not generate any optimality cuts. The flowchart for SR2 is presented below.

SolveStartProblem

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{solve_start_problem_flowchart.png}
\caption{SolveStartProblem Flowchart}
\end{figure}

1. If first run, create constraints and objective function
2. Otherwise, unhide constraints, objective function and optimality cuts
3. Solve LP
4. Store objective value and decision variables $x$’s and $b$
5. Hide constraints and objective function.

4.5.4 Sub Routine 3 - SolveMiddleProblem

When SR3 is called by SR1, this routine simply implements the LP for the middle sub problems presented in section 4.4.2. Nextimeperiod is passed into the routine to indicate which specific intermediate sub problem to solve. SR3 starts by creating or unhiding the equations, then updates the constraints with the updated values of $x$ and $b$ from the previous time period. The LP is solved, the solutions are stored, and if necessary, cuts are generated for the $t-1$ sub problem. Lastly, CutCount, which indicates whether a cut has been generated, is passed back to SR1. The flowchart for SR3 is presented below.

**SolveMiddleProblem (t)**

![Flowchart](Figure 4.8: SolveMiddleProblem Flowchart)
1. If first run, create constraints and objective function

2. Otherwise, unhide constraints, objective function and optimality cuts

3. Update interest earning and cash rebalancing constraints using \( x \)'s and \( b \) from \( t - 1 \) sub problem.

4. Solve LP

5. If LP is unbounded, assign large number as objective value

6. Store objective value and decision variables \( x \)'s and \( b \)

7. Generate optimality cuts for \( t - 1 \) sub problem if necessary

8. Return \textbf{CutCount} to SR1

9. Hide constraints and objective function

### 4.5.5 Sub Routine 4 - SolveFinalProblem

When SR4 is called by SR1, this routine simply implements the LP for the final sub problems presented in section 4.4.3. No information is passed into this sub routine, as SR4 only solves the final sub problem for SR1. SR4 starts by creating or unhiding the equations, then updates the constraints with the updated values of \( x \) and \( b \) from the previous time period, and then solves the LP. If the LP is infeasible, then SR5 is called. Otherwise, the solutions are stored, and if necessary, cuts are generated for the \( \tau - 1 \) sub problem. Lastly, \textbf{CutCount}, which indicates whether a cut has been generated, is passed back to SR1. The flowchart for SR4 is presented below.

1. If first run, create constraints and objective function

2. Otherwise, unhide constraints and objective function

3. Update interest earning and cash rebalancing constraints using \( x \)'s and \( b \) from \( \tau - 1 \) sub problem

4. Solve LP

5. If LP unbounded, call SR5

6. Store objective value and decision variables \( x \)'s and \( b \)

7. Generate optimality cuts for \( \tau - 1 \) sub problem if necessary

8. Return \textbf{CutCount} to SR1

9. Hide constraints and objective function
4.5.6 Sub Routine 5 - SolveInfeasibilityProblem

When SR5 is called by SR1, this routine simply implements the LP for the infeasibility sub problem presented in section 4.4.4. No information is passed into this sub routine as SR5 only solves the infeasibility problem for SR4. SR5 starts, first by creating or un hiding constraints, then by solving the LP. A feasibility cut is then automatically generated for the \( \tau - 1 \) sub problem. Also no information is passed back, as SR5 always creates a feasibility cut. The flowchart for SR5 is presented below.

SolveInFeasibilityProblem

1. If first run, create constraints and objective function
2. Otherwise, unhide constraints and objective function

3. Solve LP

4. Generate Benders Feasibility Cut for $\tau - 1$ sub problem

5. Hide constraints and objective function.
Chapter 5

Heuristics

The Benders Decomposition method introduced in the previous chapter converges slowly if no values are passed on to the next time period. This occurs when the current sub problem is unbounded, and is thus not solved. Consequently, the decision variables $x$’s and $b$’s do not take on any value, and no information is available for a forward pass. Recall that the algorithm starts with solving the first sub problem as follows,

$$\max z_1$$

s.t.

$$\sum_{i \in A} x_{i,1} = w_0 + b_1$$

$$b_1 \leq \phi w_0$$

$$z_1 \leq G_{2n}(x_{11}, b_{11}), \quad n \in N_2$$

When this problem is first solved, observe that no optimality cuts, equation 5.4, are generated yet. Thus, without any constraints on $z_1$, the objective value can then take on as large a value as possible, thereby making the problem unbounded. In similar fashion, recall the middle sub problem formulation,

$$\max z_t$$

s.t.

$$v_{i,t}^n = \left(1 + r_{i,m,n}^{t-1} \right) \bar{x}_{i,t-1}^m, \quad (m, n) \in E, i \in A, n \in N_t \quad (dual: D_{t,n,i}^1)$$

$$x_{i,t}^n - v_{i,t}^n - p_{i,t}^n + d_{i,t}^n = 0, \quad i \in A, n \in N_t$$

$$b_t^n \leq \phi \sum_{i \in A} v_{i,t}^n \quad n \in N_t$$
Chapter 5. **Heuristics**

\[ x_{i,t}^n - v_{i,t}^n - \sum_{i \in A \setminus \{1\}} d_{i,t}^n (1 - \sigma_{i,t}) + \sum_{i \in A \setminus \{1\}} p_{i,t}^n (1 + \sigma_{i,t}) - b_t^n = - \sum_{m \in N_{(t-1)}} \bar{b}_{m}^{t-1} (1 + \beta_{t}^m), \quad n \in N_t \quad (dual: D_{t,n}^2) \quad (5.9) \]

\[ z_t \leq G_{(t+1)m}(x_{tn}, b_{tn}), \quad m \in N_{(t+1)} \quad (5.10) \]

Similarly, when these middle sub problems are first solved, no optimality cuts, equation 5.10 are generated yet. This is because optimality cuts for sub problem \( t \) are passed on from the \( t + 1 \) sub problem. Since the \( t + 1 \) sub problem has not been solved yet, no optimality cuts for sub problem \( t \) exist.

Now, we examine in more detail the implementation of Benders Method. We start by solving the first sub problem, which we know to be unbounded. Consequently, no values are passed on to the next sub problem. Likewise, the next sub problem will also be unbounded, and no values are again passed on to the next problem. This goes on and on, until the final sub problem is reached. Since no values are passed on to this problem, the final sub problem is infeasible and thus requires the infeasibility problem to be solved. Feasibility cuts are then generated back and slowly a feasible solution is generated from scratch.

We wish to expedite this slow start by allocating values to the decision variables \( x \)'s when the sub problems are first solved. As values are now passed to the final sub problem, this provides an immediate feasible solution to the problem instead of having to generate a feasibility cut. Starting the problem with feasibility and optimality cuts also lead to longer solve times as feasible values of \( x \) and \( b \) are often only found after a large number of cuts have been generated. Thus, by immediately assigning feasible values for the decision variables, we also provide a "warm start" to the problem. We carry out this strategy

![Figure 5.1: Initial Decision Variable Values](image)

by introducing heuristics that allocate the available investment capital to different asset categories accordingly. Three such heuristics are presented below.
5.1 Simple Heuristic

5.1.1 Description

First, we introduce the simple heuristic (H1) that assigns values to the $x^t_{n,i}$ decision variables without requiring us to solve an LP. At $t = 1$, H1 equally subdivides the initial investment capital $w_0$ among all the asset categories. Hence, each asset category gets an equal share of the initial investment capital available as follows,

$$x^1_{1,i} = w_0 \text{}/|A|, \quad i \in A \quad (5.11)$$

Now, for time period $t$, H1 simply retains the amount of capital invested in each asset category from the previous period. This is shown below,

$$x^t_{n,i} = v^t_{n,i}, \quad i \in A, n \in N_t \quad (5.12)$$

where $v^t_{n,i}$ equals to the amount invested in asset $i$ after it earns interest for the period $t - 1$ and before rebalancing at time $t$. Now, recall that in the PC model, each asset is rebalanced by adding or subtracting to $v^t_{n,i}$, which is equivalently infusing or reducing capital on that asset. After the rebalancing process is carried out, the end result is $x^t_{n,i}$.

Essentially, by setting $x^t_{n,i} = v^t_{n,i}$ we are imposing that no rebalances are made during time period $t$, and the amount of capital invested in each asset class is simply carried over to the next time period.

5.1.2 Example

In the models used in this paper, the investor has an initial capital of $1$ million to allocate into five different asset classes, namely cash, stocks, long-term and short-term government bonds, and real estate. As such, when the first sub problem is first solved, then

$$x^1_{1,i} = \$1,000,000 \text{}/5 = \$200,000, \quad i \in A \quad (5.13)$$

These values are then passed on into the succeeding time periods accordingly as the Benders Decomposition Algorithm is implemented. In other words, the $x^1_{1,i}$'s are passed on and used in sub problem in $t = 2$ accordingly.

Likewise, if the sub problem for time period $t$ is first solved then the allocation for each asset is simply carried over to the next as follows,

$$x^t_{n,i} = v^t_{n,i} = (1 + r^t_{n,n,i})x^{t-1}_{n,i}, \quad i \in A, n \in N_t \quad (5.14)$$
5.2 Smart Heuristic 1

5.2.1 Description

In the previous section, H1 manually assigns values to $x^t_{n,i}$’s either by allocating an equal share of the initial capital or simply by carrying over the existing capital allocation. Here, we introduce Smart Heuristic 1 (H2) that allocates capital to $x^t_{n,i}$ by solving an LP where solutions to the LP become the values of $x^t_{n,i}$ for that time period.

The LP that we solve is essentially a simplified version of the PC model formulation where a collapsed tree is used instead of the original scenario tree. In the original tree, each node that is not terminal will have several branches, each representing a different set of asset returns. On the other hand, in the collapsed tree, each node will only have one branch and one successor where the single branches represent the average return of all branches that were collapsed.

H2 for sub problem $t$ will only take into account nodes and variables that fall in and after time $t$, as decision variables from previous time periods have already been solved and hence fixed. Consequently, this means that the branches of nodes in time $t$ are collapsed to form a single branch from that node. That is, if there are $n$ nodes at time $t$, then there will be a total of $n$ single-successor trees.

For example, if the sub problem at $t = 2$ is unbounded, H2 then solves $x^t_{n,i}$ for $t \geq 2$, as the $x^1_{n,i}$ are already fixed from solving the previous sub problem. Figures 5.2 and 5.3 below show a collapsed tree for different time periods $t$. It is also important to note that although

![Diagram of collapsed tree for $t=1, t=2$](image)

Figure 5.2: Collapsed tree for $t=1, t=2$

for the sub problem $k$, H2 solves for all $x^t_{n,i}$ for $t \geq k$, only the decision variables at $t = k$, the $x^k_{n,i}$’s, are important. That is, for sub problem at time $k$, we are only interested in assigning values to the decision variables $x^k_{n,i}$ for all nodes $n$. Hence, the values of the $x^k_{n,i}$’s from H2 are then assigned to each asset category at time period $k$ and node $n$. 
5.2.2 LP Formulation

In this section, we introduce the formulation of H2 for the sub problem $t$ as follows.

First we let $\bar{r}_{t,n}^{t_1,i}$ be the average return of asset $i$ at time $t_1$ from the branches of node $n$ at time $t$. That is, the single branches originating from node $n$ of time period $t$ will have the average returns at time $t_1$ and asset $i$ denoted by $\bar{r}_{t_1,i}^{t_n}$. Fig 5.4 below shows a collapsed tree with $\bar{r}_{t_1,i}^{t_n}$ values. Recall that for the sub problem at $t$, H2 finds the optimal allocation of capital at time $t$ such that at $t = \tau$ the returns are maximised while the risk is minimised. Thus, for sub problem $t = 1$, the LP formulation is introduced below as follows,
Chapter 5. Heuristics

\[ \max \left[ (1 - \eta) \mu - \eta \left( \sum_{n \in N_t} \pi_n (\nu_n^+ + \nu_n^-) \right) \right] \]  \hspace{1cm} (5.15)

where

\[ \mu = \sum_{n \in N_t} \pi_n (\omega_n^T) \]  \hspace{1cm} (5.16)

\[ \pi_n = 1 / |N_t| \]  \hspace{1cm} (5.17)

and such that:

\[ \nu_n^+ - \nu_n^- = \omega_n^0 - \mu, \quad n \in N_t \]  \hspace{1cm} (5.18)

\[ \sum_{i \in A} x_{1,i}^t \leq w_0 \]  \hspace{1cm} (5.19)

\[ \bar{x}_{n,i}^{t+1} = (1 + r_{t_{1,i}}^t) (\bar{x}_{n,i}^t), \quad n \in N_t, t_1 \in t \ldots (\tau - 1), i \in A \]  \hspace{1cm} (5.20)

\[ \sum_{i \in A} \bar{x}_{n,i}^t = w_n^0, \quad n \in N_t \]  \hspace{1cm} (5.21)

Equations 5.15-5.17 represent the objective function for the final time period where we wish to maximise returns while minimising risk. Since we use a collapsed tree instead, note that the number of terminal nodes now shrinks to the number of nodes at time \( t \). This is to be expected as collapsing the tree results in the number of terminals nodes to reduce to the number of nodes in \( N_t \) instead of \( N_\tau \). Likewise following the same argument, nodes in equations 5.18, 5.20, and 5.21 are also from the set \( N_t \). For example, if \( t = 1 \), then there will only be one terminal node as the entire tree is collapsed. Equation 5.5 captures this accordingly by summing across \( n \in N_t \).

Equation 5.18 captures the change in variables when we eliminate the absolute value function from the objective function. This is replaced by two variables \( \nu_n^+ \), \( \nu_n^- \), which are both always positive. This is the same formulation we introduce in Section 3.2.4.

Equations 5.19 - 5.21 are just constrains from the PC model. Equation 5.19 captures the initial wealth constraint, 5.20 captures the interest earning constraint while 5.21 captures the final wealth constraint.

Furthermore, the LP at time period \( t \setminus \{1\} \) is then,

\[ \max \left[ (1 - \eta) \mu - \eta \left( \sum_{n \in N_t} \pi_n (\nu_n^+ + \nu_n^-) \right) \right] \]  \hspace{1cm} (5.22)

such that,

\[ \nu_n^+ - \nu_n^- = \omega_n^0 - \mu, \quad n \in N_t \]  \hspace{1cm} (5.23)

\[ \sum_{i \in A} \bar{x}_{n,i}^t = \sum_{i \in A} (1 + r_{n_{1,i},n_{2,i}}^{t-1,t}) (x_{n,i}^{t-1}), \quad n \in N_t, t_1 \in t \ldots (\tau - 1) \]  \hspace{1cm} (5.24)
Chapter 5. Heuristics

\[ x_{n,t}^{l+1} = (1 + r_{l,t}^{n,i})(x_{n,t}^{l}) , \quad n \in N_t, t_1 \in t \ldots (\tau - 1), i \in A \]  \hfill (5.25)

\[ \sum_{i \in A} x_{l,t}^{n} = w_t^{n}, \quad n \in N_t \]  \hfill (5.26)

Equation 5.24 captures the initial wealth constraint instead of equation 5.9 from the LP formulation introduced before. This is again to be expected as, in solving the LP for time \( t \), we are assuming that the \( x_{n,t}^{l} \) of all sub problems before \( t \) are fixed. Hence, these variables are not accounted for in the LP at time \( t \). The initial wealth then, is now equal to the available capital at time \( t \). This is simply the available capital at \( t - 1 \) plus the interest accrued in that time period.

Other than 5.24, equations 5.22 - 5.26 are the same constraints found in the LP formulation for \( t = 1 \).

5.3 Smart Heuristic 2

5.3.1 Description

The third heuristic we introduce, the Smart Heuristic 2 (H3), is intuitively very similar to H2. The only notable difference between H3 and H2 is in the structure of the collapsed tree where instead of collapsing branches starting at time \( t \), branches are collapsed starting at \( t + 1 \). Figures 5.5 and 5.6 below show the new collapsed tree for each time period. Observe that this tree collapses one time period later than in H2. This naturally allows for more branches in the tree and thus captures more variation.

Figure 5.5:Collapsed tree for \( t = 1, t = 2 \)
5.3.2 LP Formulation

In this section, we introduce the formulation of H3 for the sub problem $t$. For sub problem $t = 1$, the LP formulation is introduced below as follows

$$\max \left[ (1 - \eta) \mu - \eta \left( \sum_{n \in N_{t+1}} \pi_n (\nu_n^+ + \nu_n^-) \right) \right]$$

(5.27)

where

$$\mu = \sum_{n \in N_{t+1}} \pi_n (\omega_n^\tau)$$

(5.28)

$$\pi_n = 1 / |N_{t+1}|$$

(5.29)

and such that:

$$\nu_n^+ + \nu_n^- = |\omega_n^\tau - \mu|, \quad n \in N_{t+1}$$

(5.30)

$$\sum_{i \in A} x_{i,1}^1 \leq w_0$$

(5.31)

$$\bar{x}_{n,i}^{t+1} = (1 + \bar{r}_{t+1,n}^{t+1}) (\bar{x}_{n,i}^t), \quad n \in N_{t+1}, t_1 \in t \ldots (\tau - 1), i \in A$$

(5.32)

$$\sum_{i \in A} x_{i,\tau}^n = w_{\tau,n}, \quad n \in N_{t+1}$$

(5.33)

Observe that in equations 5.27 - 5.29, the set of nodes are now from the set $N_{t+1}$ instead of from $N_t$, as for H2. This is because collapsing the tree branches at $t + 1$ instead of at $t$ reduces the number of final nodes to the number of nodes at $N_{t+1}$, instead of at $N_t$. This is also reflected in equations 5.30, 5.32 and 5.33, as the nodes $n$ are also from the set $N_{t+1}$.
Otherwise, 5.27 - 5.33 follow suit from H2 for \( t = 1 \) introduced in 5.2 with only the slight variation in the node sets discussed in the previous paragraph.

Furthermore, at time period \( t \backslash \{1\} \) the LP formulation is,

\[
\max \left[ (1 - \eta) \mu - \eta \left( \sum_{n \in \mathbb{N}_{t+1}} \pi_n (\nu_n^+ + \nu_n^-) \right) \right] \tag{5.34}
\]

such that,

\[
\nu_n^+ - \nu_n^- = \omega_n^{\nu} - \mu, \quad n \in \mathbb{N}_{t+1} \tag{5.35}
\]

\[
\sum_{i \in \mathcal{A}} \bar{x}_{n,i} = \sum_{i \in \mathcal{A}} (1 + r_{i,m,n}^{t-1,t}) x_{m,i}^{t-1}, \quad (n_1, n) \in \mathcal{E}, n_1 \in \mathbb{N}_{t-1}, n \in \mathbb{N}_t \tag{5.36}
\]

\[
\bar{x}_{n,i}^{t_1+1} = (1 + r_{t_1,i}) \bar{x}_{n,i}^{t_1}, \quad n \in \mathbb{N}_{t+1}, t_1 \in t \ldots (\tau - 1), i \in \mathcal{A} \tag{5.37}
\]

\[
\sum_{i \in \mathcal{A}} \bar{x}_{i,\tau} = w_{\tau}, \quad n \in \mathbb{N}_{t+1} \tag{5.38}
\]

In the same regard, in this LP formulation the nodes \( n \) are from the set \( \mathbb{N}_{t+1} \) instead of \( \mathbb{N}_t \).

Similar to the LP formulation in 5.2.2, the initial wealth constraint is replaced as Equation 5.36 replaces Equation 5.31. Otherwise, 5.34 - 5.38 are the same constraints from the LP formulation introduced in 5.2.2 with only the modification in the node set.

### 5.4 Integration into Benders Decomposition Problem

Now we discuss the implementation of H1, H2 and H3. These Heuristics are only implemented during the first run of the Benders Decomposition Algorithm.

#### 5.4.1 Simple Heuristic Integration

We first explore integrating H1 into the Benders Algorithm Code. Recall from 5.1 that for the two cases when \( t = 1 \) and \( t \neq 1 \), H1 has a different allocation strategy. Hence, two separate implementations are explored below - both for the start sub problem, and the middle sub problems.

##### 5.4.1.1 Start Sub Problem

Instead of proceeding straight to the second sub problem without any information being passed on from the first sub problem, we introduce an intermediate step which gives value
to the decision variables $x_{1,i}$. These can then be passed on and used in the second sub problem. Hence, after the first sub problem is found to be unbounded, we let,

$$x_{1,i} = w_0 \|A\|, \quad i \in A \quad (5.39)$$

This intermediate step is implemented in Xpress-MP. After the LP is solved, the program checks whether the LP is unbounded. If so, values are then assigned to the $x_{1,i}$ using the method presented in Section 5.1. The program then proceeds to store the solutions, now with $x_{1,i}$’s having values, and to hide constraints. Otherwise if the problem is bounded, the program skips the implementation of the heuristic and continues immediately to storing the solution. An updated flowchart is presented below,

**SolveStartProblem (Simple Heuristic)**

![Flowchart](image)

**Figure 5.7: Updated Start Sub Problem Flowchart**

### 5.4.1.2 Middle Sub Problem

We also introduce an intermediate step to give value to the decision variables $x_{t,n,i}$, so that these can then be passed on to solve the next sub problem. Hence, after sub problem $t \setminus \{1\}$ is found to be unbounded, we let,

$$x_{t,n,i} = v_{t,n,i}, \quad i \in A, n \in N_t \quad (5.40)$$

Equivalently, this is implemented in Xpress-MP in similar fashion. Note that the program already checks if the LP is unbounded or not. If so, a large value is then assigned to the objective function. Thus, the heuristic is simply implemented as an intermediate step between the if statement and the next step that assigns the objective value to be a very large number. This can be readily seen from the updated flowchart below.
5.4.2 Smart Heuristic Integration

5.4.2.1 Description

H2 and H3 are integrated into Benders Decomposition in the same manner as H1. Now, the Heuristic LP simply replaces H1 in Benders Decomposition Algorithm. That is, after the sub problems are each found to be unbounded during the first go, the LP for H2 or H3 is solved instead of implementing H1. The LP solutions to the $x_{n,t_i}$'s are then used as values for the variables in the sub problem and passed on to the next sub problem.

5.4.2.2 Start Sub Problem

For the first sub problem, this step is implemented in Xpress-MP in similar fashion to the H1 implementation. After the LP is solved, the program checks whether the LP is unbounded or not. If so, the subroutine SolveHeuristic, the LP for H2 or H3, is called. The program then proceeds to store the solution, now with the $x_{1,i}$'s having value, and so
on and so forth. Otherwise if the problem is bounded, the program skip the heuristic and continues immediately to storing the solution. The updated flowchart is thus presented below,

**SolveStartProblem (Smart Heuristic)**

![Updated Start Sub Problem Flowchart](image)

**Figure 5.9: Updated Start Sub Problem Flowchart**

### 5.4.2.3 Middle sub problem

For the middle sub problems, this step is also implemented in Xpress-MP similar to the H1 implementation. Since the program already checks whether the LP is unbounded or not, we simply insert the subroutine SolveHeuristic in between the if statement and the next step that assigns the objective value to be a very large number. An updated flowchart is presented below as well,
Figure 5.10: Updated Middle Sub Problem Flowchart
Chapter 6

Discussion of Results

In this chapter, the data generation method used for this thesis is introduced. The various generated data sets are also described in more detail, as with their properties and characteristics. Furthermore, computational results from implementation of the different models presented in Chapters 3, 4 and 5 are also presented in the succeeding subsections.

6.1 Data Set

When studying asset allocation models similar to the models introduced in the earlier parts of the paper, an important component for these models is to have a good scenario tree. Trees that exhibit, correctness, accuracy and consistency, as discussed in Section 2.5.2, are ideally best as no matter how good a model may be, if the scenario tree is substandard, then the resulting solutions will be substandard as well.

6.1.1 Generation Method

Several alternative scenario generation methods were considered for use in this thesis. These included moment matching methods described by Ross [22], and also stochastic simulation methods developed by Mulvey as the Towers-Perrin Scenario Generator [40] and Hoyland and Wallace [14]. Nonetheless as the primary purpose of this thesis is to examine the formulation and performance of solution algorithms, the bootstrap method, explored by Yu et al. [16], Mitra [23], Ross [22] and Albeanu et al. [41], was used.

Firstly, historical data sets were obtained for each of the five asset classes used in this model. The raw data was then processed by grouping together all asset returns by month. For example, the returns for May 1988, would then consist of the returns for stock, cash, bonds
and real estate categories in that period. These monthly return vectors are then selected at random with repetition to fill the scenario tree one branch at a time. Hence, each branch simply represents the asset returns of some month in the data set. Consequently, a realisation of a scenario would then be composed of a set of these monthly vectors, one for each branch in the path from the parent node to the terminal node.

6.1.2 Information and Properties

The historical returns for each asset category were obtained from financial websites and from the US Treasury website. These included the monthly yields of the U.S. 13 Week Treasury Bill, the U.S. 5 Year and 25 Year Treasury Bond Yields, and the average returns of the S&P 500, and of real estate returns in the United States. A total of 398 average monthly asset returns were collected representing returns from January 1977 to March 2010. Table 6.1 below provides a summary of the return and risk characteristics of each asset category.

<table>
<thead>
<tr>
<th>Asset Category</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cash</td>
<td>1.868%</td>
<td>10.712%</td>
</tr>
<tr>
<td>Stock</td>
<td>0.98%</td>
<td>0.002%</td>
</tr>
<tr>
<td>5YR US T-Bond</td>
<td>0.58%</td>
<td>9.458%</td>
</tr>
<tr>
<td>30YR US T-Bond</td>
<td>0.63%</td>
<td>6.767%</td>
</tr>
<tr>
<td>Real Estate</td>
<td>0.078%</td>
<td>0.0003%</td>
</tr>
</tbody>
</table>

The yield for the US 13 Week T-Bill rate is shown as the average yield for the duration of the entire 13 weeks. Since each of our time units is one month, the yields therefore need be adjusted to monthly returns. This is simply done by dividing the per-quarter yields by three to obtain monthly returns.

Likewise, yields for the US 5 Year and 25 Year Bonds are expressed in terms of yearly yields. Hence, to obtain the monthly yields we simply divide the yearly yields by twelve. Real estate returns are averaged every quarter, and thus have the same values for every three month period. For example, January, February and March all share the same average return. Lastly, stock returns are unchanged as the monthly average returns gathered reflect the average gain or loss during that particular month.

6.1.3 Scenario Trees

Scenario trees with varying sizes and time periods were generated for the purpose of testing the different models introduced in this thesis. It is important to note here that the same
data sets were generated for all models used in this study. That is, the trees used for the PC model and the NA models are exactly the same. This ensures that comparisons made among the models are more accurate and precise. Table 6.2 outlines the tree information for all the trees generated.

Table 6.2: Scenario Trees

<table>
<thead>
<tr>
<th>Number of Scenarios</th>
<th>Time Periods</th>
<th>Number of Branches</th>
<th>Number of Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>3</td>
<td>1-3-3</td>
<td>13</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>1-4-4</td>
<td>21</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>1-4-4-2</td>
<td>53</td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>1-4-4-4</td>
<td>85</td>
</tr>
<tr>
<td>96</td>
<td>4</td>
<td>1-4-4-6</td>
<td>117</td>
</tr>
<tr>
<td>128</td>
<td>4</td>
<td>1-4-4-8</td>
<td>149</td>
</tr>
<tr>
<td>144</td>
<td>4</td>
<td>1-4-6-6</td>
<td>173</td>
</tr>
<tr>
<td>192</td>
<td>4</td>
<td>1-4-6-8</td>
<td>221</td>
</tr>
<tr>
<td>216</td>
<td>4</td>
<td>1-6-6-6</td>
<td>259</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>1-4-8-8</td>
<td>293</td>
</tr>
<tr>
<td>288</td>
<td>4</td>
<td>1-6-6-8</td>
<td>331</td>
</tr>
<tr>
<td>384</td>
<td>4</td>
<td>1-6-8-8</td>
<td>439</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td>1-8-8-8</td>
<td>419</td>
</tr>
<tr>
<td>768</td>
<td>5</td>
<td>1-4-4-6-6</td>
<td>173</td>
</tr>
<tr>
<td>864</td>
<td>5</td>
<td>1-4-6-6-6</td>
<td>173</td>
</tr>
<tr>
<td>1024</td>
<td>5</td>
<td>1-4-4-8-8</td>
<td>173</td>
</tr>
<tr>
<td>1152</td>
<td>5</td>
<td>1-4-6-6-8</td>
<td>173</td>
</tr>
<tr>
<td>1296</td>
<td>5</td>
<td>1-6-6-6-6</td>
<td>173</td>
</tr>
<tr>
<td>2304</td>
<td>5</td>
<td>1-6-6-8-8</td>
<td>173</td>
</tr>
<tr>
<td>3072</td>
<td>5</td>
<td>1-6-8-8-8</td>
<td>173</td>
</tr>
<tr>
<td>4096</td>
<td>5</td>
<td>1-8-8-8-8</td>
<td>173</td>
</tr>
</tbody>
</table>

The first column indicates the number of final nodes or scenarios of each tree. This is the most important characteristic of the trees in general, as larger trees consequently require more computational effort than smaller trees do. These trees get consecutively larger with more nodes at each time period, as with successive time periods added into the tree. The smallest of these would be a three time period, nine scenario tree, while the largest tree generated is a five time period, 4,096 scenario tree. Henceforth, the different scenario trees will simply be identified by the tree size followed by “ST”. For example, the 4096 scenario tree will be referred to as the 4096 ST.

The second column in the table indicates the number of time periods in each tree. A tree with more time periods is more realistic and thus more useful for the investor who wishes to hedge his financial decisions, as a longer investment horizon allows for more flexibility and rebalancing, as well as higher margins of growth where interest earnings get compounded. However, including more time periods also increases the size of the tree exponentially. It is also not difficult to observe that the number of final nodes exponentially increases as the
time periods increase from four to five. As such, trees generated for this thesis have 3, 4 and 5 time periods.

The third column indicates the number of branches from nodes of the tree at each time period. All trees start at one parent node at \( t = 1 \), and branch out from this node. This is why all entries in this column start with 1-. Likewise, the second number indicates the number of nodes branched from the parent node for \( t = 2 \), and the third number indicates the number of nodes branched per node in \( t = 2 \), for \( t = 3 \). For example, consider the 1-4-4 scenario tree. Here, the tree starts with one parent node, branches out to 4 nodes in \( t = 2 \), where each of the 4 nodes in \( t = 2 \), branches out to 4 more nodes in \( t = 3 \). As a result, a total of \( 1 \times 4 \times 4 = 16 \) final nodes are present in the tree.

The fourth and last column indicates the total number of nodes present in the tree. This is simply a sum of the nodes in each time period. For example, a 1-4-6-6 tree would have 1, 4, 24 and 144 nodes for each of \( t = 1, \ldots, 4 \), respectively. And thus, the total number of nodes for this tree is \( 1 + 4 + 24 + 144 = 173 \).

### 6.2 Experimental Results

A total of 8 different models were implemented for this study. These are simply the different models introduced in Chapters 3, 4 and 5 earlier.

The first of these models are the basic Parent-Child (PC) and Non-Anticipativity (NA) models presented in Chapter 3. The next three models are varieties of the Benders Decomposition method produced in Chapter 4. These three varieties are the Benders Algorithm implementation with the Simple (B1), Smart 1 (B2) and Smart 2 (B3) heuristics, respectively. The last three models are simply the implementation of the three heuristics H1, H2 and H3, introduced in Chapter 5.

All the programs were coded in Xpress-IVE, and implemented using the Xpress-MP environment and implemented on the same machine. The specifications of both are listed in tables 6.3 and 6.4 below.

<table>
<thead>
<tr>
<th>Type</th>
<th>Dell Precision PWS490</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel(R) Xeon(R) CPU E5345 @ 2.33 GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>2.33 GHz, 3.25 GB RAM</td>
</tr>
</tbody>
</table>
Chapter 6. Discussion of Results

6.2.1 PC vs NA Model

Tables 6.5 and 6.6 present a summary of the results for the NA Model and the PC model respectively.

### Table 6.4: Xpress-IVE Version

<table>
<thead>
<tr>
<th>Xpress-IVE</th>
<th>1.19.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xpress Mosel</td>
<td>2.4.0</td>
</tr>
<tr>
<td>Xpress Optimizer</td>
<td>19.00.00</td>
</tr>
</tbody>
</table>

### Table 6.5: NA Model

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Load Time</th>
<th>Solve Time</th>
<th>Constraints</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0</td>
<td>0.047</td>
<td>2,727</td>
<td>576</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0.094</td>
<td>8,048</td>
<td>1024</td>
</tr>
<tr>
<td>32</td>
<td>0.015</td>
<td>0.375</td>
<td>33,408</td>
<td>2,720</td>
</tr>
<tr>
<td>64</td>
<td>0.062</td>
<td>1.61</td>
<td>134,016</td>
<td>5,440</td>
</tr>
<tr>
<td>96</td>
<td>0.156</td>
<td>4.047</td>
<td>301,824</td>
<td>8,160</td>
</tr>
<tr>
<td>128</td>
<td>0.265</td>
<td>7.938</td>
<td>536,832</td>
<td>10,880</td>
</tr>
<tr>
<td>144</td>
<td>0.344</td>
<td>10.422</td>
<td>668,736</td>
<td>12,240</td>
</tr>
<tr>
<td>192</td>
<td>0.594</td>
<td>20.125</td>
<td>1,189,248</td>
<td>16,320</td>
</tr>
<tr>
<td>216</td>
<td>0.703</td>
<td>26.11</td>
<td>1,391,904</td>
<td>18,360</td>
</tr>
<tr>
<td>256</td>
<td>1.062</td>
<td>-</td>
<td>2,097,664</td>
<td>21,760</td>
</tr>
</tbody>
</table>

### Table 6.6: PC Model

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Load Time</th>
<th>Solve Time</th>
<th>Constraints</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0</td>
<td>0.078</td>
<td>233</td>
<td>261</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0.063</td>
<td>398</td>
<td>434</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0.031</td>
<td>926</td>
<td>1,090</td>
</tr>
<tr>
<td>64</td>
<td>0.016</td>
<td>0.047</td>
<td>1,630</td>
<td>1,794</td>
</tr>
<tr>
<td>96</td>
<td>0.015</td>
<td>0.11</td>
<td>2,334</td>
<td>2,498</td>
</tr>
<tr>
<td>128</td>
<td>0.016</td>
<td>0.141</td>
<td>3,030</td>
<td>3,202</td>
</tr>
<tr>
<td>144</td>
<td>0.016</td>
<td>0.156</td>
<td>3,478</td>
<td>3,706</td>
</tr>
<tr>
<td>192</td>
<td>0.016</td>
<td>0.25</td>
<td>4,534</td>
<td>4,762</td>
</tr>
<tr>
<td>216</td>
<td>0.031</td>
<td>0.547</td>
<td>5,216</td>
<td>5,556</td>
</tr>
<tr>
<td>256</td>
<td>0.031</td>
<td>0.656</td>
<td>6,030</td>
<td>6,322</td>
</tr>
<tr>
<td>288</td>
<td>0.047</td>
<td>0.953</td>
<td>6,800</td>
<td>7,140</td>
</tr>
<tr>
<td>384</td>
<td>0.078</td>
<td>1.922</td>
<td>9,044</td>
<td>9,480</td>
</tr>
<tr>
<td>512</td>
<td>0.14</td>
<td>4.703</td>
<td>12,058</td>
<td>12,638</td>
</tr>
<tr>
<td>768</td>
<td>0.312</td>
<td>23.813</td>
<td>18,174</td>
<td>19,106</td>
</tr>
<tr>
<td>864</td>
<td>0.407</td>
<td>41.625</td>
<td>20,902</td>
<td>22,282</td>
</tr>
<tr>
<td>1,024</td>
<td>0.531</td>
<td>66.157</td>
<td>24,158</td>
<td>25,346</td>
</tr>
<tr>
<td>1,152</td>
<td>0.641</td>
<td>98.204</td>
<td>27,238</td>
<td>28,618</td>
</tr>
<tr>
<td>1,296</td>
<td>0.875</td>
<td>164.236</td>
<td>31,352</td>
<td>33,420</td>
</tr>
<tr>
<td>2,304</td>
<td>2.594</td>
<td>857.66</td>
<td>54,320</td>
<td>56,964</td>
</tr>
</tbody>
</table>
6.2.1.1 Solve Time

Initial results indicate that the PC model clearly outperforms the NA model. This observation is based on the scenario tree sizes that both models are able to handle. The PC model solves up to the 2304 ST while the NA model only solves up to the 216 ST. This means that the PC model is able to solve a data set ten times larger than the largest data set that the NA model can solve before the machine runs out of memory.

Likewise, observations can be inferred from a per-scenario comparison of the solve times to show that the PC model is superior to the NA model.

Firstly, we see that for the same data set, the PC model solves the problem faster than the NA model. For example, for the 32 ST, the solve time for the NA and PC models are 0.375 and 0.03 seconds, respectively. Here, the PC model solves the problem ten times faster than the NA model does.

Furthermore, an examination of the larger data sets indicate that the same results follow. For example, in the 216 ST, the solve times for the NA and PC models are 26.45 and 0.547 seconds, respectively. Here, the solve time PC model solves the problem 47 times faster. Figure 6.1 shows the increase in solve time for the NA model. Observe that as the data sets gradually get bigger from the 9 ST to the 216 ST, the NA model performs slower and slower while the PC model solve times remain relatively small.

6.2.1.2 Number of Variables and Constraints

Next, we proceed to a comparison of the number of variables and constraints between the two models. Recall that the NA model uses scenarios instead of network nodes as in the PC model. As such, for the same tree size, the NA model will have more variables than the PC model. In the NA model, each scenario will have its own corresponding set of decision variables, while on the other hand, decision variables are shared among scenarios in the PC model.

Moreover, because each scenario has its own decision variables in the NA model, NA constraints are then needed to ensure that scenarios with the same parents will have the same values. This results in significantly more constraints in the NA model that are not necessary in the PC model.

For example, in the 216 ST, the NA model has 1.4 million constraints compared to only 5,000 for the PC model. Furthermore, the NA model has 18,000 variables compared to only 5,000 in the PC model. Figure 6.1 provides a visual summary of the comparative
differences in the variable and constraint counts where the contrast between the 2 models is very easy to see.

Figure 6.1: PC & NA Comparative Results

6.2.1.3 Summary

From the experimental results, we observe that the PC model can handle larger data sets and provide faster solve times than the NA model. This leads us to conclude that based on solve times, the PC model is clearly superior to the NA model.

This is because the PC model has much less complexity as takes advantage of the network structure of scenario trees, as compared to the scenario framework used by the NA model. Whereas the NA model has decision variable representation for each scenario, the PC model simply assigns decision variables to the nodes in the tree. As a natural consequence, multiple scenarios can share one decision variable and non-anticipativity constraints are not required. This reduces the number of constraints and variables significantly, thereby reducing the complexity of the problem. As such, this allows the machine to solve larger problems before running out of memory space.

6.2.2 Benders Algorithm Results

In this sub section, we present the experimental results from the implementation of B1, B2 and B3.
Chapter 6. Discussion of Results

An important note to the reader here is that the three Benders models have a maximum run-time limit of 30 minutes. During run time, the program continually stores the best current solution found. If optimality is not reached after 30 minutes, the best solution found is then assigned to be the objective value.

This provides two ways to evaluate the performance of the models. First, we can compare the solve times of the three models against the solve time of the PC model as well as against each other. Second, we can also compare the objective values in a similar fashion. Fig 6.2 provides a plot of the solve times of B1, B2, B3 and the PC model, while Figure 6.3 provides a plot of the objective values for the 4 models as well.

In Fig 6.2, the blue line tracks the solve time of the PC model while the red, green and purple lines track the solve times for B1, B2, and B3, respectively. On the other hand, in Fig 6.3, the blue line tracks the optimal solution from the PC model, while the red, green and purple lines track the best solution found by the three models respectively. In the two figures, the x-axis corresponds to the different data sets. The index of 1 corresponds to the 9 ST, 2 to the 16 ST, so on and so forth until when the index 19 corresponds to the 2304 ST.

6.2.2.1 Solve Times

![Benders Solve Time Comparison](image)

**Figure 6.2: Benders Solve Time Comparison**

It is not difficult to see that the PC model clearly outperforms B1, B2 and B3 across all data sets. However for the first 2 data sets, 9 ST and 16 ST, the lines might not indicate otherwise. This is merely due to the scaling of the plot, as the y-axis range extends from
0s to 1800s. In fact, the average solve time of the 4 models for these 2 data sets are 0.07s for the PC model, 14.24s for B1, 16.72s for B2, and 14.53s for BS3. We can see that even this early on, the PC model already solves the two data sets on average at least 200 times quicker.

Furthermore, after the first 2 data sets, the solve times of B1, B2 and B3 all increase very significantly. From an average solve time of 25s for the 16 ST, the solve times jump to 1800s, 1800s and 1497s for the three models respectively. B2, tracked by the green line, stays at this level all throughout until 2304 ST. B1, tracked by the red line, also maintains this level all throughout except for a dip to 1689s at the 64 ST. B3, tracked by the purple line, jumps to 1714s, and then converges to 1800s afterwards.

All Benders models converge to 1800s because we have imposed the maximum run time to be as such. This means that the stopping criterion for optimality was not met. Otherwise, if an optimal solution was found, the program would have terminated before the max time of 1800s was reached.

On the other hand, the PC model solve times are significantly shorter than all three models. For example, in the 32 ST, while B1 - B3 all have solve times near to 1800s, the PC solve time was 0.031s. On average, the PC model solves the 4 time period problems in 0.87s and the 5 time period problems in 208.62s. This is significantly shorter than the B1, B2 and B3 solve times, which starting from the 96 ST are all 1800s.

Moreover, while it is clear that the PC model clearly outperforms B1, B2 and B3 in terms of solve time, not much can be observed about the relative performance between the three models as, on most data sets, all three have solve times of 1800s. This leads us to examine the objective values instead, and compare these values against the optimal solution.

### 6.2.2.2 9ST to 128ST

For the first two data sets, all the Benders models reach optimality. This is observed in figure 6.3 where the red, green and purple lines all trace the path of the blue line. All three Benders models are able to reach optimality because the first two data sets are the only three time period trees, and are thus relatively simple and easy to solve.

Furthermore, B3 solves optimality from 16 ST to 128 ST. This is observed in fig 6.3 where the purple line traces the path of the blue line. From table 6.7, the objective values for these data sets are on average 0.22% less than the optimal solution. This is merely a rounding off error as B3 solves optimality for these data sets.

However, B1 and B2 do not perform as well as B3. While B3 was able to solve for optimality, B2 was unable to reach optimality within the allotted 30 minute run-time. This is observed
in fig 6.3, where we see the green line trace a path that is below the blue and the purple lines. From table 6.7, the objective values obtained by B2 for these data sets are on average 2% off the optimal solution.

Similar to B3, B1 was also unable to reach optimality within the allotted 30 minute run-time. The objective values obtained by B1 for these data sets are on average 4.25% off from the optimal values. This 4.25% average deviance can be attributed to the large deviance in the 32-scenario set where there is a 14.7% deviation from the optimal value. Fig 6.3 shows this, where a big dip in the red line at 32 ST indicate this.

Hence, this leads to the conclusion that for the smaller data sets and in terms of objective values, B3 performs best, followed by B2, and then B1. This is to be expected, as both B2 and B3 solve LP’s instead of manually assigning values just as in B1. Moreover, as B3 captures more variation than B2, this would naturally lead to a better solution.

6.2.2.3 144ST to 512ST

<table>
<thead>
<tr>
<th>Data Set Range</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 ST - 128 ST</td>
<td>4.25%</td>
<td>2%</td>
<td>0.22%</td>
</tr>
<tr>
<td>144 ST - 512 ST</td>
<td>13%</td>
<td>12%</td>
<td>12.5%</td>
</tr>
<tr>
<td>768 ST - 1296 ST</td>
<td>32.15%</td>
<td>28.46%</td>
<td>28.27%</td>
</tr>
</tbody>
</table>
In this range, we observe an immediate drop-off for all 3 models. This is observed in fig 6.3 where all red, green and blue lines dip after the 128 ST. From table 6.7, the deviation for B2 and B3 are approximately 12% each while B1 performs worse with a deviance of 16.2%. This can be attributed to the fact that starting from the 144ST, B1, B2 and B3 were unable to obtain new solutions after obtaining a solution from the initialisation of the heuristics. A comparison between the objective values of the Benders models and the heuristics solutions will show that after the 144 ST, the values are identical for both.

Furthermore, after this dip, all 3 models maintain objective values close to this range. This is again observed from fig 6.3, where the increased gap between the blue line and the other lines formed from the dip remains the same until reaching the 512 ST mark. The only exception is a spike of B1 at the 288 ST mark, where B1 has a deviance of only 7% from the optimal solution. Otherwise, the average deviation across these data sets, these are 13% for B1, 12% for B2, and 12.5% for B3, as observed from table 6.7.

Beside the B1 spike at the 288 ST mark, we can clearly observe that in this range, the B2 and B3 lines are always above the B1 line. This indicates that B2 and B3 perform better than B1. Indeed, if we remove the two entries where B1 spikes and recompute the deviation, we see from table 6.7 that from 13% the resulting average deviations becomes 16%, 4% worse than B2 or B3.

Moreover, while B1 is clearly the inferior model, both B2 and B3 are at par with each other. Besides a spike for both models at the 288 ST mark, both the green and the purple lines trace out the same path in this region. This is further supported by the fact that the average deviation of both models are almost equal.

This leads us to conclude that in the 144-512 ST range, B1 is still clearly the inferior model, while both B2 and B3 are comparable.

6.2.2.4 768ST to 2304ST

In this range, we observe a sudden increase in the value of the optimal solution just after the 512 ST mark. This is observed in fig 6.3 where the blue line shoots up from a value of 6.08 at the 512 ST mark to 7.8 at the 768 ST mark. This increase is to be expected, as starting from the 768 ST, all trees have five time periods. And thus, by expanding from four time periods to five, the returns would be higher as interest earnings are compounded one additional time.

However, while the optimal solution is now higher, the objective values obtained from B1, B2 and B3 do not improve. This is observed from fig 6.3 where the red, green and purple lines still fall within the 5 - 5.5 range. This results in the gap between the optimal solution
and the solutions from B1, B2, and B3 to increase significantly. In fact, from the 512 ST to the 768 ST, the average deviance of the three models jump from 13.85% at the 512ST to 33.75% at the 867ST. Table 6.7 shows that the deviances are now 32.15% for B1, 28.46% for B2, and 28.27% for B3.

Moreover, the relative performances of B1, B2, and B3 in this range are similar to the 128-512 ST range. It is not hard to see that B1 is still clearly the inferior model, as evidenced by having the largest deviance among the three models. This can also be seen in fig 6.3 where the path of the red line is below the green and purple lines.

Both B2 and B3 are again at par against each other as the deviances for the two models, 28.46% and 28.27%, are very close. Fig 6.3 shows this where the green and purple lines trace almost the same paths.

This allows us to conclude that in this range, B1 is still the inferior model, while both B2 and B3 are again equally comparable.

6.2.2.5 Summary

From examining the solve times of the models, we are able to see the dominance of the original PC model over B1, B2 and B3. Whereas the Benders models were unable to solve for optimality even after 30 minutes, the PC model was able to solve for optimal solutions with ease and with significantly shorter solve times. This clearly shows the poor convergence characteristics of using Benders Decomposition on the PC model. While we are able to separate the initial problem into several subproblems with ease, we can never know about the quality of the Benders cuts generated.

Ideally, the Benders cuts constrain the feasible solution region into smaller and smaller. However, if the Benders cuts only reduce the region slightly each time a new cut is generated, then many iterations have to be made, and convergence issues arise. This is exactly the case with B1, B2 and B3, as we clearly see convergence issues arise from the long solve times.

Furthermore, we have also observed the overall poor performance of B1 as compared to the other two models. This is to be expected, as B1 merely allocates the capital without any consideration for the risk - return characteristics of the asset categories or of the scenarios in the tree. Thus, the solutions provided by B1 are worse than those of B2 and B3.

We also observed that for the smaller trees, B3 outperforms B2, while for the larger trees, both models seem to have the same performance. This is also to be expected, as B3 accounts for more variance in the heuristic LP solved. Thus, for the smaller trees with less
variation than larger trees, B3 is naturally a better choice. However, as the tree sizes get larger, more daughter nodes branch from each parent node, and trees naturally have more variation. Thus, collapsing the tree later does not create much difference, and both B2 and B3 provide identical solutions.

### 6.2.3 Heuristics Results

In this subsection, we present the experimental results from the implementation of H1, H2 and H3. These heuristics were developed primarily to provide a ‘warm start’ for the Benders Decomposition Algorithm. However, these heuristics can also be utilised to generate a feasible solution. This is carried out by implementing the heuristics one time period at a time, until we reach the final sub problem. The final sub problem is then just solved without using any heuristics to obtain an objective solution. This implementation is equivalent to the initialisation procedure of B1, B2 and B3 in section 5.4.

We then evaluate the performance of the heuristics in two ways. First, we can compare the objective values against the optimal solution provided by the PC model. Second, we can examine the comparative solve times among the three heuristics. Fig 6.4 provides a plot of the objective values while fig 6.5 provides a plot of the solve times. In fig 6.4, the blue line tracks the optimal solution from the PC model, while the red, green and purple lines track H1, H2 and H3 respectively. Fig 6.5, the red, green and purple lines track the solve times of H1, H2 and H3 respectively.
We see that across all data sets, the PC model is significantly better than H1, H2 or H3. This is observed from fig 6.4 where the path of the blue line hovers around the 5.5 level for the first 2 data sets, around the 6.0 level from 32 ST to 512 ST, and around the 7-8 range from 768 - 1296 ST. On the contrary, H1, H2 and H3 all stay in the 5 - 5.5 range across all data sets.

This is further evidenced by the large deviation of all three heuristics solutions. Table 6.8 shows that in three time period problems, the average deviances are 14.88%, 11.71%, and 11.94% for H1, H2 and H3, respectively. However, the optimal solutions get better with more time periods while the heuristics solutions stays stagnant. Consequently, the deviances increase in the 768 - 2304ST to 32.15%, 28.46% and 29.65% for H1, H2 and H3 respectively.

<table>
<thead>
<tr>
<th>Data Set Range</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 ST - 512 ST</td>
<td>14.88%</td>
<td>11.71%</td>
<td>11.94%</td>
</tr>
<tr>
<td>768 ST - 2304 ST</td>
<td>32.15%</td>
<td>28.46%</td>
<td>29.65%</td>
</tr>
</tbody>
</table>

Furthermore, we see that H2 and H3 perform better than H1. This is observed from fig 6.4, where the path of the red line hovers below the purple and green lines. Likewise from table 6.8, we see that the deviation of H1 solutions are on average 3% worse than the other two heuristics in the three time period problems and 4% worse in the four time period problems.

We can also observe that both H2 and is performs slightly better than H1. An inspection of fig 6.4 shows that both lines almost trace out the same path except for several points when the green line is above the purple line. This is further evidenced by referring to table 6.8, where we see that on average, H2 performs slightly better than H3. The difference in deviances are 0.23% and 1.19% for the three and four time period problems, respectively.

Fig 6.5 shows the solve times of all three heuristics. From this, we observe the solve time of H3 is longest, followed by H2 and then H1. This is to be expected as out of H2 and H3, H3 is the more complex problem with more variables and constraints from solving a larger tree. Similarly, both H2 and H3 solve LP’s, and thus these two heuristics solve longer than H1.

Nonetheless, one interesting observation is the exponential increase in the solve times immediately as we move from three time periods to four. This can be attributed to two factors. First, the heuristic is implemented one additional time as the time period increases. Second, this can be attributed, in general, to the greater complexity of four time period trees.
In summary, we have seen that all three heuristics perform poorly compared to the optimal solution. This is specially so with four time period problems. One would expect the objective solution from the heuristics to improve with more time periods as a natural consequence of additional interest earning and compounding. However, we have learned that the heuristics solutions remain in the same range as in three time period problems. Thus, whereas the heuristics were originally designed to tackle large problems, the experimental results from this section indicate otherwise.

**Figure 6.5: Heuristics Solve Time Comparison**
Chapter 7

Conclusion and Further Research

7.1 Conclusion

In this thesis, we have explored two different formulations of the asset allocation problem. The first model is the existing NA model found extensively in literature [16]. The second, the PC model, is a new formulation that we have introduced in this thesis. This model takes advantage of the network structure of scenario trees that allows us to switch from a scenario-based to a node-based framework. As a consequence of this, less variables and constraints are required for the problem, and, in general, one would expect better computational results.

While existing work has focused on one to two time period trees, not much work has been done to test the performance of these models on multi period trees. As such, both the PC and NA models were tested on three, four and five time period problems, where the data sets were generated from historical data using the bootstrap method. The results indicate that the PC model proves far superior to the NA model, as the PC model was able to solve data sets that are ten times larger than the NA model could handle.

However, the size of the problems that both models can handle are still relatively small. The NA can only solve up to 256 scenarios while the PC can only solve up to 2,304 scenarios. These tree sizes are still inadequate, and larger trees are required to accurately model the uncertainty of future outcomes. Thus, this casts doubts as to whether both models can be applied in an industry setting to design an investment strategy for an investor.

Hence, the main focus of research today is on finding efficient solution algorithms to solve large problems. In this thesis, this is carried out by introducing a Benders Decomposition of the PC model. This method was chosen over others because of the general structure of SP models, where the initial problem can be easily decomposed into smaller sub problems.
The idea behind this method is that we can solve smaller sub problems instead of the initial problem. This allows for the machine to accommodate larger problems that would otherwise be too large and complex to solve. This decomposition then was carried out on the PC model by splitting the multi-period problem into local sub problems consisting only of a single time period. The sub problems are then solved iteratively, with Benders cuts appended from one sub problem to another. Finally, when no cuts are generated by the final sub problem, the optimal solution is obtained.

This was implemented as three unique models that use different heuristics to initialise the algorithms. All three algorithms were then tested on the same data sets used to test the PC and NA models. Results from the experiments show extremely poor convergence for all three models, as the solve times for the algorithms are long, even for small data sets. These solve times are also significantly worse than the solve times for the PC model. In some instances, the Benders algorithm has solve times that are 200 times slower than the PC model.

This leads us to conclude that even if we can decompose the initial problem with great ease, we can never be certain about the quality of the Benders cuts generated. In this problem, we have learned that these cuts are very poor, and this decomposition method on this model should not be pursued further.

On the contrary, one advantage that this formulation provides is that larger data sets, which the PC model cannot solve, can be handled. Whereas the PC model solves only up to 2,304 scenarios before the machine ran out of memory, the Benders algorithms can be shown to handle larger problems, albeit with significantly longer solve times. Certainly, if solve time is not an issue, then the Benders algorithm can be useful in finding solutions to large data sets.

In addition to the Benders models, three heuristics were also introduced in this thesis. These heuristics were designed to tackle large data sets, sacrificing solution quality for computational speed and efficiency. The first heuristic performs an equitable allocation to all asset classes, while the other two heuristics solve LP’s with collapsed variant of the initial scenario tree.

Results from the experiments show that the objective value from the three heuristics remain stagnant, even for problems with longer time periods. One would also expect the contrary, as the objective values would naturally grow with additional interest earnings and compounding. As such, these heuristics become less and less useful as bigger data sets are used. And thus, this leads us to conclude that the three heuristics we have designed are unsuitable for use.
7.2 Further Research

Nonetheless, much more work can be done for further research. In this thesis, we have only considered the basic formulation of the asset allocation problem. This model can be further extended to include other real world processes as well. This includes adding constraints to capture liability streams, cross currency transactions, as well as allow for more sophisticated financial instruments such as swaps and options.

Similarly, other objective functions introduced in chapter 3 can be explored, after which the solutions can be compared to the results obtained from this thesis. Likewise, another possible direction is to consider the use of more sophisticated scenario generation methods outlined in chapter 2 to provide better quality trees. Then, the portfolio compositions can be tracked at each decision stage and a better qualitative understanding of the solution can be obtained.

Another appealing option is to further investigate the performance of the Benders Algorithms with longer solve times. This way, a better understanding of the performance of the algorithms can be provided. Lastly, a Benders Decomposition of the NA model can also be investigated. Similar work has been carried out in the past by Mulvey and Berger [42], and a comparison between the decomposition methods for the PC and NA model would be appealing.
Appendix A

Computational Results

Results from the Xpress-MP implementation are presented in this appendix. The first two tables show the summary from the NA and PC models, respectively. The third table presents the objective values from the three Benders algorithms and the three Heuristics, while the fourth table presents the deviation values of these models from the optimal solution. Lastly, the fifth table summarises the solve times for the B1, B2, B3 and H1, H2 and H3.
<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Load Time</th>
<th>Solve Time</th>
<th>Constraints</th>
<th>Variables</th>
<th>Presolved Elements</th>
<th>Presolved Value</th>
<th>Presolved Simplex</th>
<th>Presolved Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0.047</td>
<td>288</td>
<td>7.95663</td>
<td>5.0466</td>
<td>1.458</td>
<td>109</td>
<td>2835000</td>
<td>18360</td>
</tr>
<tr>
<td>16</td>
<td>0.094</td>
<td>168</td>
<td>3.9934</td>
<td>2.9934</td>
<td>1.166</td>
<td>272</td>
<td>4.1932</td>
<td>16320</td>
</tr>
<tr>
<td>32</td>
<td>0.015</td>
<td>60</td>
<td>2.90218</td>
<td>2.90218</td>
<td>9.180</td>
<td>409</td>
<td>33408</td>
<td>614880</td>
</tr>
<tr>
<td>64</td>
<td>0.062</td>
<td>128</td>
<td>1.09169</td>
<td>1.09169</td>
<td>1.180</td>
<td>313</td>
<td>2720</td>
<td>810184</td>
</tr>
<tr>
<td>96</td>
<td>0.156</td>
<td>256</td>
<td>1.59169</td>
<td>1.59169</td>
<td>2.312</td>
<td>73</td>
<td>214</td>
<td>10241</td>
</tr>
<tr>
<td>128</td>
<td>0.265</td>
<td>512</td>
<td>2.93208</td>
<td>2.93208</td>
<td>1.414</td>
<td>214</td>
<td>512</td>
<td>80418</td>
</tr>
<tr>
<td>192</td>
<td>0.594</td>
<td>1024</td>
<td>5.9934</td>
<td>5.9934</td>
<td>2.312</td>
<td>73</td>
<td>214</td>
<td>10241</td>
</tr>
<tr>
<td>216</td>
<td>0.703</td>
<td>2048</td>
<td>7.9934</td>
<td>7.9934</td>
<td>2.312</td>
<td>73</td>
<td>214</td>
<td>10241</td>
</tr>
<tr>
<td>Data Sets</td>
<td>Load Time</td>
<td>Solve Time</td>
<td>Constraints</td>
<td>Variables</td>
<td>Nonzero Elements</td>
<td>Constraints</td>
<td>Presolved Variables</td>
<td>Presolved Nonzero Elements</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-------------------</td>
<td>-------------</td>
<td>---------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0.078</td>
<td>233</td>
<td>261</td>
<td>769</td>
<td>46</td>
<td>84</td>
<td>288</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.063</td>
<td>398</td>
<td>434</td>
<td>1415</td>
<td>73</td>
<td>126</td>
<td>574</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0.031</td>
<td>926</td>
<td>1090</td>
<td>3863</td>
<td>217</td>
<td>414</td>
<td>2062</td>
</tr>
<tr>
<td>4</td>
<td>0.016</td>
<td>0.047</td>
<td>1630</td>
<td>1794</td>
<td>8887</td>
<td>313</td>
<td>542</td>
<td>5486</td>
</tr>
<tr>
<td>5</td>
<td>0.015</td>
<td>0.11</td>
<td>2334</td>
<td>2498</td>
<td>15959</td>
<td>409</td>
<td>670</td>
<td>10958</td>
</tr>
<tr>
<td>6</td>
<td>0.016</td>
<td>0.141</td>
<td>3038</td>
<td>3202</td>
<td>25079</td>
<td>505</td>
<td>798</td>
<td>18478</td>
</tr>
<tr>
<td>7</td>
<td>0.016</td>
<td>0.156</td>
<td>3478</td>
<td>3706</td>
<td>30759</td>
<td>601</td>
<td>974</td>
<td>23278</td>
</tr>
<tr>
<td>8</td>
<td>0.016</td>
<td>0.25</td>
<td>4534</td>
<td>4762</td>
<td>49815</td>
<td>745</td>
<td>1166</td>
<td>39934</td>
</tr>
<tr>
<td>9</td>
<td>0.031</td>
<td>0.547</td>
<td>5216</td>
<td>5556</td>
<td>61687</td>
<td>901</td>
<td>1458</td>
<td>50466</td>
</tr>
<tr>
<td>10</td>
<td>0.031</td>
<td>0.656</td>
<td>6030</td>
<td>6322</td>
<td>82743</td>
<td>985</td>
<td>1534</td>
<td>69582</td>
</tr>
<tr>
<td>11</td>
<td>0.047</td>
<td>0.953</td>
<td>6800</td>
<td>7140</td>
<td>102367</td>
<td>1117</td>
<td>1746</td>
<td>87546</td>
</tr>
<tr>
<td>12</td>
<td>0.078</td>
<td>1.922</td>
<td>9044</td>
<td>9480</td>
<td>173263</td>
<td>1477</td>
<td>2298</td>
<td>153522</td>
</tr>
<tr>
<td>13</td>
<td>0.14</td>
<td>4.703</td>
<td>12058</td>
<td>12638</td>
<td>296551</td>
<td>1954</td>
<td>3047</td>
<td>270196</td>
</tr>
<tr>
<td>14</td>
<td>0.312</td>
<td>23.813</td>
<td>18174</td>
<td>19106</td>
<td>614783</td>
<td>2972</td>
<td>4673</td>
<td>602156</td>
</tr>
<tr>
<td>15</td>
<td>0.407</td>
<td>41.625</td>
<td>20902</td>
<td>22282</td>
<td>806115</td>
<td>3597</td>
<td>5842</td>
<td>761788</td>
</tr>
<tr>
<td>16</td>
<td>0.531</td>
<td>66.157</td>
<td>24158</td>
<td>25346</td>
<td>111759</td>
<td>3913</td>
<td>6126</td>
<td>1064770</td>
</tr>
<tr>
<td>17</td>
<td>0.641</td>
<td>98.204</td>
<td>27238</td>
<td>28618</td>
<td>1404951</td>
<td>4433</td>
<td>6966</td>
<td>1345504</td>
</tr>
<tr>
<td>18</td>
<td>0.875</td>
<td>164.236</td>
<td>31352</td>
<td>33420</td>
<td>1770031</td>
<td>5382</td>
<td>8747</td>
<td>1702520</td>
</tr>
<tr>
<td>19</td>
<td>2.594</td>
<td>857.66</td>
<td>54320</td>
<td>56964</td>
<td>5463487</td>
<td>8683</td>
<td>13632</td>
<td>5344509</td>
</tr>
<tr>
<td>Data Sets</td>
<td>Original PC Model</td>
<td>BI 2</td>
<td>BI 25</td>
<td>BI 3</td>
<td>BI 5</td>
<td>BI 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------</td>
<td>------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7.31592</td>
<td>5.04278</td>
<td>5.04278</td>
<td>5.04278</td>
<td>5.04278</td>
<td>5.04278</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>7.1797</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>7.2961</td>
<td>5.11729</td>
<td>5.11729</td>
<td>5.11729</td>
<td>5.11729</td>
<td>5.11729</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>7.2313</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>7.2963</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>7.2313</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>7.3069</td>
<td>5.04188</td>
<td>5.04188</td>
<td>5.04188</td>
<td>5.04188</td>
<td>5.04188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>7.2534</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>192</td>
<td>7.2476</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>216</td>
<td>7.3185</td>
<td>5.04278</td>
<td>5.04278</td>
<td>5.04278</td>
<td>5.04278</td>
<td>5.04278</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>7.2611</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>288</td>
<td>7.2476</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>359</td>
<td>7.2611</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>409</td>
<td>7.2476</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>467</td>
<td>7.2611</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>7.2313</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td></td>
<td></td>
</tr>
<tr>
<td>769</td>
<td>7.3069</td>
<td>5.04188</td>
<td>5.04188</td>
<td>5.04188</td>
<td>5.04188</td>
<td>5.04188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>864</td>
<td>7.2534</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1023</td>
<td>7.2476</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1152</td>
<td>7.2611</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1296</td>
<td>7.2476</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td>5.01177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2304</td>
<td>7.31592</td>
<td>5.04278</td>
<td>5.04278</td>
<td>5.04278</td>
<td>5.04278</td>
<td>5.04278</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2560</td>
<td>7.2961</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2880</td>
<td>7.2313</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3840</td>
<td>7.2961</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4670</td>
<td>7.2313</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5120</td>
<td>7.2961</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td>5.09168</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6280</td>
<td>7.2313</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td>5.03621</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7680</td>
<td>7.3069</td>
<td>5.04188</td>
<td>5.04188</td>
<td>5.04188</td>
<td>5.04188</td>
<td>5.04188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8640</td>
<td>7.2534</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td>5.01496</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A.3: Benders and Heuristics Objective Values (in Millions)
<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Original PC Model</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>H1</th>
<th>H2</th>
<th>H3</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>5.44545</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.077</td>
<td>0.076</td>
<td>0.0610</td>
</tr>
<tr>
<td>16</td>
<td>5.34705</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0371</td>
<td>0.0658</td>
<td>0.0363</td>
</tr>
<tr>
<td>32</td>
<td>5.91599</td>
<td>0.1467</td>
<td>0.05929</td>
<td>0.0001</td>
<td>0.1467</td>
<td>0.1321</td>
<td>0.1284</td>
</tr>
<tr>
<td>64</td>
<td>6.06835</td>
<td>8.8986</td>
<td>0.0231</td>
<td>1.6478</td>
<td>0.1726</td>
<td>0.1532</td>
<td>0.1480</td>
</tr>
<tr>
<td>96</td>
<td>5.90548</td>
<td>0.005</td>
<td>0.01232</td>
<td>0.0056</td>
<td>0.15037</td>
<td>0.1315</td>
<td>0.1395</td>
</tr>
<tr>
<td>128</td>
<td>6.22476</td>
<td>0.0172</td>
<td>0.0237</td>
<td>0.0074</td>
<td>0.1858</td>
<td>0.1519</td>
<td>0.1589</td>
</tr>
<tr>
<td>144</td>
<td>6.00362</td>
<td>0.1626</td>
<td>0.1264</td>
<td>0.1192</td>
<td>0.1628</td>
<td>0.1264</td>
<td>0.1193</td>
</tr>
<tr>
<td>192</td>
<td>5.99739</td>
<td>0.1040</td>
<td>0.1193</td>
<td>0.1245</td>
<td>0.1619</td>
<td>0.1193</td>
<td>0.1245</td>
</tr>
<tr>
<td>216</td>
<td>5.96653</td>
<td>0.1508</td>
<td>0.1213</td>
<td>0.1170</td>
<td>0.1598</td>
<td>0.1213</td>
<td>0.1171</td>
</tr>
<tr>
<td>256</td>
<td>6.0929</td>
<td>0.01379</td>
<td>0.1332</td>
<td>0.1268</td>
<td>0.1659</td>
<td>0.1339</td>
<td>0.1268</td>
</tr>
<tr>
<td>288</td>
<td>5.9375</td>
<td>0.1475</td>
<td>0.0696</td>
<td>0.1341</td>
<td>0.14754</td>
<td>0.0696</td>
<td>0.1341</td>
</tr>
<tr>
<td>384</td>
<td>6.12011</td>
<td>0.1717</td>
<td>0.1429</td>
<td>0.1370</td>
<td>0.17171</td>
<td>0.1424</td>
<td>0.1370</td>
</tr>
<tr>
<td>512</td>
<td>6.05088</td>
<td>0.1654</td>
<td>0.1279</td>
<td>0.1224</td>
<td>0.1654</td>
<td>0.1275</td>
<td>0.1224</td>
</tr>
<tr>
<td>768</td>
<td>7.80288</td>
<td>0.3454</td>
<td>0.312</td>
<td>0.3518</td>
<td>0.3454</td>
<td>0.3150</td>
<td>0.3518</td>
</tr>
<tr>
<td>864</td>
<td>7.60694</td>
<td>0.3268</td>
<td>0.3068</td>
<td>0.3032</td>
<td>0.3268</td>
<td>0.3068</td>
<td>0.3032</td>
</tr>
<tr>
<td>1024</td>
<td>7.2994</td>
<td>0.3118</td>
<td>0.2741</td>
<td>0.2838</td>
<td>0.3118</td>
<td>0.2740</td>
<td>0.2839</td>
</tr>
<tr>
<td>1152</td>
<td>7.34489</td>
<td>0.3078</td>
<td>0.2667</td>
<td>0.2875</td>
<td>0.3078</td>
<td>0.2667</td>
<td>0.2875</td>
</tr>
<tr>
<td>1296</td>
<td>7.36767</td>
<td>0.3155</td>
<td>0.2606</td>
<td>0.2563</td>
<td>0.3153</td>
<td>0.2606</td>
<td>0.2563</td>
</tr>
</tbody>
</table>
### Table A.5: Benders and Heuristics Solve Times (in Seconds)

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1.3</td>
<td>1.5</td>
<td>1.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>16</td>
<td>2.5</td>
<td>3.0</td>
<td>3.9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>32</td>
<td>11.8</td>
<td>14.7</td>
<td>17.5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>64</td>
<td>22.0</td>
<td>27.8</td>
<td>29.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>96</td>
<td>36.0</td>
<td>38.5</td>
<td>41.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>128</td>
<td>42.2</td>
<td>47.0</td>
<td>49.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>144</td>
<td>58.0</td>
<td>61.5</td>
<td>64.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>192</td>
<td>75.0</td>
<td>78.5</td>
<td>81.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>216</td>
<td>94.0</td>
<td>97.5</td>
<td>100.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>256</td>
<td>141.0</td>
<td>144.5</td>
<td>147.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>288</td>
<td>170.0</td>
<td>173.5</td>
<td>176.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>384</td>
<td>250.0</td>
<td>253.5</td>
<td>256.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>512</td>
<td>350.0</td>
<td>353.5</td>
<td>356.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>768</td>
<td>650.0</td>
<td>653.5</td>
<td>656.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>864</td>
<td>850.0</td>
<td>853.5</td>
<td>856.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1024</td>
<td>1150.0</td>
<td>1153.5</td>
<td>1156.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1152</td>
<td>1350.0</td>
<td>1353.5</td>
<td>1356.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1296</td>
<td>1650.0</td>
<td>1653.5</td>
<td>1656.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2304</td>
<td>3350.0</td>
<td>3353.5</td>
<td>3356.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3072</td>
<td>5150.0</td>
<td>5153.5</td>
<td>5156.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Bibliography


