Title	An analytics-based decomposition approach to large-scale bilevel optimisation
Authors	Fajemisin, Adejuyigbe
Publication date	2018
Original Citation	Fajemisin, A. 2018. An analytics-based decomposition approach to large-scale bilevel optimisation. PhD Thesis, University College Cork.
Type of publication	Doctoral thesis
Rights	© 2018, Adejuyigbe Fajemisin. - http://creativecommons.org/licenses/by-nc-nd/3.0/
Download date	2024-03-29 12:11:26
Item downloaded from	https://hdl.handle.net/10468/6568

An Analytics-Based Decomposition Approach to Large-Scale Bilevel Optimisation

Adejuyigbe Fajemisin
MSc
112220401

Thesis submitted for the degree of
Doctor of Philosophy

NATIONAL UNIVERSITY OF IRELAND, CORK

Faculty of Science

Department of Computer Science

April 2018

Head of Department:	Prof Cormac Sreenan
Supervisors:	Dr Steven Prestwich Dr Laura Climent

Research supported by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289

Contents

List of Figures

List of Tables

I, Adejuyigbe Fajemisin, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

Adejuyigbe Fajemisin

Declaration

This dissertation is submitted to University College Cork, in accordance with the requirements for the degree of Doctor of Philosophy in the Faculty of Science. The research and thesis presented in this dissertation are entirely my own work and have not been submitted to any other university or higher education institution, or for any other academic award in this university. Where use has been made of other people's work, it has been fully acknowledged and referenced. Parts of this work have appeared in the following publications which have been subject to peer review.

This dissertation is borne by the following publications.

1. S. D. Prestwich, A. O. Fajemisin, L. Climent, and B. O'Sullivan: "Solving a hard cutting stock problem by machine learning and optimisation". In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2015.
2. A. O. Fajemisin, L. Climent, S. D. Prestwich and B. O'Sullivan: "An Application of Data Analytics to Large-Scale Bilevel Optimisation", 2018. *(Under Review)*
3. A. O. Fajemisin, L. Climent, S. D. Prestwich: "Combining the Cutting Stock Problem with Dynamic Harvester Routing", 2018. *(Under Review)*

The contents of this dissertation extensively elaborate upon previously published work and mistakes (if any) are corrected. Some sections of the dissertation are unpublished but may appear in future peer reviewed publications.

Dedicated to my parents, Adelumola and Omoyeni Fajemisin, for their enduring support.

Acknowledgements

I would like to thank my supervisors, Dr Steven Prestwich and Dr Laura Climent for their help, guidance, support, patience and dedication throughout this process. I would also like to thank Weipeng Huang and Milan De Cauwer for helping me work through technical challenges. Additionally, I would like to express my gratitude to Caitriona Walsh, Linda O'Sullivan and Eleanor O'Riordan for going above and beyond whenever I needed help with administrative and personal matters. Finally, I would like to thank Professors Barry O'Sullivan and Ken Brown, Dr. Chrys Ngwa, and all the people of the Insight Centre for Data Analytics at University College Cork, for letting me be a part of a very special group of people.

"Because the Sovereign LORD helps me,
I will not be disgraced.
Therefore I have set my face like flint,
and I know I will not be put to shame."
- Isaiah 50:7 (NIV)

Abstract

Bilevel optimisation problems contain several decision makers, each with different objectives and constraints, arranged in a hierarchical structure. One type of bilevel problem is the single-leader, multiple-follower problem, which has been used in applications like toll-setting, resource management, conflict resolution, and many others. This hierarchical structure allows for the reformulation of the forest harvesting problem as a multiple-follower bilevel problem.

In the forest harvesting problem of Chapter 4, trees are cut into different log types, some of which are more valuable than others. Due to the fact that harvesting machines are designed to prioritise the production of these higher-value log types, over-production and waste of the high-value logs, as well as unfulfilled demand for the low-value logs is seen. Additionally, the discrepancy between amounts of log types expected pre-harvest and the actual amounts seen post-harvest leads to the inefficient harvesting of the forest.

Despite the many approaches for solving multiple-follower problems, they are either not applicable in cases in which the follower problems are not traditional optimisation problems, or do not scale up appropriately. An example of this case occurs with the forest harvesting problem, where the follower problems are dynamic programming problems. Another example is the case where the follower problems are black-box functions. In such cases, replacing the follower problems with reformulations or optimality conditions are not applicable. Evolutionary algorithms can be used, but these are computationally-intensive schemes which do not scale up effectively.

For this reason, an analytics-based approach, which is better able to sample the solution space is needed. The thesis defended throughout this dissertation is that an analytics-based decomposition approach can be used to solve large-scale multiple-follower bilevel problems more efficiently than the other approaches. Specifically, the contributions of this thesis are: (i) a new class of multiple-follower bilevel problems is proposed; (ii) a novel analytics-based decomposition approach for solving this class of large-scale bilevel multiple-follower problems is given; (iii) the forest harvesting problem is reformulated as a bilevel optimisation problem to take into the account operation of harvester, and (iv) a reactive harvesting approach is developed to mitigate the effects of the uncertainty in the data .

Chapter 1

Introduction

1.1 Context

Mathematical Optimisation is the science of selecting the optimal solution to a problem from a larger set of possible solutions. Such a problem is known as an optimisation problem and usually contain an objective function, which is usually minimised or maximised, constraints which limit the scope of the solutions, and variables, the values of which determine the value of the objective function. In cases in which the objectives and constraints are linear, the optimisation problem is known as a linear programming (LP) problem. When some or all of the variables are confined to being integers, it is known as an integer linear programme (ILP). Since being made popular in the early 20th century by researchers like George B. Dantzig, Leonid Kantorovich and John von Neumann, linear programming has been used to solve practical problems in areas such as scheduling, planning, cutting, transportation as well as other fields [Van07, Kan60, BJS10].

In practical application however, optimisation problems are rarely straightforward as non-linearities abound. Bilevel optimisation is such a case. In contrast to standard linear programming where there is only one decision maker, bilevel problems contain several decision makers, each with different objectives and constraints. Several approaches have been developed to solve this class of problems, from reformulation and using classical algorithms like the Kth-best algorithm, to evolutionary algorithms. These approaches are sometimes either not applicable, or do not scale up efficiently for large-scale bilevel problems.

An alternative approach is to use data-driven or analytics-based methods. In data-driven optimisation, the goal is to use insights from any available data to improve the quality of the decisions made or the solution approaches taken. Analytics techniques can therefore be used to enhance decision making either by learning useful features from the data, or using data to simplify mathematical models, thus reducing the complexity and/or solution times of the models.

1.2 Bilevel Optimisation and the Forest Harvesting Problem

A bilevel optimisation problem is one in which there are two levels in the optimisation problem. This is achieved by having one optimisation problem nested inside another. Consequently, there is an outer (or leader) problem and an inner (or follower) problem. These problems affect on each other, as the outer problem reacts to the decisions made in the inner problem. This makes the solution of bilevel problems difficult [CMS07]. In certain cases, there may be multiple inner (i.e. multiple-follower) problems. These problems are known as Stackelberg problems after Heinrich Freiherr von Stackelberg who first described the problems in 1934 [Sta34]. These models are generally used when decision makers have a hierarchy to them, and have been used in applications such as toll setting [DBMS06], resource distribution [CKR71], conflict resolution [AA91] and so on. More on bilevel optimisation is given in Chapter 2.

The wood procurement problem [Uus05] is concerned with the harvesting and delivery of wood products from various forest sources to different customers, while minimising the harvesting and distribution costs. Aside from the problem of efficiently bucking the trees, the tree stems need to be delivered in a cost-effective way to the customers which may be sawmills, paper mills, power plants and so on. An illustration of this is given in figure 1.1. The focus of this thesis is the efficient harvesting of the forest.

In the forest harvesting problem, trees are cut into different log types, with each log type having its own monetary value. Figure 1.2 gives a simple illustration of this. Some log types are more valuable than others, consequently resulting in the desire of the forest owners to produce these valuable log types. For example, in the figure, log type A may be sold at a much higher price than log type D, which induces the forest owners to prioritise its production. Modern harvesting machines are designed to produce these high-value logs at the expense of the

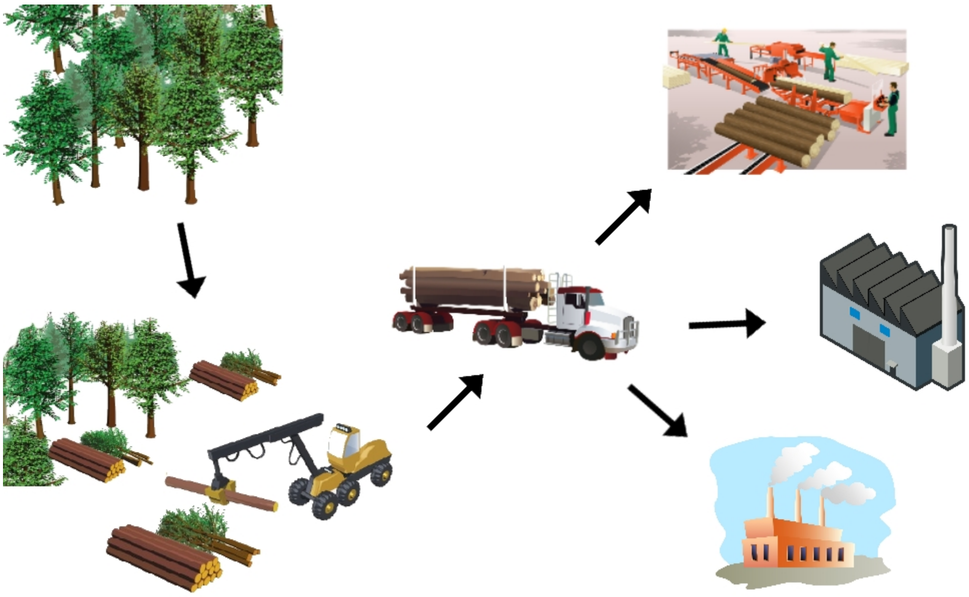


Figure 1.1: An illustration of the wood procurement problem

other log types. This method of operation does not take demand into account, consequently resulting in the over-production and waste of the high-value logs, as well as unfulfilled demand for the low-value logs. Due to the semi-autonomous nature of the harvesting machines, there is only indirect control over their operations. Additionally, their algorithms are complex and non-linear, which makes them not easily manipulated by workers in the field.

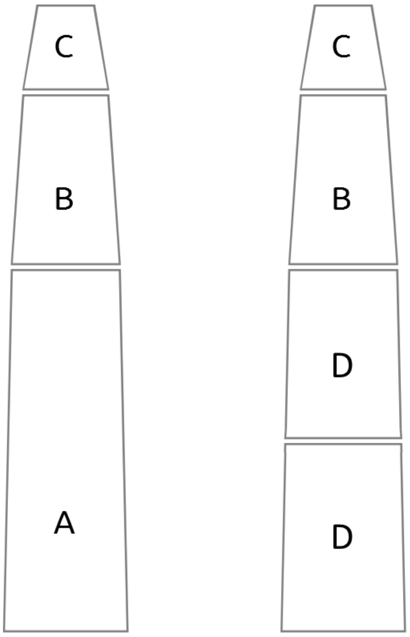


Figure 1.2: An example of different bucking patterns

In order to get an estimate of the capacity of the forest for the different wood products, a Terrestrial Laser Scanning (TLS) system is used. For each tree in its coverage area, the TLS system takes measurements of the diameter at several heights above the ground (figure 1.3). This enables the system to create a digital representation of the forest (see figure 1.4), which allows the harvest planners to estimate the forest capacity.

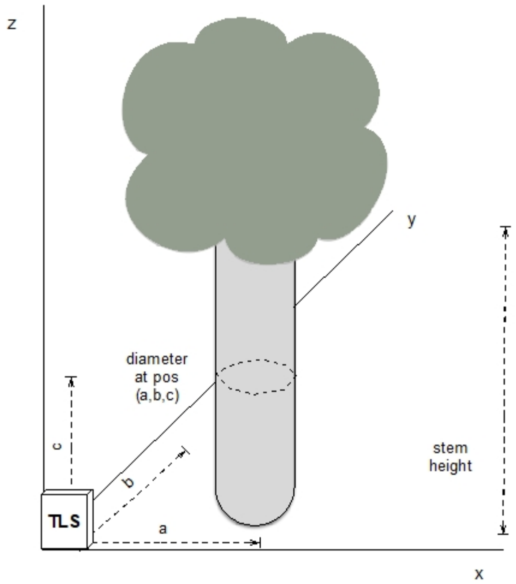


Figure 1.3: TLS measurement

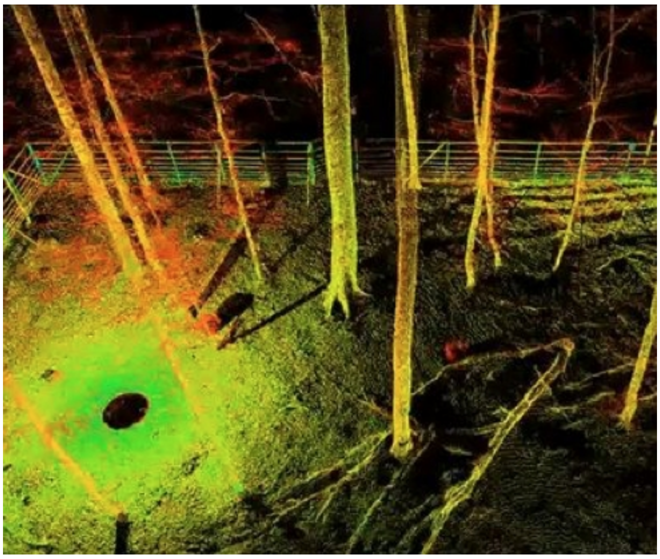


Figure 1.4: A TLS scan

In practice, forests are also subdivided into areas called *blocks*, and one of the goals of this thesis is to select blocks to harvest such that demand for logs is met, while also taking into account the method of operation of the harvesters. Also, it is preferred that blocks with a lower value are selected for harvesting, as this will ensure that the higher quality trees are left intact, thus conserving natural resources. As bilevel optimisation is used in problems in which there are two (or more) decision makers arranged in a hierarchy, this problem can be modelled as a bilevel problem with multiple followers. At the upper level, the goal is to select mostly low-value blocks to harvest in order to fulfil demand. At the lower levels, the goal is to select inputs to the harvester for each block such that the logs produced by them fit the demand. This leads to a single-leader, multiple-follower problem. Due to the size of the problem, with multiple trees per block and multiple blocks per forest, this results in a large-scale bilevel optimisation problem.

1.3 Limitations of Current Approaches

Several methods for solving bilevel problems exist. The most common approach involves reducing the bilevel problem to a single level by replacing the inner problem with its optimality conditions. This reduced problem is then solved using a classical algorithm like the Kth-best algorithm. The limitations of this approach is illustrated in cases where the follower problem is not a standard optimisation problem. If the follower problem is, for example, a black-box function, it cannot be replaced by its optimality conditions, leading to this method being inapplicable. Other methods such as Benders' Decomposition have been used. Here, the problem is first of all reduced to a single-level problem by replacing the follower problem with its Karush-Kuhn-Tucker (KKT) optimality conditions. Then, Benders' Decomposition is applied. In some other cases, the leader problem is treated as the master problem while the follower problem is treated as the sub-problem. This approach also fails in cases where the follower problem is non-standard, as Benders' decomposition is applicable to problems which either have a particular structure, or can be reduced to a single level using the KKT approach. Evolutionary approaches have also been used to solve bilevel problems. These approaches have the advantage of not needing to consider the convexity or linearity of the problem. They are however computationally-intensive approaches, and as such do not scale up well.

For the forest harvesting problem, there are several approaches designed to improve the fit between the demand, and the logs produced by the harvester. These approaches include the use of a measure called the apportionment degree (AD) index [DS12]. Here, logs with higher values are still prioritised, which sometimes results in the demand not being fulfilled. Some approaches have the machines only consider a few cutting patterns, from which one that best fits the demand is selected [DRF15]. This small number of patterns is assumed to be sufficient for all cases, however this is not always so. Another approach varies the prices of the logs in order to produce only the desired logs, but this is done using only a small set of prices [SOG89], thus not covering as many cases as possible.

An additional aspect to the forest harvesting problem is that of reactive harvesting. Forest inventory is taken using sampling, as it is impractical to measure all trees in the forest. These samples are sometimes not representative of the reality, leading to a discrepancy between the amounts expected from a harvest, and the amounts actually obtained. In the forestry industry, the popular algorithm used for this case is the Threshold Accepting (TA) meta-heuristic algorithm [MAA06]. There, demands are recomputed after each stage of the harvesting, however, the blocks selected for the next stage are randomly chosen. This approach fails to find solutions in certain cases, and does not take into consideration the fact that there are considerable costs associated with with moving harvesting equipment from one block to another.

1.4 Dissertation Overview

In summary, while there are several very good methods for solving bilevel problems, they fail for certain cases. Cases where the follower problem is non-standard do not allow for the use of classical methods. Evolutionary approaches can be used in these cases, however they are computationally intensive and do not perform well on large-scale problems. With regards to the forest harvesting problem, the approaches used either do not improve the fit between the demand and the harvesters' outputs, or they only use a small set of cutting patterns. Additionally for reactive harvesting, the algorithm popularly used in industry sometimes fails to find solutions, and also does not take into account location constraints.

The thesis that will be defended in this dissertation is that an analytics-based decomposition approach can be used to solve large-scale multiple-follower bilevel problems in which the follower problems are black-box functions. Although evolutionary approaches can be used in these cases, they can be time-consuming and for large problems, unable to efficiently search the whole solution space. Therefore, an analytics-based approach, which is better able to sample the solution space is needed. An application in the forestry industry is also given. For the forest harvesting problem, this thesis formulates the problem as a bilevel multiple-follower problem. This is a natural fit as it considers the harvester's operation for each block as an optimisation problem (which it is). A reactive harvesting algorithm which handles cases in which the sample data is not representative of the real forest is also presented. Using this bilevel multiple-follower approach allows the problem to be modelled more accurately, and solved using the decomposition approach.

The main contributions of this thesis are enumerated below:

1. A new class of multiple-follower bilevel problems is proposed in Chapter 2. This class is different from the others in that the leader variable are partitioned amongst the followers so that none of the followers share variables with each other. Also, while the other classes only consider linear programs, this new class also allows integer as well as non-linear problems.
2. A novel analytics-based decomposition approach for bilevel problems is presented in Chapter 3. This approach uses analytics techniques such as Monte Carlo sampling and clustering to reduce large-scale bilevel problems to single-level problems.
3. Reformulation of the forest harvesting problem as a bilevel optimisation problem to take into the account operation of harvester in Chapter 4.
4. A reactive harvesting approach which also combines the cutting stock and vehicle routing problems is presented in Chapter 5. It is used to mitigate the effects of the uncertainty in the data by selecting the next best block to harvest based on the yields already obtained, as well as the blocks neighbourhood information.

1.5 Outline

This dissertation consists of 6 chapters structured as follows:

- **Chapter 1, Introduction:** A general overview of the thesis is given in this chapter, including a short background on the problems to be considered and the main contributions and outline of the thesis.
- **Chapter 2, Bilevel Optimisation:** Here, an overview of bilevel optimisation is first given. Multiple-follower problems are also explained, and then a

new class of bilevel problems is proposed. A literature review on the current approaches for solving bilevel problems is also given. Finally, motivation for using an analytics-based approach for solving bilevel problems is stated.

- **Chapter 3, An Analytics-Based Decomposition Approach:** In this chapter, the decomposition technique is explained. Background on distribution learning, generating uniformly distributed random vectors, k-medoids clustering and self organising maps is then given. Two numerical examples of bilevel problems are then solved using the proposed decomposition approach. The first example is a small problem and is used to show that even for small instances, the proposed decomposition approach is competitive. A large-scale example problem is also solved using the decomposition approach, and the solutions obtained are compared with the solutions obtained from using two different genetic algorithms.
- **Chapter 4, A Large-Scale Bilevel Cutting Stock Problem:** Background on the forest harvesting problem is first of all given in this chapter, including some literature on the current approaches. The problem is then reformulated as a multiple-follower bilevel problem, and then both a small-scale and a large scale-evaluation are carried out. Additional data analysis is then done which reveals the existence of natural clusters in the data.
- **Chapter 5, Combining the Cutting Stock Problem with Dynamic Vehicle Routing:** In order to deal with the problem of the sample data sometimes differing from the real data in the forest harvesting problem, a reactive harvesting algorithm is proposed. A review of the literature on reactive harvesting, as well as the use of vehicle routing problem in agriculture is first given here. Two integer programming models are proposed for reactive harvesting - a simple integer linear program, and one in which the cutting stock problem was combined with the vehicle routing problem. Experiments comparing the approach used in the industry with the proposed reactive approach are carried out.
- **Chapter 6, Conclusions:** Finally, the contents and contributions of the thesis are summarised here, and possible avenues for future work are given.

Chapter 2

Bilevel Optimisation

2.1 Overview of Bilevel Optimisation

A bilevel optimisation problem is one in which one optimisation problem is embedded within another. There is a *leader* (or *outer/upper-level*) problem, and a *follower* (or *inner/lower-level*) problem. The variables are split into *leader variables* and *follower variables*. The leader's solution must optimise an objective, under the constraint that the follower optimises a different objective. Though the relationship is asymmetric, the leader and follower influence each other: the follower reacts to the leader's decisions, while the leader makes decisions taking the follower's objective into account. Bilevel optimisation originated in game theory (Stackelberg problems) and mathematical programming. A standard mathematical formulation is:

$$\begin{aligned} \min_{\vec{x},\vec{y}} \quad & F(\vec{x},\vec{y}) \\ \text{s.t.} \quad & \vec{y} \in \operatorname{argmin}_{\vec{y}}\{f(\vec{x},\vec{y}) \mid g(\vec{x},\vec{y}) \leq 0\} \\ & G(\vec{x},\vec{y}) \leq 0 \\ & \vec{x} \in X,\ \vec{y} \in Y \end{aligned} \tag{2.1}$$

where $\vec{x}$ represents the leader decision vector and $\vec{y}$ the follower decision vector. The objective functions at the upper- and lower-level are represented by F and f respectively. Inequality constraints at the upper- and lower-levels are represented by G and g respectively. Equality constraints may also exist. X and Y are the bound constraints for the upper-level decision vector and lower-level decision vector respectively. The condition $\vec{y} \in \operatorname{argmin}_{\vec{y}}$ forces $\vec{y}$ to be an optimal solution to the follower problem, and can be considered as a computationally expensive constraint.

The constraint region of the bilevel problem is:

$$\Omega = \{(\vec{x}, \vec{y}) \in X \times Y : G(\vec{x}, \vec{y}) \leq 0, g(\vec{x}, \vec{y}) \leq 0\} \tag{2.2}$$

For a given leader decision vector $\vec{x}$, the feasible set for the follower is:

$$\Omega(\vec{x}) = \{\vec{y} \in Y : g(\vec{x}, \vec{y}) \leq 0\} \tag{2.3}$$

The set of feasible solutions for the follower problem given $\vec{x}$ is called the rational reaction set and is given as

$$\Psi(\vec{x}) = \{\vec{y} \in Y : \vec{y} \in \text{argmin} f(\vec{x}, \vec{y}) \,|\, \vec{y} \in \Omega(\vec{x})\} \tag{2.4}$$

Finally, the feasible set of the bilevel problem, also known as the inducible region $\mathcal{IR}$, is:

$$\mathcal{IR} = \{(\vec{x}, \vec{y}) : (\vec{x}, \vec{y}) \in \Omega,\, \vec{y} \in \Psi(\vec{x})\} \tag{2.5}$$

Problems with a hierarchical decision structure can be modelled using bilevel optimisation, and applications include toll setting, structural optimization, defence applications, road network planning, optimal chemical equilibria, environmental economics, and water resource management.

Some applications have multiple followers and/or leaders, are multi-objective, or take uncertainty into account. In the case of the toll setting problem for instance, the highway management authority (the leader) must choose a set of roads to impose tolls on. On a lower level, the road users (the followers) seek to choose the shortest distance between their respective origins and destinations. The presence of multiple road users extends the simple bilevel problem with one leader and one follower, to a multiple-follower bilevel problem.

2.2 Multiple-Follower Bilevel Problems

In a *generalised Stackelberg competition model* [SMFD14] there are multiple leaders and followers. Leaders make their decisions independently of each other, then followers make their decisions independently of each other. Leaders are in a Stackelberg competition with their followers: they make their decisions with complete knowledge of how the followers will react, which gives them an advantage.

For the Bilevel Multiple-Follower (BLMF) problem with a single leader and Q multiple followers, let $\vec{x}$ represent the leader decision vector, and $\vec{y}_q$ the decision vector for follower q, $(q = 1 \ldots Q)$. The leader chooses a strategy $\vec{x}$, following which every follower selects its own strategy corresponding to $\vec{x}$. The multiple-follower problem can then be formulated as:

$$\begin{aligned} \min_{\vec{x},\vec{y}_1\ldots\vec{y}_Q} \quad & F(\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \\ \text{s.t.} \quad & \\ & G(\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \leq 0 \\ & \\ & \text{where each } \vec{y}_q \,(q=1,\ldots,Q) \text{ solves} \\ & \\ & \qquad \min_{\vec{y}_q} \; f(\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \\ & \qquad \text{s.t.} \\ & \qquad\qquad g(\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \leq 0 \end{aligned} \tag{2.6}$$

The constraint region of the BLMF is:

$$\begin{aligned} \Omega = \{(\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \in X \times Y_1 \times \ldots \times Y_Q : G(\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \leq 0, \\ g(\vec{x},\vec{y}_q) \leq 0,\, q = 1,\ldots,Q\} \end{aligned} \tag{2.7}$$

The projection of Ω onto the leader's decision space is:

$$\begin{aligned} \Omega(X) = \{\vec{x} \in X : \exists \vec{y}_q \in Y_q : G(\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \leq 0, \\ g(\vec{x},\vec{y}_q) \leq 0,\, q = 1,\ldots,Q\} \end{aligned} \tag{2.8}$$

The feasible set for follower q is affected by a given leader decision vector $\vec{x}$ so that:

$$\Omega_q(\vec{x}) = \{\vec{y}_q \in Y_q : (\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \in \Omega\} \tag{2.9}$$

and allowable choices for each follower are elements of Ω.

Table 2.1: Classes of Bilevel Multiple-Follower Problems Identified in [LSZ06]

Kind of relationship	Decision variables	Objectives	Constraints	Situations (S_i)
Uncooperative	Individual	Individual	Individual	S_1
Cooperative	Sharing	Sharing	Sharing	S_2
			Individual	S_3
		Individual	Sharing	S_4
			Individual	S_5
Partial cooperative	Partial individual & partial sharing	Sharing	Sharing	S_6
			Individual	S_7
		Individual	Sharing	S_8
			Individual	S_9

Each follower's rational reaction set is given as:

$$\Psi_q(\vec{x}) = \{\vec{y}_q \in Y_q : \vec{y}_q \in \text{argmin} f_q(\vec{x}, \vec{y}_q) \,|\, \vec{y}_q \in \Omega_q(\vec{x})\} \tag{2.10}$$

Finally, the inducible region ($\mathcal{IR}$) is:

$$\mathcal{IR} = \{(\vec{x}, \vec{y}_1, \ldots, \vec{y}_q) : (\vec{x}, \vec{y}_1, \ldots, \vec{y}_q) \in \Omega,\, \vec{y}_q \in \Psi_q(\vec{x}),\, q = 1, \ldots, Q\} \tag{2.11}$$

Depending on how much the problem's decision variables, objectives and constraints are shared among the followers, the multiple-follower bilevel problem may be either cooperative, partially cooperative or uncooperative. Based on the type of interaction among the followers, nine classes of linear bilevel multiple-follower problems have been identified in [LSZ06], and are listed in Table 2.1 and given below.

S_1: *Uncooperative Followers with Individual Variables, Objectives and Constraints*

For $x \in X \subset \mathbb{R}^n$, $y_q \in Y_q \subset \mathbb{R}^{m_q}$, $Y = (Y_1, Y_2, \ldots, Y_Q)^t$, $F : X \times Y_1 \times \ldots \times Y_Q \to \mathbb{R}$, $f_q : X \times Y_q \to \mathbb{R}$, and $q = 1, 2, \ldots Q$, S_1 is defined as:

$$\begin{array}{rl} \min_{x \in X} & F(x, y_1, \ldots, y_q) = cx + \sum_{q=1}^{Q} d_q y_q \\ \text{s.t.:} & Ax + \sum_{q=1}^{Q} B_q y_q \leq b \end{array}$$

where each y_q solves the lower level problem:

$$\begin{array}{rl} \min_{y_q} & f_q(x, y_q) = c_q x + e_q y_q \\ \text{s.t.:} & A_q x + C_q y_q \leq b_q \end{array} \tag{2.12}$$

where $c \in \mathbb{R}^n$, $c_q \in \mathbb{R}^n$, $d_q \in \mathbb{R}^{m_q}$, $e_q \in \mathbb{R}^{m_q}$, $b \in \mathbb{R}^p$, $b_q \in \mathbb{R}^{l_q}$, $A \in \mathbb{R}^{p \times n}$, $B_q \in \mathbb{R}^{p \times m_q}$, $A_q \in \mathbb{R}^{l_q \times n}$, $C_q \in \mathbb{R}^{l_q \times m_q}$, $q = 1, 2, \ldots Q$. The superscript t means transposition. With this class of problems, none of the followers share decision variables, objectives or constraints. They are completely insulated from each other.

S_2: *Cooperative Followers with Shared Variables, Objectives and Constraints*

For $x \in X \subset \mathbb{R}^n$, $y_q \in Y_q \subset \mathbb{R}^{m_q}$, $Y = (Y_1, Y_2, \ldots, Y_Q)^t$, $F : X \times Y_1 \times \ldots \times Y_Q \to \mathbb{R}$, $f : X \times Y_q \to \mathbb{R}$, and $q = 1, 2, \ldots Q$, S_2 is defined as:

$$\begin{array}{rl} \min_{x \in X} & F(x, y_1, \ldots, y_q) = cx + \sum_{q=1}^{Q} d_q y_q \\ \text{s.t.:} & Ax + \sum_{q=1}^{Q} B_q y_q \leq b \end{array}$$

where each y_q solves the lower level problem:

$$\begin{array}{rl} \min_{y_1, \ldots, y_q} & f(x, y_1, \ldots, y_q) = c'x + \sum_{q=1}^{Q} e_q y_q \\ \text{s.t.:} & A'x + \sum_{q=1}^{Q} C_q y_q \leq b' \end{array} \tag{2.13}$$

where $c \in \mathbb{R}^n$, $c' \in \mathbb{R}^n$, $d_q \in \mathbb{R}^{m_q}$, $e_q \in \mathbb{R}^{m_q}$, $b \in \mathbb{R}^p$, $b' \in \mathbb{R}^l$, $A \in \mathbb{R}^{p \times n}$, $B_q \in \mathbb{R}^{p \times m_q}$, $A' \in \mathbb{R}^{l \times n}$, $C_q \in \mathbb{R}^{l \times m_q}$, $q = 1, 2, \ldots Q$. Here, all decision variables, objectives and constraints are shared between the followers.

S_3: *Cooperative Followers with Shared Variables and Objectives, and Individual Constraints*

For $x \in X \subset \mathbb{R}^n$, $y_q \in Y_q \subset \mathbb{R}^{m_q}$, $Y = (Y_1, Y_2, \ldots, Y_Q)^t$, $F : X \times Y_1 \times \ldots \times Y_Q \to \mathbb{R}$, $f : X \times Y_q \to \mathbb{R}$, and $q = 1, 2, \ldots Q$, S_3 is defined as:

$$\begin{array}{rl} \min_{x \in X} & F(x, y_1, \ldots, y_q) = cx + \sum_{q=1}^{Q} d_q y_q \\ \text{s.t.:} & Ax + \sum_{q=1}^{Q} B_q y_q \leq b \end{array}$$

where each y_q solves the lower level problem:

$$\begin{array}{rl} \min_{y_1, \ldots, y_q} & f(x, y_1, \ldots, y_q) = c'x + \sum_{q=1}^{Q} e_q y_q \\ \text{s.t.:} & A_q x + \sum_{j=1}^{Q} C_{qj} y_j \leq b_q \end{array} \tag{2.14}$$

where $c \in \mathbb{R}^n$, $c' \in \mathbb{R}^n$, $d_q \in \mathbb{R}^{m_q}$, $e_q \in \mathbb{R}^{m_q}$, $b \in \mathbb{R}^p$, $b_q \in \mathbb{R}^{l_q}$, $A \in \mathbb{R}^{p\times n}$, $B_q \in \mathbb{R}^{p\times m_q}$, $A_q \in \mathbb{R}^{l_q\times n}$, $C_{qj} \in \mathbb{R}^{l_q\times m_q}$, $q = 1, 2, \ldots Q$. Only the decision variables and objectives are shared among the followers in this class. The constraints are individual to the followers.

S_4: *Cooperative Followers with Shared Variables, Individual Objectives, and Shared Constraints*

For $x \in X \subset \mathbb{R}^n$, $y_q \in Y_q \subset \mathbb{R}^{m_q}$, $Y = (Y_1, Y_2, \ldots, Y_Q)^t$, $F : X \times Y_1 \times \ldots \times Y_Q \to \mathbb{R}$, $f_q : X \times Y_q \to \mathbb{R}$, and $q = 1, 2, \ldots Q$, S_4 is defined as:

$$\begin{array}{rl} \min_{x\in X} & F(x, y_1, \ldots, y_q) = cx + \sum_{q=1}^{Q} d_q y_q \\ \text{s.t.:} & Ax + \sum_{q=1}^{Q} B_q y_q \leq b \end{array}$$

where each y_q solves the lower level problem:

$$\begin{array}{rl} \min_{y_1,\ldots,y_q} & f_q(x, y_1, \ldots, y_q) = c_q x + \sum_{j=1}^{Q} e_{qj} y_j \\ \text{s.t.:} & A' x + \sum_{j=1}^{Q} C_j y_j \leq b' \end{array} \tag{2.15}$$

where $c \in \mathbb{R}^n$, $c' \in \mathbb{R}^n$, $d_q \in \mathbb{R}^{m_q}$, $e_{qj} \in \mathbb{R}^{m_j}$, $b \in \mathbb{R}^p$, $b' \in \mathbb{R}^l$, $A \in \mathbb{R}^{p\times n}$, $B_q \in \mathbb{R}^{p\times m_q}$, $A' \in \mathbb{R}^{l\times n}$, $C_q \in \mathbb{R}^{l\times m_q}$, $j, q = 1, 2, \ldots Q$. With this class of problems, while the variables and constraints are shared among the followers, the objectives are not.

S_5: *Cooperative Followers with Shared Variables, and Individual Objectives and Constraints*

For $x \in X \subset \mathbb{R}^n$, $y_q \in Y_q \subset \mathbb{R}^{m_q}$, $Y = (Y_1, Y_2, \ldots, Y_Q)^t$, $F : X \times Y_1 \times \ldots \times Y_Q \to \mathbb{R}$, $f_q : X \times Y_q \to \mathbb{R}$, and $q = 1, 2, \ldots Q$, S_5 is defined as:

$$\begin{array}{rl} \min_{x\in X} & F(x, y_1, \ldots, y_q) = cx + \sum_{q=1}^{Q} d_q y_q \\ \text{s.t.:} & Ax + \sum_{q=1}^{Q} B_q y_q \leq b \end{array}$$

where each y_q solves the lower level problem:

$$\begin{array}{rl} \min_{y_1,\ldots,y_q} & f_q(x, y_1, \ldots, y_q) = c_q x + \sum_{j=1}^{Q} e_{qj} y_j \\ \text{s.t.:} & A_q x + \sum_{j=1}^{Q} C_{qj} y_j \leq b_q \end{array} \tag{2.16}$$

where $c \in \mathbb{R}^n$, $c_q \in \mathbb{R}^n$, $d_q \in \mathbb{R}^{m_q}$, $e_{qj} \in \mathbb{R}^{m_j}$, $b \in \mathbb{R}^p$, $b_q \in \mathbb{R}^{l_q}$, $A \in \mathbb{R}^{p\times n}$, $B_q \in \mathbb{R}^{p\times m_q}$, $A_q \in \mathbb{R}^{l_q\times n}$, $C_{qj} \in \mathbb{R}^{l_q\times m_q}$, $j, q = 1, 2, \ldots Q$. In this case, the followers only have common decision variables. The objectives and constraints are unique to each follower.

S_6: *Partially Cooperative Followers with Partially Shared Variables, and Shared Objectives and Constraints*

For $x \in X \subset \mathbb{R}^n$, $y_q \in Y_q \subset \mathbb{R}^{m_q}$, $Y = (Y_1, Y_2, \ldots, Y_Q)^t$, $z \in Z \subset \mathbb{R}^m$, $F : X \times Y_1 \times \ldots \times Y_Q \times Z \to \mathbb{R}$, $f : X \times Y_1 \times \ldots \times Y_Q \times Z \to \mathbb{R}$, and $q = 1, 2, \ldots Q$, S_6 is defined as:

$$\begin{aligned} \min\nolimits_{x \in X} \quad & F(x, y_1, \ldots, y_q, z) = cx + \textstyle\sum_{q=1}^{Q} d_q y_q + dz \\ \text{s.t.:} \quad & Ax + \textstyle\sum_{q=1}^{Q} B_q y_q + Bz \leq b \end{aligned}$$

where each y_q and z, for each value of x, solves the lower level problem:

$$\begin{aligned} \min\nolimits_{y_q, z} \quad & f(x, y_1, \ldots, y_q, z) = c'x + \textstyle\sum_{j=1}^{Q} e_q y_j + ez \\ \text{s.t.:} \quad & A'x + \textstyle\sum_{q=1}^{Q} C_q y_q + C'z \leq b' \end{aligned} \tag{2.17}$$

where $c \in \mathbb{R}^n$, $c' \in \mathbb{R}^n$, $d_q \in \mathbb{R}^{m_q}$, $d \in \mathbb{R}^m$, $e_q \in \mathbb{R}^{m_j}$, $e \in \mathbb{R}^m$, $b \in \mathbb{R}^p$, $b' \in \mathbb{R}^l$, $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, $B_q \in \mathbb{R}^{p \times m_q}$, $A_q \in \mathbb{R}^{l \times n}$, $C_q \in \mathbb{R}^{l_q \times m_q}$, $C' \in \mathbb{R}^{l \times m}$, $j, q = 1, 2, \ldots Q$. Here the followers share objectives and constraints, but the decision variables are only partially shared.

S_7: *Partially Cooperative Followers with Partially Shared Variables, Shared Objectives and Individual Constraints*

For $x \in X \subset \mathbb{R}^n$, $y_q \in Y_q \subset \mathbb{R}^{m_q}$, $Y = (Y_1, Y_2, \ldots, Y_Q)^t$, $z \in Z \subset \mathbb{R}^m$, $F : X \times Y_1 \times \ldots \times Y_Q \times Z \to \mathbb{R}$, $f : X \times Y_1 \times \ldots \times Y_Q \times Z \to \mathbb{R}$, and $q = 1, 2, \ldots Q$, S_7 is defined as:

$$\begin{aligned} \min\nolimits_{x \in X} \quad & F(x, y_1, \ldots, y_q, z) = cx + \textstyle\sum_{q=1}^{Q} d_q y_q + dz \\ \text{s.t.:} \quad & Ax + \textstyle\sum_{q=1}^{Q} B_q y_q + Bz \leq b \end{aligned}$$

where each y_q and z, for each value of x, solves the lower level problem:

$$\begin{aligned} \min\nolimits_{y_q, z} \quad & f(x, y_1, \ldots, y_q, z) = c'x + \textstyle\sum_{q=1}^{Q} e_q y_y + ez \\ \text{s.t.:} \quad & A_i x + \textstyle\sum_{j=1}^{Q} C_{qj} y_j + C_q z \leq b_q \end{aligned} \tag{2.18}$$

where $c \in \mathbb{R}^n$, $c' \in \mathbb{R}^n$, $d_q \in \mathbb{R}^{m_q}$, $d \in \mathbb{R}^m$, $e_q \in \mathbb{R}^{m_j}$, $e \in \mathbb{R}^m$, $b \in \mathbb{R}^p$, $b_q \in \mathbb{R}^{l_q}$, $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, $B_q \in \mathbb{R}^{p \times m_q}$, $A_q \in \mathbb{R}^{l_q \times n}$, $C_{qj} \in \mathbb{R}^{l_q \times m_j}$, $C_q \in \mathbb{R}^{l_q \times m}$, $j, q = 1, 2, \ldots Q$. The variables here are partially shared, while the objectives are completely shared among the followers. Only the constraints are individual.

S_8: *Partially Cooperative Followers with Partially Shared Variables, Individual Objectives and Shared Constraints*

For $x \in X \subset \mathbb{R}^n$, $y_q \in Y_q \subset \mathbb{R}^{m_q}$, $Y = (Y_1, Y_2, \dots, Y_Q)^t$, $z \in Z \subset \mathbb{R}^m$, $F : X \times Y_1 \times \ldots \times Y_Q \times Z \to \mathbb{R}$, $f_q : X \times Y_1 \times \ldots \times Y_Q \times Z \to \mathbb{R}$, and $q = 1, 2, \ldots Q$, S_8 is defined as:

$$\begin{array}{rl} \min_{x \in X} & F(x, y_1, \dots, y_q, z) = cx + \sum_{q=1}^{Q} d_q y_q + dz \\ \text{s.t.:} & Ax + \sum_{q=1}^{Q} B_q y_q + Bz \leq b \end{array}$$

where each y_q and z, for each value of x, solves the lower level problem:

$$\begin{array}{rl} \min_{y_q, z} & f_q(x, y_1, \dots, y_q, z) = c_q x + \sum_{j=1}^{Q} e_{qj} y_j + e_q z \\ \text{s.t.:} & A'x + \sum_{q=1}^{Q} C_q y_q \leq b' \end{array} \tag{2.19}$$

where $c \in \mathbb{R}^n$, $c' \in \mathbb{R}^n$, $d_q \in \mathbb{R}^{m_q}$, $d \in \mathbb{R}^m$, $e_{qj} \in \mathbb{R}^{m_j}$, $e \in \mathbb{R}^m$, $b \in \mathbb{R}^p$, $b' \in \mathbb{R}^l$, $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, $B_q \in \mathbb{R}^{p \times m_q}$, $A' \in \mathbb{R}^{l \times n}$, $C_q \in \mathbb{R}^{l \times m_q}$, $C' \in \mathbb{R}^{l \times m}$, $j, q = 1, 2, \ldots Q$. Here, only the constraints are fully shared among the followers. The variables are partially shared while the objective functions are individual.

S_9: *Partially Cooperative Followers with Partially Shared Variables, and Individual Objectives and Constraints*

For $x \in X \subset \mathbb{R}^n$, $y_q \in Y_q \subset \mathbb{R}^{m_q}$, $Y = (Y_1, Y_2, \dots, Y_Q)^t$, $z \in Z \subset \mathbb{R}^m$, $F : X \times Y_1 \times \ldots \times Y_Q \times Z \to \mathbb{R}$, $f_q : X \times Y_1 \times \ldots \times Y_Q \times Z \to \mathbb{R}$, and $q = 1, 2, \ldots Q$, S_9 is defined as:

$$\begin{array}{rl} \min_{x \in X} & F(x, y_1, \dots, y_q, z) = cx + \sum_{q=1}^{Q} d_q y_q + dz \\ \text{s.t.:} & Ax + \sum_{q=1}^{Q} B_q y_q + Bz \leq b \end{array}$$

where each y_q and z, for each value of x, solves the lower level problem:

$$\begin{array}{rl} \min_{y_q, z} & f_q(x, y_1, \dots, y_q, z) = c_q x + \sum_{j=1}^{Q} e_{qj} y_j + e_q z \\ \text{s.t.:} & A_q x + \sum_{j=1}^{Q} C_{qj} y_j \leq b_q \end{array} \tag{2.20}$$

where $c \in \mathbb{R}^n$, $c_q \in \mathbb{R}^n$, $d_q \in \mathbb{R}^{m_q}$, $d \in \mathbb{R}^m$, $e_{qj} \in \mathbb{R}^{m_j}$, $e \in \mathbb{R}^m$, $b \in \mathbb{R}^p$, $b_q \in \mathbb{R}^{l_q}$, $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, $B_q \in \mathbb{R}^{p \times m_q}$, $A_q \in \mathbb{R}^{l_q \times n}$, $C_{qj} \in \mathbb{R}^{l_q \times m_j}$, $C' \in \mathbb{R}^{l \times m}$, $j, q = 1, 2, \ldots Q$. Finally, this class has variables which are partially shared, but objectives and constraints which are individual.

Problems with independent followers (situation S_1) do not share objectives or constraints so that they can be written as:

$$
\begin{array}{ll}
\min_{\vec{x},\vec{y}_1\ldots\vec{y}_Q} & F(\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \\
\text{s.t.} & \\
 & G(\vec{x},\vec{y}_1,\ldots,\vec{y}_Q) \leq 0 \\
 & \\
 & \text{where each } \vec{y}_q \, (q = 1,\ldots,Q) \text{ solves} \\
 & \\
 & \quad \begin{array}{ll} \min_{\vec{y}_q} & f(\vec{x},\vec{y}_q) \\ \text{s.t.} & \\ & g(\vec{x},\vec{y}_q) \leq 0 \end{array}
\end{array}
\tag{2.21}
$$

2.3 Solution Approaches for Bilevel Problems

The following sections provide an overview of the current solution approaches for bilevel problems.

2.3.1 Decomposition Approaches

As bilevel problems are known to be non-convex, non-differentiable and strongly $\mathcal{NP}$-hard even in the simplest cases [CMS07], most approaches for solving them involve some form of decomposition of the problem into more tractable forms. A classic decomposition approach is to reduce the problems to a single-level problem which can then be solved using existing solution algorithms.

The most common method for single-level reduction is to replace the lower-level problem with its Karush-Kuhn-Tucker (KKT) [Ber99] conditions [Bar84, DZ12]. [VFIP96] present a decomposition approach for solving linear and quadratic bilevel problems. They transform the problem into a single-level one by replacing the inner problem with its KKT conditions. The problem is then decomposed into a series of primal and relaxed-dual sub-problems, whose solutions are used as lower and upper bounds. This procedure is run iteratively until a global optimum is found. This method is similar to that in [ZA14], where the problem is first reformulated into a single-level problem using KKT conditions and strong duality. Then, a similar iterative process is carried out until an optimal solution is found.

More complex approaches combine the use of the KKT conditions with other techniques. For example in [KHJ13] the stochastic bilevel problem is reduced to a single level using KKT conditions, and is then solved using Benders' decomposition. Related approaches include [SI09], where an algorithm for solving mixed-integer bilevel linear problems based on Benders' decomposition is presented. Similarly, [RBH14] uses logic-based Benders' decomposition to solve a bilevel vehicle routing problem, which is combined with a variable neighbourhood search heuristic to speed up search time and improve scalability. To address the problem of weak Benders' cuts, [NHG11] uses a Langrangian relaxation method to generate stronger cuts for a simultaneous scheduling and routing problem for automated guided vehicles.

In [DF16] lower-level problems are replaced with their Fritz-John conditions, and an algorithm is presented for solving problems with fully convex lower-levels. This method is applied in [DF14] to solve a bilevel road pricing problem. [NMMea04] show a relationship between one bilevel decomposition algorithm and a direct interior-point method based on Newton's method. A second contribution is that they bridge the gap between the convergence theories of bilevel decomposition algorithms and direct interior point methods. [IG98] present a decomposition algorithm for solving a network planning problem. The upper level is solved to get an upper bound, which is then used to get solution for the lower level problem which provides a lower bound. The process occurs iteratively, adding integer cuts along the way, until a small enough gap between the bounds is achieved.

[SNI$^+$12] solve a railway crew rostering problem. Their decomposition is in the form of cuts to reduce the feasible region of the master problem. Local search is also incorporated to improve the upper bound generated by solving the subproblems. In [CM16] the decomposition algorithm proposed consists of solving the single-level relaxation (SLR) of the Bilevel Facility Location (BFL) problem, solving the slave problem (SVP) which is the BFL for a given fixed set of open facilities, generating cuts based on the structure of the problem, and repeating until a stopping criterion is reached.

Decomposition approaches which also incorporate evolutionary approaches exist. In [CBS15] a Co-Evolutionary Decomposition based Bi-level Algorithm (CODBA) is presented, in which an algorithm is first used to generate a set of points from a discrete solution space. This allows them to generate a population of solutions for the lower-level problem. Several sub-populations of the lower-level problem are generated, and the best individuals in the sub-populations are allowed to co-evolve. CODBA II [CBSA15] is an improvement in which parallelism and co-evolution are implemented at both levels of the bilevel problem. Evolutionary approaches also exist where iterative approximation of the reaction set is used to approximate the lower-level problem [SMD13, SMD17b, SMD14]. Additionally, decomposition approaches involving evolutionary approaches are given in [LW10, LZC$^+$16]. A survey of metaheuristic approaches for bilevel optimisation is given in [Tal13].

2.3.2 Multiple-Follower Problems

As with single-follower bilevel problems, both classical and evolutionary approaches have been used for solving BLMF problems. [LSZ06] presents a general framework and solutions for these problems. Nine classes of multiple-follower problems are presented (none of which include the problem class proposed in Section 3.1) with corresponding models presented for each class. Also, an extended Kuhn-Tucker approach is presented for solving the uncooperative model to optimality. A practical example in the form of a road network problem is given. Similarly, [LSZR05] uses a Kuhn-Tucker (KT) approach for BLMF problems in which the followers may or may not have shared variables. [SLZZ05] and [LSZD07] use extended KT approaches. In [LSZR07] a branch-and-bound algorithm is used to solve the problems with referential-uncooperative followers. [CG07] reformulate a problem with multiple followers into one with one leader and one follower by replacing the lower levels with an equivalent objective and constraint region. This method cannot be applied to the BPMSIF, as neither its objectives nor its inducible region are equivalent to those of the problem class addressed in [CG07].

There is also literature on applying the Kth-best approach to problems with multiple followers. [SZL05] presents the theoretical properties of BLMFs and Kth-best approach for solving such problems, while [SZL$^+$07] uses the approach for problems with shared variables among followers. Similarly, [ZSL08] presents an extended Kth-best approach for solving referential-uncooperative BLMF decision problems, and provides an application in the form of a logistics planning problem.

Fuzzy approaches to solving BLMFs include [WWW09] which uses a fuzzy interactive algorithm to solve problems with partially shared variables among the followers. [ZL10] combines fuzzy models and a Kth-best algorithm to solve cooperative multiple-follower problems. Fuzzy models combined with a branch-and-bound algorithm have also been used in [ZLG08b, ZLG08a, ZLD07] to solve problems with shared decision variables among the followers.

Literature on evolutionary approaches to solving BLMFs include [AB15], where a differential evolution method is used to solve cases in which there is information shared between the followers. [Liu98] presents a genetic algorithm for solving non-linear multilevel problems with multiple followers. Also, [ISR16] extend their bilevel memetic algorithm to solve problems with multiple followers using a combination of global and local search. The global search algorithm is an $\epsilon-$constrained differential evolution algorithm, while the local search algorithm is an interior point algorithm incorporating both Sequential Quadratic Programming (SQP) and Trust Region (TR) methods. [KHRW16] combine fuzzy programming with an evolutionary algorithm, as well as neural networks to solve a multi-follower problem with non-cooperative followers. A good review of evolutionary approaches for solving BLMFs is also given in [SMD17a].

Chapter 3

A New Class of Bilevel Problems and An Analytics-Based Decomposition Approach

3.1 A Class of Bilevel Problems

The interest of this thesis is in problems with one leader and multiple followers $q = 1 \ldots Q$, where Q is large (i.e. hundreds or even thousands). As in evolutionary approaches, the decomposition approach used in this thesis is not restricted by conditions such as differentiability or convexity, but a different restriction is imposed. Several researchers (e.g. [LSZ06, ZL10]) have worked on bilevel optimisation with *multiple independent followers*, where *independence* means that the followers do not share variables other than leader variables (independence and other properties allow the use of a reduction technique).

This condition is now strengthened to one called *strong independence* and is one of the contributions of this thesis:

Definition 1 *A Bilevel Problem with Multiple Strongly-Independent Followers (BPMSIF) is one in which:*

(i) the followers do not share each others' follower or leader variables, so that $\vec{x}$ can be partitioned into $\vec{x}_q$ $(q = 1 \ldots Q)$.

(ii) follower problems $f_q(\vec{x}_q, \vec{y}_q)$ are allowed to be integer or non-linear.

(iii) variables from different follower problems are not tightly mutually constrai-

ned (though weak constraints such as a weighted sum of the variables are allowed).

All leader and follower variables may however appear in the leader's objective function. Thus the BPMSIF has the form:

$$\begin{aligned}
\min_{\vec{x}_1\ldots\vec{x}_Q,\vec{y}_1\ldots\vec{y}_Q} \quad & F(\vec{x}_1,\ldots,\vec{x}_Q,\vec{y}_1,\ldots,\vec{y}_Q)\\
\text{s.t.} \quad & \\
& G(\vec{x}_1,\ldots,\vec{x}_Q,\ \vec{y}_1,\ldots,\vec{y}_Q) \le 0\\
& \\
& \text{where each } \vec{y}_q\ (q=1,\ldots,Q) \text{ solves}\\
& \quad \min_{\vec{y}_q} \ f_q(\vec{x}_q,\vec{y}_q)\\
& \quad \text{s.t.}\\
& \qquad g_q(\vec{x}_q,\vec{y}_q) \le 0\\
& \qquad \vec{x}_q \in X_q,\ \vec{y}_q \in Y_q
\end{aligned} \tag{3.1}$$

where F, f_q may be any (possibly non-linear) objective functions, G, g_q may be any set of (possibly non-linear) constraints, the G constraints are weak, and X_q, Y_q may be vectors of any variable domains (real, integer, binary, or richer Constraint Programming domains such as set variables). The BPMSIF is shown to be different from the 9 classes of multiple-follower problems in section 3.2.

(3.1) satisfies the features of a BPMSIF in that:

- Each follower problem here can be seen to be a function of only its variables $\vec{y}_q$ and a portion of the the leader's variables $\vec{x}_q$.
- $G(\vec{x}_1,\ldots,\vec{x}_Q,\ \vec{y}_1,\ldots,\vec{y}_Q) \le 0$ is weak and may, for example, take the form of a simple weighted sum such as $\sum_q^Q B_q y_q \le b$, where the B_q and b are constants.

This problem is different from multiple-leader problems such as those in [RBA+16, LHHZ16], and [DX09] in that those problems have multiple leader objectives and solutions, whereas this problem has only a single leader with its single objective function.

The constraint region of the BPMSIF is:

$$\begin{aligned}\Omega = \{(\vec{x}_1,\ldots,\vec{x}_Q,\vec{y}_1,\ldots,\vec{y}_Q) \in\ & X_1 \ldots \times X_Q \times Y_1 \times \ldots \times Y_Q : \\ & G(\vec{x}_1,\ldots,\vec{x}_Q,\vec{y}_1,\ldots,\vec{y}_Q) \leq 0, g(\vec{x}_q,\vec{y}_q) \leq 0,\ q = 1,\ldots,Q\}\end{aligned} \tag{3.2}$$

The projection of Ω onto the leader's decision space is:

$$\begin{aligned}\Omega(X) = \{\vec{x}_q \in X_q :& \exists \vec{y}_q \in Y_q : G(\vec{x}_1,\ldots,\vec{x}_Q,\vec{y}_1,\ldots,\vec{y}_Q) \leq 0, \\ & g(\vec{x}_q,\vec{y}_q) \leq 0,\ q = 1,\ldots,Q\}\end{aligned} \tag{3.3}$$

The feasible set for follower q is affected by a corresponding part $\vec{x}_q$ of a given leader decision vector so that:

$$\Omega_q(\vec{x}_q) = \{\vec{y}_q : (\vec{x}_q,\vec{y}_q) \in \Omega\} \tag{3.4}$$

Each follower's rational reaction set is given as:

$$\Psi_q(\vec{x}_q) = \{\vec{y}_q \in Y_q : \vec{y}_q \in \text{argmin} f_q(\vec{x}_q,\vec{y}_q) \,|\, \vec{y}_q \in \Omega_q(\vec{x}_q)\} \tag{3.5}$$

Finally, the inducible region ($\mathcal{IR}$) is:

$$\begin{aligned}\mathcal{IR} = \{(\vec{x}_1,\ldots,\vec{x}_Q,\vec{y}_1,\ldots,\vec{y}_q) :& (\vec{x}_1,\ldots,\vec{x}_Q,\vec{y}_1,\ldots,\vec{y}_q) \\ & \in \Omega,\ \vec{y}_q \in \Psi_q(\vec{x}),\ q = 1,\ldots,Q\}\end{aligned} \tag{3.6}$$

As in standard bilevel programming min and argmin have been used without loss of generality: either problem or both could involve maximisation. In fact for all intents and purposes, the follower problems need not be linear, or even optimisation problems at all: follower q can be any algorithm that computes $\vec{y}_q$ from $\vec{x}_q$.

3.2 Motivation for an Analytics-Based Decomposition Approach

Consider the following example of a BPMSIF with Q non-linear follower problems.

$$\min_{x_1,\ldots,x_Q,y_1,\ldots,y_Q} \sum_{q=1}^{Q} c_q^t x_q + \sum_{q=1}^{Q} d_q^t y_q \tag{3.7}$$

$$\text{s.t.: } \sum_{q=1}^{Q} A_q^t x_q + \sum_{q=1}^{Q} B_q^t y_q \leq b_0, \tag{3.8}$$

where $y_q, q = 1, \ldots, Q$ solves:

$$\min_{y_q} \; v_q^t x_q \times w_q^t y_q \tag{3.9}$$

$$\text{s.t.: } E_q x_q + D_q y_q \leq b_q, \tag{3.10}$$

where $x_q \in \mathbb{R}^{n_q}, q = 1, \ldots, Q$ are the leader's variables partitioned among the q followers and $y_q \in \mathbb{R}^{n_q}, q = 1, \ldots, Q$ are the variables controlled by the qth follower. $c_q, d_q, v_q, w_q \in \mathbb{R}^{n_q}$, $b_0 \in \mathbb{R}^{m_0}$, $b_q \in \mathbb{R}^{m_q}$, $A_q, B_q \in \mathbb{R}^{m_0 \times n_q}$, $E_q, D_q \in \mathbb{R}^{m_q \times n_q}$. The superscript t means transposition.

The constraint set of the BPMSIF is:

$$\begin{aligned} S = \{(x_1, \ldots, x_Q, y_1, \ldots, y_Q) : &\sum_{q=1}^{Q} A_q^t x_q + \sum_{q=1}^{Q} B_q^t y_q \leq b_0, \\ &E_q x_q + D_q y_q \leq b_q, q = 1, \ldots, Q\} \end{aligned} \tag{3.11}$$

where S is assumed to be non-empty and compact.

The feasible set for the qth follower given corresponding x_q is:

$$S_q(x_q) = \{y_q : D_q y_q \leq b_q - E_q x_q\} \tag{3.12}$$

The projection of S onto the leader's decision space is:

$$\begin{aligned} S(X) = \{(x_1, \ldots, x_Q) : \exists (y_1, \ldots, y_Q) : &\sum_{q} A_q^t x_q + \sum_{q} B_q^t y_q \leq b_0, \\ &E_q x_q + D_q y_q \leq b_q, q = 1, \ldots, Q\} \end{aligned} \tag{3.13}$$

The rational reaction set for the qth follower for $x_q \in S(X)$ is:

$$P_q(x_q) = \text{argmin}_{y_q}\{v_q^t x_q \times w_q^t y_q : y_q \in S_q(x_q)\} \tag{3.14}$$

Finally, the inducible region or feasible region of the leader is:

$$\begin{aligned} IR_{BPMSIF} = \{(x_1, \ldots, x_Q, y_1, \ldots, y_Q) : (x_1, \ldots, x_Q, y_1, \ldots, y_Q) \in S, \\ y_q \in P_q(x_q), q = 1, \ldots, Q\} \end{aligned} \tag{3.15}$$

Using the above definitions, the BPMSIF with linear follower problems can now be written as:

$$\begin{aligned} \min_{x_1,\ldots,x_Q,y_1,\ldots,y_Q} & \sum_{q=1}^{Q} c_q^t x_q + \sum_{q=1}^{Q} d_q^t y_q \\ \text{s.t.:} & \\ & (x_1, \ldots, x_Q, y_1, \ldots, y_Q) \in IR_{BPMSIF} \end{aligned} \tag{3.16}$$

Comparing equations (1a) and (2) in [CG07] with (3.7) and (3.16) respectively, it can be seen that the BPMSIF is different from the problem in [CG07] and cannot therefore be solved using their method. Also, comparing (3.7) to (3.16) with situations S_1 to S_9 in [LSZ06] shows that the problems in addressed in this thesis belong to a different class. Additionally, the methods proposed in [CG07] and [LSZ06] function on the condition that the follower problems are linear. In fact, most classical methods for handling bilevel problems require assumptions of smoothness, linearity or convexity. An alternative approach not restricted by conditions of linearity or convexity will therefore be a useful contribution to the literature. The problems considered in this thesis are not restricted to being linear or having linear follower problems. The follower problems may be non-linear, or may be any function that computes a $\vec{y}_q$ given an $\vec{x}_q$. These non-standard cases preclude the use of traditional optimisation approaches such as replacing the follower problems with their KKT conditions, or using Benders' decomposition.

A number of evolutionary and meta-heuristic techniques have been developed which do not require assumptions of linearity or convexity, but most of these are computationally intensive nested strategies. Consequently, while these strategies may be efficient at solving smaller problems, they do not lend themselves to the efficient solution of large-scale problems.

In addition to being able to handle non-linear objectives and constraints, the decomposition approach proposed in this thesis is very suitable for solving large-scale problems faster and more efficiently than evolutionary approaches (see section 3.5.3 for examples).

3.3 A Decomposition Technique

The industrial problem modelled in Chapter 4, is a large-scale, mixed-integer, non-linear bilevel optimisation problem with tens of thousands of followers. Such problems cannot be solved using classical methods. Evolutionary methods can be applied but much better results were found using the decomposition technique described here.

Recall that the general form of the *strongly-independent* multiple-follower bilevel problem is:

$$\begin{aligned}
\min_{\vec{x}_1\ldots\vec{x}_Q,\vec{y}_1\ldots\vec{y}_Q} \quad & F(\vec{x}_1,\ldots,\vec{x}_Q,\vec{y}_1,\ldots,\vec{y}_Q) \\
\text{s.t.} \quad & \\
& G(\vec{x}_1,\ldots,\vec{x}_Q,\ \vec{y}_1,\ldots,\vec{y}_Q) \leq 0 \\
& \\
& \text{where each } \vec{y}_q \, (q=1,\ldots,Q) \text{ solves} \\
& \qquad \min_{\vec{y}_q} \ f_q(\vec{x}_q,\vec{y}_q) \\
& \qquad \ \text{s.t.} \\
& \qquad \qquad g_q(\vec{x}_q,\vec{y}_q) \leq 0 \\
& \qquad \qquad \vec{x}_q \in X_q,\ \vec{y}_q \in Y_q
\end{aligned} \tag{3.17}$$

The first step in the decomposition is to generate a reasonably large number of assignments for each set $\vec{x}_q$ (see Section 3.4.2 for details). For simplicity it will be assumed that the same number of assignments S is generated for each set, and the assignments are denoted by $\vec{X}_{sq}$ $(s = 1 \ldots S,\ q = 1 \ldots Q)$. Next, the associated follower problems are solved to obtain assignments $\vec{Y}_{sq}$. An illustration of the approach is shown in figure 3.1.

In scatter plot (a) the circle represents the hypersphere of possible assignments of $\vec{x}_q$, with a small random number of them selected shown as dots. Plot (b) shows the result of solving the associated follower problems to obtain a small set of $\vec{y}_q$ vectors. The space of the $\vec{y}_q$ vectors is unknown and might have a very

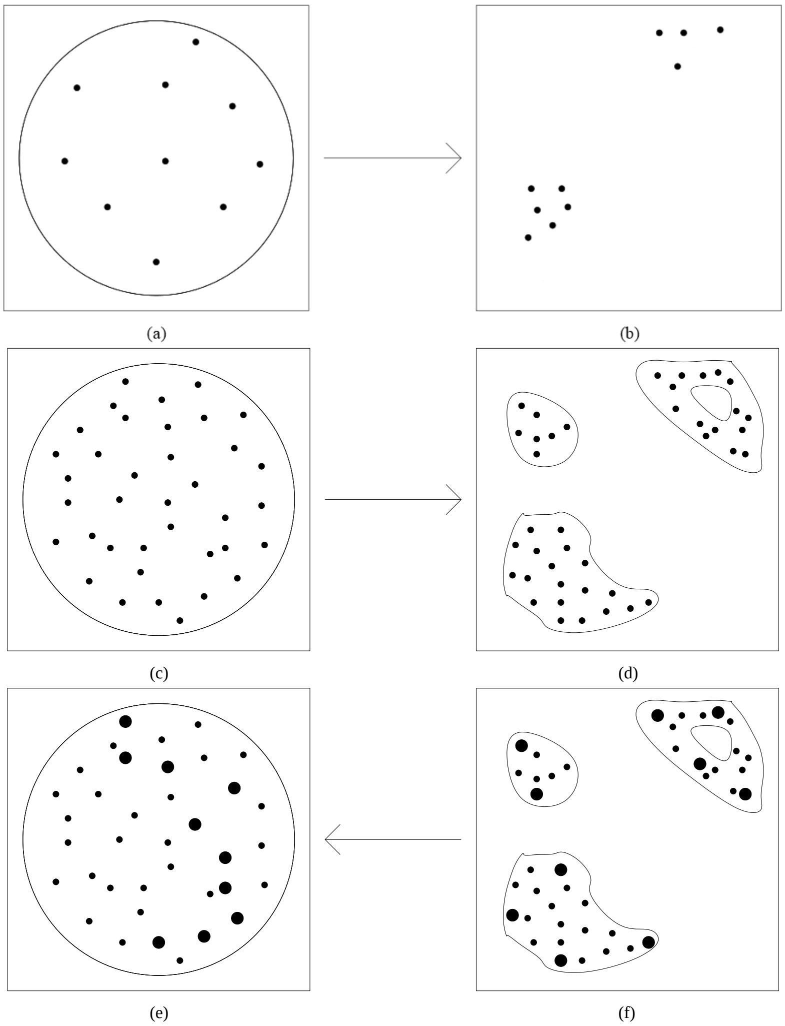


Figure 3.1: Illustration of the analytics-based approach for a single follower

different shape to that of the $\vec{x}_q$, as shown. As a consequence, a small random set of $\vec{x}_q$ might correspond to a very non-random small set of $\vec{y}_q$ vectors, showing the inadequacy of merely sampling a few leader vectors.

Instead a large number of $\vec{x}_q$ is sampled, as shown in plot (c), with their corresponding $\vec{y}_q$ shown in plot (d): this represents the use of Monte Carlo simulation to approximate the distribution of the $\vec{y}_q$.

A small number K of the $\vec{y}_q$ is then selected via a k-medoids algorithm, or a self-organising map, to approximately cover the estimated distribution, highlighted in plot (f). Finally, a record of which $\vec{x}_q$ corresponds to which $\vec{y}_q$ is used to derive the non-random set of $\vec{x}_q$ highlighted in plot (e). This results in a smaller but representative set of assignments $\vec{X}_{kq}$ and $\vec{Y}_{kq}$.

Figure 3.1 shows the decomposition process for a single follower. This process is applied to all followers, and the results are used to reduce the problem from a multi-level one into a single level.

The original bilevel problem can now be transformed into a standard optimisation problem:

$$\begin{array}{rll} \min_{\vec{x}_1 \ldots \vec{x}_Q} & F(\vec{x}_1 \ldots \vec{x}_Q, \vec{y}_1 \ldots \vec{y}_Q) & \\ \text{s.t.} & \vec{x}_q = \vec{X}_{kq} \to \vec{y}_q = \vec{Y}_{kq} & (q = 1 \ldots Q,\ k = 1 \ldots K) \\ & \vec{x}_q \in \{\vec{X}_{kq} \,|\, k = 1 \ldots K\} & (q = 1 \ldots Q) \\ & \vec{y}_q \in \{\vec{Y}_{kq} \,|\, k = 1 \ldots K\} & (q = 1 \ldots Q) \\ & G(\vec{x}_1 \ldots \vec{x}_Q,\ \vec{y}_1 \ldots \vec{y}_Q) \leq 0 & \end{array} \tag{3.18}$$

Constraint $\vec{x}_q = \vec{X}_{kq} \to \vec{y}_q = \vec{Y}_{kq}$ says that if the variables in $\vec{x}_q$ are assigned the values in $\vec{X}_{kq}$ then the variables in $\vec{y}_q$ must be assigned the values in $\vec{Y}_{kq}$. This model can be implemented directly in many Constraint Programming systems, and it can easily be linearised (assuming appropriate objective F) for solution by ILP by introducing binary variables x_{kqi}, y_{kqi} to denote that component i of $\vec{x}_q$ is assigned to component i of $\vec{X}_{kq}$ and component i of $\vec{y}_q$ is assigned to component i of $\vec{Y}_{kq}$, and posting constraints of the form

$$x_{kqi} \leq y_{kqi} \qquad \sum_k x_{kqi} \geq 1 \qquad \sum_k y_{kqi} \leq 1 \tag{3.19}$$

The motivation behind this decomposition is that ILP (or other technologies) can be used to find a global optimum from a vast number of possibilities: K^Q possible combinations of assignments to the $\vec{x}_q$, which in the application of Chapter 4 can be of the order of 10^{450}. Of course the result is almost certainly not an optimal solution to the original problem, but if the $Q \times K$ assignments are chosen to cover all possibilities reasonably well (which involves analytics methods) then it should be near-optimal.

Note that relaxing strong independence (for example by using tight constraints G) reduces the number of combinations covered: in the extreme case where all followers use the same leader variables and have exactly one optimal solution each, only S different assignments are obtained, and the approach reduces to the trivial method of finding S random solutions then choosing the best.

3.4 Analytics Approaches

The analytics approaches used in this thesis are laid out in this section.

3.4.1 Distribution Learning

Given a number of samples drawn from a distribution, the goal of distribution learning is to find the distribution from which the samples have been drawn with a high degree of certainty. Let $\mathcal{D}_n$ be a particular distribution class, and let $D \in \mathcal{D}_n$ be a distribution which has a support S, i.e. S is the range over which D is defined. To represent the probability distribution D over S, let G_D be a generator for D [KMR+94]. G_D is called a generator because, given a random input y, G_D simulates sampling from the distribution D and outputs an observation $G_D[y] \in S$.

Given independent random samples from D, as well as confidence and approximation parameters δ and ϵ respectively, the goal of any learning algorithm is to output an approximate distribution D' with a probability δ in polynomial time. The distance between this approximate distribution and the original distribution is $d(D, D')$, and can be measured in several ways. These include the Kolmogorov distance (from the Kolmogorov-Smirnoff test) [CLR67], the Total Variation distance [CA33], and the Kullback-Leibler divergence [KL51]. When $d(D, D') \leq \epsilon$, G_D is called an ϵ-good generator.

For the multiple-follower bilevel problem, it is necessary to understand the relationship between the leader decision vectors $\vec{x}_q$ and the corresponding follower vectors $\vec{y}_q$. Once this is known, given a set of leader decision vectors, the optimal follower decision vectors can be determined. This can be achieved by applying the principles of distribution learning. The distributions of the the leader and follower vectors are approximated by using Monte Carlo sampling, by generating a large number of random $\vec{x}_q$ and solving the follower problems to obtain corresponding $\vec{y}_q$. This process can be thought of as the generator G_D. To avoid bias, the $\vec{x}_q$ are generated by uniform sampling from a vector space, as opposed to simply randomising each component of $\vec{x}_q$.

3.4.2 Generating Uniformly Distributed Random Vectors

To generate a random vector $\vec{Y}$ that is uniformly distributed over an irregular n-dimensional region G, an acceptance-rejection method may be used [RK16].

For a regular region W, where W may be multidimensional in nature, a random vector $\vec{X}$, which is uniformly distributed in W, is first of all generated. If $\vec{X} \in G$, accept $\vec{Y} = \mathbf{X}$ as the random vector uniformly distributed over G. Otherwise, reject $\vec{X}$, and generate a new random vector. In the case when G is an n-dimensional unit ball, i.e.

$$G = \left\{ x : \sum_i x_i^2 \leq 1 \right\} \tag{3.20}$$

a uniformly distributed random vector $\vec{X} = (X_1, X_2, \ldots, X_n)^T$ is generated, and accepted if it falls inside the n-ball. The algorithm used for this is taken from [KTB13] and is described below.

Algorithm 1 Random Vector Generation

```
1: Generate $n$ random variables $U_1, \ldots, U_n$ as iid variables from $U(0,1)$.
2: Set $X_1 = 1 - 2U_1, \ldots, X_n = 1 - 2U_n$ and $R = \sum_{i=1}^{n} X_i^2$
3: If $R \leq 1$, accept $\vec{X} = (X_1, \ldots, X_n)^T$ as the desired vector; otherwise go to Step 1.
```

The efficiency of the acceptance-rejection method rapidly reduces as the number of dimensions n increases, with an efficiency of 0.016 for $n = 8$ [RK16]. Consequently, for dealing with problems with $n > 8$, a more efficient algorithm (the Hypersphere Point Picking method [Mul59, Mar72]) is used. For instance in Section 4.5, the problem solved has $n = 11$ and the efficiency of Algorithm 1 for such n is only $= 0.0009$. This has motivated the use of Algorithm 2 for problems with $n > 8$.

Algorithm 2 Hypersphere Point Picking

```
1: Generate $n$ random variables $x_1, \ldots, x_n$ as iid variables from $N(0,1)$.
2: Return $\frac{1}{\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ as the desired random vector.
```

By using Algorithm 1 or Algorithm 2 (depending on the size of n), a large number of leader vectors $\vec{X}_{sq}$ can be generated, which can then be used to solve the respective follower problems to obtain a corresponding large set of follower decision vectors $\vec{Y}_{sq}$. In order to then solve the bilevel problem, this large number of vectors must be reduced to a smaller number. To do this, two well-known machine learning techniques - k-medoids clustering and self-organising maps - are used.

3.4.3 k-Medoids Clustering

The k-medoids algorithm is a clustering algorithm used for partitioning a data set X into K homogeneous groups or clusters, i.e. $C = \{C_1, C_2, ..., C_K\}$ [dAF12]. Unlike the k-means algorithm, partitioning is done around *medoids* (or *exemplars*) rather than *centroids*. A medoid m_k is a data point in a cluster C_k which is most similar to all other points in that cluster. This is vital, as a small set of $\vec{y}_q$ that are each generated from some known $\vec{x}_q$ is required.

A k-medoids algorithm seeks to minimize the function

$$f = \sum_{k=1}^{K} \sum_{i \in C_k} d\left(x_i, m_k\right) \tag{3.21}$$

where $d\left(x_i, m_k\right)$ is a distance metric measuring the dissimilarity between data entity x_i and the medoid m_k of the cluster [dAF12]. Commonly used distance metrics are the Manhattan distance or Euclidean distance [MRS⁺15] and the latter is used in this thesis. The most common algorithm for k-medoid clustering is the Partitioning Around Medoids (PAM) algorithm [dAF12], presented at a high level in Algorithm 3.

Nevertheless, PAM is not very efficient with very large datasets. In such cases, the CLARA algorithm, which is a combination of PAM and random sampling, is commonly used [KR09, WLH00]. The speed-up by CLARA over PAM is achieved by analysing data subsets of fixed size, which has the effect of making both computational and storage complexity linear, as opposed to quadratic [KR09, Nag15]. The mean of the dissimilarities of the observations to their closest medoid is:

$$MeanDistance(M, X) = \frac{\sum_{i=1}^{n} d\left(x_i, rep\left(M, x_i\right)\right)}{|X|} \tag{3.22}$$

where $d(x_i, x_j)$ is the dissimilarity between two data points x_i and x_j, $rep(M, x_i)$ returns the closest medoid m_k to x_i from the set M, and $|X|$ is the number of

Algorithm 3 Partitioning Around Medoids (PAM)

```
Input: Input data $X = \{x_1, x_2, x_3, \ldots, x_n\}$, number of desired clusters $K$
Output: A set of clusters $C$
Arbitrarily select $k$ out of $n$ data entities as initial medoids $m_1, m_2, ...m_k$
Calculate the initial distance $f_{ik} = d(x_i, m_k)$
Repeat:
    for each $y_i$ not a medoid do:
        for each medoid $m_k$ do:
            Calculate the distance $f_{ik} = d(x_i, m_k)$
            If $f_{ik} <$ current $f_{ik}$ then:
                swap $y_i$ for $m_k$
            end if
        end for
        until $f_{ik} \geq$ current $f_{ik}$
    end for
for each $x_i \in X$ do:
    Assign $x_i$ to $C_k$ where $d(x_i, m_k)$ is the smallest over all medoids;
end for
Return clusters $C$ with final medoids $m'_1, m'_2, ...m'_k$
```

items in X. The steps for CLARA are outlined in Algorithm 4 below.

Algorithm 4 Clustering LARge Applications (CLARA)

```
Input: Input data $X = \{x_1, x_2, x_3, \ldots, x_n\}$, number of desired clusters $K$, $t$
Output: A set of clusters $C$
Repeat $t$ times:
    Randomly choose $p$ sub-datasets of fixed size from the large dataset $X$.
    Partition each sub-dataset into $k$ clusters using PAM, getting a set $M = \{m_1, m_2, ..., m_k\}$ of $k$ medoids.
    Associate each $x_i$ to its nearest medoid $m_k$.
    Calculate $MeanDistance(M, X)$
Return the set of medoids $M' = \{m'_1, m'_2, ...m'_k\}$ with the minimum $MeanDistance(M, X)$.
```

Once the large set of follower decision vectors $\vec{Y}_{sq}$ has been reduced to a much smaller representative set $\vec{Y}_{kq}$, the bilevel problem can subsequently be modelled and solved as an ILP (see Section 3.5 for numerical examples).

3.4.4 Self-Organising Maps

As an alternative to k-medoids clustering (which can be time consuming), a neural network approach is also considered. Self-organising maps (SOMs) [Koh82], also known as Kohonen Maps after their developer Teuvo Kohonen, are a type of artificial neural network in which, given a set of input data vectors, individual neurons in the map compete to align themselves with the vectors they are best matched with. The operation of SOMs is based on the idea of Competitive Learning, and is described briefly below.

Consider a dataset $\mathbf{X}$ in which the observations $\vec{x_i}$ $(i = 1 \ldots |\mathbf{X}|)$ are n-dimensional vectors, i.e. $\vec{x_i} \in \mathbb{R}^n$. Let these observations be known as the input vectors. Also consider a neural network with neurons arranged in a 2D lattice (see Figure 3.2). This arrangement is known as a rectangular topology, but others exist (e.g. hexagonal) and may be of any dimension (e.g. 1D, 3D, etc.). Let $\mathcal{J}$ be a set comprising of all neurons in the map, with the total number of neurons J equal to $|\mathcal{J}|$. Each input vector is connected to every neuron in the network, and each neuron has a weight vector $\vec{w_j}$ associated with it. The dimensions of the weight vector equals that of the input vectors, i.e. $\vec{w_j} \in \mathbb{R}^n$, $(j = 1 \ldots J)$.

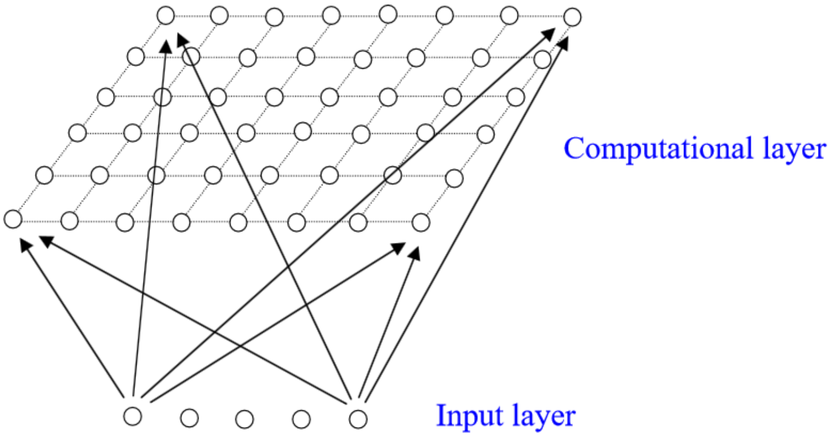


Figure 3.2: Example of a self-organising map topology [Bul04a]

Each neuron has a neighbourhood comprising of neurons within a particular radius, and this is illustrated in figure 3.3.

For example, in the left panel of figure 3.3, neuron 13 has a neighbourhood of radius 1, and a neighbourhood $N_{13}(1) = \{8, 12, 13, 14, 18\}$. On the right, neuron 13 has a neighbourhood radius of 2, with

$N_{13}(2) = \{3, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19, 23\}$.

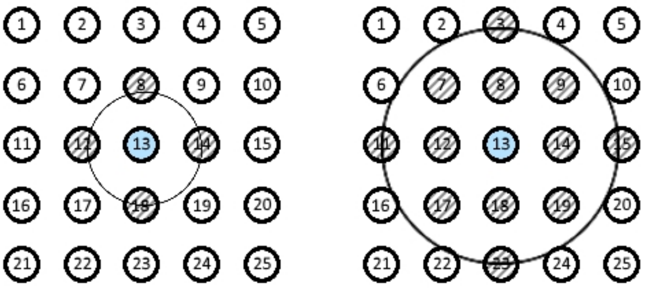


Figure 3.3: Neighbourhoods in SOMs

SOMs operate in two phases - a training phase in which the map is built using sample data, and a mapping phase in which new data is automatically classified based on the map built in the previous phase. An algorithm for the training phase is given below.

Algorithm 5 Self-Organising Map training procedure

```
procedure SOM TRAINING
$s \leftarrow 1$
$s_{max} \leftarrow$ maximum number of iterations
$\sigma_0 \leftarrow$ radius of the map
$L_0 \leftarrow$ initial learning rate
$\lambda \leftarrow$ time constant
Randomise $\vec{w_j}, \forall j \in J$
    while $s < s_{max}$ do
        for each $\vec{x_i} \in \mathbf{X}$ do
            for each neuron $j \in J$ do
                Calculate the distances $||\vec{x_i} - \vec{w_j}||^2$
                Select BMU: the neuron for which
               $||\vec{x_i} - \vec{w_c}||^2 = \min_j \{||\vec{x_i} - \vec{w_j}||^2\}$
               $\sigma(s) \leftarrow \sigma_0 e^{-s/\lambda}$
               $L(s) \leftarrow L_0 e^{-s/\lambda}$
               $d^2 \leftarrow ||\vec{r_j} - \vec{r_c}||^2$
               $\theta(s) \leftarrow e^{-d^2/2\sigma(s)^2}$
               $\vec{w_j} \leftarrow \vec{w_j} + \theta(s)L(s)[\vec{x_i} - \vec{w_j}]$
            end for
        end for
        $s \leftarrow s + 1$
    end while
end procedure
```

To begin, the weights of all neurons are randomised. Then, for every input vector

$\vec{x_i}$, the squared Euclidean distance between that vector and all neurons in the map is calculated. The best-matching neuron (BMU) is the neuron whose weight vector is closest to the input vector. The subscript c in $\vec{w_c}$ is the index of the neuron for which the distance measure is minimised. Once the BMU has been found, all the neurons in the neighbourhood of the BMU (as well as the BMU) are updated using the update formula (line 18). This formula is a function of the radius $\sigma(s)$ of the neighbourhood as well as the learning rate $L(s)$. $\sigma(s)$ is an exponential decay function so that as the number of iterations s increases, the radius of the neighbourhood decreases, until the BMU is the only neuron affected. $L(s)$ also decays exponentially which ensures that the SOM converges.

$\theta(s)$ is the neighbourhood function and it ensures that neurons closer to the BMU learn (or are modified) more than nodes farther away. d is the Euclidean distance between the current neuron under consideration j and the BMU, where the vector r_j defines the position of the node j in the lattice, and the vector r_c defines the position of the BMU. In this way, the map gets trained over several iterations, using only the competition between neurons, and without any external influence. This lack of influence is the reason why neural networks of this type are called “self-organising".

Once trained, the SOM can be used for the classification of new data points (a new input vector is automatically classified by the “mapping”). Hence, the number of neurons of the SOM determines the number of classes. In this thesis however, the intention is not to use SOMs for classifying new data. Instead, the map generated by the SOM trained over the large data set is used to select a smaller but highly representative subset of the data. Note that the size of the reduced set is equal to the number of neurons. This reduced dataset is then used to solve the bilevel optimisation problem. Figures 3.4 3.5 and 3.6 give an illustration of this approach.

Starting with a 2-D lattice of neurons, figure 3.4, the map is trained on the underlying data. After several iterations, the map will have aligned itself with the underlying distribution, as shown in figure 3.5. Once the map is aligned, the data points closest to each neuron is selected, one data point for each neuron. Figure 3.6 shows a zoomed in section of figure 3.5, with the selected data points in red. This reduced dataset can then be used to solve the transformed multiple-follower bilevel problem.

Following, four particularly useful properties of SOMs that make them ideal for this application are analysed.

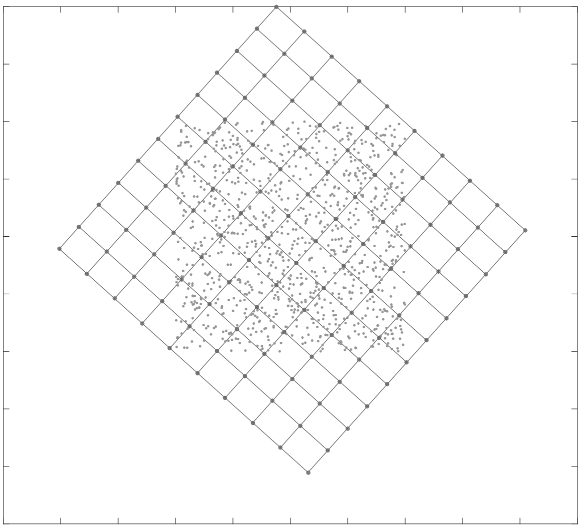


Figure 3.4: SOM lattice over data

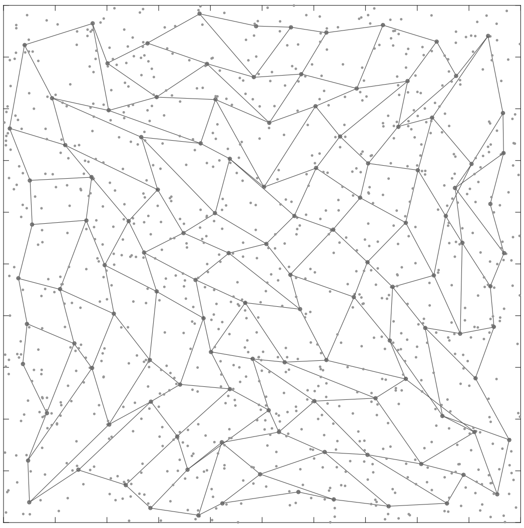


Figure 3.5: Map aligned with data

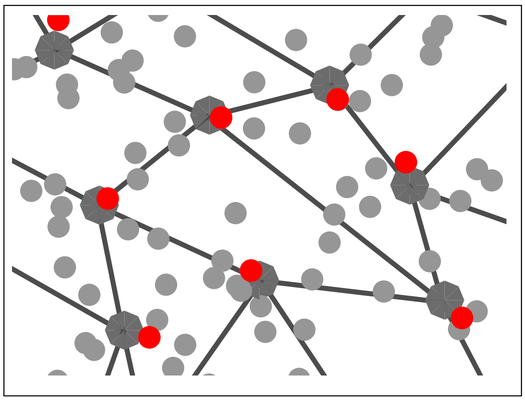


Figure 3.6: Selection of points closest to neurons

Property 1 ***Approximation of the Input Space:*** *The feature map Φ generated during training, and defined by the set of weight vectors $\vec{w_j}$ in the output (or map) space, provides a good approximation to the input (or data) space [Bul04b].*

This property means that given a set of input vectors $\mathbf{x}$ which have a probability density function $p(\mathbf{x})$, the weight vectors $\vec{w_j}$ approximate to $p(\mathbf{x})$ with minimal residual error. To illustrate this, a brief look at Vector Quantization (VQ) theory will be taken, since SOMs belong to the class of VQ algorithms [Bul04a, Koh90]. VQ aims to encode a large set of vectors into a smaller, representative set (called a codebook) to achieve (historically) data compression. This is done by dividing the dataset into groups having a number of data points and then selecting the centroid of each group as the representative or code vector, resulting in a Voronoi tessellation of the input space.

For a code vector $\vec{c_k}$, $(k = 1, \ldots, K)$, the VQ approximation or distortion error D is given as

$$D = \int p(\mathbf{x}) ||\mathbf{x} - \vec{c_k}||^2 d\mathbf{x} \tag{3.23}$$

If, as is the case in practical applications, there is some noise ν which has a distribution $\pi(\nu)$ during the encoding procedure, then D has the form

$$D_\nu = \int p(\mathbf{x}) \int \pi(\nu) ||\mathbf{x} - \vec{c_k}||^2 d\nu d\mathbf{x} \tag{3.24}$$

For a finite number of code vectors, D_ν may be rewritten as

$$D_\nu = \sum_i \int_{V_i} \sum_k \pi(i,k)||\mathbf{x} - \vec{c_k}||^2 p(\mathbf{x}) d\mathbf{x} \tag{3.25}$$

where V_i is the Voronoi region of polygon i. Clearly, in order to produce an encoding of the highest quality, VQ aims to minimize D_ν, and this may be done using iterative procedure such as stochastic gradient descent. For example, for a code vector $\vec{c_j}$, the gradient is

$$\frac{\partial \sum_k \pi(i,k)||\mathbf{x} - \vec{c_k}||^2}{\partial \vec{c_j}} = 2\pi(i,k)||x - \vec{c_j}|| \tag{3.26}$$

Thus it can be seen that the update formula (line 18) corresponds to equation 3.26, with probability density function $\pi(i,k)$ analogous to the neighbourhood function $\theta(s)$ and $||\mathbf{x} - \vec{c_k}||$ analogous to $||\mathbf{x} - \vec{w_j}||$.

Similarly, SOMs can be thought of as reducing the distortion error at each step of the algorithm, thus producing a good approximation of to the input space.

Property 2 ***Topological Ordering:*** *Φ preserves the topological order of the data space in that the position of a neuron in the map corresponds to a particular domain or attribute of the input patterns [Bul04a].*

This occurs because at each iteration, the update formula causes the weight vector $\vec{w_c}$ of the BMU to move closer to the input vector $\vec{x_i}$. Additionally, the update of neurons in the neighbourhood of the BMU results in the entire output space becoming suitably ordered.

Property 3 ***Density Matching:*** *Variations in the statistics of the input distribution are reflected in the feature map Φ. Regions in the data space from which the input vectors x_i are drawn with high probability of occurrence are mapped onto larger domains of the map space. These regions therefore have better resolution than those in the input space from which the $\vec{x_i}$ are drawn with low probability.*

It has been shown for one-dimensional maps, that the point density of the code vectors $Q(\mathbf{x})$ is proportional to the probability density of the input vectors $P(\mathbf{x})$ [Bul04b, LO92].

$$Q(\mathbf{x}) \propto P(\mathbf{x})^{2/3} \tag{3.27}$$

The difference in resolution is due to the fact that the exponent is 2/3 and not 1, so that low input density regions end up being marginally over-represented.

Property 4 ***Feature Selection:*** *Given data from an input space with a non-linear distribution, the self-organising map is able to select a set of best features for approximating the underlying distribution.*

Given the fact that the SOM provides a good approximation to the input space (Property 1), preserves the topological order (Property 2) and reflects the density of the input data (Property 3), it follows that the feature map Φ is able to select a set of features which best approximate the underlying distribution.

To use the SOM for clustering, i.e. reducing $\vec{Y}_{sq}$ to $\vec{Y}_{kq}$, the number of neurons in the lattice is first set to K. The SOM is then trained on the $\vec{Y}_{sq}$ over several iterations. Once the training phase is complete, the closest follower vector to each neuron is selected, resulting in a set $\vec{Y}_{kq}$. Finally, the leader vectors $\vec{X}_{kq}$ corresponding to the $\vec{Y}_{kq}$ are selected, resulting in a clustered set.

3.5 Numerical Examples

In order to illustrate and evaluate the decomposition approach, some example problems are considered below. Monte Carlo simulation and clustering were done in Java and R (using the `CLARA` package [MRS+17]) respectively. The CPLEX 12.6 solver was also used on a 3.0 GHz Intel Xeon Processor with 8 GB of RAM.

3.5.1 Benefits of Using Clustering

In order to illustrate the benefits of using clustering, consider the following simple example with one leader and two followers [1]. The variables x_1,x_2,y,z are real in $(0, 1)$. The problem is considered for the optimistic case in which the followers' solutions lead to the best objective function value for the leader.

---

[1] The example is being used to show the benefits of clustering, but this problem could actually be solved analytically by the leader: given any single value of x_1, there is exactly one value of x_2 that would result in the optimal solution.

$$
\begin{aligned}
\min_{\vec{x}} \quad & h(y,z) = |y - z| \\
\text{s.t.} \quad & \\
& \min_y f(x_1, y) = |y - x_1^{0.01}| \\
& \\
& \min_z g(x_2, z) = |z - (1 - x_2)^{0.01}| \\
& x_1, x_2, y, z \in (0,1)
\end{aligned}
\tag{3.28}
$$

The leader chooses x_1, x_2 to minimise $h(y,z)$, follower 1 chooses y to minimise $f(x_1, y)$, and follower 2 chooses z to minimise $g(x_2, z)$. For most values of x_1, the first follower will choose a value of y close to 1, while the second will choose values of z close to 0 for most values of x_2. This is illustrated in figures 3.7 and 3.8. $1,000$ random $\vec{x}$ variables were generated, and both $f(x_1, y)$ and $g(x_2, z)$ were evaluated using these $\vec{x}$ values. Figure 3.7 is a histogram for $f(x_1, y)$ showing that for the given x_1's, most values of y are close to 1. The histogram in figure 3.8 similarly shows that most values of $g(x_2, z)$ are close to 0.

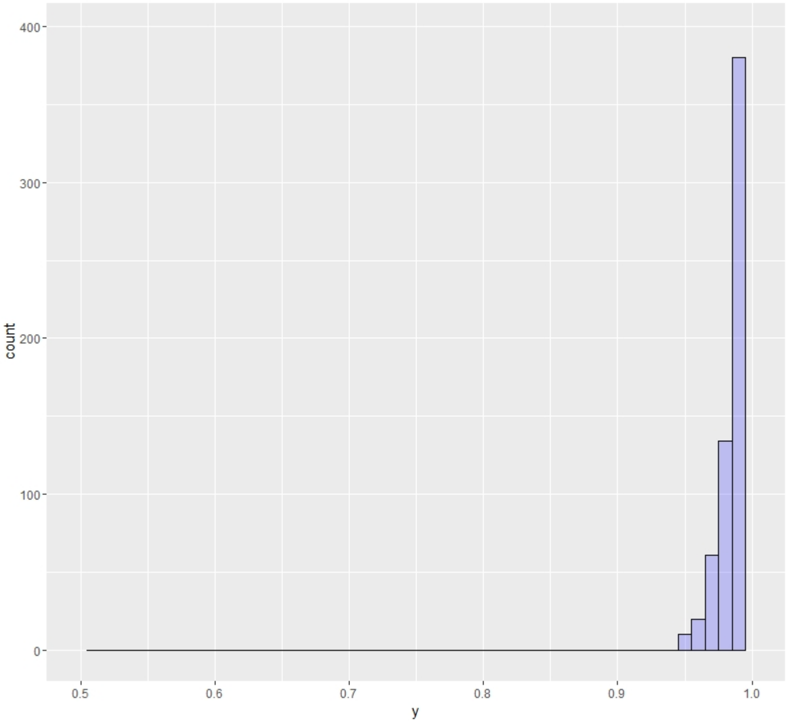


Figure 3.7: Histogram showing the distribution of the y's

In order to minimise $h(y,z)$ however, the leader must select a value for $\vec{x}$ that gives similar y and z values. These values can be anywhere between 0 and 1, but must be similar. This means that the leader must generate a large number of $\vec{x}$

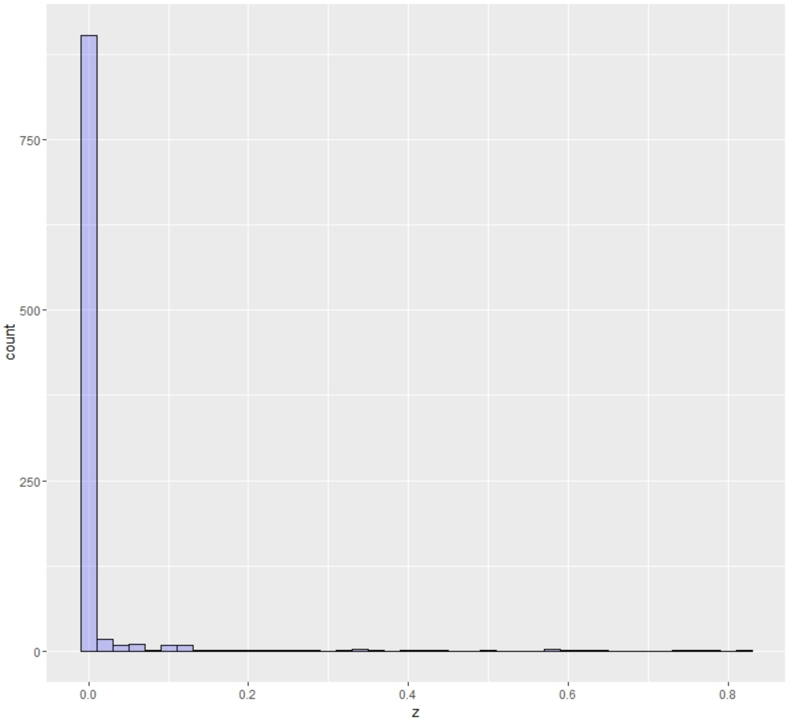


Figure 3.8: Histogram showing the distribution of the z's

values in order to find such cases with reasonable probability. This however results in a large h-optimisation problem (especially in problems with many followers). In order to be able to solve such a problem in a reasonable time frame, it is therefore essential that the large samples be reduced to a smaller but diverse set that includes values of y and z which are close to each other. This is achieved by using clustering.

First, the variables U, V and W are introduced to reformulate the absolute values as:

$$\begin{aligned}
\min\ & W \\
\text{s.t.} & \\
& -y + z \leq W \\
& y - z \leq W \\
& \\
& \min U \\
& \text{s.t.} \\
& -y + x_1^{0.01} \leq U \\
& y - x_1^{0.01} \leq U \\
& \\
& \min V \\
& \text{s.t.} \\
& -z + (1 - x_2)^{0.01} \leq V \\
& z - (1 - x_2)^{0.01} \leq V
\end{aligned} \tag{3.29}$$

The decomposition technique can now be applied to this problem. $S = 10,000$ random $\vec{x}$ vectors were generated, and f and g were solved using these vectors to get a large set of possible solutions Y_s and Z_s ($s = 1, \ldots, S$) for y and z respectively. These solutions were then clustered using $K = 100$ to get a smaller set of possible solutions. Problem (3.29) can then be reformulated as a single-level linear programme:

$$\begin{aligned}
\min\ & W \\
\text{s.t.} & \\
& -y + z \leq W \\
& y - z \leq W \\
& t_k = 1 \rightarrow y = Y_k \quad k = 1, \ldots, K \\
& t_k = 1 \rightarrow z = Z_k \quad k = 1, \ldots, K \\
& \textstyle\sum_k t_k = 1 \\
& t_k \in \{0, 1\}
\end{aligned} \tag{3.30}$$

where the binary variable t_k indicates which of the potential K solutions is chosen. Solving (3.30) gives an objective function value of 0.0456, with $\vec{x} = (0.746, 4.98 \times 10^{-5})$, $y = 0.9971$ and $z = 0.9514$. In order to show that simply increasing the number of samples does not lead to an appreciable increase in solution quality, the problem was solved using an increasing number of samples S. For each value of S, the problem was solved 10 times and the average solution recorded in order to account for the variability in the randomly generated samples.

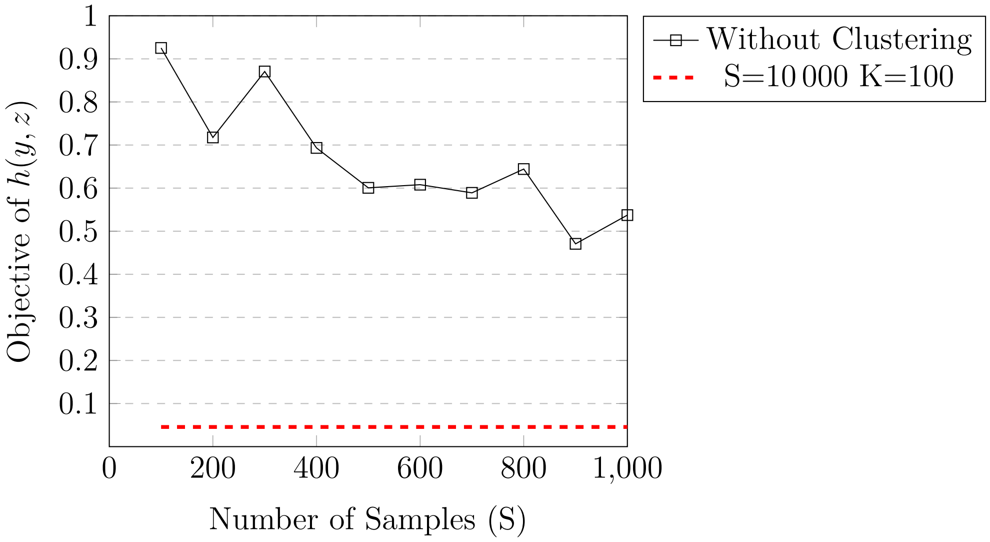


Figure 3.9: Benefits of Clustering

Figure 3.9 shows that as the number of samples increase, the objective value decreases, but not in an appreciable manner. Using increasingly larger samples without clustering also leads to an increase in the size of the problem, along with unreasonable solution times. For instance, attempting to solve the problem using $10,000$ samples did not produce a solution before the time cut-off of 3600 seconds. In contrast, using clustering (red dashed line) gives the lowest objective function value in a short amount of time (88.72 seconds).

3.5.2 A Benchmark Problem

The second problem considered is Example 2 from [Bar88], and is a two-follower problem:

$$
\begin{aligned}
\max\quad & F(\vec{x},\vec{y_1},\vec{y_2}) = (200 - y_{11} - y_{21})(y_{11} + y_{21}) \\
& +(160 - y_{12} - y_{22})(y_{12} + y_{22}) \\
\text{s.t.} & \\
& x_1 + x_2 + x_3 + x_4 \leq 40 \\
& 0 \leq x_1 \leq 10, 0 \leq x_2 \leq 5, 0 \leq x_3 \leq 15, 0 \leq x_4 \leq 20 \\
& \\
& \min f_1(\vec{y_1}) = (y_{11} - 4)^2 + (y_{12} - 13)^2 \\
& \text{s.t.} \\
& 0.4y_{11} + 0.7y_{12} \leq x_1 \\
& 0.6y_{11} + 0.3y_{12} \leq x_2 \\
& 0 \leq y_{11}, y_{12} \leq 20 \\
& \\
& \min f_2(\vec{y_2}) = (y_{21} - 35)^2 + (y_{22} - 2)^2 \\
& \text{s.t.} \\
& 0.4y_{21} + 0.7y_{22} \leq x_3 \\
& 0.6y_{21} + 0.3y_{22} \leq x_4 \\
& 0 \leq y_{21}, y_{22} \leq 40
\end{aligned}
\tag{3.31}
$$

This problem fits the problem class proposed in section 3.1 as the followers are strongly independent. The followers do not share each others' follower or leader variables, and the follower problem variables are not mutually constrained. The leader vector $\vec{x} = (x_1, x_2, x_3, x_4)$ is partitioned among the followers with variables (x_1, x_2) occurring in follower 1 and (x_3, x_4) in follower 2. The variables $\vec{y_1} = (y_{11}, y_{12})$ and $\vec{y_2} = (y_{21}, y_{22})$ are also computed by followers 1 and 2 respectively.

To solve this problem using the analytics-based decomposition method, denote (x_1, x_2) by a vector $\vec{\lambda}_1$ and (x_3, x_4) by a vector $\vec{\lambda}_2$. A large number S of assignments for $\vec{\lambda}_1$ and $\vec{\lambda}_2$ which satisfy the bounds of the $\vec{x}$'s are generated (Section 3.4.2), and denoted by $\vec{\Lambda}_{s1}$ and $\vec{\Lambda}_{s2}$ $(s = 1 \dots S)$ respectively. For each $\vec{\lambda}_1$ in $\vec{\Lambda}_{s1}$ the corresponding follower problem $\vec{f_1}$ is solved as an ILP, obtaining assignments $\vec{Y}_{s1}$; similarly for $\vec{Y}_{s2}$. Next, the $\vec{Y}_{s1}$ vectors are clustered using k-medoids to get the most diverse set of assignments $\vec{Y}_{k1}$, $(k = 1 \dots K)$. The $\vec{\Lambda}_{k1}$ vectors that correspond to the $\vec{Y}_{k1}$ are then selected. The same is done for $\vec{Y}_{s2}$ to obtain $\vec{Y}_{k2}$ along with its corresponding $\vec{\Lambda}_{k2}$. Using this decomposition, problem (3.31) can now be rewritten as a standard optimisation problem:

$$
\begin{aligned}
\max \quad & \vec{F}(\vec{x}, \vec{y}_1 \dots \vec{y}_2) = (200 - y_{11} - y_{21})(y_{11} + y_{21}) \\
& +(160 - y_{12} - y_{22})(y_{12} + y_{22}) \\
\text{s.t.} \quad & \\
& \lambda_{11} + \lambda_{12} + \lambda_{21} + \lambda_{22} \leq 40 \\
& u_k = 1 \rightarrow \vec{\lambda}_1 = \vec{\Lambda}_{k1} && k = 1 \dots K \\
& u_k = 1 \rightarrow \vec{y}_1 = \vec{Y}_{k1} && k = 1 \dots K \\
& \textstyle\sum_k^K u_k = 1 \\
& v_k = 1 \rightarrow \vec{\lambda}_2 = \vec{\Lambda}_{k2} && k = 1 \dots K \\
& v_k = 1 \rightarrow \vec{y}_2 = \vec{Y}_{k2} && k = 1 \dots K \\
& \textstyle\sum_k^K v_k = 1
\end{aligned}
\tag{3.32}
$$

where $\lambda_{11} = x_1$, $\lambda_{12} = x_2$, $\lambda_{21} = x_3$ and $\lambda_{22} = x_4$. This model can be linearised using the big-M approach, however this ILP is solved faster when CPLEX's indicator constraints are used [2]. The binary variables u_k and v_k ensure that only one assignment each is selected from $\vec{\Lambda}_{k1}$ and $\vec{Y}_{k1}$, and from $\vec{\Lambda}_{k2}$ and $\vec{Y}_{k2}$ respectively. The $\lambda_{11} + \lambda_{12} + \lambda_{21} + \lambda_{22} \leq 40$ constraint ensures that an (x_1, x_2) and an (x_3, x_4) that satisfy the original constraints on the $\vec{x}$'s are selected.

Using $S = 10,000$, figure 3.10 shows how the objective value varies with K. The red line shows the optimal value of 6600. As K increases, the value of the objective trends upwards, with the highest value of 6594.05 obtained when $K = 160$ giving $\vec{x} = (8.13, 3.80, 11.23, 16.82)$, $\vec{y}_1 = (0.74, 11.20)$ and $\vec{y}_2 = (28.04, 0.00)$ (rounded to 2 decimal places).

The clustering time when $K = 160$ is 234.53 seconds. This solution is only 0.09% less than the optimal, but *the strength of this approach is in its ability to handle large-scale problems.* This is demonstrated next.

3.5.3 A Large-Scale Problem

In this experiment, a problem with arbitrarily many followers is evaluated. The problem is also evaluated for the optimistic case in which the followers' solutions

---

[2] Indicator constraints are a way of expressing if-else relationships among variables [IBM17].

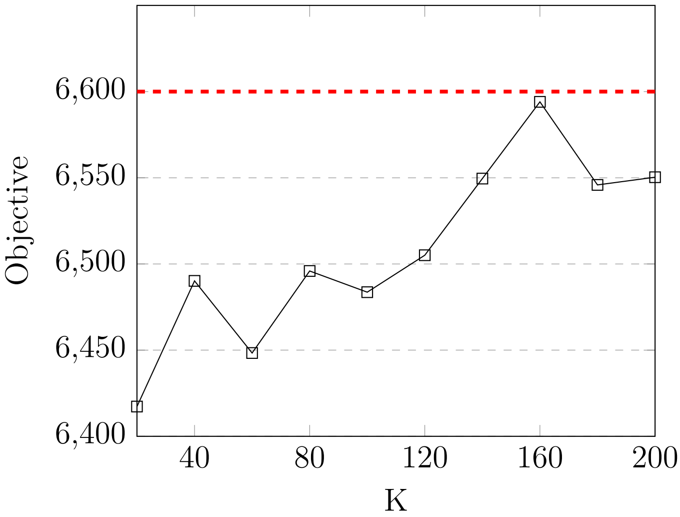


Figure 3.10: Objective value as K increases

lead to the best objective function value for the leader.

$$\begin{array}{lll} \max & \sum_q^Q \vec{a}_q\vec{x}_q + \sum_q^Q \vec{b}_q\vec{y}_q & \\ \text{s.t.} & \vec{x}_q \in \mathbb{R}^N & \forall q \\ & x_{qn} \le x_{qn}^{max} & \forall q,n \\ & q = 1\ldots Q \begin{cases} \vec{y}_q \in \text{argmin } \vec{c}_q\vec{x}_q + \vec{d}_q\vec{y}_q & \\ \text{s.t. } \sum_n^N y_{qn} \le \sum_n^N x_{qn} & \\ \quad y_{qn} \ge e_{qn}x_{qn} & n = 1\ldots N \\ \quad \mathbf{y}_q \in \mathbb{R}^n & \\ \quad y_{qn} \le y_{qn}^{max} & n = 1\ldots N \end{cases} & \end{array} \tag{3.33}$$

where $\vec{x}$ and $\vec{y}$ are the variables controlled by the leader and followers respectively, and Q is the total number of followers. Both the $\vec{x}$ and $\vec{y}$ are vectors of real numbers. The leader variables are partitioned among the followers such that each follower contains one $\vec{x}_q$ each, and each $\vec{x}_q$ is of size n. Each component of the vector x_{qi} is constrained to be $\le$ a given upper bound x_{qi}^{max}. $\vec{a}$, $\vec{b}$, $\vec{c}$, $\vec{d}$ and $\vec{e}$ are vectors of constants. (Where $\sum_q \vec{a}_q\vec{x}_q = \sum_q \sum_n a_{qn}x_{qn}$)

3.5.3.1 Decomposition

For each $\vec{x}_q$, a number S of feasible solutions, i.e. $\vec{X}_{sq}$ are generated satisfying the constraint $x_{qn} \le x_{qn}^{max}$. The follower problems are then solved exactly using an ILP solver to find corresponding $\vec{Y}_{sq}$. Then, k-medoids clustering is applied to the $\vec{Y}_{sq}$ for each follower $q \in Q$ to obtain a small but diverse set of assignments $\vec{Y}_{kq}$,

$(k = 1 \dots K)$. The $\vec{X}_{kq}$ that correspond to the selected $\vec{Y}_{kq}$ are then also selected. Using $\vec{X}_{kq}$ as the domain of $\vec{x}_q$, and $\vec{Y}_{kq}$ as the domain of $\vec{y}_q$, the problem is now formulated and solved as an ILP. The binary variable u_{kq} is used to indicate that components n of $\vec{x}_q$ and $\vec{y}_q$ are assigned to components n of $\vec{X}_{kq}$ and $\vec{Y}_{kq}$ respectively. The problem can now be written as:

$$\begin{array}{lll}
\max & \sum_q^Q \sum_n^N a_{qn}x_{qn} + \sum_q^Q \sum_n^N b_{qn}y_{qn} & \\
\text{s.t.} & u_{kq} = 1 \rightarrow x_{qn} = X_{kqn} & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\
& u_{kq} = 1 \rightarrow y_{qn} = Y_{kqn} & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\
& \sum_k^K u_{kq} = 1 & q = 1 \dots Q \\
& u_{kq} \in \{0, 1\} & k = 1 \dots K, q = 1 \dots Q
\end{array} \tag{3.34}$$

It can also be written in a more standard form using standard big-M notation, where M is a sufficiently large constant:

$$\begin{array}{lll}
\max & \sum_q^Q \sum_n^N a_{qn}x_{qn} + \sum_q^Q \sum_n^N b_{qn}y_{qn} & \\
\text{s.t.} & x_{qn} - X_{kqn} \leq M(1 - u_{kq}) & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\
& X_{kqn} - x_{qn} \leq M(1 - u_{kq}) & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\
& y_{qn} - Y_{kqn} \leq M(1 - u_{kq}) & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\
& Y_{kqn} - y_{qn} \leq M(1 - u_{kq}) & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\
& \sum_k^K u_{kq} = 1 & q = 1 \dots Q \\
& u_{kq} \in \{0, 1\} & k = 1 \dots K, q = 1 \dots Q
\end{array} \tag{3.35}$$

3.5.3.2 Evaluation

The values used for the problem are $N = 6$, $x_{qn}^{min} = 0$, $x_{qn}^{max} = 10$, $y_{qn}^{max} = 10$, $(\forall q, n)$. a_{qn}, b_{qn}, c_{qn}, and d_{qn} are Gaussian random real variables in $[0.0, 15.0)$, $[0.0, 20.0)$, $[-10.0, 10.0)$ and $[-12.0, 12.0)$ respectively. e_{qn} is a uniform random real variable in $[0.0, 1.0)$. The number of followers Q was varied between 10–1000, and the problem was solved using both the decomposition approach (using $S = 1000$, $K = 30$ for each follower) and two genetic algorithms, and the results are shown in 3.16 to 3.19. The first genetic algorithm is the Nested Bilevel Evolutionary Algorithm (N-BLEA) used in [SMFD14]. The second is the Multiple-Follower Genetic Algorithm (MFGA) described in Algorithm 6, and was custom-built for this problem.

Algorithm 6 Genetic Algorithm for multiple follower bilevel problems

```
Generate initial population of size $popSize$ of leader individuals $\mathbf{x}_q \, \forall Q$
for each follower $q$:
    for each leader individual in population:
        Solve follower problem to get a population of follower solutions
    end for
end for
Calculate $fitnessFunction$ for each member of the population
while $g < maxGens$:
    Evolve Population:
        Get elite leader individuals from population
        Generate new leader individuals using selection, crossover, mutation
        Add elite and newly generated individuals to create new population
    for each follower $q$:
        for each leader individual in new population:
            Solve follower problem to get a population of follower solutions
        end for
    end for
    Evaluate fitness of new population
    $g \leftarrow g + 1$
end while
Return $\mathbf{x}_q \, \forall Q$ with best fitness
```

N-BLEA Parameters In order to select the parameters to use, the problem with 100 followers ($Q = 100$) was first solved while varying some algorithm parameters. The number of parents μ and number of offspring λ ($\mu = \lambda$) were varied from 3 to 8. For each of these values, the number of generations (*maxGens*) was also varied from 50 to 200 in steps of 50. This operation was run 10 times for each value of μ, λ and *maxGens*, and the average objective value was recorded. It was seen that the settings where $\mu = \lambda = 8$, produced the highest objective function value on average (table 3.1, figure 3.11). The number of generations corresponding to this highest value was 150. Also, on average, using 150 generations produced the highest objective value regardless of what μ and λ were (rightmost column of table 3.1).

Tournament selection was used to select the parents, with a tournament size of 5. The number of random individuals to add to the pool was $r = 2$, the crossover probability was fixed at 0.9 and the mutation probability at 0.1. The constraint handling method used by the algorithm is given in [Deb00]. For both the upper and lower levels of the bilevel problem, the sum of the violations of

Table 3.1: N-BLEA tuning.

	$\mu = \lambda$							
maxGens	3	4	5	6	7	8	9	**Average**
50	4311.465	4203.471	4620.746	3437.061	3808.777	4189.745	3186.241	3965.358
100	4224.603	2351.168	3626.946	4918.847	4194.442	5625.257	4594.144	4219.343
150	4157.886	4792.996	4255.045	4322.606	6024.241	5688.582	3693.932	4705.041
200	4170.737	4372.070	4265.723	5936.813	3985.628	4823.669	4659.901	4602.077
Average	4216.173	3929.926	4192.115	4653.832	4503.272	5081.813	4033.554	

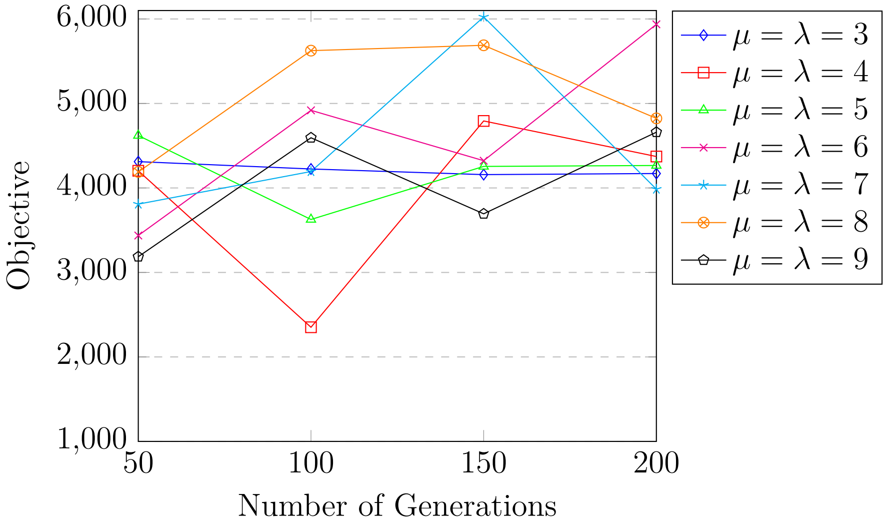


Figure 3.11: N-BLEA tuning for $Q = 100$

all the constraints for any point makes up a constraint violation. If a member has no constraint violation, it is judged feasible and is preferred over infeasible members. Population members with smaller constraint violations are preferred by the algorithm over members with higher constraint violations.

N-BLEA uses a variance-based termination criteria, which was set to 0.000001. Once the improvement in solution reaches this value, the algorithm terminates.

MFGA Parameters To generate a new offspring, uniform crossover, with a crossover rate of 0.5 (50%) was used. For each gene, if the crossover rate is less than 0.5, the offspring's gene is taken from parent 1. Otherwise it is taken from parent 2. For mutation, we go through genes and mutate a gene (by generating a new vector) if a random generated value is less than the mutation rate. Since vector generation is done using Algorithm 2 with the appropriate boundaries, MFGA always produces feasible offspring.

In order to select the parameters to use, the problem with 100 followers ($Q = 100$) was also first solved while varying some algorithm parameters. The population size (*popSize*) was varied from 30 to 90, while also varying the maximum number of generations *maxGens* from 50 to 500.

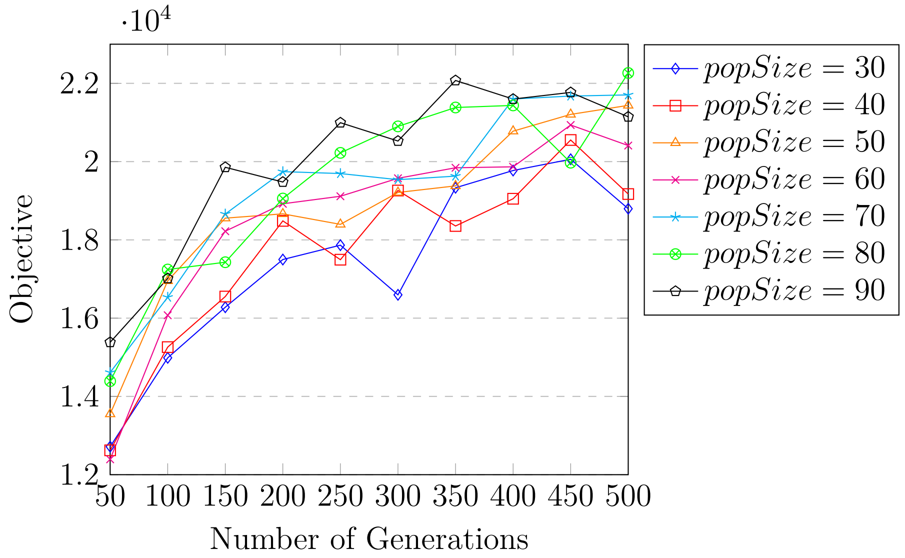


Figure 3.12: MFGA tuning for $Q = 100$ (Objective values)

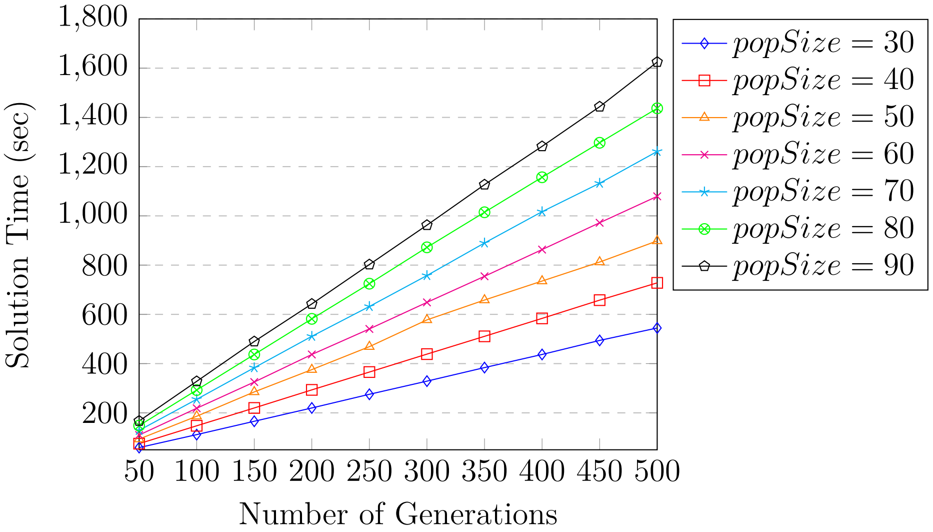


Figure 3.13: MFGA tuning for $Q = 100$ (Solution time (sec))

Figures 3.12 and 3.13 show the objective values and timing respectively for tuning MFGA. It can be seen that the objective value tends to increase as the number of generations and the population size increase. In terms of solution time, the larger the population size, the more time was taken. The MFGA parameters selected were therefore: $maxGens = 500$, $popSize = 50$. This population size was selected because although there is not much of a difference between its objective value and the best objective at $popSize = 80$, the difference in time taken is almost

50% less. Other parameters are $elitePercentage = 0.20$, $tournamentSize = 5$, $mutationRate = 0.015$ and $fitnessFunction = \sum_q^Q \sum_n^N a_{qn}x_{qn} + \sum_q^Q \sum_n^N b_{qn}y_{qn}$.

Selection of K for the Decomposition Approach In order to choose an appropriate value of K, the 100-follower problem ($Q = 100$) was solved using decompositions where the value of K varied from 10 to 80 in steps of 10. The objective function value with $K = 10$ ($\theta_{K=10}$) was taken as the base value, and the percentage increase in solution quality as K increased was plotted in figure 3.14. For example, the percentage increase seen when using $K = 40$ instead of $K = 10$ is:

$$\frac{\theta_{K=40} - \theta_{K=10}}{\theta_{K=10}} \times 100 = 19.49\% \tag{3.36}$$

and the percentage increase seen when using $K = 80$ instead of $K = 10$ is:

$$\frac{\theta_{K=80} - \theta_{K=10}}{\theta_{K=10}} \times 100 = 31.69\% \tag{3.37}$$

Using $\theta_{K=10}$ as the base value is a heuristic lower bound, as such a small number of clusters is not likely to adequately represent the data. It can be seen that increasing the number of clusters (and thus medoids) initially results in an increase in the quality of the solution. The best solution is seen when $K = 30$. At this point, further increasing K gives solutions which are not as good, as the increase in the number of clusters means that different medoids are selected. These medoids at other values of K are not as representative of the dataset as those of $K = 30$, and therefore do not provide the best inputs to the ILP. This pattern is also seen for the 90 follower problem ($Q = 90$) in figure 3.15, as well as in figure 3.10 for the benchmark problem.

Comparing all 3 approaches For both N-BLEA and MFGA, each problem size was solved 10 times, and the average objective values and solution times were recorded. It should be noted that the poor performance of N-BLEA is due to the operation of its crossover operator which is additive in nature, and frequently violates the bounds of the vectors. This crossover operator results in offspring which are frequently infeasible, and are thus *heavily* penalised by the constraint handling scheme. For this reason, MFGA was custom-made for this problem to avoid the heavy penalties seen with N-BLEA.

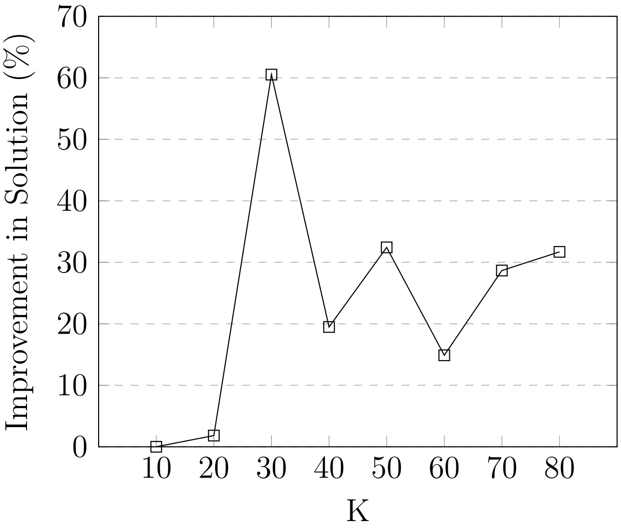


Figure 3.14: Selection of K for $Q = 100$

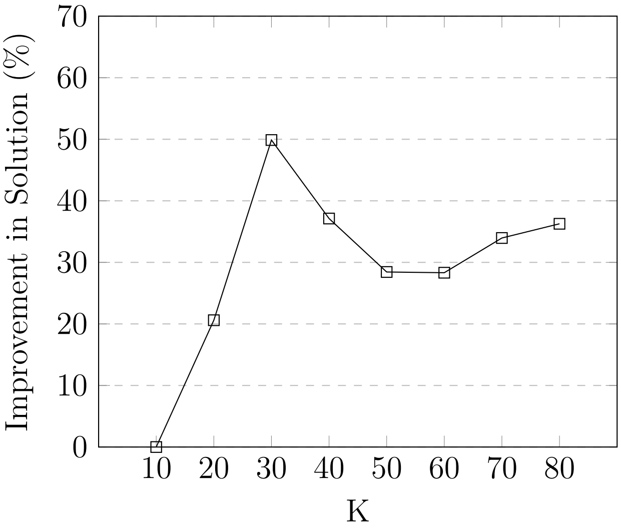


Figure 3.15: Selection of K for $Q = 90$

For 10–100 followers, the solution found by the MFGA was better in 7 out of 10 of the cases, though the decomposition approach finds a close solution in a fraction of the time (figures 3.16, 3.17). However, as the problems get larger, (Q from 100–1000) the decomposition approach is much better in terms of both the solution quality and the runtime (figures 3.18, 3.19), especially as Q gets larger.

This experiment demonstrates the usefulness of the analytics-based decomposition approach for large-scale problems. Reduction of the very large set of potential solutions to a much smaller, but highly representative set using medoids allows the ILP to choose the best solution from a vast number of possibilities.

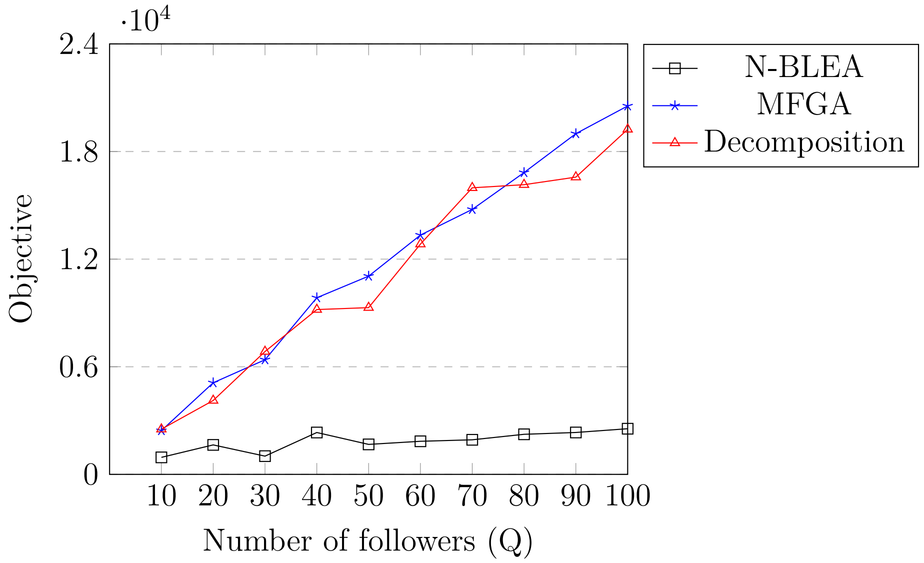


Figure 3.16: Comparing Approaches: Objectives for $Q = 10$ to 100

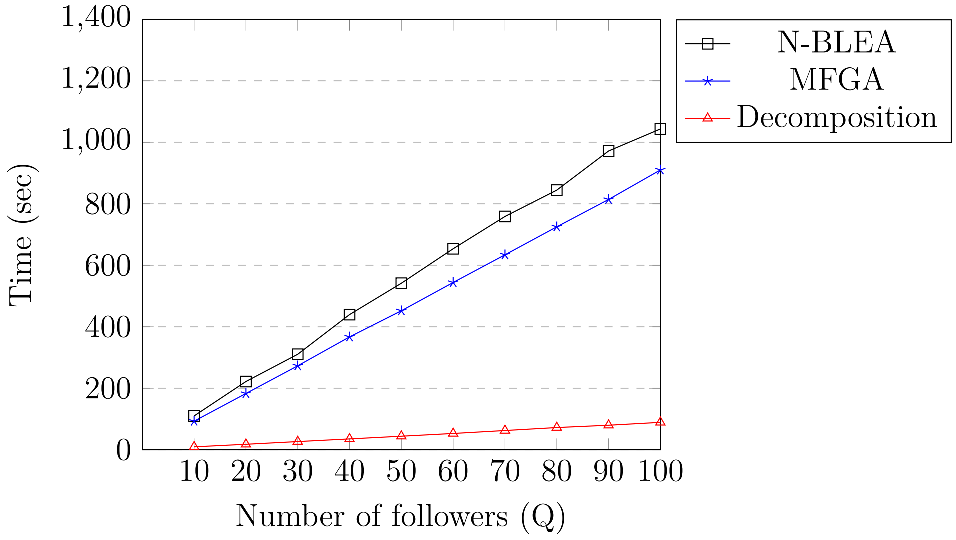


Figure 3.17: Comparing Approaches: Timing for $Q = 10$ to 100

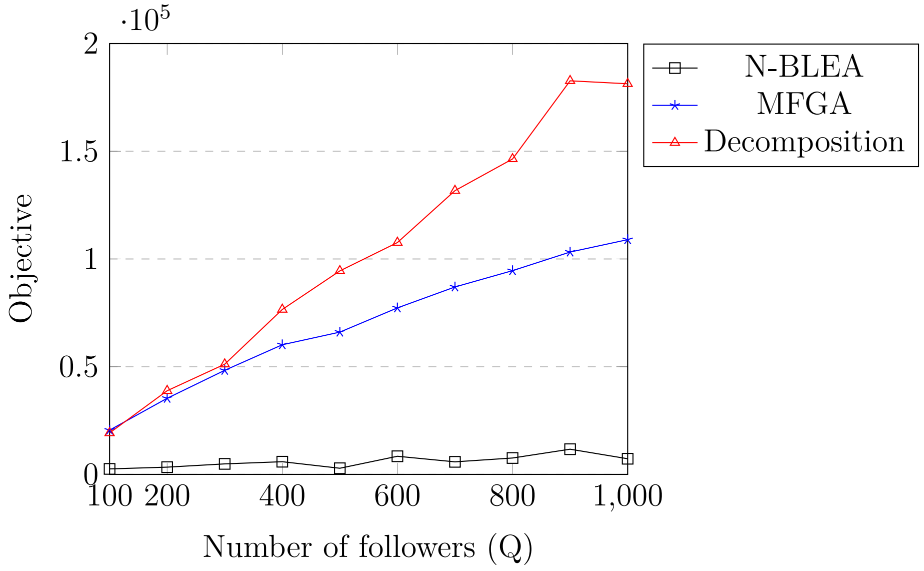


Figure 3.18: Comparing Approaches: Objectives for $Q = 100$ to 1000

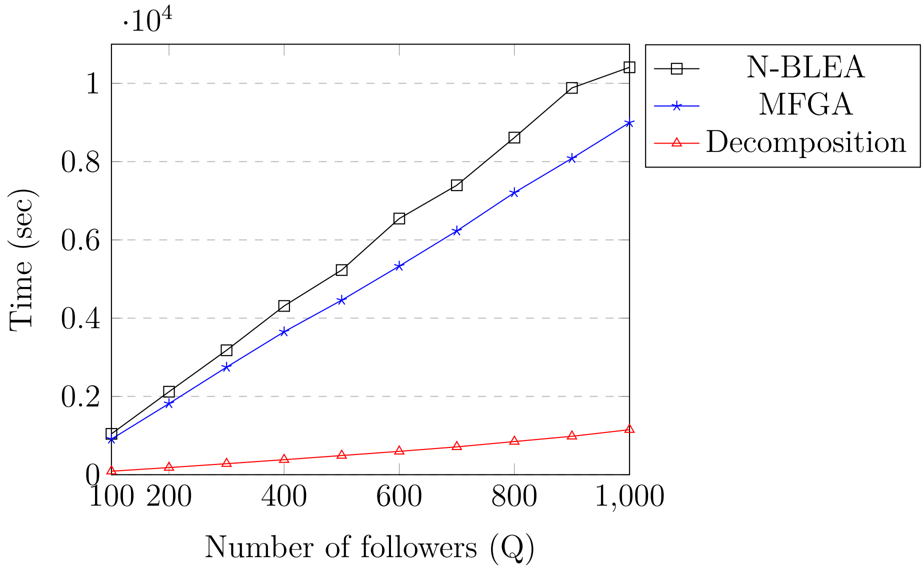


Figure 3.19: Comparing Approaches: Timing for $Q = 100$ to 1000

3.6 Conclusions

An analytics-based decomposition approach which uses random vector generation, k-medoids clustering or self-organising maps to reduce large multiple-follower bilevel problems to a single-level ILP has been proposed. A two-follower benchmark problem was solved using this approach, and the solution obtained was only about 0.09% less than optimal. As the hypothesis of the approach is that large-scale problems will benefit from this approach, a large problem with up to 1000 followers was also solved. The solutions obtained were compared to those obtained using two different genetic algorithms, and it was seen that the decomposition approach obtained better solutions, especially when the size of the problem increased. Regardless of problem size, the decomposition approach produced solutions in a much shorter time. As the clustering times for the example problems were reasonable, self-organising maps were not used here. They are however used in Section 4.5 for a large-scale problem.

Although the problems featured in this chapter were artificial problems, this approach can also be used on real-life problems such as in Chapter 4.

Chapter 4

A Large-Scale Bilevel Cutting Stock Problem

4.1 The Cutting Stock Problem

The *Cutting Stock Problem* (CSP) [GG61] is a well-known NP-hard optimisation problem in Operations Research, and is of interest in many industries because of the perennial problem of dividing large pieces of stock material into smaller pieces with minimal waste. It is often modelled as an ILP:

$$\begin{array}{ll} \min & \sum_{i=1}^{n} c_i x_i \\ \text{s.t.} & \sum_{i=1}^{n} a_{ij} x_i \geq d_j \quad \forall j = 1, \ldots, m \\ & x_i \geq 0, \ \text{integer} \end{array} \tag{4.1}$$

where there are m different types of smaller pieces, and d_j is the demand for piece-type j. There are n cutting patterns and x_i is a decision variable which states how many times pattern i is used. The number of small pieces of type j generated by pattern i is a_{ij}. The objective is to minimize the total cost, where c_i is the cost (usually the waste) associated with using pattern i.

Variants of the CSP have been studied. The above problem is one-dimensional but two or three dimensions might be necessary [GG65]. The problem might be multi-stage, involving further processing after cutting [FM13], or might be combined with other problems, e.g. [HFS96]. Additional constraints might be imposed because of user requirements. Widths might be continuous though restricted to certain ranges of values. Please see [DIM16] for a recent literature review of the

CSP.

4.2 A Multiple Stock Size Cutting Stock Problem

In the classic CSP, all stock items have the same known dimensions, which makes it easier to compute the possible cutting patterns. In certain problems however, stock items come in several different dimensions and these types of problems are known as Multiple Stock Size Cutting Stock Problems (MSSCSP) [WHS07].

An example of such a problem is the forest harvesting problem, as the trees differ in size from each other, sometimes significantly. In this problem, a forest is subdivided into areas called "blocks", with each block having a number of trees to harvest. This partitioning is illustrated in figure 4.1.

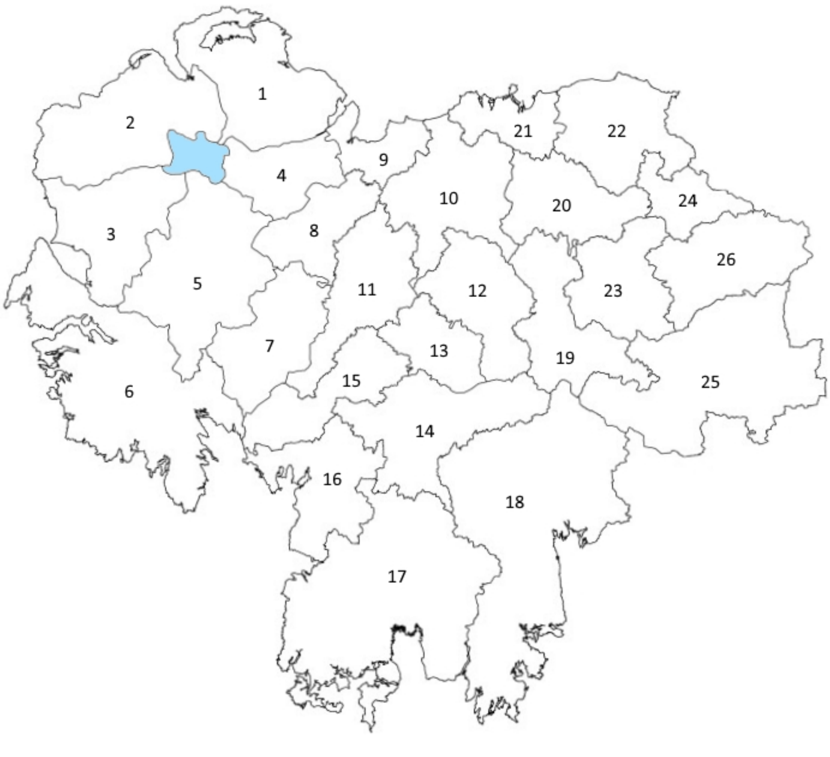


Figure 4.1: Schematic view of a forest partitioned into blocks

There are Q blocks each with market value V_q $(q = 1 \dots Q)$. Each block has R trees, with tree r $(r = 1 \dots R)$ in block q having dimensions $\vec{\sigma}_{qr}$. Each tree can be cut into L different log types, with each log type having a particular monetary value.

This problem was classified in [COP16] as $*/V/D/R$ (see Dyckoff's typology [Dyc90]), where $*$ means any dimensionality, V means that the total amount of items in stock (i.e. the total number of trees) is sufficient to accommodate all the demanded products (hence, only some of the stock will be cut), D means that all large pieces (stock items) are different (in terms of shape) and R indicates many products of few different types are demanded. The feature V (any demand can be fulfilled) requires that the stock items to be cut need to first be selected. Using the more recent typology presented in [WHS07], the problem can be classified as a variant of the MSSCSP in which there is a heterogeneous assortment of large pieces.

In practice, a semi-autonomous harvesting machine (figure 4.2) cuts a tree into logs in order to maximise its total value using an algorithm $\mathcal{A}$, which is typically based on a Dynamic Programming (DP) algorithm. DP is an approach that enables the solution of complex problems by dividing them into a collection of simpler sub-problems (see [ASW$^+$15, BD15] for example). The sub-problems must be sequential and independent, and the problem of cutting a tree stem satisfies these properties since it is recursive (maximize by cutting the first product and then maximizing the cutting of the remainder). Let L be the length of a section of a tree stem measured from the base of the stem, and L_{max} be the total usable length of the stem. If y_k is the length of a short log of type k cut at a distance $L - y_k$ from the base of the stem, and $(r(y_k, L)$ is the associated product value, this recursive relationship can be represented as

$$f(L) = \text{maximum}_{k, y_k \in Y(L)}(r(y_k, L) + f(L - y_k)) \tag{4.2}$$

where $f(0) = 0$ and $0 \leq L \leq L_{max}$ [EDW86].

The semi-autonomous nature of the harvesting machines is due to their construction, as they are hard-coded to produce log types with the highest possible monetary value wherever possible. Thus, we only have indirect control over the cutting of trees via a set of L continuous variables called a weight vector $\vec{w} \in {\mathbb{N}_+}^L$. Each w_l represents the weight (usually the price in $€/m^3$) associated with product type l.

Blocks are sold wholesale, i.e. either a block's trees are completely cut or none of its trees are cut. For each block q, a set of product types $\mathcal{L}$, a vector of tree dimensions $\vec{\sigma}_{qr}$ in the block, and a weight vector $\vec{w}_q$ is passed to $\mathcal{A}$. The result is a product vector $\vec{p}_q \in {\mathbb{N}_+}^L$ showing the total amount of each log type obtained from

Figure 4.2: Semi-autonomous harvesting machine

the block whose trees are cut, denoted by $\vec{p}_q = \langle a_1, \ldots, a_L \rangle$ where a_l represents the volume in m^3 of units of log type l, $(l = 1 \ldots L)$ obtained. Consequently, $\mathcal{A}$ can be represented by the following mapping function for block q:

$$\mathcal{A}(\mathcal{L}, \langle \vec{\sigma}_{q1}, \ldots, \vec{\sigma}_{qR} \rangle, \vec{w}_q) \rightarrow \vec{p}_q \tag{4.3}$$

Solving $\mathcal{A}$ with fixed $\vec{w}_q$ assigns values to the $\vec{p}_q$ variables, and the value obtained from the piece is $\vec{w}_q \cdot \vec{p}_q$. In some applications actual market prices are used as the weights. For this problem however, the $\vec{w}_q$ are manipulated to obtain the desired product yields. This is the only control we have over how tree stems are cut due to the hard-coding of the harvesting machines.

Given a demand vector $\vec{D}$ denoting the desired yield D_l for each product type $l \in \mathcal{L}$, the problem is to determine which blocks should be selected for harvesting, as well as the weight vector to use for each such block, in order to meet the demands while minimising the total value of the harvested blocks to conserve natural resources. There are no restrictions on which subsets of blocks can be chosen, and the trees in a block are either all cut, or none of them are.

There are several approaches that aim to achieve the desired yield D_l in the literature. [Kiv04] and [Kiv06] use genetic algorithms to try to improve the fit between the yields obtained by the harvester and the demand with varying levels of success. [DS12] uses a price-weighted apportionment degree (AD) index to try to improve the fit between output and demand. This approach still prioritises logs with higher value and may not fulfil demand, leading to overproduction of unwanted logs and consequently waste. [MP04] uses flexible variations on the AD to improve the fit between demand and supply, however their approach is not guaranteed to be optimal. [MMB06] provides three mathematical models for bucking to order using a small set of market prices, targeting certain cutting patterns, and using the AD index respectively. [KUN05] compares four different measures to determine the similarity between the demand and output log distributions. None of the four are shown to be superior, even though they can be used in the industry to some extent. [DRF15, DRF17] use the priority list approach where higher value log types are prioritised. This approach also only considers a few cutting patterns which are assumed to be sufficient, although this is not always the case. [SOG89] adjusts the price iteratively, but using only a small set of prices. [Duf80] also vary price but how they do this is not stated.

The analytics-based decomposition approach used in this thesis is a good fit for this problem since a much larger number of prices can be evaluated, thus creating a good approximation of the distribution relating the prices to the products (recall figure 3.1). Also, separating the harvester operation ($\mathcal{A}$) from the rest of the linear program using bilevel formulation allows for the more efficient solution of problems with a large number of blocks. Additionally, the use of analytics approaches presents a new way of solving a real-world bilevel problem.

4.3 Bilevel Reformulation

The above problem can be naturally modelled as a multiple-follower bilevel optimisation problem. This reformulation of the forest harvesting problem as a bilevel problem is novel, and is one of the contributions of this thesis to the literature. Here, the leader's objective is to select a set of blocks to harvest to fulfil demand, while each follower q seeks to harvest its block to get the optimal product vector $\vec{p}_q$, given a price input $\vec{w}_q$.

Define binary variables $h_q = 1$ if block q cuts its raw stock of trees, and product vectors of integers $\vec{p}_{qr}$ to describe the product yields from raw r in block q's stock. V_q is the monetary value of block q estimated by the forest providers. $\vec{\sigma}_{qr}$ are the dimensions of an uncut tree stem r in block q. The problem is thus:

$$
\begin{array}{lll}
\min_{h_1 \dots h_Q, \vec{w}_1 \dots \vec{w}_Q} & \sum_{q=1}^{Q} V_q h_q & \\
\text{s.t.} & & \\
& \sum_{q=1}^{Q} \sum_{r=1}^{R} h_q \vec{p}_{qr} \geq \vec{D} & \\
& & \\
& \text{where each } \vec{p}_{qr} \; (q = 1, \dots, Q) \text{ solves:} & \\
& \vec{p}_{qr} \in \operatorname{argmax}_{\vec{p}_{qr}} \mathcal{A}(\mathcal{L}, \vec{\sigma}_{qr}, \vec{w}_q) & r = 1, \dots, R \\
& \text{s.t.} & \\
& h_q \in \{0, 1\} & q = 1, \dots, Q \\
& \vec{p}_{qr} \in \mathbb{N}^L & q = 1, \dots, Q, \forall r \\
& \vec{w}_q \in [0, 1]^L & q = 1, \dots, Q
\end{array}
\tag{4.4}
$$

This is a non-linear, mixed-integer, bilevel optimisation problem with multiple followers which we call the *Bilevel Cutting Stock Problem with Multiple stock sizes* (BCSPM). It is also large: there might be hundreds of blocks and hundreds of (sampled) trees per block, hence tens of thousands or more follower problems (since $\mathcal{A}$ is evaluated for each r), as well as a large number of product types. It cannot be solved by classical bilevel methods but it could be tackled by evolutionary methods. Metaheuristic approaches (popularly used in industry) such as [MMB04] and [DS90] have been tried, with very poor results obtained.

The model above does not have the strong independence property because all the follower problems corresponding to a block share the same variables. It can however be transformed so that it does, by grouping each block's follower problems into a single problem via new vectors of integer variables $\vec{p}_q$, which model the total yield from each block:

$$
\begin{array}{lll}
\min_{h_1 \dots h_Q, \vec{w}_1 \dots \vec{w}_Q} & \sum_{q=1}^{Q} V_q h_q & \\
\text{s.t.} & & \\
& \sum_{q=1}^{Q} \sum_{r=1}^{R} h_q \vec{p}_{qr} \geq \vec{D} & \\
& & \\
& \text{where each } \vec{p}_q \; (q = 1, \dots, Q) \text{ solves:} & \\
& \vec{p}_q \in \sum_{r=1}^{R} \operatorname{argmax}_{\vec{p}_{qr}} \mathcal{A}(\mathcal{L}, \vec{\sigma}_{qr}, \vec{w}_q) & \\
& \text{s.t.} & \\
& h_q \in \{0, 1\} & q = 1, \dots, Q \\
& \vec{p}_q \in \mathbb{N}^L & q = 1, \dots, Q \\
& \vec{w}_q \in [0, 1]^L & q = 1, \dots, Q
\end{array}
\tag{4.5}
$$

Now the followers are strongly independent: each uses a unique set of variables $\vec{w}_q, \vec{p}_q$ and none of the follower variables are mutually constrained. The decomposition method detailed in Section 3.3 can now be applied.

For each $\vec{w}_q$ a number of feasible solutions ($\vec{W}_{sq}$) are generated. Each follower problem is then solved for the $\vec{W}_{sq}$ using the cutting simulator $\mathcal{A}$, to get corresponding $\vec{P}_{sq}$. Next, k-medoids clustering is applied for each follower q, resulting in the selection of a diverse set of assignments $\vec{P}_{kq}$, together with the corresponding $\vec{W}_{kq}$. The problem can now be formulated as an ILP:

$$\begin{array}{lll}
\min_{h_1 \dots h_Q, \vec{w}_1 \dots \vec{w}_Q} & \sum_{q=1}^{Q} V_q h_q & \\
\text{s.t.} & & \\
& \sum_{q=1}^{Q} \sum_{k=1}^{K} P_{kql} x_{qk} \geq D_1 & l = 1, \dots, L \\
& \sum_{k=1}^{K} x_{qk} = h_q & q = 1, \dots, Q \\
& h_q \in \{0, 1\} & q = 1, \dots, Q \\
& x_{qk} \in \{0, 1\} & q = 1, \dots, Q,\ k = 1, \dots, K
\end{array} \tag{4.6}$$

where $h_q = 1$ indicates that all block q's trees are cut, and $x_{qk} = 1$ indicates that they are cut using weights with index k. If block q is not selected then $h_q = 0$ which forces $x_{qk} = 0$ for $k = 1 \dots K$.

4.4 Small-Scale Evaluation

To empirically study the performance of the decomposition approach, real data from an industrial partner was used. The application is commercially sensitive so the identity of this partner will be kept confidential. The total volume of the raw material analysed was $1191.3m^3$ for 8 blocks ($Q = 8$) with each block's trees partitioned into a maximum of 4 different types of products.

20 instances of random demands were generated and solved. Monte Carlo simulation and clustering (Algorithm 4) were done in Java and R (using the `CLARA` package) [MRS+17] respectively on a 3.0 GHz Intel Xeon Processor with 8 GB of RAM. To solve the decomposed integer linear programming model, the CPLEX 12.6 solver was used with a time cut-off of 1 hour.

To approximate the unknown multivariate distribution of the follower problems, $10,000$ random weight vectors ($\vec{W}_{sq}, S = 10,000$) were generated for each block. The same number of corresponding cutting patterns ($\vec{P}_{sq}$) was obtained by ap-

Table 4.1: Clustering times.

	Time (sec)							
Block	k = 25	k = 50	k = 75	k = 100	k = 125	k = 150	k = 175	k = 200
0	5.13	14.47	31.40	56.80	107.00	158.85	229.90	310.47
1	4.94	15.68	35.66	59.88	95.85	156.30	234.19	313.49
2	4.99	14.62	27.59	56.59	94.38	154.96	235.22	313.99
3	5.06	17.59	36.63	62.06	102.29	167.75	266.71	356.86
4	5.17	16.04	33.82	60.64	99.78	164.89	240.40	331.72
5	5.43	16.83	33.04	63.48	95.85	171.37	255.47	351.97
6	4.99	15.77	33.21	61.18	98.10	163.63	241.04	327.87
7	4.83	16.27	34.37	67.03	95.04	168.46	251.94	344.71
Total Time	40.54	127.27	265.72	487.66	788.29	1306.23	1954.87	2651.08

plying the algorithm $\mathcal{A}$. For 8 blocks, this process resulted in total of $80,000$ cutting patterns. The total time required for generating all the cutting patterns was 2 hours and 30 minutes. Next, k-medoids clustering was applied to cover the distribution.

To evaluate the effect of the number of medoids used to cover the distribution, k was varied from 25 to 200 in steps of 25. The clustering times can be found in Table 4.1. Figure 4.3 also shows the total clustering times (for all blocks) as k increases. It can be seen that the clustering times increase exponentially from 40.54 seconds for $k = 25$, to $2,651.08$ seconds ($\sim$ 44 minutes) for $k = 200$. Thus for real-life problems it is very important to select an appropriate value of k, especially for on-line applications.

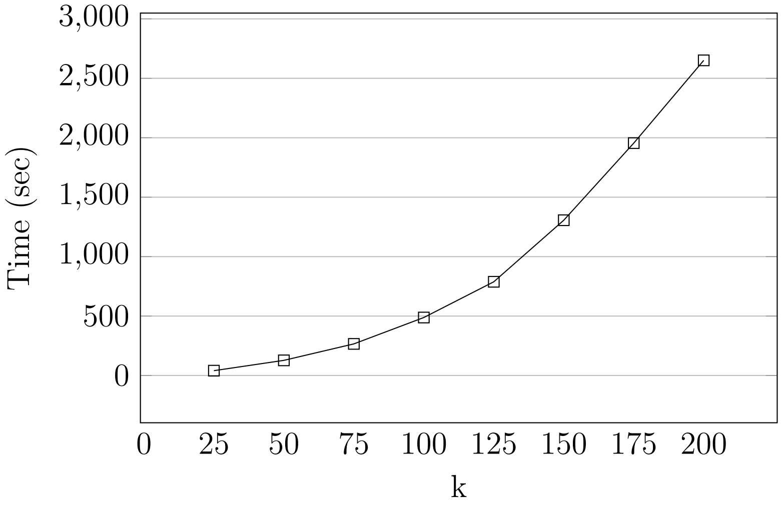


Figure 4.3: Total clustering times

Once all the medoids were obtained, (4.6) was solved for 20 instances of random demands. To study the effect of k on the decomposed model, a relaxed ILP model with the objective of calculating the lower bound of optimality of the instances was also implemented. In this variation, any linear combination of cutting patterns is considered feasible, however the combination selected may not be feasible for the real life problem. For this reason, this measurement is a lower bound of optimality since the solutions obtained from equation (4.6) are greater than this bound. The lower optimality bound is very useful since it allows the search for a better solution to be stopped once the bound has been reached.

Figure 4.4 shows the percentage difference between the solutions obtained and the lower optimality bound. It can be observed that as k increases, the percentage difference decays exponentially. After a "saturation" point (located approximately at $k = 125$) an increase in the value of k does not improve the solution quality. At this point, the percentage difference between the solutions and the lower optimality bound was $\sim 0.4\%$, which indicates that close-optimal solutions were found.

Economically speaking, the amount of raw material that was required to satisfy the demand instances when using $k = 25$ was on average almost €800 more expensive than when using $k = 125$.

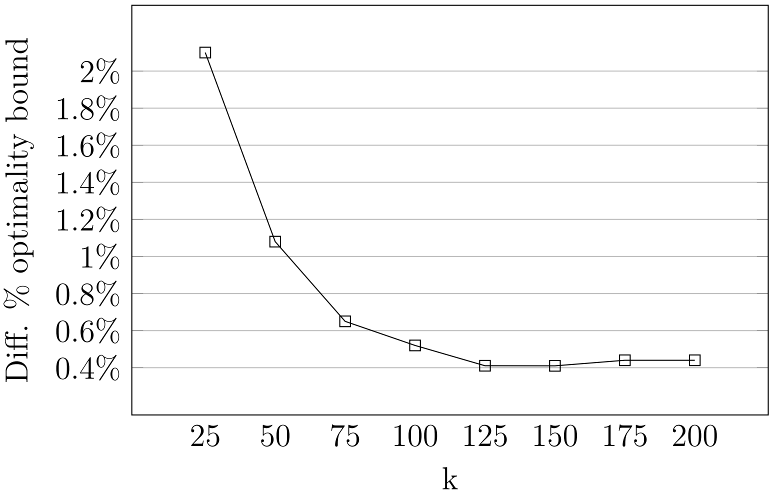


Figure 4.4: Percentage Optimality Difference

Table 4.2 and figure 4.5 (log scale) show the mean times for solving the 20 instances for the values of k used. Note that these times also increase in a non-linear fashion, from a solution time of 1.506 seconds for $k = 25$ to 407.475 seconds (~ 7 minutes) for $k = 200$. It can also be seen that there is a correlation with the saturation point above with the computation times for solving the ILP model. For higher values of $k = 125$, there is very little increase in the computation time.

Table 4.2: Mean ILP solution times.

k	Mean Time (sec)
25	1.506
50	188.454
75	369.16
100	240.120
125	410.091
150	398.914
175	411.180
200	407.475

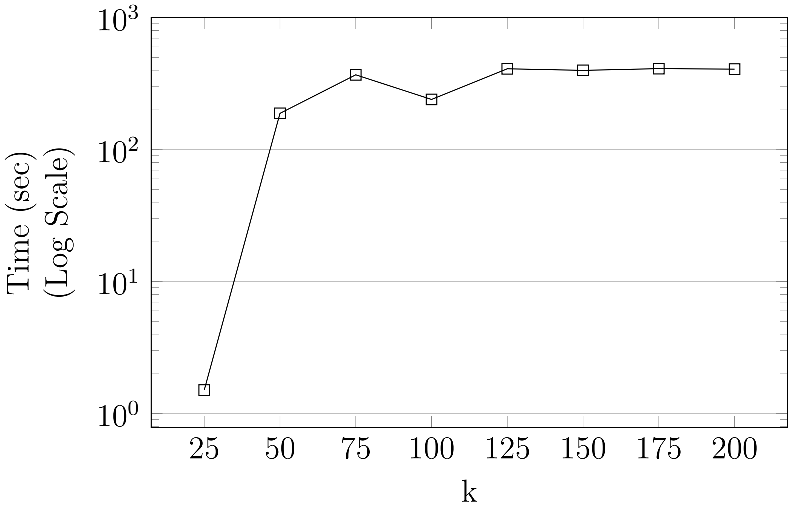


Figure 4.5: Mean ILP solution times

4.5 Large-Scale Evaluation and Comparison of Analytics Approaches

A larger evaluation was carried out on a different forest with 263 blocks ($Q = 263$) and the stock partitioned into a maximum of 11 product types. The total volume of raw material was $6149.781\,m^3$, with the data obtained from the industrial partner. $1,000$ random weight vectors $\vec{W}_{sq}$, ($S = 1000$) were generated for each block, giving a total of $263,000$ cutting pattern vectors $\vec{P}_{sq}$. The total time for generating these was approximately 18 hours and 17 minutes. The total clustering time for all blocks with $k = 125$ was 31 hours. 12 different instances of random demands were solved. In 6 of these instances, the demand for product types was in the range $[0, 300]\,m^3$ (low demand), while the remaining 6 had demands in the range $[300, 600]\,m^3$ (high demand). ILP solution times were nominal, taking less than 5 seconds for all instances evaluated.

Due to the high clustering times, self-organising maps (SOM) (Algorithm 5) were used an alternative to k-medoids. The SOM experiments were done using the Java Kohonen Neural Network Library (JKNNL) [RHW06]. For the small problem ($Q = 8$), the SOM was trained on the $80,000$ cutting pattern vectors $\vec{P}_{sq}$, using a varying number of neurons $N \in \{25, 50, 75, 100, 125, 150, 175, 200\}$ arranged in a grid topology.

A chart comparing the increase in clustering times for both k-medoids and SOM approaches is shown in figure 4.6.

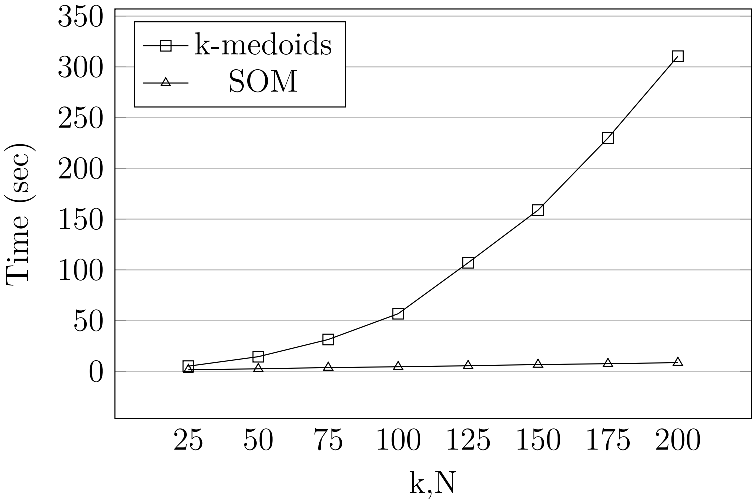


Figure 4.6: Scalability of clustering approaches for one block

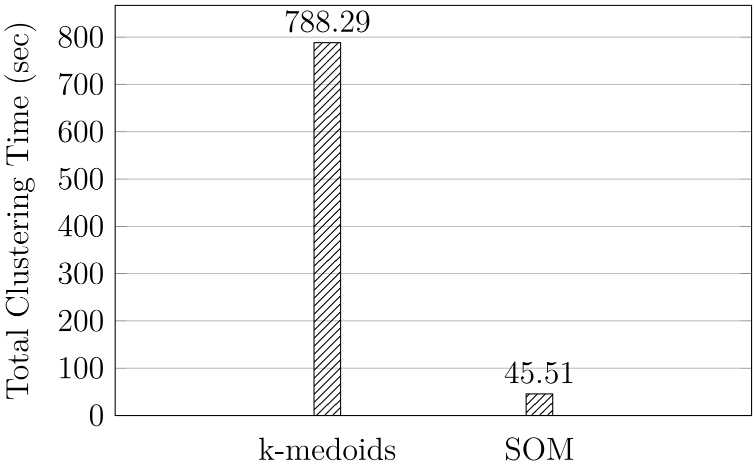


Figure 4.7: Comparing clustering times for a small problem instance ($Q = 8$), $k = N = 125$

In terms of scalability, it can be seen in figure 4.6 that using the SOM approach offers a significant improvement in data reduction time, which makes it more useful than k-medoids for large problems. In figure 4.7, when $k = N = 125$ for

$Q = 8$, the total clustering times are 788.29 and 45.51 seconds for the k-medoids and SOM respectively. When $Q = 263$, the difference in clustering and map training times is even more striking and is better visualised using a logarithmic scale (figure 4.8). Here, the the total times taken are 111898.35 seconds (31 days) and 226.72 seconds (4 minutes) for the k-medoids and SOM respectively.

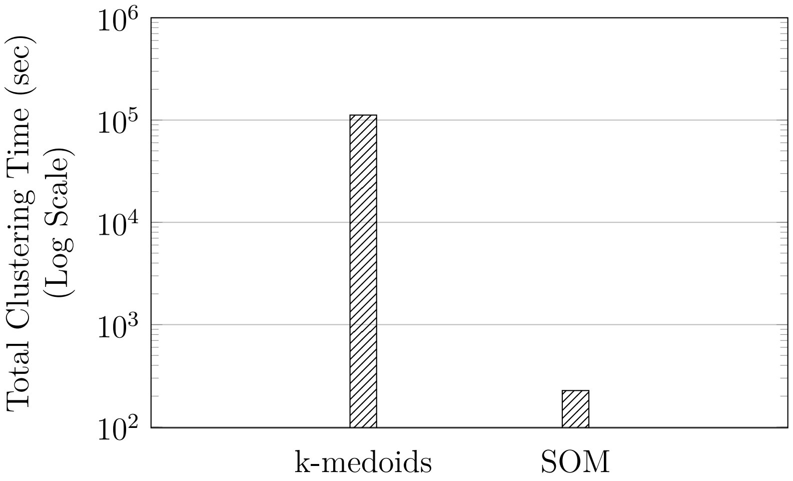


Figure 4.8: Comparing clustering times for a large problem instance ($Q = 263$), $k = N = 125$

A third approach, pure Monte Carlo sampling (MC) (Algorithm 2) was also used. In this case, instead of generating, for example, 10,000 random patterns and then clustering them to get (e.g.) 125 representative cutting patterns, only the exact number of patterns needed , i.e. 125 are generated. The model (4.6) was then solved using this random set of patterns, and the solution times and qualities of all 3 methods – k-medoids, SOM and MC – were compared.

4.5.1 Comparing Solution Quality

The 12 different instances of random demands above and two criteria (the value of the objective function and the number of blocks harvested) were used to evaluate the effectiveness of each of these methods. For both criteria the smaller these values are, the better the solutions.

Figures 4.9 and 4.10 show the average solutions for the low demand instances for the case where $Q = 8$. In terms of the value of the objective function, it can be seen that both the k-medoids and the SOM give the lowest objective functions. Considering the number of blocks harvested, the performances of the k-medoids and SOM approaches are once again similar, with both selecting the least number

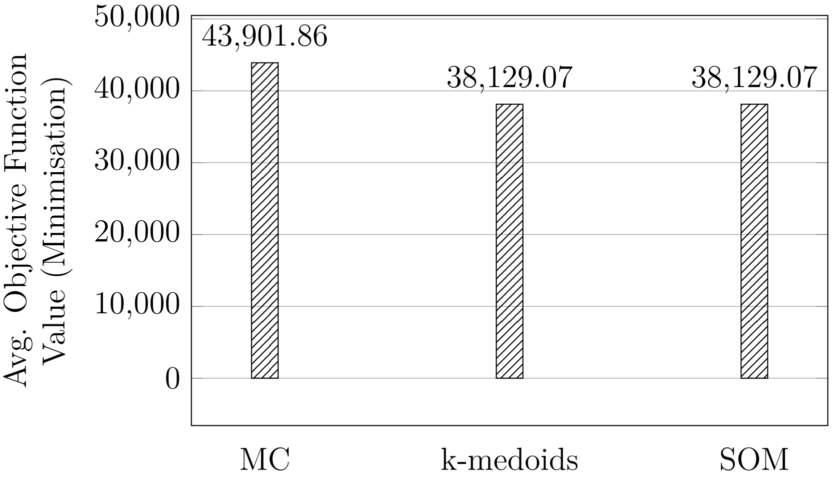


Figure 4.9: Comparison of Approaches - Average Objective Function Value ($Q = 8$, low demands)

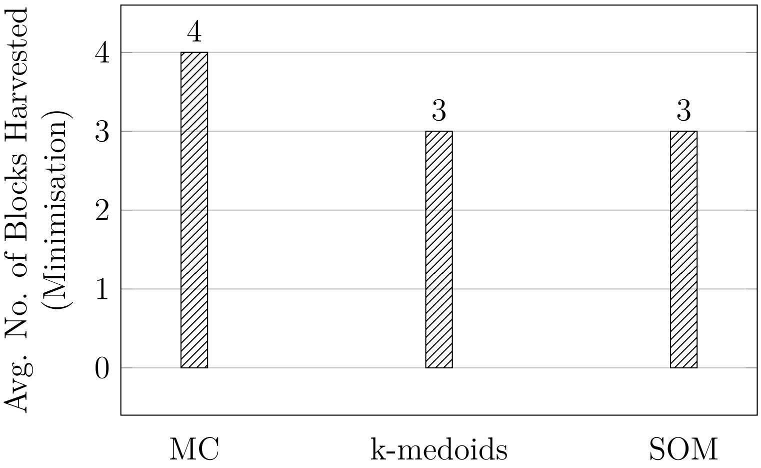


Figure 4.10: Comparison of Approaches - Average Number of Blocks Harvested ($Q = 8$, low demands)

of blocks to harvest. As expected, with both of these criteria, pure MC performs the worst.

For the high-demand instances, the difference between the objective values is small, but k-medoids and SOM are still better than pure MC (figure 4.11). As above, a smaller number of blocks is harvested when using k-medoids and SOM (figure 4.12). In these cases, it is beneficial to generate a large set of patterns and cluster them, even in cases where the improvements are small.

For the larger problem where $Q = 263$, figure 4.13 shows a comparison of the approaches for the low-demand instances. In this case, in terms of the objectives, all methods perform equally, with only a little difference ($\sim 0.5\%$) between them on average. Considering the number of blocks harvested 4.14, the SOM method

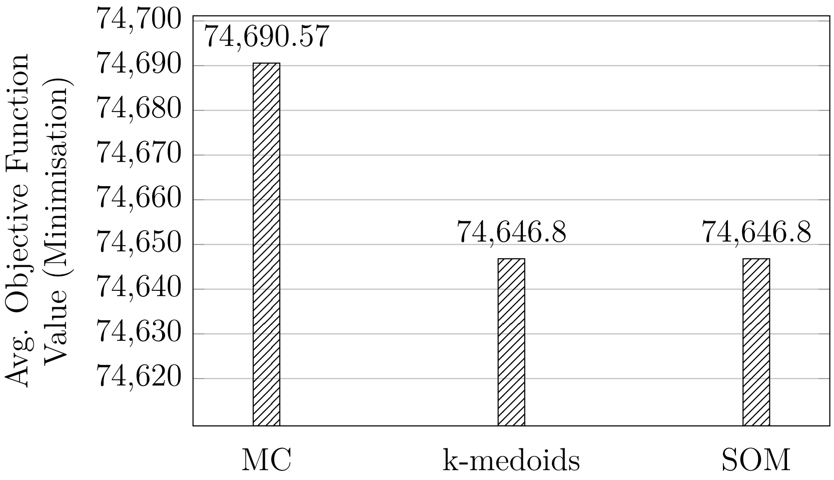


Figure 4.11: Comparison of Approaches - Average Objective Function Value ($Q = 8$, high demands)

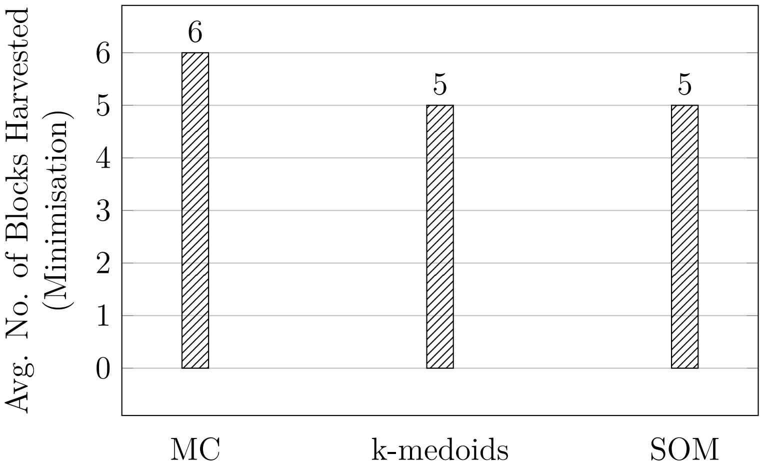


Figure 4.12: Comparison of Approaches - Average Number of Blocks Harvested ($Q = 8$, high demands)

performs better, with k-medoids being the worst. For the high-demand instances, there is again only a little difference between all three approaches in terms of the objectives (figure 4.15). Concerning the number of blocks harvested however, the SOM approach is the clear winner with 5 less blocks selected (4.16). This is because it produces a set of cutting patterns which allow the ILP to choose a different (and smaller) selection of blocks to meet the demands. In most cases, the MC approach does not perform as well as the others in terms of solution quality.

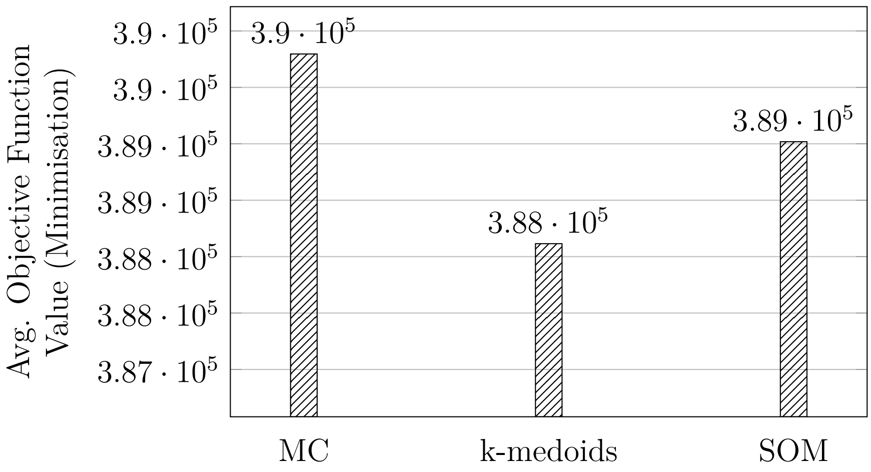


Figure 4.13: Comparison of Approaches - Average Objective Function Value ($Q = 263$, low demands)

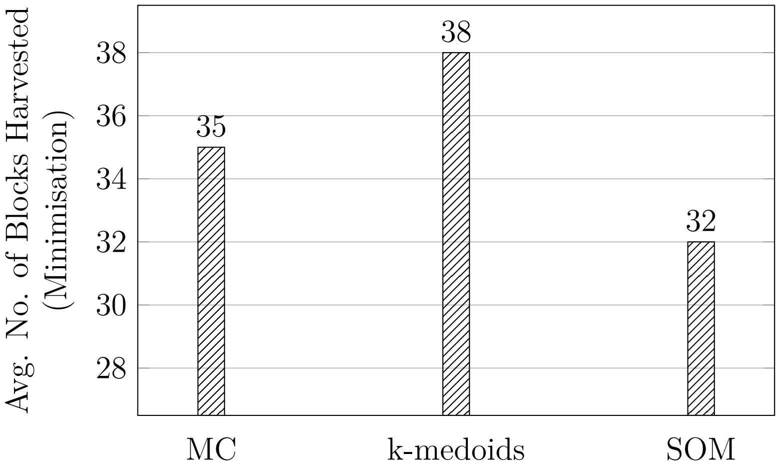


Figure 4.14: Comparison of Approaches - Average Number of Blocks Harvested ($Q = 263$, low demands)

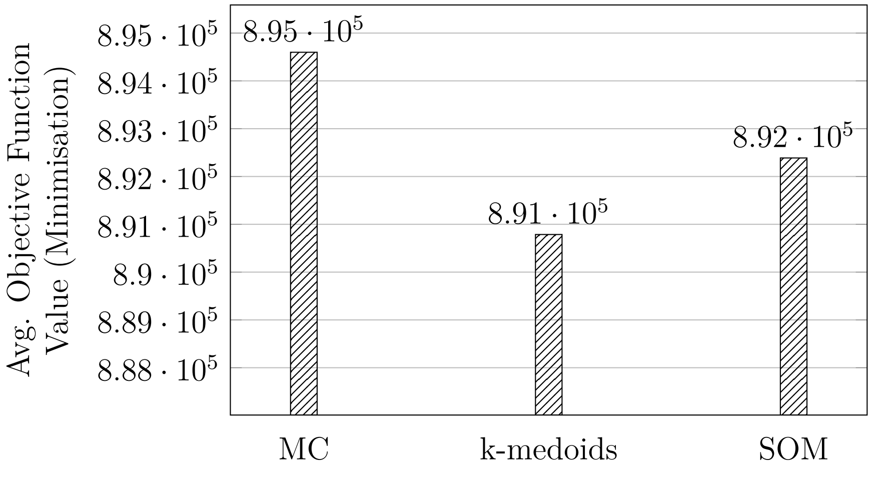


Figure 4.15: Comparison of Approaches - Average Objective Function Value ($Q = 263$, high demands)

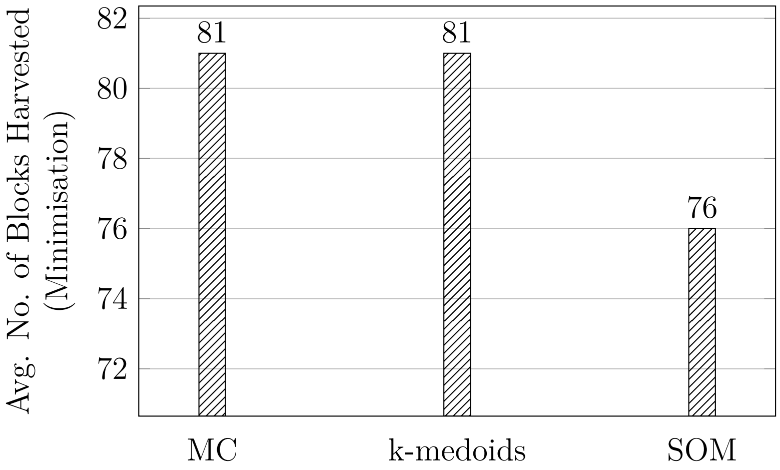


Figure 4.16: Comparison of Approaches - Average Number of Blocks Harvested ($Q = 263$, high demands)

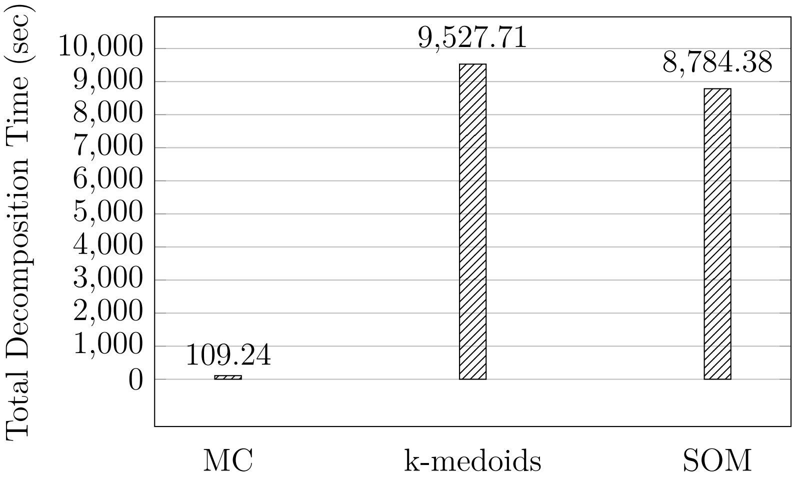


Figure 4.17: Comparison of Decomposition Times (Monte Carlo Sampling + Clustering), $Q = 8$

4.5.2 Comparing Decomposition Times

Figure 4.17 shows the total decomposition times (i.e. sampling time + data reduction time) taken by each of the three approaches to generate the number of cutting patterns needed (i.e. 125) when $Q = 8$. It can be seen that MC takes much less time (only 109.24 seconds). Obviously, the fact that only a small number of patterns are generated and there is no data reduction step speeds it up considerably. This is also the case for $Q = 263$ (figure 4.18). It can again be seen that the MC approach takes only about 6 hours in total, while the k-medoids and the SOM approaches take about 80 and 49 hours respectively.

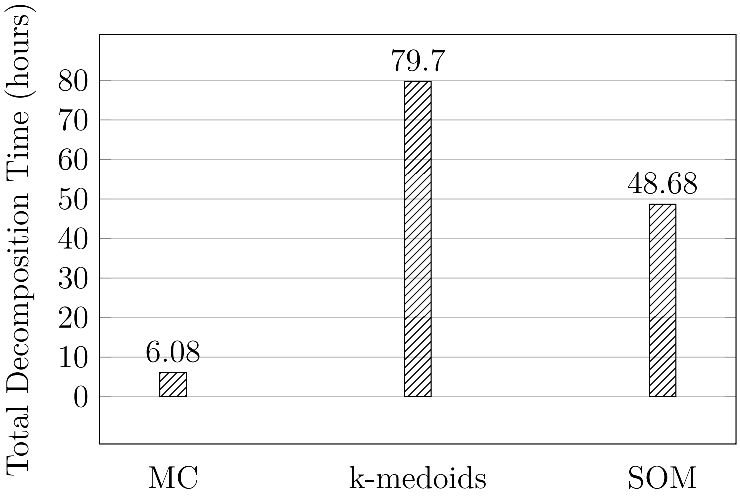


Figure 4.18: Comparison of Decomposition Times (Monte Carlo Sampling + Clustering), $Q = 263$

4.6 Data Analysis

In most cases, although there is an advantage in sampling a large amount of patterns and clustering them to get a small but highly representative set, the benefits are not commensurate with the time and effort spent. In several cases, using the MC approach produced results very close to those of the other two approaches. In order to find out why this is the case, some analysis was done on the data produced by sampling, i.e the $\vec{P}_{sq}$.

This data analysis was in the form of hierarchical clustering using the `hclust` functions in the R `stats` package. Hierarchical clustering is a cluster analysis method that seeks to build a hierarchy of clusters within clusters. The results of hierarchical clustering are often represented as a dendrogram, which is a tree diagram that represents the individual observations (in this case, the cutting patterns) on the horizontal axis, and the distance/dissimilarity between the merged clusters on the vertical axis. Agglomerative hierarchical clustering (which starts at the bottom by considering each observation as its own cluster (height 0) and merges clusters as it ascends the hierarchy) was applied.

4 blocks each from each problem size were selected, and hierarchical clustering was applied in order to visualise the data. For the small problem ($Q = 8$) the blocks selected were blocks 1, 3, 5 and 7. Blocks 0, 39, 173 and 253 were selected for the large problem ($Q = 263$). In the case of the large problem, blocks that were as different from each other as possible were selected.

Figures 4.19 to 4.22 show the dendrograms of blocks 1, 5, 39 and 253. It can be seen from the figures that there is a relationship between the number of log types (L) and the number of clusters, with the clusters becoming less well defined as the number of log types increases. There are n large natural clusters for instances with n types of logs, i.e. 4 large clusters for 4 log types, and 11 large clusters for 11 log types.

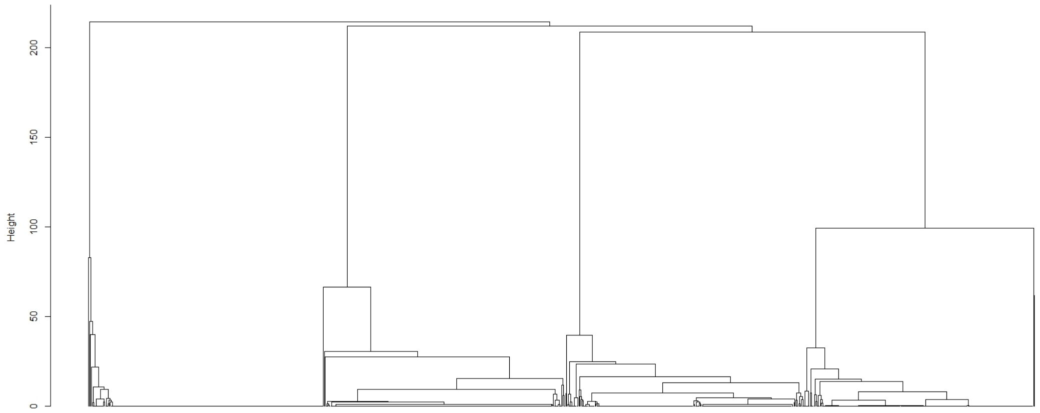


Figure 4.19: Block 1 dendrogram showing natural clusters ($Q = 8$, $L = 4$)

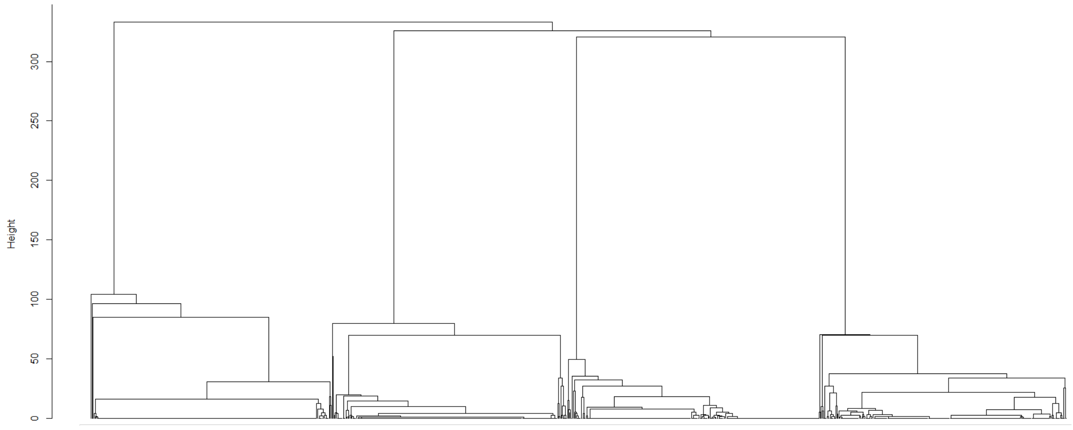


Figure 4.20: Block 5 dendrogram showing natural clusters ($Q = 8$, $L = 4$)

All of the 8 blocks analysed showed this pattern. The occurrence of such a small number of natural clusters implies for this problem, that it may not be necessary to generate a very high number of random cutting patterns in order to represent all the clusters. Note that the likelihood that all the clusters are represented tends to converge to 100% when the number of randomly generated cutting patterns increases.

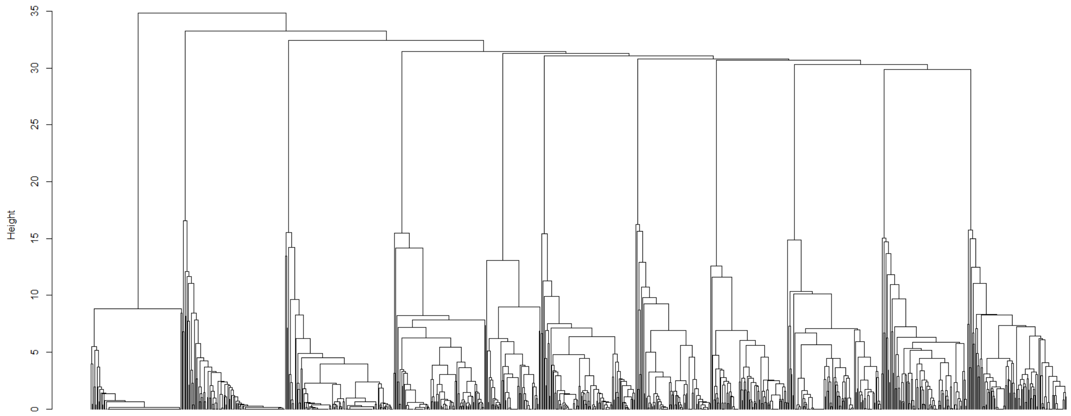


Figure 4.21: Block 39 dendrogram showing natural clusters ($Q = 263$, $L = 11$)

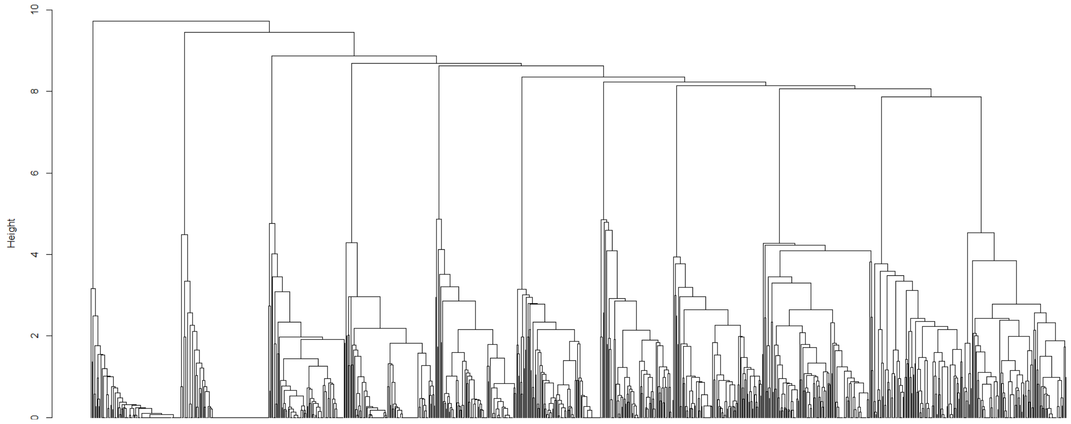


Figure 4.22: Block 253 dendrogram showing natural clusters ($Q = 263$, $L = 11$)

4.7 Conclusions

In this chapter, a large-scale MSSCSP with applications in the forestry industry was considered. This problem was reformulated as a large-scale multiple-follower bilevel problem and solved using the decomposition method. Evaluation on a small-scale problem showed that up to a point, increasing the number of clusters got the solution to about 0.4% from the optimal.

For the large-scale problem, an inordinate amount of time was required using k-medoids, so self-organising maps and pure Monte Carlo sampling were used in the decomposition step. Results showed that although the solutions using k-medoids were slightly better than the other two approaches, this improvement was not commensurate with the amount of time taken. Some analytics using hierarchical clustering revealed the occurrence of natural clusters in the data, leading to the conclusion that for this particular application, the time-intensive step of clustering may not be necessary, especially in time-sensitive cases.

Due to the presence of natural clusters in the data, as well as the fact that there were only a few log types, it was seen that there was no real need to apply clustering. However, applications in which there are many non-uniformly distributed local optima will benefit from taking a large number of samples and clustering.

Chapter 5

Combining the Cutting Stock Problem with Dynamic Harvester Routing

5.1 Introduction

As stated in Chapter 4, the forest harvesting problem is a Multiple Stock Size Cutting Stock Problem (MSSCSP). In the typology of [WHS07], this problem is classified as a variant of the MSSCSP in which there is a heterogeneous assortment of large pieces. An additional factor with this MSSCSP is that there is some uncertainty present. This case, in which there is a heterogeneous assortment of large pieces of possibly unknown dimensions, occurs in population harvesting (see [GH89]), as it is impossible to measure all possible members of the population, and the size of the members change with time.

The forest harvesting problem, is an example of such a case due to the large number of trees of varied sizes, combined with the continuous growth of the trees. It is also impractical to measure the dimensions of every tree in each block in the forest, so in practice, only a sample of each block is measured, and this measurement is extrapolated to the whole forest. These measurements ($\vec{\sigma}_{qr}$ in Chapter 4) are then used to harvest the forest. Even if the sampled stock is a good representation of the whole population (which is not always the case), the results of the harvesting in real life will not be the same as those from the simulation phase.

5.2 Data Description and Analysis

An example of the above discrepancy is shown in figures 5.1 to 5.8. Data from 8 blocks was obtained pre- and post-harvest. The following plots show the density distributions of the diameters of the trees at different heights from the ground for these 8 blocks. The first height at which the diameters are measured is 13 decimetres (or 1.3 metres) above the ground. This height is known in the industry as Diameter at Breast Height (DBH). Subsequent measurements are taken every 70 decimetres above this height. The second measurements are thus taken at DBH+70, the third are taken 140 decimetres higher than the DBH (at DBH+70 × 2), and finally, the last measurements are taken 210 decimetres higher than the DBH (at DBH+70 × 3). Few trees are greater than (DBH+70 × 3) in height. For all plots, the black lines represent the sample data (pre-harvesting) while the red lines represent the real data (post-harvesting).

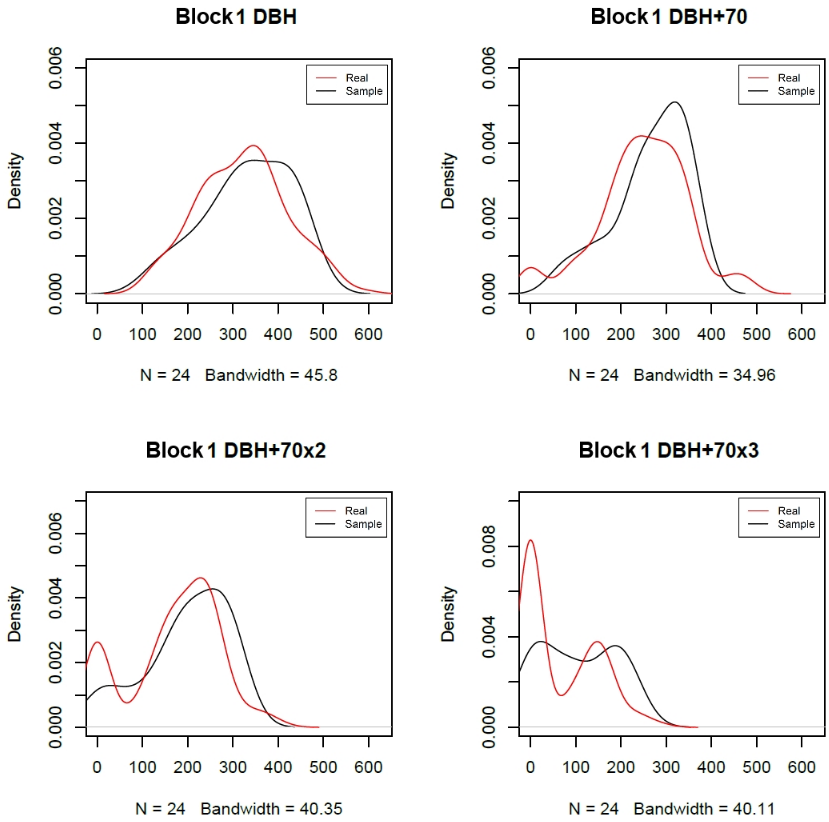


Figure 5.1: Block 1 density plots

In order to numerically measure the differences between the pre- and post-harvest distributions, the Kolmogorov-Smirnov test was applied to each plot. This test makes a null hypothesis that the distributions are equal, with the similarity de-

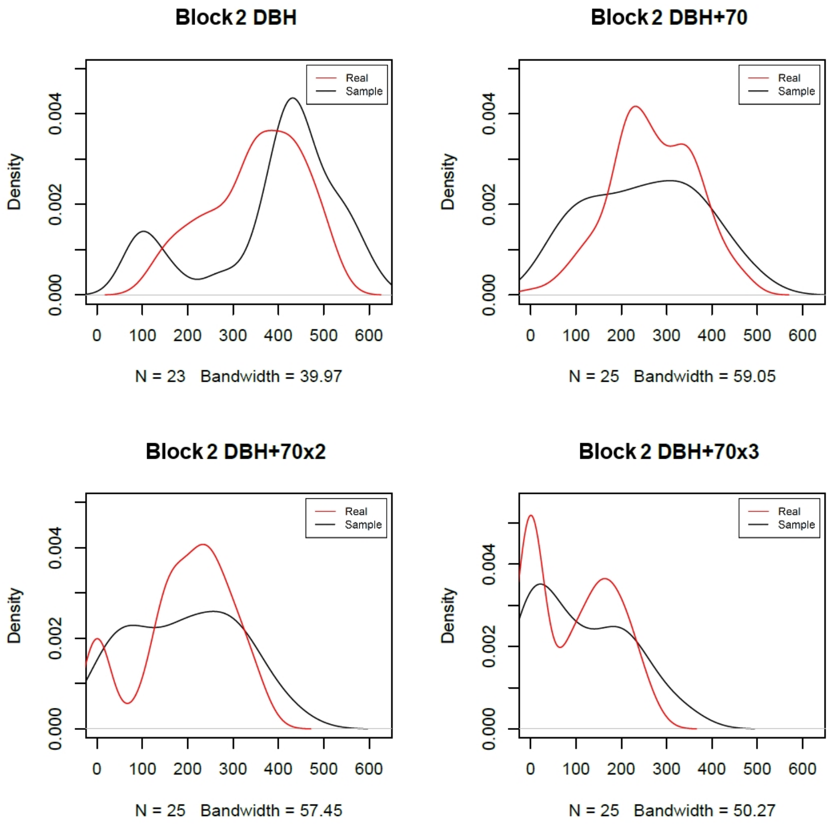


Figure 5.2: Block 2 density plots

termined by the p-value. Table 5.1 shows the p-values. If the p-value is lower than 0.05, the null hypothesis is rejected (values in red). Thus, higher p-values indicate higher similarity.

Table 5.1: p-values

Block	DBH	DBH+70	DBH+70 × 2	DBH+70 × 3	Average
1	0.6470	0.3929	0.2146	0.0137	0.4182
2	0.0360	0.2921	0.4361	0.6092	0.2547
3	0.3800	0.2921	0.4361	0.6092	0.3694
4	0.6760	0.9989	0.0092	0.0003	0.5614
5	0.4410	0.5943	0.4576	0.9188	0.4976
6	0.0087	0.0016	0.0002	3.75E-14	0.0035
7	0.3620	0.5816	0.0119	0.0119	0.3185
8	0.2050	0.1717	0.0684	0.0003	0.1484

This is the motivation for presenting a dynamic and reactive approach in which the amounts of log products targeted for harvesting are re-calculated after harvesting each block. This ensures that the obtained solutions reflect the reality of the forest. In the literature, ILP approaches for such problems are "static" since they only compute a single solution and do not consider that the expected

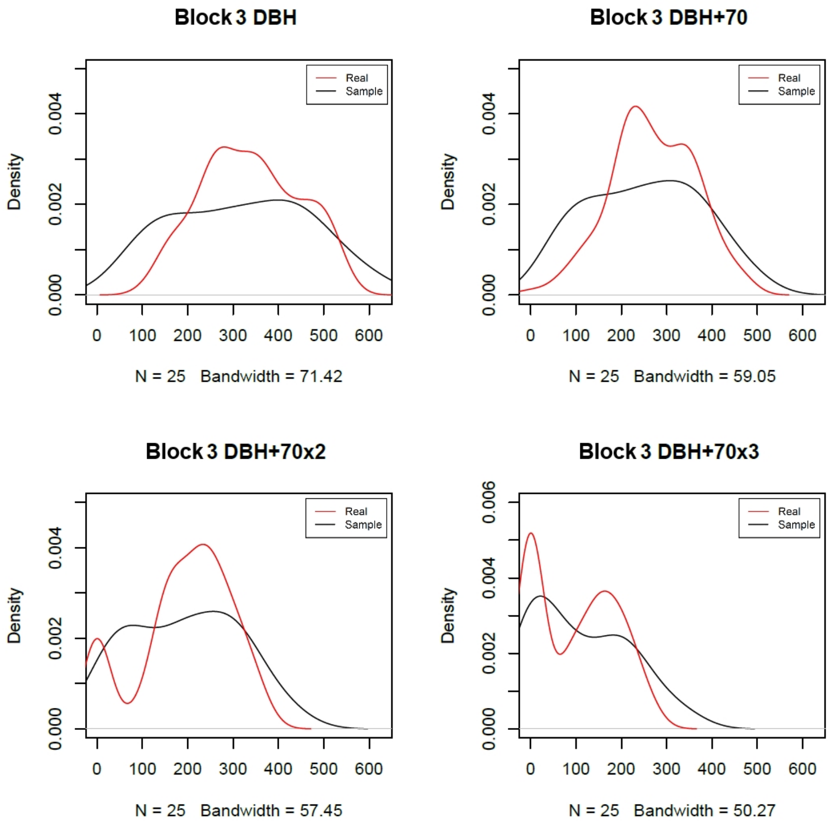


Figure 5.3: Block 3 density plots

and obtained amounts will differ in practice. As far as is known, there are no dynamic approaches in the literature that use ILP for addressing this uncertain problem, only meta-heuristic algorithms. Additionally, the cutting stock problem and vehicle routing problem (VRP) are combined in order to take into account the routing of the harvesters from one block to another, based on the amount of demand still to be fulfilled.

This dynamic approach was evaluated for a forestry instance provided by an industrial partner, and results show that given a cut-off time, the approach leads to a more efficient harvesting of the forest than standard approaches.

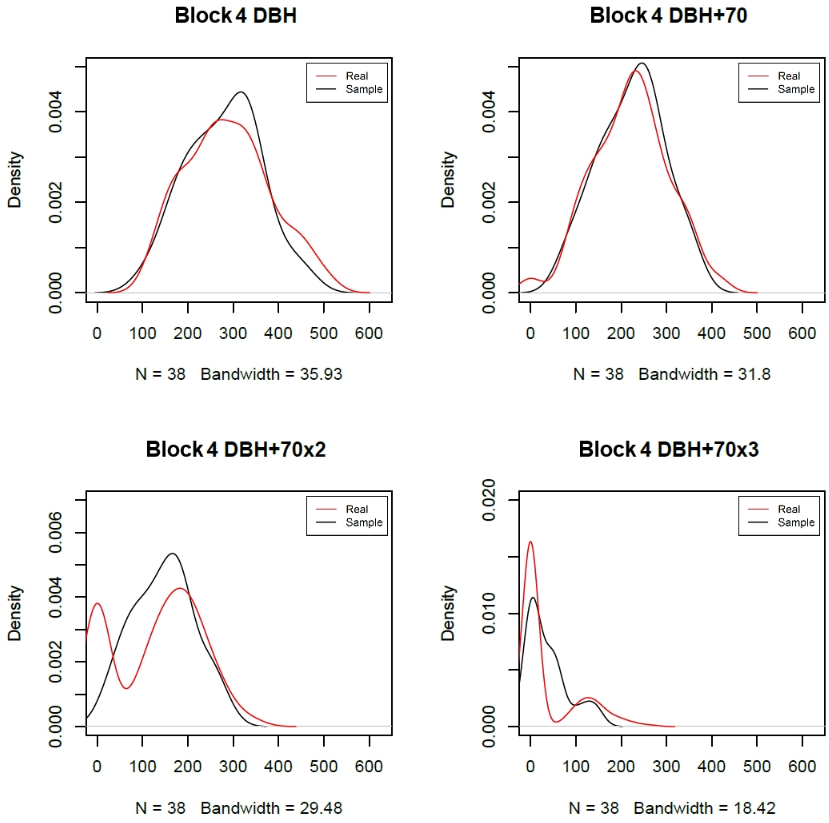


Figure 5.4: Block 4 density plots

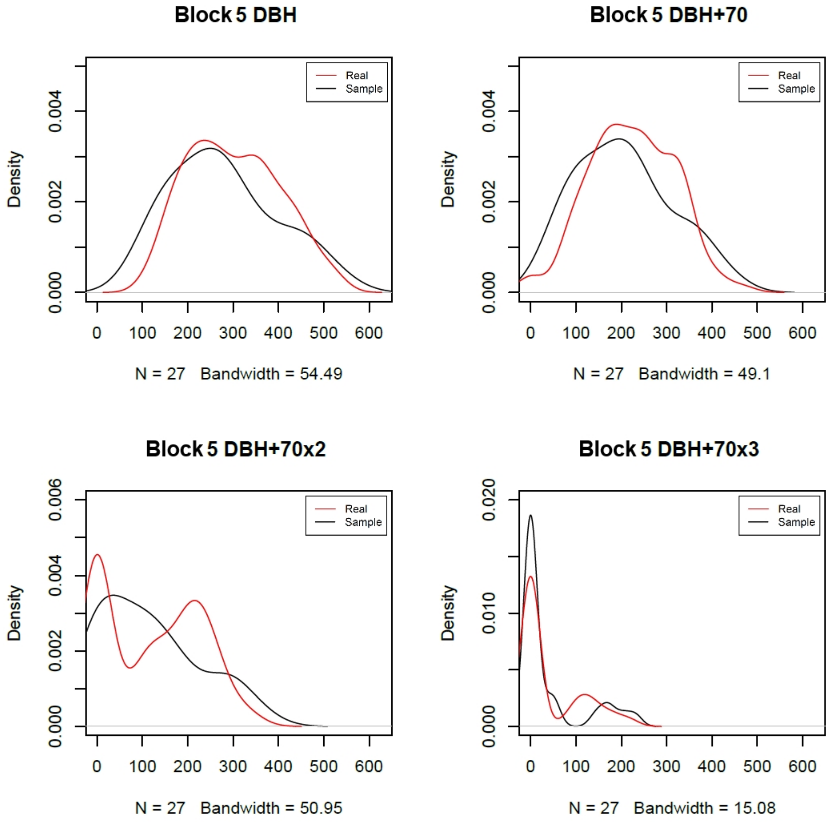


Figure 5.5: Block 5 density plots

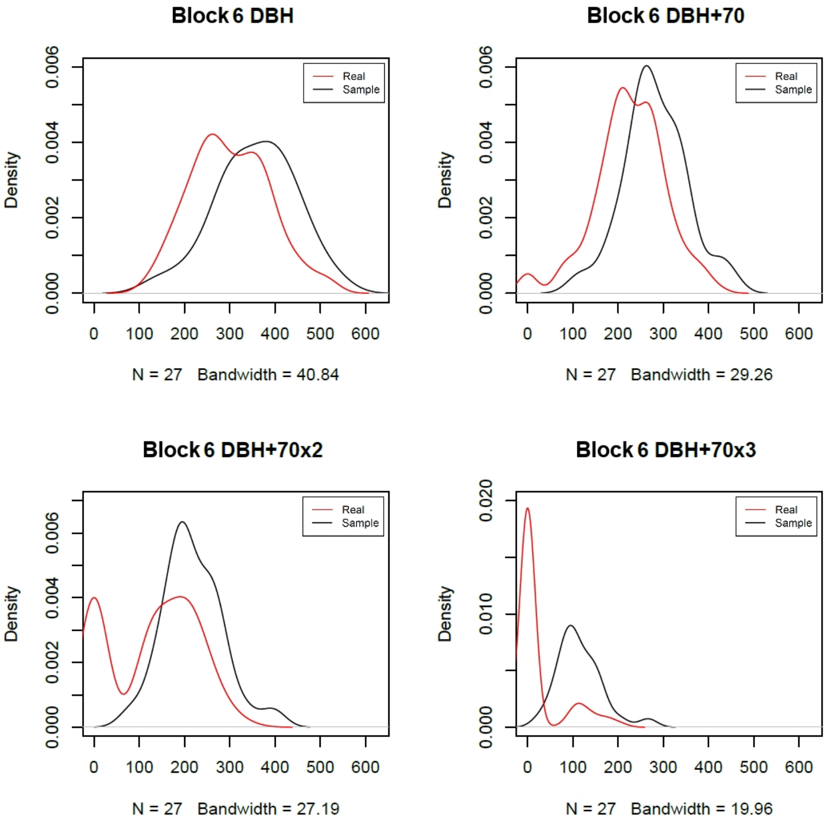


Figure 5.6: Block 6 density plots

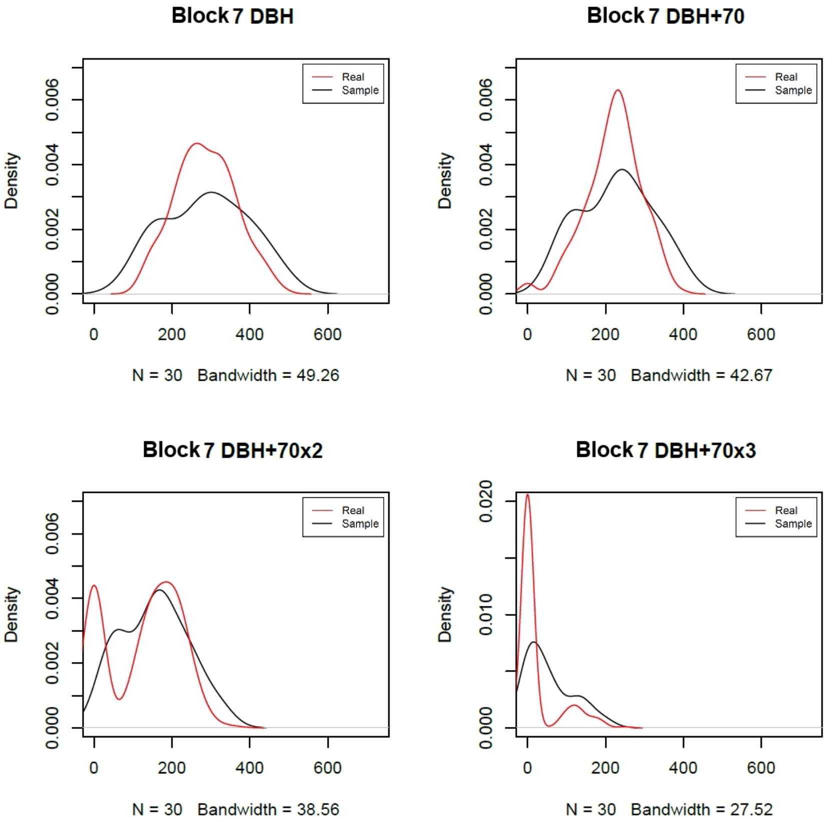


Figure 5.7: Block 7 density plots

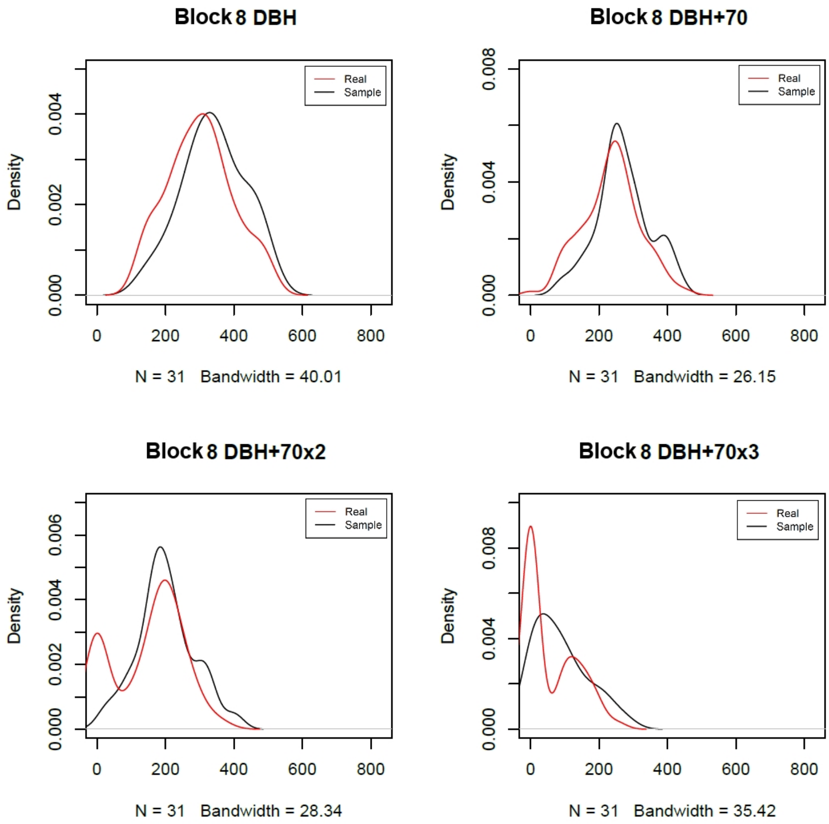


Figure 5.8: Block 8 density plots

5.3 Literature Review

A selection of the literature associated with MSSCSP, and the VRP in agriculture are given in sections 5.3.1 and 5.3.2 respectively.

5.3.1 MSSCSPs

There are several approaches in the literature for solving MSSCSPs using either exact methods such as those in [BS02] and [AdC08], heuristic methods in [PA09], or a combination of the two in [HSB12]. When looking at MSSCSPs related to the problem in Chapter 4, some of the methods are Linear Programming (LP) approaches but do not address the uncertainty of the MSSCSPs. There are meta-heuristic approaches that deal with this uncertainty, and they are discussed in Section 5.6.

The importance of considering uncertainties regarding the stock dimensions and dealing dynamically with them is evinced by the authors of [HCL12]. There, the authors deal with uncertain amounts of products types. However, in the MSSCSP under consideration, the uncertainty comes from the real dimensions of the stock (since only a sample of them is available). This fact, precludes the use of the technique in [HCL12] (precomputing patterns and on-line selecting them) because patterns cannot be computed for raw materials whose dimensions are unknown.

In [DRF17], the authors also present an LP approach for solving a similar MSSCSP but without dynamism. They also use a different algorithm, which uses a priority list of product types instead of weight vectors, to generate the cutting patterns. This is a disadvantage because most of the cutting machines in the industry work with the aforementioned cutting algorithm $\mathcal{A}$ (equation 4.3) that cuts the materials according to the weight vector. While the stock items are sometimes cut to maximize the market value of the products, at other times the goal is to satisfy the demand for the products (which sometimes comes at the expense of their market values). Furthermore, [DRF17] only handle 16 global cutting patterns, while the decomposition approach of Chapter 4 can handle hundreds of cutting patterns. The other improvement of the decomposition approach over [DRF17] and also [PFCO15] is that it can be used to address the dynamic on-line cutting process, i.e. targeted amounts are re-calculated and solutions are re-computed at the end of each cutting stage. Moreover, these two previous works do not compare themselves with a previous and popular meta-heuristic approach. A comparison of the proposed reactive algorithm with the meta-heuristic is carried out, and the results are presented in Section 5.6.

[CWOF16] presents a different approach since it is proactive. It considers the uncertainty in the amounts of products that are expected to be obtained. In order to apply such a technique however, it is necessary to know the range of values for each of the amounts expected. Experience with the industry shows that these data might be unknown, thus leading to the development of a reactive approach, which does not need to take this information into account.

Another way of addressing a CSP (without considering dynamism) is to use some form of column generation. In this approach, the CSP is first solved using a subset of the global product patterns for a fixed number of raw material pieces with similar dimensions. New patterns are then introduced by solving an auxiliary optimisation problem, and the process is repeated iteratively until completion. The column generation approach was originally applied to the standard CSP in [GG61] and [GG63], and was used in the forest harvesting problem by [EDW86], [LG97], and [MB86]. However, this technique cannot be applied since, as previously mentioned, for this problem it is not possible to pre-compute the patterns of stock with unknown dimensions.

Another disadvantage is that in real-life applications, applying a specific pattern to each raw material piece results in an increase in time and costs due to the frequent switching of patterns. This frequent switching also creates an additional difficulty for workers, whether human or mechanical. As a result of this, the solutions produced by column generation are usually not useful in practice, as noted by [SOG89] and [Lar99]. The above reasons preclude the use of column generation for the problem under consideration.

Besides the LP approaches mentioned earlier, a meta-heuristic approach was also developed earlier for MSSCSPs (see [MMB04] and [COW16]). The meta-heuristic approach is a Simulated Annealing Like Algorithm (SALA) called Threshold Accepting Algorithm (TA) [DS90]. This meta-heuristic algorithm iteratively generates new weight vectors (by making local changes) that are mapped by $\mathcal{A}$ into global cutting patterns. Its objective is to reduce the difference between the percentages of product types obtained by a pattern, and the percentages demanded. In [MAA06] a dynamic version of the meta-heuristic is presented, in which the targeted demands are re-calculated after cutting each stock subset. Their algorithm however randomly selects the next stock subset to be cut, while the proposed algorithm selects the most suitable stock subsets to be cut and their order by combining the CSP with the VRP. It is believed that this is one of the key reasons that the proposed reactive approach out-performs the algorithm in [MAA06].

5.3.2 VRP in Agriculture

The classic VRP is well known and, at its simplest, consists of finding the optimal route (or set of routes) from one or more source nodes (or depots) through a set of destination nodes, depending on certain constraints [Lap92]. A practical example of this is the routing of courier/delivery vehicles through locations in a city in order to deliver packages to customers. There are several variations of the VRP. For example, in cases where a return to the depot is not required, the VRP is known as an open VRP [MA11]. Also, each vehicle may or may not have a limit on its package carrying capacity. These cases are known as Capacitated [CLSV07] and Uncapacitated [EVR09] VRPs respectively. In cases where packages have to be delivered within specific time windows, the problem is known as VRP with Time Windows (VRPTW) [Sol84]. A good reference on the many vehicle routing problem variants and applications is given in [TV14]. A taxonomic review of the VRP, including variants and methods of solution is also given in [BRVN15].

The VRP also finds application in agriculture, and has been used for example, in scheduling the collection and transportation of livestock [OL08, SPS04], in managing the operation of farm machinery in separated fields in [Boc08], for the planning of autonomous tractor operations in [BVG09, BVG$^+$08], and also for determining the optimal routes for combine harvesters in [AVVO09]. A dedicated classification of agricultural field operations (AFO) with respect to the VRP is given in [BS09, BS10]. In [BBV15], the authors try to minimize working time (including travel time) of grape harvesters, which gives the problem some resemblance to routing problems.

While [BFW06], similar to [Boc08], use the VRP for farm-to-farm path determination for scheduling crop harvesting in multi-farm operations, they do not have the added problem of the MSSCSP. In Section 5.4.2, the MSSCSP is combined with vehicle routing in order to select contiguous blocks for harvesting, and at the same time decide how to harvest them. This combination of the VRP and the MSSCP is similar to the routing problem with loading constraints reviewed in [IM10], as cutting and packing problems are analogous. In this chapter however, there are several differences.

First of all, unlike the standard routing and loading problems, e.g. [DFH+07, FDHI10, JOCM13, MRWR12], there is no consideration for the transportation of the cut logs, as vehicles different from the harvester are responsible for this. The primary concern in this chapter is with selecting the right weight vector to be used by the harvester in each block of the forest so that the amounts of the different types of log products obtained are enough to satisfy the demand for these products. In addition to selecting the right weight vector, the secondary goal is to select neighbouring or contiguous blocks to harvest such that the harvester follows a path as short as possible in order to minimise harvesting costs. Additionally, the selection of mostly low-value blocks is of importance, as this ensures that the best trees of the forest are left intact. Finally, there is some uncertainty in this combined cutting and routing problem, as the expected yields differ from the actual yields obtained.

5.4 Models for Reactive Harvesting

To consider the uncertainty in the problem, two models (and a reactive algorithm which incorporates these models) are introduced. The first model is a simple cutting stock ILP which, when used with the reactive algorithm, is applicable to MSSCSPs for which there is uncertainty in the stock dimensions, and the order of selection of the stock for cutting is not important. The second model combines the cutting and routing problems and is of specific interest to the forest harvesting problem as the order of block selection is relatively important.

These models use the weights $\vec{w}_q$ and corresponding patterns $\vec{p}_q$ generated in 4 using Monte Carlo simulation. Monte Carlo simulation was used without clustering due to the prevalence of natural clusters in the data, as seen in Section 4.6. $\vec{w}_q$ and $\vec{p}_q$ are then provided as an input of Algorithm 7. Other input parameters are: the set of types of products $\mathcal{L}$ and the demands D_l for each product type l, the set of stock subsets (or blocks) $\mathcal{Q}$, the cost V_q of each subset q, and the dimensions of the stock items $\vec{\sigma}_{qr}$. To evaluate the performance of the reactive approach, the stock dimensions data was randomly split into two sets, with one set (25% of the original data) used as the sample data (i.e. the data on which the harvesting operation is planned), and the other set (75% of the original data) used as the real data (i.e. the data seen when actual harvesting of the forest has been carried out).

5.4.1 Simple ILP

This cutting stock model is the bilevel cutting stock problem introduced in Section 4.3 and decomposed to a single-level ILP in equation (4.6). It is used in cases in which the order of selection of blocks is not important. It is denoted as $f(\vec{D},\vec{P},\vec{V},\mathcal{Q})$ and defined as follows:

$$\begin{array}{lll}
\text{argmin } \sum_{q=1}^{Q} \sum_{k=1}^{K} V_q x_{qk} & & \\
\text{s.t.} & & \\
\sum_{q=1}^{Q} \sum_{k=1}^{K} P_{kql} x_{qk} \geq D_l & l=1,\dots,L & \\
\sum_{k=1}^{K} x_{qk} \leq 1 & q=1,\dots,Q & \\
x_{qk} \in \{0,1\} & q=1,\dots,Q,\ k=1,\dots,K &
\end{array} \tag{5.1}$$

This ILP model has as decision variables x_{qk}, which indicate if the stock subset q is cut with the global cutting pattern k. The objective function minimizes the total cost of the raw materials subsets used in the cutting process for satisfying the demands. Note that if a subset of raw materials pieces q is not used for satisfying the demands (and therefore it is not cut), then all its decision variables ($x_{qk}\ \forall q$) are zero. The first constraint ensures that all demands are fulfilled, while the second prevents the use of more than one cutting pattern for a subset of raw material pieces.

5.4.2 Combined Vehicle Routing Problem ILP

In Section 5.4.1 the stock subsets are selected randomly, without regard of the location of the subsets. However, in certain situations, some neighbourhood constraints on the subsets to be selected may exist. For example, in forestry harvesting, there are substantial costs associated with moving harvesting machines from one block to a non-neighbouring block. It is therefore important that during harvesting, contiguous blocks are selected in order to create a continuous path for the harvester to follow through the whole area. In order to do this, $f(\vec{D},\vec{P},\vec{V},\mathcal{Q})$ is modified to include neighbourhood and routing constraints, and this combined Cutting Stock and Vehicle Routing Problem is denoted as $f(\vec{D},\vec{P},\vec{V},\mathcal{Q},E)$:

$$\min \sum_{q=1}^{Q}\sum_{k=1}^{K} V_q x_{qk} + \sum_{q=0}^{Q}\sum_{t=0}^{Q} e_{qt} y_{qt} \tag{5.2}$$

s.t.

$$\sum_{q=1}^{Q}\sum_{k=1}^{K} P_{kql} x_{qk} \geq D_l \qquad \forall l \in \mathcal{L} \tag{5.3}$$

$$\sum_{q=0}^{Q} y_{qt} = \sum_{k=1}^{K} x_{tk} \qquad \forall t \in \mathcal{Q} \setminus \{0\} \tag{5.4}$$

$$\sum_{t=0}^{Q} y_{qt} = \sum_{k=1}^{K} x_{qk} \qquad \forall q \in \mathcal{Q} \setminus \{0\} \tag{5.5}$$

$$\sum_{q=0}^{Q} y_{qt} \leq 1 \qquad \forall t \in \mathcal{Q} \setminus \{0\} \tag{5.6}$$

$$\sum_{t=1}^{Q} y_{qt} \leq 1 \qquad \forall q \in \mathcal{Q} \tag{5.7}$$

$$\sum_{q=0}^{Q} y_{qu} - \sum_{t=1}^{Q} y_{ut} \leq 1 \qquad \forall u \in |\mathcal{Q}| \setminus \{0\} \tag{5.8}$$

$$\sum_{t=1}^{Q} y_{0t} = 1 \tag{5.9}$$

$$z_q - z_t + |\mathcal{Q}| \times y_{qt} \leq |\mathcal{Q}| - 1 \qquad \forall q, t \in \mathcal{Q} \setminus \{0\} \tag{5.10}$$

$$z_q, z_t \geq 0 \qquad \forall q \in \mathcal{Q} \tag{5.11}$$

$$y_{qq} = 0 \qquad \forall q \in \mathcal{Q} \tag{5.12}$$

$$x_{qk} \in \{0,1\} \qquad \forall q \in \mathcal{Q},\ k = 1, \ldots, K \tag{5.13}$$

$$y_{qt} \in \{0,1\} \qquad \forall q, t \in \mathcal{Q} \tag{5.14}$$

where E is a matrix denoting the cost e_{qt} of going from block q to block t. Blocks which are contiguous have a minimal cost of traversal, while non-contiguous blocks have a high travel cost between them. This problem also contains the additional decision variable y_{qt} which indicates whether the harvester moves from block q to block t. Also included is a dummy block to the set $\mathcal{Q}$ (at $q = 0$), which acts as the depot in the classic VRP. The objective function of $f(\vec{D}, \vec{P}, \vec{V}, \mathcal{Q}, E)$ minimizes the value of the blocks selected as well as the cost of the path followed while selecting the blocks.

Constraint (5.3) is the original demand constraint, while the rest of the constraints differ from the original ILP model. Constraints (5.4) and (5.5) are the bridging constraints that link the CSP to the VRP. They ensure the consistency between the decision variables of the blocks selected for cutting and the decision variables of their cutting order. Constraints (5.6) and (5.7) state that there is only a single entrance and a single departure from a block so that blocks cannot be re-visited. Constraint (5.8) is a flow conservation constraint that ensures the continuity of the path. Since it is allowed that the path may not have complete continuity, ≤ 1 is used instead of $= 0$. Constraint (5.9) ensures that the harvester only leaves the depot once. Although more effective formulations such as those in [Lap92] exist, the Miller-Tucker-Zemlin subtour elimination constraints in (5.10) have been used for simplicity. Constraint (5.12) eliminates trips from a block to itself. Finally, in lines (5.13) and (5.14) the binary decision variables are defined. This version of the VRP is uncapacitated, as there is no limit to how much the harvester can cut.

In order to reduce the computational time, a Rolling Horizon approach [PM95] was used to solve the problem. A rolling horizon approach divides the problem into a series of time-periods. The current time-period is modelled precisely, while the rest of the time-periods are aggregated and solved using a relaxed model. This has the effect of speeding up computation, with only a modest effect on the optimality of the solution. For this combined cutting and routing problem, blocks directly connected to the depot were considered as being in the current time-period, while the rest of the blocks were in the aggregated time-period. (5.13) was relaxed so that $x_{qk} \in [0, 1]$ for the blocks that are not connected to the depot. On the other hand, the variables x_{qk} of the blocks connected to the depot were kept as they were originally: as binary variables. In order to achieve this, define binary variables b_{qk}, so that when there is a connection from the depot to a block q, i.e. when $e_{0q} = 1$, x_{qk} is constrained to be equal to b_{qk}. In this way, the full integer problem is solved for blocks connected to the depot, while the rest of the blocks, (i.e. those not connected to the depot) are solved using a relaxed version of the problem. The optimisation problem is therefore modified as:

$$x_{qk} \in [0,1] \qquad \forall q \in \mathcal{Q}, k = 1, \ldots, K \tag{5.15}$$

$$b_{qk} \in \{0,1\} \qquad \forall q \in \mathcal{Q}, k = 1, \ldots, K \tag{5.16}$$

$$x_{qk} = b_{qk} \text{ if } e_{0q} = 1 \qquad \forall q \in \mathcal{Q}, k = 1, \ldots, K \tag{5.17}$$

$$\sum_{k=1}^{K} b_{qk} = 1 \qquad \forall q \in \mathcal{Q} \tag{5.18}$$

where (5.13) has been replaced by (5.15) to (5.18).

5.5 Reactive Cutting Algorithm

The reactive cutting algorithm re-calculates the product amounts D_l targeted after cutting each real block, and re-computes new solutions for the rest of the cutting process. This algorithm works both with the simple ILP, as well as with the combined cutting and routing problem. It is an on-line algorithm, as the dimensions of most of the raw material pieces are only known once the real cutting process is being performed. Also, the real amounts of types of products obtained from each block are only known after a block is harvested.

Algorithm 7 Reactive Cutting for the MSSCSP

```
**Data**: $\mathcal{L}, \mathcal{Q}, \vec{V}, \langle \vec{\sigma}_{q1}, \dots, \vec{\sigma}_{qr} \rangle \; (\forall q \in \mathcal{Q}), q_0, E, \vec{P}_{kq}, \vec{W}_{kq}, \vec{D}$
**Result**: $\mathcal{U}$, $tc$
$\mathcal{U} \leftarrow \emptyset$;
$tc \leftarrow 0$;
**while** $\vec{D} \neq \vec{0}$:
    **if** without VRP (Section 5.4.1):
        $\vec{x} \leftarrow f(\vec{D}, \vec{P}, \vec{V}, \mathcal{Q})$; *Cutting simulation*
        Select a random $q$ and $k$: $x_{qk} = 1$;
    **else if** with VRP (Section 5.4.2):
        $\vec{x} \leftarrow f(\vec{D}, \vec{P}, \vec{V}, \mathcal{Q}, E)$; *Cutting simulation*
        For $y_{0q} = 1$, select $k : x_{qk} = 1$; *Next subset selected is q*
        $q_0 \leftarrow q$; *Update the depot for next iteration*
    $\vec{p}_{kq} \leftarrow \mathcal{A}(\mathcal{L}, \langle \vec{\sigma}_{q1}, \dots, \vec{\sigma}_{qr} \rangle, \vec{w}_{kq})$; *Cut the q real subset*
    $D_l \leftarrow \max(D_l - p_{kql}, 0)$; *Update demands*
    $\mathcal{Q} \leftarrow \mathcal{Q} \setminus \{q\}$; *Remove q from uncut blocks*
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{q\}$; *Add q to the cut blocks*
    $tc \leftarrow tc + V_q$; *Update the total cost of the cutting*
**end while**
```

Algorithm 7 first initializes the variables $\mathcal{U}$ and tc, which store the blocks cut to satisfy the demands, and their total cost respectively. These data are returned by the algorithm once all the demands are fully satisfied (the stopping condition of the loop). In each iteration, the algorithm first solves the ILP model using the sample data. This provides the optimal blocks to cut together with the cutting patterns to use (based on the sample data). After this simulation process, the stock selected in the simulation phase is cut for real, and the leftover demands are calculated. The status of the selected block is also changed from uncut to cut, and the total cost so far tc is updated. The algorithm then selects one of those blocks either randomly (without VRP), or it selects a neighbouring stock subset (with VRP). In the latter case, the depot for the next iteration is updated to the stock subset harvested in the current iteration. In real-life applications, customers might change their demands over the time or there might be new demands from other customers. This dynamic algorithm can accommodate these additional demands while the static LP approaches cannot. This can be done by updating $\vec{D}$ at any step of Algorithm 7.

5.6 Evaluation

In this section, the reactive algorithm (with and without vehicle routing) is compared with another dynamic approach - the Simulated Annealing Like Algorithm (SALA) meta-heuristic from [MAA06] - to dynamically solve the MSSCSP treated in this chapter. This meta-heuristic is popularly used in the forestry industry. Data for the experiments was provided by the industrial partner. As previously mentioned, the objective is to minimize the total cost of the blocks harvested. The first forest under consideration is composed of 26 blocks whose volumes are in the interval $[84.3, 293]m^3$. The costs of the individual blocks are in the interval €$[4, 315.25, 20, 422.16]$ and the total cost of them all is €$306, 295.23$. The total volume of the logs in the forest is $4, 640.66m^3$.

Both the original meta-heuristic algorithm of [MAA06] and the reactive cutting algorithm were implemented in Java. The evaluation was done on a 2.3 GHz Intel Core $i7$ processor. An industrial partner also provided the black box software that carries out the dynamic-programming-based simulation that implements $\mathcal{A}$ (see Equation 4.3), which is used by the two approaches. Since DP is a complete algorithm, other DP implementations, such as the one in [PM72], could have been used for $\mathcal{A}$, with equivalent results. Therefore, these experiments are reproducible. The ILP models of Algorithm 7 were solved with CPLEX solver with a time cut-off of 90 seconds and an optimality gap of 2.5%.

5.6.1 Reactive Approach without Vehicle Routing

Figure 5.9 shows the mean results of 20 runs for each experiment performed with seven log types ($|\mathcal{L}| = 7$). The demands of each log type were randomly selected in the interval $[200, 600]m^3$. Several runs were performed because both approaches stochastically select the next block to cut. For Algorithm 7 the next block to cut is randomly selected from the best uncut blocks selected by solving the current ILP model. However, for [MAA06] the next block to be cut is randomly selected from all the uncut blocks. Because the global cutting patterns are also randomly generated, they are also re-computed in each of the 20 runs. In the meta-heuristic approach of [MAA06] the local changes in the weight vector are iteratively randomly made for each run as well.

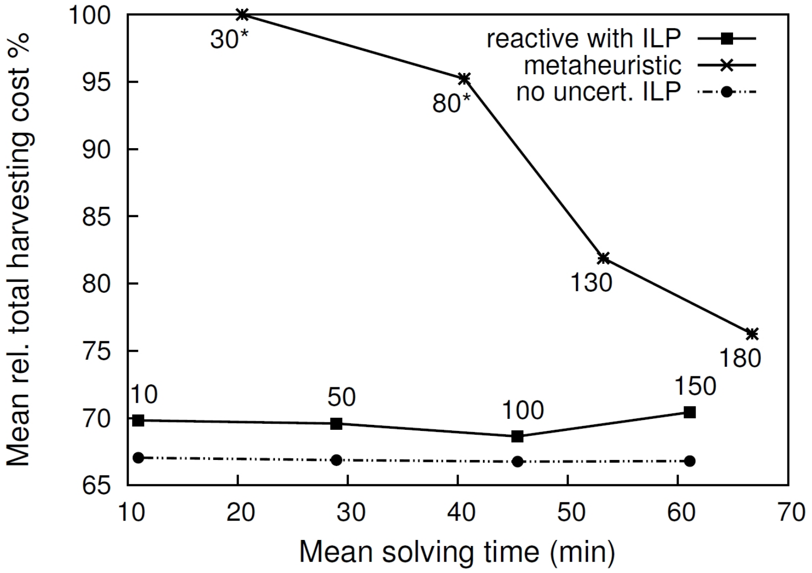


Figure 5.9: Comparing the solutions of the reactive approaches

Figure 5.9 shows the quality of the solutions obtained in the vertical axis. This is the mean relative percentage of total cost of the harvested stock subsets:

$$\text{mean relative percentage} = \frac{\text{cost of harvested subsets}}{\text{total cost of all subsets}} \times 100 \qquad (5.19)$$

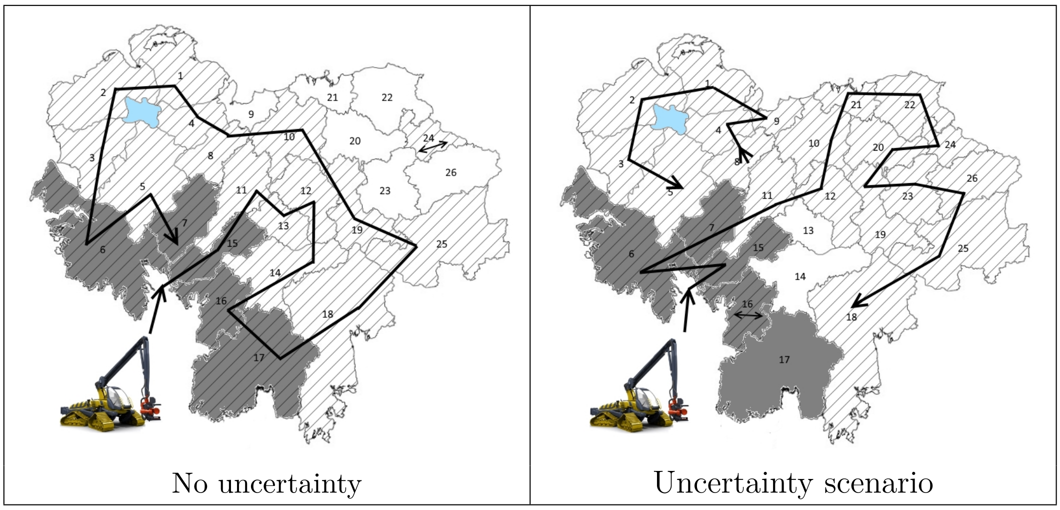
Note that this total cost is averaged for each of the twenty instances with randomly generated demands. It should be noted that the lower this percentage is, the greater the solution quality. The horizontal axis shows the mean solving time in minutes. For the approach of this thesis, this means the sum of the time for generating the global cutting patterns and the time consumed by Algorithm 7.

The three lines from top to bottom correspond to the solutions obtained by the meta-heuristic approach (star markers), the reactive ILP approach (square markers), and the "static" scenario (dotted-dashed line) in which there is no uncertainty in the dimensions of the stock. This scenario answers the question

"What would be the harvesting cost if the dimensions of the whole population are known?". For calculating this value, the ILP model was run only once with the sampled data. It should also be noted that this "static" way of solving the problem is very similar to those in [PFCO15] and [DRF17]. However, in real life applications the uncertainty is patent, and it is unrealistic to not consider it.

For the reactive approach (square markers), each number associated with a marker represents the number of generated cutting patterns for each block (e.g. for the number 100, 100 cutting patterns were generated for each of the 26 blocks, resulting in a total of 2600 cutting patterns). For the meta-heuristic, each number associated with a marker represents its cut-off time (in seconds) for each block (e.g. for each block, local changes were made during 80 seconds and the best combination is kept).

In Figure 5.9 it can be observed that the meta-heuristic algorithm behaves poorly for low cut-off times. This suggests that it needs a significant amount of time to find good solutions (this is even more prominent when the number of log types grows). The $*$ sign over the 30s and 80s cut-offs indicates that the meta-heuristic could not find a solution for some instances (all the forest was harvested but the demands were not satisfied). Specifically, for 30s no solution was found for any of the runs). Note that the quality of the solutions increases for greater cut-offs. Nevertheless, the difference between the percentage costs for both approaches is greater than 6.4% even when the reactive approach takes 66 minutes. This represents a monetary increase of at least €19,602.9. One reason for such out-performance is that the reactive approach considers the block selection as an optimality criterion, while the meta-heuristic does not. The other reason is that meta-heuristics are incomplete algorithms. Figure 5.9 also suggests that the quality of the reactive approach solutions barely improves when the number of generated cutting patterns increases from 50 to 100. For this reason, generating a high number of cutting patterns is not recommended in time-sensitive applications. The economical benefits of the reactive approach over the meta-heuristic are seen to be significant, especially considering that in real-life applications, decisions must be made quickly (on-line cutting processes).



No uncertainty

Uncertainty scenario

Figure 5.10: Path of harvester through forest.

5.6.2 Reactive Approach with Vehicle Routing

5.6.2.1 Routing through 26 Blocks

In this section, the reactive algorithm is tested with the ILP model that considers vehicle routing. Note that the same instance (volume, types of logs, demands, etc.) as in Section 5.6.1 was analysed. The cost of moving the harvesting machines between neighbouring blocks was set to €0, while the moving cost between non-neighbouring blocks was set to €10, 000. However, the costs could also depend on the distance between blocks or other criteria given by the real application. Figure 5.10 shows a map of the forest. It should be noted that since the real neighbourhood information was not provided by the industrial partners, neighbours were randomly assigned. The dark coloured blocks are those which are connected to the road, which is the only point of access to the forest. Thus, the depot is connected to blocks 6, 7, 15, 16 and 17.

30 cutting patterns were generated for each of the following two scenarios: with and without uncertainty (Figure 5.10). For the scenario without uncertainty it was assumed that the characteristics of all the stock are known a priori. For the scenario with uncertainty, only a sample (about 25%) of the real stock is known. Therefore, it is expected that this uncertainty will have an effect on the routing of the harvester. The route taken by the harvester is shown by the arrows. Each solution was found in less than 2 minutes, which is mainly due to the relaxations in rolling-horizon approach (see Section 5.4.2).

For the scenario without uncertainty (left panel of Figure 5.10), a percentage mean relative total harvesting cost of 70.85% was obtained. While this percentage is more than the one obtained by the reactive approach without VRP, it is still less than the one obtained by the meta-heuristic of [MAA06] (see Figure 5.9). This increase in the percentage of blocks harvested over the approach without VRP is due to the fact that routing restricts the freedom with which the blocks can be selected. Even so, this scenario is the most realistic for many industries, such as in forestry. It can be seen that in this case of no uncertainty, the harvester mostly follows one path (more than one path is allowed), with a jump to block 24 at the end. The reason block 24 is selected is because it is the block with the least cost that has not yet been harvested.

The right panel of Figure 5.10 shows the uncertainty case, i.e. the case in which the sample data differs from the real data. Here, the percentage mean relative total harvesting cost is 86%. Due to the uncertainty, the algorithm has a difficult time selecting a single path to follow through the forest, resulting in 3 different paths (with the associated penalties for non-contiguous blocks traversal). Even so, the algorithm performed better than the meta-heuristic algorithm with a solving time up to 45 min. (see Figure 5.9). (Note that the solving time of the reactive approach was less than 2 min.).

5.6.2.2 Routing through 100 Blocks

A larger problem instance with 100 blocks was also solved for 3 different sets of demands in order to evaluate the routing constraints. For this large instance, a “search window” was implemented so that the algorithm only looked at most 30 blocks ahead in each iteration. As before, the dark coloured block is connected to the depot, and the hashed blocks are those selected for harvesting.

In Table 5.2, three types of demands (low, medium and high) for each log type are shown. Table 5.3 shows the results of the three types of demands, where Case A corresponds to the standard case in which there is some uncertainty in the amounts obtained from the real forest. The amounts expected from the sample data differ from what are actually obtained after harvesting. In case B there is no uncertainty; what is obtained after harvesting is exactly what was expected from the sample data. In general, the blocks selected are seen to be contiguous and mostly of low value, leaving the better areas of the forest intact.

Figure 5.11 shows the blocks selected for Demand 1 for both cases. Since the demands are low, contiguous blocks are easily selected. As expected, when there is uncertainty, more blocks are harvested (Case A). In figure 5.12, the demands are much larger, leading to more blocks being selected for harvesting. The reactive approach does route the harvester through contiguous blocks, however in Case B, there are two groups of blocks selected for harvesting. Although there are two groups of blocks, the selected blocks are of low-value, and are enough to satisfy the demand, thus keeping the harvesting cost lower than that of the uncertain case. The results for Demand 3 are shown in figure 5.13. While the demands are fulfilled and the costs kept low, it comes at the cost of having a single-path through the forest. The larger demands cause three and two groups of contiguous blocks to be selected for Cases A and B respectively. Although multiple paths are allowed by the model (5.2 to 5.14), a larger movement penalty or more restrictive constraint could be used in cases in which the harvesting operators want to stringently enforce the single-path scenario.

Table 5.2: Demands (m^3).

Log type	1	2	3	4	5	6	7
Demand 1	50.4566	34.1831	25.4884	51.4319	23.6414	51.5178	59.8344
Demand 2	353.1962	239.2817	178.4188	360.0233	165.4898	360.6246	418.8408
Demand 3	432.721	572.215	369.114	357.69	489.617	332.468	477.978

Table 5.3: Results for routing through 100 blocks

	Demand 1		Demand 2		Demand 3	
Case	A	B	A	B	A	B
Total Cost (€)	58265	49361	325133	110532	276491	158011
Number of Blocks Harvested	7	6	34	11	20	18
Total Solution Time (sec)	38	3	661	130	728	361

As expected, when there is uncertainty, more blocks are harvested, as the samples do not always accurately reflect the capacity of the forest. Table 5.3 also shows that the discrepancy between the amounts expected from sampling and the amounts actually obtained leads to an increase in the number of blocks harvested, as well as the total harvesting cost. This illustrates the negative effect of having bad samples.

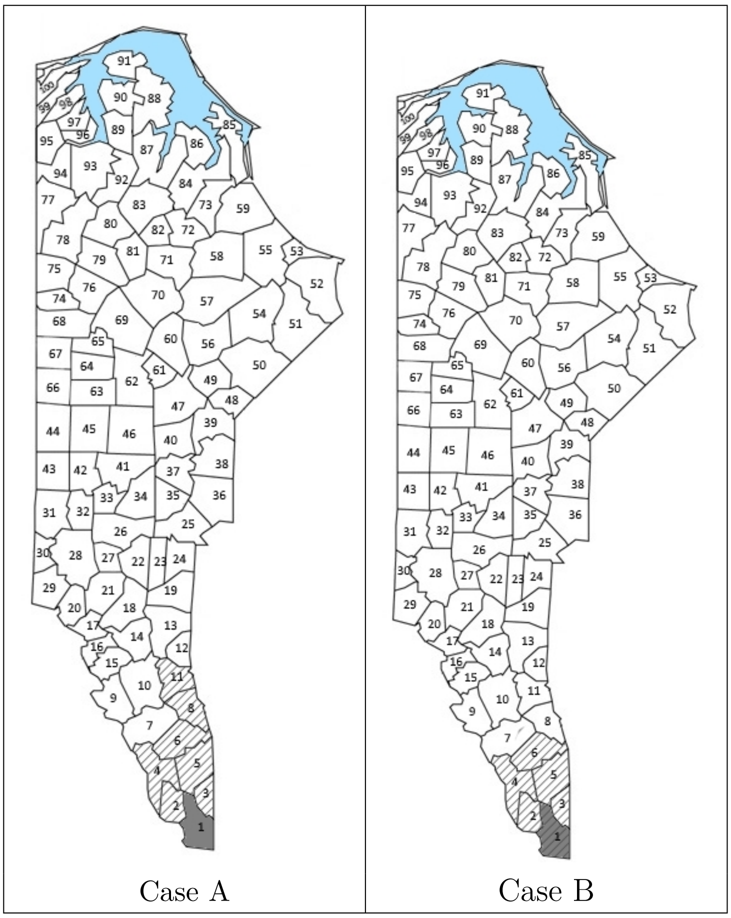
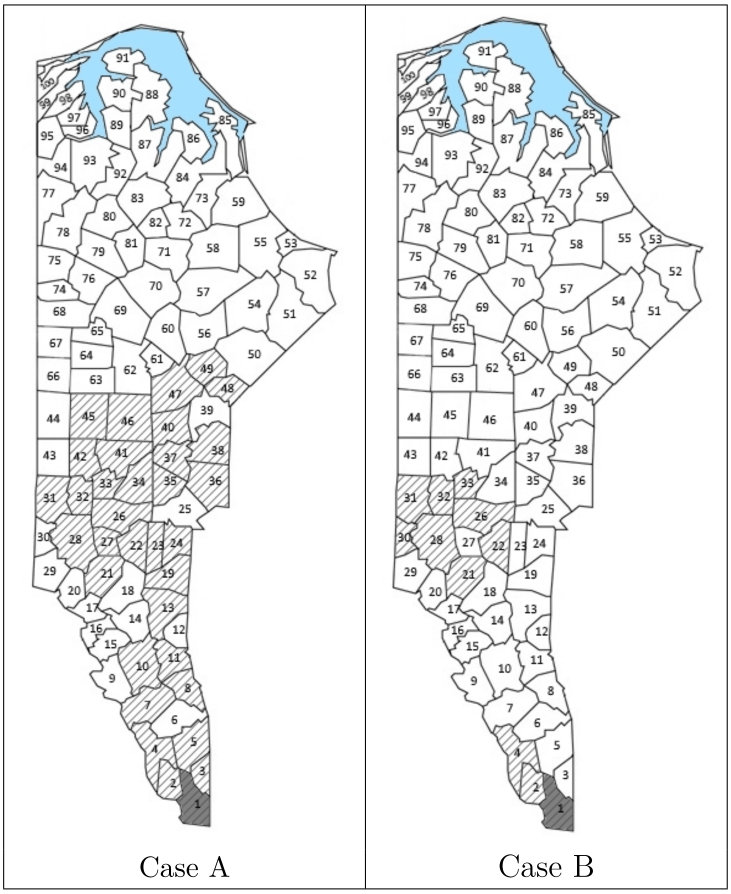
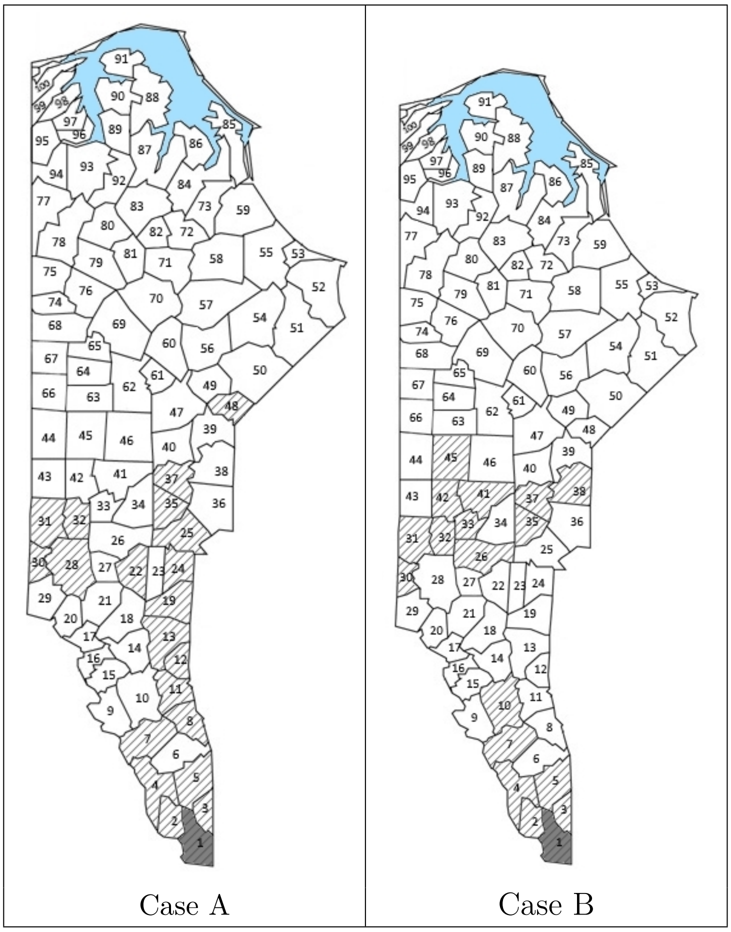


Figure 5.11: Block selection from 100 blocks for Demand 1



Case A Case B

Figure 5.12: Block selection from 100 blocks for Demand 2



Case A

Case B

Figure 5.13: Block selection from 100 blocks for Demand 3

5.7 Conclusions

In *Multiple Stock Size Cutting Stock Problems* (MSSCSPs) some sets of stock of different sizes must be cut following certain *patterns* to meet customers' demand for certain products types. The additional difficulty of the MSSCSP under consideration lies in the fact that estimates of the stock sizes differ (sometimes significantly) from real stock sizes. This creates two complications: (i) the cutting patterns of stock items with unknown dimensions can not be computed, and (ii) even if they are computed for the sampled stock, the amounts obtained from cutting the real stock will differ from what is expected.

A reactive approach, which re-computes cutting solutions after each stage of the cutting process, was presented to tackle this uncertain MSSCSP. The difference between expected and obtained amounts was first computed, then the "targeted" amounts were updated as the cutting process progresses. For the forest harvesting problem, an Integer Linear Program (ILP) that combines the dynamic MSSCSP with the Vehicle Routing Problem (VRP) was also presented in order to decide the path that the harvesting machines should follow. In the evaluation, it was shown that the proposed reactive approach out-performed the meta-heuristic algorithm [MAA06] popularly used in the forestry industry, especially for low cut-off times.

For both the 26-block and 100-block examples, when there is no uncertainty in the stock dimensions (case B), fewer blocks are harvested. The cost of harvesting is also less. As expected, the uncertainty has a negative effect on the amount of the forest harvested as more blocks are harvested resulting in a higher cost.

As future work, obtaining real location and neighbourhood data for the blocks will enable the travel costs to be based on the real distances between all blocks. It is also important to investigate means of improving the accuracy of the sampled data, which should have a positive effect on the results in cases in which the sample data is not representative enough.

Chapter 6

Conclusions

6.1 Thesis Defence

The thesis defended throughout this dissertation is that an analytics-based decomposition approach can be used to solve large-scale multiple-follower bilevel problems more efficiently than standard approaches. While good for certain cases, existing methods for solving these problems are not applicable in cases where the follower problems are black-box functions. Although evolutionary approaches can be used in these cases, they can be time-consuming and for large problems, unable to efficiently search the whole solution space. For this reason, an analytics-based approach, which is better able to sample the solution space is needed. Regarding the forest harvesting problem, the focus of this thesis has been to formulate the problem as a bilevel multiple-follower problem. This is a natural fit as it considers the harvester's operation for each block as an optimisation problem (which it is). Using this bilevel multiple-follower approach allows the problem to be modelled more accurately, and solved using the decomposition approach. Additionally, a reactive harvesting approach was developed. This approach takes into consideration the difference in the expected and harvested amounts, as well as the location of the blocks to guide the harvesting process. Specifically, the thesis contributions are presented below.

Contribution 1 *A new class of multiple-follower bilevel problems is proposed.*

The first contribution of this dissertation is the formalisation of a new class of bilevel problems not listed in [LSZ06]. In Chapter 2, a new class of bilevel multiple-follower bilevel problems was proposed. In this class, the followers do not share each others' variables, so that the leader variables can be partitioned among the followers. The followers are also allowed to be integer or non-linear, and variables from different follower problems are only connected through weak constraints.

Contribution 2 *A novel analytics-based decomposition approach for large-scale bilevel multiple-follower problems is given.*

In Chapter 3, a novel analytics-based decomposition approach for bilevel multiple-follower problems is presented. The first numerical example was given to show the benefit of using clustering. Two other numerical examples were solved using this approach, and the results compared to those obtained by using evolutionary algorithms (which is a standard approach). For the second example, a small-sized problem with only two followers is solved to within 0.09% of the optimal. This shows that even for small-scale problems, the analytics approach is competitive as it is able to cover the space of the follower problems adequately. The third example was an arbitrarily large problem evaluated for up to a thousand followers. The results were compared with those from two evolutionary approaches and it was seen that as the size of the problem increased, the decomposition approach produced significantly better results than the standard approaches.

Contribution 3 *Reformulation of the forest harvesting problem as a bilevel optimisation problem to take into the account operation of harvester.*

In Chapter 4, the forest harvesting problem is reformulated as a multiple-follower bilevel problem, which allows for the operation of the harvester to be incorporated into the optimisation problem as non-standard follower problems. This bilevel formulation combined with the decomposition approach results in the selection of the best weights to apply to the harvesting machines. These machines in turn produce the best cutting patterns to be used to harvest each block in order to satisfy demand and reduce waste. Evaluations showed that the results obtained were close optimal, with the 8 block problem getting to within 0.4% of the optimal.

Contribution 4 *A reactive harvesting approach to mitigate the effects of the uncertainty in the data.*

In Chapter 5, a reactive harvesting approach is introduced in order to diminish the effects of data uncertainty on the harvesting process. Two integer programming models are proposed for reactive harvesting - a simple integer linear program, and one in which the cutting stock problem was combined with the vehicle routing problem. These models are used in conjunction with a reactive harvesting algorithm to select the next best blocks to harvest based on the amounts already harvested, the demand yet to be fulfilled, the estimated capacity of the unharvested blocks, and, in the case of the VRP, the neighbourhood information of the unharvested blocks. Evaluations comparing the approach used in the industry with the proposed reactive approach were carried out, and it was seen that the proposed reactive approach out-performed the meta-heuristic algorithm widely used in the forestry industry, particularly for low cut-off times.

6.2 Directions for Future Work

There are a few possible lines of work that can be explored as an extension of this dissertation. These are outlined below.

Extending the Decomposition Method to Non-Independent Followers. The analytics-based decomposition approach depends on the follower problems being independent, i.e. not sharing any variables amongst themselves. It would be of benefit to be able to apply the approach to multi-follower problems which have shared variables. This would entail finding methods of decoupling the variables which are shared amongst the followers.

Analytics-Based Reaction Set Mapping In the literature, evolutionary approaches which use iterative approximations of the reaction set to guide the populations' evolution can be seen [SMD17b]. The decomposition approach of this thesis is similar to reaction set mapping in that both approaches seek to understand how the lower levels behave in order to essentially remove the need for solving the lower level problems. While the reaction set mapping approach is carried out iteratively (i.e. for each generation), an analytics-based approach could be quicker and easier way to map the reaction set of the follower, as it not iterative, and a relationship between leader and follower variables can be determined using Monte Carlo sampling and clustering.

Real Location and Neighbourhood Information for Reactive Harvesting. In Chapter 5, no neighbourhood information was provided by the industrial partner so blocks were randomly assigned neighbours. Obtaining real location and neighbourhood information for blocks will allow the travel costs to be based on the actual distances between the blocks. It is expected that this will also improve the performance of the reactive algorithm, as blocks of a similar value tend to be grouped together in real life.

Data Accuracy Improvement. Finally, as a result of the method of estimating forest capacity (i.e. sampling), the sample data is sometimes not very representative of reality. Although the proposed reactive approach helps to mitigate the effect of the uncertainty, it would be useful to also investigate means of improving the accuracy of the sample data.

References

[AA91] G. Anandalingam and V. Apprey. Multi-level programming and conflict resolution. *European Journal of Operational Research*, 51(2):233–247, 1991.

[AB15] J.S. Angelo and H.J.C. Barbosa. Differential evolution to find stackelberg-nash equilibrium in bilevel problems with multiple followers. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, pages 1675–1682, 2015.

[AdC08] C. Alves and J.M.V. de Carvalho. A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & OR*, 35(4):1315–1328, 2008.

[ASW$^+$15] D.R. Anderson, D.J. Sweeney, T.A. Williams, J.D. Camm, and J.J. Cochran. *An introduction to management science: quantitative approaches to decision making.* Cengage learning, 2015.

[AVVO09] O. Ali, B. Verlinden, and D. Van Oudheusden. Infield logistics planning for crop-harvesting operations. *Engineering Optimization*, 41(2):183–197, 2009.

[Bar84] J.F. Bard. Optimality conditions for the bilevel programming problem. *Naval Research Logistics Quarterly*, 31(1):13–26, 1984.

[Bar88] J.F. Bard. Convex two-level optimization. *Mathematical programming*, 40(1):15–27, 1988.

[BBV15] N. Briot, C. Bessiere, and P. Vismara. A constraint-based approach to the differential harvest problem. In *International Conference on Principles and Practice of Constraint Programming*, pages 541–556. Springer, 2015.

[BD15] R.E. Bellman and S.E. Dreyfus. *Applied dynamic programming.* Princeton university press, 2015.

[Ber99] D.P. Bertsekas. *Nonlinear programming.* Athena scientific Belmont, 1999.

[BFW06] C.B. Basnet, L.R. Foulds, and J.M. Wilson. Scheduling contractors' farm-to-farm crop harvesting operations. *International transactions in operational research*, 13(1):1–15, 2006.

[BJS10] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali. *Linear Programming and Network Flows.* Wiley, 2010.

[Boc08] D. Bochtis. Planning and control of a fleet of agricultural machines for optimal management of field operations. *PhD diss. Thessaloniki, Greece: Aristotle University of Thessaloniki, Department of Agricultural Engineering*, 2008.

[BRVN15] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 2015.

[BS02] G. Belov and G. Scheithauer. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research*, 141(2):274–294, 2002.

[BS09] D.D. Bochtis and C.G. Sørensen. The vehicle routing problem in field logistics part i. *Biosystems engineering*, 104(4):447–457, 2009.

[BS10] D.D. Bochtis and C.G. Sørensen. The vehicle routing problem in field logistics: Part ii. *Biosystems engineering*, 105(2):180–188, 2010.

[Bul04a] J.A. Bullinaria. Learning vector quantisation (lvq), 2004.

[Bul04b] J.A. Bullinaria. Self organizing maps: Algorithms and applications, 2004.

[BVG+08] D.D. Bochtis, S.G. Vougioukas, H-W. Griepentrog, N.A. Andersen, and et al. Effects of capacity constraints on the motion pattern of an autonomous orchard sprayer. *Proc. Agricultural and Biosystems Engineering for a Sustainable Word, EuAgEng, OP-375, 8pp, Crete, Greece*, 2008.

[BVG09] D.D. Bochtis, S.G. Vougioukas, and H-W. Griepentrog. A mission planner for an autonomous tractor. *Transactions of the ASABE*, 52(5):1429–1440, 2009.

[CA33] J.A. Clarkson and C.R. Adams. On definitions of bounded variation for functions of two variables. *Transactions of the American Mathematical Society*, 35(4):824–854, 1933.

[CBS15] A. Chaabani, S. Bechikh, and L.B. Said. A co-evolutionary decomposition-based algorithm for bi-level combinatorial optimization. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, pages 1659–1666, 2015.

[CBSA15] A. Chaabani, S. Bechikh, L.B. Said, and R. Azzouz. An improved co-evolutionary decomposition-based algorithm for bi-level combinatorial optimization. In *Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, Companion Material Proceedings*, pages 1363–1364, 2015.

[CG07] H.I. Calvete and C. Galé. Linear bilevel multi-follower programming with independent followers. *J. Global Optimization*, 39(3):409–417, 2007.

[CKR71] R.G. Cassidy, M.J.L. Kirby, and W.M. Raike. Efficient distribution of resources through three levels of government. *Management Science*, 17(8):B–462–B–473, 1971.

[CLR67] I.M. Chakravarti, R.G. Laha, and J. Roy. *Handbook of methods of applied statistics*. Number v. 1 in Wiley series in probability and mathematical statistics. Wiley, 1967.

[CLSV07] J-F. Cordeau, G. Laporte, M.W.P. Savelsbergh, and D. Vigo. Vehicle routing. *Handbooks in operations research and management science*, 14:367–428, 2007.

[CM16] M. Caramia and R. Mari. A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints. *Optimization Letters*, 10(5):997–1019, 2016.

[CMS07] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007.

[COP16] L. Climent, B. O'Sullivan, and S.D. Prestwich. Bounding the search space of the population harvest cutting problem with multiple size stock selection. In *Learning and Intelligent Optimization Conference (LION)*, 2016.

[COW16] L. Climent, B. O'Sullivan, and R.J. Wallace. An improved metaheuristic algorithm for maximizing demand satisfaction in the population harvest cutting stock problem. In *Ninth Annual Symposium on Combinatorial Search*, 2016.

[CWOF16] L. Climent, R.J. Wallace, B. O'Sullivan, and E.C. Freuder. Extrapolating from limited uncertain information in large-scale combinatorial optimization problems to obtain robust solutions. *International Journal on Artificial Intelligence Tools*, 25(01):1660005, 2016.

[dAF12] R.C. de Amorim and T. Fenner. *Weighting Features for Partition around Medoids Using the Minkowski Metric*, pages 35–44. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[DBMS06] M. Didi-Biha, P. Marcotte, and G. Savard. *Path-based formulations of a bilevel toll setting problem*, pages 29–50. Springer US, Boston, MA, 2006.

[Deb00] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, 186(2-4):311–338, 2000.

[DF14] S. Dempe and S. Franke. *The bilevel road pricing problem.* TU Bergakademie, 2014.

[DF16] S. Dempe and S. Franke. On the solution of convex bilevel optimization problems. *Computational Optimization and Applications*, 63(3):685–703, 2016.

[DFH$^+$07] K.F. Doerner, G. Fuellerer, R.F. Hartl, M. Gronalt, and M. Iori. Metaheuristics for the vehicle routing problem with loading constraints. *Networks*, 49(4):294–307, 2007.

[DIM16] M. Delorme, M. Iori, and S. Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, 2016.

[DRF15] A. Dems, L-M. Rousseau, and J-M. Frayret. Effects of different cut-to-length harvesting structures on the economic value of a wood procurement planning problem. *Annals of Operations Research*, 232(1):65–86, 2015.

[DRF17] A. Dems, L-M. Rousseau, and J-M. Frayret. Annual timber procurement planning with bucking decisions. *European Journal of Operational Research*, 259(2):713–720, 2017.

[DS90] G. Dueck and T. Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161 – 175, 1990.

[DS12] J.V.S. Divvela and B.K. Sinha. Statistical aspects of forest harvesting: Price-weighted apportionment index and related inference. *Journal of Statistics Applications & Probability*, 1(1):53, 2012.

[Duf80] W.W. Duffner. Decision making from market to stump. In *Proceedings of Weyerhaeuser Science Symposium, Tacoma, Washington, USA*, pages 81–95, 1980.

[DX09] V. DeMiguel and H. Xu. A stochastic multiple-leader stackelberg model: analysis, computation, and application. *Operations Research*, 57(5):1220–1235, 2009.

[Dyc90] H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, 1990.

[DZ12] S. Dempe and A.B. Zemkoho. On the karush–kuhn–tucker reformulation of the bilevel optimization problem. *Nonlinear Analysis: Theory, Methods & Applications*, 75(3):1202–1218, 2012.

[EDW86] G. Eng, H.G. Daellenbach, and A.G.D. Whyte. Bucking tree-length stems optimally. *Canadian journal of forest research*, 16:1030–1035, 1986.

[EVR09] B. Eksioglu, A.V. Vural, and A. Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009.

[FDHI10] G. Fuellerer, K.F. Doerner, R.F. Hartl, and M. Iori. Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research*, 201(3):751–759, 2010.

[FM13] F. Furini and E. Malaguti. Models for the two-dimensional two-stage cutting stock problem with multiple stock size. *Computers & Operations Research*, 40(8):1953–1962, 2013.

[GG61] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

[GG63] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting stock problem-part ii. *Operations research*, 11(6):863–888, 1963.

[GG65] P.C. Gilmore and R.E. Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1):94–120, 1965.

[GH89] W.M. Getz and R.G. Haight. *Population harvesting: demographic models of fish, forest, and animal resources*, volume 27. Princeton University Press, 1989.

[HCL12] J.P. Hung, H.C. Chang, and Y.L. Lai. Implementation of on-line cutting stock problem on nc machines. In *Proceedings of World Academy of Science, Engineering and Technology*, number 69. World Academy of Science, Engineering and Technology, 2012.

[HFS96] L.C. Hendry, K.K. Fok, and K.W. Shek. A cutting stock and scheduling problem in the copper industry. *Journal of the Operational Research Society*, pages 38–47, 1996.

[HSB12] V.C. Hemmelmayr, V. Schmid, and C. Blum. Variable neighbourhood search for the variable sized bin packing problem. *Computers & OR*, 39(5):1097–1108, 2012.

[IBM17] IBM. User's manual of ibm cplex optimizer for z/os, 2017.

[IG98] R.R. Iyer and I.E. Grossmann. A bilevel decomposition algorithm for long-range planning of process networks. *Industrial & Engineering Chemistry Research*, 37(2):474–481, 1998.

[IM10] M. Iori and S. Martello. Routing problems with loading constraints. *Top*, 18(1):4–27, 2010.

[ISR16] M.M. Islam, H.K. Singh, and T. Ray. A memetic algorithm for solving bilevel optimization problems with multiple followers. In *IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 1901–1908, 2016.

[JOCM13] L. Junqueira, J.F. Oliveira, M.A. Carravilla, and R. Morabito. An optimization model for the vehicle routing problem with practical three-dimensional loading constraints. *ITOR*, 20(5):645–666, 2013.

[Kan60] L.V. Kantorovich. Mathematical methods of organizing and planning production. *Management Science*, 6(4):366–422, 1960.

[KHJ13] Y. Kristianto, P. Helo, and R.J. Jiao. Mass customization design of engineer-to-order products using benders' decomposition and bi-level stochastic programming. *J. Intelligent Manufacturing*, 24(5):961–975, 2013.

[KHRW16] H. Ke, H. Huang, D.A. Ralescu, and L. Wang. Fuzzy bilevel programming with multiple non-cooperative followers: model, algorithm and application. *Int. J. General Systems*, 45(3):336–351, 2016.

[Kiv04] V-P. Kivinen. A genetic algorithm approach to tree bucking optimization. *Forest Science*, 50(5):696–710, 2004.

[Kiv06] V-P. Kivinen. A forest-level genetic algorithm based control system for generating stand-specific log demand distributions. *Canadian Journal of Forest Research*, 36(7):1705–1722, 2006.

[KL51] S. Kullback and R.A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[KMR$^+$94] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R.E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 273–282. ACM, 1994.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, Jan 1982.

[Koh90] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep 1990.

[KR09] L. Kaufman and P.J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[KTB13] D.P. Kroese, T. Taimre, and Z.I. Botev. *Handbook of Monte Carlo Methods*. Wiley Series in Probability and Statistics. Wiley, 2013.

[KUN05] V-P. Kivinen, J. Uusitalo, and T. Nummi. Comparison of four measures designed for assessing the fit between the demand and output distributions of logs. *Canadian journal of forest research*, 35(3):693–702, 2005.

[Lap92] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[Lar99] A. Laroze. A linear programming, tabu search method for solving forest-level bucking optimization problems. *Forest Science*, 45(1):108–116, 1999.

[LG97] A. Laroze and B.J. Greber. Using tabu search to generate stand-level, rule-based bucking patterns. *Forest Science*, 43(2):157–169, 1997.

[LHHZ16] J. Lu, J. Han, Y. Hu, and G. Zhang. Multilevel decision-making: A survey. *Information Sciences*, 346-347(Supplement C):463 – 487, 2016.

[Liu98] B. Liu. Stackelberg-nash equilibrium for multilevel programming with multiple followers using genetic algorithms. *Computers & Mathematics with Applications*, 36(7):79–89, 1998.

[LO92] J. Lampinen and E. Oja. Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, 2(2):261–272, Nov 1992.

[LSZ06] J. Lu, C. Shi, and G. Zhang. On bilevel multi-follower decision making: General framework and solutions. *Inf. Sci.*, 176(11):1607–1627, 2006.

[LSZD07] J. Lu, C. Shi, G. Zhang, and T.S. Dillon. Model and extended kuhn-tucker approach for bilevel multi-follower decision making in a referential-uncooperative situation. *J. Global Optimization*, 38(4):597–608, 2007.

[LSZR05] J. Lu, C. Shi, G. Zhang, and D. Ruan. Multi-follower linear bilevel programming: model and kuhn-tucker approach. In *AC 2005,* *Proceedings of the IADIS International Conference on Applied Computing, Algarve, Portugal, February 22-25, 2005, 2 Volumes*, pages 81–88, 2005.

[LSZR07] J. Lu, C. Shi, G. Zhang, and D. Ruan. An extended branch and bound algorithm for bilevel multi-follower decision making in a referential-uncooperative situation. *International Journal of Information Technology and Decision Making*, 6(2):371–388, 2007.

[LW10] H. Li and Y. Wang. An evolutionary algorithm based on a new decomposition scheme for nonlinear bilevel programming problems. *IJCNS*, 3(1):87–93, 2010.

[LZC+16] H. Li, Q. Zhang, Q. Chen, L. Zhang, and Y-C. Jiao. Multiobjective differential evolution algorithm based on decomposition for a type of multiobjective bilevel programming problems. *Knowl.-Based Syst.*, 107:271–288, 2016.

[MA11] S.A. MirHassani and N. Abolghasemi. A particle swarm optimization algorithm for open vehicle routing problem. *Expert Systems with Applications*, 38(9):11547–11551, 2011.

[MAA06] G. Murphy, M. Acuna, and D. Amishev. Adaptive control of bucking on harvesters: Target and timing effects. *Forest products journal*, 56(11/12):79, 2006.

[Mar72] G. Marsaglia. Choosing a point from the surface of a sphere. *Ann. Math. Statist.*, 43(2):645–646, 04 1972.

[MB86] G.A. Mendoza and B.B. Bare. A two-stage decision model for log bucking and allocation. *Forest Products Journal*, 36(10):70–74, 1986.

[MMB04] G. Murphy, H. Marshall, and M.C. Bolding. Adaptive control of bucking on harvesters to meet order book constraints. *Forest Products Journal and Index*, 54(12):114–121, 2004.

[MMB06] H.D. Marshall, G. Murphy, and K. Boston. Three mathematical models for bucking-to-order. *Silva Fennica*, 40(1):127, 2006.

[MP04] J. Malinen and T. Palander. Metrics for distribution similarity applied to the bucking to demand procedure. *International Journal of Forest Engineering*, 15(1):33–40, 2004.

[MRS$^+$15] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, K. Hornik, M. Studer, and Roudier. cluster: Cluster analysis extended rousseeuw et al r package version 2.0.1. 2015.

[MRS$^+$17] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik. *cluster: Cluster Analysis Basics and Extensions*, 2017. R package version 2.0.6 — For new features, see the 'Changelog' file (in the package source).

[MRWR12] L. Miao, Q. Ruan, K. Woghiren, and Q. Ruo. A hybrid genetic algorithm for the vehicle routing problem with three-dimensional loading constraints. *RAIRO - Operations Research*, 46(1):63–82, 2012.

[Mul59] M.E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, 2(4):19–20, April 1959.

[Nag15] P.S. Nagpaul. "7.1.2 clustering large applications (clara)" in guide to advanced data analysis using idams software. 2015.

[NHG11] T. Nishi, Y. Hiranaka, and I.E. Grossmann. A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. *Computers & OR*, 38(5):876–888, 2011.

[NMMea04] F.J. Nogales Martín, A.V. Miguel, and et al. On the relationship between bilevel decomposition algorithms and direct interior-point methods. Technical report, Universidad Carlos III de Madrid. Departamento de Estadística, 2004.

[OL08] J. Oppen and A. Løkketangen. A tabu search approach for the livestock collection problem. *Computers & Operations Research*, 35(10):3213–3229, 2008.

[PA09] K.C. Poldi and M.N. Arenales. Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths. *Computers & OR*, 36(6):2074–2081, 2009.

[PFCO15] S.D. Prestwich, A.O. Fajemisin, L. Climent, and B. O'Sullivan. *Solving a Hard Cutting Stock Problem by Machine Learning and Optimisation*, pages 335–347. Springer International Publishing, Cham, 2015.

[PM72] S.M. Pnevmaticos and S.H. Mann. Dynamic programming in tree bucking. *Forest Products Journal*, 22:26–30, 1972.

[PM95] S. Peeta and H.S. Mahmassani. Multiple user classes real-time traffic assignment for online operations: a rolling horizon solution framework. *Transportation Research Part C: Emerging Technologies 3.2, pp 83-98*, 1995.

[RBA+16] M.A Ramos, M. Boix, D. Aussel, L. Montastruc, and S. Domenech. Water integration in eco-industrial parks using a multi-leader-follower approach. *Computers & Chemical Engineering*, 87(Supplement C):190 – 207, 2016.

[RBH14] G.R. Raidl, T. Baumhauer, and B. Hu. *Speeding Up Logic-Based Benders' Decomposition by a Metaheuristic for a Bi-Level Capacitated Vehicle Routing Problem*, pages 183–197. Springer International Publishing, Cham, 2014.

[RHW06] J. Rybarski and S. Habdank-Wojewódzki. Java kohonen neural network library (jknnl), 2006.

[RK16] R.Y. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016.

[SI09] G.K. Saharidis and M.G. Ierapetritou. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization*, 44(1):29–51, 2009.

[SLZZ05] C. Shi, J. Lu, G. Zhang, and H. Zhou. An extended kuhn-tucker approach for linear bilevel multifollower programming with partial shared variables among followers. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, Hawaii, USA, October 10-12, 2005*, pages 3350–3357, 2005.

[SMD13] A. Sinha, P. Malo, and K. Deb. Efficient evolutionary algorithm for single-objective bilevel optimization. *CoRR*, abs/1303.3901, 2013.

[SMD14] A. Sinha, P. Malo, and K. Deb. An improved bilevel evolutionary algorithm based on quadratic approximations. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1870–1877, July 2014.

[SMD17a] A. Sinha, P. Malo, and K. Deb. A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications, 2017. arXiv:1705.06270v1.

[SMD17b] A. Sinha, P. Malo, and K. Deb. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research*, 257(2):395 – 411, 2017.

[SMFD14] A. Sinha, P. Malo, A. Frantsev, and K. Deb. Finding optimal strategies in a multi-period multi-leader–follower stackelberg game using an evolutionary algorithm. *Computers & Operations Research*, 41:374–385, 2014.

[SNI+12] T. Sugiyama, T. Nishi, M. Inuiguchi, S. Takahashi, and K. Ueda. A bilevel decomposition approach to railway crew rostering problems for fair labor condition. In *2012 IEEE International Conference on Automation Science and Engineering, CASE 2012, Seoul, Korea (South), August 20-24, 2012*, pages 383–389, 2012.

[SOG89] J. Sessions, E. Olsen, and J. Garland. Notes: Tree bucking for optimal stand value with log allocation constraints. *Forest Science*, 35(1):271–276, 1989.

[Sol84] M.M. Solomon. Vehicle routing and scheduling with time window constraints: Models and algorithms. Technical report, 1984.

[SPS04] M. Sigurd, D. Pisinger, and M. Sig. Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, 38(2):197–209, 2004.

[Sta34] H von Stackelberg. Marktform und gleichgewicht. wien, 1934.

[SZL05] C. Shi, G. Zhang, and J. Lu. The K th-best approach for linear bilevel multi-follower programming. *J. Global Optimization*, 33(4):563–578, 2005.

[SZL+07] C. Shi, H. Zhou, J. Lu, G. Zhang, and Z. Zhang. The kth-best approach for linear bilevel multifollower programming with partial shared variables among followers. *Applied Mathematics and Computation*, 188(2):1686–1698, 2007.

[Tal13] E-G. Talbi. *Metaheuristics for bi-level optimization*, volume 482. Springer, 2013.

[TV14] P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[Uus05] J. Uusitalo. A framework for ctl method-based wood procurement logistics. *International Journal of Forest Engineering*, 16(2):37–46, 2005.

[Van07] R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. International Series in Operations Research & Management Science. Springer US, 2007.

[VFIP96] V. Visweswaran, C.A. Floudas, M.G. Ierapetritou, and E.N. Pistikopoulos. A decomposition-based global optimization approach for solving bilevel linear and quadratic programs. *State of the art in global optimization*, 139, 1996.

[WHS07] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.

[WLH00] C-P. Wei, Y-H. Lee, and C-M. Hsu. Empirical comparison of fast clustering algorithms for large data sets. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2000.

[WWW09] G. Wang, X. Wang, and Z. Wan. A fuzzy interactive decision making algorithm for bilevel multi-followers programming with partial shared variables among followers. *Expert Syst. Appl.*, 36(7):10471–10474, 2009.

[ZA14] B. Zeng and Y. An. Solving bilevel mixed integer program by reformulations and decomposition. *Optimization online*, pages 1–34, 2014.

[ZL10] G. Zhang and J. Lu. Fuzzy bilevel programming with multiple objectives and cooperative multiple followers. *J. Global Optimization*, 47(3):403–419, 2010.

[ZLD07] G. Zhang, J. Lu, and T.S. Dillon. Models and algorithm for fuzzy multi-objective multi-follower linear bilevel programming. In *FUZZ-*

IEEE 2007, IEEE International Conference on Fuzzy Systems, Imperial College, London, UK, 23-26 July, 2007, Proceedings, pages 1–6, 2007.

[ZLG08a] G. Zhang, J. Lu, and Y. Gao. An algorithm for fuzzy multi-objective multi-follower partial cooperative bilevel programming. *Journal of Intelligent and Fuzzy Systems*, 19(4-5):303–319, 2008.

[ZLG08b] G. Zhang, J. Lu, and Y. Gao. Fuzzy bilevel programming: Multi-objective and multi-follower with shared variables. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 16(Supplement-2):105–133, 2008.

[ZSL08] G. Zhang, C. Shi, and J. Lu. An extended Kth-best approach for referential-uncooperative bilevel multi-follower decision making. *Int. J. Computational Intelligence Systems*, 1(3):205–214, 2008.