Solution Techniques for Bi-level Knapsack Problems and Application in Allocation of Healthcare Funds Problem

Shraddha Rahul Ghatkar

Department of Management Science

University of Strathclyde

Glasgow, UK

2024

A thesis submitted in fulfilment of the requirement for the degree of Doctor of Philosophy

Declaration

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Signed: Shraddha Rahul Ghatkar
Date: April 9, 2024

Acknowledgements

First and foremost I would like to express my deepest gratitude to my supervisors, Dr Ashwin Arulselvan and Prof Alec Morton. Their expertise, knowledge sharing, and consistent support have benefited me greatly in my PhD journey. I am grateful for their invaluable supervision in helping me become a better researcher.

I am grateful to the University of Strathclyde for funding the research studentship and providing the academic facilities through which I was able to complete my PhD research.

I would like to thank Prof Ashutosh Mahajan who appreciated the researcher in me and motivated me to pursue a PhD. I acknowledge the support I have received from colleagues in the Business School, both past and present. Special thanks to Öykü, Glory, Nicola, Vivek, and Alex here. Having conversations with them has given me moral support and encouragement, especially during the difficult time of the covid pandemic.

I owe a great deal of gratitude to my family for their love, support, and constant prayers that have been instrumental in helping me succeed on this journey. To my late grandmother, thank you for instilling in me the value of lifelong learning and consistency. To my parents and brother, thank you for always being there for me and taking care of me. Hope, I made you proud. To my husband, I realize and appreciate that my PhD journey has been a roller coaster ride, and you were always next to me. To my beloved daughters, you are the source of my strength and resilience always.

Publication

Part of this dissertation has been published in peer-reviewed journal, as listed below:

Ghatkar, S., Arulselvan, A. & Morton, A. (2023), 'Solution techniques for bi-level knapsack problems', *Computers and Operations Research* **159**, 106343

To my family

Abstract

In consideration to the significant healthcare funds given by donors, both rich countries and philanthropic organizations, it is important to allocate this aid money in an effective way. The conventional mechanism of healthcare funds allocation to the projects is based on cost-effectiveness, *i.e.* the projects are ranked in their profit to cost ratios and are prioritized based on this ranking. An alternative approach of allocating subsidies to projects that are just cost-ineffective to a country rather than funding entire projects that are cost-effective has been proposed in the literature. This approach will not only encourage the country to contribute its resources but it will promote ownership of projects that are subsidized from outside.

A Bi-level Programming Problem (BLPP) has been used to represent this approach wherein there are two participants - a leader (donor agency) and a follower (recipient country). In this specific BLPP, referred to as Donor-Recipient Bi-level Knapsack Problem (DR-BKP) in our work, the donor has choice to subsidize healthcare projects that are implemented by the recipient country and the country has choice to take these subsidies for project selection using the further funding. There is a set of healthcare projects, each one associated with a certain profit and cost, under consideration by both participants. The cost of every project is common to the participants however the profit values may differ for them. Along with the healthcare projects, the country has an external project that is of no interest to the donor in this setup. Once the donor decides on cost subsidies for the healthcare projects that are within its individual budget, the country solves a knapsack problem with the cost subsidized projects and the external project constrained by its own budget.

Starting with an introduction to BLPPs and motivation of application in a healthcare economics problem, we present the DR-BKP in chapter 1. In chapter 2, we give the literature reviewed for the BLPPs and their solution algorithms. In chapter 3, we provide evidence for Σ_2^P-hardness by showing that the DR-BKP is both NP-hard and Co-NP hard. After showing the existence of a solution for the DR-BKP, we give a pair of finitely converging exact algorithms, an enumeration algorithm and a branching algorithm, and show the convergence of these algorithms. A set of fifteen differing data sets, each having randomly generated ten instances, have been generated and solved to evaluate the performance of the proposed algorithms. These data sets are generated such that they mimic a range of instances arising in real-life, *i.e.* starting from the ones that can be trivially addressed to the ones that are complex and difficult to solve. The complexity of some of these data sets is characterized firstly by the external project having higher valuation than the healthcare projects, and then by the discrepancy in the healthcare project valuation done by the two players. From the results of the computational experiments, it can be seen that the branching algorithm performs better when the instances are complex.

We give a nested sequential approach in chapter 4 for addressing the DR-BKP, wherein the donor problem is solved using a genetic algorithm and the parameterized country problem is solved using a heuristic or an exact solver. This is followed by computational experiments of solving the fifteen data sets generated in chapter 3 using the genetic algorithm and its results. At the end of chapter 4, we summarize the performance of all the three algorithms over the fifteen data sets. The exact algorithms perform better to solve the data sets with external project having higher valuation than the healthcare projects, whereas the genetic algorithm performs better to solve the data sets having discrepancies in the healthcare project valuation by the two players.

In chapter 5, we present generalizations of the proposed solution algorithms to other bi-level optimization frameworks that are closer to realistic scenarios of the application, like a single leader having multiple followers. Finally, we conclude the work done and give directions for future research in chapter 6.

Keywords: Bi-level Programming Problems, Bi-level Knapsack Problems, Healthcare Economics, Nested Sequential Approach, Genetic Algorithm, Heuristics

Abbreviations

BLPP Bi-level Programming Problem

BKP Bi-level Knapsack Problem

DR-BKP Donor-Recipient Bi-level Knapsack Problem

MILP Mixed-Integer Programming Problem

ILP Integer Linear Programming Problem

DP Dynamic Programming

KKT Karush-Kuhn-Tucker

PSO Particle Swarm Optimization

EBO Evolutionary Bi-level Optimization

HPR High Point Relaxation

ICER Incremental Cost-Effectiveness Ratio

QALY Quality Adjusted Life Year

DALY Disability Adjusted Life Year

R-DR-BKP Relaxed Donor-Recipient Bi-level Knapsack Problem

Abbreviations

DR-BKP-NH Donor-Recipient Bi-level Knapsack Problem with Non-Healthcare projects

R-DR-BKP-NH Relaxed Donor-Recipient Bi-level Knapsack Problem with Non-Healthcare projects

DR-BKP-RPC Donor-Recipient Bi-level Knapsack Problem with Recipient profit as a Piece-wise linear Concave function

DR-BKP-RPC Relaxed Donor-Recipient Bi-level Knapsack Problem with Recipient profit as a Piece-wise linear Concave function

D-mR-BKP Donor-multiple Recipients Bi-level Knapsack problem

D-2R-BKP Donor-two Recipients Bi-level Knapsack problem

R-D-2R-BKP Relaxed Donor-two Recipients Bi-level Knapsack problem

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A Bi-level Programming Problem (BLPP) consists of two optimization models in a single instance. One optimization model is a part of the constraint set of another optimization model. This creates a relationship between the two models since a solution to one model makes an effect on another model. This rightly mimics the desired hierarchical relation between two classes of decision-makers in a setup. Typically, a central planner (or leader), with collective utility as an objective, makes an investment decision based on which one or more agents (or followers) make decisions that maximize their individual utility. Also, any behaviour made by the follower(s) is part of the constraints of the leader and any decision made by either of the participants makes an impact on the decisions made by the other participants. In the literature, the hierarchical decision-making framework can be seen to originate from two domains. One is from the domain of game theory, wherein Stackelberg (1934) developed decision behavior models and established game-theoretic equilibria. Many bi-level optimization problems we see today are originally proposed and studied as Stackelberg games in game theory when there is a hierarchy in decision-making among the players (Kleinert et al. 2021). The other origin is from the domain of mathematical programming, wherein Bracken & McGill (1973) introduced the nested structure of optimization problems as BLPPs.

Usually, there are variables in both levels of a BLPP. The leader problem is modeled at the upper-level and the follower(s) problem is modeled at the lower-level. There are

several real-life decision-making problems that have more than one decision-makers, wherein bi-level optimization framework can be used to model these problems and get valuable solutions and/or insights for supporting the decision-makers.

A general formulation of a BLPP (Colson et al. 2005*a*) can be given as follows:

$$\underset{\mathbf{y}\in Y,\ \mathbf{x}\in X}{\text{maximize}} \quad f(\mathbf{y},\mathbf{x}) \tag{1.1a}$$

$$\text{subject to} \quad F(\mathbf{y},\mathbf{x}) \leq 0 \tag{1.1b}$$

$$\mathbf{x} \in \ \arg\max_{\mathbf{x}'\in X}\{g(y,x') : G(\mathbf{y},\mathbf{x}') \leq 0\} \tag{1.1c}$$

where, the upper-level decision variable is $\mathbf{y} \in Y \subseteq \mathbb{R}^m$, and the lower-level decision variable is $\mathbf{x} \in X \subseteq \mathbb{R}^n$. The leader and the follower have their own objective functions, $f, g : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$, and constraint functions, $F : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^p$ and $G : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^q$.

Decision-making is sequentially done, first by the leader and then by the follower. Once the leader makes a decision on $\mathbf{y}$, the follower solves an optimization problem parameterised by the leader's decision.

There are two modeling approaches to a BLPP. One is an optimistic bi-level programming approach where the leader is able to influence the follower's choice whenever there are multiple options. Since this depends on the coordination of both participants, it is considered a strong Stackelberg solution. In the case where there is no coordination and/or the leader is risk-averse and wants to limit the negative impact from an undesirable choice done by the follower, it is a pessimistic bi-level programming approach and considered a weak Stackelberg solution.

There are several real-life problems modeled as BLPPs. Côté et al. (2003) have developed a BLPP for a toll-setting problem in the transportation domain. The upper-level decision-makers here are an authority or owners of the highway development who need to find a set of tolls to maximize their profit. However, they need to consider that the highway users (lower-level decision-makers) will choose their itinerary in such a way that they minimize their total cost of travel. Another BLPP has been developed by Arroyo & Fernández (2013) to assess power system vulnerability. The upper-level in

this setup finds a set of simultaneous outages in the power system and the lower-level finds an optimal power system operation under contingency with respect to the outages in the upper-level. Morton et al. (2018) have developed a BLPP for a healthcare economics problem. The model has a donor agency at the upper-level that aims to find a set of subsidies to healthcare projects in a recipient country (lower-level decision-maker) to maximize its health-related profit. Depending on the received subsidies for each project, the country then finds its optimal set of healthcare projects to maximize its health-related profit. Apart from these, there are real-life applications in national agricultural planning by Fortuny-Amat & McCarl (1981), transportation network design by Constantin & Florian (1995), systematic search of hyper-parameters in machine learning by Bennett et al. (2006), energy and electricity markets by Wogrin et al. (2020) and many others. Although the application areas are wide, there are not many implementations seen due to a lack of efficient algorithms to deal with actual problem sizes in real-life. Hence a lot of attention has been shifted recently to solving these extremely challenging set of problems.

The difficulty of these problems can be realized from the fact that even in the simplest case of linear BLPPs, wherein the objective functions and the constraints at both levels are linear, the problem is NP-hard (Jeroslow 1985). In order to merely evaluate a solution to these problems, one has to solve an NP-hard problem (Vicente et al. 1994). Bard (1984) proves that solving the linear BLPPs is equivalent to maximizing a linear function over a piece-wise linear constraint set and gives necessary first-order optimality for general bi-level programs. Other than linear BLPPs, there are different structures of BLPPs seen in the literature based on (a) the linearity or convexity of the objective functions and/or constraints in both levels, (b) variables being continuous and/or discrete in both levels and (c) occurrence of upper-level variables in the lower-level problem (Colson et al. 2005*a*, Mersha & Dempe 2006). There are BLPPs that have quadratic objective functions and linear constraints. Bard & Moore (1990*b*) give a Branch-and-Bound solution methodology to solve these. The nonlinear BLPPs are solved by Savard & Gauvin (1994) using the method of steepest descent direction and by Colson et al. (2005*a*) using a trust-region algorithm.

Another class of BLPPs that is largely studied is the mixed-integer BLPPs. Some or all the variables in either of the levels or both levels of the BLPPs are discrete in this type. These are a difficult class of problems to solve. The Mixed-Integer Programming Problem (MILP) Branch-and-Bound cannot be simply generalized for mixed-integer BLPPs since dropping the integrality constraints in the mixed-integer BLPPs does not yield a valid relaxation (DeNegre 2011). There are many sub-types of mixed-integer BLPPs seen in the literature depending on which level the discrete variables appear, if the variables are binary, and if the participants in both levels are working at the same or contradicting objectives. These also vary based on the particular applications for which the mixed-integer BLPPs have been developed. A general mixed-integer BLPP has been introduced by Moore & Bard (1990), where they present the challenges involved in solving this formulation.

In this thesis, the focus is on Bi-level Knapsack Problems (BKPs) which are a special type of mixed-integer BLPPs. It has been applied to a real-life problem in healthcare economics. A detailed motivation of this study has been given in section 1.1 along with a description of the healthcare economics problem. The BKPs are introduced in section 1.2, followed by an introduction to the Donor-Recipient Bi-level Knapsack Problem (DR-BKP) and the research contribution of this thesis in section 1.3. An outline of the thesis is given in section 1.4.

1.1 Motivation

There has been significant progress made in the direction of preventing and treating diseases in our human race all across the world (UnitedNations 2015). This advancement can be attributed not only to the technological progress in medical science but also to the aid money allocated for healthcare projects (henceforth, referred to as "project") conducted all over the world. The projects are primarily funded by the recipient country, where they are implemented, in partnership with financing organizations like the Global Fund. The funding of these organizations is from nations with large gross national income and philanthropic organizations.

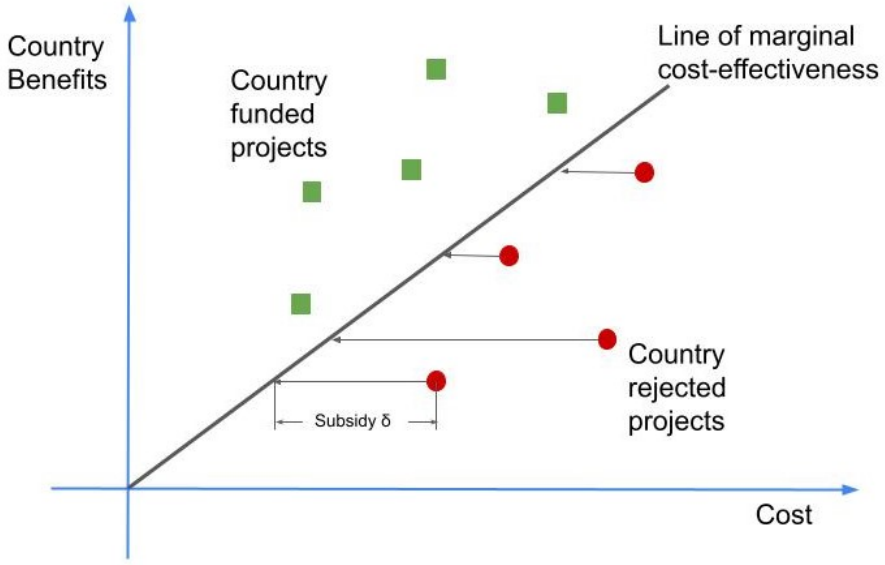


Figure 1.1: Project selection by recipient country and donor subsidies (Adapted from Morton et al. (2018))

The recipient is often a developing country that manages its budget for healthcare projects alongside numerous other projects such as education, welfare, infrastructure, defense, etc. The aid money available through the donors is limited as well. The world population is rising and so is the potential impact of diseases, as can be realized from the recent COVID pandemic that struck the world in 2019. Hence, it is important to identify an efficient and fair mechanism to allocate the funds available between the donor and the recipient to these projects.

The traditional and most followed approach is to rank the projects based on their benefit-to-cost ratio and prioritize projects that have a higher ratio. This approach is more like "value for money". However, Morton et al. (2018) claim that this approach results in crowding out of indigenous financing of interventions, and thus results in under-allocation of resources to healthcare. They instead have proposed a novel approach for the donor to allocate subsidies to projects that are just cost-ineffective to a country *i.e.* the projects that have just missed a chance to be funded by the country itself will be subsidized by the donor agency and pulled to the level of marginal projects. Marginal projects are the projects that are eligible to be funded by the country itself.

Figure 1.1 shows the line of marginal cost-effectiveness (v_0/c_0) where v_0 represents the value of all the non-healthcare projects together for the country and c_0 represents the cost of these projects together. The green squares represent the projects funded by the country, and the red circles represent the projects that can be subsidized and brought to the marginal line of cost-effectiveness. The given approach assures not only efficient allocation of the available funds but is also in line with the idea of "sustainable" aid.

The overall objective of the system here is to optimally allocate aid money available to the donor and the recipient country amongst the healthcare projects to maximize the health-related profits of both participants. The analogy of a donor-recipient country is captured by the leader-follower strategic games. These games are modeled and solved using BLPPs. The upper-level here is the donor and the lower-level is the recipient country. The subsidies given by the donor for each of the projects make an impact on the project selections done by the country and also the projects selected by the country affect the decisions of the donor. This hierarchical relationship amongst these participants emphasizes the suitability of developing a BLPP for this allocation problem. Also, the healthcare projects have different valuations from the perspectives of donor agency and recipient countries and this can be well captured with BLPPs. The preliminary findings of Morton et al. (2018) have shown promise in relaxing certain assumptions made in this paper and in developing solution techniques for this challenging and intriguing modeling framework.

The donor-recipient country relationship for healthcare funds can be generalized to many other realistic applications. For example, the donor in this setup can subsidize projects related to education in a recipient country. Considering the hierarchical decision-making in a federal government like India, the union-states relationship can be modeled using BLPPs for different types of funding like education, healthcare, and infrastructure. Also, the headquarters-regional offices' relationship in any government or private organization can be captured using BLPPs for their revenue management. Similarly, there can be many other applications seen in reality where there is a central leader who subsidizes items/projects of its interest to influence the follower(s) decisions (see DeNegre (2011), Dame & Nüsser (2011), Muraközy & Telegdy (2016), Finkelstein

et al. (2022) for this type of applications of the BLPPs).

1.2 Bi-level knapsack problems

The BKPs are a type of mixed-integer BLPPs that have a knapsack at either or both levels in the formulation. This setup is well suited to capture the framework of financing problems where more than one participant or beneficiary is involved. For example, as given by Brotcorne et al. (2009), an individual shares his/her capital in both a savings account with a fixed rate of return and a risky investment through a broker to maximize his/her profit. The individual here is a leader with a knapsack budget and the broker is a follower who wants to find the right investment options to maximize his/her profit through the returns within the leader's budget.

In general cases, the budget of the follower's knapsack problem gets determined by the leader's problem. However, there are several different formulations possible of a BKP. Depending on the structure of a BKP, there are three different variants of these as suggested by Caprara et al. (2014) and Carvalho (2016), *viz.* (1) the Dempe-Ritcher variant, (2) the Mansi-Alves-de-Carvalho-Hanafi variant, and (3) the DeNegre variant. A detailed description of the literature reviewed for these BKP variants along with some of their extensions recently seen in the literature has been given in chapter 2. In the next section, we give the specific BKP developed for the healthcare economics problem described in section 1.1.

1.3 Donor-Recipient bi-level knapsack problem

The BKP formulated for the healthcare economics problem proposed by Morton et al. (2018) is referred to as the Donor-Recipient Bi-level Knapsack Problem (DR-BKP) in this thesis. The DR-BKP has a knapsack constraint at both upper and lower-levels as given in (1.2) and (1.3), where upper-level is the donor problem `(DONOR)` and lower-level is the recipient problem `(RECIPIENT(y))` parameterized on the upper-level decision $\mathbf{y}$. The follower solves a knapsack problem that is influenced by decisions made by the leader's knapsack problem. Due to the nature of the application, cooperation is

assumed and we discuss optimistic strategies only.

Problem `DONOR`:

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{1.2a}$$

$$\text{subject to} \quad \mathbf{y} \in Y \tag{1.2b}$$

$$\mathbf{x} \in \arg\max(\texttt{RECIPIENT(y)}), \tag{1.2c}$$

Problem `RECIPIENT(y)`:

$$\text{maximize} \quad \mathbf{v}^T\mathbf{x} + v_0 x_0 \tag{1.3a}$$

$$\text{subject to} \quad \sum_{i \in I}(c_i - c_i y_i)x_i + c_0 x_0 \leq B_r \tag{1.3b}$$

$$(\mathbf{x}, x_0) \in X, \tag{1.3c}$$

where, I is a set of n projects, $I = \{1, \ldots, n\}$, that are common to both players. Each project $i \in I$, has a profit of $w_i \in \mathbb{N}$ (resp. $v_i \in \mathbb{N}$) for the donor (resp. recipient), and a cost $c_i \in \mathbb{N}$. Let $\mathbf{w}$ and $\mathbf{v}$ denote the vectors of profits of the donor and the recipient respectively and $\mathbf{c}$ be the vector of costs of these projects. We have two integer budgets, B_d and B_r, corresponding to the donor and the recipient. Besides the projects in I, the recipient has to allocate its budget to an outside option of projects. This represents a portfolio of projects that is of no interest to the leader. We will refer to this option as an "external project". The external project has a linear profit and linear cost of v_0 and c_0 respectively as per the original formulation (Morton et al. 2018). The objective is to pick a subset of items (either fractionally or wholly) such that their costs are within the budget and the profit is maximized.

So an instance of the DR-BKP is specified by the input $(\mathbf{w}, \mathbf{v}, \mathbf{c}, v_0, c_0, B_d, B_r)$. The recipient solves a knapsack problem, where each item of the knapsack corresponds to a project $i \in I$ with a profit v_i and cost $(c_i - c_i y_i)$, where y_i is the proportion of the cost of project i that is subsidized by the donor. These projects are binary and cannot be fractionally picked. Along with the healthcare projects, the recipient has to fund

the external project which can be done fractionally or wholly *i.e.* only a proportion or entire cost of the external project can be funded by the recipient.

A solution to an instance of the DR-BKP is to decide on the proportion of the cost to be subsidized, y_i, for each project, $i \in I$ with $\sum_{i \in I} c_i y_i \leq B_d$ such that profit of the donor is maximized given the projects are in the optimal solution set of the recipient's cost subsidized knapsack problem. We use the notation $\mathbf{y}$ to denote a vector of subsidy. The leader cannot subsidize a project more than its cost and the total subsidy cannot exceed the leader's budget. The set of all valid subsidies is denoted by $Y := \Big\{\mathbf{y} : \sum_{i \in I} c_i y_i \leq B_d, \mathbf{y} \in [0,1]^n\Big\}$.

We let $\mathbf{x}$ to denote a 0-1 vector representing the set of projects that are picked (i^{th} component of the vector, x_i, is 1 if project i is picked and 0 otherwise) and x_0 to denote the proportion of the cost of the external project that is being funded by the recipient. We define the set $X := \Big\{(\mathbf{x}, x_0) : \mathbf{x} \in \{0,1\}^n, x_0 \in [0,1]\Big\}$.

This modeling framework rightly captures the relationship between the donor and the recipient country. The donor problem finds the right amount of subsidies for the healthcare projects with respect to the donor's budget and the optimal solution obtained for the recipient country's problem while maximizing the valuation from the donor's perspective. The recipient's problem maximizes the valuation again but from its own perspective. It selects projects from a range of healthcare projects after considering the amount of subsidy allocated to the donor problem and with respect to the recipient's budget. The external project of the recipient is also a contending option for funding in the lower-level problem.

In Morton et al. (2018), the authors have considered a set of assumptions to carry out preliminary analysis and computational experiments with the DR-BKP. These assumptions are as follows:

1. the external project has a linear profit and cost function so that the lower-level problem is relatively easier to handle, otherwise the DR-BKP would be more difficult to solve.

2. the donor and recipient budgets are fixed although the recipient budgets tend to

decrease especially as the donor funds are received.

3. the projects are implemented independently such that there are no shared benefits coming from these projects or costs associated.

4. the recipient country is middle-income such that it can fund all its healthcare projects from its budget. However, this budget is less than the cost of the external budget, *i.e.* $\sum_{i \in I} c_i < B_r < c_0$.

As per the last assumption above and by restricting the subsidy values in the upper-level problem to $\left(c_i - v_i \frac{c_0}{v_0}\right) \ \ \forall i \in I$, Morton et al. (2018) have given a reformulated single-level knapsack problem that is equivalent to the DR-BKP. This reformulation is easy to solve using standard optimization solvers. The results have given interesting insights towards the funding decisions made by the two participants in the system. However, the assumption is quite strong from a practical perspective and does not capture all application contexts. Numerically, without this assumption, we are unlikely to get a single-level knapsack reformulation as we have shown strong evidence for Σ_2^P-hardness in subsection 3.2.1.

In this thesis, we address the above-mentioned assumption number 4 partly, the recipient country is middle income *i.e.* $\sum_{i \in I} c_i < B_r$. This makes the problem more generic and has a wider applicability. To relax this assumption, the subsidies from the upper-level problem, which are continuous variables, appear in the lower-level problem. This results in the problem getting more complicated to solve. These types of mixed-integer BLPPs tend to be ill-posed and might result in no solution (Vicente et al. 1996, Köppe et al. 2010). The issue of ill-posedness is not true and the existence of a solution can be guaranteed for the DR-BKP, we have discussed this in section 3.2. However, there are not many numerical procedures seen in the literature for such problems. Hence, solution algorithms developed for the DR-BKP can be generalized to handle other mixed-integer BLPPs, especially BKPs, having continuous variables in the upper-level problem and mixed-integer lower-level problem where at least the problem is not ill-posed.

However, the assumption that the recipient budget is always less than the cost of

the external project *i.e.* $B_r \leq c_0$ still holds in this thesis. The rationale for this has been given in section 3.3.

Research contribution

The main contribution of this thesis are:

- Two finitely converging exact algorithms have been given for the Donor-Recipient Bi-level Knapsack Problem (DR-BKP), an enumeration algorithm based on optimal-value-function reformulation and a branching technique that eliminates large regions that are not bi-level feasible. These algorithms solve the DR-BKP when one of the important assumptions made by Morton et al. (2018), the recipient country is middle-income *i.e.* $\sum_{i \in I} c_i < B_r$, has been relaxed. The generalized DR-BKP has continuous variables in the upper-level problem that appear in the lower-level problem. Such class of mixed-integer BLPPs are difficult to solve. Most of the literature for mixed-integer BLPPs either assumes that there are no continuous variables in the upper-level problem or that the upper-level continuous variables do not appear in the lower-level problem. This algorithmic development is a novel pair of solution techniques for the mixed-integer BLPPs, especially BKPs, with continuous variables in the upper-level problem that appear in the lower-level problem. A set of differing data sets are generated based on real-life instances of the healthcare economics problem in order to test and compare the performance of the developed algorithms.

- A genetic algorithm having a nested sequential approach has been developed to solve the DR-BKP. The two levels of the problem are decoupled such that the upper-level is solved using a genetic algorithm, followed by separately solving the parameterized lower-level problem either using an exact solver or a heuristic solution method. It has been tested on a set of data sets solved completely by the exact solvers. The genetic algorithm has then been used to solve the instances that could not be completely solved by the exact solvers in a reasonable time. Finally, we have compared the performance of all the developed algorithms based

on the data sets.

1.4 Outline of thesis

Following from the introduction to the thesis given in this chapter, a comprehensive literature review of the BKPs and solution techniques for these has been given in chapter 2. In chapter 3, the DR-BKP has been presented along with two exact solvers *i.e.* (1) an enumeration algorithm and (2) a branching algorithm. This is followed by the computational experiments we performed to understand the performance of these algorithms. The genetic algorithm for the DR-BKP has been presented in chapter 4 along with the results of computational experiments. In chapter 5, we have presented the generalization of the DR-BKP along with the potential ideas in which the given model formulations can be addressed. We have concluded the work and have given future research directions in chapter 6.

Chapter 2

Literature review

The Bi-level Programming Problems (BLPPs) have received a lot of interest recently due to the suitability of using this framework of problems in real-life applications. The BLPPs are a complex class of problems and hence intriguing for many researchers around the world. Some of the classical approaches for developing algorithms and solving BLPPs can be categorized into four types, (1) reformulating a BLPP into a single-level problem and solving with an appropriate method (Fortuny-Amat & McCarl 1981, Bialas & Karwan 1984, Bard & Moore 1990*a*, Edmunds & Bard 1991, Shi et al. 2005), (2) using descent methods to search the solution space (Kolstad & Lasdon 1990, Savard & Gauvin 1994, Vicente et al. 1994, Solodov 2007), (3) using penalty function methods and solving a series of optimization problems having these penalty terms (Aiyoshi & Shimizu 1981, 1984, Ishizuka & Aiyoshi 1992, Lv et al. 2007), and (4) using iterative trust-region methods to refine the search space (Liu et al. 1998, Marcotte et al. 2001, Colson et al. 2005*b*). The latest and by far the largest annotated list of references can be found in the book by Dempe & Zemkoho (2020).

In this chapter, we have first given a review of the literature focused on the problem of allocation of funds by donor agencies to healthcare projects and different methods used to address this in section 2.1. This section is followed by a detailed literature review of the mixed-integer BLPPs in section 2.2 and that of the Bi-level Knapsack Problems (BKPs) in particular in section 2.3. A literature review of the heuristic methods to solve general BLPPs has been given in section 2.4.

2.1 Allocation of healthcare funds

The success and sustainability of healthcare projects depend on an increased recipient country contribution along with donor funding (Heller 2005). Continuing aid-funded health programs will be difficult without support from the national government if donor funding depletes or ends. Treating health aid as a replacement for national government spending can weaken a country's health system (Lu et al. 2010). In the literature, there are different attempts to understand the trends in both recipient country's and donor's financing for healthcare projects and their effectiveness using appropriate methods (Biesma et al. 2009, Lu et al. 2010, Roodman 2012, Van de Sijpe 2013, Shaw et al. 2015)

Furthermore, it is critical to identify an efficient and fair mechanism to allocate healthcare funds by donor agencies to recipients. As mentioned in section 1.1, the conventional and widespread approach is a cost-effective analysis that prioritizes projects that deliver value for money (Teerawattananon et al. 2013). Although cost-effective analysis has no intrinsic link to health as an outcome, researchers and practitioners rely on this approach since it is based on the idea that health is a benefit on its own terms (Lauer et al. 2020). There is growing interest in using health technology assessment as a resource allocation approach. It determines whether an intervention's Incremental Cost-Effectiveness Ratio (ICER) indicates that it would constitute an efficient allocation (Chi et al. 2020). ICER is the ratio of the incremental costs to the incremental benefits relative to the current standard of care, wherein benefits are typically measured in a metric such as Quality Adjusted Life Year (QALY) or Disability Adjusted Life Year (DALY) averted. A common prescription is to invest only in those projects that meet some cost-effectiveness threshold. This idea has the merit of being both grounded in economic theory and also practically implementable: it informs decision-making in many countries (Morton et al. 2018). Also, studies such as by Drake et al. (2024) use separate cost-effectiveness thresholds, for donor and recipient country, to reflect the perspective of decision-makers and create a structure for resource allocation.

In the literature, there are optimization modeling approaches for healthcare funding allocation (examples: Weinstein & Zeckhauser (1973), Stinnett & Paltiel (1996), Cleary et al. (2010), Morton et al. (2016), Duro et al. (2024)). Since they have budget constraints and objectives, these models provide a framework for setting goals and discussing outcomes. Such frameworks provide flexibility for the decision-maker to consider issues such as indivisibilities, returns to scale, interactions between alternative investments, and the availability of recourse actions if investment decisions taken under uncertainty do not yield satisfactory results (Morton 2014). As already set out in chapter 1 of the thesis, we have taken an optimization modeling approach. The problem of healthcare funds allocation by donor agencies to recipients has been modeled as a bi-level optimization model that can capture the hierarchical relationship between the two decision-makers in the system.

2.2 Mixed-integer bi-level programming problems

A mixed-integer BLPP is a BLPP with one or more variables that are discrete in either of the two levels of the problem or in both levels. The difficulty of solving the mixed-integer BLPPs depends on which level the discrete variables are present and how they impact the other level of the problem. Vicente et al. (1996), Dempe (2001), and Fanghänel & Dempe (2009) discussed properties and optimality conditions of BLPPs with different structures related to the integer and continuous variables appearing in both levels of the models.

To illustrate the inducible regions of mixed-integer BLPPs, we use an example from Vicente et al. (1996). Consider the mixed-integer BLPP in (2.1) and (2.2):
Upper-level Problem:

$$\underset{y,\ x}{\text{minimize}} \quad f(y,x) \tag{2.1a}$$

$$\text{subject to} \quad x \in \arg\min(\texttt{Lower-level Problem}). \tag{2.1b}$$

Lower-level Problem:

$$\underset{x'}{\text{minimize}} \quad x' \tag{2.2a}$$

$$\text{subject to} \quad y + x' \leq 2 \tag{2.2b}$$

$$-y + x' \leq 2 \tag{2.2c}$$

$$5y - 4x' \leq 10 \tag{2.2d}$$

$$-5y - 4x' \leq 10. \tag{2.2e}$$

There are four different scenarios that can occur as per the integrality of the upper and lower-level variables in the problem.

1. If the variables in both levels are continuous *i.e* $y \in \mathbb{R}$ and $x' \in \mathbb{R}$, these are referred to as Continuous-Continuous BLPPs or linear BLPPs in the literature.

2. If the variables in the upper-level are discrete and those in the lower-level are continuous *i.e* $y \in \mathbb{Z}$ and $x' \in \mathbb{R}$, these are referred to as Discrete-Continuous BLPPs.

3. If the variables in both levels are discrete *i.e* $y \in \mathbb{Z}$ and $x' \in \mathbb{Z}$, these are referred to as Discrete-Discrete BLPPs.

4. If the variables in the upper-level are continuous and those in the lower-level are discrete *i.e* $y \in \mathbb{R}$ and $x' \in \mathbb{Z}$, these are referred to as Continuous-Discrete BLPPs.

The inducible regions of each of these cases are given in Figure 2.1 (Vicente et al. 1996). The discrete variables at either of the levels cause the search space to get disconnected. The optimality conditions for the first three cases is guaranteed easily (as can be seen in Vicente et al. (1996)). However, studying the inducible region of Continuous-Discrete BLPPs and proving the existence of their optimal solutions is non-trivial. The inducible region of Continuous-Discrete BLPPs is a non-compact set and there may not be a bi-level optimal solution existing for the problem although non-empty inducible region.

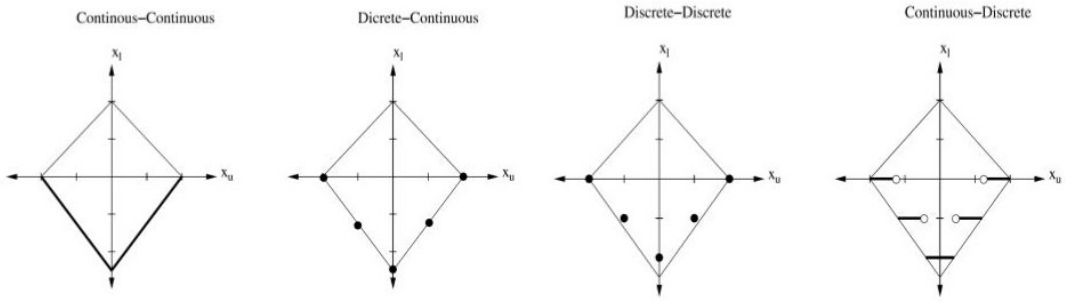


Figure 2.1: Inducible regions of problems wherein the upper and lower-level variables are continuous and/or discrete (Adapted from Vicente et al. (1996))

The Donor-Recipient Bi-level Knapsack Problem (DR-BKP) studied in this thesis is a type of mixed-integer BLPP that has continuous variables in the upper-level and mixed-integer problem in the lower-level. The optimality conditions for this problem in particular and the existence of bi-level optimal solution are studied in chapter 3.

Solution techniques:

A Branch-and-Bound algorithm has been given by Moore & Bard (1990) at the very beginning for general mixed-integer BLPPs. They have given the background of solving these problems using traditional Branch-and-Bound techniques for Mixed-Integer Programming Problems (MILPs). The fathoming rules for MILPs are not directly applicable to mixed-integer BLPPs - if a solution to a relaxed sub-problem is fathomed due to this being worse than the incumbent, a bi-level feasible solution may get disregarded. Hence, the authors have proposed a Branch-and-Bound algorithm with more strict fathoming conditions. They also have proposed some heuristics to trade off accuracy for speed and to obtain good solutions for larger instances. In another paper, the same authors have given an enumeration approach for BLPPs having only binary variables where they find incremental improvements in the upper-level problem (see Bard & Moore (1992)). Another Branch-and-Bound algorithm has been given by Edmunds & Bard (1992) for mixed-integer BLPPs where discrete variables appear only

in the upper-level. A cutting plane approach has been given by Dempe (1996) for Continuous-Discrete BLPPs using Chvátal-Gomory cuts.

A Bender's decomposition-based approach has been given by Saharidis & Ierapetritou (2009) and later on modified for their own problem by Caramia & Mari (2016). In this type of approach, the Discrete-Continuous BLPP is decomposed into a restricted master and a slave problem. The slave problem is converted into bi-level linear problems by fixing its integer values and it is solved using Karush-Kuhn-Tucker (KKT) conditions. The obtained solutions to the slave problems generate cuts for the master problem which is then solved till a bi-level optimal solution is obtained.

A Branch-and-Cut approach for integer BLPPs has been given by DeNegre & Ralphs (2009) by introducing cutting planes derived in a similar way for standard Integer Linear Programming Problems (ILPs). This work has been improved and a generalized Branch-and-Cut Algorithm has been proposed and implemented in an open-source solver by Tahernejad et al. (2020). Xu & Wang (2014) have proposed an improved Branch-and-Bound algorithm for BLPPs with only discrete variables in the upper-level, wherein they propose new pruning rules to eliminate large regions that are not bi-level feasible. They have proposed another exact solution technique, called "watermelon algorithm" (see Wang & Xu (2017)), for solving BLPPs with discrete variables in both levels where they have used multi-way branching to remove bi-level infeasible points from the search space. For BLPPs with only discrete variables in the upper-level, Lozano & Smith (2017) have given an exact finite algorithm using optimal-value-function reformulation. They iteratively generate primal bounds using relaxed BLPPs and dual bounds using bi-level feasible solutions obtained until the bounds are within desired solution gaps.

A finitely-convergent solver has then been given by Fischetti et al. (2017) for general mixed-integer BLPPs, assuming that the upper-level variables that appear in the lower-level must be discrete and bounded. Along with a modified Branch-and-Bound algorithm for the solver, they have proposed new classes of linear inequalities that include intersection cuts based on convex feasible-free sets. They adapt the idea of branching on solutions using an enumeration scheme from Xu & Wang (2014), wherein

they use the branching rule to derive valid inequalities to cut bi-level infeasible solutions from the solution space. This work has been improved upon in their proceeding article (Fischetti, Ljubić, Monaci & Sinnl 2018) in which they have proposed new families of intersection cuts and separation algorithms. An extensive computational study has been done by them on a set of varying classes of problems from the literature and these results have been reported in their article. Liu et al. (2021) have recently proposed an enhanced Branch-and-Bound algorithm for BLPPs with discrete variables that are bounded in both levels. Their algorithm has improved the branching rule over that given in Xu & Wang (2014) and hence can disregard larger bi-level infeasible spaces in each iteration during the search. The following section gives the literature that we have studied related to BKPs.

2.3 Bi-level knapsack problems

As already seen in section 1.2, the BKPs are a class of mixed-integer BLPPs that have knapsack at either or both levels in the model. There are three variants of BKPs as discussed by Caprara et al. (2014) and Carvalho (2016). We discuss these along with some of their extensions recently seen in the literature.

Dempe-Ritcher Variant:
First is the Dempe-Ritcher variant by Dempe & Richter (2000) where the knapsack budget is decided by the leader and items in this knapsack are selected by the follower. This model has continuous variables in the upper-level and binary variables in the lower-level. The objective of both leader and follower is to maximize their respective profits. Dempe & Richter (2000) have given a pseudo-polynomial exact algorithm and polynomial time approximate algorithm. A Dynamic Programming (DP) algorithm has been given by Brotcorne et al. (2009) for BKPs with upper-level controlling the continuous capacity of the lower-level knapsack and the follower solves a binary knapsack problem with the chosen capacity.

Mansi-Alves-de-Carvalho-Hanafi:

Second variant is called the Mansi-Alves-de-Carvalho-Hanafi variant given by Mansi et al. (2012) in which the knapsack is shared by both leader and follower with a pre-decided budget. A reformulation approach has been given by Brotcorne et al. (2013) for integer BKPs, which has then been solved using a two-step algorithm. The authors used a DP approach to find all possible reactions of the follower in the first step and all the obtained reactions have been used to reformulate the BKP as a single-level MILP in the second step. This reformulation has been solved using an MILP solver.

DeNegre variant:

DeNegre (2011) has given the third variant of BKPs. In this variant, both leader and follower have their independent knapsacks and they select items from a common set of items. This variant is a type of interdiction models. Interdiction models are leader–follower games in which the leader takes interdiction actions to maximize the minimum objective a follower can obtain in solving its optimization problem (McMasters & Mustin 1970, Wood 1993). The leader's interdiction actions can impact the follower's objective, feasible region, or both (Smith & Song 2020). The author has developed a Branch-and-Cut framework to solve a pure integer framework and a reformulation approach to solve this variant. Another solver has been given by Caprara et al. (2016), where the authors use continuous relaxation of the follower's problem to get a single-level reformulation and then compute the upper bounds iteratively till a stopping condition is satisfied. Della Croce & Scatamacchia (2020) first compute effective bounds for this variant of the BKPs. These bounds are then used to explore promising sub-problems through constraint generation and pruning. The authors have extended this solution approach to the Min-max Regret Knapsack Problem (`MRKP`), which shows improved performance over a Lagrangian-based Branch-and-Cut approach proposed by Furini et al. (2015). An exact Branch-and-Cut algorithm has been recently given by Fischetti et al. (2019) for interdiction games that have follower problem satisfying certain monotonicity property. One of the examples of the problems that have this property are DeNegre's variant and the authors have conducted a computational study

on the benchmark instances of the variant.

In the literature, we can find approximation-guaranteed algorithms for some variants of the BKPs. In the problem setup by Briest et al. (2012), the follower has to select a set of items within a given weight and at minimum cost. Since the follower is computationally bounded, it uses a greedy 2-approximation algorithm. The authors give a $(2+\epsilon)$-approximation algorithm to maximize the leader's revenue in this setup. Other pseudo-polynomial algorithms are given by Chen & Zhang (2013) and then improved upon by Qiu & Kern (2015) for different versions of a BKP variant in which both leader and follower pack their items simultaneously in their own knapsacks. The follower maximizes its own profit however leader is concerned to maximize both profits.

Two variants of BKPs with continuous variables in the upper-level and binary variables in the lower-level are given by Pferschy et al. (2019) and Pferschy et al. (2021). Greedy heuristics and pseudo-polynomial time exact algorithms have been provided for these problems. In these variants, the items of knapsack are partitioned as leader and follower items. The follower decides which of these items get picked in the knapsack that are within some budget. There is a maximum profit level that can be attained for a leader's item. The leader decides on the profit levels that it will receive while awarding the remaining profits to the follower thereby incentivizing the follower to pick the leader's items in the knapsack. Incentives can also be offered as weight offsets in the knapsack and these are deducted from the leader's profit (Pferschy et al. 2019). For instance, in the application provided in Pferschy et al. (2019), there is a trader that provides cost offsets to the products offered to his/her customers. This is modeled as the trader having a reduced return due to the cost offset he provides. Typically traders can borrow and invest the loan for the offset. One would typically then maximize the return on the profits provided by the products after the interest from the loan has been deducted.

2.4 Heuristic-based solution techniques

A state-of-the-art collection of meta-heuristics, hybrid meta-heuristics, and exact methods has been given by Talbi (2013) for solving different types of BLPPs that have applications in a wide range of applications. Also, a comprehensive review has been given by Sinha et al. (2018) wherein the authors give classical and evolutionary algorithms for BLPPs. The complex structure of the BLPPs has led to the increasing popularity of using heuristic-based algorithms to handle these challenging problems (Deb et al. 2020). The developed heuristics are either surrogate-based or non-surrogate-based models. If an approximation of the actual model is used to evaluate it fast, it is called a surrogate-based model. The non-surrogate-based models as seen in the relevant literature can be classified into the following types:

- Nested sequential approach
- Single-level reduction approach
- Co-evolutionary approach

2.4.1 Nested sequential approach

As the name implies, these types of heuristic approaches first deal with the leader's problems. For each of the leader solutions obtained, the follower's problem is solved. The leader first generates solutions and then the follower reacts to the leader's decisions. Every iteration involves evaluation by considering the upper and lower-level decision variables. These type of approaches can be computationally expensive though, since the follower's problem is solved for each of the leader's decision. The first time an evolutionary approach was used by Mathieu et al. (1994) for solving BLPPs had a nested structure. They used a genetic algorithm for solving the upper-level problem and a linear program for the lower-level problem. These approaches are generally used in two ways, *viz.* (1) an evolutionary algorithm for the upper-level and an exact solver for the lower-level problem, and (2) evolutionary algorithms for the upper and lower-level problems.

Yin (2000) has given a nested approach for solving a real-world application in transportation system planning problem, and shown how this can be effectively adapted to handle non-convex BLPPs. In the application, the upper-level problem is solved using a genetic algorithm where its decision variables are used as individuals and the fitness of these individuals is measured using the lower-level solutions. Yin (2000) has used the Franke-Wolfe algorithm to solve the lower-level problem.. A nested approach using Particle Swarm Optimization (PSO) has been given by Li et al. (2006) wherein the authors use two variants of PSOs at the upper and the lower-level of the BLPPs. Calvete et al. (2011) have given a nested approach for a production-distribution planning problem formulated as a BLPP. The upper-level is solved using an ant colony algorithm and lower-level is solved to optimality. Another nested approach for a production-distribution planning problem formulated as a BLPP has been given by Camacho-Vallejo, Muñoz-Sánchez & González-Velarde (2015). The authors have used a Scatter Search algorithm in the upper-level problem and an off-the-shelf exact solver for the lower-level problem.

Another nested approach has been given by Arroyo & Fernández (2013) for power-system vulnerability assessment through an attacker-defender model which is a mixed-integer BLPP. The upper-level problem is solved using a genetic algorithm and the lower-level problem (an MILP) is solved using an off-the-shelf exact solver. Angelo & Barbosa (2015) have given a nested approach where they use ant colony optimization for upper-level problem and differential evolution method for the lower-level problem. They have formulated a mixed-integer BLPP to solve a production-distribution problem. Camacho-Vallejo, Mar-Ortiz, López-Ramos & Rodríguez (2015) have proposed a nested approach for local network design problem wherein they solve the lower-level problem with a heuristic approach and the upper-level problem with a genetic algorithm. Cheraghalipour et al. (2019) have modeled a realistic rice supply chain problem as a mixed-integer BLPP and solved it using two nested meta-heuristics and also a few hybrid approaches.

2.4.2 Single-level reduction approach

BLPPs are transformed into equivalent single-level reformulations in these types of approaches followed by using heuristic methods to solve the reformulation. Usually, these reformulations are non-linear due to some of the constraints. Heuristic methods can be developed to avoid struggling with the complicated constraints and the resulting formulations. Similar to the exact solution techniques, the KKT conditions of the lower-level problem that follow certain regularity conditions can be used to transform the BLPP into a single-level problem. Using heuristic methods in these approaches allows these regularity conditions to be more general.

Hejazi et al. (2002) have transferred their BLPP into a single-level reformulation and then solved using a genetic algorithm. A single-level reformulation for a BLPP has been given by Wang et al. (2005), using the KKT conditions. The authors have used an evolutionary algorithm, that is based on a constraint handling scheme and a crossover operator, to effectively solve the reformulation. It is a useful approach for also the BLPPs that have nondifferentiability in upper-level objective function. A new improved evolutionary algorithm has been given by Wang et al. (2011) that performed better than their previous algorithm and it is also useful for BLPPs that have nondifferentiability in the upper-level objective function and non-convexity in the lower-level problem. A simplex-based genetic algorithm has been given by Wang et al. (2008) for linear-quadratic BLPPs. They transform the BLPP into a single-level problem using KKT conditions of the lower-level problem. Another single-level transformation of the BLPPs is given by Wan et al. (2013) where the authors use KKT conditions of the lower-level problem for this. Then they solve the reformulation using the PSO and chaos search technique together. Fischetti, Monaci & Sinnl (2018) have developed heuristic schemes based on single-level reformulation for solving a type of BLPPs called the interdiction problems.

2.4.3 Co-evolutionary approach

In co-evolutionary approaches, there are two levels maintained in the meta-heuristic. Each level has a population of partial solutions, one for the leader and another for

the follower. These levels evolve in parallel to improve their objectives. There is a co-evolutionary mechanism that exchanges information between these populations, for example, sending the leader's elite solutions to the follower's population. This exchange of information is to obtain the complete set of solutions and also to keep track of the overall objective of the BLPP. Co-evolutionary approaches are generally developed and used where other approaches like nested and reformulation are not practically usable. These approaches however are less common in the literature.

Oduguwa & Roy (2002) have given a bi-level genetic algorithm (`BiGA`) for different classes of BLPPs that are within a single framework. The algorithm maintains two different populations in parallel and an external elite population is used to identify elite individuals in the two populations after every generation. The co-evolution is achieved by sending lower-level variables to the upper-level population using a crossover operator. Another co-evolutionary algorithm (`CoBRA`) is given by Legillon et al. (2012) for a more general class of BLPPs. The cooperation between the two players is symmetric *i.e.* both players cooperate with each other as against the asymmetric cooperation given by Oduguwa & Roy (2002). A co-evolutionary decomposition-based algorithm to solve discrete BLPPs has been given by Chaabani et al. (2017). They have further extended this algorithm to solve BKPs in Chaabani & Said (2020).

2.5 Other solution approaches

In the relevant literature, other approaches can be seen that develop solution algorithms for BLPPs. We have given some of these approaches in this section.

Multi-objective optimization approach:
As the name suggests, in this type of approach, the BLPPs are transformed into a multi-objective optimization framework such that solution methods of the multi-objective optimization problems can be used to solve relevant BLPPs. There are several attempts in the literature to find a link between BLPPs having single-objective and multi-objective optimization problems. Some of the earlier

attempts are by Bard (1984) and Ünlü (1987), however, the optimal solution to the original problem cannot be seen in the Pareto-front of the bi-objective equivalent problem. Fülöp (1993) investigated the link between BLPPs having single-objective and multi-objective optimization problems using an efficient set. The author shows that a linear multi-objective programming problem can be constructed such that the feasible solutions of a linear BLPP can represented with the efficient solutions of the multi-objective programming problem, followed by exploring the converse direction of the reformulation. Ecker & Song (1994), Glackin et al. (2009) have used these findings for developing solution algorithms of linear BLPPs. Also, Ruuska et al. (2012) extended work by Fülöp (1993) to general vector-valued functions, thus allowing the lower-level problem in a BLPP to be a nonlinear multi-objective optimization problem.

Fliege & Vicente (2006) have proposed a binary relation based on cone dominance such that a bi-level optimal solution of the BLPP is optimal if and only if it is efficient with respect to the proposed relation. They have used this relation between the bi-level optimal solutions and the non-dominated solutions of the related multi-objective problem for reformulation of a BLPP as a multi-objective optimization problem. An extension of this binary relation and problem reformulation has been given by Ivanenko & Plyasunov (2008), wherein the models can have upper-level constraint functions depending on the lower-level decision. For a realistic application of this approach, one can refer to the exact integer-linear multi-objective optimization methodology given by Andrade et al. (2020). The authors solve a metabolic engineering problem previously solved using bi-level optimization framework with the aim to expand the current set of tools for the problem.

Surrogate-based methods:

The non-surrogate-based models require high computational time typically. In order to deal with this, there has been research done on surrogate-based methods

to solve the BLPPs. These methods are commonly used for problems that have functions which are difficult and expensive to evaluate. A surrogate model is an approximation of the actual model of the problem such that the approximated model is faster to evaluate compared to the original model. In simpler problems, the surrogate model can be trained and used further using a small sample of the actual model. However, the BLPPs are intrinsically complex set of problems and hence there are iterative meta-modeling techniques used to approximate the actual model. There is a surrogate model of the lower-level problem in the BLPPs combined with evolutionary algorithms for the upper-level problem to search for good solutions for the overall BLPP. The meta-modeling techniques are based on two different mappings in the BLPPs (Sinha et al. 2018), *viz.*:

1. Based on the rational reaction set wherein, the evolutionary algorithms use iterative approximation of the reaction set mapping to solve the BLPPs. Some of the literature using this approach are Angelo et al. (2014), Sinha et al. (2014) and Sinha et al. (2017).

2. Based on the lower-level optimal value function. The authors have combined evolutionary algorithms with an iterative approximation of the optimal value function to solve the BLPPs in these type of approaches (Sinha et al. 2016, 2020).

Although these approaches seem to be effective, they have been found difficult to use and ineffective for large-scale BLPPs (Kieffer et al. 2020). Instead of such approaches, Kieffer et al. (2020) have proposed a Genetic Programming Hyper Heuristics approach to address large scale BLPPs. They have used a machine learning model to train their heuristics to find heuristics that best solve unseen lower-level instances.

2.6 Concluding remarks

In this chapter, we have given a review of the literature on solution techniques for BLPPs for both exact and heuristic methods. We have started the discussion with exact solution methods for the general BLPPs, then for the mixed-integer BLPPs. Later a detailed literature review has been given for the BKPs since these are the class of problems closely related to our problem formulation. One can perceive the BKP that we have proposed in this work as an extension or variation of the Dempe-Ritcher variant (Dempe & Richter 2000). In this variant, a subsidy is directly provided to expand the budget of the recipient. In our model, the leader has greater control over how the subsidy is allocated by providing offsets to the costs of the individual projects that are of interest to the leader. The leader makes a single decision on the continuous budget of the follower. The DR-BKP, as introduced in section 1.3, presents a challenge that cannot be addressed using any of the exact or heuristic-based solution methods currently available in the literature. In chapter 3, we present the DR-BKP and related properties, followed by exact solution methods that we have developed along with the computational experiments that test their performance. We have also developed a genetic algorithm to address some data sets of the DR-BKP and give this along with the results in chapter 4.

Chapter 3

Exact solution techniques for the DR-BKP

Bi-level Programming Problems (BLPPs) are an interesting and complex class of problems that have been drawing researchers' attention more recently. They are widely applicable since they rightly capture the hierarchical relationship between the participants that are involved in actual decision-making. A BLPP can model the impact of decisions made by one participant on the decisions made by another participant. A special class of mixed-integer BLPPs, called Bi-level Knapsack Problem (BKP), has a knapsack at either or both levels in their formulation. The healthcare funds allocation problem given by Morton et al. (2018) is a BKP and we refer to it as Donor-Recipient Bi-level Knapsack Problem (DR-BKP).

Contributions:
The main contribution of this chapter is two finitely converging exact algorithms for the DR-BKP. These algorithms are based on the ideas of two finitely converging exact algorithms, (a) an enumeration algorithm using optimal-value-function reformulation by Lozano & Smith (2017) and (b) a branching technique to eliminate large regions that are not bi-level feasible by Xu & Wang (2014) to solve the DR-BKP model. There are other optimal-value-function-reformulation based ap-

proaches, like DeNegre & Ralphs (2009), Fischetti et al. (2017), Fischetti, Ljubić, Monaci & Sinnl (2018), Tahernejad et al. (2020). In general, they solve the linear relaxation of the BLPP, iteratively generate valid inequalities to improve the bounds, and branch when necessary. The lower-level optimal values are used to generate cuts that reduce solution space in their procedures. Whereas, Lozano & Smith (2017) rely on partial enumeration of lower-level solutions. They generate cuts by fixing lower-level variables to accelerate the convergence of their algorithm.

We differ from the models given by Lozano & Smith (2017) and Xu & Wang (2014) in many ways. Firstly, our problem has continuous variables in its upper-level and both continuous and discrete variables in the lower-level. Most mixed-integer BLPPs assume that the lower-level problem is parameterized exclusively by the upper-level integer variables. Continuous upper-level variables in the lower-level problem impose two difficulties. The first one being the ill-posedness of the problem due to non-compact feasible region *i.e.* an optimal solution may not exist in such cases as shown by Vicente et al. (1996). An example is given by Köppe et al. (2010) to illustrate this case. Our upper-level objective is a discrete function, which circumvents this issue. The second difficulty lies in the design of the algorithm. The constraints added in both Lozano & Smith (2017) and Xu & Wang (2014) require that the lower-level problem is parameterized by integer upper-level variables to avoid open feasible sets. We have shown that convergence is guaranteed even without this assumption. In addition, the upper and lower-level variables interact non-linearly at the lower-level constraints but the parameterized lower-level problem is a mixed-integer linear program. This requires us to modify both constraint and branching rules of Lozano & Smith (2017) and Xu & Wang (2014).

We have made an assumption on the cost of projects that we will discuss later in the sequel. With this assumption, we have shown convergence and the computational experiments are performed and presented. We have also provided

evidence for Σ_2^P-complexity in Section 3.2.1, by showing the decision version of the problem is both NP-hard and Co-NP hard.

The problem has been formally defined in Section 3.1 followed by the complexity of the DR-BKP. The enumeration algorithm and the branching technique have been given in Sections 3.3 and 3.4 respectively, to solve the DR-BKP. A set of 150 instances (10 in each of the 15 different data sets) have been solved, compared and presented in Section 3.5. These data sets have been generated to mimic the different scenarios arising in real-life healthcare problems. Finally, the last section concludes this chapter.

3.1 Notation and definitions

An instance of a knapsack problem comprises a budget and a set of items, each with a profit and cost. The objective is to pick a subset of items (either fractionally or wholly) such that their costs are within the budget and the profit is maximized. An instance of the DR-BKP comprises two players, a donor and a recipient. We have a set I of n projects, $I = \{1, \ldots, n\}$, that are common to both players. Each project $i \in I$, has a profit of $w_i \in \mathbb{N}$ (resp. $v_i \in \mathbb{N}$) for the donor (resp. recipient), and a cost $c_i \in \mathbb{N}$. Let $\mathbf{w}$ and $\mathbf{v}$ denote the vectors of profits of the donor and the recipient respectively and $\mathbf{c}$ be the vector of costs of these projects. We have two integer budgets, B_d and B_r, corresponding to the donor and the recipient. Besides the projects in I, the recipient has to allocate its budget to an outside option of projects. This represents a portfolio of projects that is of no interest to the leader. We refer to this option as an "external project". We consider the model introduced in Morton et al. (2018), where the external project has a linear profit and linear cost of v_0 and c_0 respectively. So an instance of the DR-BKP is specified by the input $(\mathbf{w}, \mathbf{v}, \mathbf{c}, v_0, c_0, B_d, B_r)$. The recipient solves a knapsack problem, where each item of the knapsack corresponds to a project $i \in I$ with a profit v_i and cost $c_i - c_i y_i$, where y_i is the proportion of the

cost of project i that is subsidized by the donor. These projects are binary and cannot be fractionally picked. Along with the healthcare projects, the recipient has to fund the external project which can be done fractionally or wholly *i.e.* only a proportion or entire of the cost of the external project can be funded by the recipient.

A solution to an instance of the DR-BKP is to decide on the proportion of cost to be subsidized, y_i, for each project $i \in I$ with $\sum_{i\in I} c_i y_i \leq B_d$ such that profit of the donor is maximized given the projects are in the optimal solution set of the recipient's cost subsidized knapsack problem. We use the notation $\mathbf{y}$ to denote a vector of subsidy. The leader cannot subsidize a project more than its cost and the total subsidy cannot exceed the leader's budget. The set of all valid subsidies is denoted by $Y := \left\{\mathbf{y} : \sum_{i\in I} c_i y_i \leq B_d, \mathbf{y} \in [0,1]^n\right\}$.

We let $\mathbf{x}$ to denote a 0-1 vector representing the set of projects that are picked (i^{th} component of the vector, x_i, is 1 if project i is picked and 0 otherwise) and x_0 to denote the proportion of cost of the external project that is being funded by the recipient. We define the set $X := \left\{(\mathbf{x}, x_0) : \mathbf{x} \in \{0,1\}^n, x_0 \in [0,1]\right\}$. Let $\mathcal{X} := \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^K\}$ be the set of all possible subsets of projects. We define the set of all valid projects corresponding to a subsidy $\mathbf{y} \in Y$ as

$$\mathcal{G}(\mathbf{y}) := \{\mathbf{x} \in \mathcal{X} : \sum_{i\in I}(c_i - c_i y_i)x_i \leq B_r\}. \tag{3.1}$$

The DR-BKP proposed by Morton et al. (2018) has been given in (3.2) and (3.3) where upper-level is the donor problem `(DONOR)` and lower-level is the recipient problem `(RECIPIENT(y))` parameterised on the upper-level decision $\mathbf{y}$.
Problem `DONOR`:

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{3.2a}$$

$$\text{subject to} \quad \mathbf{y} \in Y \tag{3.2b}$$

$$\mathbf{x} \in \arg\max(\texttt{RECIPIENT(y)}). \tag{3.2c}$$

Problem RECIPIENT(y):

$$\text{maximize} \quad \mathbf{v}^T\mathbf{x} + v_0x_0 \tag{3.3a}$$

$$\text{subject to} \quad \sum_{i\in I}(c_i - c_iy_i)x_i + c_0x_0 \leq B_r \tag{3.3b}$$

$$(\mathbf{x}, x_0) \in X. \tag{3.3c}$$

We now introduce a few more notations.
Relaxed feasible set:

$$S = \Big\{(\mathbf{y}, (\mathbf{x}, x_0)) : \sum_{i\in I}(c_i - c_iy_i)x_i + c_0x_0 \leq B_r, \mathbf{y} \in Y, (\mathbf{x}, x_0) \in X\Big\}. \tag{3.4}$$

Follower's rational reaction set for a fixed $\hat{\mathbf{y}} \in Y$:

$$P(\hat{\mathbf{y}}) = \Big\{(\mathbf{x}, x_0) : (\mathbf{x}, x_0) \in \arg\max\Big\{\mathbf{v}^T\mathbf{x} + v_0x_0 : \sum_{i\in I} c_ix_i + c_0x_0 \leq B_r + \sum_{i\in I} c_i\hat{y}_ix_i, (\mathbf{x}, x_0) \in X\Big\}\Big\}. \tag{3.5}$$

Inducible Region:

$$IR = \{(\mathbf{y}, (\mathbf{x}, x_0)) \in S : (\mathbf{x}, x_0) \in P(\mathbf{y})\}. \tag{3.6}$$

With these notations, DR-BKP can also be defined as:

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{3.7a}$$

$$\text{subject to} \quad (\mathbf{y}, (\mathbf{x}, x_0)) \in IR. \tag{3.7b}$$

3.2 Properties and assumptions

Let Relaxed Donor-Recipient Bi-level Knapsack Problem (R-DR-BKP), given as (3.8), denote the Relaxed DR-BKP *i.e.* the DR-BKP after ignoring the objective

function of the RECIPIENT problem. This relaxation of a bi-level optimization problem is generally called as High Point Relaxation (HPR) in the literature.

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{3.8a}$$

$$\text{subject to} \quad (\mathbf{y}, (\mathbf{x}, x_0)) \in S. \tag{3.8b}$$

The integer constraints in Mixed-Integer Programming Problems (MILPs) are relaxed and standard rules are applied for pruning off low quality solutions in Branch-and-Bound solution techniques. However, this methodology cannot be adopted for mixed-integer BLPPs. Let us call the MILP HPR as $\overline{\texttt{HPR}}$ after relaxing its integer constraints. The inducible region of $\overline{\texttt{HPR}}$ may not contain the inducible region of original problem (Moore & Bard 1990). Also unlike in the case of standard MILPs, unboundedness of $\overline{\texttt{HPR}}$ relaxation cannot be used to derive the optimal solution of original bi-level problem. An unbounded $\overline{\texttt{HPR}}$ region can imply either infeasible, unbounded or occurrence of an optimal solution (Xu & Wang 2014). This situation however is not an issue in our problem R-DR-BKP since the integer variables have finite bounds. The finite bounds on integer variables also assure that there would never be a situation when RECIPIENT problem is infeasible or unbounded for any donor decision $\mathbf{y}$.

In general, mixed-integer BLPPs with continuous upper-level variables that appear in the constraints at the lower-level problem may have a non-compact feasible region resulting in no optimal solution even if the feasible region is non-empty (Vicente et al. 1996, Köppe et al. 2010). An example in these works conveys the idea that an optimal solution may never be attained. However in our problem DR-BKP, the upper-level objective is a discrete function taking discrete variables with finite bounds as input, and hence a maximum always exists. This is given in proposition 1.

Proposition 1. *DR-BKP has a maximum.*

Proof. For a fixed set of projects, $\mathbf{x}^k \in \mathcal{X}$, let

$$R(\mathbf{x}^k) := \Big\{\mathbf{y} \in Y : \max_{(\mathbf{x},x_0)\in X} \Big\{\mathbf{v}^T\mathbf{x} + v_0x_0 : \sum_{i\in I}(c_i - c_iy_i)x_i \le B_r\Big\} \\ \le \mathbf{v}^T\mathbf{x}^k + v_0x_0(\mathbf{x}^k,\mathbf{y}), \\ \sum_{i\in I}(c_i - c_iy_i)x_i^k \le B_r\Big\} \tag{3.9}$$

where

$$x_0(\mathbf{x}^k,\mathbf{y}) = \min\Big\{1, \frac{B_r - \sum_{i\in I} c_ix_i^k + \sum_{i\in I} c_iy_ix_i^k}{c_0}\Big\}.$$

For each such solution $\mathbf{x}^k \in \mathcal{X}, k = 1,\dots,K$, with the corresponding upper objective $\mathbf{w}^T\mathbf{x}^k$, we are interested in knowing whether $R(\mathbf{x}^k) = \emptyset$ or not and the non-compactness of $R(\mathbf{x}^k)$ is not relevant. Due to the finiteness of $\mathcal{X}$, we could simply order the solutions in $\mathcal{X}$ in the decreasing order of their corresponding upper-level objective values. More formally, let π be the ordering, such that $\mathbf{w}^T\mathbf{x}^{\pi_1} \ge \cdots \ge \mathbf{w}^T\mathbf{x}^{\pi_K}$. Pick the first solution in this order for which $R(\mathbf{x}^{\pi_k}) \neq \emptyset$. So $\exists\hat{\mathbf{y}} \in Y$ such that $\mathbf{x}^{\pi_k} \in P(\hat{\mathbf{y}})$ and for all ℓ, such that $\mathbf{w}^T\mathbf{x}^{\pi_\ell} \ge \mathbf{w}^T\mathbf{x}^{\pi_k}$ we have $\mathbf{x}^{\pi_\ell} \notin P(\mathbf{y})$ for all $\mathbf{y} \in Y$. □

We now make the following assumption. The cost of external project in RECIPIENT problem is at least equal to the budget of the RECIPIENT, i.e.,

$$c_0 \ge B_r. \tag{3.10}$$

The reason for this assumption is given in section 3.3. Under this assumption we have

$$x_0(\mathbf{x}^k,\mathbf{y}) = \frac{B_r - \sum_{i\in I} c_ix_i^k + \sum_{i\in I} c_iy_ix_i^k}{c_0}.$$

From an application perspective, this assumption is not restrictive as the external option summarises the cost of all other projects that a recipient country incurs and this typically exceeds the recipient's budget.

3.2.1 Complexity of the DR-BKP

Regardless of the cost assumption (3.10), the results of this section hold. We now provide evidence for DR-BKP to be Σ_2^p-hard. We do this by showing that it is both NP-hard and Co-NP hard. However, there are no immediate certificates to show that they are in either NP or Co-NP. So, unless NP = Co-NP, it is likely to be complete in a higher complexity class in the polynomial hierarchy. The class of problems in P are solvable in deterministic polynomial time, whereas the class of problems in NP have a polynomial time verifier, given a polynomial-size certificate. These problems are difficult to solve however, if a solution is provided, it takes polynomial time to check if the given solution is correct. In case it takes polynomial time to check if the given solution is incorrect, these problems are in Co-NP (*i.e. Complementary to class NP*).The class of problems in NP-complete are a subset of problems that are considered the hardest in the NP class. An NP-complete problem can be verified efficiently however no known efficient algorithms exist to solve them. A problem is in class NP-hard if every problem in NP-complete can be reduced to this problem in polynomial time. If any NP-hard problem could be solved efficiently, then all NP-complete problems can be solved efficiently. The class of problems that are complementary to NP-hard class are called Co-NP-hard.

We now define the decision version of the DR-BKP to show our hardness results.

Definition 1. *The input to the decision problem D-DR-BKP is an instance of DR-BKP* $(\mathbf{w}, \mathbf{v}, \mathbf{c}, v_0, c_0, B_d, B_r)$ *and a number* k *and it answers*

- ***YES****, if there is a subsidy* $\hat{\mathbf{y}} \in Y$ *and a project set,* $\hat{\mathbf{x}} \in \mathcal{G}(\mathbf{y})$*, such that for all* $(\mathbf{x}, x_0) \in P(\hat{\mathbf{y}})$*, we have* $\mathbf{v}^T\hat{\mathbf{x}} + v_0 x_0(\hat{\mathbf{y}}, \hat{\mathbf{x}}) \geq \mathbf{v}^T\mathbf{x} + v_0 x_0$ *and* $\mathbf{w}^T\hat{\mathbf{x}} \geq k$*, and*
- ***NO****, otherwise.*

Theorem 1. *D-DR-BKP is NP-hard.*

Proof. We show this by reducing an instance of knapsack problem to D-DR-BKP. In the knapsack instance, we are given a set of n items with profits $\{p_1, \ldots, p_n\}$, weights $\{w_1, \ldots, w_n\}$ and a budget B. The decision version of the problem asks whether there exists a set of items $S \subset \{1, \ldots, n\}$ with $\sum_{i \in S} p_i \geq k$ and $\sum_{i \in S} w_i \leq B$. We create an instance of D-DR-BKP by creating one project for each knapsack item and there is no external project, i.e., we have $c_0 = v_0 = 0$. The cost of a project is the corresponding knapsack item's weight. Both donor's and recipient's profits will be the corresponding knapsack item's profit. The recipient's budget B_r is 0 and donor's budget B_d is the knapsack budget B. The D-DR-BKP instance has an optimal value of k if and only if the knapsack instance has a solution value of at least k. We observe that an item can never be picked unless it is completely subsidized by the leader. Otherwise, it is infeasible to the follower. If the knapsack instance is yes, then the leader could simply subsidize the items in this set fully. Otherwise, no subset of items that can fully be subsidized (within the budget B) will have a profit of at least k. □

Theorem 2. *D-DR-BKP is Co-NP-hard.*

Proof. We show this by reducing the inverse subset sum problem (ISSP). An instance of this problem comprises of a set, A, of n integers $a_1, \ldots, a_n$ and a target integer B. We answer NO to this instance if there exists a subset, $S \subset A$, of integers that add up to exactly B and YES otherwise. For the reduction, we take projects in I corresponding to the n integers in A. We will refer to these projects as integer projects. We also take one extra project in I. There are no external projects, i.e., we have $c_0 = v_0 = 0$. The costs and the recipient profits of the integer projects are the same as the corresponding integers. The donor's profits for integer projects are all 0. The extra project has a cost of 1 with a donor profit of 1 and recipient profit of 1/2. The recipient's budget is B and the donor's budget is 0. Now the ISSP has a solution if and only if the constructed D-DR-BKP has a solution value of at least 1. To see this, first note that donor does not have any budget and cannot subsidize any project and it is entirely up

to the recipient to pick projects. If there exists a set of integers for ISSP that adds up to B, then the recipient will pick the corresponding integer projects and get a profit of B and the extra project will not be picked. If there are no subset of items that add up to B, then recipient will definitely pick the extra project to maximize its profits which results in a donor profit of 1. □

Unfortunately, we do not have a direct reduction from a Σ_2^p-complete problem and we leave this as a conjecture.

3.3 Enumeration algorithm

Consider the model (3.8) after linearising the products of donor subsidies y_i and project selections x_i in the lower-level budget constraint.

Problem R-DR-BKP:

$$\begin{aligned}
\text{maximize} \quad & \mathbf{w}^T\mathbf{x} && && \text{(3.11a)}\\
\text{subject to} \quad & \mathbf{c}^T\mathbf{y} \leq B_d && && \text{(3.11b)}\\
& \mathbf{c}^T\mathbf{x} + c_0 x_0 \leq B_r + \mathbf{c}^T\mathbf{y} && && \text{(3.11c)}\\
& y_i \leq x_i && \forall i \in I && \text{(3.11d)}\\
& \mathbf{y} \in [0,1]^n && && \text{(3.11e)}\\
& \mathbf{x} \in \{0,1\}^n && && \text{(3.11f)}\\
& x_0 \in [0,1]. && && \text{(3.11g)}
\end{aligned}$$

Constraint (3.11d) assures that there are no subsidies given in case the project is not picked. This is not restrictive. For a fixed set of projects, $\hat{\mathbf{x}}$, if $R(\hat{\mathbf{x}})$ is non-empty then there exists a subsidy $\hat{\mathbf{y}} \in R(\hat{\mathbf{x}})$ such that $\hat{y}_i \leq \hat{x}_i$ for all $i \in I$. We can reduce the subsidy of a project for an arbitrary subsidy vector, which was not picked by the recipient's optimal solution, to 0. This will not change the optimal solution of the recipient's problem. This also allows us to avoid the

bi-linear terms in (1.3b) and rewrite that constraint as (3.11c).

For an optimal solution $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ of R-DR-BKP to be bi-level feasible to DR-BKP, we need $(\mathbf{x}^*, x_0^*) \in P(\mathbf{y}^*)$. Since it is a relaxation, we also achieve optimality. We will formalise this soon. We are now interested to know how to tighten this relaxation if $(\mathbf{x}^*, x_0^*) \notin P(\mathbf{y}^*)$. In other words, we want to eliminate this point from the search space. In this case, for any $(\bar{\mathbf{x}}, \bar{x}_0) \in P(\mathbf{y}^*)$ the inequality

$$\mathbf{v}^T\mathbf{x} + v_0 x_0 \geq \mathbf{v}^T\bar{\mathbf{x}} + v_0\bar{x}_0 \tag{3.12}$$

will eliminate $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ from the search space. Since $(\bar{\mathbf{x}}, \bar{x}_0)$ is optimal to `RECIPIENT(`$\mathbf{y}^*$`)`,

$$\bar{x}_0 = x_0(\bar{\mathbf{x}}, \mathbf{y}^*). \tag{3.13}$$

Using this, inequality (3.12) can be written as

$$\mathbf{v}^T\mathbf{x} + v_0 x_0 \geq \mathbf{v}^T\bar{\mathbf{x}} + v_0 x_0(\bar{\mathbf{x}}, \mathbf{y}). \tag{3.14}$$

Under assumption(3.10), we have

$$x_0(\bar{\mathbf{x}}, \mathbf{y}) = \frac{\left(B_r - \sum_{i\in I} c_i(\bar{x}_i - \bar{x}_i y_i)\right)}{c_0}.$$

This gives us linear components in the RHS of (3.14) and we do not have to introduce binary variables. We define $c'_i := \frac{v_0 c_i}{c_0}$. Inequality (3.14) can then be re-written as

$$\mathbf{v}^T\mathbf{x} + v_0 x_0 - \sum_{i\in I} c'_i \bar{x}_i y_i \geq \mathbf{v}^T\bar{\mathbf{x}} + \frac{v_0}{c_0}(B_r - \mathbf{c}^T\bar{\mathbf{x}}). \tag{3.15}$$

However, inequality (3.15) can only be added if $(\mathbf{y}, (\bar{\mathbf{x}}, \bar{x}_0)) \in S$. Otherwise, we will cutoff valid subsidies from our search space. This is added as con-

straints (3.16d) and (3.16e) in EBKP given below. Big M - M_1 and M_2 - are used here to handle the "if-then" nature of the constraint. The choice of big M is discussed further in Section 3.5.

$$\begin{aligned}
\text{maximize} \quad & \mathbf{w}^T\mathbf{x} && \text{(EBKP)}\\
\text{subject to} \quad & \mathbf{c}^T\mathbf{y} \leq B_d && \text{(3.16a)}\\
& \mathbf{c}^T\mathbf{x} + c_0x_0 \leq B_r + \mathbf{c}^T\mathbf{y} && \text{(3.16b)}\\
& y_i \leq x_i \quad \forall i \in I && \text{(3.16c)}\\
& \mathbf{c}^T\mathbf{x}^k - \sum_{i\in I} c_i x_i^k y_i + M_1 t^k \geq B_r + \epsilon \quad \forall k \in \{1,\ldots,K\} && \text{(3.16d)}\\
& \mathbf{v}^T\mathbf{x} + v_0x_0 - \sum_{i\in I} c_i' x_i^k y_i + M_2(1-t^k) \\
& \qquad \geq \mathbf{v}^T\mathbf{x}^k + \frac{v_0}{c_0}(B_r - \mathbf{c}^T\mathbf{x}^k) \quad \forall k \in \{1,\ldots,K\} && \text{(3.16e)}\\
& t^k \in \{0,1\} \quad \forall k \in \{1,\ldots,K\} && \text{(3.16f)}\\
& \mathbf{y} \in [0,1]^n && \text{(3.16g)}\\
& \mathbf{x} \in \{0,1\}^n && \text{(3.16h)}\\
& x_0 \in [0,1]. && \text{(3.16i)}
\end{aligned}$$

For any solution $\mathbf{x}^k \in \mathcal{X}$, constraint (3.16d) forces the binary variable t^k to 1 if the cost of projects in $\mathbf{x}^k$ that are subsidized by $\mathbf{y}$ does not strictly exceed the budget B_r. In other words, t^k is set to 1, if $\mathbf{x}^k$ is a feasible solution to `RECIPIENT(y)`. We have modeled this using a parameter ϵ to avoid open feasible sets. In the case t^k is set to 1, any solution we pick must be at least as good as $\mathbf{x}^k$ with respect to the recipient's objective for it to be bi-level feasible. Constraint (3.16e) ensures this. $\mathcal{X}$ is subset of all projects in I and can have exponentially many of them. We solve (EBKP) iteratively. At each iteration we obtain a solution $(\mathbf{y}^*, (\mathbf{x}^*, x_0^*))$. We then determine if $(\mathbf{x}^*, x_0^*) \in P(\mathbf{y}^*)$ by solving `RECIPIENT`$(\mathbf{y}^*)$. If $(\mathbf{x}^*, x_0^*) \in P(\mathbf{y}^*)$, then we terminate otherwise `RECIPIENT`$(\mathbf{y}^*)$ returns an optimal solution $(\bar{\mathbf{x}}, \bar{x}_0) \in$

$P(\mathbf{y}^*)$ that we add as constraints of the form (3.16d) and (3.16e). Since these constraints are of the "If-then" nature, we have to introduce a binary variable for every such constraint. This is given in Algorithm 1. This procedure is very similar to the one proposed in Lozano & Smith (2017). They aggregate their constraints (3.16d) and add a single constraint for all $\mathbf{x}^k$. This makes sense when the upper-level decision variables are present in many different constraints at the lower-level. We, however, have a single constraint at the lower-level in which the upper-level variable is present. We add them as dis-aggregated constraints that provide a tighter relaxation. This does not affect the running time as one new inequality of the form (3.16e) and one new variable have to be added at every iteration in Lozano & Smith (2017). We instead add two new inequalities and one new variable at every iteration. In addition, in order to deal with open feasible sets and ill-posedness of the problem, in Lozano & Smith (2017), the authors assumed integer restrictions on upper-level variables. We show that for sufficiently small ϵ, our algorithm would terminate at optimality, which we discuss in Theorem 3.

Algorithm 1: Enumeration Scheme for DR-BKP

Solve R-DR-BKP and Let $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ be its optimal solution;
Solve RECIPIENT$(\mathbf{y}^*)$ and let $(\bar{\mathbf{x}}, \bar{x}_0)$ be the optimal solution;
Set $k = 0$, $(\mathbf{x}^k, \bar{x}_0^k) = (\bar{\mathbf{x}}, \bar{x}_0)$, Set UB $= \mathbf{w}^T\mathbf{x}^*$, LB $= \mathbf{w}^T\mathbf{x}^k$;
while $\frac{(UB-LB)}{LB} \leq$ gap **do**
  if $\mathbf{v}^T\mathbf{x}^k + v_0x_0^k > \mathbf{v}^T\mathbf{x}^* + v_0x_0^*$ **then**
    if $\mathbf{w}^T\mathbf{x}^k = \mathbf{w}^T\mathbf{x}^*$ **then**
      Return $(\mathbf{x}^k, x_0^k, \mathbf{y}^*)$
    else
      Set LB $= \max(\text{LB}, \mathbf{w}^T\mathbf{x}^k)$;
      Add following constraints to R-DR-BKP:
      $\mathbf{c}^T\mathbf{x}^k - \sum_{i\in I} c_ix_i^ky_i + M_1t^k \geq B_r + \epsilon$;
      $\mathbf{v}^T\mathbf{x} + v_0x_0 - \sum_{i\in I} c_i'x_i^ky_i + M_2(1-t^k) \geq \mathbf{v}^T\mathbf{x}^k + \frac{v_0}{c_0}(B_r - \mathbf{c}^T\mathbf{x}^k)$;
    end
  else
    Return $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$
  end
  Set $k = k + 1$;
  Solve R-DR-BKP and Let $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ be its optimal solution;
  Set UB $= \mathbf{w}^T\mathbf{x}^*$;
  Solve RECIPIENT$(\mathbf{y}^*)$ and Let $(\mathbf{x^k}, x_0^k)$ be the optimal solution;
end

Algorithm 1 gives the Enumeration Scheme to find bi-level optimal solution for the DR-BKP problem. The R-DR-BKP is solved first using an MILP solver and from the solution, $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$, subsidy $\mathbf{y}^*$ is used to solve RECIPIENT$(\mathbf{y}^*)$. If $(\mathbf{x}^*, x_0^*) \in P(\mathbf{y}^*)$, then $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ is returned as a solution, else the constraints of type (3.16d) and (3.16e) corresponding to some optimal solution $(\mathbf{x}^k, x_0^k) \in P(\mathbf{y}^*)$ are added to R-DR-BKP and solved again. Note that if $\mathbf{w}^T\mathbf{x}^k = \mathbf{w}^T\mathbf{x}^*$ then $(\mathbf{x}^k, \lfloor \mathbf{y}^* \rfloor)$ is an alternative optimal to the current iteration of R-DR-BKP, where

$\lfloor \mathbf{y}^* \rfloor$ is obtained by setting y_i^* to 0 if $x_i^k = 0$ and to y_i^* otherwise. Feasibility is easy to see because $(\mathbf{x}^k, x_0^k) \in P(\lfloor \mathbf{y}^* \rfloor)$ and we check for optimality in the condition $\mathbf{w}^T\mathbf{x}^k = \mathbf{w}^T\mathbf{x}^*$. Whether we use $\lfloor \mathbf{y}^* \rfloor$ or $\mathbf{y}^*$ as subsidy to obtain $\mathbf{x}^k$ does not matter to our original bi-level DR-BKP problem. Every time R-DR-BKP is solved, upper bound of the problem is updated to the obtained donor profit. The lower bound of the problem is the donor profit obtained with the projects selected by RECIPIENT($\mathbf{y}^*$) and it gets updated every time the inner problem is solved. The algorithm runs till bounds are within some predefined gap.

Theorem 3. *Algorithm 1 terminates at an optimal solution.*

Proof. We enumerate the set of integer solutions in $\mathcal{X}$, i.e, a subset of projects in every iteration. And at every iteration, we enumerate a new subset of projects and there are finitely many of them so the algorithm terminates in finite time. We say a subsidy $\tilde{\mathbf{y}}$ is feasible for a subset of projects $\tilde{\mathbf{x}}$ if $\sum_{i \in I}(c_i - c_i\tilde{y}_i)\tilde{x}_i \leq B_r$ and infeasible otherwise. The formulation looks for a set of projects $\mathbf{x}^*$ and corresponding subsidy $\mathbf{y}^*$ that is better than any subset of projects $\tilde{\mathbf{x}}$ (with regards to the inner objective) if the subsidy $\mathbf{y}^*$ is feasible for $\tilde{\mathbf{x}}$. The solution is then obviously bi-level feasible. In order to see that it is also optimal to the DR-BKP, first observe that we used the parameter ϵ in (3.16d) to avoid strict inequalities. Let us refer to the theoretical model obtained from (EBKP) with (3.16d) replaced by the strict inequality

$$\mathbf{c}^T\mathbf{x}^k - \sum_{i \in I} c_i x_i^k y_i + M_1 t^k > B_r \qquad \forall k \in \{1, \ldots, K\} \tag{3.17}$$

as (OBKP). The strict inequalities require us to search for a solution in an open feasible set. So, we instead use (EBKP). This, however is not an issue if ϵ is sufficiently small. Let ϵ_1 be numerical tolerance used to solve (EBKP). Clearly, we need $\epsilon > \epsilon_1$, otherwise we can set the t^k to 0 instead of the actual value of 1 in (3.16d). The possible issue arises when we do not consider a subsidy $\tilde{\mathbf{y}}$ that is feasible for (OBKP) but infeasible for (EBKP). This happens when $\tilde{\mathbf{y}}$ is infeasible

for some subsets of projects $\{\bar{\mathbf{x}}^1, \ldots, \bar{\mathbf{x}}^\kappa\} \subseteq \mathcal{X}$ but not by $\epsilon - \epsilon_1$ amount, i.e., for each $k = 1, \ldots, \kappa$

$$B_r < \mathbf{c}^T \bar{\mathbf{x}}^k - \sum_{i \in I} c_i \bar{x}_i^k \tilde{y}_i < B_r + \epsilon - \epsilon_1, \tag{3.18}$$

$$\sum c_i \bar{x}_i \tilde{y}_i + (B_r - \mathbf{c}^T \bar{\mathbf{x}}^k) < 0 < \sum c_i \bar{x}_i \tilde{y}_i + (B_r - \mathbf{c}^T \bar{\mathbf{x}}^k) + \epsilon - \epsilon_1. \tag{3.19}$$

We first assume $\epsilon = \frac{n+1}{n}\epsilon_1$ and we will soon make the reasoning for this assumption clear. Since this ensures $\epsilon > \epsilon_1$, this is a valid assumption. In addition, with a sufficiently small tolerance, we can assume $\epsilon < 1$. We can now take a component of $\tilde{\mathbf{y}}$ that is non-zero, say i for which $\bar{x}_i^k = 1$ and decrease this value by $\frac{\epsilon - \epsilon_1}{c_i}$. The idea behind this is that by doing this reduction, we can make the infeasibility of the reduced $\tilde{\mathbf{y}}$ for $\bar{\mathbf{x}}^k$ by at least $\epsilon - \epsilon_1$. Note that not all $\tilde{y}_i$ with $\bar{x}_i^k = 1$ can be strictly less than $\frac{\epsilon - \epsilon_1}{c_i}$. If this is true, then

$$0 \leq \sum_{i \in I} c_i \bar{x}_i^k \tilde{y}_i < \sum_{i \in I} c_i \frac{\epsilon - \epsilon_1}{c_i} \bar{x}_i^k \leq \sum_{i \in I} \frac{\epsilon_1}{n} \bar{x}_i^k \leq \epsilon_1.$$

Since B_r and $\mathbf{c}^T \bar{\mathbf{x}}^k$ are both integers, one cannot satisfy (3.19) unless $\epsilon > 1$. Now we need to do reduction of components of $\tilde{\mathbf{y}}$ for every subset in $\{\bar{\mathbf{x}}^1, \ldots, \bar{\mathbf{x}}^\kappa\}$ and in the worst case, we could reduce every component of $\tilde{\mathbf{y}}$. Let us call this reduced subsidy vector $\hat{\mathbf{y}}$. For some $(\tilde{\mathbf{x}}, \tilde{x}_0, \tilde{\mathbf{y}})$ feasible for (OBKP), we want to show that $(\tilde{\mathbf{x}}, \tilde{x}_0, \hat{\mathbf{y}})$ is feasible for (EBKP). Now by construction $\hat{\mathbf{y}}$ will be feasible for constraint (3.16d). Since we are only reducing the value of $\tilde{\mathbf{y}}$ to get $\hat{\mathbf{y}}$, (3.16a), (3.16c) and (3.16e) are also feasible. In order to show feasibility of (3.16b), we first observe our assumption of $\epsilon = \frac{n+1}{n}\epsilon_1$. From feasibility of

$(\tilde{\mathbf{x}}, \tilde{x}_0, \tilde{\mathbf{y}})$ to (OBKP), we have

$$\mathbf{c}^T\tilde{\mathbf{x}} + c_0\tilde{x}_0 \leq B_r + \mathbf{c}^T\tilde{\mathbf{y}}, \tag{3.20}$$

$$\mathbf{c}^T\tilde{\mathbf{x}} + c_0\tilde{x}_0 \leq B_r + \mathbf{c}^T\hat{\mathbf{y}} + n(\epsilon - \epsilon_1), \tag{3.21}$$

$$\mathbf{c}^T\tilde{\mathbf{x}} + c_0\tilde{x}_0 \leq B_r + \mathbf{c}^T\hat{\mathbf{y}} + \epsilon_1. \tag{3.22}$$

□

3.4 Branching algorithm

In Section 3.3, we have seen enumeration algorithm where two cuts are added to the R-DR-BKP every time there is an optimal solution $(\mathbf{y}^*, (\mathbf{x}^*, x_0))$ to R-DR-BKP but $(\mathbf{x}^*, x_0) \notin P(\mathbf{y}^*)$. After the cuts are added, R-DR-BKP is resolved again until the bi-level optimal solution is achieved. A mixed-integer program is solved iteratively and in addition we introduce two new constraints and a binary variable at every iteration. An alternative approach has been proposed by Xu & Wang (2014), where $(\mathbf{y}^*, (\mathbf{x}^*, x_0))$ is eliminated from search using a branching rule. The branching rule proposed in Xu & Wang (2014) cannot be directly used for our problem for two reasons. First they require that upper-level variables that are involved in the lower-level are discrete. In addition they require that the upper-level variables do not have non-linear interaction with lower-level variable. Neither of these are true in our model. We provide a modified branching rule that addresses these issues and handles the elimination of $(\mathbf{x}^*, x_0) \notin P(\mathbf{y}^*)$ from search space but none of the bi-level feasible solutions.

The pseudo-code of branching algorithm is given in Algorithm 2. The branching rule is created only when an incumbent solution $(\mathbf{y}^*, (\mathbf{x}^*, x_0^*))$ is found. The usual rules based on bounds cannot be applied anymore. First `RECIPIENT(`$\mathbf{y}^*$`)` is solved which returns an optimal solution $(\bar{\mathbf{x}}, \bar{x}_0^*)$. If $\mathbf{v}^T\bar{\mathbf{x}} + v_0\bar{x}_0 = \mathbf{v}^T\mathbf{x}^* + v_0x_0^*$, then we can prune the node as $(\mathbf{y}^*, (\mathbf{x}^*, x_0^*))$ is a bi-level feasible solution. Else the solution $(\mathbf{y}^*, (\bar{\mathbf{x}}, \bar{x}_0))$ is appended in a queue generated to store potential solutions

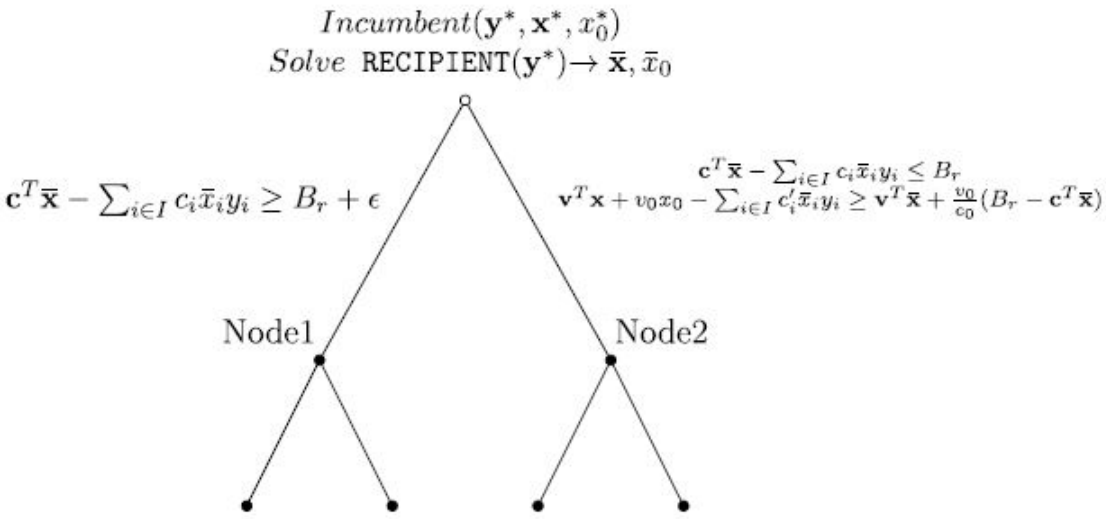


Figure 3.1: Branching from an incumbent solution

that need to be branched (called BrQueue) and then the solution $(\mathbf{y}^*, (\mathbf{x}^*, x_0^*))$ is rejected. Every time the branching callback is activated and there is at least one solution in the BrQueue, the solution with maximum donor profit is used to branch upon. As shown in Figure 3.1, there are two branches generated. Node 1 is explored where a valid subsidy $\mathbf{y}$ is such that $\bar{x}$ is infeasible for `RECIPIENT(y)`. Node 2 is explored where a valid solution $(\mathbf{y}, \mathbf{x}, x_0)$ is such that $\bar{\mathbf{x}}$ is feasible for `RECIPIENT(y)` and $\mathbf{v}^T\mathbf{x} + v_0 x_0 \geq \mathbf{v}^T\bar{\mathbf{x}} + v_0 x_0(\bar{\mathbf{x}}, \mathbf{y})$. This idea is similar to constraints (3.16d) and (3.16e) in `MILP-DR-BKP`.

Algorithm 2 terminates with an optimal solution. When there is an incumbent to the R-DR-BKP, say $(\mathbf{y}^*, (\mathbf{x}^*, x_0^*))$, we solve RECIPIENT$(\mathbf{y}^*)$ and get solution $(\mathbf{y}^*, (\bar{\mathbf{x}}, \bar{x}_0))$. The search space is divided into two, (1) one is explored where a valid subsidy $\mathbf{y}$ is such that $\bar{\mathbf{x}}$ is infeasible for RECIPIENT$(\mathbf{y})$, and (2) the other is explored where a valid solution $(\mathbf{y}, \mathbf{x}, x_0)$ is such that $\bar{\mathbf{x}}$ is feasible for `RECIPIENT(y)` and $\mathbf{v}^T\mathbf{x} + v_0 x_0 \geq \mathbf{v}^T\bar{\mathbf{x}} + v_0 x_0(\bar{\mathbf{x}}, \mathbf{y})$. The division of search space into two is similar to the idea of EBKP formulation of the DR-BKP. So instead of adding two new constraints and a binary variable and refining the solution space using new bounds as per the EBKP, the solution space gets divided in two parts

in every branching call in Algorithm 2. In both algorithms, the only space that remains un-searched is that for the subsidies y_i that remain in $\epsilon - \epsilon_1$. We may lose some bi-level feasible subsidy values in this range and related project selections. However, this will not occur as shown in Theorem 3. If either of the solutions feasible to OBKP (where in all subsidies are searched) say $(\tilde{\mathbf{x}}, \tilde{x}_0, \tilde{\mathbf{y}})$ is reduced by amount $\epsilon - \epsilon_1$ for at least one project i, resulting in a solution $(\tilde{\mathbf{x}}, \tilde{x}_0, \hat{\mathbf{y}})$, it is feasible to EBKP.

Algorithm 2: Branching approach for DR-BKP

Define a queue, BrQueue = [];
Solve R-DR-BKP and Let $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ be its optimal solution;
Solve RECIPIENT$(\mathbf{y}^*)$ and let $(\bar{\mathbf{x}}, \bar{x}_0)$ be its optimal solution;
Set $k = 0$, $(\mathbf{x}^k, \bar{x}_0^k) = (\bar{\mathbf{x}}, \bar{x}_0)$, UB $= \mathbf{w}^T\mathbf{x}^*$, LB $= \mathbf{w}^T\mathbf{x}^k$;
while $\frac{(UB-LB)}{LB} \leq$ gap **do**
  if $\mathbf{v}^T\mathbf{x}^k + v_0x_0^k > \mathbf{v}^T\mathbf{x}^* + v_0x_0^*$ **then**
    if $\mathbf{w}^T\mathbf{x}^k = \mathbf{w}^T\mathbf{x}^*$ **then**
      Return $(\mathbf{x}^k, x_0^k, \mathbf{y}^*)$
    else
      Set LB $= \max($LB$, \mathbf{w}^T\mathbf{x}^k)$;
      Append $(\bar{\mathbf{x}}, \bar{x}_0)$ in BrQueue ;
      Reject solution $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$;
    end
  else
    Return $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$
  end
  if BrQueue has at least one solution set **then**
    Select solution from BrQueue that yields maximum profit, say $(\bar{\mathbf{x}}, \bar{x}_0)$;
    Make branches as per Figure 3.1
  end
  Solve R-DR-BKP and Let $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ be its optimal solution;
  Set UB $= \mathbf{w}^T\mathbf{x}^*$;
  Solve RECIPIENT$(\mathbf{y}^*)$ and Let $(\mathbf{x^k}, x_0^k)$ be the optimal solution;
end

3.5 Computational experiments

To understand, analyze and compare the performance of both proposed algorithms, a computational study has been performed. We have used an HP com-

	Class1			Class2			Class3				DBudget	CBudget
DataSet	N1	P/C1	$\alpha 1$	N2	P/C2	$\alpha 2$	N3	P/C3	$\alpha 3$	γ	(% of TotalHCProjectsCost)	
1	10	1	[1,1]	10	0.7	[1,1]	10	0.5	[1,1]	[1,1]	20	30
2	20	1	[1,1]	20	0.7	[1,1]	20	0.5	[1,1]	[1,1]	20	30
3	40	1	[1,1]	40	0.7	[1,1]	40	0.5	[1,1]	[1,1]	20	30
4	100	1	[1,1]	100	0.7	[1,1]	100	0.5	[1,1]	[1,1]	20	30
5	10	1	[1,1]	10	0.7	[1,1]	10	0.5	[1,1]	[1,1]	10	15
6	10	1	[1,1]	10	0.7	[1,1]	10	0.5	[1,1]	[1,1]	5	7
7	10	1	[1,1]	10	0.7	[1,1]	10	0.5	[1,1]	[0.5,1]	20	30
8	10	1	[1,1]	10	0.7	[1,1]	10	0.5	[1,1]	[1,1.5]	20	30
9	10	1	[1,1]	10	0.7	[1,1]	10	0.5	[1,1]	[1.5,2]	20	30
10	10	1	[1,1]	10	0.7	[1,1]	10	0.5	[1,1]	[2,2.5]	20	30
11	10	1	[1,1]	10	0.7	[1,1]	10	0.5	[1,1]	[2.5,3]	20	30
12	3	1	[0.01,0.5]	24	1	[0.5,1.5]	3	1	[1.5,5]	[1,1]	20	30
13	10	1	[1,10]	10	0.7	[1,1]	10	0.5	[1,1]	[1,1]	20	30
14	10	1	[0.1,1]	10	0.7	[1,1]	10	0.5	[1,1]	[1,1]	20	30
15	10	1	[0.01,0.1]	10	0.7	[1,1]	10	0.5	[1,1]	[1,1]	20	30

Table 3.1: Input parameters for data generation

puter (Windows 10 Enterprise with 64-bit operating system, 3.19 GHz processor and 8GB RAM) for the experiments. Both algorithms are coded and solved in Python 3.8 using CPLEX 20.1.0.

3.5.1 Data generation

Instance generation has been guided by the real-world instance presented in Morton et al. (2018). There are 15 data sets [1] generated as shown in Table 3.1 and 10 instances are generated and solved in every data set. The first data set has a total 30 projects (10 in each of the three classes, as given in columns N1, N2 and N3). Classes of the projects are made based on their profit to cost ratios of recipient, as given in columns P/C1, P/C2 and P/C3. These division of projects in classes have been made to understand the allocations preferred by donor and recipient. A parameter called α is used in each class here to influence the leader or follower's decisions. The profit of a project for the donor is the profit of recipient for that project scaled by the parameter $\alpha 1$ in Class 1, $\alpha 2$ in Class 2, and $\alpha 3$ in Class 3. Both donor and recipient budgets are generated as a percentage of the total cost of healthcare projects (columns DBudget and CBudget) to be considered for funds allocation. For the profit values of the external project, a parameter called γ has been used. Profit to cost ratio of external project in each instance is the

[1] All data used in this work are available at https://github.com/ashwin-1983/DR-BKP/

average of profit to cost ratios of all healthcare projects in the instance scaled by an input parameter called γ. The cost values of healthcare and external projects are random integers in [5000, 10000] and [1000000, 2000000] respectively. In every instance, there are $(2n + 1)$ variables and $(n + 2)$ constraints where n is the total number of projects.

Each data set has perturbation in one of the parameters with respect to the first data set. These step-by-step changes on the data sets are made to understand the performance of the developed algorithms on every parameter in the instances generated. For the second, third, and fourth data sets, the number of projects has been increased to 20 projects, 40 projects, and 100 projects in each class respectively. The donor and recipient budgets have been decreased in data set 5 from 20% and 30% to 10% and 15% of total cost of healthcare projects respectively, and further more for data set 6. The γ value has been maintained to 1 for all other data sets except for data sets 7 to 11. The range of γ values has been increased gradually in these data sets. In the case of data set 12, a combination of changes in the parameters 'number of projects in each class' and 'range of α values' have been made. In further data sets, only the ranges of α values have been changed for Class 1 projects. It will be useful to understand how the project allocations and/or time to solve these instances are affected by the divergence in priorities of donor and recipient.

3.5.2 Results

We have conducted computational experiments to compare the performance of both algorithms at two different tolerance parameters (at $\epsilon = 1e{-}2$ and $\epsilon = 1e{-}4$) for a set time limit of 3600 seconds. It is critical to determine big M values used to solve a single-level reformulation of any BLPP, *i.e.* the reformulation EBKP of the DR-BKP in our case. Kleinert et al. (2020) show that validating that a given big M does not cut off any bi-level optimal solution is as hard as solving the original bi-level problem when the single-level reformulation is based

on Karush-Kuhn-Tucker (KKT) conditions of the lower-level problem. However, the big M approach can be adopted without this issue when valid values of big M can be determined based on problem-specific knowledge (Kleinert & Schmidt 2023). Since we do not use KKT conditions to reformulate the DR-BKP and have problem-specific knowledge like values of project cost and budget, we go ahead with using the big M approach. The big M values used in the EBKP are set such that they just exceed the right hand side of related constraints. The value of big M, M_1, in constraint (3.16c) is set to the tightest possible value - the right hand side of the constraint *i.e.* $B_r + 1$ (B_r is budget of the recipient). The value of big M, M_2, in constraint (3.16d) can be either set to right hand side of the constraint as it changes in every iteration or set to a constant value of $\sum v_i + v_0$ which is not as tight as the former. To see the impact of the differing values on the performance of the algorithm, we solve all the instances using both of these: (1) $M_2 = M_2^k = \mathbf{v}^T\mathbf{x}^k + \frac{v_0}{c_0}(B_r - \mathbf{c}^T\mathbf{x}^k) \;\; \forall k \in 1, \ldots, K$ and (2) $M_2 = \sum v_i + v_0$.

The minimum, average, and maximum solution times of both algorithms for solving the 10 instances in each of the data sets are given in Table 3.2 when $\epsilon = 1e-2$ and Table 3.3 when $\epsilon = 1e-4$. These are the data sets that are solved within the set time limit. In cases of data sets 14 and 15, none of the instances are solved to optimality within the set time limit. Their minimum, average, and maximum solution gaps at the termination of algorithms are given in Table 3.4 when $\epsilon = 1e-2$ and Table 3.5 when $\epsilon = 1e-4$.

It can be observed from the result tables that as the number of projects increases and hence the number of variables in data sets 2, 3, and 4 compared to data set 1, the average solution time increases for both algorithms in case of lower tolerance parameter. However, in the case of higher tolerance parameter, both algorithms take less time to solve. From data set 1, 5, and 6, we can observe that the algorithms take less time to solve if the budgets are lower for similar-sized instances. When parameter γ is increased (data sets 8 to 11 as compared to data sets 1 and 7), *i.e.* external project has higher profit and starts competing with

DataSet	BranchingAlgo			EnumerationAlgo					
				$bigM_2 = M_2^k$			$bigM_2 = M_2$		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1	0.238	0.348	0.434	0.101	0.157	0.348	0.088	0.171	0.251
2	0.047	0.325	0.916	0.062	0.108	0.341	0.047	0.081	0.203
3	0.049	0.309	0.454	0.062	0.142	0.214	0.080	0.183	0.430
4	0.066	0.458	1.036	0.078	0.166	0.325	0.078	0.168	0.258
5	0.062	0.096	0.133	0.056	0.069	0.094	0.062	0.074	0.093
6	0.078	0.138	0.250	0.061	0.086	0.181	0.061	0.122	0.291
7	0.045	0.085	0.166	0.042	0.084	0.184	0.048	0.118	0.241
8	0.167	2.416	18.089	0.096	14.540	102.491	0.131	12.712	95.759
9	2.116	17.089	43.792	4.100	21.041	59.338	4.918	19.015	56.586
10	1.247	28.504	74.147	6.780	165.973	250.540	6.654	167.280	277.944
11	8.369	62.662	149.575	44.327	114.624	200.056	31.662	131.119	246.953
12	0.057	0.074	0.125	0.047	0.069	0.094	0.047	0.072	0.094
13	0.047	0.192	0.345	0.055	0.103	0.212	0.062	0.110	0.175

Table 3.2: Average solution time (in seconds) for data sets that are solved within time limit ($\epsilon = 1e-2$)

DataSet	BranchingAlgo			EnumerationAlgo					
				$bigM_2 = M_2^k$			$bigM_2 = M_2$		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1	0.273	0.599	1.665	0.109	0.448	1.493	0.109	0.445	1.666
2	0.266	0.639	0.994	0.156	0.351	1.366	0.173	0.329	0.830
3	0.328	2.203	5.471	0.270	2.540	10.742	0.283	2.390	11.762
4	5.625	13.370	37.202	0.368	65.958	187.436	0.480	67.152	185.231
5	0.067	0.128	0.165	0.060	0.091	0.199	0.058	0.088	0.217
6	0.095	0.158	0.266	0.057	0.079	0.099	0.063	0.088	0.187
7	0.104	0.289	0.875	0.052	0.222	1.251	0.047	0.353	2.404
8	0.453	10.961	47.005	0.105	35.436	145.728	0.109	26.219	103.939
9	4.601	22.661	49.743	5.316	32.232	111.013	5.408	28.578	92.281
10	2.009	36.293	90.415	8.335	200.500	301.985	12.251	205.050	302.571
11	10.963	72.226	156.519	50.653	128.790	283.821	43.976	147.279	295.884
12	0.053	0.085	0.141	0.040	0.076	0.142	0.047	0.065	0.078
13	0.329	0.574	1.050	0.101	0.250	0.640	0.099	0.238	0.610

Table 3.3: Average solution time (in seconds) for data sets that are solved within time limit ($\epsilon = 1e-4$)

DataSet	BranchingAlgo			EnumerationAlgo					
				$bigM_2 = M_2^k$			$bigM_2 = M_2$		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
14	6.7%	15.4%	31.2%	5.4%	19.5%	33.5%	7.0%	19.5%	34.8%
15	49.4%	65.6%	81.4%	50.3%	87.3%	109.2%	50.3%	87.3%	109.2%

Table 3.4: Average solution gaps for data sets that are not solved within time limit ($\epsilon = 1e-2$)

DataSet	BranchingAlgo			EnumerationAlgo					
				$bigM_2 = M_2^k$			$bigM_2 = M_2$		
	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
14	8.8%	18.4%	31.2%	3.4%	19.5%	34.8%	5.0%	19.7%	34.8%
15	74.7%	87.7%	101.5%	76.8%	92.8%	109.2%	76.8%	92.8%	109.2%

Table 3.5: Average solution gaps for data sets that are not solved within time limit ($\epsilon = 1e - 4$)

the healthcare projects for the recipient budget, branching algorithm performs significantly better than the enumeration algorithm for both tolerance limits.

Another complexity of the healthcare funds allocation problem is the divergence between valuations of projects by donor and recipient. If the α value increases above 1 in either of the three classes of projects, the donor values its projects more than the recipient does in that particular class. Else if the α value is below 1, the donor values its projects lesser than the recipient. As seen in data sets 13 to 15, the range of α values for class 1 are decreased gradually. In data set 13 where the donor values its projects more than the recipient, all instances are solved using both algorithms very fast. However, none of the instances from data sets 14 and 15 are solved where the donor values its projects lesser than the recipient (refer to Table 3.4 and Table 3.5).

While comparing the performance of the enumeration algorithm for both mentioned big M values, there is hardly any difference observed in the solution times of all the data sets except for data sets 7 to 11. For these particular data sets, the γ value is increased gradually. However, since the solution times are not consistently lower in either for either of the big M values, we keep this open for further research.

It can be observed from these results that there is evidence to believe that the branching algorithm performs better when the instances are generated with more complexity where there is a greater discrepancy in the valuation of the projects by the two players.

3.6 Conclusion

We have carried out a complexity study and computational experiments on DR-BKP that has been introduced in Morton et al. (2018). We have first shown that the problem is well-posed. We have then extended and adapted the algorithms proposed for discrete bi-level problems to DR-BKP and have proved its convergence. We have provided some complexity results for the problem. A predominant issue of having continuous upper-level decision variables in lower-level constraints is the non-compact feasible set. This complicates both proving the existence of a solution and the convergence of algorithms. We have observed that a guaranteed solution exists when the upper-level objective function is discrete, involving only the lower-level variables, and the solution set is finite. A simple enumeration of the solutions in the finite set and evaluation of their bi-level feasibility would provide this. This is generalizable and goes beyond DR-BKP. The convergence of the known enumeration schemes tends to work for our problem despite the continuous upper-level variables. These enumeration schemes aim to restrict the search space to a closed set by cutting off the open feasible set at a threshold. This is easier to do when we have integer upper-level variables. Despite having continuous upper-level variables, we have shown that these enumeration schemes work for our problem for sufficiently small thresholds. We have shown this by constructing an equivalent solution in the closed set for any valid solution cut-off. This is dependent on the problem structure and the generalizability of this procedure is not clear, keeping a number of questions still open for further research.

Chapter 4

Genetic algorithm for the DR-BKP

The Donor-Recipient Bi-level Knapsack Problem (DR-BKP) has been introduced in chapter 1. Starting with a discussion of the problem's complexity in chapter 3, we have proposed two exact solution approaches and conducted computational experiments to test their performance on a wide range of instances generated using realistic data. There are fifteen data sets with ten instances in each data set. The nature of data sets has been changed by either increasing the number of healthcare projects, varying the budgets of participants, varying the cost to profit ratio of external project as compared to those of the healthcare projects, or varying valuations of healthcare projects by the donor as compared to that by the country. After studying the results for these generated instances, we hav realized that the solution time for data sets that are complex in nature grows significantly, especially for the data sets that vary the valuations of healthcare projects by the donor as compared to that by the country. Since a heuristic-based approach is flexible and has the potential to provide good quality solutions in a reasonable time, we take this approach in this chapter.

We have developed a meta-heuristics-based approach to address the complex data sets. A genetic algorithm, which is a specific type of evolutionary algorithm, has been developed and presented in this chapter. Evolutionary algorithms are search algorithms inspired by natural selection principles of genetics. Evo-

lutionary Bi-level Optimization (EBO) algorithms are gaining attention due to their flexibility, implicit parallelism, and ability to customize for specific problem-solving tasks (Sinha et al. 2018). These algorithms allow flexibility for customizing the solution procedure by enabling any heuristics or rules to be embedded in their operators (Deb et al. 2020).

In Bi-level Programming Problems (BLPPs), the upper-level problem is sensitive to the quality of lower-level solutions. Hence challenge here is the computational effort needed to solve the lower-level problem to a reasonable accuracy for every upper-level solution. This also offers an advantage to use two different optimization methods for the upper and lower-level problems depending on their complexities (Deb et al. 2020). However, such approaches may be computationally expensive for large scale problems. We have explored this avenue using the developed genetic algorithm for DR-BKP and comparing its performance with the exact solvers for the varying data sets we have generated previously.

Contribution: The main contributions of this chapter are

1. A genetic algorithm has been developed for solving the DR-BKP. We have solved the upper-level of the problem using the genetic operators. The lower-level problem can be solved either using a heuristic approach or an exact approach; we have used both these with a greedy heuristic and an available solver respectively. We have given performance results of the genetic algorithm using both these approaches.

2. The results obtained for all the instances using the developed genetic algorithm have been compared with that by the exact solvers as presented in chapter 3. These are promising results to solve complex large instances that are closer to realistic scenarios.

The DR-BKP has been formally defined in section 3.1 in chapter 3. We introduce genetic algorithm followed by its adoption for solving the DR-BKP in section 4.1.

The results of solving all data sets with the genetic algorithm and their comparison with exact solvers have been given in section 4.2. We have concluded this chapter in the last section with a discussion and findings of the algorithm and suggestions for future research.

4.1 Genetic algorithm

Meta-heuristics are a family of approximate optimization techniques that provide "acceptable" solutions in suitable time to solve complex optimization problems for which exact solution approaches are unable to be efficient (Talbi 2013). Typically, a meta-heuristic consists of two steps for the search procedure, *viz.*, (a) initializing with a population of solutions and (b) improving the population of solutions by searching new solutions using a set of rules. A genetic algorithm is a meta-heuristic for generating high-quality solutions for optimization problems, inspired by the ideas of natural evolution and genetics (Holland 1992). In a genetic algorithm, the search for good solutions starts with a population of solutions. These solutions are refined and evolved to find better solutions by biologically inspired processes like selection, mutation and crossover over a series of iterations referred to as generations. The fittest solutions from population in every generation are selected for reproduction to generate offspring solutions that enter the next generation population. Since the genetic algorithm operates on a set of solutions, it can be harnessed to generate several solutions that meet given criteria.

Genetic algorithm for the DR-BKP:

A genetic algorithm has a framework to find good solutions denoted by individuals in the process of evolving or improving a constant-sized population of solutions over given number of generations. The solutions are referred to as chromosomes and these are encoded as strings of symbols. A gene represents the position of a

symbol and an allele represents value of the symbol. Solutions that are feasible are evaluated for fitness and high performing solutions are selected wherein the fitness of a solution is generally determined by the objective value of the optimization problem. The selected solutions reproduce to generate offspring solutions which inherit features of their high performing parents.

There are different approaches in the literature to handle the two optimization levels in the problem - (1) Nested sequential approach, (2) Single-level transformation approach, (3) Co-evolutionary approach and, (4) Multi-objective approach as given by Talbi (2013). A detailed description of each of these approaches has been given in section 2.4 from chapter 2.

We have used a nested sequential approach in this chapter. It is one of the easy approaches to use heuristic methods for solving BLPPs since it does not require any assumptions on convexity or differentiability of the lower-level problem, as

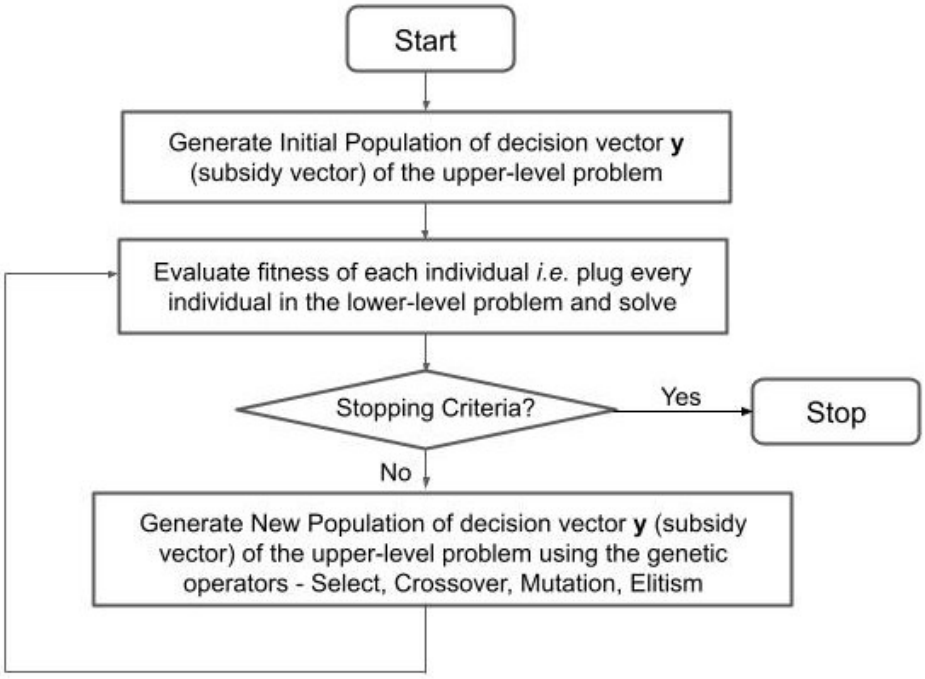


Figure 4.1: Block diagram of genetic algorithm for the DR-BKP

can be seen in Calvete et al. (2008) and Hejazi et al. (2002). The original formulation of both optimization levels is achieved by the decoupling of the optimization levels in the nested approach (Arroyo & Fernández 2013). After decoupling both optimization levels, the upper-level problem has been solved first using a genetic algorithm, followed by solving the parameterized lower-level problem. To solve the lower-level problem, we can use either an exact solver or a heuristic solution method. Although the bi-level methods that solve both leader's and follower's problems heuristically can only find semi-feasible solutions, these are worth developing when both problems are difficult to solve exactly (Alekseeva & Kochetov 2013). The bi-level meta-heuristics can provide a quick solution. Since such solutions are not feasible to the original bilevel problem, we could retrieve feasibility by solving the inner problem exactly parametrized by the final subsidies calculated by the heuristics.

A diagrammatic representation of the genetic algorithm that we propose for the DR-BKP is shown in Figure 4.1 and its detailed description is as following.

Algorithm framework

Firstly, we introduce the following definition before discussing the framework of the algorithm. The High Point Relaxation (HPR) is the formulation obtained after relaxing the lower-level objective function of the original model. The HPR of the DR-BKP can be given as follows and it is referred as Relaxed Donor-Recipient Bi-level Knapsack Problem (R-DR-BKP). This will be used subsequently in the evaluation of candidate solutions. The DR-BKP is given in (3.2) and (3.3) in chapter 3.

Problem R-DR-BKP:

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{4.1a}$$

$$\text{subject to} \quad \mathbf{c}^T\mathbf{y} \leq B_d \tag{4.1b}$$

$$\mathbf{c}^T\mathbf{x} + c_0x_0 \leq B_r + \mathbf{c}^T\mathbf{y} \tag{4.1c}$$

$$y_i \leq x_i \qquad \forall i \in I \tag{4.1d}$$

$$\mathbf{y} \in [0,1]^n \tag{4.1e}$$

$$\mathbf{x} \in \{0,1\}^n \tag{4.1f}$$

$$x_0 \in [0,1]. \tag{4.1g}$$

To remove the bi-linear term in (3.3b) in the DR-BKP, constraint (4.1d) has been introduced. It will ensure that if a project is not selected, there will be no subsidy allocated to that project.

In a genetic algorithm, it is important to define the fitness function of the solutions *i.e.* the individuals in the evolution process. The evaluated fitness of the individuals and population in each generation forms the basis of the search algorithm. Each of the structural components of the genetic algorithm are given below:

4.1.1 Solution coding

A candidate solution to the DR-BKP (*i.e.* decision vector $\mathbf{y}$) is represented by an individual in the population. Each of these individuals are encoded as an array of length n where n is the size of decision vector $\mathbf{y}$ (*i.e.* number of healthcare projects that require funding). The subsidy of every healthcare project is a gene and its value is the allele of chromosomes of the individuals. At the start of the algorithm, an initial population of these individuals is provided.

Every generated individual is considered upper-level feasible, if it is within the donor's budget. All feasible individuals in the upper-level are then sent to

the lower-level problem. The individuals that are found feasible to the lower-level problem are given a fitness value which is nothing but the upper-level objective achieved using this individual. We discuss the fitness of individuals in detail in the next section.

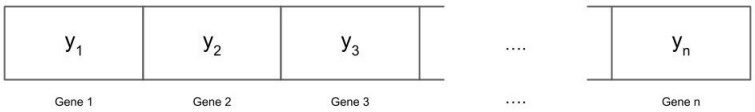


Figure 4.2: Solution coding of an example individual in the population

Figure 4.2 shows an example of solution coding of an n-sized individual. The allele of each gene is value y_i which represents the proportion of cost that the donor will subsidize for each project $i \in I$. Since y_i is a continuous variable, we have restricted these values to five decimal places in order to avoid numerical inconsistencies. The actual subsidy values are obtained from these cost proportions ranging from 0 to 1, up to five decimal points and multiplied by four digit cost values.

4.1.2 Evaluation function

The competence of an individual is determined by its fitness value computed using a fitness function. For the DR-BKP, the fitness of an individual solution is determined by the upper-level objective achieved using this solution. Let $\mathbf{y}^*$ be an individual solution, its fitness will be given by optimal value of the following instance:

$$\text{maximize} \quad \mathbf{w}^T \bar{\mathbf{x}} \tag{4.2a}$$

$$\text{subject to} \quad \mathbf{y}^* \in Y \tag{4.2b}$$

$$\bar{\mathbf{x}} \in \arg\max(\texttt{RECIPIENT}(\mathbf{y}^*)). \tag{4.2c}$$

The parameterized lower-level problem (as given in 1.3) with given upper-level solution $\mathbf{y}^*$ is represented by RECIPIENT($\mathbf{y}^*$). Solution of RECIPIENT($\mathbf{y}^*$) gives the decision vector $\bar{\mathbf{x}}$ and fitness of individual $\mathbf{y}^*$. $\bar{\mathbf{x}}$ represents the project allocations optimal to lower-level problem when subsidy $\mathbf{y}^*$ is received. Using these project allocations, the upper-level objective $\mathbf{w}^T\bar{\mathbf{x}}$ determines fitness of the individual solution $\mathbf{y}^*$.

RECIPIENT($\mathbf{y}^*$) is a 0-1 knapsack problem with one continuous variable. It can be solved using either a heuristic or an exact method to find objective values of the parameterized lower-level problem for each individual $\mathbf{y}^*$. We use both these approaches since it will be interesting to see the impact of non-optimal lower-level solutions on the upper-level objectives. Although a greedy method to solve the lower-level problem and a genetic algorithm to solve the upper-level problem give an approximate solution, we have performed computational experiments using this to find and analyze the quality of these quick solutions. We have reported true solutions to these approximate solutions. We have compared the performance of these solutions along with the ones obtained using the genetic algorithm that uses exact lower-level solutions. The greedy heuristic has been described along with its pseudo-code in the next section. After giving details of the evolution process of the genetic algorithm in the section 4.1.3, we give the pseudo-codes for both approaches (see Algorithm 4 for genetic algorithm where lower-level is solved using an off-the-shelf exact solver and Algorithm 5 for genetic algorithm where lower-level is solved using the greedy heuristic).

Greedy heuristic for lower-level problem

A greedy heuristic makes choice of picking the largest valuable item (in our case, a project) until the budget has run out. It has been introduced by Dantzig (1957) and later several variants of the greedy heuristic have been proposed and seen in the literature (Kellerer et al. 2004).

A feasible individual $\mathbf{y}^*$ obtained by the genetic algorithm running for the upper-level problem is plugged in the lower-level problem given in 1.3.
RECIPIENT($\mathbf{y}^*$) is solved using the greedy heuristic. After finding subsidized cost $(c_i - c_i y_i^*)$ of each healthcare project i, the projects with zero subsidized costs are selected by the lower-level problem and those projects that have non-zero subsidized costs along with the external project are ranked on the basis of their profit to cost ratio. Let k be the number of projects that have non-zero costs where $k \leq n+1$. Let π be the ranking of these k projects *i.e.* it maps the ranks to the original ordering of projects. Note that $\frac{v_0}{c_0}$ can be at the start, in between or at the end of this ranking which we denote by ratio R. We have just given a representation of the ranking order of the projects in 4.3:

$$\frac{v_{\pi(1)}}{c_{\pi(1)} - c_{\pi(1)} y_{\pi(1)^*}} \geq \cdots \geq R \geq \cdots \geq \frac{v_{\pi(k)}}{c_{\pi(k)} - c_{\pi(k)} y^*_{\pi(k)}}. \tag{4.3}$$

Starting from the highest ranked project to the lowest rank, all projects are considered sequentially. A project is selected if its cost is within the leftover lower-level budget. At the end, if the external project is not allocated any funds during the sequential allocation, the leftover lower-level budget is allocated to the external project using the Equation 4.4:

$$\bar{x}_0 = \frac{B_r - \sum_{i \in I} c_i (\bar{x}_i - \bar{x}_i y_i^*)}{c_0} \tag{4.4}$$

where, $\bar{x}_i$ is project selection as per the greedy heuristic $\forall i \in I$. The pseudo-code for the greedy heuristic is given in Algorithm 3.

Algorithm 3: Greedy Heuristic to solve lower-level problem

A feasible individual solution (y^*) is received from the upper-level;

for $i \in I$ **do**
  if $y_i^* = 1$ **then**
    $x_i^* = \bar{x}_i = 1$
  end
end

for $i \in \{\pi(1), \ldots \pi(k)\}$ **do**
  if $B_r > 0$ **then**
    if $i = R$ and $c_0 <= B_r$ **then**
      $\bar{x}_0 = 1$;
      $B_r = B_r - c_0$
    else
      if $c_i - c_i y_i^* <= B_r$ **then**
        $\bar{x}_i = 1$;
        $B_r = B_r - (c_i - c_i y_i^*)$
      end
    end
  else
    break ;
  end
end

if $\bar{x}_0 = 0$ and $B_r > 0$ **then**
  $\bar{x}_0 = \frac{B_r}{c_0}$
end

Return fitness value $= \mathbf{w}^T \bar{\mathbf{x}}$

4.1.3 Evolution process

At the start, the size of the population and the number of generations are defined. To make an initial population, first a large set of random individuals are generated and these are included in the initial population if they satisfy the upper-level constraints. For every individual in the initial population, its fitness is computed and these fitness values are used in further generations development. In every new generation, the old population is replaced by new individual solutions using the operators - selection, crossover, mutation and elitism. These operators are as follows:

Selection:

To select parents of individuals for reproducing in the next generation, the roulette wheel scheme is used in this proposed genetic algorithm. In this scheme, individuals are obtained randomly with the probability that is proportional to the fitness of the individuals *i.e.* the **y** vectors which are individuals in our case, are selected with higher probability if they give higher donor profit (fitness value). The number of parents selected is the size of the population and they are then paired for crossover to generate two child individuals. Hence after moving to the next generation, the size of the population in maintained.

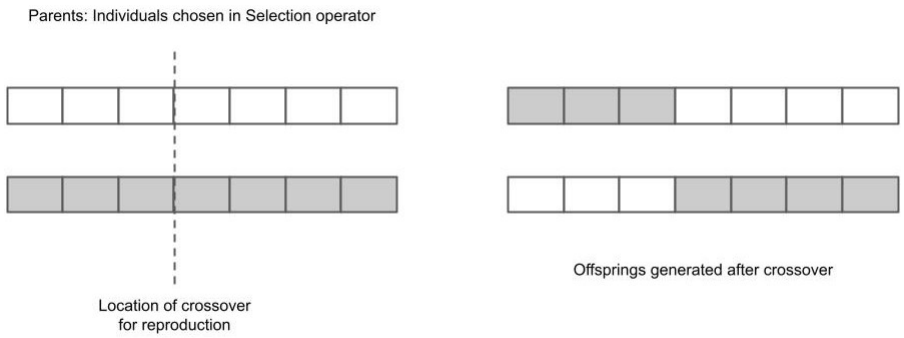


Figure 4.3: Diagrammatic representation of an example single-point crossover in a pair of parent individuals

Crossover:

After the parent individuals are paired, there are different ways in which crossover can be implemented. In the proposed genetic algorithm, a single-point crossover has been implemented. In a single-point crossover, a random location is chosen in the parents' array. The divided arrays of each of the parents are now swapped to generate the children as shown in an example in Figure 4.3.

Mutation:

It is important to incorporate new features to the population that is getting evolved over a course of generations which allows the algorithm to explore new regions of the solution space. Hence the new individuals are mutated with a predefined probability (called as mutation rate) to include diversity in the new population. In this algorithm, mutation is implemented by selecting a gene of an individual and changing its allele *i.e.* the proportion of cost subsidized for the chosen project is replaced with a new random value.

Elitism:

In this genetic operator, a set of high performing solutions are carried forward in the next generation of the evolution process. This assures that there is a reference maintained in the promising areas of the search space in all the generations.

Criteria for termination:

The genetic algorithm can be terminated based on varying criteria like maximum number of generations of the algorithm, solution time limit and acceptable solution gap. In the proposed genetic algorithm, we terminate it if maximum number of generations are completed.

Algorithm 4: Genetic algorithm for DR-BKP, using an exact solver for the lower-level problem

Set number of generations = 30, and $ni = 0$;

Set size of population of individual solutions = 750;

Generate an initial population of upper-level variables $\mathbf{y}$, pop_{init};

$pop \leftarrow pop_{init}$;

while $ni < 30$ **do**

  for each $\mathbf{y}$ in pop **do**

    $y^* \leftarrow y$;

    Solve RECIPIENT($\mathbf{y}^*$) using an exact solver, let the obtained solution be $(\bar{\mathbf{x}}, \bar{x}_0)$;

    Allocate fitness value to individual solution = $\mathbf{w}^T\bar{\mathbf{x}}$;

  end

  Get $neopop$ using Selection(pop), Crossover(pop),

  Mutation(pop), and Elitism(pop);

  $pop \leftarrow neopop$;

  $ni++$;

end

Return $(\bar{\mathbf{x}}, \bar{x}_0, \mathbf{y}^*)$ that has the highest fitness value

Algorithm 5: Genetic algorithm for DR-BKP, using a heuristic method to solve the lower-level problem

Set number of generations = 30, $ni = 0$;
Set size of population of individual solutions = 750;
Generate an initial population of upper-level variables $\mathbf{y}$, pop_{init};
$pop \leftarrow pop_{init}$;
while $ni < 30$ **do**
    for each $\mathbf{y}$ in pop **do**
        $y^* \leftarrow y$;
        Solve RECIPIENT($\mathbf{y}*$) using a greedy method as per algorithm 3, let the obtained solution be $(\bar{\mathbf{x}}, \bar{x}_0)$;
        Allocate fitness value to individual solution = $\mathbf{w}^T\bar{\mathbf{x}}$;
    end
    Get $neopop$ using Selection(pop), Crossover(pop),
    Mutation(pop), and Elitism(pop);
    $pop \leftarrow neopop$;
    $ni++$;
end
Return $(\bar{\mathbf{x}}, \bar{x}_0, \mathbf{y}^*)$ that has the highest fitness value;
Solve RECIPIENT($\mathbf{y}^*$) using an exact solver to report true optimal solution

4.2 Computational experiments

We have conducted computational experiments to evaluate the performance of the genetic algorithm with that of the exact solvers given in chapter 3. There are fifteen varying data sets, each data set consisting of ten instances generated randomly. First, we have solved just one of the instances in each data set, and present their solution time and conversion of solutions by the genetic algorithm.

This is followed by giving the performance comparison of the genetic algorithm with the branching algorithm for test data sets 1 to 7. We then have solved and compared all the remaining data sets with all the proposed algorithms

4.2.1 Experimental settings

The genetic algorithm has been coded in Python 3.10 and the experiments have been performed on a DELL computer (Windows Enterprise with 64-bit operating system, 1.30 GHz processor and 16.0 GB RAM). Both exact solvers have been solved with two different optimality tolerance parameters (at $\epsilon = 1e-2$ and $\epsilon = 1e-4$). In this chapter, we have used results of $\epsilon = 1e-2$ for performance comparison of the algorithms.

To generate an initial population of the genetic algorithm, we have used the upper-level decision vector $\tilde{\mathbf{y}}$ that yields a lower bound of the instance. The lower bound of an instance is found by solving its HPR (given by model 4.1) and plugging the upper-level decision vector in the lower-level problem. We have used CPLEX 20.1.0 to solve HPR of an instance. One of the individuals in the initial population is $\tilde{\mathbf{y}}$. To generate other individuals, we select either one or two projects and change the subsidy proportions in $\tilde{\mathbf{y}}$.

The parameters of the genetic algorithm used in this computational study are empirically chosen after completing some preliminary experiments. We have a population size of 750 individual solutions and the algorithm is executed for 30 generations. The genetic operator crossover has been set to single-point crossover at a random location in the array of individuals and the mutation rate has been set to 0.9. We have maintained 10% elite population of high performing individuals in the evolution process.

As discussed in subsection 4.1.2, either a heuristic or an exact method can be used to solve the lower-level problem for every individual solution $\mathbf{y}^*$ in the evaluation process of the algorithm. We have used both of these, the pseudo-code for the greedy heuristic is given in Algorithm 3 and we use CPLEX 20.1.0 to find

exact solutions in the latter approach.

4.2.2 Results

First, we used the genetic algorithm to solve the first instance in each of the fifteen data sets given in chapter 3, and present these results in Table 4.1. Followed by this, the performance of the genetic algorithm is compared with that of the branching algorithm on few of the data sets that are easily solvable by the exact solvers (see Table 4.3 in section 4.2.2). Finally, we give summary results of solving all instances in each of the data sets using the genetic algorithm in section 4.2.2 (see Table 4.4).

Since the branching algorithm comparatively performed better than the enumeration algorithm for most of the data sets as can be seen in subsection 3.5.2 in chapter 3, we compare the performance of the genetic algorithm with the branching algorithm in Table 4.1. In this table, the upper bound (`UB`), upper-level profit obtained (`z_br`), and solution time in seconds (`t_br`) of the branching algorithm are reported for each of the solved instances with the genetic algorithm. Alongside these values, we give the upper-level profit obtained when the lower-level problem is solved using greedy heuristic (`z_ga-heur`), upper-level profit obtained when lower-level problem is solved using exact solver (`z_ga-exact`) and time (in seconds) to achieve these values (`t_ga-heur`) and (`t_ga-exact`) respectively as the generations progress (`GenN`). We report only the solutions obtained for a few generations out of a total 30 in the algorithm.

The genetic algorithm with a greedy heuristic for lower-level problem performs equally well with the branching algorithm for data sets 1 to 11, and 13. The solution quality is good and it is achieved very early in the evolution process with solution gaps lesser than 0.1 %. For the case of data sets 12, 14, and 15, wherein there are healthcare projects with changing α values, the solution quality is not good. This indicates that the genetic algorithm is not able to evolve and generate a better population of solutions after a few generations when there is divergence

DS	Inst	BranchingAlgorithm			GeneticAlgorithm				
		UB	z_br	t_br	GenN	z_ga-heur	t_ga-heur	z_ga-exact	t_ga-exact
1	1	103659	103364	0.365	0	101640	0.091	101133	0.000
					5	102610	93.320	102808	60.898
					10	103256	261.472	102808	160.009
					15	103256	544.961	102808	300.061
					20	103364	969.192	102808	482.196
					25	103448	1572.797	102808	726.718
					30	103624	2397.534	103342	1008.608
2	1	199263	199232	0.062	0	199232	0.026	195104	0.001
					5	199232	75.790	197145	74.818
					10	199232	242.300	198713	226.572
					15	199232	527.317	198713	490.900
					20	199232	961.775	198713	905.519
					25	199232	1592.558	198713	1292.690
					30	199232	2427.101	198713	1723.470
3	1	405743	404481	0.409	0	405280	0.063	388276	0.001
					5	405520	68.455	392325	114.600
					10	405622	234.627	401946	280.795
					15	405622	515.463	403761	492.470
					20	405622	899.263	404600	755.805
					25	405622	1438.144	404600	1059.094
					30	405622	2247.206	404600	1425.203
4	1	1017941	1017941	0.219	0	1015626	0.140	770933	0.001
					5	1016560	56.577	792134	140.729
					10	1016572	179.475	802220	361.155
					15	1017513	393.036	821628	605.577
					20	1017513	884.067	830237	881.060
					25	1017513	1574.543	842376	1258.793
					30	1017513	2574.831	857130	1577.947
5	1	54541	54343	0.109	0	54343	0.040	54541	0.001
					5	54343	86.904	54541	187.570
					10	54343	272.042	54541	450.457
					15	54343	589.225	54541	791.684
					20	54343	1075.403	54541	1210.937
					25	54343	1743.755	54541	1720.352
					30	54343	2643.287	54541	2291.207
6	1	28074	28074	0.116	0	25538	0.054	28074	0.001
					5	26502	105.156	28074	121.494
					10	26502	310.287	28074	301.258
					15	26502	663.867	28074	536.789
					20	26502	1017.324	28074	827.930
					25	26502	1425.912	28074	1193.948
					30	26502	1966.269	28074	1627.125
7	1	105029	104732	0.045	0	104732	0.056	104732	0.000
					5	104926	123.707	104873	80.752
					10	104926	393.505	104926	199.769
					15	104926	864.920	104926	358.996
					20	104926	1513.799	104926	561.914
					25	104926	2330.057	104926	802.129
					30	104926	3277.299	104926	1091.966
8	1	96220	96139	18.090	0	94325	0.043	85215	0.007
					5	95850	158.056	90495	136.066
					10	95850	496.098	93833	339.370
					15	96124	1060.207	95850	606.206
					20	96124	1717.405	96176	937.479
					25	96124	2387.902	96176	1358.604
					30	96124	3813.574	96176	1843.421

Table 4.1: Results for data sets solved using genetic algorithm and compared with results given by branching algorithm

DS	Inst	BranchingAlgorithm			GeneticAlgorithm				
		UB	z_br	t_br	GenN	z_ga-heur	t_ga-heur	z_ga-exact	t_ga-exact
9	1	107374	106953	26.386	0	106657	0.061	50813	0.001
					5	106841	172.549	69738	63.328
					10	106981	377.831	72877	151.640
					15	107352	688.847	77062	268.021
					20	107352	1137.793	77062	413.463
					25	107352	1794.658	78623	590.126
					30	107352	3085.603	78623	795.264
10	1	99832	99758	15.185	0	99588	0.033	66679	0.000
					5	99770	77.544	75339	60.366
					10	99812	249.782	81638	143.426
					15	99812	571.634	82949	255.818
					20	99827	1032.902	86395	397.021
					25	99827	1693.552	87076	563.965
					30	99827	2538.128	88214	763.137
11	1	81271	81271	26.632	0	80667	0.030	42355	0.001
					5	81265	121.043	54008	63.556
					10	81265	375.032	58828	150.256
					15	81265	781.214	62955	264.533
					20	81269	1388.431	70227	407.624
					25	81270	2222.499	70227	579.071
					30	81270	3331.431	70227	781.741
12	1	199686	198684	0.064	0	117974	0.030	191401	0.000
					5	163219	78.059	196396	72.127
					10	171570	245.703	197328	173.277
					15	175922	537.789	197328	310.267
					20	177761	967.689	197328	482.674
					25	181730	1578.903	197328	687.723
					30	182300	2391.932	197328	929.418
13	1	378679	378565	0.260	0	376459	0.028	375302	0.000
					5	378586	81.392	377192	84.303
					10	378586	256.763	378586	216.356
					15	378586	563.819	378586	397.604
					20	378586	1035.684	378586	628.542
					25	378586	1939.833	378586	919.370
					30	378586	3477.612	378586	1432.773
14	1	80907	72197	3600.114	0	72218	0.027	75623	0.009
					5	72218	77.588	77573	89.385
					10	73857	244.641	77948	242.765
					15	73857	524.936	78879	483.411
					20	73857	950.609	79095	755.752
					25	73857	1545.262	80207	1094.692
					30	73857	2354.257	80207	1532.895
15	1	70523	47206	3600.087	0	50783	0.053	43575	0.000
					5	52397	81.593	47358	113.029
					10	59294	251.329	51592	297.571
					15	60975	525.733	53839	550.410
					20	63030	928.222	55802	838.614
					25	64010	1492.392	56120	1203.000
					30	64469	2246.385	59472	1651.631

Table 4.1: Results for data sets solved using genetic algorithm and compared with results given by branching algorithm (continued)

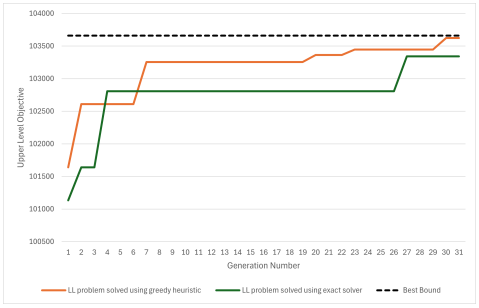
DS	Inst	UB	z_br	z_ga-exact	z_ga-heur	true-z_ga-heur
1	1	103659	103364	103342	103624	99051
2	1	199263	199232	198713	199232	153097
3	1	405743	404481	404600	405622	307071
4	1	1017941	1017941	857130	1017513	886735
5	1	54541	54343	54541	54343	54541
6	1	28074	28074	28074	26502	26502
7	1	105029	104732	104926	104926	104036
8	1	96220	96139	96176	96124	52169
9	1	107374	106953	78623	107352	47238
10	1	99832	99758	88214	99827	29786
11	1	81271	81271	70227	81270	20120
12	1	199686	198684	197328	182300	172092
13	1	378679	378565	378586	378586	361705
14	1	80907	72197	80207	73857	69767
15	1	70523	47206	59472	64469	62341

Table 4.2: True values of the upper-level objective for instances solved using genetic algorithm for upper-level problem and greedy heuristic for lower-level problem

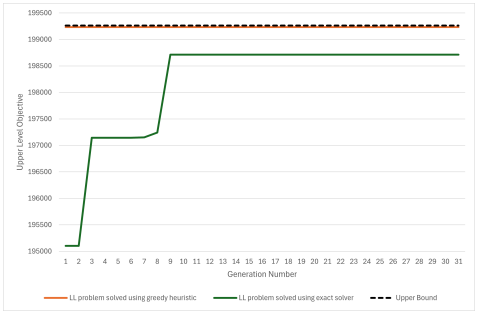
in the project valuations by the donor and recipient country.

Since the greedy heuristic is approximate, we report the true bi-level optimal solutions of the solved instances in Table 4.2. To find a true bi-level optimal solution (`true-z_ga-heur`) related to a heuristic solution (`z_ga-heur`), we first solve the lower-level problem exactly using the heuristic upper-level solution (*i.e.* the subsidy) to find the optimal lower-level allocation. The upper-level objective with this optimal lower-level allocation is the true solution of the heuristic solution. The only data sets that have good quality true solutions are 5-7 and 13. In the case of data sets 8-11, the solution quality is particularly very low. Since the γ values of these data sets are higher than the others, it can be observed that if the profit to cost ratio of external project is higher than the average profit to cost of healthcare projects, the greedy heuristic poorly performs to optimally allocate the projects.

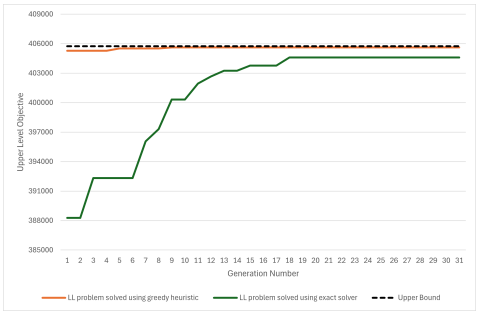
The genetic algorithm with an exact solver for the lower-level problem obtains bi-level optimal solutions for data sets 1 to 3, 5 to 8, and 12 to 14 (see Table 4.1).



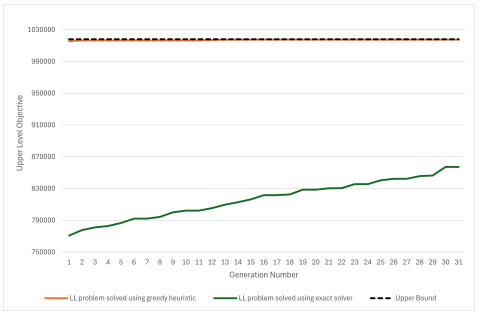
(a) Data Set 1 Instance 1



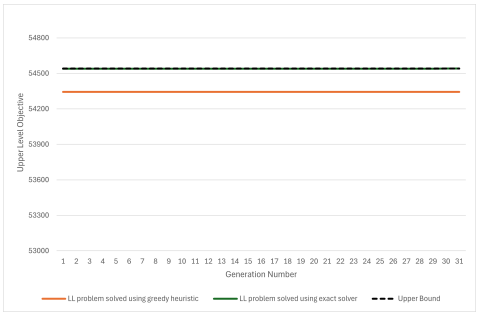
(b) Data Set 2 Instance 1



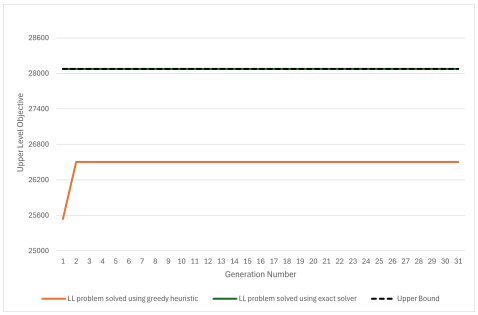
(c) Data Set 3 Instance 1



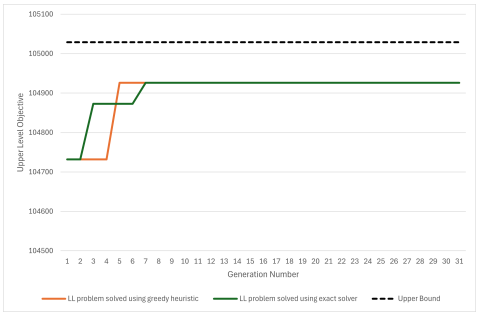
(d) Data Set 4 Instance 1



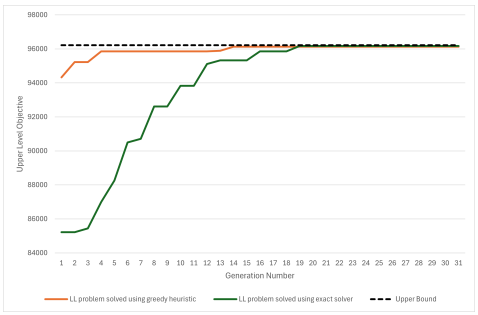
(e) Data Set 5 Instance 1



(f) Data Set 6 Instance 1

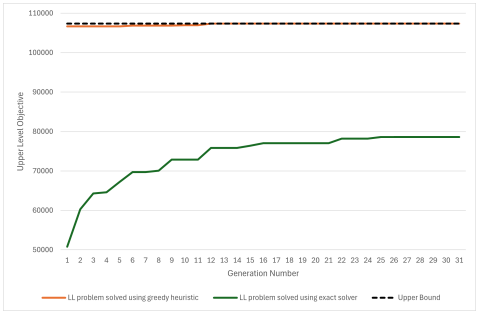


(g) Data Set 7 Instance 1

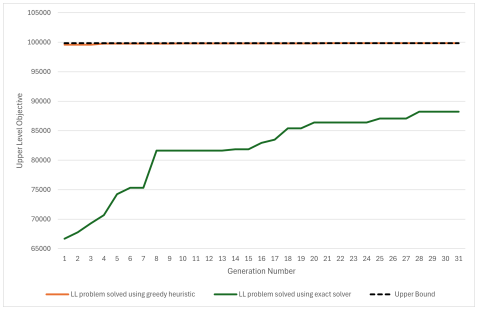


(h) Data Set 8 Instance 1

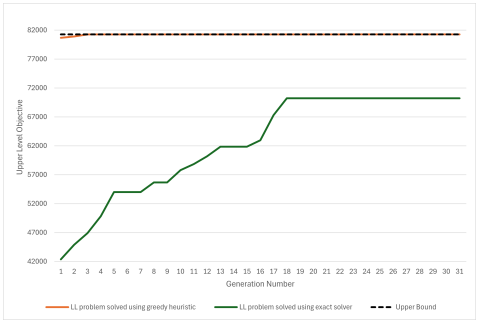
Figure 4.4: Evolution of population for genetic algorithm solved for the upper-level problem where lower-level is solved using (1) greedy heuristic and (2) exact solver



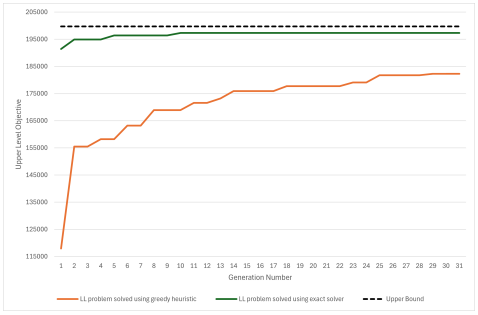
(i) Data Set 9 Instance 1



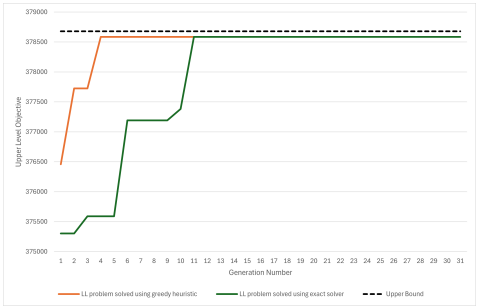
(j) Data Set 10 Instance 1



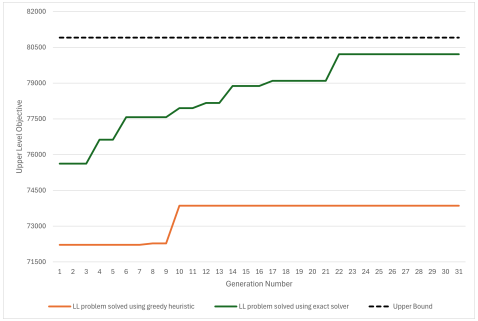
(k) Data Set 11 Instance 1



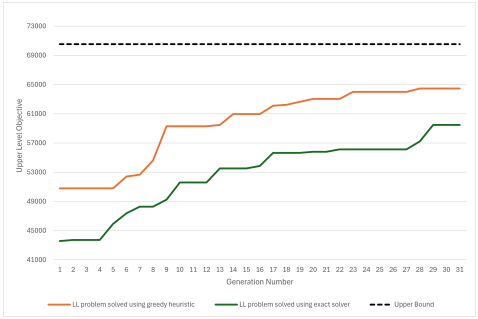
(l) Data Set 12 Instance 1



(m) Data Set 13 Instance 1



(n) Data Set 14 Instance 1



(o) Data Set 15 Instance 1

Figure 4.4: Evolution of population for genetic algorithm solved for the upper-level problem where lower-level is solved using (1) greedy heuristic and (2) exact solver (continued)

The effectiveness of this approach can be particularly seen in data sets 14 and 15 which are a few of the complex data sets that could not be solved by the enumeration and branching algorithms given in chapter 3. The algorithm has converged to a bi-level optimal solution within approximately 1100 seconds for data set 14. Although this was not the case for data set 15, an improved bound has been achieved within 1500 seconds, with a far improved solution gap of approximately 18.5 %, where the percentage solution gap is given by Equation 4.5. The optimality gap of the solution obtained for this instance with the branching algorithm is approximately 49.4 %.

$$\%g_{UL} = \frac{UB - \hat{Z}_{UL}}{\hat{Z}_{UL}} \times 100 \tag{4.5}$$

where,

UB: upper bound of the upper-level solution and,

$\hat{Z}_{UL}$: upper-level solution obtained within the given time limit.

However there are few data sets wherein this approach did not work effectively. Firstly as can be seen in Figure 4.4, the convergence of solutions with this approach is slower compared to the genetic algorithm with a greedy heuristic for the lower-level problem, especially for data sets 2, 3, and 4. The number of projects increases eventually resulting in more decision variables in these data sets. Secondly, solutions for data sets 9 to 11 did not converge after 30 generations. These data sets have higher γ values as compared to other data sets which means that the profit to cost ratio of external profit is higher as compared to the average profit to cost of the healthcare projects. Even with a large population size of 750 individuals, the best-performing solution amongst the initial population is low. The increments in solutions in every generation are slow and cease to occur after a certain number of generations.

The genetic algorithm with a greedy heuristic for the lower-level problem

is not sufficient to give good solutions for complex data sets like data sets 8 to 12, 13, and 14. However, these solutions converge faster towards the best bound as compared to the solutions obtained with the genetic algorithm with an exact solver for lower-level problem. These two approaches can be combined to achieve faster good quality solutions in the future. For example, when the genetic algorithm progresses, few of the best solutions obtained using greedy heuristic can be plugged into an exact solver to find optimal allocations. This new set of solutions can then be reused in the population of the heuristic approach so that the low-quality solutions are discarded from the search process.

Following from the findings so far, we continue using only the results that are obtained using the genetic algorithm with an exact solver for the lower-level problem in the rest of the chapter.

Test data sets

Data sets 1 to 7 have been easily solved using both exact solvers (see chapter 3). Hence, we have used these data sets to test the performance of the genetic algorithm against the branching algorithm since it has generally performed better than the enumeration algorithm. In Table 4.3, the average solution time (in seconds) of all the instances in each of the data sets has been given. The solution time that we report here is the time when the genetic algorithm first finds a solution within an acceptable optimality gap during the search. For the data sets in which few of the instances could not be solved completely, we report their average solution gaps followed by a bracket that includes the number of instances that could not be solved to optimality. For example, in the case of data set 3, all instances have been solved by both branching and enumeration algorithms. However, only six instances are solved completely by the genetic algorithm whose solution time is reported first. And for the remaining four instances, we report their solution gaps at the bottom of the particular row.

The genetic algorithm has given optimal solutions to all instances in all data

DS	BranchingAlgorithm			GeneticAlgorithm		
	Min	Avg	Max	Min	Avg	Max
1	0.238	0.348	0.434	36.760	162.205	383.707
2	0.047	0.325	0.916	30.880	267.679	994.403
3	0.049	0.309	0.454	20.704	270.865	412.956
				2.4% (4)	4.2% (4)	6.4% (4)
4	0.066	0.458	1.036	1484.448	1484.448	1484.448
				6.4% (9)	14.9% (9)	21.5% (9)
5	0.062	0.096	0.133	14.318	25.934	34.250
6	0.078	0.138	0.250	14.307	21.180	42.551
7	0.045	0.085	0.166	12.845	49.471	180.563

Table 4.3: Summary of solution time (in seconds) and solution gaps of all instances solved in data sets 1 to 7 using (a) branching algorithm, and (b) genetic algorithm

sets except data sets 3 and 4 in acceptable solution time. Four out of ten instances in data set 3 could not be solved with an average solution gap of 4.2%, and nine out of ten instances in data set 4 could not be solved with an average solution gap of 14.9%. This clearly indicates that the genetic algorithm is not performing well when the number of variables (number of projects) increases. However, the results are promising enough to explore how the genetic algorithm performs when the data sets are complex wherein the number of projects remains the same.

All data sets for performance comparison of the proposed algorithms

Finally, we compare the performance of all three proposed solution methods - (1) Enumeration Algorithm, (2) Branching Algorithm, and (3) Genetic Algorithm and present in Table 4.4. We have solved all the instances in each of the data sets and compared their solution quality. For the data sets wherein all the instances are solved to optimality, their minimum, average, and maximum solution time (in seconds) are given. The solution time that we report here is the time when the algorithm first finds a solution within an acceptable optimality gap during the search. For the data sets in which few of the instances could not be solved completely, we report their minimum, average, and maximum solution gaps followed by a bracket that includes the number of instances that could not be solved to optimality. For example, in the case of data set 3, all instances have been solved

DS	BranchingAlgorithm			EnumerationAlgorithm			GeneticAlgorithm		
	Min	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**
1	0.238	0.348	0.434	0.101	0.157	0.348	36.760	162.205	383.707
2	0.047	0.325	0.916	0.062	0.108	0.341	30.880	267.679	994.403
3	0.049	0.309	0.454	0.062	0.142	0.214	20.704	270.865	412.956
							2.4% (4)	4.2% (4)	6.4% (4)
4	0.066	0.458	1.036	0.078	0.166	0.325	1484.448	1484.448	1484.448
							6.4% (9)	14.9% (9)	21.5% (9)
5	0.062	0.096	0.133	0.056	0.069	0.094	14.318	25.934	34.250
6	0.078	0.138	0.250	0.061	0.086	0.181	14.307	21.180	42.551
7	0.045	0.085	0.166	0.042	0.084	0.184	12.845	49.471	180.563
8	0.167	2.416	18.089	0.096	14.540	102.491	32.325	191.879	490.245
9	2.116	17.089	43.792	4.100	21.041	59.338	1.2% (10)	14.6% (10)	65.9% (10)
10	1.247	28.504	74.147	6.780	165.973	250.540	8.6% (10)	15.9% (10)	49.7% (10)
11	8.369	62.662	149.575	44.327	114.624	200.056	6.9% (10)	11.8% (10)	15.7% (10)
12	0.057	0.074	0.125	0.047	0.069	0.094	10.889	143.618	929.418
13	0.047	0.192	0.345	0.055	0.103	0.212	9.567	17.057	30.353
14	6.7% (10)	15.4% (10)	31.2% (10)	5.4% (10)	19.5% (10)	33.5% (10)	12.387	1341.482	3689.977
							1.6% (5)	5.0% (5)	17.2% (5)
15	49.4% (10)	65.6% (10)	81.4% (10)	50.3% (10)	87.3% (10)	109.2% (10)	10.3% (10)	17.5% (10)	27.8% (10)

Table 4.4: Summary of solution time (in seconds) and solution gaps of all instances solved in each data set using (a) enumeration algorithm (from subsection 3.5.2 in chapter 3), (b) branching algorithm (from subsection 3.5.2 in chapter 3) and (c) genetic algorithm where lower-level problem is solved using an exact solver

by both branching and enumeration algorithms. However, only six instances are solved completely by the genetic algorithm whose summary of solution time is reported first. And for the remaining four instances, we report a summary of their solution gaps at the bottom of the particular row. For the data sets wherein none of the instances could be solved to optimality, a summary of their solution gap is given followed by the number 10 in a bracket since all 10 instances remain unsolved.

Out of all the data sets, data sets 9 to 11 and data sets 14 and 15 are closer to realistic scenarios however these have been found difficult to solve. Both enumeration and branching algorithms have solved all instances in data sets 9 to 11 to optimality in less time. However, the genetic algorithm has terminated without optimal solutions after a given limit on generations and with high solution gaps. Contrasting to this, the genetic algorithm has performed effectively in the case of data sets 14 and 15 as compared to the exact solvers. It has given optimal solutions for five of the ten instances in data set 14. For the remaining five instances, the obtained solution gaps are improved as compared to those given by the exact solvers. For data set 15, the branching and enumeration algorithms could only achieve solution gaps of at most 49.4% and 50.3% respectively, whereas the genetic algorithm has managed to achieve 27.8% solution gap or better over the ten instances.

The results for each of the instance in every data set solved using all the proposed algorithms are uploaded on Github[1]. A summary file is included in this results folder that gives tables provided in this chapter.

4.3 Conclusion

In this chapter, we have given a genetic algorithm with a nested procedure to solve the DR-BKP. Being a difficult problem to solve exactly especially for complex and large instances that are close to real-life scenarios, we took a meta-heuristic-based

[1] https://github.com/ashwin-1983/DR-BKP/tree/main/GeneticAlgorithm_Results

Data sets	Changes in the data sets	Algorithm with better performance	Observations
1 to 4	Number of healthcare projects in each class is increased	Branching algorithm	Solution time increases with number of variables, this is addressed by the branching algorithm
5 to 6	Donor and recipient budgets are decreased	All algorithms	Lower budget does not complicate the problem
7 to 11	Valuation of external project is increased	Branching algorithm	Leads to increase in number of constraints and branches, this is addressed by the branching algorithm
12 to 15	Divergence in project valuations by both participants is increased	Genetic algorithm	Both exact methods cannot reach optimal solutions, the genetic algorithm can improve bounds faster

Table 4.5: Summary of performance comparison of all the proposed algorithms with respect to changes in the data sets

approach to address the DR-BKP. The DR-BKP has a knapsack at both upper and lower levels. In the developed genetic algorithm, the upper-level problem is dealt with by the genetic operators and a population of solutions (*i.e.* subsidies). The population evolves over a set of generations wherein the fitness of solutions is given by the upper-level objective achieved using the lower-level allocation done after using the particular solution or subsidies.

A performance summary of both exact algorithms and the genetic algorithm has been given in Table 4.5, for the groups of data sets based on key changes that have been done to generate these data sets. Although the genetic algorithm has not performed well on the complex data sets where valuations of the external projects are increased (especially, data sets 9, 10, and 11), it shows potential to at least improve the bounds of solutions much faster as compared to both of the exact solvers. The exact solvers are ineffective in these sets since the cuts generated in the enumeration algorithm cease to reduce the solution space effectively after certain iterations and the number of branches increases so large in the branching algorithm that the search process becomes extremely slow. The genetic algorithm approach can be used along with these exact solvers to generate cuts at a faster rate to reduce the feasible solution space during the exact solution procedures. It is an interesting direction that can be taken for further research

in Bi-level Knapsack Problems. The developed algorithm is also flexible enough to be extended to single-leader multi-follower bi-level knapsack problems.

Chapter 5

Generalization and applications of interest

In this chapter, we have given generalizations of the proposed solution algorithms to other bi-level optimization frameworks that are related to the Donor-Recipient Bi-level Knapsack Problem (DR-BKP), *viz.*,
(1) DR-BKP with lower-level problem having multiple non-healthcare projects that cannot be picked fractionally,
(2) DR-BKP with lower-level objective having a piece-wise linear concave function, and
(3) DR-BKP with multiple lower-level problems *i.e.* recipient countries.

5.1 DR-BKP with lower-level problem having multiple non-healthcare projects

The DR-BKP introduced in chapter 1 and considered for developing solution techniques in chapter 3 and chapter 4 has a framework such that the lower-level problem (the recipient country) has an outside option of projects along with the healthcare projects that compete for funding. This outside option of projects are the non-healthcare projects that are of interest only to the recipient country;

these are projects like education, welfare, and defense and these are referred to as the "external project". Along with the healthcare projects, the recipient has to fund the external project fractionally or wholly. The external project has a linear profit and linear cost of v_0 and c_0 respectively.

We consider the original set of one or more non-healthcare projects in the lower-level problem instead of a single representative external project in order to take a realistic approach to the current problem formulation. The non-healthcare projects are eligible to be selected wholly by the recipient country for funding along with the healthcare projects in the bi-level model given in this section. This model is referred to as the Donor-Recipient Bi-level Knapsack Problem with Non-Healthcare projects (DR-BKP-NH) henceforth.

5.1.1 Problem definition

An instance of the DR-BKP-NH comprises two players, a donor and a recipient country. The donor is leader and the recipient is follower in the bi-level problem framework. There is a set I of n healthcare projects, $I = \{1, \ldots, n\}$, which are of interest to both players. Also, there is a set J of m non-healthcare projects, $J = \{1 \ldots m\}$, which are of interest to the recipient country (lower-level player) only. Each project $i \in I$, has a profit of $w_i \in \mathbb{N}$ (resp. $v_i \in \mathbb{N}$) for the donor (resp. recipient), and a cost $c_i \in \mathbb{N}$. Let $\mathbf{w}$ and $\mathbf{v}$ denote the vectors of profits of the donor and the recipient respectively and $\mathbf{c}$ be the vector of costs of the healthcare projects. Each project $j \in J$, has a profit of $v_{0j} \in \mathbb{N}$ for the recipient, and a cost $c_{0j} \in \mathbb{N}$. Let $\mathbf{v_0}$ and $\mathbf{c_0}$ denote the vectors of profits and the costs of the non-healthcare projects. We have two integer budgets, B_d and B_r, corresponding to the donor and the recipient.

To solve an instance of the DR-BKP-NH, the donor solves upper-level knapsack problem to allocate subsidies y_i to the healthcare projects $i \in I$ with a profit w_i and cost c_i. Here, a subsidy y_i is the proportion of cost of project i. Using the allocated subsidies y_i for each of the project i, the recipient solves its own

knapsack problem to allocate funds amongst the healthcare and non-healthcare projects. Each item of this knapsack corresponds to (1) a healthcare project $i \in I$ with a profit v_i and cost $c_i - c_i y_i$ after considering the donor subsidies, and (2) a non-healthcare project $j \in J$ with a profit v_{0j} and cost c_{0j}. Both healthcare and non-healthcare projects are binary and cannot be fractionally picked.

The donor aims to maximize its profit such that budget B_d is not exceeded and the projects selected are in the optimal solution set of the recipient's cost subsidized knapsack problem. Let $\mathbf{y}$ denote the vector of subsidy. Since the donor cannot subsidize a project more than its cost and the total subsidy cannot exceed its budget, the set of all valid subsidies is denoted by $Y := \Big\{\mathbf{y} : \sum_{i\in I} c_i y_i \leq B_d, \mathbf{y} \in [0,1]^n\Big\}$.

For the decision vectors to represent project selection, let $\mathbf{x}$ denote 0-1 vector representing set of healthcare projects that are picked (i^{th} component of the vector, x_i, is 1 if project i is picked and 0 otherwise) and let $\mathbf{x_0}$ denote 0-1 vector representing set of non-healthcare projects that are picked (j^{th} component of the vector, x_{0j}, is 1 if project j is picked and 0 otherwise). We define the set $X := \Big\{(\mathbf{x}, \mathbf{x_0}) : \mathbf{x} \in \{0,1\}^n, \mathbf{x_0} \in \{0,1\}^m\Big\}$.

The DR-BKP-NH has been given in (5.1) and (5.2) where upper-level is the donor problem `(DONOR)` and lower-level is the recipient problem `(RECIPIENT(y))` parameterised on the upper-level decision $\mathbf{y}$.

Problem `DONOR`:

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{5.1a}$$

$$\text{subject to} \quad \mathbf{y} \in Y \tag{5.1b}$$

$$\mathbf{x} \in \ \arg\max(\texttt{RECIPIENT(y)}). \tag{5.1c}$$

Problem `RECIPIENT(y)`:

$$\text{maximize} \quad \mathbf{v}^T\mathbf{x} + \mathbf{v_0}^T\mathbf{x_0} \tag{5.2a}$$

$$\text{subject to} \quad \sum_{i\in I}(c_i - c_i y_i)x_i + \sum_{j\in J} c_{0j}x_{0j} \leq B_r \tag{5.2b}$$

$$(\mathbf{x}, \mathbf{x_0}) \in X. \tag{5.2c}$$

The DR-BKP-NH is continuous in the upper-level and discrete in the lower-level. The enumeration algorithm given for the DR-BKP in chapter 3 can be extended for this problem setup. Let Relaxed Donor-Recipient Bi-level Knapsack Problem with Non-Healthcare projects (R-DR-BKP-NH) denote the High Point Relaxation High Point Relaxation (HPR) given in Equation 5.3.
Problem R-DR-BKP-NH:

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{5.3a}$$

$$\text{subject to} \quad \mathbf{c}^T\mathbf{y} \leq B_d \tag{5.3b}$$

$$\mathbf{c}^T\mathbf{x} + \mathbf{c_0}^T\mathbf{x_0} \leq B_r + \mathbf{c}^T\mathbf{y} \tag{5.3c}$$

$$y_i \leq x_i \qquad \forall i \in I \tag{5.3d}$$

$$\mathbf{y} \in [0,1]^n \tag{5.3e}$$

$$\mathbf{x} \in \{0,1\}^n \tag{5.3f}$$

$$\mathbf{x_0} \in \{0,1\}^m. \tag{5.3g}$$

To ensure that an optimal solution $(\mathbf{x}^*, \mathbf{x_0^*}, \mathbf{y}^*)$ to the R-DR-BKP-NH but $(\mathbf{x}^*, \mathbf{x_0^*}) \notin P(\mathbf{y}^*)$, following inequality will remove $(\mathbf{x}^*, \mathbf{x_0^*}, \mathbf{y}^*)$ from the search space for any $(\bar{\mathbf{x}}, \bar{\mathbf{x}}_\mathbf{0}) \in P(\mathbf{y}^*)$. Big M can be used to handle "if-then" constraint in order to refrain from cutting off valid subsidies.

$$\mathbf{v}^T\mathbf{x} + \mathbf{v_0}^T\mathbf{x_0} \geq \mathbf{v}^T\bar{\mathbf{x}} + \mathbf{v_0}^T\bar{\mathbf{x}}_\mathbf{0} \tag{5.4}$$

where $P(\mathbf{y}^*)$ is the follower's rational reaction set for a fixed $\mathbf{y}^* \in Y$:

$$P(\mathbf{y}^*) = \Big\{(\mathbf{x}, \mathbf{x_0}) : (\mathbf{x}, \mathbf{x_0}) \in \arg\max \Big\{\mathbf{v}^T\mathbf{x} + \mathbf{v_0}^T\mathbf{x_0} : \sum_{i\in I} c_i x_i + \sum_{j\in J} c_j x_j \leq B_r + \sum_{i\in I} c_i {y^*}_i x_i, (\mathbf{x}, \mathbf{x_0}) \in X\Big\}\Big\}. \tag{5.5}$$

The only difference between the DR-BKP and the DR-BKP-NH is that the DR-BKP has a continuous variable, x_0, in the lower-level problem whereas DR-BKP-NH has a set of discrete variables in the lower-level problem. This continuous variable is one of the difficulties to directly use the inequality 3.12 to eliminate $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ when $(\mathbf{x}^*, x_0^*) \notin P(\mathbf{y}^*)$ in the solution algorithm. To handle the issue of evaluating a range of continuous x_0 values, assumption (3.10) *i.e.* $c_0 \geq B_r$ has been used. In the case of DR-BKP-NH, since all variables in the lower-level problem are discrete, these can be potentially enumerated before adding inequality 5.4 when $(\mathbf{x}^*, \mathbf{x_0^*}) \notin P(\mathbf{y}^*)$ and hence it will be a straightforward process.

The challenge in this approach may arise when the problem size increases. The solution time to handle several binary variables will increase immensely as the number of both healthcare and non-healthcare projects increases. Taking a heuristic approach like the given genetic algorithm in chapter 4 can reduce the solution time in such cases. The upper-level problem will be solved using the genetic algorithm, and the lower-level problem will be solved using an off-the-shelf exact solver. However with the increasing size of a problem, due to the binary variables of both healthcare and non-healthcare projects, the lower-level problems may get computationally expensive. It will be interesting to compare the performance of both bi-level exact solvers and the genetic algorithm when the problem size increases.

5.2 DR-BKP with lower-level objective having a piece-wise linear concave function

The DR-BKP has a set of both healthcare and non-healthcare projects. The non-healthcare projects are of interest only to the recipient country. These have been represented by a single external project with linear profit and cost functions in the lower-level problem in the previous chapters. However in realistic cases, as the proportion of funds allocated by the recipient country to a representative external project increases, its profit achieved is not linear and tends to decrease. This decrease in the lower-level profit is approximated by a piece-wise linear concave function in the model that we present in this section. We refer to this model as Donor-Recipient Bi-level Knapsack Problem with Recipient profit as a Piece-wise linear Concave function (DR-BKP-RPC) henceforth.

5.2.1 Problem definition

An instance of DR-BKP-RPC comprises two players, a donor, and a recipient country. The donor is leader and the recipient is follower in the bi-level problem framework. There is a set I of n healthcare projects, $I = \{1, \ldots, n\}$, which are of interest to both players. Each project $i \in I$, has a profit of $w_i \in \mathbb{N}$ (resp. $v_i \in \mathbb{N}$) for the donor (resp. recipient), and a cost $c_i \in \mathbb{N}$. Let $\mathbf{w}$ and $\mathbf{v}$ denote the vectors of profits of the donor and the recipient respectively and $\mathbf{c}$ be the vector of costs of the healthcare projects. We have two integer budgets, B_d and B_r, corresponding to the donor and the recipient.

The recipient has to allocate its budget to the external project that has a linear cost c_0. This funding can be fractionally or wholly done. The profit of the external project is given by function Equation 5.6:

$$f(x_0) = \max_{(j=1\ldots m)} (a_j + u_j x_0). \tag{5.6}$$

This function is divided into intervals (also called pieces of the function) of

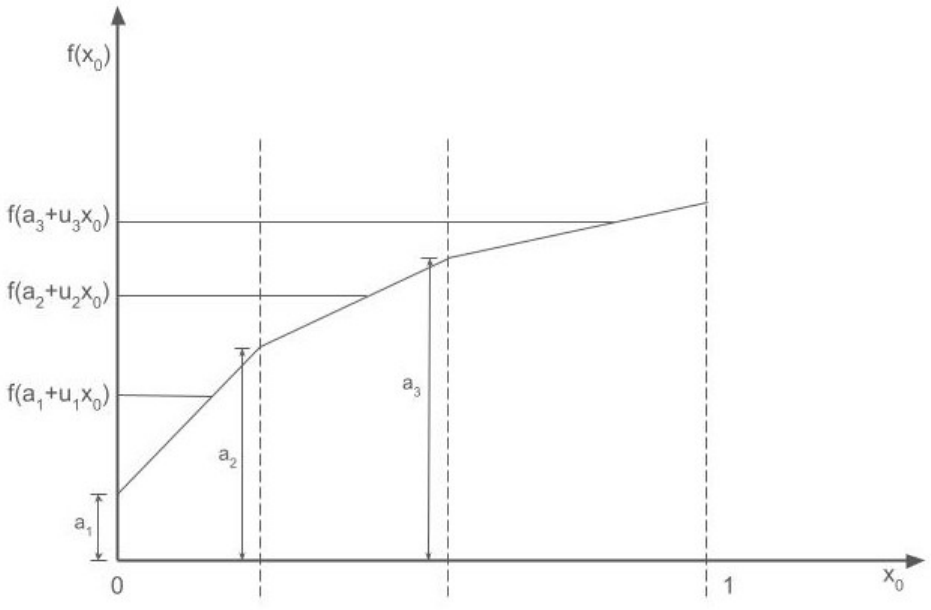


Figure 5.1: Piece-wise linear concave function for the external project of the lower-level problem

the decision variable x_0, also shown in Figure 5.1 that is non-negative and has an upper bound 1 since it represents the proportion of cost of the external project that gets funded by the recipient country. The slope of the lower-level profit function decreases as x_0 increases, given by slopes u_j, $\forall j$, and slope intercepts a_j, $\forall j$, for pieces $j = 1, \ldots, m$ where m is the total number of profit function intervals. In the current setup, we consider only three pieces of the piece-wise linear concave function, as also represented in Figure 5.1. Let $\mathbf{u}$ and $\mathbf{a}$ denote the vectors of slopes and slope intercepts of the piece-wise linear concave function respectively.

An instance of DR-BKP-RPC is specified by the input $(\mathbf{w}, \mathbf{v}, \mathbf{c}, \mathbf{u}, c_0, B_d, B_r)$. The recipient solves a knapsack problem, where each item of the knapsack corresponds to a project $i \in I$ with a profit function $f(x_0)$ given in Equation 5.6 and cost $c_i - c_i y_i$, where y_i is the proportion of cost of project i that is subsidized by the donor. These projects are binary and cannot be fractionally picked.

A solution to an instance of DR-BKP-RPC is to decide on the proportion of cost to be subsidized, y_i, for each project $i \in I$ with $\sum_{i\in I} c_i y_i \leq B_d$ such that profit of the donor is maximized given the projects are in the optimal solution set of the recipient's cost subsidized knapsack problem. We use the notation $\mathbf{y}$ to denote a vector of subsidy. The leader cannot subsidize a project more than its cost and the total subsidy cannot exceed the leader's budget. The set of all valid subsidies is denoted by $Y := \left\{\mathbf{y} : \sum_{i\in I} c_i y_i \leq B_d, \mathbf{y} \in [0,1]^n\right\}$.

We let $\mathbf{x}$ to denote a 0-1 vector representing the set of projects that are picked (i^{th} component of the vector, x_i, is 1 if project i is picked and 0 otherwise). We define the set $X := \left\{(\mathbf{x}, x_0) : \mathbf{x} \in \{0,1\}^n, x_0 \in [0,1]\right\}$. Let $\mathcal{X} := \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^K\}$ be the set of all possible subsets of projects. We define the set of all valid projects corresponding to a subsidy $\mathbf{y} \in Y$ as

$$\mathcal{G}(\mathbf{y}) := \{\mathbf{x} \in \mathcal{X} : \sum_{i\in I}(c_i - c_i y_i)x_i \leq B_r\}. \tag{5.7}$$

The DR-BKP-RPC has been given in (5.8) and (5.9) where upper-level is the donor problem `(DONOR)` and lower-level is the recipient problem `(RECIPIENT(y))` parameterised on the upper-level decision $\mathbf{y}$.

Problem `DONOR`:

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{5.8a}$$

$$\text{subject to} \quad \mathbf{y} \in Y \tag{5.8b}$$

$$\mathbf{x} \in \arg\max(\texttt{RECIPIENT(}\mathbf{y}\texttt{)}). \tag{5.8c}$$

Problem `RECIPIENT(y)`:

$$\text{maximize} \quad \mathbf{v}^T\mathbf{x} + f(x_0) \tag{5.9a}$$

$$\text{subject to} \quad \sum_{i\in I}(c_i - c_i y_i)x_i + c_0 x_0 \leq B_r \tag{5.9b}$$

$$(\mathbf{x}, x_0) \in X. \tag{5.9c}$$

The DR-BKP-RPC is continuous in the upper-level and both discrete and continuous in the lower-level. The HPR of DR-BKP-RPC or the Relaxed Donor-Recipient Bi-level Knapsack Problem with Recipient profit as a Piece-wise linear Concave function (DR-BKP-RPC) will be the same as the Relaxed Donor-Recipient Bi-level Knapsack Problem (R-DR-BKP) (see Equation 4.1 in chapter 3). However, to use the enumeration algorithm for this setup, the inequality to cut off bi-level infeasible solutions will be different.

To ensure that an optimal solution $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ to the DR-BKP-RPC but $(\mathbf{x}^*, x_0^*) \notin P(\mathbf{y}^*)$, following inequality will remove $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ from the search space for any $(\bar{\mathbf{x}}, \bar{x}_0) \in P(\mathbf{y}^*)$.

$$\mathbf{v}^T\mathbf{x} + f(x_0) \geq \mathbf{v}^T\bar{\mathbf{x}} + f(\bar{x}_0) \tag{5.10}$$

where $P(\mathbf{y}^*)$ is the follower's rational reaction set for a fixed

$$P(\mathbf{y}^*) = \Big\{(\mathbf{x}, x_0) : (\mathbf{x}, x_0) \in \arg\max \Big\{\mathbf{v}^T x + f(x_0) : \sum_{i\in I} c_i x_i + c_0 x_0 \leq B_r + \sum_{i\in I} c_i {y^*}_i x_i, (\mathbf{x}, x_0) \in X\Big\}\Big\}. \tag{5.11}$$

Since it is known that in order to use the inequality 3.12 to eliminate $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ when $(\mathbf{x}^*, x_0^*) \notin P(\mathbf{y}^*)$ in the solution algorithm,

It is known now that the continuous variable x_0 is one of the difficulties to

directly use the inequality 3.12 to eliminate $(\mathbf{x}^*, x_0^*, \mathbf{y}^*)$ when $(\mathbf{x}^*, x_0^*) \notin P(\mathbf{y}^*)$ in the solution algorithm of DR-BKP. Using assumption (3.10) *i.e.* $c_0 \geq B_r$, inequality 3.15 has been used to address this issue. In every iteration of the enumeration algorithm, the search space is cut using this inequality if bi-level optimal solution is not reached. However, the challenge here is to find the exact piece/ interval of the piece-wise linear concave function where the bi-level optimal solution will lie. Depending on the piece where the bi-level optimal solution lies, the profit function u_j changes and hence the right hand side of the inequality 5.10. In case all the pieces are enumerated on the right hand side of the inequality using the profit to cost ratio of each piece, there will be many branches created in just one iteration of the search process. This can result in intractability over just a few iterations.

5.3 DR-BKP with multiple lower-level problems

The model framework considered in this thesis has a single leader and single follower *i.e.* the DR-BKP has a single donor in the system that allocates funds to healthcare projects in a single recipient country. The decision made by the donor (example: a subsidy is increased for either of the projects) can make an impact in the decision made by the recipient (example: the recipient can select this project after it receives more subsidy if it was not selected before).

However, in realistic cases, the donor has multiple recipient countries in its consideration for allocating the funds for healthcare projects. In this section, we introduce a model framework of single leader-multiple followers to address this realistic generalization of the DR-BKP *i.e.* the DR-BKP with multiple lower-level problems. We refer to this model framework as Donor-multiple Recipients Bi-level Knapsack problem (D-mR-BKP) henceforth. This problem is a special case of the general framework 'Single-Leader Multi-Follower Games' given by Aussel & Svensson (2020).

5.3.1 Problem definition

In order to keep the problem definition simpler, we consider only two recipient countries *i.e.* Donor-two Recipients Bi-level Knapsack problem (D-2R-BKP). An instance of D-2R-BKP has a single donor who is the leader and two recipient countries who are the followers. There is a set I of n healthcare projects, $I = \{1, \ldots, n\}$, that is of interest to recipient 1 and there is a set J of m healthcare projects, $J = \{1, \ldots, m\}$, that is of interest to recipient 2. The donor is interested in all the healthcare projects across both recipient countries to maximize its profit. Each project $i \in I$, has a profit of $w1_i \in \mathbb{N}$ (resp. $v1_i \in \mathbb{N}$) for the donor (resp. recipient 1), and a cost $c1_i \in \mathbb{N}$. Let $\mathbf{w1}$ and $\mathbf{v1}$ denote the vectors of profits of the donor and the recipient 1 respectively and $\mathbf{c1}$ be the vector of costs of the healthcare projects. Similarly, each project $j \in J$, has a profit of $w_j \in \mathbb{N}$ (resp. $v_j \in \mathbb{N}$) for the donor (resp. recipient 2), and a cost $c2_i \in \mathbb{N}$. Let $\mathbf{w2}$ and $\mathbf{v2}$ denote the vectors of profits of the donor and the recipient 2 respectively and $\mathbf{c2}$ be the vector of costs of the healthcare projects.

We have integer budgets, B_d, $B1_r$ and $B2_r$, corresponding to the donor, recipient 1 and recipient 2. Both recipients also allocate their respective budgets to their own external projects. An external project of each recipient is a representative project of all other non-healthcare projects wherein the donor is not interested. These external projects have linear profits, $v1_0$ and $v2_0$, and linear costs, $c1_0$ and $c2_0$, for recipient 1 and recipient 2 respectively.

An instance of D-2R-BKP is specified by the input $(\mathbf{w1}, \mathbf{w2}, \mathbf{v1}, \mathbf{v2}, \mathbf{c1}, \mathbf{c2}, v1_0, v2_0, c1_0, c2_0, B_d, B1_r, B2_r)$. Recipient 1 solves its own knapsack problem, where each item of the knapsack corresponds to a project $i \in I$ with a profit $v1_i$ and cost $c1_i - c1_i y1_i$, where $y1_i$ is the proportion of cost of project i that is subsidized by the donor. Similarly, recipient 2 solves its own knapsack problem, where each item of the knapsack corresponds to a project $j \in J$ with a profit $v2_j$ and cost $c2_j - c2_j y2_j$, where $y2_j$ is the proportion of cost of project j that is subsidized by the donor. All the healthcare projects are binary and cannot be

fractionally picked. Along with the healthcare projects, the recipients have to fund their respective external projects which can be done fractionally or wholly *i.e.* only a proportion or entire of cost of the external projects can be funded by the recipients.

A solution to an instance of D-2R-BKP is to decide on the proportion of cost to be subsidized, $y1_i$, for each project $i \in I$ and $y2_j$, for each project $j \in J$ with $\sum_{i\in I} c1_i y1_i + \sum_{j\in J} c2_j y2_j \leq B_d$ such that profit of the donor is maximized given the projects are in the optimal solution set of the recipients' cost subsidized knapsack problems. We use the notation $\mathbf{y1}$ and $\mathbf{y2}$ to denote the vectors of subsidy for recipient 1 and recipient 2 respectively. The leader cannot subsidize a project more than its cost and the total subsidy cannot exceed the leader's budget. The set of all valid subsidies is denoted by
$Y := \Big\{(\mathbf{y1}, \mathbf{y2}) : \sum_{i\in I} c1_i y1_i + \sum_{j\in J} c2_j y2_j \leq B_d, \mathbf{y1} \in [0,1]^n, \mathbf{y2} \in [0,1]^m\Big\}$.

We let $\mathbf{x1}$ denote a 0-1 vector representing the set of projects that are picked (i^{th} component of the vector, $x1_i$, is 1 if project i is picked and 0 otherwise) and $x1_0$ to denote proportion of cost of external project that is being funded by the recipient 1. We define the set $X1 := \Big\{(\mathbf{x1}, x1_0) : \mathbf{x1} \in \{0,1\}^n, x1_0 \in [0,1]\Big\}$. Similarly, let $\mathbf{x2}$ denote a 0-1 vector representing the set of projects that are picked (j^{th} component of the vector, $x2_j$, is 1 if project j is picked and 0 otherwise) and $x2_0$ to denote proportion of cost of external project that is being funded by the recipient 2. We define the set $X2 := \Big\{(\mathbf{x2}, x2_0) : \mathbf{x2} \in \{0,1\}^m, x2_0 \in [0,1]\Big\}$.

The D-2R-BKP has been given in (5.12), (5.13) and (5.14) where upper-level is the donor problem `(DONOR)` and there are two problems in the lower-level - (1) recipient 1 problem `(RECIPIENT1(y1))` parameterised on the upper-level decision $\mathbf{y1}$ and (2) recipient 2 problem `(RECIPIENT2(y2))` parameterised on the upper-level decision $\mathbf{y2}$.

Problem `DONOR`:

$$\begin{aligned} \text{maximize} \quad & \mathbf{w1}^T\mathbf{x1} + \mathbf{w2}^T\mathbf{x2} && (5.12a)\\ \text{subject to} \quad & (\mathbf{y1}, \mathbf{y2}) \in Y && (5.12b)\\ & \mathbf{x1} \in \arg\max(\texttt{RECIPIENT1(}\mathbf{y1}\texttt{)}) && (5.12c)\\ & \mathbf{x2} \in \arg\max(\texttt{RECIPIENT2(}\mathbf{y2}\texttt{)}). && (5.12d) \end{aligned}$$

Problem `RECIPIENT1(`$\mathbf{y1}$`)`:

$$\begin{aligned} \text{maximize} \quad & \mathbf{v1}^T\mathbf{x1} + v1_0x1_0 && (5.13a)\\ \text{subject to} \quad & \sum_{i\in I}(c1_i - c1_iy1_i)x1_i + c1_0x1_0 \leq B1_r && (5.13b)\\ & (\mathbf{x1}, x1_0) \in X1. && (5.13c) \end{aligned}$$

Problem `RECIPIENT2(`$\mathbf{y2}$`)`:

$$\begin{aligned} \text{maximize} \quad & \mathbf{v2}^T\mathbf{x2} + v2_0x2_0 && (5.14a)\\ \text{subject to} \quad & \sum_{j\in J}(c2_j - c2_jy2_j)x2_j + c2_0x2_0 \leq B2_r && (5.14b)\\ & (\mathbf{x2}, x2_0) \in X2. && (5.14c) \end{aligned}$$

The D-2R-BKP is continuous in the upper-level and both continuous and discrete in the lower-level. We first define the HPR of the D-2R-BKP *i.e.* Relaxed Donor-two Recipients Bi-level Knapsack problem (R-D-2R-BKP) as follows.

Problem R-D-2R-BKP:

$$\text{maximize} \quad \mathbf{w1}^T\mathbf{x1} + \mathbf{w2}^T\mathbf{x2} \tag{5.15a}$$

$$\text{subject to} \quad \mathbf{c}^T\mathbf{y} \leq B_d \tag{5.15b}$$

$$\mathbf{c1}^T\mathbf{x1} + c1_0x1_0 \leq B1_r + \mathbf{c1}^T\mathbf{y1} \tag{5.15c}$$

$$\mathbf{c2}^T\mathbf{x2} + c2_0x2_0 \leq B2_r + \mathbf{c2}^T\mathbf{y2} \tag{5.15d}$$

$$y1_i \leq x1_i \qquad \forall i \in I \tag{5.15e}$$

$$y2_j \leq x2_j \qquad \forall j \in J \tag{5.15f}$$

$$\mathbf{y1} \in [0,1]^n \tag{5.15g}$$

$$\mathbf{y1} \in [0,1]^m \tag{5.15h}$$

$$\mathbf{x1} \in \{0,1\}^n \tag{5.15i}$$

$$\mathbf{x2} \in \{0,1\}^m \tag{5.15j}$$

$$x1_0 \in [0,1] \tag{5.15k}$$

$$x2_0 \in [0,1]. \tag{5.15l}$$

To extend the enumeration algorithm given for the DR-BKP in chapter 3 in this problem setup, first inequalities need to be identified which can be used to cut off the search space that have bi-level infeasible solutions. To ensure that an optimal solution $(\mathbf{x1}^*, \mathbf{x2}^*, \mathbf{x1_0}^*, \mathbf{x2_0}^*, \mathbf{y1}^*, \mathbf{y2}^*)$ to the R-D-2R-BKP but $(\mathbf{x1}^*, \mathbf{x1_0}^*) \notin P1(\mathbf{y1}^*)$ and $(\mathbf{x2}^*, \mathbf{x2_0}^*) \notin P2(\mathbf{y2}^*)$, following inequalities will remove $(\mathbf{x1}^*, \mathbf{x2}^*, \mathbf{x1_0}^*, \mathbf{x2_0}^*, \mathbf{y1}^*, \mathbf{y2}^*)$ from the search space for any $(\bar{\mathbf{x1}}, \bar{\mathbf{x1_0}}) \in P1(\mathbf{y1}^*)$ and $(\bar{\mathbf{x2}}, \bar{\mathbf{x2_0}}) \in P2(\mathbf{y2}^*)$ after assumption 3.10 in chapter 3.

$$\mathbf{v1}^T\mathbf{x1} + v1_0x1_0 - \sum_{i \in I} c1'_i\bar{x1}_iy1_i \geq \mathbf{v1}^T\bar{\mathbf{x1}} + \frac{v1_0}{c1_0}(B1_r - \mathbf{c1}^T\bar{\mathbf{x1}}), \tag{5.16}$$

$$\mathbf{v2}^T\mathbf{x2} + v2_0x2_0 - \sum_{j\in J} c2'_i\bar{x2}_iy2_i \geq \mathbf{v2}^T\bar{\mathbf{x2}} + \frac{v2_0}{c2_0}(B2_r - \mathbf{c2}^T\bar{\mathbf{x2}}). \tag{5.17}$$

where $P1(\mathbf{y1}^*)$ is the recipient 1's rational reaction set for a fixed $\mathbf{y1}^* \in Y$:

$$P1(\mathbf{y1}^*) = \Big\{(\mathbf{x1}, x1_0) : (\mathbf{x1}, x1_0) \in \arg\max \Big\{\mathbf{v1}^T\mathbf{x1} + v1_0{}^Tx1_0 : \sum_{i\in I} c1_ix1_i + c1_0x1_0 \leq B1_r + \sum_{i\in I} c1_iy1^*{}_ix1_i, (\mathbf{x1}, x1_0) \in X1\Big\}\Big\}, \tag{5.18}$$

and $P2(\mathbf{y2}^*)$ is the recipient 2's rational reaction set for a fixed $\mathbf{y2}^* \in Y$:

$$P2(\mathbf{y2}^*) = \Big\{(\mathbf{x2}, x2_0) : (\mathbf{x2}, x2_0) \in \arg\max \Big\{\mathbf{v2}^T\mathbf{x2} + v2_0{}^Tx2_0 : \sum_{j\in J} c2_jx2_j + c2_0x2_0 \leq B2_r + \sum_{j\in J} c2_jy1^*{}_jx2_j, (\mathbf{x2}, x2_0) \in X2\Big\}\Big\}. \tag{5.19}$$

The solution method will start by solving the R-D-2R-BKP first. In every iteration, cuts will be added when either

1. $(\mathbf{x1}^*, \mathbf{x1_0^*}) \notin P1(\mathbf{y1}^*)$, inequality 5.16 will be added if $\mathbf{c1}^T\bar{\mathbf{x1}} - \sum_{i\in I} c1_i\bar{x1}_iy1_i \leq B1_r$ or
2. $(\mathbf{x2}^*, \mathbf{x2_0^*}) \notin P2(\mathbf{y2}^*)$, inequality 5.17 will be added if $\mathbf{c2}^T\bar{\mathbf{x2}} - \sum_{j\in J} c2_i\bar{x2}_iy2_i \leq B2_r$

In order to handle the "if-then" constraints, there will be a need to use more than two big Ms. This might affect the solution time of the enumeration algorithm. An interesting direction will be to use the branching algorithm wherein the cuts can be systematically added at every incumbent.

The genetic algorithm can be easily extended to a DR-BKP with multiple lower-level problems like the D-2R-BKP to find good quality solution in reasonable time.. The upper-level will be solved by the genetic algorithm. For every

upper-level feasible solution, each of the lower-level problems will be solved using an off-the-shelf exact solver. However, both solution quality and time will be significantly impacted by the size of the problem.

Chapter 6

Conclusion and future research

In this thesis, we have studied a specific application of Bi-level Knapsack Problems (BKPs) in a real-world problem of allocation of healthcare funds and developed algorithms to solve different instances that range on differing parameters. We start with an introduction to a general Bi-level Programming Problem Bi-level Programming Problem (BLPP) and then discuss about the mixed-integer BLPPs. Following this, we give the real-world problem which has played a key role in motivating this thesis. The application problem is about allocating the healthcare funds to healthcare projects by two participants in the system, a donor agency and a recipient country, as introduced by Morton et al. (2018). They solved this problem using an assumption that the recipient country is a middle-income country capable of funding all the healthcare projects from its own budget. In order to generalize the model and get solutions for realistic cases, this assumption needs to be relaxed. However, the resulting BKP, referred to as Donor-Recipient Bi-level Knapsack Problem (DR-BKP) in this thesis, is a very difficult but interesting problem to solve.

After studying different types of mixed-integer BLPPs and BKPs in the literature, as presented in chapter 2, we realized none of the solution methods in literature can be directly used to solve the DR-BKP. In chapter 3, we define the DR-BKP along with its properties and assumptions. The complexity of the DR-BKP

is then discussed while giving evidence to this problem being $\sum_2^p$-hard. We have then proposed two exact solution techniques for solving the DR-BKP. First, a relaxed DR-BKP is presented along with a rationale for developing and using cuts to exclude bi-level infeasible solutions during the search process. The developed cuts are used to give a reformulation of the DR-BKP. A finitely-terminating enumeration algorithm has been presented then that uses this reformulated DR-BKP to find the bi-level optimal solution. Using the same reformulation, a branching algorithm has been proposed that makes use of the branching functionality in the Mixed-Integer Programming Problem (MILP) solver called CPLEX. We have presented a computational study following this, where in first a range of data have been generated based on available real data of the application problem. The performance of both enumeration and branching algorithms has been tested by solving all instances in the generated data sets. The obtained results show that the enumeration algorithm performs better in case the problem instances are less complicated. As the complexity of the instances increases, like when the external project starts competing with healthcare projects in the lower-level problem and/or when there is divergence in the healthcare project valuation by the two participants, the branching algorithm performs better. We also encountered two data sets so complex that the exact solvers could not find reasonable solutions in the given time limit. This motivated us to develop a heuristic approach for solving the DR-BKP.

A genetic algorithm has been developed to solve the DR-BKP and presented in chapter 4. The developed algorithm is a nested sequential approach to handle the two levels of the DR-BKP. The upper-level decision variables are coded as a genome used in the evolution process of the genetic algorithm for each population. These genomes are generated randomly to get a solution of the upper-level problem. Each upper-level solution that is within the upper-level budget is then sent to the lower-level problem for evaluation. The parameterized lower-level knapsack problem can now be solved using any heuristic or exact solvers. We have used two

methods, a greedy heuristic and an available exact solver to solve the lower-level problem for each upper-level solution. The obtained lower-level solution is a set of project allocations which are then plugged back into the upper-level objective to find the upper-level profit. This value of the upper-level objective gives us the fitness of the generated solution. The population of upper-level solutions evolves over each generation in the genetic algorithm. After a given number of generations, the algorithm terminates and gives the best solution obtained so far. The performance of the genetic algorithm is pronounced in complex data sets that have divergence in the donor and country valuations of the healthcare projects, however, it finds difficult to handle instances that have profit to cost ratio of external project higher than the average profit to cost ratio of healthcare projects.

In chapter 5, we have given extensions to the DR-BKP. The DR-BKP has a single external project as a representative project for the non-healthcare projects in the lower-level problem. We have given a model formulation in section 5.1 that considers multiple non-healthcare projects in the lower-level for selection by the recipient country in its knapsack problem along with the healthcare projects. The healthcare projects are eligible to be subsidized partially or wholly by the donor and selected for funding by the recipient after receiving the donor subsidies. However, the non-healthcare projects can be selected wholly for funding by the recipient only. In section 5.2, we relax the assumption made in the DR-BKP that the external project has linear cost and linear profit. As in realistic cases, the profit of the external project decreases as the funding for this project increases. This relation is captured using a piece-wise linear concave function for the lower-level profit and a model formulation has been given such that the lower-level problem has this updated profit function. We also have considered another realistic situation where there is a single donor and multiple followers *i.e.* there are more than one recipient countries in the system. The donor considers all the healthcare projects in these countries for its funds' allocation. A model

formulation for this scenario has been given in section 5.3.

From this study, we have looked into different methods to solve the DR-BKP such that various data sets can be handled and solved effectively. Nevertheless, there are some directions in which this research work can be taken further to solve the DR-BKP and its variations more effectively.

The cost of the external project in the lower-level problem is assumed to be at least equal to the lower-level budget in the solution methods developed in this thesis. It would be interesting to explore how the developed algorithms can be extended to operate without relying on the assumption (see 3.10 in chapter 3). Firstly, the problem needs to be investigated to check if an optimal solution exists *i.e.* if the problem has a maximum. After this primary study, when assumption (3.10) is relaxed, the right hand side of the introduced cut will not remain linear anymore. This will need another binary variable to linearise these terms and hence impact the efficiency of the search algorithm.

The proof of complexity that we give does not provide evidence that a polynomial solution is not possible for unary encoding. However, the existence of a pseudo-polynomial algorithm for this problem would be beneficial in practical scenarios. We have only provided evidence for Σ_p^2-hardness in chapter 3. A direct reduction from a Σ_p^2-complete problem to DR-BKP is open.

For the genetic algorithm developed and presented in chapter 5, there are different directions in which it can be improved to get good solutions fast. The initial population that we generate is set to the upper-level decision variables that yield a lower bound of the instance. In our computational experiments, we realized other heuristic methods can be used to generate an initial population like a greedy heuristic that prioritizes projects valued more by both upper-level and lower-level participants based on pre-defined weights. Although the best bound has been improved by the genetic algorithm for the most difficult data sets, *viz.* 14 and 15, we would be interested in obtaining bi-level optimal solutions in a reasonable time. The genetic algorithm approach can be used along with the

exact solvers to generate cuts at a faster rate to reduce the feasible solution space during the exact solution procedures. As the size of the instances increases *i.e.* the number of healthcare projects increases, all the solution algorithms tend to take longer time to solve. Hence further research can be done to modify the evaluation function to address this issue, like a machine learning algorithm to approximate the upper-level objective for a candidate solution or perturbation in the candidate solution. There are few attempts in the literature that term this approach as meta-modeling (Angelo et al. 2014, Sinha et al. 2014, 2016).

Bibliography

Aiyoshi, E. & Shimizu, K. (1981), 'Hierarchical decentralized systems and its new solution by a barrier method.', *IEEE Transactions on Systems, Man and Cybernetics* (6), 444–449.

Aiyoshi, E. & Shimizu, K. (1984), 'A solution method for the static constrained stackelberg problem via penalty method', *IEEE Transactions on Automatic Control* **29**(12), 1111–1114.

Alekseeva, E. & Kochetov, Y. (2013), Matheuristics and exact methods for the discrete (r— p)-centroid problem, *in* 'Metaheuristics for bi-level optimization', Springer, pp. 189–219.

Andrade, R., Doostmohammadi, M., Santos, J. L., Sagot, M.-F., Mira, N. P. & Vinga, S. (2020), 'Momo-multi-objective metabolic mixed integer optimization: application to yeast strain engineering', *BMC bioinformatics* **21**, 1–13.

Angelo, J. S. & Barbosa, H. (2015), 'A study on the use of heuristics to solve a bilevel programming problem', *International Transactions in Operational Research* **22**.

Angelo, J. S., Krempser, E. & Barbosa, H. J. (2014), Differential evolution assisted by a surrogate model for bilevel programming problems, *in* '2014 IEEE Congress on Evolutionary Computation (CEC)', IEEE, pp. 1784–1791.

Arroyo, J. M. & Fernández, F. J. (2013), A genetic algorithm for power system vulnerability analysis under multiple contingencies, *in* E.-G. Talbi,

ed., 'Metaheuristics for Bi-level Optimization', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 41–68. https://link.springer.com/chapter/10.1007/978-3-642-37838-6_2.

Aussel, D. & Svensson, A. (2020), *A Short State of the Art on Multi-Leader-Follower Games*, Springer International Publishing, Cham, pp. 53–76. https://doi.org/10.1007/978-3-030-52119-6_3.

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

Bard, J. F. & Moore, J. T. (1990*a*), 'A branch and bound algorithm for the bilevel programming problem', *SIAM Journal on Scientific and Statistical Computing* **11**(2), 281–292.

Bard, J. F. & Moore, J. T. (1992), 'An algorithm for the discrete bilevel programming problem', *Naval Research Logistics (NRL)* **39**(3), 419–435.

Bard, J. & Moore, J. (1990*b*), 'A branch and bound algorithm for the bilevel programming problem', *SIAM Journal on Scientific and Statistical Computing* **11**(2), 281–292.

Bennett, K., Hu, J., Ji, X., Kunapuli, G. & Pang, J.-S. (2006), Model selection via bilevel optimization, *in* 'The 2006 IEEE International Joint Conference on Neural Network Proceedings', pp. 1922–1929.

Bialas, W. F. & Karwan, M. H. (1984), 'Two-level linear programming', *Management science* **30**(8), 1004–1020.

Biesma, R. G., Brugha, R., Harmer, A., Walsh, A., Spicer, N. & Walt, G. (2009), 'The effects of global health initiatives on country health systems: a review of the evidence from hiv/aids control', *Health policy and planning* **24**(4), 239–252.

Bracken, J. & McGill, J. T. (1973), 'Mathematical programs with optimization problems in the constraints', *Operations Research* **21**(1), 37–44.

Briest, P., Gualà, L., Hoefer, M. & Ventre, C. (2012), 'On stackelberg pricing with computationally bounded customers', *Networks* **60**(1), 31–44. https://onlinelibrary.wiley.com/doi/abs/10.1002/net.20457.

Brotcorne, L., Hanafi, S. & Mansi, R. (2009), 'A dynamic programming algorithm for the bilevel knapsack problem', *Operations Research Letters* **37**(3), 215 – 218.

Brotcorne, L., Hanafi, S. & Mansi, R. (2013), 'One-level reformulation of the bilevel knapsack problem using dynamic programming', *Discrete Optimization* **10**(1), 1 – 10.

Calvete, H. I., Galé, C. & Oliveros, M.-J. (2011), 'Bilevel model for production–distribution planning solved by using ant colony optimization', *Computers & operations research* **38**(1), 320–327.

Calvete, H. I., Galé, C. & Mateo, P. M. (2008), 'A new approach for solving linear bilevel problems using genetic algorithms', *European Journal of Operational Research* **188**(1), 14–28. https://www.sciencedirect.com/science/article/pii/S0377221707003773.

Camacho-Vallejo, J.-F., Mar-Ortiz, J., López-Ramos, F. & Rodríguez, R. P. (2015), 'A genetic algorithm for the bi-level topological design of local area networks', *PloS one* **10**(6), e0128067.

Camacho-Vallejo, J.-F., Muñoz-Sánchez, R. & González-Velarde, J. L. (2015), 'A heuristic algorithm for a supply chain s production-distribution planning', *Computers & Operations Research* **61**, 110–121.

Caprara, A., Carvalho, M., Lodi, A. & Woeginger, G. J. (2014), 'A study on the computational complexity of the bilevel knapsack problem', *SIAM Journal on Optimization* **24**(2), 823–838. https://doi.org/10.1137/130906593.

Caprara, A., Carvalho, M., Lodi, A. & Woeginger, G. J. (2016), 'Bilevel knapsack

with interdiction constraints', *INFORMS Journal on Computing* **28**(2), 319–333.

Caramia, M. & Mari, R. (2016), 'A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints', *Optimization Letters* **10**, 997–1019.

Carvalho, M. (2016), Computation of equilibria on integer programming games, PhD thesis, Universidade do Porto (Portugal).

Chaabani, A., Bechikh, S. & Said, L. B. (2017), 'A co-evolutionary decomposition-based chemical reaction algorithm for bi-level combinatorial optimization problems', *Procedia Computer Science* **112**, 780–789. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France.

Chaabani, A. & Said, L. B. (2020), 'A co-evolutionary decomposition-based algorithm for the bi-level knapsack optimisation problem', *International Journal of Computational Intelligence Studies* **9**(1-2), 52–67.

Chen, L. & Zhang, G. (2013), 'Approximation algorithms for a bi-level knapsack problem', *Theoretical Computer Science* **497**, 1–12.

Cheraghalipour, A., Paydar, M. M. & Hajiaghaei-Keshteli, M. (2019), 'Designing and solving a bi-level model for rice supply chain using the evolutionary algorithms', *Computers and Electronics in Agriculture* **162**, 651–668. `https://www.sciencedirect.com/science/article/pii/S0168169917314631`.

Chi, Y.-L., Blecher, M., Chalkidou, K., Culyer, A., Claxton, K., Edoka, I., Glassman, A., Kreif, N., Jones, I., Mirelman, A. J. et al. (2020), 'What next after gdp-based cost-effectiveness thresholds?', *Gates open research* **4**.

Cleary, S., Mooney, G. & McIntyre, D. (2010), 'Equity and efficiency in hiv-

treatment in south africa: the contribution of mathematical programming to priority setting', *Health economics* **19**(10), 1166–1180.

Colson, B., Marcotte, P. & Savard, G. (2005*a*), 'Bilevel programming: A survey', *4OR* **3**(2), 87–107.

Colson, B., Marcotte, P. & Savard, G. (2005*b*), 'A trust-region method for nonlinear bilevel programming: algorithm and computational experience', *Computational Optimization and Applications* **30**, 211–227.

Constantin, I. & Florian, M. (1995), 'Optimizing frequencies in a transit network: a nonlinear bi-level programming approach', *International Transactions in Operational Research* **2**(2), 149–164. `https://www.sciencedirect.com/science/article/pii/096960169400023M`.

Côté, J.-P., Marcotte, P. & Savard, G. (2003), 'A bilevel modelling approach to pricing and fare optimisation in the airline industry', *Journal of Revenue and Pricing Management* **2**, 23–36.

Dame, J. & Nüsser, M. (2011), 'Food security in high mountain regions: agricultural production and the impact of food subsidies in ladakh, northern india', *Food Security* **3**, 179–194.

Dantzig, G. B. (1957), 'Discrete-variable extremum problems', *Operations Research* **5**(2), 266–288. `https://doi.org/10.1287/opre.5.2.266`.

Deb, K., Sinha, A., Malo, P. & Lu, Z. (2020), Approximate Bilevel Optimization with Population-Based Evolutionary Algorithms, *in* S. Dempe & A. Zemkoho, eds, 'Bilevel Optimization', Springer Optimization and Its Applications, Springer, chapter 13, pp. 361–402.

Della Croce, F. & Scatamacchia, R. (2020), 'An exact approach for the bilevel knapsack problem with interdiction constraints and extensions', *Mathematical Programming* **183**, 249–281.

Dempe, S. (1996), *Discrete bilevel optimization problems*, Inst. für Wirtschaftsinformatik.

Dempe, S. (2001), 'Discrete bilevel optimization problems', *Technical Report D-04109, Institut fur Wirtschaftsinformatik, Universitat Leipzig, Leipzig, Germany* .

Dempe, S. & Richter, K. (2000), 'Bilevel programming with knapsack constraints', *Central European Journal of Operations Research* **8**(2), 93–107.

Dempe, S. & Zemkoho, A., eds (2020), *Bilevel Optimization*, number 978-3-030-52119-6 *in* 'Springer Optimization and Its Applications', Springer.

DeNegre, S. (2011), Interdiction and Discrete Bilevel Linear Programming, PhD thesis, USA.

DeNegre, S. T. & Ralphs, T. K. (2009), 'A branch-and-cut algorithm for integer bilevel linear programs', *Book: Operations Research and Cyber-Infrastructure* pp. 65–78.

Drake, T., Chi, Y.-L., Morton, A. & Pitt, C. (2024), 'Why cost-effectiveness thresholds for global health donors should differ from thresholds for ministries of health (and why it matters)', *F1000Research* **12**, 214.

Duro, J., Elsenbroich, C., Fergie, G., Ghatkar, S., Hoehn, A., Li, P., Meier, P., Veres, D., Winterbottom, J., Kadirkamanathan, V. et al. (2024), 'Balancing improvements in average life expectancy with inequality reductions across scotland: An inclusive economy trade-off study'. `https://eprints.gla.ac.uk/337650/`.

Ecker, J. G. & Song, J. H. (1994), 'Optimizing a linear function over an efficient set', *Journal of Optimization Theory and Applications* **83**(3), 541–563.

Edmunds, T. A. & Bard, J. F. (1991), 'Algorithms for nonlinear bilevel mathematical programs', *IEEE transactions on Systems, Man, and Cybernetics* **21**(1), 83–89.

Edmunds, T. A. & Bard, J. F. (1992), 'An algorithm for the mixed-integer nonlinear bilevel programming problem', *Annals of operations research* **34**(1), 149–162.

Fanghänel, D. & Dempe, S. (2009), 'Bilevel programming with discrete lower level problems', *Optimization* **58**(8), 1029–1047. https://doi.org/10.1080/02331930701763389.

Finkelstein, D. M., Harding, J. F., Paulsell, D., English, B., Hijjawi, G. R. & Ng'andu, J. (2022), 'Economic well-being and health: The role of income support programs in promoting health and advancing health equity: Study examines income support programs and their role in promoting health and advancing health equity.', *Health Affairs* **41**(12), 1700–1706.

Fischetti, M., Ljubić, I., Monaci, M. & Sinnl, M. (2017), 'A new general-purpose algorithm for mixed-integer bilevel linear programs', *Operations Research* **65**(6), 1615–1637.

Fischetti, M., Ljubić, I., Monaci, M. & Sinnl, M. (2019), 'Interdiction games and monotonicity, with application to knapsack problems', *INFORMS Journal on Computing* **31**(2), 390–410.

Fischetti, M., Ljubić, I., Monaci, M. & Sinnl, M. (2018), 'On the use of intersection cuts for bilevel optimization', *Mathematical Programming* **172**, 77–103.

Fischetti, M., Monaci, M. & Sinnl, M. (2018), 'A dynamic reformulation heuristic for generalized interdiction problems', *European Journal of Operational Research* **267**(1), 40–51. https://www.sciencedirect.com/science/article/pii/S0377221717310548.

Fliege, J. & Vicente, L. N. (2006), 'Multicriteria approach to bilevel optimization', *Journal of optimization theory and applications* **131**, 209–225.

Fortuny-Amat, J. & McCarl, B. (1981), 'A representation and economic interpretation of a two-level programming problem', *The Journal of the Operational Research Society* **32**(9), 783–792.

Fülöp, J. (1993), 'On the equivalence between a linear bilevel programming problem and linear optimization over the efficient set', *Techn. Rep. WP* pp. 93–1.

Furini, F., Iori, M., Martello, S. & Yagiura, M. (2015), 'Heuristic and exact algorithms for the interval min–max regret knapsack problem', *INFORMS J. Comput.* **27**, 392–405.

Glackin, J., Ecker, J. & Kupferschmid, M. (2009), 'Solving bilevel linear programs using multiple objective linear programming', *Journal of optimization theory and applications* **140**, 197–212.

Hejazi, S., Memariani, A., Jahanshahloo, G. & Sepehri, M. (2002), 'Linear bilevel programming solution by genetic algorithm', *Computers Operations Research* **29**(13), 1913–1925. `https://www.sciencedirect.com/science/article/pii/S0305054801000661`.

Heller, M. P. S. (2005), *Understanding fiscal space*, International Monetary Fund.

Holland, J. H. (1992), *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, Cambridge, MA, USA.

Ishizuka, Y. & Aiyoshi, E. (1992), 'Double penalty method for bilevel optimization problems', *Annals of Operations Research* **34**(1), 73–88.

Ivanenko, D. & Plyasunov, A. (2008), 'Reducibility of bilevel programming problems to vector optimization problems', *Journal of Applied and Industrial Mathematics* **2**(2), 179–195.

Jeroslow, R. G. (1985), 'The polynomial hierarchy and a simple model for competitive analysis', *Mathematical Programming* **32**, 146–164. https://doi.org/10.1007/BF01586088.

Kellerer, H., Pferschy, U. & Pisinger, D. (2004), *Knapsack Problems*, Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24777-7.

Kieffer, E., Danoy, G., Brust, M. R., Bouvry, P. & Nagih, A. (2020), 'Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic', *IEEE Transactions on Evolutionary Computation* **24**(1), 44–56.

Kleinert, T., Labbé, M., Ljubić, I. & Schmidt, M. (2021), 'A survey on mixed-integer programming techniques in bilevel optimization', *EURO Journal on Computational Optimization* **9**.

Kleinert, T., Labbé, M., Plein, F. a. & Schmidt, M. (2020), 'There's no free lunch: on the hardness of choosing a correct big-m in bilevel optimization', *Operations research* **68**(6), 1716–1721.

Kleinert, T. & Schmidt, M. (2023), 'Why there is no need to use a big-m in linear bilevel optimization: A computational study of two ready-to-use approaches', *Computational Management Science* **20**(1), 3.

Kolstad, C. D. & Lasdon, L. S. (1990), 'Derivative evaluation and computational experience with large bilevel mathematical programs', *Journal of optimization theory and applications* **65**, 485–499.

Köppe, M., Queyranne, M. & Ryan, C. T. (2010), 'Parametric integer programming algorithm for bilevel mixed integer programs', *Journal of Optimization Theory and Applications* **146**(1), 137–150. http://dx.doi.org/10.1007/s10957-010-9668-3.

Lauer, J., Bertram, M. & Morton, A. (2020), Cost effectiveness analysis, *in* 'Global Health Priority-Setting: Beyond Cost-Effectiveness', Oxford University Press.

Legillon, F., Liefooghe, A. & Talbi, E.-G. (2012), Cobra: A cooperative coevolutionary algorithm for bi-level optimization, *in* '2012 IEEE Congress on evolutionary computation', IEEE, pp. 1–8.

Li, X., Tian, P. & Min, X. (2006), A hierarchical particle swarm optimization for solving bilevel programming problems, *in* 'International Conference on Artificial Intelligence and Soft Computing', Springer, pp. 1169–1178.

Liu, G., Han, J. & Wang, S. (1998), 'A trust region algorithm for bilevel programing problems', *Chinese science bulletin* **43**, 820–824.

Liu, S., Wang, M., Kong, N. & Hu, X. (2021), 'An enhanced branch-and-bound algorithm for bilevel integer linear programming', *European Journal of Operational Research* **291**(2), 661–679.

Lozano, L. & Smith, J. C. (2017), 'A value-function-based exact approach for the bilevel mixed-integer programming problem', *Operations Research* **65**(3), 768–786.

Lu, C., Schneider, M. T., Gubbins, P., Leach-Kemon, K., Jamison, D. & Murray, C. J. (2010), 'Public financing of health in developing countries: a cross-national systematic analysis', *The Lancet* **375**(9723), 1375–1387.

Lv, Y., Hu, T., Wang, G. & Wan, Z. (2007), 'A penalty function method based on kuhn–tucker condition for solving linear bilevel programming', *Applied mathematics and computation* **188**(1), 808–813.

Mansi, R., Alves, C., Carvalho, J. & Hanafi, S. (2012), 'An exact algorithm for bilevel 0-1 knapsack problems', *Mathematical Problems in Engineering* **2012**.

Marcotte, P., Savard, G. & Zhu, D. (2001), 'A trust region algorithm for nonlinear bilevel programming', *Operations research letters* **29**(4), 171–179.

Mathieu, R., Pittard, L. & Anandalingam, G. (1994), 'Genetic algorithm based approach to bi-level linear programming', *RAIRO-Operations Research* **28**(1), 1–21.

McMasters, A. W. & Mustin, T. M. (1970), 'Optimal interdiction of a supply network', *Naval Research Logistics Quarterly* **17**(3), 261–268.

Mersha, A. G. & Dempe, S. (2006), 'Linear bilevel programming with upper level constraints depending on the lower level solution', *Applied Mathematics and Computation* **180**(1), 247–254.

Moore, J. T. & Bard, J. F. (1990), 'The mixed integer linear bilevel programming problem', *Operations Research* **38**(5), 911–921.

Morton, A. (2014), 'Aversion to health inequalities in healthcare prioritisation: A multicriteria optimisation perspective', *Journal of health economics* **36**, 164–173.

Morton, A., Arulselvan, A. & Thomas, R. (2018), 'Allocation rules for global donors', *Journal of Health Economics* **58**, 67 – 75.

Morton, A., Thomas, R. & Smith, P. C. (2016), 'Decision rules for allocation of finances to health systems strengthening', *Journal of Health Economics* **49**, 97–108.

Muraközy, B. & Telegdy, Á. (2016), 'Political incentives and state subsidy allocation: Evidence from hungarian municipalities', *European Economic Review* **89**, 324–344.

Oduguwa, V. & Roy, R. (2002), Bi-level optimisation using genetic algorithm, *in* 'Proceedings 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002)', IEEE, pp. 322–327.

Pferschy, U., Nicosia, G. & Pacifici, A. (2019), 'A stackelberg knapsack game with weight control', *Theoretical Computer Science* **799**, 149–159.

Pferschy, U., Nicosia, G., Pacifici, A. & Schauer, J. (2021), 'On the stackelberg knapsack game', *European Journal of Operational Research* **291**(1), 18–31.

Qiu, X. & Kern, W. (2015), 'Improved approximation algorithms for a bilevel knapsack problem', *Theoretical computer science* **595**, 120–129.

Roodman, D. (2012), 'Doubts about the evidence that foreign aid for health is displaced into non-health uses', *The Lancet* **380**(9846), 972–973.

Ruuska, S., Miettinen, K. & Wiecek, M. M. (2012), 'Connections between single-level and bilevel multiobjective optimization', *Journal of Optimization Theory and Applications* **153**, 60–74.

Saharidis, G. K. & Ierapetritou, M. G. (2009), 'Resolution method for mixed integer bi-level linear problems based on decomposition technique', *Journal of Global optimization* **44**, 29–51.

Savard, G. & Gauvin, J. (1994), 'The steepest descent direction for the nonlinear bilevel programming problem', *Operations Research Letters* **15**(5), 265 – 272.

Shaw, R. P., Wang, H., Kress, D. & Hovig, D. (2015), 'Donor and domestic financing of primary health care in low income countries', *Health Systems & Reform* **1**(1), 72–88.

Shi, C., Lu, J. & Zhang, G. (2005), 'An extended kuhn–tucker approach for linear bilevel programming', *Applied Mathematics and Computation* **162**(1), 51–63.

Sinha, A., Lu, Z., Deb, K. & Malo, P. (2020), 'Bilevel optimization based on iterative approximation of multiple mappings', *Journal of Heuristics* **26**, 151–185.

Sinha, A., Malo, P. & Deb, K. (2014), An improved bilevel evolutionary algorithm based on quadratic approximations, *in* '2014 IEEE congress on evolutionary computation (CEC)', IEEE, pp. 1870–1877.

Sinha, A., Malo, P. & Deb, K. (2016), Solving optimistic bilevel programs by iteratively approximating lower level optimal value function, *in* '2016 IEEE Congress on Evolutionary Computation (CEC)', IEEE, pp. 1877–1884.

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

Sinha, A., Malo, P. & Deb, K. (2018), 'A review on bilevel optimization: From classical to evolutionary approaches and applications', *IEEE Transactions on Evolutionary Computation* **22**(2), 276–295.

Smith, J. C. & Song, Y. (2020), 'A survey of network interdiction models and algorithms', *European Journal of Operational Research* **283**(3), 797–811.

Solodov, M. (2007), 'An explicit descent method for bilevel convex optimization', *Journal of Convex Analysis* **14**(2), 227.

Stackelberg, H. v. (1934), *Marktform und Gleichgewicht*, J. Springer.

Stinnett, A. A. & Paltiel, A. D. (1996), 'Mathematical programming for the efficient allocation of health care resources', *Journal of Health Economics* **15**(5), 641–653.

Tahernejad, S., Ralphs, T. K. & DeNegre, S. T. (2020), 'A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation', *Mathematical Programming Computation* **12**(4), 529–568.

Talbi, E.-G. (2013), *Metaheuristics for Bi-level Optimization*, Springer Berlin Heidelberg, Berlin, Heidelberg.

Teerawattananon, Y., McQueston, K., Glassman, A., Yothasamut, J. & Myint, C. Y. (2013), 'Health technology assessments as a mechanism for increased value for money: recommendations to the global fund', *Globalization and Health* **9**, 1–9.

UnitedNations, A. (2015), 'We can end poverty: Millenium development goals and beyond'. `http://www.un.org/millenniumgoals/aids.shtml`.

Van de Sijpe, N. (2013), 'The fungibility of health aid reconsidered', *Journal of Development Studies* **49**(12), 1746–1754.

Vicente, L., Savard, G. & Júdice, J. (1994), 'Descent approaches for quadratic bilevel programming', *Journal of optimization theory and applications* **81**(2), 379–399.

Vicente, L., Savard, G. & Judice, J. (1996), 'Discrete linear bilevel programming problem', *Journal of optimization theory and applications* **89**(3), 597–614.

Wan, Z., Wang, G. & Sun, B. (2013), 'A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems', *Swarm and Evolutionary Computation* **8**, 26–32.

Wang, G., Wan, Z., Wang, X. & Lv, Y. (2008), 'Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem', *Computers & Mathematics with Applications* **56**(10), 2550–2555.

Wang, L. & Xu, P. (2017), 'The watermelon algorithm for the bilevel integer linear programming problem', *SIAM Journal on Optimization* **27**(3), 1403–1430.

Wang, Y., Jiao, Y.-C. & Li, H. (2005), 'An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-handling scheme', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **35**(2), 221–232.

Wang, Y., Li, H. & Dang, C. (2011), 'A new evolutionary algorithm for a class of nonlinear bilevel programming problems and its global convergence', *INFORMS Journal on Computing* **23**(4), 618–629.

Weinstein, M. & Zeckhauser, R. (1973), 'Critical ratios and efficient allocation', *Journal of Public Economics* **2**(2), 147–157.

Wogrin, S., Pineda, S. & Tejada-Arango, D. A. (2020), Applications of bilevel optimization in energy and electricity markets, *in* S. Dempe & A. Zemkoho, eds, 'Bilevel Optimization: Advances and Next Challenges', number 978-3-030-52119-6 *in* 'Springer Optimization and Its Applications', Springer, pp. 139–168.

Wood, R. K. (1993), 'Deterministic network interdiction', *Mathematical and Computer Modelling* **17**(2), 1–18.

Xu, P. & Wang, L. (2014), 'An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions', *Computers Operations Research* **41**, 309 – 318.

Yin, Y. (2000), 'Genetic-algorithms-based approach for bilevel programming models', *Journal of Transportation Engineering* **126**(2), 115–120.

Ünlü, G. (1987), 'A linear bilevel programming algorithm based on bicriteria programming', *Computers Operations Research* **14**(2), 173–179. `https://www.sciencedirect.com/science/article/pii/0305054887900086`.