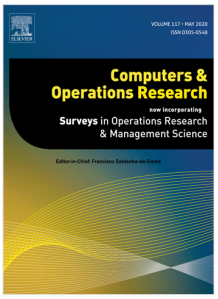
Journal Pre-proof



Solution techniques for Bi-level Knapsack Problems

Shraddha Ghatkar, Ashwin Arulselvan, Alec Morton

PII: S0305-0548(23)00207-1
DOI: https://doi.org/10.1016/j.cor.2023.106343
Reference: CAOR 106343

To appear in: *Computers and Operations Research*

Received date : 27 June 2022
Revised date : 29 June 2023
Accepted date : 30 June 2023

Please cite this article as: S. Ghatkar, A. Arulselvan and A. Morton, Solution techniques for Bi-level Knapsack Problems. *Computers and Operations Research* (2023), doi: https://doi.org/10.1016/j.cor.2023.106343.

Solution Techniques for Bi-level Knapsack Problems

Shraddha Ghatkar[a,*], Ashwin Arulselvan[a], Alec Morton[a]

[a]*Department of Management Science, University of Strathclyde, 199 Cathedral Street, G4 0QU, Glasgow, United Kingdom*

Abstract

Traditional funding mechanisms for healthcare projects involve ranking the projects and awarding funds based on their cost to benefit ratio. An alternative funding mechanism based on Bi-level programming was proposed in the literature. We refer to this as Donor-Recipient Bi-level Knapsack Problem (DR-BKP), which we explore further in this work. There are two participants, a leader (a donor agency) and a follower (recipient country) in this problem. Both the participants have their individual budgets. There is a set of projects, each having a certain cost and profit associated. The cost of projects are common to both the participants however the profits can be different for them. There is an external project that is of exclusive interest to the follower. The leader decides on cost subsidies to provide for the projects that is within her budget, while the follower solves a knapsack problem with the cost subsidised projects and the external project. Two enumerative algorithms were proposed in the literature for Bi-level problems with discrete upper level variables. We adapt them for DR-BKP that has continuous upper level variables having non-linear interaction with lower level variables. We first show the existence of a solution for DR-BKP and show the convergence of these algorithms. We provide evidence for Σ_2^P-hardness by showing that the problem is both NP-hard and Co-NP hard. Finally, we have implemented these two enumerative algorithms and shared the results and analyses of the computational experiments. A set of fifteen differing data sets each having randomly generated 10 instances have been solved to evaluate the performance of the proposed algorithms.

*Corresponding author

Email addresses: shraddha.ghatkar@strath.ac.uk (Shraddha Ghatkar), ashwin.arulselvan@strath.ac.uk (Ashwin Arulselvan), alec.morton@strath.ac.uk (Alec Morton)

Keywords: Bi-level optimization, Bi-level knapsack problems, Stackelberg games, Healthcare economics

1. Introduction

A Bi-level Programming Problem (BLPP) can be formally stated as

$$\underset{\mathbf{y}\in Y,\ \mathbf{x}\in X}{\text{maximize}} \quad f(\mathbf{y},\mathbf{x}) \tag{1a}$$

$$\text{subject to} \quad F(\mathbf{y},\mathbf{x}) \leq 0 \tag{1b}$$

$$\mathbf{x} \in \ \arg\max_{\mathbf{x}'\in X}\{g(y,x') : G(\mathbf{y},\mathbf{x}') \leq 0\} \tag{1c}$$

We have two decision makers. A leader at the upper level with a set of decision variables, $\mathbf{y} \in Y \subseteq \mathbb{R}^m$, and a follower at the lower level with a set of decision variables, $\mathbf{x} \in X \subseteq \mathbb{R}^n$. 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 takes a decision on $\mathbf{y}$, the follower solves an optimisation problem parameterised by the leader's decision. Bracken and McGill [1] introduced this class of optimization models, wherein the feasible region of leader's optimization problem is the optimal solution set of follower's optimization problem. Typically, a central planner (or leader), with collective utility as objective, makes an investment decision based on which one or more agents (or followers) make decisions that maximises their individual utility. This models the hierarchical relation between two classes of decision makers. Many bi-level optimization problems were originally proposed and studied as Stackelberg games in the field of game theory [2], when there is a hierarchy in decision making among players.

The BLPPs have two different solution approaches depending on the problem, namely, (1) optimistic approach and (2) pessimistic approach. Assuming a follower's optimal solution is not unique, the leader and follower can cooperate in an optimistic approach permitting the leader to pick the best solution among the follower's optimal set of solutions. However, when there is no possibility of cooperation, it is typically assumed that the follower will choose the least favourite solution of the leader from the follower's optimal set and a pessimistic approach is taken. In this work, due to the nature of the application, cooperation is assumed and we discuss optimistic strategies.

We will not discuss pessimistic approaches. We are dealing with a specific type of mixed integer BLPPs, where the leader and the follower have a single knapsack constraint each. The follower solves a knapsack problem that is influenced by decisions made by the leader's knapsack problem. We will formalise this in the sequel.

Motivation: Significant advancements have been made towards prevention and treatment of human race against diseases world-wide in the last two decades [3]. This achievement is 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 organisations like the Global Fund. These organisations raise funds from public sector contributions of nations with large gross national income and philanthropic organizations.

The recipient is often a developing country that manages her 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. 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 most followed approach for allocating the healthcare funds is of cost-effective analysis. The projects with higher benefits and lower costs are funded first. However, Morton *et al.* [4] 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 herself 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 herself. Figure 1 shows the line of marginal cost-effectiveness, projects funded by the country herself and projects that can be subsidized and brought to the marginal line. The given approach assures not only efficient allocation of the available funds but it is also inline with the idea of "sustainable" aid. The authors referred to this as Bi-level Knapsack Problem (BKP). In this work, we refer to this approach as Donor-Recipient Bi-level Knapsack problem (DR-BKP) to distinguish it from other bi-level knapsack problems that are available in the

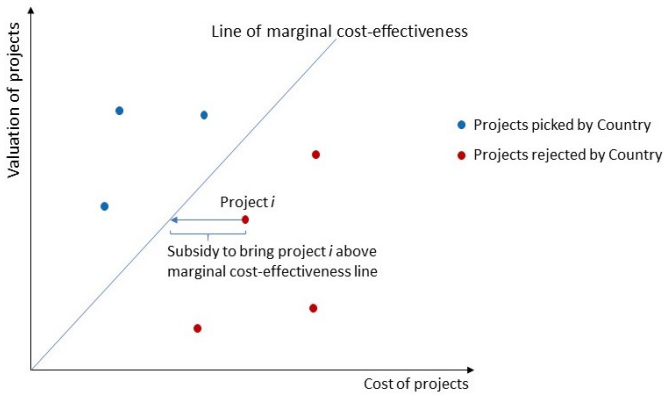


Figure 1: Project selection by Recipient Country and Donor subsidies

literature (see Caprara *et al.* [5]). This model was first reformulated into a single level mixed integer program under the assumption that the recipient is a middle income country capable of funding all the health-care projects from her own budget and then it was solved using a standard optimization solver in [4]. The assumption made in [4] is relaxed in this paper in order to generalize the model for wider use.

1.1. Contribution:

The main contributions of this work are extending and adapting the ideas of two finitely converging exact algorithms, (a) an enumeration algorithm by Lozano and Smith [6] and (b) a branching technique by Xu and Wang [7] to solve the `DR-BKP` model. We differ from both these models in a number of ways. Firstly, our problem has continuous variables in its upper level and both continuous and discrete variables at the lower level. Most mixed integer bi-level problems assume that the lower level problem is parameterised exclusively by the upper level integer variables. Continuous upper level variables in the lower level problem impose two difficulties. The first one being 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.* [8]. An example is given by Köppe *et al.* [9] 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 [6] and [7] require that the lower level problem is parameterised by integer upper level variables to avoid open feasible sets. We show 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 parameterised lower level problem is a mixed integer linear program. This requires us to modify both the constraint and branching rules of [6] and [7].

We make an assumption on the cost of projects that we will discuss later in the sequel. With this assumption, we show convergence and the computational experiments were performed and presented. We also provide evidence for Σ_2^P-complexity in Section 3.1, by showing the decision version of the problem is both NP-hard and Co-NP hard.

The problem is formally defined in Section 2 followed by complexity of the `DR-BKP`. The enumeration algorithm and the branching technique are given in Sections 4 and 5 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 6. These data sets have been generated to mimic the different scenarios arising in real-life healthcare problems. Finally, the last section concludes this paper along with findings and suggestions for future research.

1.2. Previous work

There are several real-life problems modeled as `BLPPs`, such as transportation network design by Constantin and Florian [10], national agricultural planning by Fortuny-Amat and McCarl [11] and revenue management by Morton *et al.* [4], and Côté *et al.* [12]. Although the application areas are wide, there are not many implementations seen due to the lack of efficient algorithms to deal with actual problem sizes in real-life. Hence a lot of attention has been shifted recently to solve these extremely challenging set of problems. The difficulty of these problems can be realised 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 (see Jeroslow [13]). 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 (see Colson *et al.* [14] and Mersha and Dempe [15]). These developed models have different solution techniques depending on the individual problem's structures.

The latest and by far the largest annotated list of references can be found in the book by Dempe and Zemkoho [16].

Vicente *et al.* [8], Dempe [17], and Fanghänel and Dempe [18] discussed properties and optimality conditions of BLPPs with different structures related to the integer and continuous variables appearing in both levels of the models. A Branch-and-Bound algorithm was given by Moore and Bard [19] for general mixed integer BLPPs, along with some heuristics to trade-off accuracy for speed and obtain good solutions for larger instances. Another Branch-and-Bound algorithm was given by Edmunds and Bard [20] for mixed integer BLPPs where discrete variables appear only in the upper level. Later, a Branch-and-Cut approach for integer BLPPs was given by DeNegre and Ralphs [21] by introducing cutting planes derived in a similar way for standard Integer Linear Programs (ILPs). This work was improved and a generalized Branch-and-Cut Algorithm was proposed and implemented in an open source solver by Tahernejad *et al.* [22].

Xu and Wang [7] 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 and Xu [23]), for solving BLPPs with discrete variables in both the 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 and Smith [6] give an exact finite algorithm using optimal-value-function reformulation. They iteratively generate primal bounds using relaxed BLPP and dual bounds using bi-level feasible solutions obtained until the bounds are within desired solution gaps.

A finitely-convergent solver was then given by Fischetti *et al.* [24] 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. This work was improved upon in their proceeding article [25] in which they have proposed new families of intersection cuts and separation algorithms. An extensive computational study was 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.* [26] have recently proposed an enhanced Branch-and-Bound algorithm for BLPPs with discrete variables that are bounded in both the levels. Their algorithm has improved branching

rule over that given in [7] and hence can disregard larger bi-level infeasible spaces in each iteration during the search. The following section gives the literature that we studied related to bi-level knapsack problems.

Bi-level Knapsack Problems:

In general, there are three variants of bi-level knapsack problems (BKPs) as discussed by Caprara *et al.* [5] and Carvalho [27]. We discuss these along with some of their extensions recently seen in the literature. First is the Dempe-Ritcher variant by Dempe and Ritcher [28] 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 the leader and the follower is to maximize their respective profits. A pseudo-polynomial exact algorithm and polynomial time approximate algorithm were given by the authors. A Dynamic Programming (DP) algorithm was given by Brotcorne *et al.* [29] 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.

Second variant is called the Mansi-Alves-de-Carvalho-Hanafi variant given by Mansi *et. al.* [30] in which the knapsack is shared by both the leader and the follower with a pre-decided budget. A reformulation approach was given by Brotcorne *et al.* [31] for integer BKPs, which was then solved using a two-step algorithm. A DP approach was used to find all possible reactions of the follower in the first step and all the obtained reactions were used to reformulate the BKPs as a single level MILP in the second step. This reformulation was then solved using an MILP solver.

DeNegre [32] has given the third variant of the BKPs. In this variant, both the leader and the follower have their independent knapsacks and they select items from a common set of items. The objective of follower is to maximize her profit whereas that of the leader is to minimize profit of the follower. This variant is a type of Interdiction models. DeNegre [32] has developed a Branch-and-Cut framework to solve pure integer framework and a reformulation approach to solve this variant. Another solver has been given by Caprara *et al.* [33], 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 and Scatamacchia [34] 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 an improved performance for over a Lagrangian based Branch-and-Cut approach proposed by Furini *et al.* [35]. An exact Branch-and-Cut algorithm has been recently given by Fischetti *et al.* [36] 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 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.* [37], the follower has to select a set of items within a given weight and in minimum cost. Since the follower is computationally bounded, she 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 and Zhang [38] and then improved upon by Qiu and Kern [39] for different versions of a BKP variant in which both the leader and the follower pack their items simultaneously in their own knapsacks. The follower maximizes her own profit however leader is concerned to maximize both the profits.

Two variants of BKPs with continuous variables in upper level and binary variables in lower level are given by Pferschy *et al.* [40] and Pferschy *et al.* [41]. Greedy heuristics and pseudo-polynomial time exact algorithms were provided for these problems. In these variants, the items of the knapsack are partitioned as leader and follower's 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 she will receive while awarding the remaining profits to the follower and thereby incentivising the follower to pick the leaders' items in the knapsack. Incentives can also be offered as weight offsets in the knapsack and these are deducted from the leader's profit [40]. For instance, in the application provided in [40], we have a trader that provides cost offsets to the products offered to his customers. This is modelled as the trader having a reduced return due to the cost offset she provides. Typically traders can borrow and invest the loan for the offset. One would typically then maximise the return on the profits provided by the products after the interest from the loan has been deducted. In addition, our methodology could possibly be applied to other BKPs, whenever the upper level decisions are present exclusively in the knapsack constraint. For instance, follower's

problem with knapsack precedence constraints (see Johnson and Niemi [42]) will not be affected by our methodology.

One can perceive the BKP that we have proposed in this paper as an extension or variation of the Dempe-Ritcher variant. In this variant [28], a subsidy is directly provided to expand the budget of the recipient. In our model, the leader has a 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.

2. Notation and Definitions

An instance of a knapsack problem comprises of 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 maximised. An instance of Donor-Recipient Bi-level Knapsack problem (DR-BKP) comprises of two players, a donor and a recipient. We have a set, I, of n projects ($I = \{1 \dots n\}$) that are common to both players. Each project $i \in I$, has a profit of $w_i \in \mathbb{N}$ (corr. $v_i \in \mathbb{N}$) for the donor (corr. 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 her 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". We will consider the model introduced in [4], where the external project has a linear profit and linear cost of v_0 and c_0 respectively. So an instance of 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 cost of project i that is subsidised by the donor. These projects are binary and cannot be fractionally picked. Along with the healthcare projects, the recipient has to fund external project which can be done fractionally or wholly *i.e.* only a proportion or entire of cost of the external project can be funded by the recipient.

A solution to an instance of DR-BKP is to decide on the proportion of cost to be subsidised, 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 maximised given the projects are in the optimal solution

set of the recipient's cost subsidised knapsack problem. We use the notation $\mathbf{y}$ to denote a vector of subsidy. The leader cannot subsidise a project more than her 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 proportion of cost of 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

$$\mathcal{G}(\mathbf{y}) := \{\mathbf{x} \in \mathcal{X} : \sum_{i\in I}(c_i - c_i y_i)x_i \leq B_r\} \tag{2}$$

The DR-BKP proposed by Morton *et al.* [4] has been given in (3) and (4) where the upper level is the donor problem (DONOR) and the lower level is the recipient problem (RECIPIENT(y)) parameterised on the upper level decision $\mathbf{y}$.

Problem DONOR:

$$\begin{aligned} \text{maximize} \quad & \mathbf{w}^T\mathbf{x} && \text{(3a)}\\ \text{subject to} \quad & \mathbf{y} \in Y && \text{(3b)}\\ & \mathbf{x} \in \arg\max(\texttt{RECIPIENT(y)}) && \text{(3c)} \end{aligned}$$

Problem RECIPIENT(y):

$$\begin{aligned} \text{maximize} \quad & \mathbf{v}^T\mathbf{x} + v_0 x_0 && \text{(4a)}\\ \text{subject to} \quad & \sum_{i\in I}(c_i - c_i y_i)x_i + c_0 x_0 \leq B_r && \text{(4b)}\\ & (\mathbf{x}, x_0) \in X && \text{(4c)} \end{aligned}$$

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

$$S = \left\{(\mathbf{y}, (\mathbf{x}, x_0)) : \sum_{i\in I}(c_i - c_i y_i)x_i + c_0 x_0 \leq B_r, \mathbf{y} \in Y, (\mathbf{x}, x_0) \in X\right\} \tag{5}$$

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

$$P(\hat{\mathbf{y}}) = \Big\{(\mathbf{x}, x_0) : (\mathbf{x}, x_0) \in \arg\max \Big\{\mathbf{v}^T\mathbf{x} + v_0 x_0 : \sum_{i\in I} c_i x_i + c_0 x_0 \leq B_r + \sum_{i\in I} c_i \hat{y}_i x_i, (\mathbf{x}, x_0) \in X\Big\}\Big\} \tag{6}$$

Inducible Region:

$$IR = \{(\mathbf{y}, (\mathbf{x}, x_0)) \in S : (\mathbf{x}, x_0) \in P(\mathbf{y})\} \tag{7}$$

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

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{8a}$$

$$\text{subject to} \quad (\mathbf{y}, (\mathbf{x}, x_0)) \in IR \tag{8b}$$

3. Properties and assumptions

Let R-DR-BKP, given as (9), 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.

$$\text{maximize} \quad \mathbf{w}^T\mathbf{x} \tag{9a}$$

$$\text{subject to} \quad (\mathbf{y}, (\mathbf{x}, x_0)) \in S \tag{9b}$$

The integer constraints in 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 $\overline{\text{HPR}}$ after relaxing its integer constraints. The inducible region of $\overline{\text{HPR}}$ may not contain the inducible region of original problem [19]. Also unlike in the case of standard MILPs, unboundedness of $\overline{\text{HPR}}$ relaxation cannot be used to derive the optimal solution of original bi-level problem. An unbounded $\overline{\text{HPR}}$ region can imply either infeasible, unbounded or occurrence of an optimal solution [7]. 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 [8], [9]. An example in these works convey 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. It is not difficult to see this. For a fixed set of projects, $\mathbf{x}^k \in \mathcal{X}$, let

$$R(\mathbf{x}^k) := \Big\{\mathbf{y} \in Y : \max_{(\mathbf{x},x_0)\in X} \Big\{\mathbf{v}^t\mathbf{x} + v_0x_0 : \sum_{i\in I}(c_i - c_iy_i)x_i \le B_r\Big\} \le \mathbf{v}^t\mathbf{x}^k + v_0x_0(\mathbf{x}^k,\mathbf{y}), \quad (10)$$
$$\sum_{i\in I}(c_i - c_iy_i)x_i^k \le B_r\Big\}$$

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 $\mathbf{x}^k \in \mathcal{X}, k = 1,\ldots,K$, 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 $R(\mathbf{x}^{\pi_k}) \ne \emptyset$. 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 $\mathbf{y} \in Y$. So we have the following result.

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

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

$$c_0 \ge B_r \quad (11)$$

The reason for this assumption is given in Section 4. Under this assumption we have

$$x_0(\mathbf{x}^k, \mathbf{y}) = \frac{B_r - \sum_{i \in I} c_i x_i^k + \sum_{i \in I} c_i y_i x_i^k}{c_0}$$

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.

3.1. Complexity of the BKP

Regardless of the cost assumption (11), the results of this section hold. We now provide evidence for 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. 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 $\mathbf{w}^T\hat{\mathbf{x}} \geq k$*
- ***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 a 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 the donor and recipient's profit 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 subsidised by the leader. Otherwise it is infeasible to the follower. If KNAPSACK instance is yes, then the leader could simply subsidise the items in this set fully. Otherwise, no subset of items that can fully be subsidised (within the budge 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 subsidise 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 maximise her 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.

4. Enumeration Algorithm

Consider the model (9) after linearising the products of donor subsidies y_i and project selections x_i in the lower level budget constraint.

Problem R-DR-BKP:

$$\begin{aligned}
\text{maximize} \quad & \mathbf{w}^T\mathbf{x} && && (12a)\\
\text{subject to} \quad & \mathbf{c}^T\mathbf{y} \leq B_d && && (12b)\\
& \mathbf{c}^T\mathbf{x} + c_0x_0 \leq B_r + \mathbf{c}^T\mathbf{y} && && (12c)\\
& y_i \leq x_i && \forall i \in I && (12d)\\
& \mathbf{y} \in [0,1]^n && && (12e)\\
& \mathbf{x} \in \{0,1\}^n && && (12f)\\
& x_0 \in [0,1] && && (12g)
\end{aligned}$$

Constraint (12d) 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 (4b) and rewrite that constraint as (12c).

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

$$\mathbf{v}^T\mathbf{x} + v_0x_0 \geq \mathbf{v}^T\bar{\mathbf{x}} + v_0\bar{x}_0 \tag{13}$$

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

$$\bar{x}_0 = x_0(\bar{\mathbf{x}}, \mathbf{y}^*) \tag{14}$$

Using this, inequality (13) can be written as

$$\mathbf{v}^T\mathbf{x} + v_0x_0 \geq \mathbf{v}^T\bar{\mathbf{x}} + v_0x_0(\bar{\mathbf{x}}, \mathbf{y}) \tag{15}$$

Under assumption(11), 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 (15) and we do not have to introduce binary variables. We define $c'_i := \frac{v_0 c_i}{c_0}$. Inequality (15) can then be re-written as

$$\mathbf{v}^T\mathbf{x} + v_0 x_0 - \sum_{i\in I} c'_i \bar{x}_i y_i \geq \mathbf{v}^T\bar{\mathbf{x}} + \frac{v_0}{c_0}(B_r - \mathbf{c}^T\bar{\mathbf{x}}) \tag{16}$$

However, inequality (16) 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 constraints (17d) and (17e) 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 6.

$$\begin{aligned}
\text{maximize} \quad & \mathbf{w}^T\mathbf{x} && \text{(EBKP)}\\
\text{subject to} \quad & \mathbf{c}^T\mathbf{y} \leq B_d && \text{(17a)}\\
& \mathbf{c}^T\mathbf{x} + c_0 x_0 \leq B_r + \mathbf{c}^T\mathbf{y} && \text{(17b)}\\
& y_i \leq x_i \quad \forall i \in I && \text{(17c)}\\
& \mathbf{c}^T\mathbf{x}^k - \sum_{i\in I} c_i x_i^k y_i + M_1 t^k \geq B_r + \epsilon \quad \forall k \in \{1,\dots,K\} && \text{(17d)}\\
& \mathbf{v}^T\mathbf{x} + v_0 x_0 - \sum_{i\in I} c'_i x_i^k y_i + M_2(1 - t^k) \\
& \qquad \geq \mathbf{v}^T\mathbf{x}^k + \frac{v_0}{c_0}(B_r - \mathbf{c}^T\mathbf{x}^k) \quad \forall k \in \{1,\dots,K\} && \text{(17e)}\\
& t^k \in \{0,1\} \quad \forall k \in \{1,\dots,K\} && \text{(17f)}\\
& \mathbf{y} \in [0,1]^n && \text{(17g)}\\
& \mathbf{x} \in \{0,1\}^n && \text{(17h)}\\
& x_0 \in [0,1] && \text{(17i)}
\end{aligned}$$

For any solution $\mathbf{x}^k \in \mathcal{X}$, constraint (17d) forces the binary variable t^k to 1 if the cost of projects in $\mathbf{x}^k$ that are subsidised 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 modelled 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 (17e) 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 $\texttt{RECIPIENT}(\mathbf{y}^*)$. If $(\mathbf{x}^*, x_0^*) \in P(\mathbf{y}^*)$, then we terminate otherwise $\texttt{RECIPIENT}(\mathbf{y}^*)$ returns an optimal solution $(\bar{\mathbf{x}}, \bar{x}_0) \in P(\mathbf{y}^*)$ that we add as constraints of the form (17d) and (17e). 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 [6]. They aggregate their constraints (17d) 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 (17e) and one new variable have to be added at every iteration in [6]. 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 [6], 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):
      $\mathbf{c}^T\mathbf{x}^k - \sum_{i\in I} c_i x_i^k y_i + M_1 t^k \geq B_r + \epsilon$;
      $\mathbf{v}^T\mathbf{x} + v_0x_0 - \sum_{i\in I} c_i' x_i^k y_i + M_2(1-t^k) \geq \mathbf{v}^T\mathbf{x}^k + \frac{v_0}{c_0}(B_r - \mathbf{c}^T\mathbf{x}^k)$;
    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 $(\mathbf{x}^\mathbf{k}, x_0^k)$ 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 (17d) and(17e) 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 bilevel 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 (17d) to avoid strict inequalities. Let us refer to the theoretical model obtained from (EBKP) with (17d) replaced by the strict inequality

$$\mathbf{c}^T\mathbf{x}^k - \sum_{i\in I} c_i x_i^k y_i + M_1 t^k > B_r \qquad \forall k \in \{1,\ldots,K\} \tag{18}$$

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 (17d). 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 $k = 1,\ldots,\kappa$

$$B_r < \mathbf{c}^T\bar{\mathbf{x}}^k - \sum_{i\in I} c_i\bar{x}_i^k\tilde{y}_i < B_r + \epsilon - \epsilon_1 \tag{19}$$

$$\sum c_i\bar{x}_i\tilde{y}_i + (B_r - \mathbf{c}^T\bar{\mathbf{x}}^k) < 0 < \sum c_i\bar{x}_i\tilde{y}_i + (B_r - \mathbf{c}^T\bar{\mathbf{x}}^k) + \epsilon - \epsilon_1 \tag{20}$$

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 (20) 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 (17d). Since we are only reducing the value of $\tilde{\mathbf{y}}$ to get $\hat{\mathbf{y}}$, (17a), (17c) and (17e) are also feasible. In order to show feasibility of (17b), 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

$$\mathbf{c}^T\tilde{\mathbf{x}} + c_0\tilde{x}_0 \leq B_r + \mathbf{c}^T\tilde{\mathbf{y}} \tag{21}$$
$$\mathbf{c}^T\tilde{\mathbf{x}} + c_0\tilde{x}_0 \leq B_r + \mathbf{c}^T\hat{\mathbf{y}} + n(\epsilon - \epsilon_1) \tag{22}$$
$$\mathbf{c}^T\tilde{\mathbf{x}} + c_0\tilde{x}_0 \leq B_r + \mathbf{c}^T\hat{\mathbf{y}} + \epsilon_1 \tag{23}$$

□

5. Branching Algorithm

In Section 4, 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 was proposed by [7], where $(\mathbf{y}^*, (\mathbf{x}^*, x_0))$ is eliminated from search using a branching rule. The branching rule proposed in [7] 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.

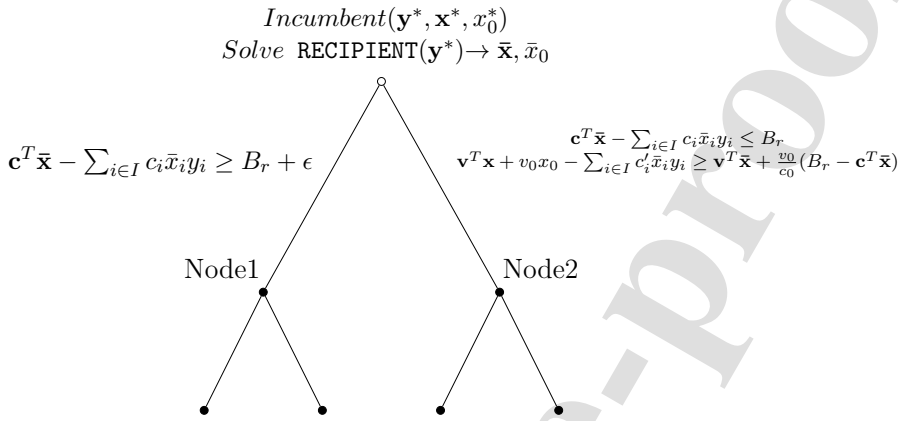


Figure 2: Branching from an incumbent solution

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 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 2, there are two branches generated. Node 1 is explored where a valid subsidy $\mathbf{y}$ is such that $\bar{x}$ is infeasible for RECIPIENT$(\mathbf{y})$. Node 2 is explored where a valid solution $(\mathbf{y}, \mathbf{x}, x_0)$ is such that $\bar{\mathbf{x}}$ is feasible for RECIPIENT$(\mathbf{y})$ and $\mathbf{v}^T\mathbf{x} + v_0x_0 \geq \mathbf{v}^T\bar{\mathbf{x}} + v_0x_0(\bar{\mathbf{x}}, \mathbf{y})$. This idea is similar to constraints (17d) and (17e) in MILP-DR-BKP.

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 2
  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

6. Computational Experiments

To understand, analyze and compare the performance of both the proposed algorithms, the computational study was performed on an HP computer (Windows 10 Enterprise with 64-bit operating system, 3.19 GHz processor and 8GB RAM). The algorithms were coded and solved in Python 3.8 using CPLEX 20.1.0.

6.1. Data Generation

Instance generation was guided by the real-world instance presented in [4]. There are 15 data sets generated[1] as shown in Table 1 and 10 instances

	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]	40	0.7	[1,1]	40	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 1: Input Parameters for Data Generation

are generated and solved in every data set. The first data set has 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 were 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. 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 the `DONOR` and `RECIPIENT` budgets are generated as percentage of 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 was used. Profit to cost ratio of external project in each instance is the 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 were made to understand the performance of the developed algorithms on every parameter in the instances generated. For second, third and fourth data sets, the

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

number of projects have 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 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 divergence in priorities of `DONOR` and `RECIPIENT`.

6.2. Results

We have conducted the computational experiments to compare performance of both the algorithms at two different tolerance parameters (at $\epsilon = 1e-2$ and $\epsilon = 1e-4$) for set time limit of 3600 seconds. The value of big M, M_1, in constraint (17c) 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 (17d) 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) $M_2 = M_2^k = \mathbf{v}^T\mathbf{x}^k + \frac{v_0}{c_0}(B_r - \mathbf{c}^T\mathbf{x}^k) \quad \forall k \in 1, \dots K$ and (2) $M_2 = \sum v_i + v_0$.

The minimum, average and maximum solution times of both the algorithms for solving the 10 instances in each of the data sets are given in Table 2 when $\epsilon = 1e-2$ and Table 3 when $\epsilon = 1e-4$. These are the data sets that were solved within the set time limit. In cases of data sets 14 and 15, none of the instances were solved to optimality within the set time limit. Their minimum, average and maximum solution gaps at termination of the algorithms are given in Table 4 when $\epsilon = 1e-2$ and Table 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 the algorithms in case of lower tolerance parameter. However in case of higher tolerance parameter, both the algorithms take lesser time to solve. From Data Set 1, 5 and 6, we can observe that the algorithms take lesser time to solve if the

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 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: 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 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 5: Average Solution Gaps for Data Sets that are not solved within Time Limit (Epsilon = 1e-4)

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 start competing with the healthcare projects for the `RECIPIENT` budget, Branching Algorithm performs significantly better than the Enumeration Algorithm for both tolerance limits.

Another complexity of 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 her projects more than the `RECIPIENT` does in that particular class. Else if the α value is below 1, the `DONOR` values her 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 her projects more than the `RECIPIENT`, all instances are solved using both the algorithms very fast. However, none of the instances from data sets 14 and 15 are solved where the `DONOR` values her projects lesser than the `RECIPIENT` (refer to Table 4 and Table 5).

While comparing the performance of enumeration algorithm for both the 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 can 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.

7. Conclusion

We carried out a complexity study and computational experiments on DR-BKP that was introduced in [4]. We first showed that the problem is well-posed. We then extended and adapted the algorithms proposed for discrete bi-level problems to DR-BKP and proved its convergence. We 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 existence of a solution and convergence of algorithms. We have made a simple observation that existence of a solution can be guaranteed if the upper level objective function is discrete, involving only the lower level variables, and if the solution set is finite. This is generalisable and goes beyond DR-BKP. The convergence of the known enumeration schemes tend to work for our problem despite the continuous upper level variables. The idea behind these enumeration schemes is to cutoff the open feasible set at some threshold and restrict the search space to a closed set. This is easier to do when we have integer upper level variables. Despite having continuous upper level variables, we showed that these enumeration schemes work for our problem for sufficiently small thresholds. We showed 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 generalisability of this procedure is not clear.

A number of questions remain open. We explained the similarity between our model and the one presented by Dempe in [17]. The authors have shown that problem has a solution with the inclusion of continuous variable in the upper level objective. A similar extension to our problem is open. Our current complexity proof does not rule out a polynomial solution for unary encoding. A pseudo-polynomial algorithm for this problem (if one exists) would be a valuable practical tool. We have only provided evidence for Σ_p^2-hardness. A direct reduction from a Σ_p^2-complete problem to DR-BKP is open. We will look into extending our ideas to design algorithms that work without the need for assumption (11). At the moment, the external project costs are linear. We want to look at models where the external project costs are concave.

References

[1] J. Bracken and J. T. McGill, "Mathematical programs with optimization problems in the constraints," *Operations Research*, vol. 21, no. 1, pp. 37–44, 1973.

[2] T. Kleinert, M. Labbé, I. Ljubić, and M. Schmidt, "A survey on mixed-integer programming techniques in bilevel optimization," *EURO Journal on Computational Optimization*, vol. 9, 2021.

[3] U. Nations, "We can end poverty: Millenium development goals and beyond," 2015.

[4] A. Morton, A. Arulselvan, and R. Thomas, "Allocation rules for global donors," *Journal of Health Economics*, vol. 58, pp. 67 – 75, 2018.

[5] A. Caprara, M. Carvalho, A. Lodi, and G. J. Woeginger, "A study on the computational complexity of the bilevel knapsack problem," *SIAM Journal on Optimization*, vol. 24, no. 2, pp. 823–838, 2014.

[6] L. Lozano and J. C. Smith, "A value-function-based exact approach for the bilevel mixed-integer programming problem," *Operations Research*, vol. 65, no. 3, pp. 768–786, 2017.

[7] P. Xu and L. Wang, "An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions," *Computers & Operations Research*, vol. 41, pp. 309 – 318, 2014.

[8] L. Vicente, G. Savard, and J. Judice, "Discrete linear bilevel programming problem," *Journal of optimization theory and applications*, vol. 89, no. 3, pp. 597–614, 1996.

[9] M. Köppe, M. Queyranne, and C. T. Ryan, "Parametric integer programming algorithm for bilevel mixed integer programs," *Journal of Optimization Theory and Applications*, vol. 146, p. 137–150, Feb 2010.

[10] I. Constantin and M. Florian, "Optimizing frequencies in a transit network: a nonlinear bi-level programming approach," *International Transactions in Operational Research*, vol. 2, no. 2, pp. 149–164, 1995.

[11] J. Fortuny-Amat and B. McCarl, "A representation and economic interpretation of a two-level programming problem," *The Journal of the Operational Research Society*, vol. 32, no. 9, pp. 783–792, 1981.

[12] J.-P. Côté, P. Marcotte, and G. Savard, "A bilevel modelling approach to pricing and fare optimisation in the airline industry," *Journal of Revenue and Pricing Management*, vol. 2, pp. 23–36, 2003.

[13] R. G. Jeroslow, "The polynomial hierarchy and a simple model for competitive analysis," *Mathematical Programming*, vol. 32, pp. 146–164, 1985.

[14] B. Colson, P. Marcotte, and G. Savard, "Bilevel programming: A survey," *4OR*, vol. 3, pp. 87–107, 06 2005.

[15] A. G. Mersha and S. Dempe, "Linear bilevel programming with upper level constraints depending on the lower level solution," *Applied Mathematics and Computation*, vol. 180, no. 1, pp. 247–254, 2006.

[16] S. Dempe and A. Zemkoho, eds., *Bilevel Optimization*. No. 978-3-030-52119-6 in Springer Optimization and Its Applications, Springer, September 2020.

[17] S. Dempe, "Discrete bilevel optimization problems," *Technical Report D-04109, Institut fur Wirtschaftsinformatik, Universitat Leipzig, Leipzig, Germany*, 2001.

[18] D. Fanghänel and S. Dempe, "Bilevel programming with discrete lower level problems," *Optimization*, vol. 58, no. 8, pp. 1029–1047, 2009.

[19] J. T. Moore and J. F. Bard, "The mixed integer linear bilevel programming problem," *Operations Research*, vol. 38, no. 5, pp. 911–921, 1990.

[20] T. A. Edmunds and J. F. Bard, "An algorithm for the mixed-integer nonlinear bilevel programming problem," *Annals of operations research*, vol. 34, no. 1, pp. 149–162, 1992.

[21] S. T. DeNegre and T. K. Ralphs, "A branch-and-cut algorithm for integer bilevel linear programs," *Book: Operations Research and Cyber-Infrastructure*, pp. 65–78, 2009.

[22] S. Tahernejad, T. K. Ralphs, and S. DeNegre, "A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation," 2017.

[23] L. Wang and P. Xu, "The watermelon algorithm for the bilevel integer linear programming problem," *SIAM Journal on Optimization*, vol. 27, no. 3, pp. 1403–1430, 2017.

[24] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl, "On the use of intersection cuts for bilevel optimization," *Mathematical Programming*, vol. 172, p. 77–103, 2018.

[25] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl, "A new general-purpose algorithm for mixed-integer bilevel linear programs," *Operations Research*, vol. 65, no. 6, pp. 1615–1637, 2017.

[26] S. Liu, M. Wang, N. Kong, and X. Hu, "An enhanced branch-and-bound algorithm for bilevel integer linear programming," *European Journal of Operational Research*, vol. 291, no. 2, pp. 661–679, 2021.

[27] M. Carvalho, *Computation of equilibria on integer programming games.* PhD thesis, 04 2016.

[28] S. Dempe and K. Richter, "Bilevel programming with knapsack constraints," *Central European Journal of Operations Research*, vol. 8, no. 2, pp. 93–107, 2000.

[29] L. Brotcorne, S. Hanafi, and R. Mansi, "A dynamic programming algorithm for the bilevel knapsack problem," *Operations Research Letters*, vol. 37, no. 3, pp. 215 – 218, 2009.

[30] R. Mansi, C. Alves, J. Carvalho, and S. Hanafi, "An exact algorithm for bilevel 0-1 knapsack problems," *Mathematical Problems in Engineering*, vol. 2012, 02 2012.

[31] L. Brotcorne, S. Hanafi, and R. Mansi, "One-level reformulation of the bilevel knapsack problem using dynamic programming," *Discrete Optimization*, vol. 10, no. 1, pp. 1 – 10, 2013.

[32] S. Denegre, *Interdiction and Discrete Bilevel Linear Programming.* PhD thesis, USA, 2011.

[33] A. Caprara, M. Carvalho, A. Lodi, and G. J. Woeginger, "Bilevel knapsack with interdiction constraints," *INFORMS Journal on Computing*, vol. 28, no. 2, pp. 319–333, 2016.

[34] F. Della Croce and R. Scatamacchia, "An exact approach for the bilevel knapsack problem with interdiction constraints and extensions," *Mathematical Programming*, vol. 183, pp. 249–281, 2020.

[35] F. Furini, M. Iori, S. Martello, and M. Yagiura, "Heuristic and exact algorithms for the interval min–max regret knapsack problem," *INFORMS J. Comput.*, vol. 27, p. 392–405, 2015.

[36] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl, "Interdiction games and monotonicity, with application to knapsack problems," *INFORMS Journal on Computing*, vol. 31, no. 2, pp. 390–410, 2019.

[37] P. Briest, L. Gualà, M. Hoefer, and C. Ventre, "On stackelberg pricing with computationally bounded customers," *Networks*, vol. 60, no. 1, pp. 31–44, 2012.

[38] L. Chen and G. Zhang, "Approximation algorithms for a bi-level knapsack problem," *Theoretical Computer Science*, vol. 497, pp. 1–12, 2013.

[39] X. Qiu and W. Kern, "Improved approximation algorithms for a bilevel knapsack problem," *Theoretical computer science*, vol. 595, pp. 120–129, 2015.

[40] U. Pferschy, G. Nicosia, and A. Pacifici, "A stackelberg knapsack game with weight control," *Theoretical Computer Science*, vol. 799, pp. 149–159, 2019.

[41] U. Pferschy, G. Nicosia, A. Pacifici, and J. Schauer, "On the stackelberg knapsack game," *European Journal of Operational Research*, vol. 291, no. 1, pp. 18–31, 2021.

[42] D. S. Johnson and K. A. Niemi, "On knapsacks, partitions, and a new dynamic programming technique for trees," *Mathematics of Operations Research*, vol. 8, no. 1, pp. 1–14, 1983.

- Healthcare funding allocation problem modelled using bi-level optimization framework
- Bi-level Knapsack Problem with continuous variables in upper level and both continuous and discrete variables in lower level
- Evidence for Σ_2^p-hardness of the Bi-level Knapsack Problem
- Comparison of two finitely converging algorithms for solving Bi-level Knapsack Problems

Shraddha Ghatkar: Methodology, Software, Data curation, Analysis, Writing - original draft, Writing - review & editing. **Ashwin Arulselvan:** Conceptualization, Methodology, Validation, Writing - review & editing, Supervision, Funding acquisition. **Alec Morton:** Methodology, Validation, Writing - review & editing, Supervision, Funding acquisition.