A STUDY ON THE COMPUTATIONAL COMPLEXITY OF THE BILEVEL KNAPSACK PROBLEM

ALBERTO CAPRARĂ†, MARGARIDA CARVALHO‡, ANDREA LODI†, AND GERHARD J. WOEGINGER§

Abstract. We analyze the computational complexity of three fundamental variants of the bilevel knapsack problem. All three variants are shown to be complete for the second level of the polynomial hierarchy. We also discuss the somewhat easier situation where the weight and profit coefficients in the knapsack problem are encoded in unary: two of the considered bilevel variants become solvable in polynomial time, whereas the third becomes NP-complete. Furthermore, we design a polynomial time approximation scheme for this third variant, whereas the other two variants cannot be approximated in polynomial time within any constant factor (assuming P ≠ NP).

Key words. bilevel programming, knapsack problem, computational complexity, polynomial hierarchy, approximability, approximation scheme

AMS subject classifications. 90C27, 90C59, 90C60

DOI. 10.1137/130906593

1. Introduction. In bilevel optimization the decision variables are split into two groups that are controlled by two decision makers called leader (on the upper level) and follower (on the lower level). Both decision makers have an objective function of their own and a set of constraints on their variables. Furthermore, there are coupling constraints that connect the decision variables of leader and follower. The decision making process is as follows. First, the leader makes his decision and fixes the values of his variables, and afterwards the follower reacts by setting his variables. The leader has perfect knowledge of the follower’s scenario (objective function and constraints) and also of the follower’s behavior. The follower observes the leader’s action, and then optimizes his own objective function subject to the decisions made by the leader (and subject to the imposed constraints). As the leader’s objective function does depend on the follower’s decision, the leader must take the follower’s reaction into account.

Bilevel optimization is a special case of the general multilevel optimization problem, which deals with a hierarchy of decision makers at an arbitrary number of levels. Bilevel and multilevel optimization have received enormous interest in the literature over the last decades. These concepts go back to the economic duopoly model of von Stackelberg [26] from the 1930s. For an overview of this area, we refer the reader to the book edited by Migdalas, Pardalos, and Várbrand [21] and to the book by Dempe [6], and also to the annotated bibliographies of Vicente and Calamai [30], Dempe [7], and Colson, Marcotte, and Savard [5].
Generally speaking, multilevel optimization problems are extremely difficult from the computational point of view and cannot be expressed in terms of classical integer programs (which can only handle a single level of optimization). A groundbreaking paper by Jeroslow [15] established that various multilevel problems are complete for various levels of the polynomial hierarchy in computational complexity theory; see Stockmeyer [27] and Papadimitriou [23] for more information on the relevant background in computational complexity. Further hardness results for broad families of multilevel optimization problems are due to Deng [10] and Dudás, Klinz, and Woeginger [11].

Standard knapsack problems and bilevel knapsack problems. The knapsack problem is one of the most fundamental problems in combinatorial optimization. It has been studied extensively, as certified, for example, by the books by Martello and Toth [20] and Kellerer, Pferschy, and Pisinger [16]. An instance of the knapsack problem consists of a set of items with given weights and profits together with a knapsack with a given weight capacity. The objective is to select a subset of the items with maximum total profit, subject to the constraint that the overall selected item weight must fit into the knapsack. The knapsack problem is well known to be NP-complete [14].

Over the last few years, a variety of authors has studied certain bilevel variants of the knapsack problem. Dempe and Richter [8] considered the variant where the leader controls the weight capacity of the knapsack, and where the follower decides which items are packed into the knapsack. Mansi et al. [19] consider a bilevel knapsack variant where the item set is split into two parts, one of which is controlled by the leader and one controlled by the follower. DeNegre [9] suggests yet another variant, where both players have a knapsack of their own; the follower can only choose from those items that the leader did not pack. Section 2 gives precise definitions of these three variants and provides further information on them.

Our contributions. We pinpoint the computational complexity of the three bilevel knapsack variants mentioned above: we will show that all three variants are complete for the complexity class \( \Sigma_p^2 \) and hence located at the second level of the polynomial hierarchy. If a problem is \( \Sigma_p^2 \)-complete, there is no way of formulating it as a single-level integer program of polynomial size unless the polynomial hierarchy collapses (a highly unlikely event which would cause a revolution in complexity theory, quite comparable to the revolution that would be caused by a proof that \( P = NP \)). The complexity class \( \Sigma_p^2 \) is the natural hotbed for bilevel problems that are built on top of NP-complete single-level problems; as a rule of thumb, the bilevel version of an NP-complete problem should always be expected to be \( \Sigma_p^2 \)-complete.

In a second line of investigation, we study the three bilevel knapsack variants under so-called unary encodings (where an integer \( n \) is represented as a string of \( n \) ones). The classical knapsack problem becomes much easier and polynomially solvable if the input is encoded in unary, and it is only natural to expect a similar behavior from our bilevel knapsack problems. And indeed, two of our three bilevel variants become polynomially solvable if the input is encoded in unary, and thus show exactly the type of behavior that one would expect from a knapsack variant. The third variant, however, behaves stubbornly and becomes NP-complete under unary encodings, which is not the behavior one would expect from a knapsack variant.

Our third line of results studies the approximability of the three bilevel knapsack variants. As a rule of thumb \( \Sigma_p^2 \)-hard problems do not allow good approximation algorithms. Indeed, the literature only contains negative results in this direction that
establish the inapproximability of various \( \Sigma^p_2 \)-hard optimization problems; see Ko and Lin [17] and Umans [28, 29]. Of particular interest is the paper [29] by Umans that derives strong inapproximability results for \( \Sigma^p_2 \)-hard optimization problems from certain error-correcting codes.

Two of our bilevel knapsack variants (actually the same ones that are easy under unary encodings) behave exactly as expected and do not allow polynomial time approximation algorithms with finite worst case guarantee, assuming \( P \neq \text{NP} \). For the third variant, however, we derive a polynomial time approximation scheme. This is the first approximation scheme for a \( \Sigma^p_2 \)-hard optimization problem in the history of approximation algorithms, and from the technical point of view it is the most sophisticated result in our paper.

Organization of this paper. This paper is structured as follows. Section 2 defines the three bilevel knapsack variants and summarizes the literature on them. Section 3 presents the \( \Sigma^p_2 \)-completeness results for these problems (under the standard binary encoding), and section 4 discusses their behavior under unary encodings. Section 5 discusses the approximability and inapproximability behavior of the considered bilevel problems, and section 6 gives some concluding remarks.

2. Definitions and preliminaries. In bilevel optimization the follower observes the leader’s action, and then optimizes his own objective function value subject to the decisions made by the leader and subject to the imposed constraints. Note that this statement does not fully determine the follower’s behavior: there might be many feasible solutions that all are optimal for the follower but yield different objective values for the leader. Which one will the follower choose? In the optimistic scenario the follower always picks the optimal solution that yields the best objective value for the leader, and in the pessimistic scenario he picks the solution that yields the worst objective value for the leader. All of our negative (hardness) results and all of our positive (polynomial time) results hold for the optimistic scenario as well as for the pessimistic scenario.

The following subsections give exact definitions of the bilevel knapsack variants that will be studied in this paper. Throughout, we use \( x \) and \( x_1, \ldots, x_m \) to denote the variables controlled by the leader, and \( y_1, \ldots, y_n \) to denote the variables controlled by the follower. Furthermore, we use \( a_i, b_i, c_i \) and \( A, B, C, C' \) to denote item profits, item weights, cost coefficients, upper bounds, and lower bounds. All of these numbers are assumed to be nonnegative integers (or rationals). As usual, we will sometimes use the notation \( a(I) = \sum_{i \in I} a_i \) for an index set \( I \), and \( a(x) = \sum_i a_i x_i \) for a 0-1 vector \( x \).

2.1. The Dempe–Richter (DeRi) variant. The first occurrence of a bilevel knapsack problem in the optimization literature seems to be due to Dempe and Richter [8]. In their problem variant DeRi, as depicted in Figure 1, the leader controls the capacity \( x \) of the knapsack while the follower controls all items and decides which of them are packed into the knapsack. The objective function of the leader depends on the knapsack capacity \( x \) as well as on the packed items, whereas the objective function of the follower solely depends on the packed items.

All decision variables in this bilevel program are integers; the knapsack capacity \( x \) is integer, and the variables \( y_1, \ldots, y_n \in \{0, 1\} \) encode whether item \( i \) is packed into the knapsack \( (y_i = 1) \) or not \( (y_i = 0) \). We note that in the original model in [8] the knapsack capacity \( x \) is continuous; one nasty consequence of this continuous knapsack capacity is that the problem (1a)–(1c) may fail to have an optimal solution.
Maximize \( f_1(x, y) = Ax + \sum_{i=1}^{n} a_i y_i \)
subject to (s.t.) \( C \leq x \leq C' \)
where \( y_1, \ldots, y_n \) solves the follower’s problem
\[
\max \sum_{i=1}^{n} b_i y_i \quad \text{s.t.} \quad \sum_{i=1}^{n} b_i y_i \leq x
\]

Fig. 1. The bilevel knapsack problem DeRi.

The computational complexity of the problem remains the same, no matter whether \( x \) is integral or continuous.


2.2. The Mansi–Alves–de-Carvalho–Hanafi (MACH) variant. Mansi et al. [19] consider a bilevel knapsack variant where both players pack items into the knapsack. There is a single common knapsack for both players with a prespecified capacity of \( C \). The item set is split into two parts, which are, respectively, controlled by the leader and the follower. The leader starts the game by packing some of his items into the knapsack, and then the follower adds some further items from his set. The objective function of the leader depends on all items packed by leader and follower, whereas the objective function of the follower solely depends on his own items. Figure 2 specifies the bilevel problem MACH.

Maximize \( f_2(x, y) = \sum_{j=1}^{m} a_j x_j + \sum_{i=1}^{n} a'_i y_i \)
subject to (s.t.) \( y_1, \ldots, y_n \) solves the follower’s problem
\[
\max \sum_{i=1}^{n} b'_i y_i \quad \text{s.t.} \quad \sum_{i=1}^{n} c'_i y_i \leq C - \sum_{j=1}^{m} c_j x_j
\]

Fig. 2. The bilevel knapsack problem MACH.

Mansi et al. [19] describe several applications of their problem in revenue management, telecommunication, capacity allocation, and transportation. Variant MACH has also been studied in a more general form by Brotcorne, Hanafi, and Mansi [2], who reduced the model to one-level in pseudopolynomial time.
2.3. The DeNegre (DNeg) variant. DeNegre [9] proposes another bilevel knapsack variant where both players hold their own private knapsacks and choose items from a common item set. First, the leader packs some of the items into his private knapsack, and then the follower picks some of the remaining items and packs them into his private knapsack. The objective of the follower is to maximize the profit of the items in his knapsack, and the objective of the hostile leader is to minimize this profit.

Minimize
\[ f_3(x, y) = \sum_{i=1}^{n} b_i y_i \]

s.t.
\[ \sum_{i=1}^{n} a_i x_i \leq A \]

where \( y_1, \ldots, y_n \) solves the follower’s problem

\[ \max \sum_{i=1}^{n} b_i y_i \quad \text{s.t.} \quad \sum_{i=1}^{n} b_i y_i \leq B \quad \text{and} \]
\[ y_i \leq 1 - x_i \quad \text{for} \quad 1 \leq i \leq n \]

Figure 3 depicts the bilevel problem DNeg. The 0-1 variables \( x_1, \ldots, x_n \) (for the leader) and \( y_1, \ldots, y_n \) (for the follower) encode whether the corresponding item is packed into the knapsack. The interdiction constraint \( y_i \leq 1 - x_i \) in (3d) enforces that the follower cannot take item \( i \) once the leader has picked it. Note that leader and follower have exactly opposing objectives.

The paper [3] studies a slightly more general version, where the constraint \( \sum_{i=1}^{n} b_i y_i \leq B \) in (3c) reads \( \sum_{i=1}^{n} c_i y_i \leq B \), and thus has cost coefficients that differ from the coefficients in the objective functions of leader and follower.

3. Hardness results under binary encodings. Throughout this section we consider bilevel knapsack problems where the input data is encoded in binary. As usual, we consider the decision versions of these optimization problems: “Does there exist an action of the leader that makes his objective value at least as good as some given bound?” The following theorem summarizes the results of this section.

**Theorem 3.1.** The decision versions of the following bilevel problems (in binary encoding) are \( \Sigma_2 \)-complete, both under the optimistic and under the pessimistic scenario:

(a) The Dempe–Richter (DeRi) variant.
(b) The Mansi–Alves–de-Carvalho–Hanafi (MACH) variant.
(c) The De-Negre (DNeg) variant.

We recall that the complexity class \( \Sigma_2 \) contains all decision problems that can be written in the form \( \exists x, \forall y P(x, y) \); that is, as a logical formula starting with an existential quantifier followed by a universal quantifier followed by a Boolean predicate \( P(x, y) \) that can be evaluated in polynomial time; see, for instance, Chapter 17 in Papadimitriou's book [23]. The decision versions of our bilevel problems DeRi, MACH, DNeg ask whether there exists a way of fixing the variables controlled by the leader,
such that all possible settings of the variables controlled by the follower yield a good objective value for the leader. Since this question is exactly of the form $\exists x \forall y P(x, y)$, we conclude that all three considered bilevel knapsack variants are contained in $\Sigma_2^p$. It remains to show that the three bilevel knapsack variants encapsulate the full difficulty of class $\Sigma_2^p$. The $\Sigma_2^p$-hardness proofs in this section will all be done by reductions from the following decision problem \textsc{Subset-Sum-Interval}, which has been proved to be $\Sigma_2^p$-complete by Eggermont and Woeginger [12].

\textbf{Problem:} \textsc{Subset-Sum-Interval}

\textbf{Instance:} A sequence $q_1, q_2, \ldots, q_k$ of positive integers; two positive integers $R$ and $r$ with $r \leq k$.

\textbf{Question:} Does there exist an integer $S$ with $R \leq S < R + 2^r$ such that none of the subsets $I \subseteq \{1, \ldots, k\}$ satisfies $\sum_{i \in I} q_i = S$?

In our reductions, all feasible solutions that are optimal for the follower will yield the same objective value for the leader. Hence, the constructed instances do not depend on whether the follower behaves benevolently or malevolently towards the leader, and the theorem holds unconditionally under the optimistic scenario as well as under the pessimistic scenario.

3.1. The hardness proof for DeRi. Our reduction starts from an instance of \textsc{Subset-Sum-Interval}. We construct the following instance of DeRi.

- We set $A = 0$, $C = R$, and $C' = R + 2^r - 1$.
- For $i = 1, \ldots, k$, we create a so-called ordinary item $i$ with leader’s profit $a_i = 0$ and follower’s profit/weight $b_i = q_i$.
- Furthermore, there is a special magic item 0 with the leader’s profit $a_0 = 1$ and the follower’s profit $b_0 = 1/2$.

We claim that in the constructed instance of DeRi the leader can make his objective value $\geq 1$ if and only if the \textsc{Subset-Sum-Interval} instance has the answer YES.

(Proof of if.) Assume that the \textsc{Subset-Sum-Interval} instance has the answer YES, and consider the corresponding integer $S$ that cannot be represented as a subset sum. Then a good strategy for the leader is to choose $x = S$ for the knapsack capacity. Suppose for the sake of contradiction that the follower does not pack the magic item. Then the weight of the packed set (and hence the follower’s profit) is at most $S - 1$, which he could improve by adding the magic item to it. This contradiction shows that the magic item must be packed by the follower, which yields a profit of 1 for the leader.

(Proof of only if.) Now assume that the \textsc{Subset-Sum-Interval} instance has the answer NO, and consider the optimal knapsack capacity $x$ for the leader. There exists a subset $I \subseteq \{1, \ldots, k\}$ with $\sum_{i \in I} q_i = x$, and the corresponding set of ordinary items brings a profit of $x$ to the follower. If the follower packs the magic item, then his profit is at most $(x - 1) + 1/2 = x - 1/2$. Consequently, the follower will not pick the magic item, and the objective value of the leader is 0. This completes the proof of Theorem 3.1(a).

3.2. The hardness proof for MACH. We will essentially recycle and imitate the hardness argument from the preceding section. Hence let us take an instance of \textsc{Subset-Sum-Interval} and construct the following instance of MACH from it.

- For $j = 0, \ldots, r - 1$ we create a so-called padding item $j$ that is owned by the leader. The $j$th padding item has a profit $a_j = 0$ and a weight $c_j = 2^j$.
- For $i = 1, \ldots, k$, we create a so-called ordinary item $i$ that is owned by the
follower. The $ith$ ordinary item has a profit $a'_i = 0$ for the leader and a profit/weight $b'_i = c'_i = q_i$ for the follower.

- There is a magic item 0 owned by the follower, with a profit $a'_0 = 1$ for the leader and a profit/weight $b'_0 = c'_0 = 1/2$ for the follower.
- The knapsack capacity is $C = R + 2^r - 1$.

This completes the construction of the MACH instance. Now let us discuss the possible actions of leader and follower.

The leader decides which of the padding items are to be packed into the knapsack. Note that the overall weight of a subset of padding items can take any value between 0 and $2^r - 1$, and note, furthermore, that padding items bring no profit to the leader. Hence, the decision power of the leader boils down to deciding how much of the knapsack capacity should be consumed by padding items; the remaining knapsack capacity after the leader’s move can be any number between $C - (2^r - 1) = R$ and $C - 0 = R + 2^r - 1$. This means that the leader has essentially the same decision power as in the reduction in section 3.1.

Then the follower has to react. The follower selects some of the ordinary items and possibly the magic item for the knapsack. As these items with their weights and profits are identical to those used in the reduction in section 3.1, the follower also has the same decision power as in the preceding section. Summarizing, we see that the leader and follower both face the same situation as in the proof of Theorem 3.1(a). This completes the proof of Theorem 3.1(b).

3.3. The hardness proof for $\text{DNeg}$. We consider an instance of $\text{Subset-Sum-Interval}$, and we define $Q = \sum_{i=1}^k q_i$. We construct the following instance of $\text{DNeg}$.

- For $j = 0, \ldots, r-1$, we create a padding item $p_j$ with $a(p_j) = 1$ and $b(p_j) = Q + 2^j$.
- For $j = 0, \ldots, r-1$, we create a dummy item $d_j$ with $a(d_j) = 1$ and $b(d_j) = Q$.
- For $i = 1, \ldots, k$, we create an ordinary item $o_i$ with $a(o_i) = r + 1$ and $b(o_i) = q_i$.
- The knapsack capacities are $A = r$ and $B = R + 2^r - 1 + rQ$.

We claim that in the constructed instance of $\text{DNeg}$ the leader can make his objective value $\leq B - 1$ if and only if the $\text{Subset-Sum-Interval}$ instance has the answer YES.

(Proof of if.) Assume that the integer $S$ with $R \leq S < R + 2^r$ cannot be represented as a subset sum of the $q_i$. Then we make the leader pick $r$ items among the padding items and dummy items whose $b$-values add up to a total of $rQ + (S - R)$. How does the follower react to this? We distinguish two cases. First, if the follower does not pick all $r$ remaining padding items and dummy items, then his objective value is at most the $b$-value of the $r$ most valuable padding items plus the $b$-value of all ordinary items; this $b$-value is smaller than $B$. Second, if the follower does pick all $r$ remaining padding items and dummy items, then he picks a total $b$-value of $rQ + (2^r - 1) + (R - S) = B - S$. The remaining capacity in the follower’s knapsack hence equals $S$, and by the definition of $S$ there is no way of filling this remaining capacity with the ordinary items. Hence, the follower’s objective value always remains strictly below $B$.

(Proof of only if.) Now assume that the $\text{Subset-Sum-Interval}$ instance has the answer NO. The leader must pack his knapsack with at most $r$ padding items and dummy items, and he must leave at least $r$ of the padding items and dummy items for the follower. The follower may react as follows. He arbitrarily picks $r$ of the remaining
padding items and dummy items, whose total $b$-value will lie somewhere between $rQ$ (if all of them are dummy items) and $rQ+2^r-1$ (if all of them are padding items). Then the remaining capacity $S$ in the follower’s knapsack lies between $B-(rQ+2^r-1) = R$ and $B-rQ = B+2^r-1$. Since the SUBSET-SUM-INTERVAL instance has the answer NO, there exists a subset of the numbers $q_i$ that adds up to $S$. The follower picks the corresponding ordinary items and fills his knapsack up to its limit $B$. This completes the proof of Theorem 3.1(c).

4. Complexity results under unary encodings. Throughout this section we consider bilevel knapsack problems where the input data is encoded in unary. As the $\Sigma^p_2$-complete problem SUBSET-SUM-INTERVAL from section 3 is solvable in polynomial time under unary encodings (Eggermont and Woeginger [12]), the hardness results in Theorem 3.1 do not carry over to the unary bilevel knapsack versions. We will show that variants DeRi and MACH under unary encodings are solvable in polynomial time, whereas variant DNeg under unary encodings is NP-complete.

4.1. A polynomial time solution for unary-DeRi. We consider the bilevel knapsack variant DeRi in (1a)–(1c). Our main tool is the polynomial time algorithm for the standard knapsack problem under unary encodings; see, for instance, Martello and Toth [20].

The leader simply checks all $O(\delta)$ values $x$ in the interval $C \leq x \leq C'$. For every fixed value of $x$, the optimization problem of the follower is a standard knapsack problem in unary encoding, and hence can be solved in polynomial time. The leader determines the corresponding optimal objective value $V(x)$ of the follower, and then computes the resulting objective value for himself under both the optimistic and pessimistic scenarios; this amounts to solving another standard knapsack problem under unary encoding. In the end the leader chooses the value $x$ that brings him the best objective value.

This result is essentially due to Dempe and Richter [8]. A more sophisticated analysis of the approach yields the time complexity in the following theorem.

**Theorem 4.1** (Brotcorne, Hanafi, and Mansi [1]). The bilevel knapsack problem DeRi in unary encoding can be solved to optimality in polynomial time $O(nC')$, both for the optimistic and the pessimistic scenario.

4.2. A polynomial time solution for unary-MACH. Next, let us turn to the variant MACH in (2a)–(2b). In a preprocessing phase we compute the following auxiliary information; note that the 0-1 variables $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ in these auxiliary problems have the same meaning as in the program (2a)–(2b).

- For $z = 0, \ldots, C$, we determine the maximum value $g(z)$ of $\sum_{j=1}^m a_jx_j$ subject to the constraint $\sum_{j=1}^m c_jx_j = z$.
- For $t = 0, \ldots, C$, we determine the maximum value $h(t)$ of $\sum_{i=1}^n b'_iy_i$ subject to the constraint $\sum_{i=1}^n c'_iy_i \leq t$.
- For $u = 0, \ldots, \sum_{j=1}^m b'_j$ and $v = 0, \ldots, C$, we determine the maximum value $k_{\max}(u,v)$ and the minimum value $k_{\min}(u,v)$ of $\sum_{i=1}^n a'_iy_i$ subject to the constraints $\sum_{j=1}^m b'_jy_i = u$ and $\sum_{i=1}^n c'_iy_i \leq v$.

The computations of the values $g(z)$ and $h(t)$ are again standard knapsack problems under unary encoding, and hence solvable in polynomial time. The computation of the values $k_{\max}(u,v)$ and $k_{\min}(u,v)$ can also be done in polynomial time by routine dynamic programming methods; we omit the straightforward details.

What are the options of the leader? The leader will pack a certain subset of his items into the knapsack, whose overall weight we want to denote by $z := \sum_{j=1}^m c_jx_j$. 


Then the follower is left with a remaining knapsack capacity of $C - z$. The follower will pick an item set that gives him the largest possible personal profit, which by definition equals $h(C - z)$. The follower’s item set gives the leader a resulting profit of $k_{\text{max}}(h(C - z), C - z)$ in the optimistic scenario and a profit of $k_{\text{min}}(h(C - z), C - z)$ in the pessimistic scenario. Summarizing, once the leader has chosen his value of $z$, then his maximum profit in the optimistic scenario equals

\[(4) \quad g(z) + k_{\text{max}}(h(C - z), C - z),\]

whereas his maximum profit in the pessimistic scenario equals

\[(5) \quad g(z) + k_{\text{min}}(h(C - z), C - z).\]

Hence the decision making of the leader boils down to picking a value $z$ from the range $0 \leq z \leq C$ that maximizes the expression in (4), respectively, (5). And as all the data is encoded in unary, this once again can be done in polynomial time. We summarize our findings in the following theorem.

**Theorem 4.2.** The bilevel knapsack problem MACH in unary encoding can be solved in polynomial time, both for the optimistic and the pessimistic scenario.

### 4.3. NP-completeness of unary-DNeg

Our reduction is from the standard VERTEX-COVER problem in undirected graphs; see Garey and Johnson [14].

**Problem: VERTEX-COVER**

Instance: An undirected graph $G = (V, E)$; an integer bound $t$.

Question: Does $G$ possess a vertex cover of size $t$, that is, a subset $T \subseteq V$ such that every edge in $E$ has at least one vertex in $T$?

A **Sidon sequence** is a sequence $s_1 < s_2 < \cdots < s_n$ of positive numbers in which all pairwise sums $s_i + s_j$ with $i < j$ are different. Erdős and Turán [13] showed that for any odd prime $p$, there exists a Sidon sequence of $p$ integers that are all below $2p^2$. The argument in [13] is constructive and yields a simple polynomial time algorithm for finding Sidon sequences of length $n$ whose elements are bounded by $O(n^2)$. For more information on Sidon sequences, the reader is referred to O’Bryant [22].

We start our polynomial time reduction from an arbitrary instance $G = (V, E)$ and $k$ of VERTEX-COVER. Let $n = |V| \geq 10$, and let $v_1, \ldots, v_n$ be an enumeration of the vertices in $V$. We construct a Sidon sequence $s_1 < s_2 < \cdots < s_n$ whose elements are polynomially bounded in $n$. We define $S = \sum_{i=1}^{n} s_i$ as the sum of all numbers in the Sidon sequence, and we construct the following instance of DNeg as specified in (3a)–(3d).

- For every vertex $v_i$, we create a corresponding vertex-item with leader’s weight $a(v_i) = 1$ and follower’s weight $b(v_i) = S + s_i$.
- For every edge $e = [v_i, v_j]$, we create a corresponding edge-item with leader’s weight $a(e) = t + 1$ and follower’s weight $b(e) = 5S - s_i - s_j$.
- The capacity of the leader’s knapsack is $A = t$, and the capacity of the follower’s knapsack is $B = 7S$.

We claim that in the constructed instance of DNeg the leader can make his objective value $\leq 7S - 1$ if and only if the VERTEX-COVER instance has the answer YES.

(Proof of if.) Assume that there exists a vertex cover $T$ of size $|T| = t$. Then a good strategy for the leader is to put the $t$ vertex-items that correspond to vertices in $T$ into his knapsack, which fills his knapsack of capacity $A = t$ to the limit. Suppose, for the sake of contradiction, that afterwards the follower can still fill his knapsack with a total weight $7S$. Then the follower must pick at least one edge-item (he can pack at
most six vertex-items, and their weight would stay strictly below 7S). Furthermore, the follower cannot pick two edge-items (since every edge-item has weight greater than 4S). Consequently, the follower must pick exactly one edge-item that corresponds to some edge e = [vi, vj]. The remaining space in the follower’s knapsack is 2S + si + sj and must be filled by two vertex-items. By the definition of a Sidon sequence, the only way of doing this would be by picking the two vertex-items corresponding to vi and vj. But that’s impossible, as at least one of the vertices vi and vj is in the cover T so that the item has already been picked by the leader. This contradiction shows that the follower cannot reach an objective value of 7S.

(The proof of only if.) Now let us assume that the graph G does not possess any vertex cover of size t, and let us consider the game right after the move of the leader. Since the leader can pack at most t vertex-items, there must exist some edge e = [vi, vj] in E for which the leader has neither picked the item corresponding to vi nor the item corresponding to vj. Then the follower may pick the vertex-item vi, the vertex-item vj, and the edge-item e, which brings him a total weight of 7S.

Theorem 4.3. The decision version of the bilevel knapsack problem DNeg in unary encoding is NP-complete, both for the optimistic scenario and the pessimistic scenario.

Proof. The above construction can be performed in polynomial time. As the elements in the Sidon sequence are polynomially bounded in |V|, their sum S and all the integers in our construction are also polynomially bounded in |V|. In particular, this yields that the unary encoding length of the constructed DNeg instance is polynomially bounded in |V|. Together with the above arguments, this implies that DNeg in unary encoding is NP-hard.

It remains to show that DNeg in unary encoding is contained in NP. We use the optimal move of the leader as NP-certificate. This certificate is short, as it just specifies a subset of the items. To verify the certificate, we have to check that the follower cannot pick any item set of high weight. Since all weights are encoded in unary, this checking amounts to solving a standard knapsack problem in unary encoding, which can be done in polynomial time.

5. Approximability and inapproximability. Our \(\Sigma_p^2\)-completeness proofs in sections 3.1 and 3.2 have devastating consequences for the polynomial time approximation of problems DeRi and MACH. Recall that our reduction for problem DeRi yields the following: it is \(\Sigma_p^2\)-hard to distinguish the DeRi instances in which the leader can reach an objective value of 1 from those DeRi instances in which the leader can only reach objective value 0. An analogous statement holds for problem MACH. As a polynomial time approximation algorithm with finite worst case guarantee would be able to distinguish between these two instance types, we get the following result.

Corollary 5.1. Problems DeRi and MACH do not possess a polynomial time approximation algorithm with finite worst case guarantee, unless \(P = \Sigma^P_2\) holds (which is equivalent to \(P=\Sigma^P_2\)).

The statement in Corollary 5.1 is not surprising at all: the literature on the approximability of \(\Sigma^P_2\)-hard optimization problems consists entirely of such negative statements that show the inapproximability of various problems; see Ko and Lin [17] and Umans [28]. The following theorem breaks with this old tradition and presents the first approximation scheme for a \(\Sigma^P_2\)-hard optimization problem in the history of approximation algorithms.

Theorem 5.2. Problem DNeg has a polynomial time approximation scheme.

The rest of this section is dedicated to the proof of Theorem 5.2. We apply and
extend a number of rounding tricks from the seminal paper [18] by Lawler, we use approximation schemes from the literature as a black box, and we also add a number of new ingredients and rounding tricks.

Throughout the proof we will consider a fixed instance of problem DNeg. Without loss of generality, we assume that no item \( i \) in the instance satisfies \( b_i > B \): such items could never be used by the follower, and hence are irrelevant and may as well be ignored. Let \( \varepsilon \) with \( 0 < \varepsilon < 1/3 \) be a small positive real number; for the sake of simplicity we will assume that the reciprocal value \( 1/\varepsilon \) is an integer.

Our global goal is to determine in polynomial time a feasible solution for the leader that yields an objective value of at most \((1 + \varepsilon)^4\) times the optimum. This will be done by a binary search over the range \( 0, 1, \ldots, B \) that (approximately) sandwiches the optimal objective value between a lower and an upper bound. Whenever we bisect the search interval between these bounds at some value \( U \), we have to decide whether the optimal objective value lies below or above \( U \). If the optimal objective value lies below \( U \), then Lemmas 5.4 (derived in section 5.1) and 5.5 (derived in section 5.2) show how to find and how to verify in polynomial time an approximate solution for the leader whose objective value is bounded by \((1 + \varepsilon)^3U\). If these lemmas succeed, then we make \( U \) the new upper bound. If the lemmas fail to produce an approximate objective value of at most \((1 + \varepsilon)^3U\), then we make \( U \) the new lower bound. The binary search process terminates as soon as the upper bound comes within a factor of \( 1 + \varepsilon \) of the lower bound. Note that we then lose a factor of \( 1 + \varepsilon \) between upper and lower bound, and that we lose a factor of at most \((1 + \varepsilon)^3\) by applying the lemmas. All in all, this yields the desired approximation guarantee of \((1 + \varepsilon)^4\) and completes the proof of Theorem 5.2.

5.1. How do handle the central cases. Throughout this section, we assume that \( U \) is an upper bound on the optimal objective value with

\[
B/2 \leq U \leq B/(1 + \varepsilon).
\]

The items \( i = 1, \ldots, n \) are partitioned according to their \( b \)-values into so-called large items that satisfy \( U < b_i \) into medium items that satisfy \( \varepsilon U < b_i \leq U \), and into small items that satisfy \( b_i \leq \varepsilon U \). We denote by \( L, M, S \), respectively, the set of large, medium, small items. Furthermore, a medium item \( i \) belongs to class \( C_k \) if it satisfies

\[
k \varepsilon^2 U \leq b_i < (k + 1) \varepsilon^2 U.
\]

Note that only classes \( C_k \) with \( 1/\varepsilon \leq k \leq 1/\varepsilon^2 \) play a role in this classification. By (6) the overall size of \( 2/\varepsilon \) medium items exceeds the capacity of the follower’s knapsack. Hence, the follower can fit at most \( 2/\varepsilon \) medium items into his knapsack.

In the following we will analyze two scenarios. In the first scenario, the solution \( x^* \) used by the leader and the solution \( y^* \) for the follower both will carry a superscript *. The sets of large, medium, small items packed by \( x^* \) into the leader’s knapsack will be denoted, respectively, by \( L_x^*, M_x^*, S_x^* \), and the corresponding sets for \( y^* \) and the follower will be denoted \( L_y^*, M_y^*, S_y^* \). In the second scenario we use analogous notations with the superscript #. The first scenario is centered around an optimal solution \( x^* \) for the leader. The second scenario considers another feasible solution \( x^\# \) for the leader that we call the aligned version of \( x^* \).

- Solution \( x^\# \) packs all large items into the knapsack; hence \( L_x^\# = L \).
- Solution \( x^\# \) packs the following medium items from class \( C_k \) (note that \( M_x^\# \subseteq M_x^* \)):
(i) If \(|C_k - M^*_x| \leq 2/\varepsilon\), then solution \(x^\#\) packs all items in \(M^*_x \cap C_k\).
(ii) If \(|C_k - M^*_x| > 2/\varepsilon\), then \(x^\#\) packs an item \(i \in M^*_x \cap C_k\) if and only if there are at most \(2/\varepsilon\) items \(j \in C_k - M^*_x\) with smaller \(b\)-value \(b_j < b_i\). (By this choice, the \(2/\varepsilon\) items with smallest \(b\)-value in \(C_k - M^*_x\) coincide with the \(2/\varepsilon\) items with smallest \(b\)-value in \(C_k - M^*_x\).

- For the small items we first determine a \((1 + \varepsilon)\)-approximate solution to the following auxiliary problem (Aux): find a subset \(Z \subseteq S\) of the small items that minimizes \(b(Z)\), subject to the covering constraint \(a(Z) \geq a(L^*_x \cup M^*_x) + a(S) - A\). Solution \(x^\#\) then packs the complementary set \(S^*_x = S - Z\).

This completes the description of \(x^\#\), which is easily seen to be a feasible action for the leader. Note that also the optimal solution \(x^*\) packs all the large items, as otherwise the follower may pack a large item and push the objective value above the bound \(U\).

Then \(L^*_x = L^*_x\) and \(M^*_x \subseteq M^*_x\) imply \(a(L^*_x \cup M^*_x) \geq a(L^*_x \cup M^*_x)\), which yields

\[
A \geq a(L^*_x \cup M^*_x \cup S^*_x) \geq a(L^*_x \cup M^*_x) + a(S^*_x).
\]

As \(a(S^*_x) = a(S) - a(S - S^*_x)\), we conclude from (7) that the set \(S - S^*_x\) satisfies the covering constraint in the auxiliary problem (Aux). Hence, the optimal objective value of (Aux) is upper bounded by \(b(S - S^*_x)\), and any \((1 + \varepsilon)\)-approximate solution \(Z\) to (Aux) must satisfy \(b(Z) \leq (1 + \varepsilon)b(S - S^*_x)\), which is equivalent to

\[
b(S - S^*_x) \leq (1 + \varepsilon)b(S - S^*_x).
\]

The following lemma demonstrates that the aligned solution \(x^\#\) is almost as good for the leader as the underlying optimal solution \(x^*\).

**Lemma 5.3.** Given an optimal solution \((x^*, y^*)\) with \(f_3(x^*, y^*) \leq U\), let \(x^\#\) be the solution aligned to \(x^*\). If the leader uses \(x^\#\), then every feasible reaction \(y^\#\) for the follower yields an objective value \(f_3(x^\#, y^\#) \leq (1 + 2\varepsilon)U\).

**Proof.** Suppose, for the sake of contradiction, that there exists a reaction \(y^\#\) for the follower that yields an objective value of \(f_3(x^\#, y^\#) > (1 + 2\varepsilon)U\). Based on \(y^\#\) we will construct another solution \(y^*\) for the follower:

- Solution \(y^*\) does not use any large item; hence \(L^*_y = \emptyset\).
- Solution \(y^*\) picks the same number of items from every class \(C_k\) as \(y^\#\) does. It avoids items in \(x^\#\) and selects the \(|C_k \cap M^*_y|\) items in \(C_k - M^*_x\) that have the smallest \(b\)-values.
- Finally, we add small items from \(S - S^*_x\) to the follower’s knapsack, until no further item fits or until we run out of items.

Solution \(y^\#\) packs at most \(2/\varepsilon\) medium items, and hence uses at most \(2/\varepsilon\) items from \(C_k\). By our choice of medium items for \(x^\#\) we derive \(b(C_k \cap M^*_y) \leq b(C_k \cap M^*_y)\) for every \(k\), which implies

\[
b(M^*_y) \leq b(M^*_y) \leq B.
\]

Solution \(y^\#\) selects only items that are not used by \(x^\#\), and inequality (9) implies that all the selected items indeed fit into the follower’s knapsack. Hence, \(y^*\) constitutes a feasible reaction of the follower if the leader chooses \(x^*\).

Next, let us quickly go through the item types. First of all, neither solution \(y^*\) nor solution \(y^\#\) can use any large item, so that we have

\[
b(L^*_y) = b(L^*_y) = 0.
\]
For the medium items, the ratio between the smallest $b$-value and the largest $b$-value in class $C_k$ is at least $b/(k+1) \geq 1 - \varepsilon$. Hence, we certainly have $b(C_k \cap M_y^*) \geq (1 - \varepsilon) b(C_k \cap M_y^*)$, which implies

$$b(M_y^*) \geq (1 - \varepsilon) b(M_y^*).$$

(11)

Let us turn to the small items. Suppose that $y^*$ cannot accommodate all small items from $S - S_y^*$ in the follower’s knapsack. Then some small item $i$ with $b_i < \varepsilon U$ does not fit, which with (6) leads to $b(y^*) > B - \varepsilon U \geq U$. As this violates our upper bound $U$ on the optimal objective value, we conclude that $y^*$ accommodates all such items and satisfies $S_y^* = S - S_y^*$. This relation, together with (8), and the disjointness of the sets $S_y^*$ and $S_y^*$ yields

$$b(S_y^*) = b(S - S_y^*) \geq \frac{b(S - S_y^*)}{1 + \varepsilon} \geq \frac{b(S_y^*)}{1 + \varepsilon} > (1 - \varepsilon) b(S_y^*).$$

(12)

Now let us wrap things up. If the leader chooses $x^*$, the follower may react with the feasible solution $y^*$ and get an objective value

$$f_3(x^*, y^*) = b(L_y^*) + b(M_y^*) + b(S_y^*)$$

$$> (1 - \varepsilon) b(L_y^*) + (1 - \varepsilon) b(M_y^*) + (1 - \varepsilon) b(S_y^*)$$

$$= (1 - \varepsilon) f_3(x, y^*) > (1 - \varepsilon)(1 + 2\varepsilon)U > U.$$

Here we used the estimates in (10), (11), and (12). As this objective value violates the upper bound $U$, we have reached the desired contradiction.

**Lemma 5.4.** Given an upper bound $U$ on the objective value that satisfies (6), one can compute in polynomial time a feasible solution $x$ for the leader, such that every reaction $y$ of the follower has $f_3(x, y) \leq (1 + \varepsilon)^3 U$.

**Proof.** If we did not only know the bound $U$ but also an optimal solution $x^*$, then we could simply determine the corresponding aligned solution $x^*$ and apply Lemma 5.3. We will bypass this lack of knowledge by checking many candidates for the set $M_x^*$.

Let us recall how the aligned solution $x^*$ picks medium items from class $C_k$.

- If $|C_k - M_y^*| \leq 2/\varepsilon$, then $M_y^* \cap C_k = M_y^* \cap C_k$. Note that there are only $O(|C_k|^{2/\varepsilon})$ different candidates for $M_y^* \cap C_k$.
- If $|C_k - M_y^*| > 2/\varepsilon$, then $M_y^* \cap C_k$ is a subset of $M_y^*$; an item $i$ from $M_y^* \cap C_k$ enters $M_y^*$ if there are at most $2/\varepsilon$ items $j \in C_k - M_y^*$ with $b_j \leq b_i$. Note that $M_y^* \cap C_k$ is fully determined by the $2/\varepsilon$ items with smallest $b$-value in $C_k - M_y^*$. As there are only $O(|C_k|^{2/\varepsilon})$ ways for choosing these $2/\varepsilon$ items, there are only $O(|C_k|^{2/\varepsilon})$ different candidates for $M_y^* \cap C_k$.

 Altogether there are only $O(|C_k|^{2/\varepsilon})$ ways of picking the medium items from class $C_k$. As every class satisfies $|C_k| \leq n$ and as there are only $1/\varepsilon^2$ classes to consider, we get a polynomial number $O(n^{2/\varepsilon})$ of possibilities for choosing the set $M_y^*$ in the aligned solution. Summarizing, we need only check a polynomial number of candidates for set $M_y^*$.

How do we check such a candidate $M_y^*$? The aligned solution always uses $L_x^* = L$, and the auxiliary problem (Aux) is fully determined once $M_y^*$ and $L_x^*$ have been fixed. We approximate the auxiliary problem by standard methods (see, for instance, Pruhs and Woeginger [25]), and thus also find the set $S_y^*$ in polynomial time. This yields the full corresponding aligned solution $x^*$. It remains to verify the quality of this aligned
solution for the leader, which amounts to analyzing the resulting knapsack problem at the follower’s level. We use one of the standard approximation schemes for knapsack as, for instance, described by Lawler [18], and thereby get a \((1 + \varepsilon)\)-approximate solution for the follower’s problem.

While checking and scanning through the candidates, we eventually must hit a good candidate \(M^*\) that yields the correct aligned version \(x\) of an optimal solution. By Lemma 5.3 the corresponding objective value \(f_3(x, y)\) is bounded by \((1 + 2\varepsilon)U\). Then the approximation scheme finds an objective value of at most \((1 + \varepsilon)(1 + 2\varepsilon)U \leq (1 + \varepsilon)^3U\). This completes the proof of the lemma.

5.2. How do handle the boundary cases. Finally, let us discuss the remaining cases where \(U\) does not satisfy the bounds in (6). The first case \(U > B/(1 + \varepsilon)\) is trivial, as the objective value never exceeds the follower’s knapsack capacity \(B\); hence in this case the objective value will always stay below \((1 + \varepsilon)U\). The second case \(U < B/2\) is settled by the following lemma.

**Lemma 5.5.** Given an upper bound \(U < B/2\) on the objective value, one can compute in polynomial time a feasible solution \(x\) for the leader, such that every reaction \(y\) of the follower has \(f_3(x, y) \leq (1 + \varepsilon)U\).

**Proof.** If the objective value is below \(B/2\), then the leader must pick all items \(i\) with \(b_i \geq B/2\); otherwise the follower could pick one and push the objective value to \(B/2\) or more. Once the leader has chosen his solution \(x\), all remaining items will fit into the follower’s knapsack: the follower’s knapsack has free capacity of at least \(B - U > B/2\), and hence every item \(i\) with \(b_i < B/2\) will fit there.

With these observations, the goal of the leader boils down to the following: partition the item set into two parts \(Z_\ell\) and \(Z_f\) such that the value \(b(Z_f)\) is minimized subject to the condition that the items in \(Z_\ell\) altogether fit into the leader’s knapsack. This minimization problem belongs to the class of subset selection problems studied by Pruhs and Woeginger [25]: determine a subset \(Z_f\) of items that has minimum cost \(b(Z_f)\) subject to the feasibility constraint that the total size of all items outside \(Z_f\) is at most the size of the leader’s knapsack. This subset selection problem can be solved in pseudopolynomial time by routine dynamic programming; the resulting time complexity is bounded in \(B\), in \(n\), and in the logarithm of \(A\). With this, Theorem 1.2 in [25] yields the existence of an approximation scheme for the subset selection problem. Hence, there is an approximation scheme which yields the desired solution \(x\) for the leader.

6. Conclusions. We have analyzed the computational complexity of three bilevel knapsack problems from the literature. All three problems DeRi, MACH, DNeg turn out to be \(\Sigma_3^p\)-complete under the standard binary encoding of the input. Our results provide strong evidence that bilevel knapsack problems cannot be formulated as a classical single-level integer program of polynomial size; otherwise the entire polynomial hierarchy would collapse to its first level, which is considered to be extremely unlikely in the area of computational complexity theory. Furthermore, we have settled the complexity of these three bilevel knapsack problems under \textit{unary} encodings of the input: unary-DeRi and unary-MACH are polynomially solvable, whereas unary-DNeg is NP-complete. Finally, we studied the approximability of the three problems. DeRi and MACH turned out to be inapproximable, whereas DNeg has a polynomial time approximation scheme.

Our investigations provide a complete and clean picture of the complexity landscape of the considered bilevel knapsack problems. We expect that our results will also be useful in classifying and understanding other bilevel problems, and that our
hardness proofs will serve as stepping stones for future results. As an open problem, we propose to study the approximability of the bilevel knapsack variant introduced by Chen and Zhang [4].

REFERENCES


