

A tight $1/3$ -approximation algorithm and fully polynomial-time approximation schemes for the Colored Knapsack Problem

Fabio Ciccarelli^{a,*}, Fabio Furini^a

^a*Department of Computer, Control and Management Engineering Antonio Ruberti, Sapienza University of Rome, Via Ariosto 25, 00185 Rome, Italy,*

Abstract

The *Colored Knapsack Problem* (ColKP) generalizes the classical Knapsack Problem by partitioning the items into color classes and requiring the selected items to admit an ordering in which consecutive items have different colors. The problem is weakly $\mathcal{NP}$ -hard and admits two pseudo-polynomial dynamic programming (DP) algorithms proposed in the literature. These two DP algorithms have worst-case running times $O(bn^4)$ and $O(b^2n^3)$, respectively, where b is the knapsack capacity and n is the number of items. We develop the first approximation algorithm for the ColKP. By rounding an optimal basic solution of the linear programming relaxation of its natural integer programming formulation and repairing color feasibility, we obtain a linear-time approximation-algorithm whose worst-case performance ratio is $1/3$. We then reformulate both DP algorithms so that profit, rather than knapsack capacity, indexes their pseudo-polynomial dimension, and combine them with profit scaling to obtain two fully polynomial-time approximation schemes (FPTASs). The first FPTAS runs in $O(n^5/\varepsilon)$ time for nonnegative profits and in $O(n^6/\varepsilon)$ time for arbitrary integer profits. The second FPTAS runs instead in $O(n^5/\varepsilon^2)$ and $O(n^7/\varepsilon^2)$ time, respectively. The approximation guarantee, along with new structural insights, provides the bounds needed to control the scaled profit range and establish these running times.

Keywords: knapsack problem, approximation algorithms, fully polynomial-time approximation schemes, dynamic programming

1. Introduction

Let $\mathcal{I} = \{1, 2, \dots, n\}$ be a set of n items and let $b \in \mathbb{Z}_{>0}$ be the capacity of a knapsack. Each item $i \in \mathcal{I}$ has a weight $w_i \in \mathbb{Z}_{>0}$, a profit $p_i \in \mathbb{Z}$, and a color $\kappa_i \in \mathcal{C} = \{1, 2, \dots, m\}$. For a subset $S \subseteq \mathcal{I}$, we denote its total weight and profit by

$$w(S) = \sum_{i \in S} w_i \quad \text{and} \quad p(S) = \sum_{i \in S} p_i,$$

respectively. For each color $c \in \mathcal{C}$, we further define

$$\mathcal{I}_c = \{i \in \mathcal{I} : \kappa_i = c\} \quad \text{and} \quad k_c(S) = |S \cap \mathcal{I}_c|,$$

*Corresponding author.

Email addresses: f.ciccarelli@uniroma1.it (Fabio Ciccarelli), fabio.furini@uniroma1.it (Fabio Furini)

that is, the set of items of color c and the number of such items in S .

The *Colored Knapsack Problem* (ColKP) asks for a maximum-profit subset of items $S \subseteq \mathcal{I}$ with $w(S) \leq b$ that can be ordered so that no two consecutive items share the same color. As shown by Borges et al. [3], such an ordering exists if and only if, for every color, its number of selected items is at most one greater than the number of selected items of all other colors, i.e., if and only if

$$k_c(S) \leq |S \setminus \mathcal{I}_c| + 1, \quad c \in \mathcal{C}. \quad (1.1)$$

For a subset $S \subseteq \mathcal{I}$, let

$$d(S) = \max\{k_c(S) : c \in \mathcal{C}\}.$$

A color $c \in \mathcal{C}$ is *dominant* in S if $k_c(S) = d(S)$, that is, if it has the largest number of selected items. Condition (1.1) is then equivalent to the single inequality

$$2d(S) \leq |S| + 1, \quad (1.2)$$

which states that the number of items of a dominant color cannot exceed the number of all selected items by more than one. Accordingly, the ColKP can be stated compactly as

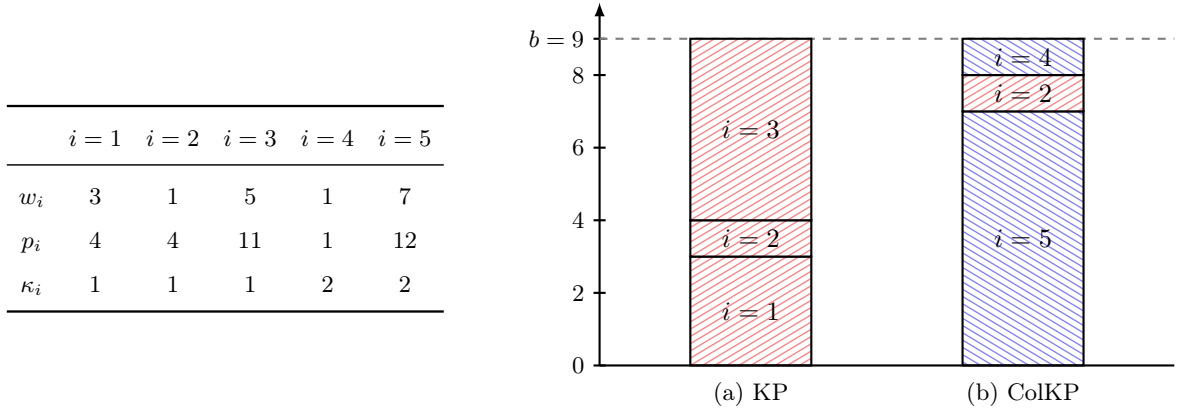
$$\text{OPT} = \max\{p(S) : S \subseteq \mathcal{I}, w(S) \leq b, 2d(S) \leq |S| + 1\}. \quad (1.3)$$

We call a subset S satisfying (1.2) *color-feasible*, and *feasible* if in addition $w(S) \leq b$. A dominant color c is *critical* for S if $k_c(S) = |S \setminus \mathcal{I}_c| + 1$. Observe that at most one color can be critical for a given feasible ColKP solution S .

Two features distinguish the ColKP from the classical Knapsack Problem (KP) and are essential to our analysis. First, its family of feasible sets is not closed under taking subsets: removing an item may destroy color feasibility. Consequently, items with zero or even negative profit may belong to an optimal solution, because they can act as *separators* that allow additional profitable items of a dominant color to be selected. We therefore formulate the ColKP with integer profits $p_i \in \mathbb{Z}$ for all $i \in \mathcal{I}$, rather than restricting profits to be positive.

Example 1.1 illustrates a ColKP instance with $n = 5$ items, $m = 2$ colors, and capacity $b = 9$. If colors are ignored, $\{1, 2, 3\}$ is an optimal KP solution with profit 19. Nonetheless, it contains three items of color 1 and none of color 2, and is therefore infeasible for the ColKP, since $2d = 6 > |S| + 1 = 4$. $S^* = \{2, 4, 5\}$ is instead an optimal ColKP solution with profit $\text{OPT} = 17$. Color 2 is its unique dominant color, with $d(S^*) = 2$, and it is also critical because $2d(S^*) = |S^*| + 1$.

A two-color version of the ColKP was studied as a subproblem of the Black and White Bin Packing Problem (see Balogh et al. [1, 2]). Nonetheless, the results obtained for the two-color case do not extend to the m -color case. The general variant was instead introduced by Ciccarelli et al. [11]. To the best of our knowledge, their paper is the only previous work devoted specifically to this problem. They proposed a natural integer linear programming (ILP) formulation and studied its linear programming (LP) relaxation, proving that an optimal basic LP solution has at most two fractional items and can be computed in $O(n)$ time. They also established that the ColKP is weakly $\mathcal{NP}$ -hard and developed two exact pseudo-polynomial dynamic programming (DP) algorithms. The first processes the items one at a time and runs in $O(bn^4)$ time, whereas the second processes one color class at a time and runs in $O(b^2n^3)$ time. They derived dominance and fathoming rules for both algorithms and compared their implementations computationally with off-the-shelf solvers applied to the natural ColKP ILP formulation.



Example 1.1: ColKP instance with $n = 5$ items, $m = 2$ colors and capacity $b = 9$. The item weights, profits, and colors are given in the table on the left. The figure on the right illustrates an optimal KP solution and an optimal ColKP solution to the instance. The vertical axis represents consumed capacity, and the height of each rectangle is the corresponding item weight. Red and blue hatching denote colors 1 and 2, respectively. Ignoring colors, the KP optimum in (a) selects items 1, 2, and 3. The ColKP optimum in (b) selects items 5, 2, and 4, thus respecting the color constraints.

That study settled the basic complexity status of the ColKP and provided pseudo-polynomial exact algorithms, but left its approximability open: indeed, neither a polynomial-time approximation algorithm with a provable performance guarantee nor an approximation scheme was proposed. The ColKP inherits weak $\mathcal{NP}$ -hardness from the classical KP, since a KP instance can be viewed as a ColKP instance where each item has a distinct color. However, the two pseudo-polynomial DPs proposed in [11] do not immediately yield an approximation scheme, nor is it clear which of the two recursions is the more suitable starting point to develop such an algorithm. This paper addresses both issues: it gives the first approximation algorithm for the ColKP, to the best of our knowledge, and derives a fully polynomial-time approximation scheme from each of the two DPs of [11].

1.1. Approximation algorithms and approximation schemes for knapsack problems

In this section, we recall the definitions and results on approximation algorithms and approximation schemes that will be used throughout the paper. We also review the classical scaling argument underlying the fully polynomial-time approximation schemes for the KP and several of its variants, which we later adapt to the ColKP in Sections 4 and 5.

We first recall the performance measures used throughout the paper. Let $\mathcal{A}$ be an algorithm for a maximization problem. For an instance I , let $\mathcal{A}(I)$ be the value of the feasible solution it returns and let $\text{OPT}(I) > 0$ be the instance optimum. We assume that $0 \leq \mathcal{A}(I) \leq \text{OPT}(I)$. The *absolute worst-case performance ratio* of $\mathcal{A}$ is

$$R_{\mathcal{A}} := \inf_I \left\{ \frac{\mathcal{A}(I)}{\text{OPT}(I)} \right\} \in [0, 1]. \quad (1.4)$$

If $\mathcal{A}$ runs in polynomial time and $R_{\mathcal{A}} \geq \alpha$ for some $\alpha \in (0, 1]$, then $\mathcal{A}$ is an α -*approximation algorithm*. A guarantee α is *tight* when $R_{\mathcal{A}} = \alpha$.

A *polynomial-time approximation scheme* (PTAS) is a family of algorithms $\{\mathcal{A}_{\varepsilon}\}_{\varepsilon \in (0, 1)}$ such that

$$\mathcal{A}_{\varepsilon}(I) \geq (1 - \varepsilon) \text{OPT}(I) \quad (1.5)$$

for every instance I , and such that, for every fixed ε , the running time of $\mathcal{A}_\varepsilon$ is polynomial in the input size. A *fully polynomial-time approximation scheme* (FPTAS) is a PTAS whose running time is polynomial both in the input size and in $1/\varepsilon$.

The classical KP is the canonical example of a weakly $\mathcal{NP}$ -hard problem admitting an FPTAS (see Kellerer et al. [20] for a detailed overview). The standard scaling-based scheme starts from the pseudo-polynomial dynamic program indexed by profit. Let

$$W_i(z) := \min\{w(S) : S \subseteq \{1, 2, \dots, i\}, p(S) = z\}$$

be the minimum weight of a subset of the first i items with profit exactly z . In case no such subset exists, we set $W_i(z) = +\infty$. The optimum is the largest profit z for which $W_n(z) \leq b$. Let $p_{\max} = \max\{p_i : i \in \mathcal{I}\}$ and set

$$\lambda = \frac{\varepsilon p_{\max}}{n} \quad \text{and} \quad \hat{p}_i = \left\lfloor \frac{p_i}{\lambda} \right\rfloor, \quad i \in \mathcal{I}. \quad (1.6)$$

For $S \subseteq \mathcal{I}$, write $\hat{p}(S) = \sum_{i \in S} \hat{p}_i$. Running the same recursion on the scaled profits makes its profit range polynomially bounded, since $\sum_{i \in \mathcal{I}} \hat{p}_i \leq n^2/\varepsilon$. If S_ε is optimal for the scaled instance and S^* is optimal for the original one, then

$$\begin{aligned} p(S_\varepsilon) &\geq \lambda \hat{p}(S_\varepsilon) \geq \lambda \hat{p}(S^*) \\ &\geq p(S^*) - n\lambda \geq (1 - \varepsilon) \text{OPT}, \end{aligned} \quad (1.7)$$

where the last inequality uses $p_{\max} \leq \text{OPT}$. Scaling therefore trades an additive loss of less than λ per selected item for a profit-indexed table of polynomial size. This is the basic idea that we adapt to the ColKP in Sections 4 and 5.

Sahni [30] gave the first PTAS for the KP, based on the enumeration of a bounded number of high-profit items. The first published FPTAS is instead due to Ibarra and Kim [16]. Lawler [21] refined that scheme and improved its complexity bounds, and Magazine and Oguz [22] proposed a further dynamic-programming-based scheme. Later work continued improving the running time and, in some cases, the space requirements [18, 19, 8]. A recent sequence of improvements [17, 13] led to the $\tilde{O}(n + 1/\varepsilon^2)$ -time FPTAS of Chen et al. [9].

FPTASs are also known for several variants of the KP; for a broader survey, see Cacchiani et al. [4]. Binary and integer versions of the *Multiple-Choice Knapsack Problem* were studied from an approximation perspective by Chandra et al. [7]. In their binary version, the items are partitioned into classes and at most one item is selected from each class. The integer version permits instead multiple copies of the selected item type. An early exact study of the exact-one binary variant, including two branch-and-bound algorithms, is due to Nauss [26], while Lawler [21] proposed an FPTAS for the multiple-choice problem. For the k -item KP, which imposes an upper bound on the cardinality of the selected set, Caprara et al. [5] gave a pseudo-polynomial dynamic program, a PTAS, and an FPTAS. More involved side constraints require problem-specific arguments. The *Knapsack Problem with Conflicts* is strongly $\mathcal{NP}$ -hard on general conflict graphs, whereas Pferschy and Schauer [27] derived pseudo-polynomial dynamic programs, and hence FPTASs, for bounded-treewidth and chordal conflict graphs. A more recent exact branch-and-bound algorithm is given by Coniglio et al. [12]. The *Knapsack Problem with Setup* was studied by Chajakis and Guignard [6], who proposed exact algorithms, including a pseudo-polynomial DP. Chebil and Khemakhem [10] developed a further DP, while Furini et al. [15] embedded tailored LP-relaxation algorithms in an

implicit-enumeration scheme and ran it in parallel with an improved DP. Moreover, Pferschy and Scatamacchia [28] proved an inapproximability result for the general problem and derived FPTASs for three relevant special cases. Finally, D’Ambrosio et al. [14] introduced the *Product Knapsack Problem*, whose objective is to maximize the product of the selected item profits, which may be positive or negative, and proposed mixed-integer linear and nonlinear formulations and a pseudo-polynomial exact DP algorithm. Pferschy et al. [29] subsequently gave the first FPTAS for this problem.

1.2. Contributions and outline of the paper

Section 2 first establishes the structural properties needed to handle arbitrary integer profits. It shows that, for any ColKP instance, items with profit at most $-p_{\max}$ can be removed in linear time without changing the optimal value, so that all relevant profits belong to the $(-p_{\max}, p_{\max}]$ interval. It also proves that an optimal solution containing a negative-profit item must have a unique critical color and can contain at most $(n-1)/2$ negative-profit items. These properties make it possible to bound the signed profit ranges used later.

Section 3 then develops the first approximation algorithm for the ColKP. Starting from an optimal basic solution of the natural ColKP formulation LP relaxation, we round down its fractional variables (at most two) and, when necessary, repair color feasibility by removing one item. Comparing the resulting set with the most profitable singleton yields a $1/3$ -approximation in $O(n)$ time. We also show that the absolute worst-case performance ratio of this algorithm is exactly $1/3$. Besides being a result of independent interest, the algorithm provides lower and upper bounds on OPT that differ by at most a factor of three and are therefore crucial for controlling the scaled profit range.

Sections 4 and 5 build the main approximation scheme. We first reformulate the item-by-item dynamic program DP_1 of [11] by exchanging the roles of capacity and profit. We then combine this profit-indexed recursion with scaling and the aforementioned bounds to obtain an FPTAS that runs in $O(n^5/\varepsilon)$ time for nonnegative profits and in $O(n^6/\varepsilon)$ time for arbitrary integer profits.

Section 6 investigates whether the alternative color-by-color recursion DP_2 of [11] can yield a faster approximationscheme. Its profit-indexed reformulation also leads to an FPTAS, but the resulting running times are $O(n^5/\varepsilon^2)$ for nonnegative profits and $O(n^7/\varepsilon^2)$ for arbitrary integer profits, showing that the item-by-item recursion provides a better basis for the direct scaling approach developed here.

Finally, Section 7 summarizes the results and outlines directions for future research.

2. Structural properties of ColKP solutions

Throughout this section, and in the rest of the paper, we let

$$p_{\max} = \max \{p_i : i \in \mathcal{I}\} \quad (2.1)$$

denote the largest item profit. Moreover, in the remainder of the paper we assume without loss of generality that $w_i \leq b$ for all $i \in \mathcal{I}$.

Observation 2.1. The empty set $S = \emptyset$ is a feasible solution to the ColKP, and so is every singleton $S = \{i\}$ with $i \in \mathcal{I}$, because $|S| = d(S) = 1$ satisfies (1.2) and $w_i \leq b$ for all $i \in \mathcal{I}$. Consequently

$$\text{OPT} \geq p_{\max} \geq 0. \quad (2.2)$$

Moreover, if $p_{\max} \leq 0$ then the empty set is optimal and problem (1.3) is solved in $O(n)$ time. We therefore assume $p_{\max} > 0$ from now on.

Observation 2.1 is the reason why an FPTAS is meaningful at all for a problem with possibly negative profits: the optimal value is guaranteed to be nonnegative, and in fact bounded from below by the largest item profit, which is exactly the quantity that drives the classical scaling argument for the classical KP.

The next lemma describes two ways in which a feasible ColKP solution can be modified without losing color feasibility. It is used repeatedly in the rest of the paper.

Lemma 2.2. *Let S be a feasible ColKP solution, let $t = |S|$ and $d = d(S)$.*

- (a) *If $i \in S$ belongs to a dominant color of S , then $S \setminus \{i\}$ is color-feasible.*
- (b) *If $i \in S$ belongs to a dominant color of S and $j \in S$ belongs to a color that is not dominant in S , then $S \setminus \{i, j\}$ is color-feasible.*

Proof. (a) Let $c = \kappa_i$, so that $k_c(S) = d$, and let $S' = S \setminus \{i\}$, with $|S'| = t - 1$. If some color $c' \neq c$ also satisfies $k_{c'}(S) = d$, then $t \geq 2d$ and $d(S') = d$, so that $2d(S') = 2d \leq t = |S'| + 1$. Otherwise c is the unique dominant color of S , every other color appears at most $d - 1$ times, and $d(S') = d - 1$. From feasibility of S , we have that $2d \leq t + 1$, hence $2d(S') = 2d - 2 \leq t - 1 = |S'|$.

(b) Let $S' = S \setminus \{i, j\}$, with $|S'| = t - 2$, and let $c = \kappa_i$. If some color $c'' \neq c$ satisfies $k_{c''}(S) = d$, then S contains d items of color c , d items of color c'' and the item j , whose color is not dominant by assumption, hence $t \geq 2d + 1$. Since $d(S') \leq d$ we obtain $2d(S') \leq 2d \leq t - 1 = |S'| + 1$. Otherwise c is the unique dominant color of S , all colors appear at most $d - 1$ times in S' , and $2d(S') \leq 2d - 2 \leq t - 1 = |S'| + 1$, where the middle inequality follows from $2d \leq t + 1$. $\square$

Part (b) of Lemma 2.2 immediately bounds how negative a *relevant* profit can be.

Lemma 2.3. *There exists an optimal solution of (1.3) containing no item $i \in \mathcal{I}$ with $p_i \leq -p_{\max}$.*

Proof. Let S be an optimal solution minimizing the number of items with $p_i \leq -p_{\max}$ and suppose by contradiction that $i \in S$ satisfies $p_i \leq -p_{\max}$. Note that $S \neq \{i\}$, since $p(\{i\}) \leq -p_{\max} < 0 \leq \text{OPT}$. If $\kappa_i \in \mathcal{C}$ is a dominant color of S , then $S \setminus \{i\}$ is feasible by Part (a) of Lemma 2.2 and $p(S \setminus \{i\}) = p(S) - p_i \geq p(S) + p_{\max} > p(S)$, contradicting optimality. Otherwise, pick any item $j \in S$ of a dominant color of S ; then $j \neq i$ and, by Part (b) of Lemma 2.2, $S' = S \setminus \{i, j\}$ is feasible, with $p(S') = p(S) - p_i - p_j \geq p(S) + p_{\max} - p_{\max} = p(S)$. Hence S' is again optimal and contains fewer items with profit at most $-p_{\max}$ than S , a contradiction. $\square$

An immediate consequence of Lemma 2.3, which will be often used in the following, is that all items $i \in \mathcal{I}$ such that $p_i \leq -p_{\max}$ can be deleted in $O(n)$ time, and after this preprocessing every item satisfies

$$-p_{\max} < p_i \leq p_{\max}. \quad (2.3)$$

For this reason, we will assume (2.3) to hold in the remainder of the paper.

We now present the last result of this section, which explains when negative items can be used at all in an optimal ColKP solution.

Lemma 2.4. *Let S be an optimal solution containing an item i with $p_i < 0$. Then S has a unique dominant color, which is also critical, i.e., $|S| = 2d(S) - 1$. Furthermore, S contains at most $d(S) - 1 \leq (n - 1)/2$ items of negative profit.*

Proof. Let $t = |S|$, $d = d(S)$, and $\sigma = t + 1 - 2d \geq 0$. Suppose by contradiction that $\sigma \geq 1$. If $\kappa_i \in \mathcal{C}$ is a dominant color, then $S \setminus \{i\}$ is color-feasible by Part (a) of Lemma 2.2. If it is not, then $d(S \setminus \{i\}) = d$ and $2d \leq t + 1 - \sigma \leq t = |S \setminus \{i\}| + 1$, so $S \setminus \{i\}$ is color-feasible as well. In both cases $w(S \setminus \{i\}) \leq w(S) \leq b$ and $p(S \setminus \{i\}) = p(S) - p_i > p(S)$, contradicting the optimality of S . Hence $\sigma = 0$, i.e., $t = 2d - 1$, meaning that the dominant color is critical and unique. Moreover, no items of the critical color can have negative profit, thus S contains at most $t - d = d - 1 \leq (n - 1)/2$ items of negative profit. $\square$

3. A linear-time 1/3-approximation

The scaling parameter of many FPTASs of knapsack type is $\lambda = \varepsilon \text{LB}/n$, where LB is a lower bound on OPT. After scaling the item profits, the size of the dynamic programming table is proportional to UB/λ , where UB is an upper bound on OPT. The running time of the scheme therefore depends on the ratio UB/LB . Taking $\text{LB} = p_{\max}$ and $\text{UB} = z_{LP}$ (the value of the LP relaxation) may be a natural choice, but it only guarantees $\text{UB}/\text{LB} = O(n)$. In this section we show that a constant ratio can be achieved in linear time, exploiting the results introduced in [11].

We recall the LP relaxation of the ColKP,

$$\begin{aligned}
 (\text{LP}_{\text{Col}}) \quad z_{LP} = \max \Big\{ & \sum_{i \in \mathcal{I}} p_i x_i : \sum_{i \in \mathcal{I}} w_i x_i \leq b, \\
 & \sum_{i \in \mathcal{I}_c} x_i - \sum_{i \in \mathcal{I} \setminus \mathcal{I}_c} x_i \leq 1, \quad c \in \mathcal{C}, \\
 & x_i \in [0, 1], \quad i \in \mathcal{I} \Big\}, \tag{3.1}
 \end{aligned}$$

together with the two properties established in [11]: every basic feasible solution to LP_{Col} has at most two fractional variables (because at most one color constraint can be active at a feasible point), and an optimal solution can be computed in $O(n)$ time (by first solving the KP relaxation to identify a color that is critical in some optimal solution to LP_{Col} , and then applying the multidimensional search algorithm by Megiddo [24] to a resulting two-constraint linear program). Selecting an extreme-point optimum in the final reduced linear program preserves this running time, so the optimal solution used below may be assumed basic.

In Algorithm 3.1, we describe an LP-based primal algorithm, referred to as the LP-ROUND algorithm, which rounds down an optimal basic LP solution and repairs a possible violation of the color constraints by removing one item from the solution.

Theorem 3.1. *Algorithm 3.1 returns a feasible ColKP solution $\hat{S}$ with*

$$p(\hat{S}) \geq \frac{1}{3} z_{LP} \geq \frac{1}{3} \text{OPT},$$

and runs in $O(n)$ time.

Algorithm 3.1 The LP-ROUND algorithm for the ColKP

Require: a ColKP instance

Ensure: a feasible ColKP solution $\hat{S} \subseteq \mathcal{I}$

```

1:  $p_{\max} \leftarrow \max\{p_i : i \in \mathcal{I}\}$ 
2: compute an optimal basic solution  $\{x_1^*, x_2^*, \dots, x_n^*\}$  to  $\text{LP}_{\text{Col}}$ 
3:  $S_0 \leftarrow \{i \in \mathcal{I} : x_i^* = 1\}$ 
4: if  $S_0$  violates (1.2) then
5:   let  $c$  be the unique dominant color and let  $j \in \arg \min\{p_i : i \in S_0 \cap \mathcal{I}_c\}$ 
6:    $S_0 \leftarrow S_0 \setminus \{j\}$  ▷ repair
7: end if
8: let  $r \in \arg \max\{p_i : i \in \mathcal{I}\}$ 
9:  $\hat{S} \in \arg \max\{p(S) : S \in \{S_0, \{r\}\}\}$ 
10: return  $\hat{S}$ 

```

Proof. Let $\{x_1^*, x_2^*, \dots, x_n^*\} \in \mathbb{Q}^n$, be an optimal basic solution to LP_{Col} , let $F = \{i \in \mathcal{I} : 0 < x_i^* < 1\}$, and put $S_0 = \{i \in \mathcal{I} : x_i^* = 1\}$. By [11], $|F| \leq 2$, and therefore $\sum_{i \in F} x_i^* < 2$. Moreover, $w(S_0) \leq \sum_{i \in \mathcal{I}} w_i x_i^* \leq b$, and

$$p(S_0) = z_{LP} - \sum_{i \in F} p_i x_i^* \geq z_{LP} - 2p_{\max}. \quad (3.2)$$

For every $c \in \mathcal{C}$, we have that

$$k_c(S_0) = \sum_{i \in \mathcal{I}_c} x_i^* - \sum_{i \in F \cap \mathcal{I}_c} x_i^* \quad \text{and} \quad |S_0 \setminus \mathcal{I}_c| = \sum_{i \in \mathcal{I} \setminus \mathcal{I}_c} x_i^* - \sum_{i \in F \setminus \mathcal{I}_c} x_i^*.$$

It follows that

$$\begin{aligned}
k_c(S_0) - |S_0 \setminus \mathcal{I}_c| &= \underbrace{\left(\sum_{i \in \mathcal{I}_c} x_i^* - \sum_{i \in \mathcal{I} \setminus \mathcal{I}_c} x_i^* \right)}_{\leq 1 \text{ by feasibility of the LP solution}} - \sum_{i \in F \cap \mathcal{I}_c} x_i^* + \sum_{i \in F \setminus \mathcal{I}_c} x_i^* \\
&\leq 1 + \sum_{i \in F \setminus \mathcal{I}_c} x_i^* < 3.
\end{aligned}$$

Since the left-hand side is an integer, we have that $k_c(S_0) - |S_0 \setminus \mathcal{I}_c| \leq 2$ for every color $c \in \mathcal{C}$, thus S_0 violates each color constraint by at most one unit.

If S_0 is color-feasible, no repair is needed and (3.2) applies directly. Suppose otherwise, and let $c \in \mathcal{C}$ be the violated color. Such a color is unique: if two distinct colors were violated, each would occur at least $(|S_0| + 2)/2$ times, which is impossible. Letting $d = k_c(S_0)$, we therefore have

$$|S_0 \setminus \mathcal{I}_c| = d - 2 \quad \implies \quad |S_0| = 2d - 2.$$

Removing any item $j \in S_0 \cap \mathcal{I}_c$ restores color feasibility. Indeed, the resulting set has $2d - 3$ items, color c occurs $d - 1$ times, and every other color occurs at most $d - 2$ times. Hence the color constraint associated with color c is tight, since $2(d - 1) = (2d - 3) + 1$, and all other color constraints are satisfied as well.

We now bound the amount of profit lost when this repair is needed. We first determine the structure of the fractional variables. The contribution of the integral variables to the color constraint of c is

$$k_c(S_0) - |S_0 \setminus \mathcal{I}_c| = d - (d - 2) = 2.$$

Separating the integral and fractional variables in that constraint, and by feasibility of the LP solution, we obtain

$$2 + \sum_{i \in F \cap \mathcal{I}_c} x_i^* - \sum_{i \in F \setminus \mathcal{I}_c} x_i^* \leq 1,$$

or, equivalently,

$$\sum_{i \in F \setminus \mathcal{I}_c} x_i^* - \sum_{i \in F \cap \mathcal{I}_c} x_i^* \geq 1. \quad (3.3)$$

Recall that $|F| \leq 2$ and that $0 < x_i^* < 1$ for every $i \in F$. If $|F| \leq 1$, the left-hand side of (3.3) is strictly smaller than one. The same is true if $|F| = 2$ and at least one of the two items belongs to $\mathcal{I}_c$: in that case at most one positive term appears in the first sum, and that term is strictly smaller than one. Therefore

$$F = \{a, b\}, \quad a, b \notin \mathcal{I}_c, \quad \text{and} \quad x_a^* + x_b^* \geq 1.$$

We now show that the color constraint of c is active for the LP solution. To this end, consider any color $h \neq c$. Since all items of color h in S_0 belong to $S_0 \setminus \mathcal{I}_c$, we have $k_h(S_0) \leq d - 2$. Moreover, $|S_0| = 2d - 2$. We write the left-hand side of the color constraint of h at the LP solution by separating the integral and fractional variables:

$$\begin{aligned} & \sum_{i \in \mathcal{I}_h} x_i^* - \sum_{i \in \mathcal{I} \setminus \mathcal{I}_h} x_i^* \\ &= \left(k_h(S_0) + \sum_{i \in F \cap \mathcal{I}_h} x_i^* \right) - \left(|S_0 \setminus \mathcal{I}_h| + \sum_{i \in F \setminus \mathcal{I}_h} x_i^* \right) \\ &= k_h(S_0) - \left(|S_0| - k_h(S_0) \right) + \sum_{i \in F \cap \mathcal{I}_h} x_i^* - \sum_{i \in F \setminus \mathcal{I}_h} x_i^* \\ &= 2k_h(S_0) - |S_0| + \sum_{i \in F \cap \mathcal{I}_h} x_i^* - \sum_{i \in F \setminus \mathcal{I}_h} x_i^* \\ &\leq 2(d - 2) - (2d - 2) + \sum_{i \in F \cap \mathcal{I}_h} x_i^* - \sum_{i \in F \setminus \mathcal{I}_h} x_i^* \\ &\leq -2 + \sum_{i \in F \cap \mathcal{I}_h} x_i^* < 0. \end{aligned}$$

Hence, every color constraint other than that of c is slack.

We further observe that the $n - 2$ integral variables x_i^* , $i \in \mathcal{I} \setminus F$, are at one of their bounds and yield $n - 2$ linearly independent active bound constraints. Since the LP solution is basic, there must exist two additional linearly independent active constraints. These are the capacity constraint and the color constraint associated with c , which can be the only tight color constraint. Since $a, b \notin \mathcal{I}_c$, its left-hand side is $2 - x_a^* - x_b^*$, and hence

$$2 - x_a^* - x_b^* = 1, \quad \implies \quad x_a^* + x_b^* = 1.$$

We can now bound the profit loss. We have that

$$p_a x_a^* + p_b x_b^* \leq p_{\max} (x_a^* + x_b^*) = p_{\max}.$$

Therefore

$$p(S_0) = z_{LP} - p_a x_a^* - p_b x_b^* \geq z_{LP} - p_{\max}.$$

The repair removes an item $j \in S_0 \cap \mathcal{I}_c$. Since $p_j \leq p_{\max}$, the repaired set satisfies

$$p(S_0 \setminus \{j\}) = p(S_0) - p_j \geq z_{LP} - 2p_{\max}.$$

If no repair is needed, the same lower bound follows directly from (3.2). Thus, in either case, the set obtained by rounding the LP solution, and possibly repairing the solution obtained, has profit at least $z_{LP} - 2p_{\max}$.

Finally, we notice that Algorithm 3.1 compares the solution obtained with a singleton of profit $p_{\max}$. If $p_{\max} \geq z_{LP}/3$, the singleton has profit at least $z_{LP}/3$. If $p_{\max} < z_{LP}/3$, the rounded and possibly repaired set has profit

$$z_{LP} - 2p_{\max} > z_{LP} - \frac{2}{3} z_{LP} = \frac{1}{3} z_{LP}.$$

In both cases $p(\hat{S}) \geq z_{LP}/3$. Since LP_{Col} is a relaxation, $z_{LP} \geq \text{OPT}$, and the claimed approximation guarantee follows.

All operations other than solving LP_{Col} take $O(n)$ time. Moreover, an optimal basic solution to LP_{Col} can also be computed in $O(n)$ time (as shown in [11]), which proves the running time claim. $\square$

Corollary 3.2. *The worst-case performance ratio of LP-ROUND is exactly $1/3$.*

Proof. Theorem 3.1 shows that the ratio is at least $1/3$. To prove that this bound is tight, let $\ell \geq 2$ be an integer and consider the following instance with two colors and four items:

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	
w_i	1	ℓ	$\ell - 1$	2ℓ	and $b = 3\ell$.
p_i	ℓ	ℓ	$\ell - 1$	ℓ	
κ_i	1	1	1	2	

Set $t = 1 - 1/(3\ell)$. We first show that

$$x_1^* = 1, \quad x_2^* = t, \quad x_3^* = 0, \quad x_4^* = t$$

is an optimal solution to LP_{Col} for this instance. This solution is feasible: we have $0 < t < 1$, its weight is $1 + 3\ell t = 3\ell$, and the color constraint of color 1 is tight, since $1 + t - t = 1$. The color constraint of color 2 is respected as well, since $t - (1 + t) = -1$.

We prove optimality by deriving two upper bounds on the profit of an arbitrary feasible choice of values x_i , $i \in \mathcal{I}$. First, comparing its profit with its weight gives

$$\begin{aligned} \sum_{i \in \mathcal{I}} p_i x_i &= \sum_{i \in \mathcal{I}} w_i x_i + (\ell - 1)x_1 - \ell x_4 \\ &\leq 3\ell + (\ell - 1) - \ell x_4 = 4\ell - 1 - \ell x_4. \end{aligned} \tag{3.4}$$

The equality follows by substituting the weights and profits of the four items. The inequality follows from the capacity constraint $\sum_{i \in \mathcal{I}} w_i x_i \leq 3\ell$ and from $x_1 \leq 1$.

A second upper bound is obtained by rewriting the profit of this solution in terms of the left-hand side of the color constraint associated with color 1, thus obtaining

$$\begin{aligned} \sum_{i \in \mathcal{I}} p_i x_i &= \ell(x_1 + x_2 + x_3 - x_4) - x_3 + 2\ell x_4 \\ &\leq \ell + 2\ell x_4. \end{aligned} \tag{3.5}$$

Here the inequality follows from the color constraint $x_1 + x_2 + x_3 - x_4 \leq 1$ and from $x_3 \geq 0$.

The bound in (3.4) decreases with x_4 , whereas the bound in (3.5) increases with x_4 . They coincide at $x_4 = t$: indeed,

$$4\ell - 1 - \ell t = \ell + 2\ell t = 3\ell - \frac{2}{3}.$$

Therefore, if $x_4 \leq t$, (3.5) gives $\sum_{i \in \mathcal{I}} p_i x_i \leq 3\ell - 2/3$; if $x_4 \geq t$, the same conclusion follows from (3.4). Since $\sum_{i \in \mathcal{I}} p_i x_i^* = \ell(1 + 2t) = 3\ell - 2/3$, the proposed solution is optimal. We now show that the proposed solution is the unique optimal solution as well. As observed above, a solution of profit $3\ell - 2/3$ must have $x_4 = t$. Equality in (3.4) then requires $\sum_{i \in \mathcal{I}} w_i x_i = 3\ell$ and $x_1 = 1$, while equality in (3.5) requires $x_3 = 0$ and $x_1 + x_2 + x_3 - x_4 = 1$. It follows that $x_2 = x_4 = t$. Hence $x_i = x_i^*$ for all $i \in \mathcal{I}$, proving uniqueness. We also observe that the optimal solution is basic.

Rounding down the optimal solution gives $S_0 = \{1\}$, which is already color-feasible, so no repair is performed. Since $p_{\max} = \ell$, LP-ROUND returns a solution $\hat{S}_\ell$ of profit ℓ .

On the other hand, $S_\ell^* = \{1, 3, 4\}$ is color-feasible, has weight 3ℓ , and has profit $3\ell - 1$. This solution is optimal. Indeed, every set with at most two items has profit at most 2ℓ , while a color-feasible solution with three items must contain item 4 and two items of color 1. The other two such solutions apart from S_ℓ^* , $\{1, 2, 4\}$ and $\{2, 3, 4\}$, have weight $3\ell + 1$ and $4\ell - 1$, respectively, and are therefore infeasible. Finally, the set of all four items violates both the capacity constraint and the color constraint of color 1. Therefore

$$\frac{p(\hat{S}_\ell)}{p(S_\ell^*)} = \frac{\ell}{3\ell - 1} \xrightarrow{\ell \rightarrow \infty} \frac{1}{3}.$$

Together with Theorem 3.1, this proves the claim. $\square$

We now run Algorithm 3.1 on the instance of Example 1.1. An optimal solution to the KP relaxation assigns $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$, and $x_5 = \frac{3}{7}$, and has value $141/7$. The color constraint of color 1 reads $2 - \frac{3}{7} = \frac{11}{7} > 1$ and is violated, so by Lemma 2 of [11] color 1 is critical in some optimal solution to LP_{Col} . Solving the resulting linear program, obtained by removing all the color constraints except the one for color 1, gives

$$x_1^* = 0, \quad x_2^* = 1, \quad x_3^* = 1, \quad x_4^* = \frac{2}{3}, \quad x_5^* = \frac{1}{3}, \quad \text{and} \quad z_{LP} = \frac{59}{3}.$$

The two fractional variables are associated with items 4 and 5, both of color $\kappa_4 = \kappa_5 = 2$, which illustrates that two fractional variables of an optimal basic solution may share a (non-critical) color. Rounding down gives $S_0 = \{2, 3\}$, two items of color 1 and none of color 2: the color constraint is violated by exactly one unit, since $k_1(S_0) = 2 = |S_0 \setminus \mathcal{I}_1| + 2$. The repair removes the least profitable item of color 1 in S_0 , namely item 2, leaving $\{3\}$ of profit 11, which is feasible. The

best singleton is item 5, of profit $p_{\max} = 12$, and the algorithm returns it, thus $\text{LB} = 12$. This is a lower bound on the optimal value $\text{OPT} = 17$ of the instance, which is attained by the feasible solution $\{2, 4, 5\}$, and it also shows that the ratio LB/OPT can be, in practice, much better than the worst-case guarantee of $1/3$.

From now on we denote by

$$\text{LB} = p(\hat{S}) \quad \text{and} \quad \text{UB} = 3 \cdot \text{LB} \quad (3.6)$$

the lower and upper bounds provided by Theorem 3.1, so that $p_{\max} \leq \text{LB} \leq \text{OPT} \leq \text{UB}$.

4. A first profit-indexed DP algorithm

In this section we describe the exact DP algorithm on which the first approximation scheme proposed in this paper is built. It coincides with the item-by-item dynamic program DP_1 of [11] with the roles of the capacity and of the objective exchanged.

Assume the items are sorted by color, so that each color class occupies a contiguous block of indices. The state of DP_1 after processing the first i items records the number t of selected items, the number d of selected items of a dominant color, the number a of selected items of the current color $\kappa_i \in \mathcal{C}$, and the consumed capacity q . Its value is the maximum profit. The key observation is that the color feasibility of a partial solution $S \subseteq \{1, 2, \dots, i\}$, and of every extension of S by items of $\{i+1, i+2, \dots, n\}$, is completely determined by the triple $(t, d, a) = (|S|, d(S), k_{\kappa_i}(S))$ and by the items still to be processed. It does not depend on $w(S)$ nor on $p(S)$. Hence, for two partial solutions with the same (i, t, d, a) and the same profit, the one of smaller weight can never be worse. This dominance property makes the classical exchange of dimensions legitimate. We observe that this is not automatic for KP variants with combinatorial side constraints: it would fail, for instance, under a classwise lower bound on the total selected weight, as in one variant of the KP with group fairness [23], since then a lighter partial solution would not dominate a heavier one.

For $i \in \{1, 2, \dots, n\}$, $t, d, a \in \{0, 1, \dots, n\}$, and an integer profit value z , define

$$g(i, t, d, a, z) = \min \left\{ w(S) : S \subseteq \{1, 2, \dots, i\}, |S| = t, d(S) = d, k_{\kappa_i}(S) = a, p(S) = z \right\}, \quad (4.1)$$

with $g = +\infty$ when no such subset exists. Let $\sigma(a, d) = 1$ if $a = d$ and $\sigma(a, d) = 0$ otherwise. At stage $i = 0$, we use the initialization convention $g(0, 0, 0, 0, 0) = 0$, with all other states having value equal to $+\infty$. Algorithm 4.1 computes g by the two usual transitions (pack / do not pack item i), resetting the current-color counter at every color change.

Proposition 4.1. *Algorithm 4.1 computes $g(i, t, d, a, z)$ as defined in (4.1) for every state, and returns an optimal ColKP solution.*

Proof. We proceed by induction on i . For $i = 0$ the only subset is $\emptyset$, with $t = d = a = z = 0$ and weight 0, which is the initialization. Assume the claim holds for $i - 1$. Since the items are sorted by color, either $i = 1$ or $\kappa_i = \kappa_{i-1}$ (the current color continues, and the counter a is inherited) or $\kappa_i \neq \kappa_{i-1}$ (item i is the first of a new color, and the counter is reset to $a' = 0$). Let $S \subseteq \{1, 2, \dots, i\}$ realize a state (i, t, d, a'', z) . If $i \notin S$, then $S \subseteq \{1, 2, \dots, i-1\}$ realizes the state $(i-1, t, d, a, z)$ for the appropriate value of a and is propagated by the first transition, with unchanged t, d and z . If

Algorithm 4.1 DP_1^P : profit-indexed dynamic program

Require: a ColKP instance, with items indexed by nondecreasing color index

Ensure: an optimal ColKP solution $S^* \subseteq \mathcal{I}$ and its value $p(S^*) = \text{OPT}$

```
1:  $p_{\text{abs}} \leftarrow \sum_{i \in \mathcal{I}} |p_i|$ ;  $\mathcal{Z} \leftarrow \{-p_{\text{abs}}, -p_{\text{abs}} + 1, \dots, p_{\text{abs}}\}$ 
2:  $g(i, t, d, a, z) \leftarrow +\infty$  for all  $(i, t, d, a, z) \in \{0, 1, \dots, n\}^4 \times \mathcal{Z}$ 
3:  $g(0, 0, 0, 0, 0) \leftarrow 0$ 
4: for  $i = 1, \dots, n$  do
5:   for all  $(t, d, a, z)$  with  $g(i-1, t, d, a, z) < +\infty$  do
6:      $a' \leftarrow a$  if  $i > 1$  and  $\kappa_i = \kappa_{i-1}$ , else  $a' \leftarrow 0$ 
7:      $g(i, t, d, a', z) \leftarrow \min\{g(i, t, d, a', z), g(i-1, t, d, a, z)\}$   $\triangleright$  do not pack  $i$ 
8:      $d' \leftarrow d + \sigma(a', d)$ 
9:      $g(i, t+1, d', a'+1, z+p_i) \leftarrow \min\{g(i, t+1, d', a'+1, z+p_i), g(i-1, t, d, a, z) + w_i\}$   $\triangleright$  pack  $i$ 
10:   end for
11: end for
12:  $z^* \leftarrow \max\{z \in \mathcal{Z} : \exists (t, d, a) \text{ with } 2d \leq t+1 \text{ and } g(n, t, d, a, z) \leq b\}$ 
13: choose  $(t^*, d^*, a^*)$  with  $2d^* \leq t^*+1$  and  $g(n, t^*, d^*, a^*, z^*) \leq b$ 
14: recover  $S^*$  by backtracking the transitions leading to  $g(n, t^*, d^*, a^*, z^*)$ 
15: return  $S^*$  and  $z^*$ 
```

$i \in S$, then $S \setminus \{i\}$ realizes a state at stage $i-1$ with $t-1$ items, profit $z-p_i$, weight $w(S)-w_i$, current-color counter $a' = a'' - 1$ and dominant cardinality $d(S \setminus \{i\})$; since $d(S) = \max\{d(S \setminus \{i\}), a' + 1\}$ and $a' \leq d(S \setminus \{i\})$, we have $d(S) = d(S \setminus \{i\}) + \sigma(a', d(S \setminus \{i\}))$, which is exactly the update of line 6. Moreover $S \setminus \{i\}$ must be of minimum weight among the subsets realizing its state, otherwise replacing it would contradict the minimality of $w(S)$. By the induction hypothesis its weight is stored at stage $i-1$, and the second transition adds w_i . Since the two cases are exhaustive and mutually exclusive and the algorithm takes minima over all incoming transitions, the claim follows. The output is correct because a set S is feasible if and only if $2d(S) \leq |S| + 1$ and $w(S) \leq b$, and for each profit level it is enough to know the minimum weight realizing it. $\square$

It is worth noting that the state-reduction rules described in [11] can be adapted to DP_1^P , but they are optional and are not used in the correctness or running time results below.

5. A first approximation scheme

We are now ready to state and analyze the first approximation scheme we propose in this article, summarized in Algorithm 5.1. The idea is the same as in the classical FPTAS for the KP: profits are scaled so that the profit dimension of the DP table becomes polynomial in n and $1/\varepsilon$. More precisely, given the lower bound LB of (3.6), we set $\lambda = \varepsilon \text{LB}/n$ and replace every profit p_i by $\hat{p}_i = \lfloor p_i/\lambda \rfloor$, $i \in \mathcal{I}$. It can be shown that the total loss caused by rounding down the scaled profits is then at most $n\lambda = \varepsilon \text{LB} \leq \varepsilon \text{OPT}$. At the same time, the number of scaled profit levels in the DP table depends on the value UB/λ , and hence by the ratio UB/LB .

The following proposition establishes the formal correctness and approximation guarantee of the algorithm.

Algorithm 5.1 FPTAS for the ColKP

Require: a ColKP instance and an accuracy $\varepsilon \in (0, 1)$

Ensure: a feasible ColKP solution $\hat{S} \subseteq \mathcal{I}$

- 1: let $p_{\max} = \max\{p_i : i \in \mathcal{I}\}$ ▷ Lemma 2.3
 - 2: $S^H \leftarrow \text{LP-ROUND}$; $\text{LB} \leftarrow p(S^H)$; $\text{UB} \leftarrow 3\text{LB}$ ▷ Theorem 3.1
 - 3: $\lambda \leftarrow \varepsilon \text{LB}/n$; $\hat{p}_i \leftarrow \lfloor p_i/\lambda \rfloor$ for all $i \in \mathcal{I}$
 - 4: **if** $p_i \geq 0$ for all $i \in \mathcal{I}$ **then**
 - 5: $z_{\min} \leftarrow 0$; $z_{\max} \leftarrow \lceil \text{UB}/\lambda \rceil$
 - 6: **else**
 - 7: $z_{\min} \leftarrow \lfloor -np_{\max}/\lambda \rfloor$; $z_{\max} \leftarrow \lceil (\text{UB} + np_{\max})/\lambda \rceil$
 - 8: **end if**
 - 9: sort the items by color
 - 10: run Algorithm 4.1 with profits $\hat{p}_i$, $i \in \mathcal{I}$, restricted to profit levels $z \in [z_{\min}, z_{\max}]$ and to states with $g \leq b$
 - 11: let $\hat{S}$ be a feasible set of maximum $\hat{p}(\hat{S})$ found, and recover it by backtracking
 - 12: **return** $\hat{S}$
-

Proposition 5.1. *For every $\varepsilon \in (0, 1)$, Algorithm 5.1 returns a feasible ColKP solution of value at least $(1 - \varepsilon)\text{OPT}$.*

Proof. We observe that (3.6) gives $\text{LB} \geq p_{\max} > 0$, so the scaling factor $\lambda = \varepsilon \text{LB}/n$ is well defined and positive. Moreover, every item has profit at most $p_{\max}$, and hence $\text{OPT} \leq np_{\max}$. Since $\text{LB} \leq \text{OPT}$ and $\varepsilon < 1$, it follows that

$$\lambda = \frac{\varepsilon \text{LB}}{n} \leq \varepsilon p_{\max} < p_{\max}, \quad \text{and thus} \quad \frac{p_{\max}}{\lambda} > 1. \quad (5.1)$$

Let S^* be an optimal solution to a ColKP instance. For every item i , the definition $\hat{p}_i = \lfloor p_i/\lambda \rfloor$ yields

$$\hat{p}_i \leq \frac{p_i}{\lambda} < \hat{p}_i + 1,$$

or, equivalently, $\lambda \hat{p}_i \leq p_i < \lambda(\hat{p}_i + 1)$. Summing these inequalities over an arbitrary set $S \subseteq \mathcal{I}$ gives

$$\lambda \hat{p}(S) \leq p(S) < \lambda \hat{p}(S) + |S|\lambda \leq \lambda \hat{p}(S) + n\lambda. \quad (5.2)$$

We now verify that truncating the profit axis does not remove any state needed to represent S^* . Recall that the items have been sorted by color before the dynamic program is run. For every $i \in \{1, 2, \dots, n\}$, let $S_i^* = S^* \cap \{1, 2, \dots, i\}$ be the prefix of S^* containing the first i items, and let $N = \{j \in S^* : p_j < 0\}$ be the set of items in S^* with negative profit. Every $j \in N$ satisfies $|p_j| < p_{\max}$ by (2.3). Hence $\sum_{j \in N} |p_j| < np_{\max}$. Furthermore, the profit of every prefix S_i^* is at least the sum of all the negative profits in S^* and at most the sum of all the positive profits in S^* . Therefore

$$-np_{\max} < \sum_{j \in N} p_j \leq p(S_i^*) \leq \sum_{j \in S^* \setminus N} p_j = \sum_{j \in S^*} p_j + \sum_{j \in N} |p_j| < \text{UB} + np_{\max}. \quad (5.3)$$

We now want to prove that the scaled profit of every partial solution S_i^* belongs to the interval $[z_{\min}, z_{\max}]$. Suppose first that some item profit is negative. If $N = \emptyset$, the scaled profit of each

prefix is nonnegative. Otherwise, Lemma 2.4 gives $|N| \leq (n-1)/2$. For every $j \in N$,

$$\hat{p}_j = \left\lfloor \frac{p_j}{\lambda} \right\rfloor > \frac{p_j}{\lambda} - 1 > -\frac{p_{\max}}{\lambda} - 1,$$

while $\hat{p}_j \geq 0$ for every nonnegative-profit item. Consequently, for every partial solution S_i^* ,

$$\begin{aligned} \hat{p}(S_i^*) &> -\frac{n-1}{2} \left(\frac{p_{\max}}{\lambda} + 1 \right) \\ &> -\frac{np_{\max}}{\lambda} \geq \left\lfloor -\frac{np_{\max}}{\lambda} \right\rfloor = z_{\min}. \end{aligned}$$

On the other hand, (5.2) and (5.3) give

$$\hat{p}(S_i^*) \leq \frac{p(S_i^*)}{\lambda} < \frac{\text{UB} + np_{\max}}{\lambda} \leq \left\lceil \frac{\text{UB} + np_{\max}}{\lambda} \right\rceil = z_{\max}.$$

Thus the scaled profit of every partial solution belongs to $[z_{\min}, z_{\max}]$. Observe that, if all profits are nonnegative, $\hat{p}_i \geq 0$ for all $i \in \mathcal{I}$, and the tighter bounds used by the algorithm follow directly from

$$0 \leq \hat{p}(S_i^*) \leq \hat{p}(S^*) \leq \frac{p(S^*)}{\lambda} = \frac{\text{OPT}}{\lambda} \leq \frac{\text{UB}}{\lambda}.$$

We therefore conclude that the scaled profit of every partial solution belongs to $[z_{\min}, z_{\max}]$ in this case as well.

We now want to show that the proposed dynamic program returns a solution whose scaled profit is at least $\hat{p}(S^*)$. For every $i \in \{1, 2, \dots, n\}$, since all weights are positive, we have that $w(S_i^*) \leq w(S^*) \leq b$: consequently, the state corresponding to S_i^* is not discarded for exceeding the capacity. For $i = n$, the resulting state has profit $\hat{p}(S_n^*) = \hat{p}(S^*)$ and satisfies $2d(S^*) \leq |S^*| + 1$, because S^* is feasible by assumption. Hence $\hat{p}(S^*)$ is one of the terminal profit levels over which the dynamic program maximizes. Since $\hat{S}$ is chosen with maximum scaled profit among those terminal states, we obtain

$$\hat{p}(\hat{S}) \geq \hat{p}(S^*). \quad (5.4)$$

The value returned by Algorithm 5.1 is therefore the value of a feasible solution and, using (5.2) and (5.4), we obtain

$$\begin{aligned} p(\hat{S}) &\geq \lambda \hat{p}(\hat{S}) \\ &\geq \lambda \hat{p}(S^*) \\ &> p(S^*) - |S^*|\lambda \\ &\geq \text{OPT} - n\lambda \\ &= \text{OPT} - \varepsilon \text{LB} \\ &\geq (1 - \varepsilon) \text{OPT}. \end{aligned}$$

□

After proving the approximation guarantee, we now bound the size of the scaled dynamic-programming table and derive the running time of the proposed approximation scheme.

Theorem 5.2. *Algorithm 5.1 runs in $O(n^5/\varepsilon)$ time when all profits are nonnegative, and in $O(n^6/\varepsilon)$ time for arbitrary integer profits.*

Proof. Consider first the case $p_i \geq 0$ for all $i \in \mathcal{I}$, in which $z_{\min} = 0$ and the profit axis has

$$\Pi = z_{\max} + 1 = O\left(\frac{\text{UB}}{\lambda}\right) = O\left(\frac{3\text{LB}n}{\varepsilon\text{LB}}\right) = O\left(\frac{n}{\varepsilon}\right)$$

levels.

In Algorithm 4.1 the state components t and d take $O(n)$ values each, while the counter a of the current color is bounded by $|\mathcal{I}_c|$ for all colors $c \in \mathcal{C}$. Summing over the items grouped by color, the number of states is

$$\sum_{c \in \mathcal{C}} \sum_{i \in \mathcal{I}_c} O(n \cdot n \cdot |\mathcal{I}_c| \cdot \Pi) = O\left(n^2 \Pi \sum_{c \in \mathcal{C}} |\mathcal{I}_c|^2\right) = O\left(\frac{n^3}{\varepsilon} \sum_{c \in \mathcal{C}} |\mathcal{I}_c|^2\right),$$

and each state is processed with $O(1)$ transitions. Since $\sum_c |\mathcal{I}_c|^2 \leq (\sum_c |\mathcal{I}_c|)^2 = n^2$, the bound $O(n^5/\varepsilon)$ follows.

For arbitrary integer profits, the profit axis has instead

$$\Pi = z_{\max} - z_{\min} + 1 \leq \frac{\text{UB} + 2np_{\max}}{\lambda} + 3 = \frac{3\text{LB}n + 2n^2p_{\max}}{\varepsilon\text{LB}} + 3 = O\left(\frac{n^2}{\varepsilon}\right).$$

The running time therefore becomes $O(n^6/\varepsilon)$. □

The additional factor n comes from the need to retain every prefix of an optimal solution. With nonnegative profits, the profit of a partial solution increases monotonically and remains between 0 and the profit of the complete solution, so using the interval $[0, \text{UB}/\lambda]$ ensures that all possible profits are covered. With arbitrary profits, a partial solution may instead have negative profit or may exceed the final profit before later negative-profit items are processed. Both ends of the scaled profit axis must therefore include an additional contribution of order $np_{\max}/\lambda$.

We conclude the section by illustrating the behaviour of the FPTAS detailed in Algorithm 5.1 on the instance depicted in Example 1.1. We first set $\varepsilon = 1/2$. Algorithm 3.1 returns $\text{LB} = 12$ on this instance, as described in Section 3, so $\lambda = \varepsilon\text{LB}/n = 6/5$, the scaled profits are those of Table 5.1, and the profit axis is truncated at $z_{\max} = \lceil \text{UB}/\lambda \rceil = 30$. The largest scaled profit reached by a feasible solution is $z = 13$, attained both by $\{2, 5\}$, of weight 8, and by the optimal solution $S^* = \{2, 4, 5\}$, of weight 9. These sets correspond to distinct terminal DP states, so either may be returned depending on tie-breaking. In particular, $\{2, 5\}$ is a possible output, of true profit 16. This illustrates the effect of scaling: $\hat{p}_4 = 0$, so item 4 can become invisible in the scaled objective although it still consumes capacity. The performance guarantee holds, since $16 \geq (1 - \varepsilon)\text{OPT} = \frac{17}{2}$.

Consider instead $\varepsilon = 1/100$. The scaling factor becomes $\lambda = \varepsilon\text{LB}/n = 3/125$, the scaled profits are given in the last row of Table 5.1, and the profit axis is truncated at $z_{\max} = \lceil \text{UB}/\lambda \rceil = 1500$. Now item 4 has scaled profit 41, rather than zero. The largest scaled profit reached by a feasible solution is 707, uniquely attained by the optimal solution $S^* = \{2, 4, 5\}$. Such profit is strictly larger than the scaled profit 666 of $\{2, 5\}$. Hence the DP returns S^* . In fact, the optimality of the returned solution also follows directly from the approximation analysis: it gives $p(\hat{S}) > \text{OPT} - \varepsilon\text{LB} = \frac{422}{25}$, and, since all solution profits are integral, this implies $p(\hat{S}) \geq 17 = \text{OPT}$. Thus, for this instance, reducing ε not only tightens the guaranteed ratio, but makes the FPTAS exact, at the cost of a profit axis that is fifty times longer.

Table 5.1: Scaling of the item profits in the instance of Example 1.1, with $LB = 12$, for $\varepsilon = 1/2$ and $\varepsilon = \frac{1}{100}$.

	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
p_i	4	4	11	1	12
$\hat{p}_i$ ($\varepsilon = 1/2$)	3	3	9	0	10
$\hat{p}_i$ ($\varepsilon = \frac{1}{100}$)	166	166	458	41	500

6. Can the color-by-color recursion do better?

Neither of the exact dynamic programs proposed in [11] dominates the other: DP_1 runs in $O(bn^4)$ and DP_2 in $O(b^2n^3)$. Thus DP_2 has the smaller asymptotic bound when $b = o(n)$ and is no worse in order when $b = O(n)$. It is therefore natural to ask whether DP_2 may provide a basis for a faster approximation scheme than the one based on DP_1 . We show that the direct profit-scaling implementation considered here does not improve the worst-case running time, and explain the source of the additional factor.

Recall the structure of DP_2 . For each color $c \in \mathcal{C}$, an *inner* dynamic program enumerates the relevant packings of the items of that color. An *outer* dynamic program then combines one packing per color, in the spirit of a Multiple-Choice Knapsack Problem, keeping track of the number of packed items and of the number of items of a dominant color. DP_2 is indexed by the amount of consumed capacity, and its value corresponds to a solution profit. We denote by DP_2^P the DP obtained by exchanging the two dimensions, exactly as DP_1^P was obtained from DP_1 in Section 4: the inner and the outer DPs are therefore indexed by the profit, and their value is the minimum weight of a solution achieving that profit. The exchange can be performed here for the same reason as in the case of DP_1^P . By (1.2), the color feasibility of a set $S \subseteq \mathcal{I}$ depends only on the color cardinalities $k_c(S)$, $c \in \mathcal{C}$. Hence, replacing a packing $S_c \subseteq \mathcal{I}_c$ chosen for a color $c \in \mathcal{C}$ by any other packing $S'_c \subseteq \mathcal{I}_c$ with $|S'_c| = |S_c|$ and $p(S'_c) = p(S_c)$ leaves the cardinalities and the profit of the overall solution unchanged, and does not increase its weight if $w(S'_c) \leq w(S_c)$. Within a color class, therefore, we need to keep only one minimum-weight packing per pair cardinality-profit.

Specifically, for every color $c \in \mathcal{C}$, cardinality $k \in \{0, 1, \dots, |\mathcal{I}_c|\}$ and integer profit level ζ , let

$$h_c(k, \zeta) = \min\{w(S) : S \subseteq \mathcal{I}_c, |S| = k, p(S) = \zeta\}, \quad (6.1)$$

where $h_c(k, \zeta) = +\infty$ if no solution with k items and profit ζ exists. This table can be computed by the usual item-by-item knapsack DP with an additional cardinality counter, with the roles of capacity and profit exchanged. Following the notation of [11], we then define

$$\mathcal{S}_c = \{(k, \zeta) \in \{0, 1, \dots, |\mathcal{I}_c|\} \times \mathbb{Z} : h_c(k, \zeta) < +\infty\}$$

that is, a set containing all combinations of number of items k and profit ζ for which there exists a feasible packing of the items having color $c \in \mathcal{C}$. The weight associated with a pair (k, ζ) is $h_c(k, \zeta)$. In particular, $(0, 0) \in \mathcal{S}_c$ represents an empty solution.

For $c \in \{1, 2, \dots, m\}$, $t, d \in \{0, 1, \dots, n\}$ and integer profit level z , the outer program of DP_2^P

then computes

$$g(c, t, d, z) = \min \left\{ \sum_{j=1}^c h_j(k_j, \zeta_j) : (k_j, \zeta_j) \in \mathcal{S}_j \text{ for all } j \in \{1, 2, \dots, c\}, \right. \\ \left. \sum_{j=1}^c k_j = t, \quad \max_{1 \leq j \leq c} k_j = d, \quad \sum_{j=1}^c \zeta_j = z \right\}. \quad (6.2)$$

We initialize the empty solution as $g(0, 0, 0, 0) = 0$, with all other states having value equal to $+\infty$. For each state $(c-1, t, d, z)$ such that $g(c-1, t, d, z) < +\infty$, and each $(k, \zeta) \in \mathcal{S}_c$, the forward transition is

$$g(c, t+k, \max\{d, k\}, z+\zeta) \leftarrow \min \{ g(c, t+k, \max\{d, k\}, z+\zeta), \\ g(c-1, t, d, z) + h_c(k, \zeta) \}. \quad (6.3)$$

After running the outer DP, the maximum profit level with $g(m, t, d, z) \leq b$ and $2d \leq t+1$ for some $t, d \in \{0, 1, \dots, n\}$ is therefore the optimal profit, and a corresponding solution can be recovered by backtracking through the outer and inner programs.

Algorithm 6.1 reports the complete pseudocode of the profit-indexed color-by-color dynamic program DP_2^P .

Algorithm 6.1 DP_2^P : profit-indexed color-by-color dynamic program

Require: a ColKP instance

Ensure: an optimal ColKP solution $S^* \subseteq \mathcal{I}$ and its value $p(S^*) = \text{OPT}$

```

1:  $p_{\text{abs}} \leftarrow \sum_{i \in \mathcal{I}} |p_i|$ ;  $\mathcal{Z} \leftarrow \{-p_{\text{abs}}, -p_{\text{abs}} + 1, \dots, p_{\text{abs}}\}$ 
2: for  $c = 1, \dots, m$  do
3:   compute  $\mathcal{S}_c$  and the values  $h_c(k, \zeta)$  for  $(k, \zeta) \in \mathcal{S}_c$ 
4: end for
5:  $g(c, t, d, z) \leftarrow +\infty$  for all  $(c, t, d, z) \in \{0, \dots, m\} \times \{0, \dots, n\}^2 \times \mathcal{Z}$ 
6:  $g(0, 0, 0, 0) \leftarrow 0$ 
7: for  $c = 1, \dots, m$  do
8:   for all  $(t, d, z)$  with  $g(c-1, t, d, z) < +\infty$  do
9:     for all  $(k, \zeta) \in \mathcal{S}_c$  do
10:       $t' \leftarrow t+k$ ;  $d' \leftarrow \max\{d, k\}$ ;  $z' \leftarrow z+\zeta$ 
11:       $\bar{w} \leftarrow g(c-1, t, d, z) + h_c(k, \zeta)$ 
12:       $g(c, t', d', z') \leftarrow \min\{g(c, t', d', z'), \bar{w}\}$ 
13:    end for
14:   end for
15: end for
16:  $z^* \leftarrow \max\{z \in \mathcal{Z} : \exists (t, d) \text{ with } 2d \leq t+1 \text{ and } g(m, t, d, z) \leq b\}$ 
17: choose  $(t^*, d^*)$  with  $2d^* \leq t^*+1$  and  $g(m, t^*, d^*, z^*) \leq b$ 
18: recover  $S^*$  by backtracking through the outer and inner transitions
19: return  $S^*$  and  $z^*$ 
```

The next proposition states that DP_2^P is exact.

Proposition 6.1. *Algorithm 6.1 computes $g(c, t, d, z)$ as defined in (6.2) for every state, and returns an optimal ColKP solution.*

Proof. For every color $c \in \mathcal{C}$, a preliminary item-by-item DP computes $h_c(k, \zeta)$ as defined in (6.1); consequently, $\mathcal{S}_c$ contains exactly the attainable cardinality–profit pairs for color c .

We now prove the claim on the outer DP by induction on c . For $c = 0$ the only admissible choice in (6.2) is the empty one, which gives $g(0, 0, 0, 0) = 0$ and $g(0, t, d, z) = +\infty$ for every other state. Assume that the claim holds for the states of stage $c - 1$, and consider a state (c, t, d, z) .

We first show that the value computed at (c, t, d, z) is at most the right-hand side of (6.2). Let $(k_j, \zeta_j)_{j=1}^c$ be admissible for (c, t, d, z) and let

$$t' = t - k_c, \quad d' = \max_{1 \leq j \leq c-1} k_j, \quad z' = z - \zeta_c.$$

The prefix $(k_j, \zeta_j)_{j=1}^{c-1}$ is admissible for the state $(c-1, t', d', z')$, so that $g(c-1, t', d', z') \leq \sum_{j=1}^{c-1} h_j(k_j, \zeta_j) < +\infty$ by the induction hypothesis. Moreover $t' + k_c = t$, $\max\{d', k_c\} = d$ and $z' + \zeta_c = z$. Hence the transition (6.3) applied to the state $(c-1, t', d', z')$ with the pair $(k_c, \zeta_c) \in \mathcal{S}_c$ writes into (c, t, d, z) , and

$$g(c, t, d, z) \leq g(c-1, t', d', z') + h_c(k_c, \zeta_c) \leq \sum_{j=1}^c h_j(k_j, \zeta_j).$$

Taking the minimum over all admissible choices gives the desired inequality.

Conversely, every finite value written into (c, t, d, z) is of the form $g(c-1, t', d', z') + h_c(k, \zeta)$, for some state $(c-1, t', d', z')$ of finite value and some pair $(k, \zeta) \in \mathcal{S}_c$ with $t' + k = t$, $\max\{d', k\} = d$ and $z' + \zeta = z$. By the induction hypothesis, $g(c-1, t', d', z')$ is attained by a choice $(k_j, \zeta_j)_{j=1}^{c-1}$ admissible for $(c-1, t', d', z')$. Setting $(k_c, \zeta_c) = (k, \zeta)$ extends it to a choice that is admissible for (c, t, d, z) and whose value is exactly the value written. No state of the table has therefore value smaller than the right-hand side of (6.2), and the two coincide.

It remains to examine the output. Let (m, t, d, z) be such that $g(m, t, d, z) < +\infty$, let $(k_c, \zeta_c)_{c=1}^m$ be an admissible choice attaining that value, and let $S_c \subseteq \mathcal{I}_c$ be a minimum-weight solution featuring k_c items, with a profit of ζ_c , and a total weight of $h_c(k_c, \zeta_c)$. The set $S = \bigcup_{c \in \mathcal{C}} S_c$ satisfies $k_c(S) = k_c$ for every color $c \in \mathcal{C}$, and hence

$$|S| = \sum_{c=1}^m k_c, \quad d(S) = \max_{c \in \mathcal{C}} k_c = d, \quad p(S) = \sum_{c=1}^m \zeta_c = z, \quad w(S) = g(m, t, d, z).$$

A profit level z is therefore obtained by a feasible solution if and only if $g(m, t, d, z) \leq b$ and $2d \leq t + 1$ for some $t, d \in \{0, 1, \dots, n\}$, so that the largest such z equals OPT and backtracking returns an optimal solution. $\square$

For the approximation scheme, both the inner and the outer program of DP_2^{P} are run on the scaled item profits $\hat{p}_i$, $i \in \mathcal{I}$, and both are restricted to the profit levels $\mathcal{Z} = \{z_{\min}, z_{\min} + 1, \dots, z_{\max}\}$, with endpoints as in Algorithm 5.1. Let $\Pi = |\mathcal{Z}| = z_{\max} - z_{\min} + 1$. States with a weight greater than b are discarded, so that a transition of both the inner and the outer DP is performed only if the profit level it produces belongs to $\mathcal{Z}$ and the weight it produces is at most b . In the next proposition, we show that these restrictions preserve the states needed for the approximation guarantee. The resulting scheme is detailed in Algorithm 6.2.

Proposition 6.2. *For every $\varepsilon \in (0, 1)$, Algorithm 6.2 returns a feasible ColKP solution of value at least $(1 - \varepsilon)\text{OPT}$.*

Algorithm 6.2 FPTAS for the ColKP based on DP_2^p

Require: a ColKP instance and an accuracy $\varepsilon \in (0, 1)$

Ensure: a feasible ColKP solution $\hat{S} \subseteq \mathcal{I}$

- 1: let $p_{\max} = \max\{p_i : i \in \mathcal{I}\}$ $\triangleright$ Lemma 2.3
 - 2: $S^H \leftarrow \text{LP-ROUND}$; $\text{LB} \leftarrow p(S^H)$; $\text{UB} \leftarrow 3\text{LB}$ $\triangleright$ Theorem 3.1
 - 3: $\lambda \leftarrow \varepsilon \text{LB}/n$; $\hat{p}_i \leftarrow \lfloor p_i/\lambda \rfloor$ for all $i \in \mathcal{I}$
 - 4: **if** $p_i \geq 0$ for all $i \in \mathcal{I}$ **then**
 - 5: $z_{\min} \leftarrow 0$; $z_{\max} \leftarrow \lceil \text{UB}/\lambda \rceil$
 - 6: **else**
 - 7: $z_{\min} \leftarrow \lfloor -np_{\max}/\lambda \rfloor$; $z_{\max} \leftarrow \lceil (\text{UB} + np_{\max})/\lambda \rceil$
 - 8: **end if**
 - 9: run Algorithm 6.1 with profits $\hat{p}_i$, restricting both inner and outer tables to profit levels $z \in [z_{\min}, z_{\max}]$ and to states of weight at most b
 - 10: let $\hat{S}$ be a feasible set of maximum $\hat{p}(\hat{S})$ found, and recover it by backtracking
 - 11: **return** $\hat{S}$
-

Proof. By positivity of $p_{\max}$, we have $\text{LB} \geq p_{\max} > 0$, while (5.1) gives $p_{\max}/\lambda > 1$.

Let S^* be an optimal solution to a ColKP instance. As in (5.2), for every set $S \subseteq \mathcal{I}$ we have

$$\lambda \hat{p}(S) \leq p(S) \leq \lambda \hat{p}(S) + |S|\lambda \leq \lambda \hat{p}(S) + n\lambda. \quad (6.4)$$

We now verify that truncating the profit axis does not remove any state needed to represent S^* . The inner and outer programs build subsets of S^* in different orders, so we prove the stronger statement that $\hat{p}(T) \in \mathcal{Z}$ for every $T \subseteq S^*$. If all profits are nonnegative, then

$$0 \leq \hat{p}(T) \leq \hat{p}(S^*) \leq \text{OPT}/\lambda \leq \text{UB}/\lambda \leq z_{\max}.$$

Suppose now that some item profit is negative, and let $N = \{i \in S^* : p_i < 0\}$. If $N = \emptyset$, the lower bound follows from $\hat{p}(T) \geq 0$. Otherwise, Lemma 2.4 gives $|N| \leq (n-1)/2$, while $p_i > -p_{\max}$ for every $i \in N$. Consequently,

$$\hat{p}(T) \geq \sum_{i \in N} \hat{p}_i > -\frac{n-1}{2} \left(\frac{p_{\max}}{\lambda} + 1 \right) > -\frac{np_{\max}}{\lambda} \geq z_{\min}.$$

For the upper bound, whether or not N is empty,

$$\hat{p}(T) \leq \frac{1}{\lambda} \sum_{i \in S^* \setminus N} p_i = \frac{\text{OPT} + \sum_{i \in N} |p_i|}{\lambda} < \frac{\text{UB} + np_{\max}}{\lambda} \leq z_{\max}.$$

We next show that the restricted program reaches a terminal state whose scaled profit is $\hat{p}(S^*)$. Let $S_c^* = S^* \cap \mathcal{I}_c$. Every prefix of S_c^* in the item order of $\mathcal{I}_c$ is a subset of S^* , hence its scaled profit belongs to $\mathcal{Z}$ and its weight is at most b . No transition along the path that builds S_c^* in the inner program is therefore discarded. The inner table consequently contains a state with cardinality $|S_c^*|$, scaled profit $\hat{p}(S_c^*)$, and weight no greater than $w(S_c^*)$.

For $c = 0, 1, \dots, m$, let $T_c^* = \bigcup_{j=1}^c S_j^*$. Since $T_c^* \subseteq S^*$, its scaled profit belongs to $\mathcal{Z}$ and its weight is at most b . Induction on c now shows that the outer program retains a state at

$$\left(c, |T_c^*|, \max_{j \leq c} |S_j^*|, \hat{p}(T_c^*) \right)$$

of weight no greater than $w(T_c^*)$. The base case is the initialization at $c = 0$. For the inductive step, combine the retained state for T_{c-1}^* with the inner-table pair for S_c^* . The resulting profit and weight are those of T_c^* , so neither restriction discards the transition.

At $c = m$, the retained state has scaled profit $\hat{p}(S^*)$, cardinality $|S^*|$, dominant cardinality $d(S^*)$, and weight at most $w(S^*) \leq b$. Since S^* is color-feasible, it satisfies $2d(S^*) \leq |S^*| + 1$. Thus $\hat{p}(S^*)$ is among the terminal profit levels over which the algorithm maximizes. By Proposition 6.1, every retained terminal state represents an actual subset. Therefore the returned set $\hat{S}$ is feasible and

$$\hat{p}(\hat{S}) \geq \hat{p}(S^*). \quad (6.5)$$

Using (6.4) and (6.5), we conclude that

$$\begin{aligned} p(\hat{S}) &\geq \lambda \hat{p}(\hat{S}) \\ &\geq \lambda \hat{p}(S^*) \\ &> p(S^*) - |S^*|\lambda \\ &\geq \text{OPT} - n\lambda \\ &= \text{OPT} - \varepsilon \text{LB} \\ &\geq (1 - \varepsilon) \text{OPT}. \end{aligned}$$

□

Theorem 6.3. *Algorithm 6.2 runs in $O(n^5/\varepsilon^2)$ time when all profits are nonnegative and in $O(n^7/\varepsilon^2)$ time for arbitrary integer profits.*

Proof. For a nonempty color class $\mathcal{I}_c$, each item layer of the inner dynamic program has $O(|\mathcal{I}_c|\Pi)$ cardinality–profit states. Over the $|\mathcal{I}_c|$ item layers, the inner DP algorithm therefore takes $O(|\mathcal{I}_c|^2\Pi)$ time. Since $\sum_{c \in \mathcal{C}} |\mathcal{I}_c|^2 \leq n^2$, the whole inner phase takes $O(n^2\Pi)$ time. Each stage of the outer dynamic program has $O(n^2\Pi)$ states and, from each of them, considers the pairs of $\mathcal{S}_c$, which are at most $(|\mathcal{I}_c| + 1)\Pi = O(|\mathcal{I}_c|\Pi)$. Since every color class is nonempty by assumption, the total running time of the outer DP algorithm is

$$\sum_{c \in \mathcal{C}} O(n^2\Pi \cdot |\mathcal{I}_c|\Pi) = O(n^3\Pi^2),$$

which dominates the inner phase. For nonnegative profits, $\Pi = \lceil 3n/\varepsilon \rceil + 1 = \Theta(n/\varepsilon)$. For arbitrary integer profits, $\Pi = O(n^2/\varepsilon)$ by the same bounds as in Theorem 5.2. Substitution yields the stated running times.

□

The difference between the two approaches lies in how they process a color. The item-by-item DP algorithm adds one item at a time. It must therefore store the counter a for the number of selected items of the current color, but each state has only the two constant-time transitions *pack* and *do not pack*. If the pseudo-polynomial dimension has q levels, this gives the running time $O(n^4 q)$. The color-by-color program avoids the counter a in its outer table because the inner dynamic program has already summarized every possible packing of the next color by its cardinality, profit, and minimum weight. This removes a factor of at most n from the number of outer states. The price is that an outer state can no longer be updated by a constant number of choices: it

must be combined with all attainable profit levels for the next color. For fixed cardinalities, this combination is a convolution along the pseudo-polynomial dimension and contributes a factor q . The resulting running time is $O(n^3 q^2)$. Thus, the color-by-color recursion exchanges a factor n in the state space for a factor q in the transition cost. For the exact capacity-indexed programs, $q = b + 1$, so this exchange is advantageous when the capacity is small relative to n . In the FPTAS, however, $q = \Pi$: already for nonnegative profits, $\Pi = \Theta(n/\varepsilon)$, which is larger than the saved factor n by a factor of order $1/\varepsilon$. Accordingly, the running times are $O(n^5/\varepsilon)$ for the item-by-item scheme and $O(n^5/\varepsilon^2)$ for the color-by-color scheme. With arbitrary integer profits, $\Pi = O(n^2/\varepsilon)$, and the corresponding running times become $O(n^6/\varepsilon)$ and $O(n^7/\varepsilon^2)$, respectively. In this case the ratio between the two is $O(n/\varepsilon)$. Hence, the color-by-color recursion, that is useful for small capacities, offers no improvement once the scaled profit range becomes the pseudo-polynomial dimension.

7. Conclusions

In this paper, we studied the approximability of the ColKP, which generalizes the classical Knapsack Problem by adding a color-ordering requirement of the packed items. This requirement can make negative-profit items necessary in an optimal solution and complicates the use of standard knapsack approximation techniques. Our results show that the ColKP nevertheless admits both a constant-factor approximation algorithm and fully polynomial-time approximation schemes, even when item profits are arbitrary integers.

We began by investigating the structure of ColKP optimal solutions. In particular, we identified when negative-profit items can occur and bounded their number, providing the control over signed profits needed for the approximation schemes. We then developed a linear-time approximation algorithm based on the LP relaxation of the ColKP natural ILP formulation, having a worst-case performance ratio of $1/3$.

To obtain the FPTASs, we adapted the two exact dynamic programming algorithms of [11] to work with scaled profits. The first scheme, based on a item-by-item recursion, runs in $O(n^5/\varepsilon)$ time for nonnegative profits and $O(n^6/\varepsilon)$ time for arbitrary integer profits. The second, based instead on a color-by-color recursion, runs in $O(n^5/\varepsilon^2)$ and $O(n^7/\varepsilon^2)$ time, respectively. Thus, although both DP algorithms from the literature yield an FPTAS, the item-by-item recursion gives the better worst-case running times under the direct scaling approach considered here.

These results suggest two directions for further research. One is to reduce the running times of the FPTASs, particularly for instances with negative-profit items, where the signed profit range makes the DP algorithms more computationally expensive. A further potential investigation may concern the development of approximation algorithms with a performance guarantee stronger than $1/3$. Nonetheless, this would require to exploit the structure of color-feasible solutions in ways beyond the LP-rounding approach considered here.

References

- [1] János Balogh, József Békési, Gyorgy Dosa, Hans Kellerer, and Zsolt Tuza. 2013. Black and white bin packing. In *Approximation and Online Algorithms* (Lecture Notes in Computer Science), Thomas Erlebach and Giuseppe Persiano (Eds.). Springer, Berlin, Heidelberg, 131–144.
- [2] J. Balogh, J. Békési, G. Dósa, L. Epstein, H. Kellerer, A. Levin, Z. Tuza. Offline black and white bin packing. *Theoretical Computer Science*, 596:92–101, 2015. doi:10.1016/j.tcs.2015.06.045.

- [3] Y. G. F. Borges, R. C. S. Schouery, F. K. Miyazawa. Mathematical models and exact algorithms for the colored bin packing problem. *Computers & Operations Research*, 164:106527, 2024.
- [4] V. Cacchiani, M. Iori, A. Locatelli, S. Martello. Knapsack problems — an overview of recent advances. Part I: single knapsack problems. *Computers & Operations Research*, 143:105692, 2022.
- [5] A. Caprara, H. Kellerer, U. Pferschy, D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123(2):333–345, 2000.
- [6] E. D. Chajakis, M. Guignard. Exact algorithms for the setup knapsack problem. *INFOR: Information Systems and Operational Research*, 32(3):124–142, 1994.
- [7] A. K. Chandra, D. S. Hirschberg, C. K. Wong. Approximate algorithms for some generalized knapsack problems. *Theoretical Computer Science*, 3(3):293–304, 1976.
- [8] T. M. Chan. Approximation schemes for 0-1 knapsack. In *1st Symposium on Simplicity in Algorithms (SOSA 2018)*, 5:1–5:12, 2018.
- [9] L. Chen, J. Lian, Y. Mao, G. Zhang. A nearly quadratic-time FPTAS for knapsack. *SIAM Journal on Computing*, 2025.
- [10] K. Chebil, M. Khemakhem. A dynamic programming algorithm for the knapsack problem with setup. *Computers & Operations Research*, 64:40–50, 2015.
- [11] F. Ciccarelli, A. Helber, E. Mühmer. The colored knapsack problem: structural properties and exact algorithms. arXiv:2602.12214, 2026.
- [12] S. Coniglio, F. Furini, P. San Segundo. A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts. *European Journal of Operational Research*, 289(2):435–455, 2021.
- [13] M. Deng, C. Jin, X. Mao. Approximating knapsack and partition via dense subset sums. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA 2023)*, 2961–2979, 2023.
- [14] C. D’Ambrosio, F. Furini, M. Monaci, E. Traversi. On the product knapsack problem. *Optimization Letters*, 12(4):691–712, 2018.
- [15] F. Furini, M. Monaci, E. Traversi. Exact approaches for the knapsack problem with setups. *Computers & Operations Research*, 90:208–220, 2018.
- [16] O. H. Ibarra, C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.
- [17] C. Jin. An improved FPTAS for 0-1 knapsack. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, 76:1–76:14, 2019.
- [18] H. Kellerer, U. Pferschy. A new fully polynomial time approximation scheme for the knapsack problem. *Journal of Combinatorial Optimization*, 3(1):59–71, 1999.

- [19] H. Kellerer, U. Pferschy. Improved dynamic programming in connection with an FPTAS for the knapsack problem. *Journal of Combinatorial Optimization*, 8(1):5–11, 2004.
- [20] H. Kellerer, U. Pferschy, D. Pisinger. *Knapsack Problems*. Springer, Berlin Heidelberg, 2004.
- [21] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979.
- [22] M. J. Magazine, O. Oguz. A fully polynomial approximation algorithm for the 0-1 knapsack problem. *European Journal of Operational Research*, 8(3):270–273, 1981.
- [23] E. Malaguti, P. Paronuzzi, A. Santini. Algorithms and complexity results for the 0–1 knapsack problem with group fairness. *Optimization Letters*, 20:577–594, 2026.
- [24] Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM (JACM)*, 31(1):114–127, 1984.
- [25] M. Mucha, K. Węgrzycki, M. Włodarczyk. A subquadratic approximation scheme for partition. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, 70–88, 2019.
- [26] R. M. Nauss. The 0–1 knapsack problem with multiple-choice constraints. *European Journal of Operational Research*, 2(2):125–131, 1978.
- [27] U. Pferschy, J. Schauer. The knapsack problem with conflict graphs. *Journal of Graph Algorithms and Applications*, 13(2):233–249, 2009.
- [28] U. Pferschy, R. Scatamacchia. Improved dynamic programming and approximation results for the knapsack problem with setups. *International Transactions in Operational Research*, 25(2):667–682, 2018.
- [29] U. Pferschy, J. Schauer, C. Thielen. Approximating the product knapsack problem. *Optimization Letters*, 15:2529–2540, 2021.
- [30] S. Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM*, 22(1):115–124, 1975.