Optimal Macroitem Sequences in the Precedence Constrained Knapsack Problem

Valerio Dose
Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma
valerio.dose@uniroma1.it

Fabio Furini
Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma
fabio.furini@uniroma1.it

Marco Locatelli
Dipartimento di Ingegneria e Architettura, Università di Parma
marco.locatelli@unipr.it

Abstract

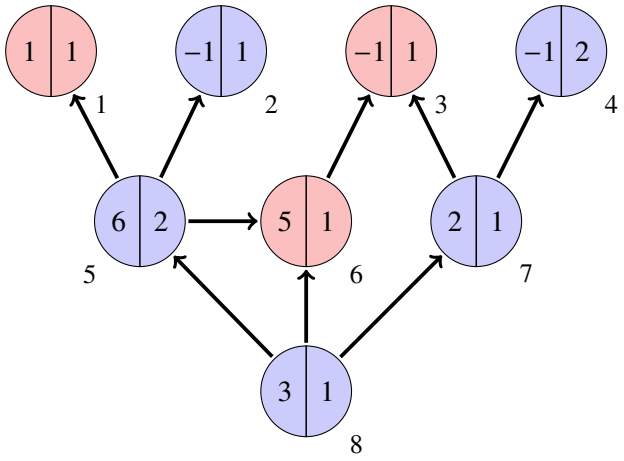
The Precedence Constrained Knapsack Problem (PCKP) asks for a maximum-profit subset of items, subject to a knapsack capacity constraint and precedence constraints encoded by a directed acyclic graph. We study the structure of optimal solutions of the Linear Programming (LP) relaxation of the natural Integer Linear Programming formulation of the PCKP. We introduce the notion of *macroitem* and of *feasible sequence of macroitems*, which partitions the item set while respecting the precedence structure. We establish that an optimal LP solution is fully characterized by the *optimal sequence of macroitems*: items are packed in nonincreasing order of the profit-to-weight ratio of their macroitem, with at most one macroitem fractionally included. We further show that the breakpoints of the parametric Lagrangian function of the capacity constraint coincide with the profit-to-weight ratios of the macroitems in the optimal sequence, and provide a complete combinatorial characterization of optimal dual solutions in terms of a feasible flow within each macroitem. Finally, for the special case in which the precedence graph is a forest, we devise an $O(n^2)$ algorithm to compute the optimal sequence, which improves to $O(n \log n)$ for in-trees or out-trees, where n denotes the number of items.

Keywords: Precedence Constrained Knapsack Problem, Linear Programming relaxation, Lagrangian relaxation, Forests

1 Introduction

Let $\mathcal{G} := (\mathcal{I}, \mathcal{A})$ be a directed acyclic graph where $\mathcal{I} := \{1, 2, \ldots, n\}$ is a set of n items, which are the vertices of $\mathcal{G}$, and $\mathcal{A}$ is its set of m arcs. An arc $(i, j) \in \mathcal{A}$ models the fact that item j precedes

Figure 1: A directed acyclic graph associated to a PCKP instance with $n = 8$ items and $m = 9$ arcs. The index i of an item is indicated next to the associated item. For each item $i \in \{1, 2, \ldots, 8\}$, the profit p_i is shown on the left side of the associated item, while the weight w_i is shown on the right side; the two values are separated by a vertical line. An optimal PCKP solution with knapsack capacity $c = 4$ is the subset of items $\{1, 3, 6\}$, whose items are colored in red. The optimal solution value is 5 given by the sum of the item profits p_1, p_3 and p_6.



item i, i.e., item i can be selected only if item j is selected as well. Each item $i \in \mathcal{I}$ is associated to a (possibly negative) integer profit $p_i \in \mathbb{Z}$ and a positive integer weight $w_i \in \mathbb{Z}_{>0}$. Finally, let $c \in \mathbb{Z}_{>0}$ be the positive integer capacity of a knapsack. Given these input data, the *Precedence Constrained Knapsack Problem* (PCKP) asks for selecting a subset of items with maximum total profit such that their total weight does not exceed the knapsack capacity and they satisfy the precedence constraints induced by the arcs of the graph $\mathcal{G}$. Fig. 1 shows a PCKP instance and an optimal solution of the problem.

The PCKP is NP-complete ([11, 17]) and it is a relevant generalization of the classical *Knapsack Problem* (KP). The reader is referred to [18, 20] for comprehensive books on models and algorithms of the classical KP and its variants. The PCKP arises naturally in many applications like investment planning ([15]), production planning ([16, 26]) and network design ([25]). Moreover, in the context of open pit mining problems, the PCKP plays an important role, since it corresponds to the *Constrained Pit Limit Problem* (CPIT) ([7, 9]) with a single resource and a single time period.

The relaxed PCKP without the capacity constraint is known in the literature as the *Maximum Closure Problem* (MCP), see e.g., [22]. This problem asks for selecting a maximum-profit subset of items such that they satisfy the precedence constraints. The MCP is also referred to by different names in the mining literature such as *Ultimate Pit Limit Problem* (UPIT) [7] or the *Open-Pit Mining Problem* [14]. The MCP can be solved in polynomial time and efficient specialized algorithms have been developed. The related literature is extensive, we refer the interested reader to, e.g.,

[2, 3, 8, 12, 14, 19].

We assume that $\mathcal{G}$ does not contain directed cycles, since all items in such a cycle can be merged into a single one. We also assume that the knapsack capacity is strictly smaller than the total weight of all items. Otherwise, the capacity constraint is redundant, and the PCKP reduces to the MCP.

In this paper, we develop a novel way of interpreting the structure of parametric optimal solutions of the *Linear Programming* (LP) relaxation of the natural *Integer Linear Programming* (ILP) formulation for the PCKP when varying the capacity. The mathematical structure of such solutions is known, see, e.g., [7, 19], and it is based on the knowledge of a parametric solution of the MCP. Our results reveal a new interpretation that links the structure of optimal LP solutions for the PCKP and the KP. Indeed, in optimal LP solutions of the KP, items are selected to be put in the knapsack following their nonincreasing *efficiencies* given by the profit over weight ratio. One of the contributions of this work is to show that an analogous result holds for the PCKP: subsets of items, which we call *macroitems*, are selected simultaneously in nonincreasing order of efficiency, defined as the ratio of their total profit to their total weight. The subsets of items can be obtained by maximizing this ratio while maintaining the relative precedences satisfied. We also characterize the structure of optimal dual solutions, thus providing a novel combinatorial interpretation for both primal and dual LP solutions. The optimal sequence of macroitems can be computed through parametric pseudoflow methods in $O(mn \log n)$ time [13], which for the special case of forests becomes $O(n^2 \log n)$. We improve this bound by devising an $O(n^2)$ algorithm for forests, and two $O(n \log n)$ algorithms for the cases of in-forests or out-forests, i.e., forests where all items have in-degree or out-degree not larger than one, respectively. We also provide computational evidence that these improvements translate into practical performance gains over the parametric pseudoflow approach.

The remainder of the paper is organized as follows. In Section 2 we present the natural ILP formulation for the PCKP and its LP relaxation. We then collect preliminary results on a parametric version of MCP. Finally, we introduce a related precedence-constrained ratio optimization problem. In Section 3 we introduce feasible and optimal sequences of macroitems and relate them to the breakpoints of the parametric closure value function. In Section 4 we show how the optimal sequence of macroitems determines primal and dual optimal solutions of the LP relaxation. In Section 5 we present the algorithms for directed forests and their complexity analysis. In Section 6 we describe the generation of the benchmark instances and report the computational comparison with the parametric pseudoflow approach. In Section 7 we summarize the main results and outline possible directions for future research.

2 The natural ILP formulation of the PCKP and its relaxations

For each item $i \in \mathcal{I}$, let us introduce a binary variable $x_i \in \{0, 1\}$ taking value 1 if and only if item i is selected in the knapsack. Using these n binary variables, the natural ILP formulation for the PCKP reads as follows:

$$\max_{\boldsymbol{x} \in \{0,1\}^{\mathcal{I}}} \left\{ \sum_{i \in \mathcal{I}} p_i x_i : \quad \sum_{i \in \mathcal{I}} w_i x_i \leq c, \quad x_i - x_j \leq 0, \ \ (i, j) \in \mathcal{A} \right\}. \tag{1}$$

The objective function in (1) maximizes the total profit of the selected items. The *capacity constraint* in (1) imposes that the total weight of the selected items is no larger than the knapsack capacity. The *precedence constraints* in (1) impose respecting the precedence relationships between pairs of items, that is, if there is an arc $(i, j) \in \mathcal{A}$, item i can be selected only if item j is selected. The formulation (1) is called $\mathscr{F}$ in the remainder of this manuscript. Moreover, we denote by $\zeta(\mathscr{F})$ the optimal solution value of $\mathscr{F}$, i.e., the optimal value of the PCKP.

The $\mathscr{F}$ formulation has been studied mainly from a polyhedral perspective. In particular, [4] investigates the polyhedron associated with the PCKP and introduces clique-based facet-defining inequalities that strengthen the natural formulation. The work of [10] focuses instead on separation, developing procedures for finding maximally violated valid inequalities and showing how these cuts can be used to improve exact solution methods for the PCKP.

2.1 Linear programming relaxation and its dual problem

By replacing the binary variables in (1) with the following continuous variables:

$$x_i \in [0, 1], \quad i \in \mathcal{I},$$

we obtain the LP relaxation of $\mathscr{F}$ which we denote by $\mathscr{F}_{\text{LP}}$. Moreover, we denote by $\zeta(\mathscr{F}_{\text{LP}})$ its optimal value that provides a valid upper bound on $\zeta(\mathscr{F})$. In this paper, we are interested in determining the parametric optimal solutions to $\mathscr{F}_{\text{LP}}$, for every possible capacity value $c > 0$.

Using a non-negative dual variable λ for the capacity constraint, a non-negative dual variable α_{ij} for each precedence constraint corresponding to an arc $(i, j) \in \mathcal{A}$, and, for each item $i \in \mathcal{I}$, a non-negative dual variable μ_i for each less than or equal to one constraint, the dual problem of $\mathscr{F}_{\text{LP}}$ reads as follows:

$$\min_{\lambda \geq 0, \boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\mu} \geq \mathbf{0}} \quad c\,\lambda + \sum_{i \in \mathcal{I}} \mu_i \tag{2a}$$

$$w_i\,\lambda + \mu_i + \sum_{j \in \mathcal{I}^+(i)} \alpha_{ij} - \sum_{k \in \mathcal{I}^-(i)} \alpha_{ki} \geq p_i, \qquad i \in \mathcal{I}. \tag{2b}$$

For each item $i \in \mathcal{I}$, the set $\mathcal{I}^+(i) := \{j \in \mathcal{I} : (i,j) \in \mathcal{A}\}$ is the set of its *out-neighbors*, and the set $\mathcal{I}^-(i) := \{k \in \mathcal{I} : (k,i) \in \mathcal{A}\}$ is the set of its *in-neighbors*. The value of a variable α_{ij} can be interpreted as a flow passing through the associated arc $(i,j) \in \mathcal{A}$ and, accordingly, $\sum_{j \in \mathcal{I}^+(i)} \alpha_{ij}$ and $\sum_{k \in \mathcal{I}^-(i)} \alpha_{ki}$ become the outgoing and incoming total flows of an item $i \in \mathcal{I}$, respectively. Within this interpretation, constraints (2b) impose for each item $i \in \mathcal{I}$ that the difference between the outgoing and incoming total flows must be greater than or equal to the profit p_i of the item minus λ times the weight w_i of the item and the value of the variable μ_i.

We denote the LP formulation (2) by $\mathscr{F}_{\mathrm{D(LP)}}$ in the remainder of this manuscript. Moreover, we denote by $\zeta(\mathscr{F}_{\mathrm{D(LP)}})$ its optimal value, and, by the LP strong duality theorem, we have $\zeta(\mathscr{F}_{\mathrm{LP}}) = \zeta(\mathscr{F}_{\mathrm{D(LP)}})$.

2.2 Parametric maximum closure problem

An explicit solution of $\mathscr{F}_{\mathrm{LP}}$ is known since [19], and it is also derived in [7, Proposition 3.1], where it is applied in relation with problems in mine production planning. This solution is based on the determination of the function $u\colon [0,+\infty) \to \mathbb{R}$ given by

$$u(\lambda) := \max_{\boldsymbol{x} \in \{0,1\}^{\mathcal{I}}} \left\{ \sum_{i \in \mathcal{I}} (p_i - \lambda\, w_i)\, x_i : \quad x_i - x_j \leq 0, \quad (i,j) \in \mathcal{A} \right\}, \tag{3}$$

which is related to the Lagrangian relaxation of the capacity constraint in $\mathscr{F}$. For a fixed λ problem (3) is known in the literature as the (maximum) closure problem on the graph $\mathcal{G}$, which can be reduced to a max-flow/min-cut problem on an extended graph (see, for example, [12, Section 2.2]).

We denote with $\mathcal{P}$ the set of feasible solutions of (3). Every feasible solution $\boldsymbol{x}$ of (3) defines an affine function with equation

$$f_{\boldsymbol{x}}(\lambda) := P(\boldsymbol{x}) - \lambda\, W(\boldsymbol{x}),$$

where, for any $\boldsymbol{x} \in \{0,1\}^n$, we have

$$P(\boldsymbol{x}) := \sum_{i \in \mathcal{I}} p_i x_i, \qquad W(\boldsymbol{x}) := \sum_{i \in \mathcal{I}} w_i x_i.$$

With abuse of notation, for any $M \subset \mathcal{I}$ we also write

$$P(M) := \sum_{i \in M} p_i, \qquad W(M) := \sum_{i \in M} w_i.$$

Note that $f_{\boldsymbol{x}}$ is strictly decreasing if $\boldsymbol{x} \neq 0$ because weights are always positive. Since the set $\mathcal{P}$ has finite cardinality, for every $\lambda \geq 0$ the value $u(\lambda)$ is defined as the largest of the finite number of values $f_{\boldsymbol{x}}(\lambda)$, i.e.,

$$u(\lambda) = \max_{\boldsymbol{x} \in \mathcal{P}} f_{\boldsymbol{x}}(\lambda).$$

Then, we have that the function u is a nonincreasing piecewise-affine convex function with a finite number of breakpoints. We indicate the breakpoints with the symbols

$$\lambda_0 = +\infty > \lambda_1 > \lambda_2 > \cdots > \lambda_k > \lambda_{k+1} = 0.$$

Breakpoints λ_r, with $r \in \{1, 2, \ldots, k\}$, are λ values such that the set

$$X^\star(\lambda) = \{\boldsymbol{x} \in \mathcal{P} \ : \ u(\lambda) = f_{\boldsymbol{x}}(\lambda)\},$$

of optimal solutions of problem (3) is not a singleton.

To compute the function u and its breakpoints, one can systematically search for its distinct affine components by solving problem (3) for specific values of λ. At these fixed values, the formulation simplifies into a standard non-parametric graph closure problem. This strategy is applied, for instance, in the implementation of the algorithms developed by [7] for the Open-Pit Mine Production Scheduling Problem. Alternative approaches leverage algorithms designed for the general parametric max-flow/min-cut problem, with the one presented in [13] representing the state of the art.

The next observation records the closure property of the feasible set of the parametric closure problem (3), which is the basic order-theoretic ingredient used to compare its optimal solutions.

Proposition 1. *Given the set $\mathcal{P}$ of feasible solutions of (3), we have that $(\mathcal{P}, \wedge, \vee)$ is a lattice, where $\wedge$ and $\vee$ are the logical and and logical or between binary vectors, respectively.*

Proof. Proof. To prove that $\mathcal{P}$ is a lattice we need to show that $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{P}$ implies that $\boldsymbol{x} \vee \boldsymbol{y} \in \mathcal{P}$ and that $\boldsymbol{x} \wedge \boldsymbol{y} \in \mathcal{P}$. If we assume by contradiction that $\boldsymbol{x} \vee \boldsymbol{y} \notin \mathcal{P}$, then $\exists\ (i, j) \in \mathcal{A}$ such that $(\boldsymbol{x} \vee \boldsymbol{y})_i = 1$ and $(\boldsymbol{x} \vee \boldsymbol{y})_j = 0$. But if $(\boldsymbol{x} \vee \boldsymbol{y})_i = 1$, then either $x_i = 1$ or $y_i = 1$, while $(\boldsymbol{x} \vee \boldsymbol{y})_j = 0$ implies that $x_j = y_j = 0$. But due to the fact that $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{P}$, we must have that $x_j = 1$ or $y_j = 1$. Similarly, if we assume by contradiction that $\boldsymbol{x} \wedge \boldsymbol{y} \notin \mathcal{P}$, then $\exists\ (i, j) \in \mathcal{A}$ such that $(\boldsymbol{x} \wedge \boldsymbol{y})_i = 1$

and $(\boldsymbol{x} \wedge \boldsymbol{y})_j = 0$. But $(\boldsymbol{x} \wedge \boldsymbol{y})_i = 1$ implies that $x_i = y_i = 1$, while $(\boldsymbol{x} \wedge \boldsymbol{y})_j = 0$ implies $x_j = 0$ or $y_j = 0$. But $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{P}$ implies $x_j = y_j = 1$. □

As already observed, each breakpoint λ_r is such that the set $X^\star(\lambda_r)$ of optimal solutions of problem (3) for $\lambda = \lambda_r$ is not a singleton. Note that if $z \in X^\star(\lambda_r)$, then the optimal value of (3) is $f_z(\lambda_r) = P(z) - \lambda_r W(z)$. The next result shows that the lattice structure discussed in Proposition 1 is inherited by the set of optimal solutions of the parametric closure problem at a breakpoint.

Proposition 2. *Let $z^1, z^2 \in X^\star(\lambda_r)$. Then, $z^1 \wedge z^2, z^1 \vee z^2 \in X^\star(\lambda_r)$.*

Proof. Proof. If $z^1 \leq z^2$, then $z^1 \wedge z^2 = z^1$ and $z^1 \vee z^2 = z^2$, so that the result is true. Similarly, if $z^1 \geq z^2$. Therefore, let us assume that $z^1 \not\leq z^2$ and $z^1 \not\geq z^2$. First of all, in view of Proposition 1, we have that $z^1 \vee z^2, z^1 \wedge z^2 \in \mathcal{P}$. Now, let us set

$$q(\lambda_r) = \sum_{j \in \mathcal{M}(z^1) \setminus \mathcal{M}(z^2)} (p_j - \lambda_r w_j).$$

If $q(\lambda_r) > 0$, we have that

$$\begin{aligned} &P(z^1 \vee z^2) - \lambda_r W(z^1 \vee z^2) = \\ &= P(z^2) - \lambda_r W(z^2) + q(\lambda_r) > P(z^2) - \lambda_r W(z^2), \end{aligned} \tag{4}$$

which contradicts the optimality of z^2. If $q(\lambda_r) < 0$, we have that

$$\begin{aligned} &P(z^1 \wedge z^2) - \lambda_r W(z^1 \wedge z^2) = \\ &= P(z^1) - \lambda_r W(z^1) - q(\lambda_r) > P(z^1) - \lambda_r W(z^1), \end{aligned} \tag{5}$$

which contradicts the optimality of z^1.

Therefore, only $q(\lambda_r) = 0$ is possible. But in this case (4) and (5) show that $z^1 \vee z^2$ and $z^1 \wedge z^2$ are also both optimal for (3) as we wanted to prove. □

Now, for each breakpoint λ_r we set

$$\begin{aligned} \boldsymbol{x}^{r-1} &= \bigwedge_{z \in X^\star(\lambda_r)} z \\ \boldsymbol{x}^{r} &= \bigvee_{z \in X^\star(\lambda_r)} z, \end{aligned}$$

i.e., $\boldsymbol{x}^{r-1}$ is the minimal optimal solution in $X^\star(\lambda_r)$, while $\boldsymbol{x}^r$ is the maximal optimal solution in $X^\star(\lambda_r)$. Obviously $\boldsymbol{x}^r \geq \boldsymbol{x}^{r-1}$. Noticing that $q(\lambda)$ is decreasing, then the maximal solution $\boldsymbol{x}^r$ will be the unique optimal solution of (3) for $\lambda \in (\lambda_{r+1}, \lambda_r)$, while the minimal solution $\boldsymbol{x}^{r-1}$ will be the

unique optimal solution of (3) for $\lambda \in (\lambda_r, \lambda_{r-1})$. Stated in another way, we have that $u(\lambda) = f_{\boldsymbol{x}^r}(\lambda)$ for $\lambda \in (\lambda_{r+1}, \lambda_r)$, while $u(\lambda) = f_{\boldsymbol{x}^{r-1}}(\lambda)$ for $\lambda \in (\lambda_r, \lambda_{r-1})$. Note that the breakpoint λ_r is the intersection of the two lines $f_{\boldsymbol{x}^{r-1}}(\lambda)$ and $f_{\boldsymbol{x}^r}(\lambda)$, so that

$$\lambda_r = \frac{P(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})}{W(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})}. \tag{6}$$

The following theorem gives the main interpretation of each breakpoint: it is the best profit-to-weight ratio that can be obtained by adding a new feasible block of items.

Theorem 1. *For every $r \in \{1, 2, \ldots, k\}$, the vector $\boldsymbol{x}^r - \boldsymbol{x}^{r-1}$ is the optimal solution with largest support of the binary linear fractional problem*

$$\begin{aligned} \max_{\boldsymbol{x} \in \{0,1\}^I} \quad & \frac{P(\boldsymbol{x})}{W(\boldsymbol{x})} \\ & x_i - x_j \leq 0, \quad (i,j) \in \mathcal{A} \text{ such that } (\boldsymbol{x}^{r-1})_i = (\boldsymbol{x}^{r-1})_j = 0, \\ & x_i = 0, \qquad\quad i \text{ such that } (\boldsymbol{x}^{r-1})_i = 1, \\ & \textstyle\sum_{i \in I} x_i \geq 1, \end{aligned} \tag{7}$$

where the last constraint guarantees that the objective function is always defined in a feasible point. Equivalently, in view of (6), the breakpoint λ_r is the optimal value of the above problem.

Proof. Proof. Let us assume by contradiction that there exists $\boldsymbol{x}^t \in \mathcal{P}$ with $\boldsymbol{x}^t - \boldsymbol{x}^{r-1}$ feasible for (7) and such that

$$\frac{P(\boldsymbol{x}^t - \boldsymbol{x}^{r-1})}{W(\boldsymbol{x}^t - \boldsymbol{x}^{r-1})} > \frac{P(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})}{W(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})}. \tag{8}$$

We prove that (8) implies

$$f_{\boldsymbol{x}^t}(\lambda_r) = P(\boldsymbol{x}^t) - \lambda_r W(\boldsymbol{x}^t) > P(\boldsymbol{x}^r) - \lambda_r W(\boldsymbol{x}^r) = f_{\boldsymbol{x}^r}(\lambda_r), \tag{9}$$

which contradicts the optimality of $\boldsymbol{x}^r$ for (3). To see this, we first recall that

$$\lambda_r = \frac{P(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})}{W(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})}, \tag{10}$$

so that

$$\begin{aligned} P(\boldsymbol{x}^t) - \lambda_r W(\boldsymbol{x}^t) &= P(\boldsymbol{x}^t) - \tfrac{P(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})}{W(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})} W(\boldsymbol{x}^t) \\ P(\boldsymbol{x}^r) - \lambda_r W(\boldsymbol{x}^r) &= P(\boldsymbol{x}^r) - \tfrac{P(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})}{W(\boldsymbol{x}^r - \boldsymbol{x}^{r-1})} W(\boldsymbol{x}^r). \end{aligned}$$

Then, after setting $z^r = \boldsymbol{x}^r - \boldsymbol{x}^{r-1}$, we have that:

$$\begin{aligned} & W(z^r)\left\{[P(\boldsymbol{x}^t) - \lambda_r W(\boldsymbol{x}^t)] - [P(\boldsymbol{x}^r) - \lambda_r W(\boldsymbol{x}^r)]\right\} = \\ & = \left[P(\boldsymbol{x}^t)W(z^r) - W(\boldsymbol{x}^t)P(z^r) - P(\boldsymbol{x}^r)W(z^r) + W(\boldsymbol{x}^r)P(z^r)\right] = \\ & = \left[P(\boldsymbol{x}^t)W(z^r) - W(\boldsymbol{x}^t)P(z^r) + P(\boldsymbol{x}^r)W(\boldsymbol{x}^{r-1}) - P(\boldsymbol{x}^{r-1})W(\boldsymbol{x}^r)\right]. \end{aligned} \tag{11}$$

We can rewrite (8) as follows:

$$P(\boldsymbol{x}^t)W(z^r) - P(\boldsymbol{x}^{r-1})W(z^r) > P(\boldsymbol{x}^r)W(\boldsymbol{x}^t - \boldsymbol{x}^{r-1}) - P(\boldsymbol{x}^{r-1})W(\boldsymbol{x}^t - \boldsymbol{x}^{r-1}),$$

and then also as:

$$P(\boldsymbol{x}^t)W(z^r) - W(\boldsymbol{x}^t)P(z^r) + P(\boldsymbol{x}^r)W(\boldsymbol{x}^{r-1}) - P(\boldsymbol{x}^{r-1})W(\boldsymbol{x}^r) > 0,$$

which, in view of the last equation in (11) and of $W(z^r) > 0$, implies (9). □

Remark 1. Notice that if the subset of items associated to the last vector $\boldsymbol{x}^k$ is not the whole set $\mathcal{I}$, then Problem (7) has feasible solutions also choosing $r = k + 1$, since the constraints depend only on $\boldsymbol{x}^{r-1}$. The optimal value in that case would be negative, and it would not be associated to an affine piece of the function u. This allows us to extend the sequence $\boldsymbol{x}^r$ until the last element of the sequence is always associated with the full set of items $\mathcal{I}$. Accordingly, the sequence $\boldsymbol{\lambda}$ of breakpoints can be extended to include some negative values computed according to Eq. (6). For the purpose of our exposition, we consider these extended sequences and we define, for every index r in these sequences, the subsets $\mathcal{M}_r$ as

$$\mathcal{M}_r := \{i \in \mathcal{I} : (\boldsymbol{x}^r)_i = 1\}.$$

The last observation in this section is a simple ratio property for disjoint sets, used later to compare unions of candidate macroitems.

Proposition 3. *Let $M_1, M_2 \subset \mathcal{I}$ be nonempty. If M_1 and M_2 are disjoint, i.e., $M_1 \cap M_2 = \emptyset$, then it cannot hold that*

$$\frac{P(M_1 \cup M_2)}{W(M_1 \cup M_2)} > \frac{P(M_1)}{W(M_1)}, \frac{P(M_2)}{W(M_2)} \text{ or } \frac{P(M_1 \cup M_2)}{W(M_1 \cup M_2)} < \frac{P(M_1)}{W(M_1)}, \frac{P(M_2)}{W(M_2)}, \tag{12}$$

equivalently, $\frac{P(M_1\cup M_2)}{W(M_1\cup M_2)} \leq \max\left\{\frac{P(M_1)}{W(M_1)}, \frac{P(M_2)}{W(M_2)}\right\}$ *and* $\frac{P(M_1\cup M_2)}{W(M_1\cup M_2)} \geq \min\left\{\frac{P(M_1)}{W(M_1)}, \frac{P(M_2)}{W(M_2)}\right\}$. *Moreover,*

$$\frac{P(M_1 \cup M_2)}{W(M_1 \cup M_2)} \geq (or\ \leq) \frac{P(M_1)}{W(M_1)}, \frac{P(M_2)}{W(M_2)} \Rightarrow \frac{P(M_1 \cup M_2)}{W(M_1 \cup M_2)} = \frac{P(M_1)}{W(M_1)} = \frac{P(M_2)}{W(M_2)}. \tag{13}$$

Proof. Proof. Let us assume by contradiction that the first inequality in (12) holds (the proof for the second inequality is analogous). Then, since $M_1 \cap M_2 = \emptyset$:

$$\frac{P(M_1 \cup M_2)}{W(M_1 \cup M_2)} = \frac{P(M_1) + P(M_2)}{W(M_1) + W(M_2)} > \frac{P(M_1)}{W(M_1)}, \frac{P(M_2)}{W(M_2)}.$$

But

$$\begin{aligned} \tfrac{P(M_1)+P(M_2)}{W(M_1)+W(M_2)} > \tfrac{P(M_1)}{W(M_1)} &\Rightarrow \tfrac{P(M_2)}{W(M_2)} > \tfrac{P(M_1)}{W(M_1)} \\ \tfrac{P(M_1)+P(M_2)}{W(M_1)+W(M_2)} > \tfrac{P(M_2)}{W(M_2)} &\Rightarrow \tfrac{P(M_2)}{W(M_2)} < \tfrac{P(M_1)}{W(M_1)}, \end{aligned}$$

which is not possible. In a completely similar way we can also prove (13). □

2.3 A related ratio optimization problem

Let us also introduce the following precedence-constrained ratio optimization problem, which corresponds to the MCP with a fractional objective function. The problem reads as follows:

$$\max_{\boldsymbol{x}\in\{0,1\}^{\mathcal{I}}} \left\{ \frac{\sum_{i\in\mathcal{I}} p_i x_i}{\sum_{i\in\mathcal{I}} w_i x_i} : \ x_i - x_j \leq 0,\ (i,j) \in \mathcal{A}, \quad \sum_{i\in\mathcal{I}} w_i x_i > 0 \right\}. \tag{14}$$

In this model, the coefficient p_i can be seen as the return, benefit, or revenue generated by item i, while w_i may represent the amount of capital, budget, or resource consumption required to select it. Thus, the objective maximizes the return per unit of invested resource over all feasible solutions satisfying the precedence constraints. The strict positivity condition on the denominator simply excludes the empty solution. Ratio objectives of this type are classical in fractional programming [6, 23] and arise naturally when the goal is to maximize an efficiency measure. In finance, for instance, related ratios are used to compare returns with the amount of capital or risk required to obtain them, as in return-on-investment and risk-adjusted performance criteria [24]. Problem (14) is a combinatorial problem in its own right, and the results of this paper can be used to solve it efficiently.

3 Macroitems

In order to show the combinatorial structure of an optimal solution of $\mathscr{F}_{\mathrm{LP}}$, we introduce a few definitions.

Definition 1. A *macroitem* is a subset $\mathcal{M} \subset \mathcal{I}$ of items.

In what follows we denote with $\boldsymbol{x}^{\mathcal{M}}$ the binary vector corresponding to the indicator of set $\mathcal{M}$, i.e., $\left(\boldsymbol{x}^{\mathcal{M}}\right)_i = 1$ if and only if $i \in \mathcal{M}$. We introduce the following definition.

Definition 2. A *feasible sequence $\mathcal{S}$ of macroitems* is an ordered partition of the item set $\mathcal{I}$ into $k(\mathcal{S})$ macroitems

$$\mathcal{S} = \left(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})}\right)$$

such that, for each $r \in \{1, 2, \ldots, k(\mathcal{S})\}$ and each $j \in \mathcal{I}_r$, all out-neighbors of j belong to preceding macroitems or to the macroitem itself, i.e.,

$$\mathcal{I}^+(j) \subset \bigcup_{s=1}^{r} \mathcal{I}_s, \tag{15}$$

where $\mathcal{I}^+(j)$ is the set of out-neighbors of item j in graph $\mathcal{G}$. Stated in another way, a sequence is feasible if for each $r \in \{1, 2, \ldots, k(\mathcal{S})\}$ it holds that the binary vector $\boldsymbol{x}^{\cup_{s=1}^{r} \mathcal{I}_s} \in \mathcal{P}$, i.e., it is feasible for (3).

For each macroitem $\mathcal{I}_r$ in a feasible sequence, the profit P_r and the weight W_r are defined as

$$P_r := P(\mathcal{I}_r) = \sum_{i \in \mathcal{I}_r} p_i, \qquad W_r := W(\mathcal{I}_r) = \sum_{i \in \mathcal{I}_r} w_i.$$

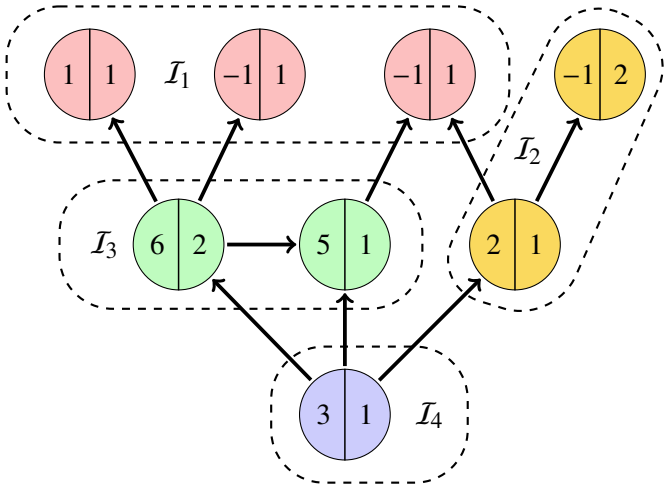
We introduce the notion of *split macroitem* of a feasible sequence.

Definition 3. Given a feasible sequence of macroitems $\mathcal{S} = \left(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})}\right)$ and the capacity c of a knapsack, the *split macroitem* of $\mathcal{S}$ is the macroitem $\mathcal{I}_{h(\mathcal{S})}$ such that

$$\sum_{r=1}^{h(\mathcal{S})-1} W_r \leq c, \qquad \sum_{r=1}^{h(\mathcal{S})} W_r > c.$$

By the standing assumption on the capacity, the split macroitem exists for every feasible sequence. Given a feasible sequence of macroitems, we can construct a feasible solution of $\mathscr{F}_{\mathrm{LP}}$ as

Figure 2: The sequence $\mathcal{S} = (\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4)$ is a feasible sequence of macroitems. With capacity $c = 4$ the split macroitem of $\mathcal{S}$ is $\mathcal{I}_2$ and the associated solution of maximum capacity assigns $x_i = 1$ if $i \in \mathcal{I}_1$, $x_i = \frac{1}{3}$ if $i \in \mathcal{I}_2$, $x_i = 0$ if $i \in \mathcal{I}_3 \cup \mathcal{I}_4$, for a total profit of $-\frac{2}{3}$. The sequence $\mathcal{S}' = (\mathcal{I}_1, \mathcal{I}_3, \mathcal{I}_2, \mathcal{I}_4)$ is also feasible, and would give us an associated solution of total profit $\frac{8}{3}$.



follows:

$$x_i = \begin{cases} 1, & i \in \mathcal{I}_r, \quad r \in \{1, 2, \ldots, h(\mathcal{S}) - 1\}, \\ \dfrac{c - \sum_{s=1}^{h(\mathcal{S})-1} W_s}{W_{h(\mathcal{S})}}, & i \in \mathcal{I}_{h(\mathcal{S})}, \\ 0, & i \in \mathcal{I}_r, \quad r \in \{h(\mathcal{S}) + 1, h(\mathcal{S}) + 2, \ldots, k(\mathcal{S})\}. \end{cases}$$

In other words, we take entire macroitems of the ordered sequence up to the split macroitem, we put only a fraction of the split macroitem (the fraction needed to fill the capacity), and we do not take all the other macroitems. A feasible sequence of macroitems and its associated solution of maximum capacity for the example in Fig. 1, is represented in Fig. 2. Note that the feasible solution found is very bad (it has negative profit) and it could easily be improved by just reordering the sequence of macroitems, while keeping it feasible. For this reason it is useful to introduce the following definitions.

Definition 4. A *nonincreasing sequence of macroitems* $\mathcal{S} := (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})})$ is a feasible sequence such that

$$r < s \Rightarrow \frac{P_r}{W_r} \geq \frac{P_s}{W_s},$$

where for any macroitem $\mathcal{I}_r$, we call $\frac{P_r}{W_r}$ the ratio of the macroitem. If

$$r < s \Rightarrow \frac{P_r}{W_r} > \frac{P_s}{W_s},$$

then we call the sequence a *decreasing sequence* of macroitems.

Observing that

$$\frac{P_r}{W_r} = \frac{P_{r+1}}{W_{r+1}} \quad \Rightarrow \quad \frac{P_r}{W_r} = \frac{P_{r+1} + P_r}{W_{r+1} + W_r},$$

any nonincreasing sequence $\mathcal{S}$ can always be replaced by a decreasing sequence by replacing each pair of macroitems with the same ratio with the union of the two macroitems.

Now we can define the sequences of macroitems which allow us to compute optimal solutions of $\mathscr{F}_{\mathrm{LP}}$.

Definition 5. An *optimal sequence of macroitems* $\mathcal{S}$ is a feasible sequence $(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})})$, which is also maximal in the lexicographic order induced by the order $\succ$ on macroitems defined by stating that $\mathcal{M} \succ \mathcal{M}'$ if

- the ratio of $\mathcal{M}$ is strictly larger than the ratio of $\mathcal{M}'$ or
- $\mathcal{M}$ and $\mathcal{M}'$ have the same ratio and $\mathcal{M}$ has a strictly larger weight than $\mathcal{M}'$.

Remark 2. Note that an optimal sequence of macroitems exists and it is unique. It is also decreasing by definition.

The following result gives us the interpretation of the breakpoints of the function u, as the profit/weight ratios associated to the macroitems in the optimal sequence with positive ratio. Recall that $\mathcal{M}_1, \ldots, \mathcal{M}_k$ are the subsets with indicator vectors $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k$, obtained by the iterative solution of Problem (7). Following Remark 1, the sequence is extended to include also macroitems with negative profit. The first vectors of this sequence, more precisely, those with positive profit, are also those giving the affine pieces of the function u as explained in Theorem 1 and Remark 1.

Theorem 2. *The sequence* $\mathcal{S} := (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k)$ *where*

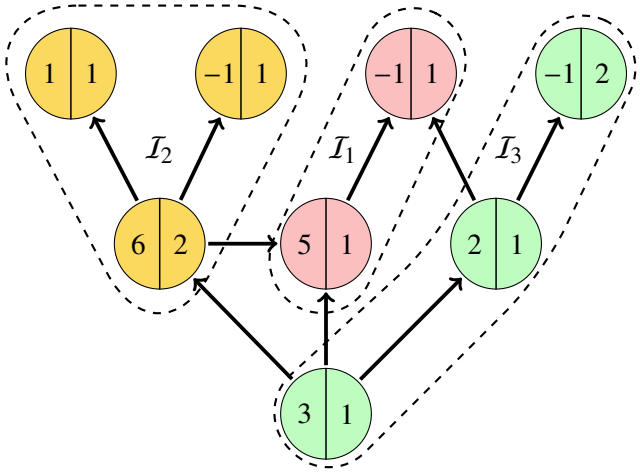
$$\mathcal{I}_1 = \mathcal{M}_1, \qquad \mathcal{I}_r = \mathcal{M}_r \setminus \mathcal{M}_{r-1} \quad \text{for } r \in \{2, 3, \ldots, k\},$$

is the optimal sequence of macroitems.

Proof. Proof. We need to show that the sequence $\mathcal{S}$ is feasible and optimal.

- The sequence is feasible, since for each $r \in \{1, 2, \ldots, k\}$, we have that $\boldsymbol{x}^{\cup_{s=1}^{r} \mathcal{I}_s} = \boldsymbol{x}^r$, and $\boldsymbol{x}^r \in \mathcal{P}$.
- The sequence is optimal because the ratio of a macroitem $\mathcal{I}_r$ is equal to the breakpoint λ_r as computed in (6) and Remark 1, and for all $r \in \{2, 3, \ldots, k\}$ it holds that $\lambda_r < \lambda_{r-1}$.

Figure 3: The optimal sequence of macroitems $(\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3)$. In this example, the optimal solution of $\mathscr{F}_{\text{LP}}$, with capacity $c = 4$, assigns $x_i = 1$ if $i \in \mathcal{I}_1$, $x_i = \frac{1}{2}$ if $i \in \mathcal{I}_2$, and $x_i = 0$ if $i \in \mathcal{I}_3$.



Moreover, for every $r \in \{1, 2, \ldots, k\}$, $\mathcal{I}_r$ is equal to the set with indicator vector $\boldsymbol{x}^r - \boldsymbol{x}^{r-1}$, i.e., the binary vector corresponding to the optimal solution of (7) with largest support, as stated in Theorem 1 and Remark 1.

□

For the example of Fig. 1, the optimal sequence of macroitems has three elements $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{I}_3$ and they are represented in Fig. 3. In particular, $P(\mathcal{I}_1) = 4$, $W(\mathcal{I}_1) = 2$, $P(\mathcal{M}_1) = 4$, and $\lambda_1 = \frac{4}{2}$; $P(\mathcal{I}_2) = 6$, $W(\mathcal{I}_2) = 4$, $P(\mathcal{M}_2) = 10$, and $\lambda_2 = \frac{6}{4}$; and $P(\mathcal{I}_3) = 4$, $W(\mathcal{I}_3) = 4$, $P(\mathcal{M}_3) = 14$, and $\lambda_3 = \frac{4}{4}$.

In Fig. 4, we can find the plot of the function u for this example, this time indicating the breakpoints, which are the ratios of the macroitems in the optimal sequence.

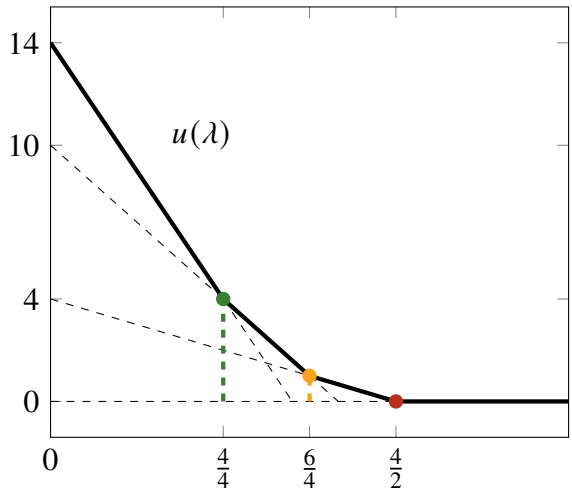
Remark 3. In the case of the classical knapsack problem we have that $\mathcal{G}$ is the graph with empty set of arcs. Therefore, applying Theorem 2 to such an instance we obtain

$$\mathcal{I}_1 = \arg\max_{i \in \mathcal{I}} \frac{p_i}{w_i}, \quad \mathcal{I}_r = \arg \max_{i \in \mathcal{I} \setminus \cup_{j=1}^{r-1} \mathcal{I}_j} \frac{p_i}{w_i}, \quad r \in \{2, 3, \ldots, t\},$$

where t is the number of *distinct* values of the ratios $\frac{p_i}{w_i}$, $i \in \mathcal{I}$. In particular, if all the ratios have distinct values, we have that

$$\mathcal{I}_j = \{i_j\}, \ j \in \{1, 2, \ldots, n\},$$

Figure 4: Representation of the breakpoints of the function u for the example in Fig. 1, which are the profit/weight ratios of the macroitems in the optimal sequence represented in Fig. 3.



where $i_1, i_2 \ldots, i_n$ are such that:

$$\frac{p_{i_1}}{w_{i_1}} > \frac{p_{i_2}}{w_{i_2}} > \cdots > \frac{p_{i_{n-1}}}{w_{i_{n-1}}} > \frac{p_{i_n}}{w_{i_n}},$$

which is the usual ordering leading to the definition of the optimal solution of the linear relaxation of the classical knapsack problem.

Remark 4. The optimal sequence of macroitems also provides an optimal solution of the ratio optimization problem (14) introduced in Section 2.3. Indeed, its optimal value is the ratio of the first macroitem $\mathcal{I}_1$ of the optimal sequence, and the incidence vector of $\mathcal{I}_1$ is an optimal solution of (14): this is exactly the first iteration of the construction in Theorem 1. The corresponding minimization version of (14) is obtained analogously, using the dual variant of the construction presented in Section 5.3.

4 Optimal Solutions of $\mathscr{F}_{\mathrm{LP}}$

In this section, we will show that an optimal solution for $\mathscr{F}_{\mathrm{LP}}$ can be constructed as in the continuous relaxation of the classical knapsack problem, using an optimal sequence of macroitems, instead of just a nonincreasing sequence of simple items. An analogue to the classical knapsack also holds for the dual problem, and for the optimal multiplier in the Lagrangian relaxation of the capacity constraint.

For the rest of the discussion we consider optimal sequences of macroitems $\mathcal{S} = (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})})$ and we assume that the split macroitem $\mathcal{I}_{h(\mathcal{S})}$ has a positive profit. Indeed, otherwise, all macroitems in the sequence with a positive profit, would fit in the capacity c. In that case the optimal solution is easily given by taking all items contained in all macroitems with positive profit. Moreover, we define the *residual capacity* $\tilde{c}(\mathcal{S})$ as

$$\tilde{c}(\mathcal{S}) = c - \sum_{r=1}^{h(\mathcal{S})-1} W_r. \tag{16}$$

The following theorem establishes the optimal solution of $\mathscr{F}_{\mathrm{LP}}$ and of the corresponding dual problem $\mathscr{F}_{\mathrm{D(LP)}}$.

Theorem 3. *Given an optimal sequence of macroitems $\mathcal{S} := (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})})$, let us assume that the split macroitem $\mathcal{I}_{h(\mathcal{S})}$ has a positive profit. Then, an optimal solution of $\mathscr{F}_{\mathrm{LP}}$ is:*

$$x_i = \begin{cases} 1, & i \in \mathcal{I}_r, \quad r \in \{1, 2, \ldots, h(\mathcal{S}) - 1\}, \\ \dfrac{\tilde{c}(\mathcal{S})}{W_{h(\mathcal{S})}}, & i \in \mathcal{I}_{h(\mathcal{S})}, \\ 0, & i \in \mathcal{I}_r, \quad r \in \{h(\mathcal{S}) + 1, h(\mathcal{S}) + 2, \ldots, k(\mathcal{S})\}. \end{cases} \tag{17}$$

An optimal solution of the dual $\mathscr{F}_{\mathrm{D(LP)}}$ of $\mathscr{F}_{\mathrm{LP}}$ is a nonnegative solution given by:

$$\begin{aligned} &\lambda = \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}}, \\ &\mu_i = w_i \left(\frac{P_r}{W_r} - \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}} \right), \quad i \in \mathcal{I}_r, \quad r \in \{1, 2, \ldots, h(\mathcal{S}) - 1\}, \\ &\mu_i = 0, \quad i \in \mathcal{I}_r, \quad r \in \{h(\mathcal{S}), h(\mathcal{S}) + 1, \ldots, k(\mathcal{S})\}, \\ &\alpha_{ij} = 0, \quad i \in \mathcal{I}_r, \quad j \in \mathcal{I}_s, \quad r \neq s, \end{aligned} \tag{18}$$

together with, for each $r \in \{1, 2, \ldots, k(\mathcal{S})\}$, any nonnegative solution of the following linear system

$$\sum_{j \in \mathcal{I}^+(i)} \alpha_{ij} - \sum_{k \in \mathcal{I}^-(i)} \alpha_{ki} = p_i - w_i \frac{P_r}{W_r}, \quad i \in \mathcal{I}_r. \tag{19}$$

Remark 5. Notice that in (18) the values of the μ_i dual variables for every $i \in \mathcal{I}_r$ with $r \in \{1, 2, \ldots, h(\mathcal{S}) - 1\}$ are analogous to the corresponding optimal values of the dual solutions in the classical knapsack problem. Indeed, in the classical knapsack problem, if we assume that all the items i have different $\frac{p_i}{w_i}$ ratio, the optimal sequence of macroitems is just the sequence of

singletons containing one item, ordered by decreasing ratio (see Remark 3), and we have a split item h, depending on the capacity of the knapsack. For the classical problem, the values of the dual variables associated to the relaxed primal constraints $x_i \leq 1$, for the items i with $\frac{p_i}{w_i} > \frac{p_h}{w_h}$ are $\mu_i = p_i - w_i \frac{p_h}{w_h}$. In the general PCKP setting, where we have precedences, the profit and weight of the item i become the profit and weight of the macroitem $\mathcal{I}_r$ containing i, the ratio of the split item becomes the ratio of the split macroitem $\mathcal{I}_{h(\mathcal{S})}$. Then, the value $P_r - W_r \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}}$, which is associated to macroitem $\mathcal{I}_r$ has to be split among the items $i \in \mathcal{I}_r$, and this is done by weighing the quantity proportionally to the weight of the item within the macroitem. Hence we have

$$\mu_i = \left(P_r - W_r \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}}\right) \frac{w_i}{W_r} = w_i \left(\frac{P_r}{W_r} - \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}}\right)$$

which is positive since $\mathcal{I}_r$ precedes $\mathcal{I}_{h(\mathcal{S})}$ in the optimal sequence of macroitems.

Proof. Proof. Vector $\boldsymbol{x}$ as defined in (17) is a feasible solution of $\mathcal{F}_{\text{LP}}$ with primal objective function value given in (22). The dual objective function value at a dual solution fulfilling (18) is equal to:

$$c \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}} + \sum_{r=1}^{h(\mathcal{S})-1} \left(P_r - W_r \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}}\right) = \sum_{r=1}^{h(\mathcal{S})-1} P_r + \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}} \left(c - \sum_{r=1}^{h(\mathcal{S})-1} W_r\right) = \sum_{r=1}^{h(\mathcal{S})-1} P_r + P_{h(\mathcal{S})} \frac{\tilde{c}(\mathcal{S})}{W_{h(\mathcal{S})}},$$

which is equal to the primal objective function value at vector $\boldsymbol{x}$ defined in (17) . Therefore, if we are able to prove that a nonnegative solution of (18) exists and it is feasible for the dual problem $\mathcal{F}_{\text{D(LP)}}$, we can conclude that the primal solution (17) is optimal for the primal problem, and the dual solution obtained through a nonnegative solution of (18) is optimal for the dual problem.

Thus, we need to check that a nonnegative solution of (18) fulfills the constraints of (2). For every $i \in \mathcal{I}_r$ with $r \in \{1, 2, \ldots, h(\mathcal{S}) - 1\}$, we note that

$$w_i \lambda + \mu_i + \sum_{j \in \mathcal{I}^+(i)} \alpha_{ij} - \sum_{k \in \mathcal{I}^-(i)} \alpha_{ki} = w_i \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}} + w_i \left(\frac{P_r}{W_r} - \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}}\right) + p_i - \frac{P_r}{W_r} w_i = p_i.$$

For every $i \in \mathcal{I}_r$ with $r \in \{h(\mathcal{S}), h(\mathcal{S}) + 1, \ldots, k(\mathcal{S})\}$, we note that

$$w_i \lambda + \mu_i + \sum_{j \in \mathcal{I}^+(i)} \alpha_{ij} - \sum_{k \in \mathcal{I}^-(i)} \alpha_{ki} = w_i \frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}} + 0 + p_i - w_i \frac{P_r}{W_r} \geq p_i,$$

where the last inequality holds since $\frac{P_{h(\mathcal{S})}}{W_{h(\mathcal{S})}} \geq \frac{P_r}{W_r}$, because $r \geq h(\mathcal{S})$ and the optimal sequence of macroitems $(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})})$ is decreasing.

Then, it remains to prove that for every $r \in \{1, 2, \ldots, k(\mathcal{S})\}$, there exists a nonnegative vector α such that $\alpha_{ij} = 0$ if i and j belong to different macroitems in the optimal sequence, and

$$\sum_{j \in \mathcal{I}^+(i) \cap \mathcal{I}_r} \alpha_{ij} - \sum_{k \in \mathcal{I}^-(i) \cap \mathcal{I}_r} \alpha_{ki} = p_i - w_i \frac{P_r}{W_r}, \qquad i \in \mathcal{I}_r. \tag{20}$$

To see this, we introduce the subgraph $\mathcal{G}(\mathcal{I}_r) = (\mathcal{I}_r, \mathcal{A}(\mathcal{I}_r))$ induced by $\mathcal{I}_r$, where

$$\mathcal{A}(\mathcal{I}_r) = \{(i, j) \in \mathcal{A} \ : \ i, j \in \mathcal{I}_r\},$$

and for each $i \in \mathcal{I}_r$ we set

$$b_i = p_i - w_i \frac{P_r}{W_r}.$$

Then, a nonnegative vector α fulfilling (20) is a feasible flow for the min-cost flow problem over graph $\mathcal{G}(\mathcal{I}_r)$ where all arcs have infinite capacities, and the items $i \in \mathcal{I}_r$ are subdivided into supply items with supply $b_i > 0$, demand items with demand $-b_i > 0$, and transit items with $b_i = 0$. Existence of such a feasible flow is proved as follows. We have that:

- the total demand equals the total supply, i.e.,

$$\sum_{i \in \mathcal{I}_r} b_i = \sum_{i \in \mathcal{I}_r} \left(p_i - w_i \frac{P_r}{W_r} \right) = P_r - W_r \frac{P_r}{W_r} = 0.$$

- For every proper subset $\mathcal{M}$ of $\mathcal{I}_r$ such that

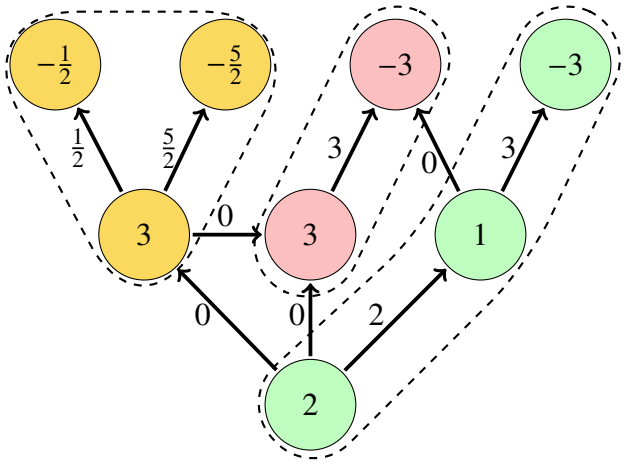
$$\sum_{i \in \mathcal{M}} b_i > 0, \tag{21}$$

  we have $\mathcal{A}_r^+(\mathcal{M}) := \{(i, j) \in \mathcal{A} \colon i, j \in \mathcal{I}_r,\ i \in \mathcal{M},\ j \notin \mathcal{M}\} \neq \emptyset$. Indeed, the left-hand side of (21) is equal to

$$\sum_{i \in \mathcal{M}} \left(p_i - w_i \frac{P_r}{W_r} \right) = P(\mathcal{M}) - W(\mathcal{M}) \frac{P_r}{W_r},$$

  where $P(\mathcal{M}) := \sum_{i \in \mathcal{M}} p_i$ and $W(\mathcal{M}) := \sum_{i \in \mathcal{M}} w_i$. Hence, inequality (21) is equivalent to $\frac{P(\mathcal{M})}{W(\mathcal{M})} > \frac{P_r}{W_r}$. Now, if we assume by contradiction that there is no arc exiting from $\mathcal{M}$ within $\mathcal{I}_r$ (i.e., $\mathcal{A}_r^+(\mathcal{M}) = \emptyset$), then $\mathcal{M}$ alone can be put in place of $\mathcal{I}_r$ in any feasible sequence of macroitems containing $\mathcal{I}_r$, while maintaining the feasibility of the sequence and advancing its position in the lexicographic order defined in Definition 5, thus contradicting the fact that $\mathcal{S} = (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})})$ is an optimal sequence of macroitems.

Figure 5: Representation of the flow giving the optimal solution of $\mathscr{F}_{\mathrm{D(LP)}}$, with capacity $c = 4$, for the example of Fig. 1. The outflow-inflow balance of an item, prescribed by (18), is written inside the relative item, while the flow components are written along the corresponding arc.



These two facts allow us to apply classic theorems on the existence of feasible flows on networks, to show that a flow within $\mathcal{G}(\mathcal{I}_r)$ satisfying (20) exists (see, for example, [1, Theorem 6.12]).

□

It follows from Theorem 3 that the optimal value of $\mathscr{F}_{\mathrm{LP}}$ and $\mathscr{F}_{\mathrm{D(LP)}}$ is

$$\sum_{r=1}^{h(\mathcal{S})-1} P_r + P_{h(\mathcal{S})} \frac{\tilde{c}(\mathcal{S})}{W_{h(\mathcal{S})}}. \tag{22}$$

As an illustration of the previous result, we refer to Fig. 3 for the optimal solution of the $\mathscr{F}_{\mathrm{LP}}$ arising from the graph in Fig. 1, while in Fig. 5 we represent the flow inside the macroitems, which gives the optimal solution of the dual problem $\mathscr{F}_{\mathrm{D(LP)}}$, for the same example.

The final result concerns the value of the optimal multiplier for the Lagrangian relaxation of the capacity constraint, i.e., the optimal solution of the dual Lagrangian problem. This is the solution of

$$\min_{\lambda \geq 0} \left(c\,\lambda + \max_{x \in \mathcal{P}} \sum_{i \in \mathcal{I}} (p_i - \lambda\, w_i) x_i \right) = \min_{\lambda \geq 0} c\,\lambda + u(\lambda). \tag{23}$$

We denote the optimization problem (23) by $\mathscr{F}_{\mathrm{LAG}}$ in the remainder of this manuscript. Moreover, we denote by $\zeta(\mathscr{F}_{\mathrm{LAG}})$ its optimal value.

We are able to prove the following result, analogous to the one for the classical knapsack problem:

Theorem 4. *Let $\mathcal{S} = (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k)$ be the optimal sequence of macroitems, and assume that the split macroitem $\mathcal{I}_h$ for this sequence has positive profit. Then, the optimal solution of $\mathscr{F}_{\text{LAG}}$ is $\lambda_h := \frac{P_h}{W_h}$ and its optimal value is equal to the optimal value of $\mathscr{F}_{\text{LP}}$, i.e., $\zeta(\mathscr{F}_{\text{LAG}}) = \zeta(\mathscr{F}_{\text{LP}})$.*

Proof. Proof. Given the optimal sequence of macroitems $(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k)$, the function $u(\lambda) := \max \sum_{i \in \mathcal{I}} (p_i - \lambda\, w_i) x_i$ can be written, for each $r \in \{0, 1, \ldots, k\}$ and each $\lambda \in [\lambda_{r+1}, \lambda_r)$, in the form

$$u(\lambda) = P(\mathcal{M}_r) - \lambda\, W(\mathcal{M}_r),$$

where $\mathcal{M}_0 := \emptyset, \lambda_0 := +\infty$ and $\lambda_{k+1} := 0$. The subgradient of the objective function $g(\lambda) = c\,\lambda + u(\lambda)$ of (23) evaluated at breakpoint λ_r is the interval:

$$\frac{\partial\, g(\lambda_r)}{\partial\, \lambda} = [c - W(\mathcal{M}_r), c - W(\mathcal{M}_{r-1})]\,.$$

Then, by definition of the split macroitem, it holds that

$$0 \in [c - W(\mathcal{M}_h), c - W(\mathcal{M}_{h-1})] = \frac{\partial\, g(\lambda_h)}{\partial\, \lambda},$$

i.e., λ_h fulfills the optimality condition of the convex nonsmooth optimization problem (23).

The optimal value of the dual Lagrangian problem is:

$$\zeta(\mathscr{F}_{\text{LAG}}) = g(\lambda_h) = c\,\lambda_h + u(\lambda_h) = c\,\frac{P_h}{W_h} + P(\mathcal{M}_h) - \frac{P_h}{W_h}\,W(\mathcal{M}_h) = \sum_{r=1}^{h-1} P_r + P_h\,\frac{c - \sum_{r=1}^{h-1} W_r}{W_h},$$

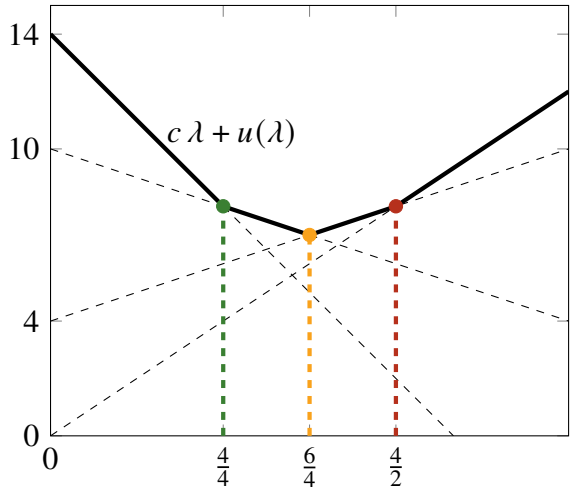
which is equal to the optimal value of $\mathscr{F}_{\text{LP}}$, i.e., $\zeta(\mathscr{F}_{\text{LAG}}) = \zeta(\mathscr{F}_{\text{LP}})$. □

The plot of the objective function of $\mathscr{F}_{\text{LAG}}$ for the example of Fig. 1 with capacity $c = 4$ is represented in Fig. 6. Note that since in this case $h = 2$, the minimum of the function $c\,\lambda + u(\lambda)$ is attained at $\lambda_2 = \frac{6}{4}$, which is the ratio of the split macroitem $\mathcal{I}_2$.

5 Algorithms for computing the optimal sequence of macroitems on directed forests

In this section, we present new algorithms for computing the optimal sequence of macroitems when the precedence graph $\mathcal{G}$ is a *directed forest*, i.e., a disjoint union of directed trees. We show that, despite the additional combinatorial structure imposed by the precedence constraints, the optimal sequence can be computed with an $O(n^2)$ algorithm, where n is the number of items. We

Figure 6: Plot of the function $c\,\lambda + u(\lambda)$ for the example of Fig. 1 with capacity $c = 4$. The minimum is achieved at the breakpoint giving the profit/weight ratio of the split macroitem in the optimal sequence.



further show that when the arcs are all directed in the same direction (i.e., the forest is composed of *out-trees* or *in-trees*), a close variant of this algorithm reduces the complexity to $O(n \log n)$, matching the time needed to compute the parametric LP relaxation of the classical KP for all capacity values, which requires sorting items by efficiency and thus runs in $O(n \log n)$ [18, 20]. As already discussed in the Introduction, the optimal sequence of macroitems can also be obtained on arbitrary precedence graphs via parametric flow algorithms in $O(mn \log n)$ time, where m is the number of arcs [13]; on forests, where $m = O(n)$, this general-purpose bound becomes $O(n^2 \log n)$, and the algorithms introduced in this section improve on it by exploiting the tree structure directly. A computational comparison between these two approaches is provided in Section 6.

5.1 Notation

In order to describe the algorithms of this section, we introduce the following notation.

Definition 6. Let $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ be a directed forest and let $j \in \mathcal{I}$ be an item. The *preceding set* of item j is the set of items which can be reached from j with a directed path, which is the set

$$F_j = \{k \in \mathcal{I} : \text{there exists a directed path from } j \text{ to } k\}.$$

Let $a = (i, j) \in \mathcal{A}$ be an edge. The *minimal preceding set* of arc $a = (i, j)$ is the set of items

which are in the preceding set of i but not in the preceding set of j, namely the set

$$F_a := F_i \setminus F_j.$$

Definition 7. Let $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ be a directed forest and let $j \in \mathcal{I}$ be an item. The *succeeding set* of item j is the set of items which can reach j with a directed path, which is the set

$$B_j = \{k \in \mathcal{I} : \text{there exists a directed path from } k \text{ to } j\}.$$

Let $a = (i, j) \in \mathcal{A}$ be an edge. The *minimal succeeding set* of arc $a = (i, j)$ is the set of items which are in the succeeding set of j but not in the succeeding set of i, namely the set

$$B_a := B_j \setminus B_i.$$

Definition 8. We indicate with $\mathcal{I}_{\mathrm{f}}$ the set of the *final items* of $\mathcal{G}$, and with $\mathcal{I}_{\mathrm{i}}$ the set of its *initial items*, which are defined respectively by

$$\mathcal{I}_{\mathrm{f}} := \{j \in \mathcal{I} : \mathcal{I}^+(j) = \emptyset\}, \qquad \mathcal{I}_{\mathrm{i}} := \{j \in \mathcal{I} : \mathcal{I}^-(j) = \emptyset\}.$$

Definition 9. Let $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ be a directed forest and let $\mathcal{M}' \subset \mathcal{M} \subset \mathcal{I}$ be a chain of subsets of items. The set of arcs connecting items in $\mathcal{M}$ is the set

$$\mathcal{A}(\mathcal{M}) := \{(i, j) \in \mathcal{A} : i, j \in \mathcal{M}\}.$$

The set of items in $\mathcal{M}$ which are connected with $\mathcal{M}'$ by an arc exiting $\mathcal{M}'$ is the set

$$\mathcal{M}^+(\mathcal{M}') := \{j \in \mathcal{M} \setminus \mathcal{M}' : \exists i \in \mathcal{M}',\ (i, j) \in \mathcal{A}\},$$

and the set of items in $\mathcal{M}$ which are connected with $\mathcal{M}'$ by an arc entering $\mathcal{M}'$ is the set

$$\mathcal{M}^-(\mathcal{M}') := \{i \in \mathcal{M} \setminus \mathcal{M}' : \exists j \in \mathcal{M}',\ (i, j) \in \mathcal{A}\}.$$

An illustration of preceding/succeeding sets associated to items and arcs, and of final/initial items, can be found in Fig. 7, while an illustration of the sets $\mathcal{M}^+(\mathcal{M}')$ and $\mathcal{M}^-(\mathcal{M}')$ is in Fig. 8.

Notice that a sequence $\mathcal{S} = \left(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})}\right)$ of macroitems is feasible if and only if for every item $j \in \mathcal{I}$, all items in F_j are contained in macroitems of the sequence $\mathcal{S}$ that precede the macroitem containing j, or are contained in the same macroitem. Furthermore, given an arc $a = (i, j)$, item i can be in the same macroitem as item j only if all items in F_a are contained in

Figure 7: Representation of a directed tree. The arc $a = (i, j)$ is highlighted in bold. Two dashed lines enclose the sets F_i and B_j. The items in F_a, and their connecting arcs, are highlighted in orange, while the same is done in green for B_a. Final items (i.e., items v with $\mathcal{I}^+(v) = \emptyset$) are filled with a north-east line pattern, while initial items (i.e., items v with $\mathcal{I}^-(v) = \emptyset$) are filled with a dot pattern.

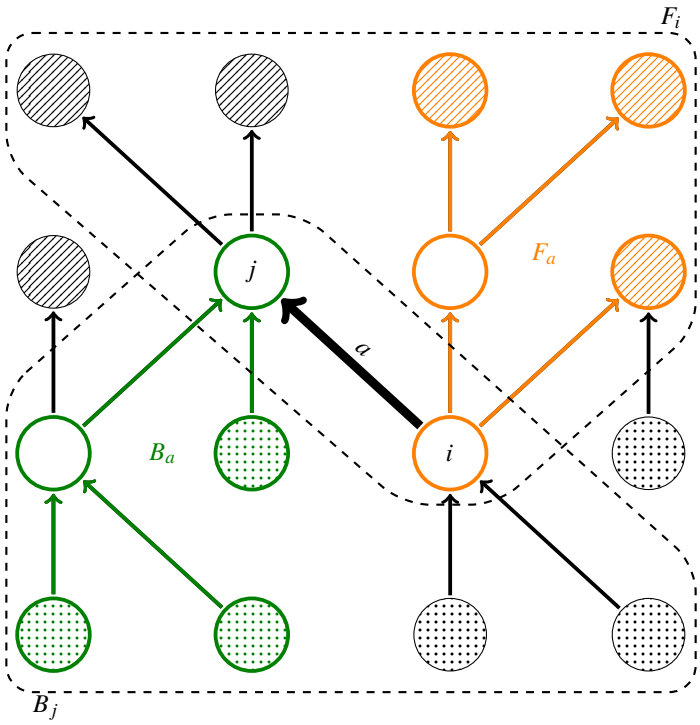
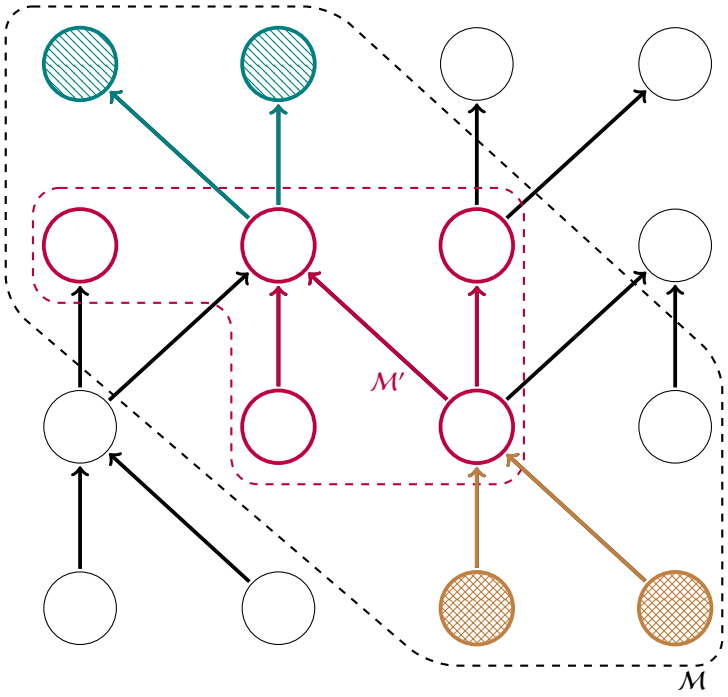


Figure 8: Directed graph with dashed set $\mathcal{M}$ and highlighted subset $\mathcal{M}'$ in purple. Nodes in $\mathcal{M}^+(\mathcal{M}')$ are highlighted in teal with a north-west line pattern, and the arcs from $\mathcal{M}'$ to $\mathcal{M}^+(\mathcal{M}')$ are highlighted in teal. Nodes in $\mathcal{M}^-(\mathcal{M}')$ are highlighted in brown with a crosshatch pattern, and the arcs from $\mathcal{M}^-(\mathcal{M}')$ to $\mathcal{M}'$ are highlighted in brown.



a macroitem of the sequence $\mathcal{S}$ that precedes the macroitem containing j, or are contained in the same macroitem.

Remark 6. Notice that $\mathcal{I}^+(F_j) = \{k \in \mathcal{I} \setminus F_j : \exists i \in F_j,\ (i,k) \in \mathcal{A}\} = \emptyset$ for every $j \in \mathcal{I}$ by Definition 6. Furthermore $\mathcal{I}^+(F_{(i,j)} \setminus \{i\}) = \emptyset$ for every $(i,j) \in \mathcal{A}$ since $F_{(i,j)} \setminus \{i\}$ is the disjoint union of the sets F_k with $k \in \mathcal{I}^+(i) \setminus \{j\}$

5.2 Main algorithm to compute the optimal sequence of macroitems on a forest

To show that our main algorithm outputs the optimal sequence of macroitems we prove three propositions.

Proposition 4. *Suppose $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ is a directed forest and let $\mathcal{M} \subset \mathcal{I}$ be a subset of items. If $\mathcal{I}^+(\mathcal{M}) = \emptyset$, then*

$$\mathcal{M} = \left(\bigsqcup_{a \in \tilde{\mathcal{A}}} F_a \right) \sqcup \tilde{\mathcal{I}}_{\mathrm{f}}$$

for some subset $\tilde{\mathcal{A}} \subset \mathcal{A}$ and $\tilde{\mathcal{I}}_{\mathrm{f}} \subset \mathcal{I}_{\mathrm{f}}$, i.e., $\mathcal{M}$ is the disjoint union of the minimal preceding sets associated to some arcs and some subset of final items.

Proof. Proof. We can show this constructively. We start with $\tilde{\mathcal{A}} = \emptyset$ and $\tilde{\mathcal{I}}_{\mathrm{f}} = \emptyset$. Then, we add one final item in $\mathcal{I}_{\mathrm{f}} \cap \mathcal{M}$ to $\tilde{\mathcal{I}}_{\mathrm{f}}$ for every connected component of the graph $(\mathcal{M}, \mathcal{A}(\mathcal{M}))$. Such final items exist since $\mathcal{I}^+(\mathcal{M}) = \emptyset$ and $\mathcal{G}$ is a forest. Thus we obtain a first set $\mathcal{M}' \subset \mathcal{M}$ defined by

$$\mathcal{M}' = \tilde{\mathcal{I}}_{\mathrm{f}}$$

and we have $\mathcal{I}^+(\mathcal{M}') = \emptyset$.

Now if $\mathcal{M}^-(\mathcal{M}') \neq \emptyset$ (i.e., $\mathcal{M} \neq \mathcal{M}'$), we consider any item $j \in \mathcal{M}'$ which is connected to some item in $\mathcal{M}^-(\mathcal{M}')$. Then we take any maximal backwards path $\pi_j \subset \mathcal{A}(\mathcal{M}) \setminus \mathcal{A}(\mathcal{M}')$ from j to some item $v(j)$ with $\mathcal{M}^-(\{v(j)\}) = \emptyset$, and we add the arcs in π_j to $\tilde{\mathcal{A}}$, updating $\mathcal{M}'$ accordingly. Since the new elements added to $\tilde{\mathcal{A}}$ are chosen in a backwards path in $\mathcal{A}(\mathcal{M}) \setminus \mathcal{A}(\mathcal{M}')$, we have $F_a \cap F_{a'} = \emptyset$, for every $a, a' \in \tilde{\mathcal{A}}$ with $a \neq a'$. Thus we obtain

$$\mathcal{M}' = \left(\bigsqcup_{a \in \tilde{\mathcal{A}}} F_a \right) \sqcup \tilde{\mathcal{I}}_{\mathrm{f}}$$

and we have enlarged the set $\mathcal{M}'$ to also include $\bigcup F_{v(j)}$, where the union is extended to all items $j \in \mathcal{M}'$ which are connected to some item in $\mathcal{M}^-(\mathcal{M}')$. Hence we have again $\mathcal{I}^+(\mathcal{M}') = \emptyset$.

Then, we repeat the last step until $\mathcal{M}' = \mathcal{M}$. □

An illustration of the outcome of the constructive procedure described in the proof of Proposition 4 can be found in Fig. 9.

Proposition 5. *Suppose $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ is a directed forest and let $a^* \in \arg\max_{a \in \mathcal{A}} \frac{P(F_a)}{W(F_a)}$ with $a^* = (u, v)$. If $\frac{P(F_{a^*})}{W(F_{a^*})} \geq \frac{p_f}{w_f}$ for every $f \in \mathcal{I}_{\mathrm{f}}$, then the two items u and v belong to the same macroitem in the optimal sequence.*

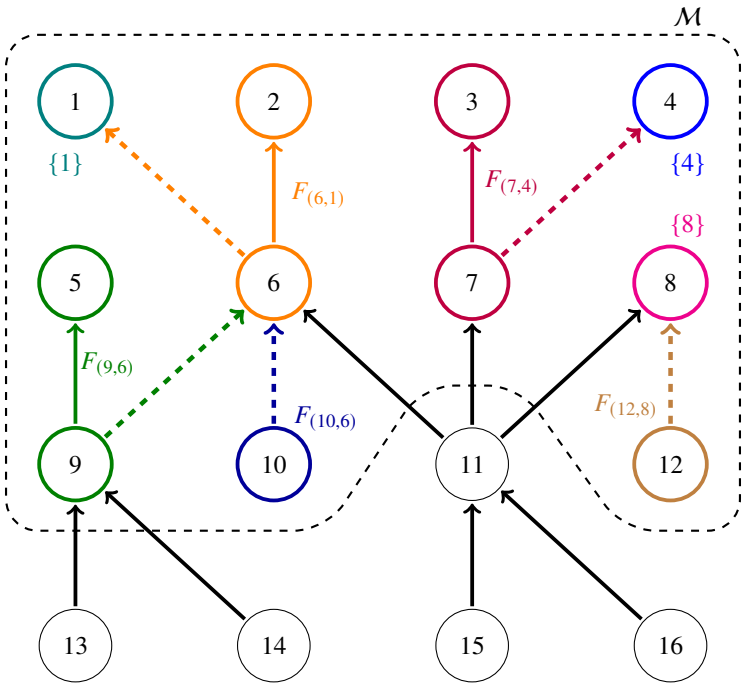
Proof. Proof. Let $\mathcal{S} = \left(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})}\right)$ be the optimal sequence of macroitems and let $t(v) \in \{1, 2, \ldots, k(\mathcal{S})\}$ be the index of the macroitem containing v, i.e., $v \in \mathcal{I}_{t(v)}$. Consider the set of items

$$\mathcal{M} := \bigcup_{r=1}^{t(v)} \mathcal{I}_r.$$

Notice that $\mathcal{I}^+(\mathcal{M}) = \emptyset$ since $\mathcal{S}$ is a feasible sequence of macroitems. By Proposition 4 we have

$$\mathcal{M} = \left(\bigsqcup_{a \in \tilde{\mathcal{A}}} F_a \right) \sqcup \tilde{\mathcal{I}}_{\mathrm{f}} \tag{24}$$

Figure 9: Constructive decomposition of a set $\mathcal{M}$ with $\mathcal{I}^+(\mathcal{M}) = \emptyset$ according to the proof of Proposition 4. We start from one final item per connected component in $\mathcal{M}$, i.e., $\mathcal{M}' = \tilde{\mathcal{I}}_{\mathrm{f}} = \{1, 4, 8\}$, and then iteratively add sets F_a associated with arcs a from $\mathcal{M}^-(\mathcal{M}')$ to $\mathcal{M}'$ (shown as dashed colored arcs), until $\mathcal{M}' = \mathcal{M}$. The resulting disjoint union is $\mathcal{M} = \{1, 4, 8\} \sqcup F_{(6,1)} \sqcup F_{(7,4)} \sqcup F_{(12,8)} \sqcup F_{(9,6)} \sqcup F_{(10,6)}$.



for some $\tilde{\mathcal{A}} \subset \mathcal{A}$ and $\tilde{\mathcal{I}}_{\mathrm{f}} \subset \mathcal{I}_{\mathrm{f}}$.

Suppose by contradiction that $u \notin \mathcal{I}_{t(v)}$ and thus $u \notin \mathcal{M}$ since $(u, v) \in \mathcal{A}$ and $\mathcal{S}$ is a feasible sequence of macroitems. Hence by Remark 6 we have $\mathcal{I}^+(F_{a^*} \cap \mathcal{M}) = \emptyset$. Thus by Proposition 4 we have

$$F_{a^*} \cap \mathcal{M} = \left(\bigsqcup_{a \in \mathcal{A}^*} F_a \right) \sqcup \mathcal{I}_{\mathrm{f}}^* \tag{25}$$

for some $\mathcal{A}^* \subset \mathcal{A}$ and $\mathcal{I}_{\mathrm{f}}^* \subset \mathcal{I}_{\mathrm{f}}$.

We have

$$\frac{P(F_{a^*} \setminus \mathcal{M})}{W(F_{a^*} \setminus \mathcal{M})} \geq \frac{P(F_{a^*})}{W(F_{a^*})} \geq \frac{P(\mathcal{M})}{W(\mathcal{M})} \geq \frac{P(\mathcal{I}_{t(v)})}{W(\mathcal{I}_{t(v)})}$$

where in the last inequality we used the fact that $\mathcal{S} = \left(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_{k(\mathcal{S})}\right)$ is a decreasing sequence of macroitems. The second inequality follows from the fact that $\frac{P(F_{a^*})}{W(F_{a^*})} \geq \frac{P(F_a)}{W(F_a)}$ for every $a \in \mathcal{A}$ and $\frac{P(F_{a^*})}{W(F_{a^*})} \geq \frac{p_f}{w_f}$ for every $f \in \mathcal{I}_{\mathrm{f}}$, Eq. (24) and Proposition 3. The first inequality follows again from the same properties of F_{a^*}, Eq. (25) and Proposition 3. This chain of inequalities contradicts the hypothesis that $\mathcal{S}$ is the optimal sequence of macroitems, because the sequence obtained from

$$\left(\mathcal{I}_1, \ldots, \mathcal{I}_{t(v)-1},\ \mathcal{I}_{t(v)} \cup (F_{a^*} \setminus \mathcal{M}),\ \mathcal{I}_{t(v)+1} \setminus (F_{a^*} \setminus \mathcal{M}), \ldots,\ \mathcal{I}_{k(\mathcal{S})} \setminus (F_{a^*} \setminus \mathcal{M})\right)$$

after possibly removing some empty sets at the end, is feasible and higher than $\mathcal{S}$ in the lexicographic order associated to $\succ$. Indeed, $\frac{P(F_{a^*} \setminus \mathcal{M})}{W(F_{a^*} \setminus \mathcal{M})} > \frac{P(\mathcal{I}_{t(v)})}{W(\mathcal{I}_{t(v)})}$ and since $F_{a^*} \setminus \mathcal{M}$ and $\mathcal{I}_{t(v)}$ are disjoint, the ratio of their union is not lower than the ratio of $\mathcal{I}_{t(v)}$, by Proposition 3. □

Proposition 5 allows us, by collapsing the arc a^*, to reduce the problem of finding the optimal sequence of macroitems to a smaller graph. An illustration of such an operation on an instance can be found in Fig. 10, where a full run of the main algorithm is presented.

Proposition 6. *Suppose $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ is a directed forest and let $g \in \mathcal{B}_{\mathrm{f}} := \arg\max_{f \in \mathcal{I}_{\mathrm{f}}} \frac{p_f}{w_f}$. If $\frac{p_g}{w_g} > \frac{P(F_a)}{W(F_a)}$ for every $a \in \mathcal{A}$, then $\mathcal{B}_{\mathrm{f}}$ is the first macroitem in the optimal sequence of macroitems.*

Proof. Proof. Since the optimal sequence of macroitems must be feasible, any candidate subset $\mathcal{M} \subset \mathcal{I}$ to be the first macroitem in the optimal sequence must satisfy $\mathcal{I}^+(\mathcal{M}) = \emptyset$. By Proposition 4 we have

$$\mathcal{M} = \left(\bigsqcup_{a \in \tilde{\mathcal{A}}} F_a \right) \sqcup \tilde{\mathcal{I}}_{\mathrm{f}}$$

for some subsets $\tilde{\mathcal{A}} \subset \mathcal{A}$, $\tilde{\mathcal{I}}_{\mathrm{f}} \subset \mathcal{I}_{\mathrm{f}}$. Since $g \in \mathcal{B}_{\mathrm{f}} = \arg\max_{f \in \mathcal{I}_{\mathrm{f}}} \frac{p_f}{w_f}$ and $\frac{p_g}{w_g} > \frac{P(F_a)}{W(F_a)}$ for every $a \in \mathcal{A}$, by Proposition 3 the ratio associated to the set $\mathcal{B}_{\mathrm{f}}$ is larger than or equal to the ratio associated to $\mathcal{M}$, with the equality possible only if $\tilde{\mathcal{A}} = \emptyset$. This implies that $\mathcal{B}_{\mathrm{f}}$ is the subset of $\mathcal{I}$ of largest ratio with the largest support, i.e., the larger subset in the order $\succ$. □

Algorithm 1 Algorithm to compute the optimal sequence of macroitems on a forest

1: **input** $\mathcal{I}$, $\mathcal{A}$, $\mathcal{I}_\mathrm{f}$, $\boldsymbol{p}$, $\boldsymbol{w}$
2: ▷ Initialization:
3: $\mathcal{I}' = \mathcal{I}$, $\mathcal{A}' = \mathcal{A}$, $\mathcal{I}'_\mathrm{f} = \mathcal{I}_\mathrm{f}$
4: **for** $j \in \mathcal{I}'$ **do**
5: Set $p'_j = p_j$, $w'_j = w_j$ profit and weight for item $j \in \mathcal{I}'$; $\mathcal{M}_j = \{j\}$
6: **end for**
7: $k = 0$
8: **while** $\mathcal{I}' \neq \emptyset$ **do**
9: ▷ Determine $a^* = (u, v)$ from Proposition 5 and its ratio $\frac{P(F_{a^*})}{W(F_{a^*})}$:
10: $(u, v) \leftarrow$ FindBestWing$(\mathcal{I}', \mathcal{A}')$
11: ▷ Determine the set $\mathcal{B}_\mathrm{f}$ from Proposition 6 and the ratio $\frac{p'_g}{w'_g}$:
12: $(g, \mathcal{B}_\mathrm{f}) \leftarrow$ FindBestFinalNodes$(\mathcal{I}'_\mathrm{f})$
13: ▷ If largest ratio is achieved by some final items (items in $\mathcal{B}_\mathrm{f}$), we get a new macroitem:
14: **if** $\frac{p'_g}{w'_g} > \frac{P(F_{(u,v)})}{W(F_{(u,v)})}$ **then**
15: RemoveFinalNodes$(\mathcal{B}_\mathrm{f})$
16: **else**
17: ▷ If largest ratio is achieved by (or also by) F_{a^*} we "contract" the arc a^*:
18: ContractArc(u, v)
19: **end if**
20: **end while**
21: **return** $\mathcal{S} = (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k)$

In Algorithm 1, after the initialization (lines 3-7), which creates a copy $(\mathcal{I}', \mathcal{A}')$ of the graph and associates to each copied item j a set of items $\mathcal{M}_j \subset \mathcal{I}$ in the original graph, then in the main cycle (lines 8-20) at each iteration we either find a new macroitem in the optimal sequence as prescribed by Proposition 6, and we remove it from the graph $(\mathcal{I}', \mathcal{A}')$, or we "contract" an arc joining two adjacent items according to Proposition 5.

In particular, in procedure FindBestWing (Algorithm 2) we determine the arc $a^* = (u, v) \in \mathcal{A}'$ whose associated set F_{a^*} is the one with the largest ratio. Then, in procedure FindBestFinalNodes (Algorithm 3) we determine the set $\mathcal{B}_\mathrm{f}$ of the final items of $\mathcal{I}'$ with the largest ratio. Now, if the largest determined ratio is achieved only by the final items in $\mathcal{B}_\mathrm{f}$, then the subset of items $\bigcup_{f \in \mathcal{B}_\mathrm{f}} \mathcal{M}_f \subset \mathcal{I}$ is the next macroitem in the optimal sequence by Proposition 6. Hence, procedure RemoveFinalNodes (Algorithm 4) adds to the optimal sequence the new macroitem, and removes the items in $\mathcal{B}_\mathrm{f}$ together with the corresponding arcs from the graph $(\mathcal{I}', \mathcal{A}')$. Conversely, if the largest determined ratio is achieved by (or also by) a set $F_{a^*} \subset \mathcal{I}'$ associated to some arc $a^* \in \mathcal{A}'$, procedure ContractArc (Algorithm 5) removes arc $a^* = (u, v)$ from $\mathcal{A}'$ according to Proposition 5, joining the two items u and v and the corresponding sets $\mathcal{M}_u$, $\mathcal{M}_v$ of associated items in $\mathcal{I}$. An illustration of a full run of Algorithm 1 on a small instance can be found in Fig. 10.

Theorem 5. *Algorithm 1 computes the optimal sequence of macroitems on a directed forest in*

Algorithm 2 Sub-procedure FindBestWing: determine arc $a^* = (u, v)$ with largest ratio $\frac{P(F_{a^*})}{W(F_{a^*})}$

procedure FindBestWing($\mathcal{I}', \mathcal{A}'$)
  Compute $P(F_j)$, $W(F_j)$ for every $j \in \mathcal{I}'$
  $u = 0$, $v = 0$, $P(F_{(0,0)}) = -\infty$, $W(F_{(0,0)}) = 1$
  for $a = (i, j) \in \mathcal{A}'$ **do**
    $P(F_a) = P(F_i) - P(F_j)$, $W(F_a) = W(F_i) - W(F_j)$
    if $\frac{P(F_a)}{W(F_a)} > \frac{P(F_{(u,v)})}{W(F_{(u,v)})}$ **then** $u = i$, $v = j$
    end if
  end for
  return (u, v)
end procedure

Algorithm 3 Sub-procedure FindBestFinalNodes: determine set $\mathcal{B}_\mathrm{f}$ of final items with largest ratio $\frac{p'_g}{w'_g}$

procedure FindBestFinalNodes($\mathcal{I}'_\mathrm{f}$)
  $g = 0$, $p'_0 = -\infty$, $w'_0 = 1$, $\mathcal{B}_\mathrm{f} = \emptyset$
  for $f \in \mathcal{I}'_\mathrm{f}$ **do**
    if $\frac{p'_f}{w'_f} > \frac{p'_g}{w'_g}$ **then** $g = f$, $\mathcal{B}_\mathrm{f} = \{f\}$
    else if $\frac{p'_f}{w'_f} = \frac{p'_g}{w'_g}$ **then** $\mathcal{B}_\mathrm{f} = \mathcal{B}_\mathrm{f} \cup \{f\}$
    end if
  end for
  return $(g, \mathcal{B}_\mathrm{f})$
end procedure

$O(n^2)$ time.

Proof. Proof. At each iteration, either Proposition 6 identifies the next macroitem and removes it, or Proposition 5 identifies an arc whose endpoints must belong to the same macroitem and contracts it. These operations preserve the optimal sequence of the original instance, after replacing each contracted item by the associated set $\mathcal{M}_j$. Since each iteration decreases the cardinality of $\mathcal{I}'$ by at least one, there are at most n iterations. In a forest, completing the four procedures in Algorithms 2 to 5 can be done in $O(n)$ time. This is rather obvious for the last three procedures, each involving one or two `for` cycles with at most n iterations, and with a constant number of operations per iteration. Concerning the procedure in Algorithm 2, it also takes $O(n)$ by a recursive computation of the values $P(F_j)$ and $W(F_j)$ starting from the final items of the graph, up to the initial ones. Hence, the (at most) n iterations, with a computational cost per iteration equal to $O(n)$, require a number of operations which is $O(n^2)$.

□

Remark 7. We remark that the proposed approach does not suffer from numerical precision prob-

Algorithm 4 Sub-procedure RemoveFinalNodes: create next macroitem from $\mathcal{B}_\mathrm{f}$ and update the graph

procedure RemoveFinalNodes($\mathcal{B}_\mathrm{f}$)
  $k = k + 1$, $\mathcal{I}_k = \bigcup_{f \in \mathcal{B}_\mathrm{f}} \mathcal{M}_f$, $C_\mathrm{f} = \emptyset$
  for $i \in \mathcal{I}'^{-}(\mathcal{B}_\mathrm{f})$ **do** $C_\mathrm{f} = C_\mathrm{f} \cup \{i\}$
    for $(i, f) \in \mathcal{A}'$ with $f \in \mathcal{B}_\mathrm{f}$ **do**
      $\mathcal{A}' = \mathcal{A}' \setminus \{(i, f)\}$
    end for
  end for
  for $i \in C_\mathrm{f}$ **do**
    if $\mathcal{I}'^{+}(i) = \emptyset$ **then** $\mathcal{I}'_\mathrm{f} = \mathcal{I}'_\mathrm{f} \cup \{i\}$
    end if
  end for
  $\mathcal{I}'_\mathrm{f} = \mathcal{I}'_\mathrm{f} \setminus \mathcal{B}_\mathrm{f}$, $\mathcal{I}' = \mathcal{I}' \setminus \mathcal{B}_\mathrm{f}$
end procedure

Algorithm 5 Sub-procedure ContractArc: contract arc $a^* = (u, v)$ and merge the associated item sets

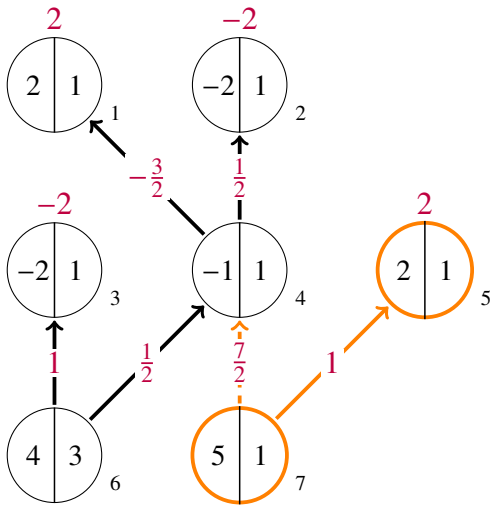
procedure ContractArc(u, v)
  for $i \in \mathcal{I}'^{-}(u)$ **do** $\mathcal{A}' = \mathcal{A}' \setminus \{(i, u)\}$, $\mathcal{A}' = \mathcal{A}' \cup \{(i, v)\}$
  end for
  for $j \in \mathcal{I}'^{+}(u) \setminus \{v\}$ **do** $\mathcal{A}' = \mathcal{A}' \setminus \{(u, j)\}$, $\mathcal{A}' = \mathcal{A}' \cup \{(v, j)\}$
    if $v \in \mathcal{I}'_\mathrm{f}$ **then**
      $\mathcal{I}'_\mathrm{f} = \mathcal{I}'_\mathrm{f} \setminus \{v\}$
    end if
  end for
  $\mathcal{M}_v = \mathcal{M}_u \cup \mathcal{M}_v$, $p'_v = p'_u + p'_v$, $w'_v = w'_u + w'_v$
  $\mathcal{A}' = \mathcal{A}' \setminus \{(u, v)\}$, $\mathcal{I}' = \mathcal{I}' \setminus \{u\}$
end procedure

lems. Indeed, comparisons between ratios of integers can be obviously done by comparing integer numbers, while all other operations only involve integer values. This is a notable difference with respect, e.g., to the approach based on pseudoflow computations. Indeed, such approach might be unable to recognize two distinct macroitems when the difference between their ratios is below a given numerical precision, as discussed in Section 6.

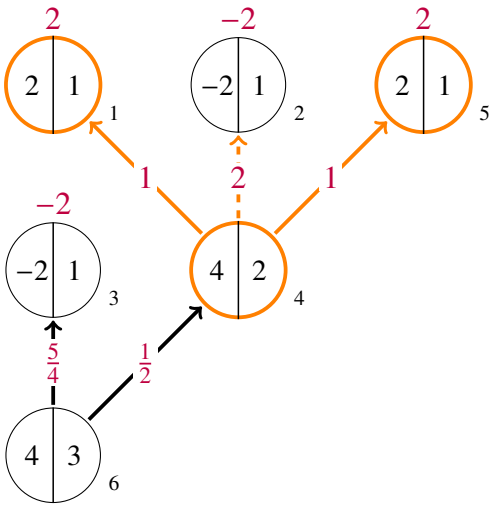
5.3 Dual Variant

Algorithm 1 finds the macroitems in the optimal sequence in decreasing order with respect to their ratio. It is possible to devise an analogous variant of Algorithm 1, which finds the macroitems in the optimal sequence in increasing order. In this variant, instead of computing the ratio of final items and preceding sets, we need to compute the ratio of *initial items* and *succeeding sets*, namely the sets $\bar{\mathcal{I}}_\mathrm{i}$, B_j and B_a already introduced in Section 5.1. A graphical illustration of initial items,

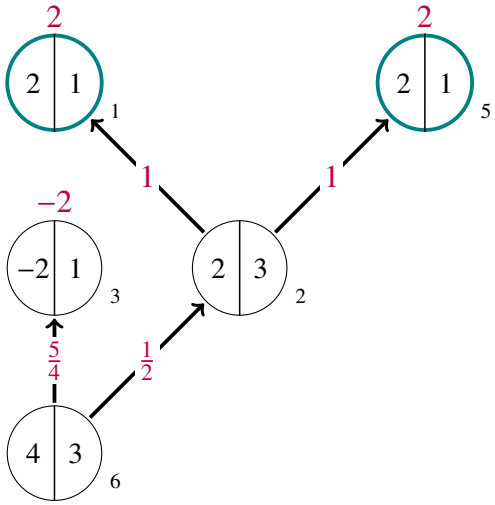
Figure 10: Run of Algorithm 1. Ratios on arcs/final items are shown in purple. At each step, the selected maximum-ratio part is highlighted (orange for arc-based selections, teal for final-item-only selections).



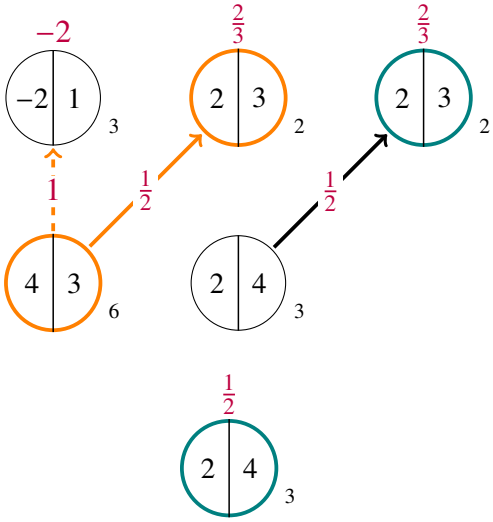
$a^* = (7, 4)$ with ratio $\frac{7}{2}$; $\mathcal{B}_f = \{1, 5\}$ with ratio 2



$a^* = (4, 2)$ with ratio 2; $\mathcal{B}_f = \{1, 5\}$ with ratio 2;
$\mathcal{M}_4 = \{4, 7\}$



$a^* = (6, 3)$ with ratio 1; $\mathcal{B}_f = \{1, 5\}$ with ratio 2;
$\mathcal{M}_2 = \{2, 4, 7\}$; $\mathcal{I}_1 = \cup_{j \in \mathcal{B}_f} \mathcal{M}_j = \{1, 5\}$



top-left: $a^* = (6, 3)$ with ratio 1; $\mathcal{B}_f = \{2\}$ with ratio $\frac{2}{3}$;
$\mathcal{M}_2 = \{2, 4, 7\}$
top-right: $a^* = (3, 2)$ with ratio $\frac{1}{2}$; $\mathcal{B}_f = \{2\}$ with ratio $\frac{2}{3}$;
$\mathcal{M}_2 = \{2, 4, 7\}$, $\mathcal{M}_3 = \{3, 6\}$; $\mathcal{I}_2 = \cup_{j \in \mathcal{B}_f} \mathcal{M}_j = \{2, 4, 7\}$
bottom: no arcs; $\mathcal{B}_f = \{3\}$ with ratio $\frac{1}{2}$; $\mathcal{I}_3 = \mathcal{M}_3 = \{3, 6\}$

and succeeding sets associated to items and arcs, can be found in Fig. 7.

We have the following three propositions, whose proofs are analogous to those of Propositions 4 to 6, and we omit them in this paper.

Proposition 7. *Suppose $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ is a directed forest and let $\mathcal{M} \subset \mathcal{I}$ be a subset of items. If $\mathcal{I}^-(\mathcal{M}) = \emptyset$, then*

$$\mathcal{M} = \left(\bigsqcup_{a \in \tilde{\mathcal{A}}} B_a \right) \sqcup \tilde{\mathcal{I}}_{\mathrm{i}}$$

for some subset $\tilde{\mathcal{A}} \subset \mathcal{A}$ and $\tilde{\mathcal{I}}_{\mathrm{i}} \subset \mathcal{I}_{\mathrm{i}}$, i.e., $\mathcal{M}$ is the disjoint union of the minimal succeeding sets associated to some arcs and some subset of initial items.

Proposition 8. *Suppose $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ is a directed forest and let $a^* \in \arg\min_{a \in \mathcal{A}} \frac{P(B_a)}{W(B_a)}$ with $a^* = (u, v)$. If $\frac{P(B_{a^*})}{W(B_{a^*})} \leq \frac{p_i}{w_i}$ for every $i \in \mathcal{I}_{\mathrm{i}}$, then the two items u and v belong to the same macroitem in the optimal sequence.*

Proposition 9. *Suppose $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ is a directed forest and let $g \in \mathcal{D}_{\mathrm{i}} := \arg\min_{i \in \mathcal{I}_{\mathrm{i}}} \frac{p_i}{w_i}$. If $\frac{p_g}{w_g} < \frac{P(B_a)}{W(B_a)}$ for every $a \in \mathcal{A}$, then $\mathcal{D}_{\mathrm{i}}$ is the last macroitem in the optimal sequence of macroitems.*

Thanks to these results, Algorithm 1 can be converted to find the macroitems in the optimal sequence in increasing order with respect to their ratio. We refer to this algorithm as the *dual variant* of Algorithm 1, and an illustration of a full run of this dual variant can be found in Fig. 11.

We also have the same complexity of $O(n)$ per iteration and $O(n^2)$ total complexity, for which we give only the statement, since the proof is analogous to the one of Theorem 5.

Theorem 6. *The dual variant of Algorithm 1 computes the optimal sequence of macroitems on a directed forest in $O(n^2)$ time.*

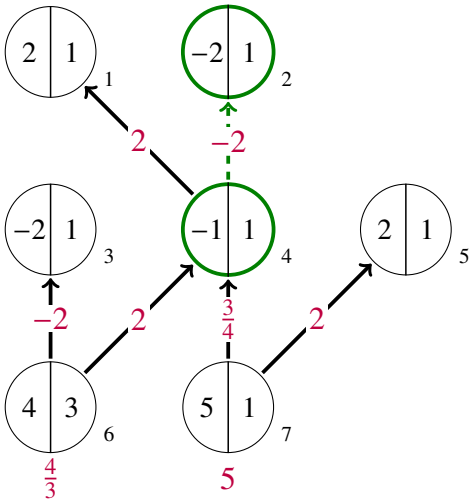
Remark 8. One can also combine the two approaches in a single iteration to halve the number of total iterations, but at the cost of increasing (doubling) the number of operations per iteration.

5.4 Improved variants with all out-trees or all in-trees

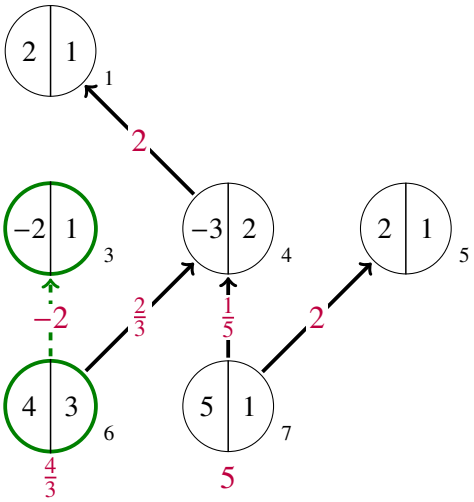
If we know that the forest graph $\mathcal{G}$ is composed of all trees whose arcs all point in the same direction, we can employ a heap data structure in Algorithm 1 and obtain a better total complexity of $O(n \log n)$. To describe this variant we first give the definition of out-tree and in-tree.

Definition 10. Let $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ be a directed connected tree. We say that $\mathcal{G}$ is an *in-tree* if for every item $j \in \mathcal{I}$ we have $|\mathcal{I}^+(j)| \leq 1$. As a consequence, there is only one item j_0 with $\mathcal{I}^+(j_0) = 0$, which we call the *root* of the in-tree.

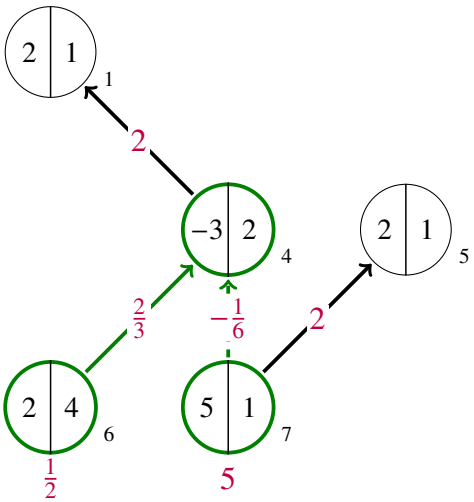
Figure 11: Run of the dual variant of Algorithm 1 on the same instance as Fig. 10. Ratios on arcs/initial items are shown in purple. At each step, the selected minimum-ratio part is highlighted (green for arc-based sets B_a, brown for initial-item selections).



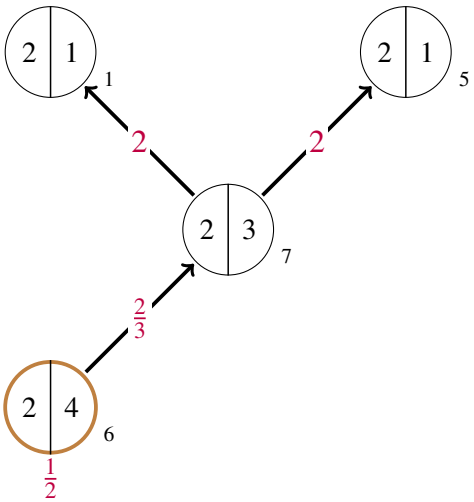
$a^* = (4, 2)$ with ratio -2; $\mathcal{D}_i = \{6\}$ with ratio $\frac{4}{3}$



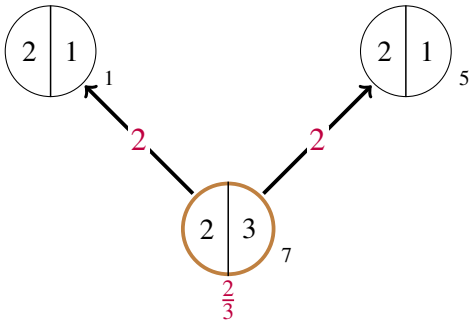
$a^* = (6, 3)$ with ratio -2; $\mathcal{D}_i = \{6\}$ with ratio $\frac{4}{3}$; $\mathcal{M}_4 = \{2, 4\}$



$a^* = (7, 4)$ with ratio $-\frac{1}{6}$; $\mathcal{D}_i = \{6\}$ with ratio $\frac{1}{2}$; $\mathcal{M}_4 = \{2, 4\}$, $\mathcal{M}_6 = \{3, 6\}$



$a^* = (6, 7)$ with ratio $\frac{2}{3}$; $\mathcal{D}_i = \{6\}$ with ratio $\frac{1}{2}$; $\mathcal{M}_7 = \{2, 4, 7\}$; $\mathcal{I}_3 = \mathcal{M}_6 = \{3, 6\}$



$a^* = (7, 1)$ with ratio 2; $\mathcal{D}_i = \{7\}$ with ratio $\frac{2}{3}$; $\mathcal{I}_2 = \mathcal{M}_7 = \{2, 4, 7\}$

no arcs; $\mathcal{D}_i = \{1, 5\}$ with ratio 2; $\mathcal{I}_1 = \{1, 5\}$

Definition 11. Let $\mathcal{G} = (\mathcal{I}, \mathcal{A})$ be a directed connected tree. We say that $\mathcal{G}$ is an *out-tree* if for every item $j \in \mathcal{I}$ we have $|\mathcal{I}^-(j)| \leq 1$. As a consequence, there is only one item j_0 with $\mathcal{I}^-(j_0) = 0$, which we call the *root* of the out-tree.

Algorithm 6 is devised for in-trees. In the initialization (lines 3-7) we create a copy of the set of items and we insert the elements in a heap with respect to their ratio. Then in the main cycle (lines 8-23), at each iteration we extract the item g at the top of the heap. If item g is a final item, we add it to the current (or to a new) macroitem (lines 11-17), we delete it from the graph and we update the heap accordingly. If item g is not a final item, we contract the only arc exiting g and we update the heap accordingly.

Theorem 7. *If $\mathcal{G}$ is a directed forest of in-trees, Algorithm 6 computes the optimal sequence of macroitems in $O(n \log n)$ time.*

Proof. Proof. We assume that the heap is a Fibonacci one (see, e.g., [5, Section 1.1]). Building the heap requires $O(n)$ operations. Next, at each iteration of the main cycle, we first remove the element with largest value from the heap, which requires $O(\log n)$. Next, when we are contracting an arc, we need to update the value of a single element in the heap. This, again, requires $O(\log n)$ operations. Since the number of iterations is $O(n)$, we can conclude that the overall complexity of Algorithm 6 is $O(n \log n)$.

□

Remark 9. Notice that Algorithm 6 is doing essentially the same operations of Algorithm 1, except the recomputation, at each iteration in the main cycle, of the values $P(F_j)$, $W(F_j)$ for every $j \in \mathcal{I}'$ (line Line 2 in Algorithm 2). Indeed, when the graph is only composed of in-trees, for every arc $a = (i, j) \in \mathcal{A}$ the set of items F_a is just the singleton $\{i\}$. Furthermore, if at the end of an iteration of the main cycle we remove some final items, we do not have to update any of the values $P(F_a), W(F_a)$ with $a \in \mathcal{A}$, since final items do not belong to sets F_a with $a \in \mathcal{A}$, when there are only in-trees. If instead, at the end of an iteration of the main cycle we contract an arc $a = (i, j) \in \mathcal{A}$, we only need to update the value of $F_{a'}$, where a' is the only arc exiting j, since in the graph there are only in-trees. Thus, exploiting the in-tree structure, we can employ a heap to extract the item with the largest ratio at each iteration, without unnecessary recomputation.

Remark 10. Remark 9 also explains why a heap data structure cannot improve the worst case complexity of Algorithm 1, when we have general trees. Indeed, if at the end of an iteration of the main cycle we are removing some final item $f \in \mathcal{I}'_{\mathrm{f}}$, then it is necessary to update the values of $P(F_a)$, $W(F_a)$ for every $a = (i, j) \in \mathcal{A}'$ with $i \in B_f$ and $j \notin B_f$, which could be a number of updates of the order of $O(n)$ in the worst case. Similarly, if at the end of an iteration of the main cycle we are contracting an arc $a^* \in \mathcal{A}'$, then it is necessary to update the values of $P(F_a)$, $W(F_a)$

Algorithm 6 Improved variant of Algorithm 1 to compute the optimal sequence of macroitems on a forest of in-trees

input $\mathcal{I}$, $\mathcal{A}$, $\mathcal{I}_\mathrm{f}$, $\boldsymbol{p}$, $\boldsymbol{w}$
▷ Initialization:
$\mathcal{I}' = \mathcal{I}$, $\mathcal{A}' = \mathcal{A}$, $\mathcal{I}_\mathrm{f}' = \mathcal{I}_\mathrm{f}$
for $j \in \mathcal{I}'$ **do**
    Set $p_j' = p_j$, $w_j' = w_j$ profit and weight for item $j \in \mathcal{I}'$; $\mathcal{M}_j = \{j\}$
end for
$k = 0$; Build max-heap with values $\frac{p_j'}{w_j'}$ for every $j \in \mathcal{I}'$; $r = +\infty$
while $\mathcal{I}' \neq \emptyset$ **do**
    Let $g \in \mathcal{I}'$ be the item at the top of the heap (i.e., one of the items with largest ratio)
    ▷ If top of the heap g is a final item, we add it to the next macroitem
    if $g \in \mathcal{I}_\mathrm{f}'$ **then**
        if $\frac{p_g'}{w_g'} < r$ **then** $k = k + 1$, $\mathcal{I}_k = \mathcal{M}_g$, $r = \frac{p_g'}{w_g'}$
        else $\mathcal{I}_k = \mathcal{I}_k \cup \mathcal{M}_g$
        end if
        for $i \in \mathcal{I}'^{-}(g)$ **do** $\mathcal{I}_\mathrm{f}' = \mathcal{I}_\mathrm{f}' \cup \{i\}$, $\mathcal{A}' = \mathcal{A}' \setminus \{(i, g)\}$
        end for
        $\mathcal{I}_\mathrm{f}' = \mathcal{I}_\mathrm{f}' \setminus \{g\}$, $\mathcal{I}' = \mathcal{I}' \setminus \{g\}$; remove g from heap and update heap
    ▷ If top of the heap g is not a final item, we "contract" the only arc (g, v) exiting g
    else
        $\mathcal{M}_v = \mathcal{M}_g \cup \mathcal{M}_v$, $p_v' = p_g' + p_v'$, $w_v' = w_g' + w_v'$
        $\mathcal{A}' = \mathcal{A}' \setminus \{(g, v)\}$, $\mathcal{I}' = \mathcal{I}' \setminus \{g\}$; update $\frac{p_v'}{w_v'}$ in heap
    end if
end while
return $\mathcal{S} = (\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k)$

for every $a = (i, j)$ with $i \in B_{a^*}$ and $j \notin B_{a^*}$, which could be a number of updates of the order of $O(n)$ in the worst case. Thus, maintaining a heap data structure with the relevant ratios, in the case of general trees, would cost $O(n \log n)$ operations per iteration of the main cycle, in the worst case, which would worsen the total complexity to $O(n^2 \log n)$ to complete all iterations, instead of $O(n^2)$ if we just recompute all relevant ratios at each iteration. Nevertheless, using a heap in the main algorithm still provides a notable speedup in practice, as observed in Section 6.

Remark 11. The dual variant of Algorithm 1 presented in Section 5.3, can also be improved using a heap-based data structure, in the case of a forest of out-trees. In this case we obtain an algorithm which computes the optimal sequence of macroitems in reverse order, i.e., starting from the macroitem with lowest ratio. The total complexity is $O(n \log n)$ also in this case. We state this result without proof, since it is analogous to the one of Theorem 7.

Theorem 8. *If $\mathcal{G}$ is a directed forest of out-trees, the dual variant of Algorithm 6 computes the optimal sequence of macroitems in $O(n \log n)$ time.*

6 Computational Results

This section evaluates the practical performance of the algorithms developed in Section 5. All algorithms were implemented in C++ and run single-threaded, and all experiments were conducted on a Linux machine equipped with an Intel i7-12700 processor (2.1 GHz) and 32 GB of RAM. We first describe the generation of the benchmark instances used throughout the section. We then justify the heap-based implementation of the main forest algorithm by comparing it with a non-heap implementation of the same algorithm. We then study how the heap-based algorithm scales on the instances with a large number of items, and we assess the impact of the specialized variants for in-trees and out-trees. Finally, we compare the heap-based forest algorithm with a bounded-precision implementation of the parametric pseudoflow approach.

6.1 Instance Generation

This section describes the generation of the benchmark instances used throughout the computational experiments. The instances are grouped by number of items, forest density, orientation pattern, and profit-weight correlation class.

Each benchmark instance is obtained by combining two independent components: the item coefficients, namely weights and profits, and the precedence graph. The item coefficients are generated following the standard test classes used for the classical knapsack problem [21]. For each item i, the weight w_i is sampled uniformly from $\{1, 2, \ldots, R\}$, with $R = 1000$. Profits are generated according to three classes: uncorrelated (`uncorr`), where p_i is sampled independently and uniformly from $\{1, 2, \ldots, R\}$; weakly correlated (`weakly-corr`), where p_i is sampled uniformly from the integer interval between $\max\{1, w_i - R/10\}$ and $w_i + R/10$; and strongly correlated (`strongly-corr`), where $p_i = w_i + R/10$. We also generate signed variants of these three classes by independently changing the sign of each profit with probability 0.25. We denote them by `uncorr-neg`, `weakly-corr-neg`, and `strongly-corr-neg`, respectively.

The second component is the precedence graph. We consider three families of directed forests. In an `in-forest`, every item has out-degree at most one. It is generated by scanning items $i \in \{1, 2, \ldots, n-1\}$ and, independently with probability ρ, adding one arc (i, j) with j chosen uniformly from $\{i+1, i+2, \ldots, n\}$. In an `out-forest`, every item has in-degree at most one. It is generated by scanning items $i \in \{2, 3, \ldots, n\}$ and, independently with probability ρ, adding one arc (j, i) with j chosen uniformly from $\{1, 2, \ldots, i-1\}$. Finally, in a `gen-forest`, we first generate an undirected forest by connecting each item $i \in \{2, 3, \ldots, n\}$ to a uniformly chosen predecessor $j < i$ with probability ρ, and then orient each selected edge independently in one of the two possible directions with probability 0.5. The density parameter is $\rho \in \{0.3, 0.6, 0.9, 1.0\}$, referred to as `sparse`, `medium`, `dense`, and `conn`, respectively. The first three density classes generate forests,

possibly with several connected components. The class `conn` corresponds to $\rho = 1.0$ and therefore generates a connected forest, which is in fact a single spanning tree. For every combination of topology, profit-weight class, density, and size, we generate 10 independent instances.

We use two main test beds in the computational experiments. The `medium-sized` test bed contains 7 200 instances: all three topologies, all six profit-weight classes, all four density values, and ten seeds for each size in $n \in \{100, 200, 300, \ldots, 1\,000\}$. The `large-sized` test bed contains 7 200 instances with the same combinations of topologies, profit-weight classes, densities, and seeds, for each size in $n \in \{10\,000, 20\,000, \ldots, 100\,000\}$.

6.2 Justification of the Heap-Based Variant

We implemented the main algorithm of the paper in a heap-based version, following the practical variant anticipated in Remark 10. We refer to this implementation as the *Heap-based Forest Macroitem Algorithm* (`HFMA`). To isolate the effect of the heap data structure, we compare `HFMA` with a non-heap implementation of the same forest macroitem algorithm, denoted by `FMA`. This implementation follows Algorithm 1 and uses the same incremental updates of closure sums as `HFMA`, but it selects the maximum-ratio final item or arc by direct linear scans, instead of maintaining the candidate ratios in a heap.

We tested both algorithms on the `gen-forest` instances of the `medium-sized` test bed, plus the sizes $n = 10\,000$ and $n = 20\,000$ of the `large-sized` test bed. Figure 12 reports the average CPU time of `HFMA` and `FMA` on these benchmark instances. Each point is the mean over all profit-weight classes, arc densities, and seeds available for that value of n.

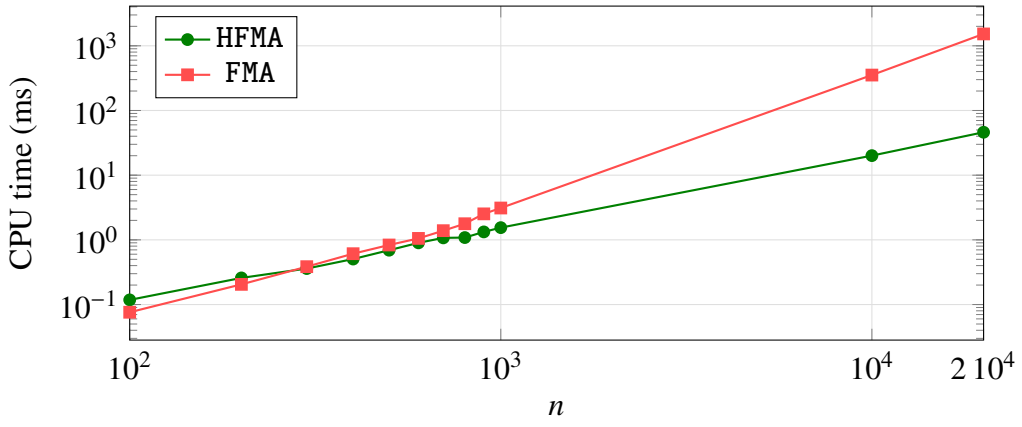


Figure 12: `HFMA` vs `FMA` CPU time on the `medium-sized` `gen-forest` benchmark instances, plus the sizes $n = 10\,000$ and $n = 20\,000$ of the `large-sized` test bed.

The two implementations have comparable running times on the smallest instances, where the overhead of maintaining a heap offsets the benefit of faster selection. As n grows, however, the

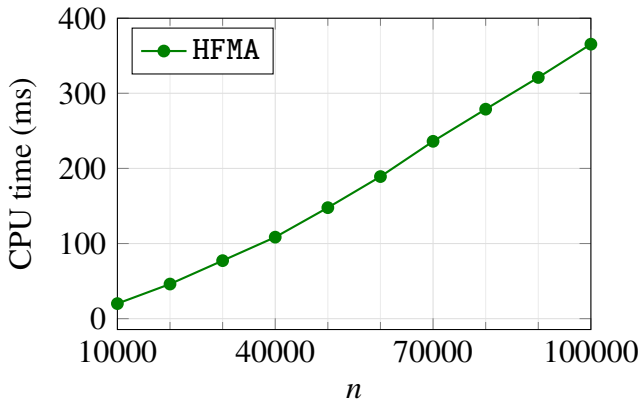
repeated linear scans of FMA become dominant. At $n = 1000$, FMA is about 2.0 times slower than HFMA; at $n = 10000$, the gap increases to about 17.6 times, and at $n = 20000$ it further widens to about 33.1 times. This confirms that the heap-based implementation, despite not being asymptotically preferable in the worst-case analysis, is suitable to obtain stable performance on larger forest instances.

6.3 Scaling of HFMA

We now evaluate the scalability of HFMA on the gen-forest instances of the large-sized test bed. Figure 13 reports, for each value of n, the number of tested instances and the corresponding mean CPU time. Each row averages over all profit-weight classes, arc densities, and seeds, for a total of 240 instances.

Figure 13: HFMA CPU time on the large-sized benchmark instances.

n	#inst	CPU time HFMA (ms)
10 000	240	20.1
20 000	240	46.1
30 000	240	77.2
40 000	240	108.5
50 000	240	147.8
60 000	240	189.1
70 000	240	236.1
80 000	240	278.8
90 000	240	321.1
100 000	240	365.4



The observed growth is smooth over the whole large-sized range. The CPU times appear to follow a $O(n \log n)$ growth. This empirical behavior is well below the worst-case complexity $O(n^2 \log n)$ of the heap-based implementation on general forests. But this is not in conflict with the theoretical analysis, since the complexity result refers to a worst-case analysis, whereas the reported times are averages over the generated benchmark instances.

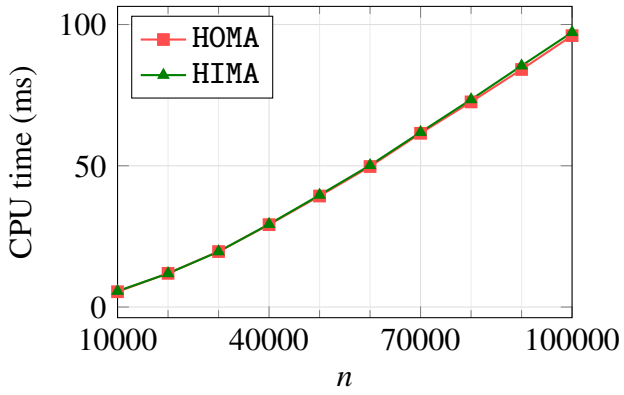
6.4 Specialized Variants for In-Trees and Out-Trees

We finally evaluate the specialized heap-based variants for forests composed only of in-trees or only of out-trees. We denote by HIMA the *Heap-based In-tree Macroitem Algorithm* and by HOMA the *Heap-based Out-tree Macroitem Algorithm*. The former is the algorithm of Section 5.4; the latter is the dual heap-based variant described in Remark 11. Both are tested on the corresponding large-sized instances.

Figure 14 reports the number of tested instances and the mean CPU times of the specialized algorithms.

Figure 14: Specialized heap-based variants on the `large-sized` benchmark instances.

		CPU-time (ms)	
n	#inst	HIMA	HOMA
10 000	240	5.6	5.4
20 000	240	11.9	11.9
30 000	240	19.6	19.6
40 000	240	29.3	29.2
50 000	240	39.6	39.3
60 000	240	50.2	49.7
70 000	240	61.9	61.5
80 000	240	73.5	72.6
90 000	240	85.4	84.1
100 000	240	97.2	96.1



In this case, the CPU times of both HIMA and HOMA display a $O(n \log n)$ growth, which is aligned with the worst-case complexity of these algorithms. To quantify the benefit of specialization, we compute, for each value of n, the ratio between the mean CPU time of HFMA and the mean CPU time of the specialized algorithm, both measured on the same set of instances (in-forest instances for HIMA, out-forest instances for HOMA). With this measure, HIMA is between 7.5 and 9.8 times faster than HFMA on in-forest instances, while HOMA is between 4.3 and 5.0 times faster than HFMA on out-forest instances.

6.5 Comparison with Pseudoflow

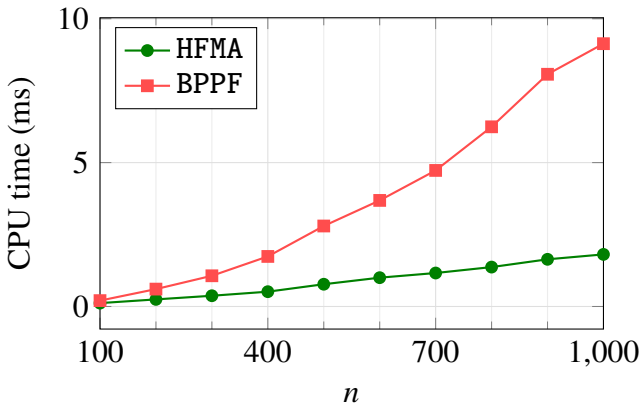
As a benchmark against a general parametric-flow approach, we use the *Bounded-Precision Parametric Pseudoflow* algorithm (BPPF), namely the public implementation of Hochbaum's parametric pseudoflow method available at https://github.com/hochbaumGroup/Bounded-precision-simple-para git.

We compare HFMA and BPPF on the `gen-forest` instances of the `medium-sized` test bed. This gives 2 400 instances, obtained from all 6 profit-weight classes, all 4 arc densities, and 10 seeds for each value of n. For each instance, HFMA computes the optimal sequence of macroitems and the corresponding breakpoint ratios. The same instance is also solved by BPPF, and the two outputs are compared in terms of the number and composition of macroitems.

Figure 15 reports the CPU time comparison. The table on the left gives, for each value of n, the number of tested instances, the average CPU time of HFMA and BPPF, and their ratio. The plot on the right represents the same average CPU times as a function of n.

Figure 15: `HFMA` vs `BPPF` on the `medium-sized gen-forest` benchmark instances. The table reports the number of instances, the plotted mean CPU times, and their ratio; the graph shows the corresponding CPU-time profiles as a function of n.

		CPU-time (ms)		
n	#inst	`HFMA`	`BPPF`	ratio
100	240	0.1	0.2	1.7
200	240	0.2	0.6	2.4
300	240	0.4	1.1	2.9
400	240	0.5	1.7	3.4
500	240	0.8	2.8	3.6
600	240	1.0	3.7	3.7
700	240	1.2	4.7	4.1
800	240	1.4	6.2	4.6
900	240	1.6	8.1	4.9
1000	240	1.8	9.1	5.0



The gap between `HFMA` and `BPPF` widens as n grows: the ratio `BPPF/HFMA` increases from about $1.7\times$ at $n = 100$ to about $5.1\times$ at $n = 1000$, roughly tripling over the range. This is consistent with `BPPF` solving a general parametric minimum-cut problem on the whole network, whereas `HFMA` exploits the special forest structure of the precedence graph directly, so its running time grows more slowly with n.

As a consistency check for the performance comparison, we also verified the macroitem partitions returned by the two algorithms. Since `BPPF` is a bounded-precision implementation, we ran it with a specified tolerance of 10^{-6} on the input coefficients and on the resulting breakpoint ratios. The number and composition of the macroitems coincide on 2 392 out of the 2 400 tested instances. In the remaining 8 instances, `BPPF` merges two consecutive macroitems whose breakpoint ratios differ by less than this tolerance. These cases occur only for instances with $700 \le n \le 1\,000$. This is consistent with the numerical-precision issue discussed in Remark 7. The precision parameter of `BPPF` cannot be increased without qualification: the implementation scales decimal capacities to integers, and using more decimal digits increases the magnitude of the internal integer coefficients, thereby creating overflow. We also tested `BPPF` on instances with more than 1000 items, but on these instances the same numerical-precision issues prevent a reliable comparison with `HFMA`.

7 Conclusion

In this paper we have studied the combinatorial structure of optimal solutions of the LP relaxation of the natural ILP formulation of the Precedence Constrained Knapsack Problem (PCKP).

Our central contribution is the introduction of the concept of *optimal sequence of macroitems*, an ordered partition of the item set into precedence-closed groups ranked by nonincreasing profit-to-

weight ratio, and the proof that this sequence fully characterizes optimal LP solutions. Specifically, the optimal LP solution assigns value 1 to all items in macroitems preceding the split macroitem, a common fractional value to all items in the split macroitem, and 0 to all remaining items. This result generalizes the classical greedy structure of LP-optimal solutions, where the role of individual items is taken over by macroitems.

We have further characterized the structure of optimal dual solutions, showing that they correspond to feasible flows within each macroitem of the optimal sequence. As an additional consequence, we have identified the optimal Lagrangian multiplier for the capacity constraint as the profit-to-weight ratio of the split macroitem, recovering the analogue of the classical result for the standard Knapsack Problem.

On the algorithmic side, we have presented an $O(n^2)$ algorithm for computing the optimal sequence of macroitems when the precedence graph is a directed forest, based on iterative contraction of the arc with the highest profit-to-weight ratio among all minimal preceding sets. When the forest is composed of in-trees (or out-trees), we have shown that a heap-based variant of the algorithm achieves $O(n \log n)$ complexity, by exploiting the fact that each arc's minimal preceding set is a singleton in this case. As a by-product, since its optimal solution coincides with the first (resp. last) macroitem of the optimal sequence, the same algorithms also compute the optimal solution of the precedence-constrained ratio optimization problem (14) introduced in Section 2.3, within the same complexity bounds.

An interesting open future research direction is to study whether efficient algorithms for computing the optimal sequence of macroitems can be devised beyond the forest case, and whether the LP structure identified here can be embedded into branch-and-bound frameworks to yield improved exact algorithms for the PCKP.

Code and Data Availability

The source code and computational material used in the experiments are available online, with the aim of stimulating further research on these topics and facilitating reproducible comparisons. The code repository is https://github.com/fabiofurini/macroitems-cpp; it contains the C++ implementation of the forest macroitem algorithms and the scripts used to reproduce the benchmark runs. The data repository is the `data` directory of the same GitHub repository: https://github.com/fabiofurini/macroitems-cpp/tree/main/data; it contains the raw CSV files underlying the computational figures and tables. In particular, the data files report one row per tested instance and include the instance class, topology, density, seed, number of macroitems, and running time. The pseudoflow comparison uses the public Bounded-Precision Parametric Pseudoflow implementation available at https://github.com/hochbaumGroup/

Bounded-precision-simple-parametric.git.

References

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows: Theory, algorithms, and applications*. Prentice-Hall, Inc., 1993.

[2] Michel L Balinski. On a selection problem. *Management Science*, 17(3):230–231, 1970.

[3] F Barahona and D. Jensen. Plant location with minimum inventory. *Mathematical Programming*, 83 (1):101–111, 1998.

[4] Natashia Boland, Andreas Bley, Christopher Fricke, Gary Froyland, and Renata Sotirov. Clique-based facets for the precedence constrained knapsack problem. *Mathematical Programming*, 133:481–511, 2012.

[5] Gerth Stølting Brodal, George Lagogiannis, and Robert E. Tarjan. Strict fibonacci heaps. *ACM Transactions on Algorithms*, 21(2), January 2025.

[6] Abraham Charnes and William W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3–4):181–186, 1962.

[7] Renaud Chicoisne, Daniel Espinoza, Marcos Goycoolea, Eduardo Moreno, and Enrique Rubio. A new algorithm for the open-pit mine production scheduling problem. *Operations Research*, 60(3):517–528, 2012.

[8] Denis Cornaz, Fabio Furini, Mathieu Lacroix, Enrico Malaguti, A Ridha Mahjoub, and Sébastien Martin. The vertex k-cut problem. *Discrete Optimization*, 31:8–28, 2019.

[9] Daniel Espinoza, Marcos Goycoolea, Eduardo Moreno, and Alexandra Newman. Minelib: A library of open pit mining problems. *Annals of Operations Research*, 206:93–114, 2013.

[10] Daniel Espinoza, Marcos Goycoolea, and Eduardo Moreno. The precedence constrained knapsack problem: Separating maximally violated inequalities. *Discrete Applied Mathematics*, 194:65–80, 2015.

[11] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[12] Dorit S Hochbaum. A new–old algorithm for minimum-cut and maximum-flow in closure graphs. *Networks*, 37(4):171–193, 2001.

[13] Dorit S Hochbaum. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations research*, 56(4):992–1009, 2008.

[14] Dorit S Hochbaum and Anna Chen. Performance analysis and best implementations of old and new algorithms for the open-pit mining problem. *Operations Research*, 48(6):894–914, 2000.

[15] Oscar H Ibarra and Chul E Kim. Approximation algorithms for certain scheduling problems. *Mathematics of Operations Research*, 3(3):197–204, 1978.

[16] David S Johnson and KA Niemi. On knapsacks, partitions, and a new dynamic programming technique for trees. *Mathematics of Operations Research*, 8(1):1–14, 1983.

[17] Thys Brentwood Johnson. *Optimum open pit mine production scheduling*. University of California, Berkeley, 1968.

[18] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin Heidelberg, 2004.

[19] Helmut Lerchs and Ingo Grossmann. Optimum design of open-pit mines. *CIM bulletin*, 58:47–54, 1965.

[20] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, New York, 1990.

[21] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, 1999.

[22] Jean-Claude Picard. Maximal closure of a graph and applications to combinatorial problems. *Management Science*, 22(11):1268–1272, 1976.

[23] Siegfried Schaible. Fractional programming. I. duality. *Management Science*, 22(8):858–867, 1976.

[24] William F. Sharpe. Mutual fund performance. *The Journal of Business*, 39(1):119–138, 1966.

[25] Dong X Shaw, Geon Cho, and Hsuliang Chang. A depth-first dynamic programming procedure for the extended tree knapsack problem in local access network design. *Telecommunication Systems*, 7:29–43, 1997.

[26] Kathryn E Stecke and Ilyong Kim. A study of fms part type selection approaches for short-term production planning. *International Journal of Flexible Manufacturing Systems*, 1:7–29, 1988.