LAD models, trees and an analog of the fundamental theorem of arithmetic

Nadia Brauner, Sylvain Gravier, Louis-Philippe Kronek, Frédéric Meunier

LAD MODELS, TREES AND AN ANALOG OF THE FUNDAMENTAL THEOREM OF ARITHMETIC

NADIA BRAUNER, SYLVAIN GRAVIER, LOUIS-PHILIPPE KRONEK, AND FRÉDÉRIC MEUNIER

Abstract. Motivated by applications of Logical Analysis of Data (LAD) in medical contexts, original discrete optimization problems are proposed. When a patient arrives with a presumption of a disease, he is submitted to a sequence of tests. From one patient to another, the tests allowing to detect the disease may vary. A subset of tests whose results detect the disease in a given part of the population is called a *pattern*, which has its own prevalence in the population.

If there is only a limited number of tests that can be done, which ones must be selected in order to maximize the number of spotted patients ? Or, if each test has a cost, in which order the tests have to be done, in order to minimize the cost ? It is the kind of questions that are investigated in this paper. For various special cases, polynomial algorithms are proposed, especially when the hypergraph whose vertices are the tests and whose edges are the patterns is a tree graph.

One of these questions involves a criterion which is not a number but a sequence of numbers. The objective is then to find the best sequence for the lexicographic order. To solve this question, a new product on finite sequences is defined, namely the *maximum shuffle product*, which maps two sequences to their shuffle that is maximal for the lexicographic order. Surprisingly, this product leads to a theorem similar to the fundamental theorem of arithmetic: every sequence can be written uniquely as the product of *prime sequences*, with the suitable definition of prime sequences.

1. Introduction

1.1. **Context.** Discrete mathematics and medicine see their interactions growing these last years. This is particularly true for the area of classification models, which help practitioners to make diagnosis on the basis of patterns (rules). Formalizing questions arising when one implements a set of patterns (a model), we get nice combinatorial optimization problems. This paper aims to present a series of such problems, to focus on their relationships to issues already addressed in former papers and to extract nice combinatorial questions and properties.

Logical Analysis of Data (LAD) aims at helping practitioners for decision making when a new patient arrives with a presumption of a disease. It is a rule based method where a classification model is composed of a set of patterns [5]. Assume that we are given possible tests (*e.g.* age, headache...). From one patient to another, the tests allowing to confirm the disease may vary. A *pattern* is a boolean conjunction on the results of tests. If a patient can be *classified* by these results, a positive (resp. negative) class meaning having (resp. not having) the disease, we say that the pattern *covers* the patient. We shall consider an

Date: September 9, 2011.

Key words and phrases. decomposition theorem, lexicographic order, logical analysis of data, product of sequences, tree.

approximation of the global problem with the following hypothesis: every patient is covered by exactly one pattern.

The problems we are interested in concern for instance the ordering of the tests allowing to classify a new patient as soon as possible, or the selection of a subset of tests if all tests can not be done. To estimate the quality of an ordering, or of a subset of the tests, a database is used. It is composed of already classified patients, for whom there were a presumption of the disease. The prevalences of the patterns in the database are supposed to be good approximations of the pattern prevalences among the future patients, who will have to be classified. Therefore, when a solution (an ordering, a subset, ...) is proposed, its quality is measured by its results on the database.

A pattern is identified with the subset of tests whose results it takes into account. We will often work with the *pattern-test hypergraph* $H = (V, E)$, having the tests as vertex set and the patterns as edge set. The prevalence of a pattern $e \in E$ is the number $p(e)$ of patients of the database covered by this pattern.

Table 1 presents a LAD model where the patterns are based on 4 tests. The last column gives the number of patients covered by the pattern in the database. In terms of the pattern-test hypergraph, $V = \{1, 2, 3, 4\}$ and $E = \{1, 23, 13, 14\}$, with $p(1) = 18$, $p(23) = 10$, $p(13) = 70$ and $p(14) = 38$.

Pattern	Test 1	Test 2	Test 3	Test 4	Class	Cover
P_1	> 30				positive	18
P_2		negative	positive		positive	10
P_3	< 15		negative		negative	70
P_4	< 15			positive	negative	38

TABLE 1. A LAD model

Note that when the tests are done one after the other, each time a complete pattern is contained in the tests already done, all the patients in the database covered by this pattern can be classified. With the example of Table 1, after Test 1 it is possible to classify 18 people as positive. If the next test is Test 3, 70 additionnal people are classified, in this case as negative. So after Test 1 and Test 3 we get a total of 88 people classified.

If each test v has a specific cost $c(v)$, tests with high costs have to be at the end of the process, since we can then hope that most of the patients have already been spotted. It suggests the following problem.

Problem TESTING COST
INSTANCE: A hypergraph $H = (V, E)$ with n vertices, two maps $p : E \to \mathbb{R}_+ \setminus \{0\}$ and $c : V \to \mathbb{R}_+$
SOLUTION: An ordering $v_1, \ldots, v_n$ of the vertices such that

$$\sum_{i=1}^{n} \sum_{e \in E \setminus E[v_1,\ldots,v_{i-1}]} p(e)c(v_i)$$

is minimum.

This problem was initiated when working on the diagnosis of growth hormone deficiency [9]. During the diagnosis process many practitioners are involved successively (general practitioners, biologists, endocrinologists, radiologists) and the tests (height, bone age, karyotype...) have different performances and costs.

Remark. In this model, any patient whose status has not been settled will be subject to the tests. However, in many cases, the first results of the tests will be sufficient for some patients to predict the result of the next test, and hence that this next test is not necessary for these patients. The modelling of such features deserves future works.

Finding the k tests covering the most patients can be formalized as follows.

Problem Extraction of a sub-model
Instance: A hypergraph $H = (V, E)$ with n vertices, a map $p : E \to \mathbb{R}_+ \setminus \{0\}$ and an integer k.
Solution: A subset $S \subseteq V$ of cardinality at most k maximizing $p(E[S])$.

This problem appeared when working on the diagnosis of pulmonary embolism. The medical question was to identify the first tests to be implemented at the emergency room in order to be able to make rapidly the greatest number of diagnosis for this pathology.

We may want to give a complete ordering $v_1, \ldots, v_n$ of the tests in such a way that if for a reason we do not want to do all tests and we stop them say at v_k, we do not regret our choice of an ordering. We write $\preceq$ the lexicographic order on sequences.

Problem Non-dominated ordering
Instance: A hypergraph $H = (V, E)$ with n vertices and a map $p : E \to \mathbb{R}_+ \setminus \{0\}$
Solution: An ordering $v_1, \ldots, v_n$ of the vertices such that the sequence $h_1, \ldots, h_n$, with $h_i = p(E[\{v_1, \ldots, v_i\}])$, is maximal for the lexicographic order.

It is also interesting to compare two sequences with respect to the Pareto-order instead of lexicographic order.

A *Pareto-optimal ordering* $v_1, \ldots, v_n$ is such that for every other ordering $v'_1, \ldots, v'_n$, either there is a $i \in \{1, \ldots, n\}$ with $h'_i < h_i$, or for all $i \in \{1, \ldots, n\}$ we have $h'_i = h_i$. A sequence maximal for the lexicographic order is Pareto-optimal, but the converse is not true.

Problem Pareto-optimal ordering
Instance: A hypergraph $H = (V, E)$ with n vertices and a map $p : E \to \mathbb{R}_+ \setminus \{0\}$
Solution: An ordering $v_1, \ldots, v_n$ of the vertices such that the sequence $h_1, \ldots, h_n$, with $h_i = p(E[\{v_1, \ldots, v_i\}])$, is Pareto-optimal.

1.2. **Main results.** Testing costs, Extraction of a sub-model, and Non-dominated ordering are checked to be NP-hard, even if the hypergraph H is a graph in the case of the last two, but the complexity of Pareto-optimal ordering is open.

However, we prove the following

Proposition 1.1. Pareto-optimal ordering *is polynomial (in $O(n^2 \log n)$) when the hypergraph is a tree graph.*

Theorem 1.2. Non-dominated ordering *is polynomial (in $O\left(n^3 \log n \log d\right)$) when the hypergraph is a tree graph and d its maximal degree.*

The polynomiality stated in Proposition 1.1 is a corollary of Theorem 1.2, since as we have already noted, an ordering of the tests that is maximal for the lexicographic order is necessarily Pareto-optimal. Proposition 1.1 finds its interest in a better complexity ($O(n^2 \log n)$ versus $O\left(n^3 \log n \log d\right)$).

The proof of Theorem 1.2 motivates the definition of shuffles of sequences.

A sequence $\boldsymbol{c} = c_1 \dots c_n$ is a *shuffle* of s sequences $\boldsymbol{a}_i = a_{i1} \dots a_{in_i}$, $i = 1, \dots, s$ if

- $n = \sum_{i=1}^{s} n_i$.
- there are s strictly increasing functions $\alpha_i : [n_i] \to [n]$ – the *prints* – whose images are pairwise disjoint and such that $c_{\alpha_i(j)} = a_{ij}$ for all i, j.

As an example, take $\boldsymbol{a} = 21, 32, 2, 4$ and $\boldsymbol{b} = 4, 1, 5$; the sequence $\boldsymbol{c} = 21, 4, 32, 2, 1, 5, 4$ is a shuffle of them.

Denote by $\mathrm{ms}(\boldsymbol{a}, \boldsymbol{b})$ the shuffle of two sequences of real numbers $\boldsymbol{a}$ and $\boldsymbol{b}$ that is maximal for the lexicographic order $\preceq$. With $\boldsymbol{a}$ and $\boldsymbol{b}$ as in the previous example, we get $\mathrm{ms}(\boldsymbol{a}, \boldsymbol{b}) = 21, 32, 4, 2, 4, 1, 5$.

We discuss the way for computing $\mathrm{ms}(\boldsymbol{a}, \boldsymbol{b})$ given two sequences $\boldsymbol{a}$ and $\boldsymbol{b}$. Theorem 1.2 is a consequence of the polynomiality of this operation.

On our track, we discovered a combinatorial analog of the fundamental theorem of arithmetic, which states the uniqueness of decomposition of an integer as product of prime numbers. To our knowledge, this combinatorial analog is completely new. There exist other decompositions theorems which generalize the classical one (see [11] for instance), but they do not contain it.

We define a notion of prime sequences. A *prime sequence* $\boldsymbol{p} = p_1 \dots p_n$ is such that $p_i p_{i+1} \dots p_n \succeq \boldsymbol{p}$ for each $i \geq 1$. For instance, the sequence $2, 14, 13, 6$ is prime, but not the sequence $2, 14, 13, 6, 2$ since $2 \prec 2, 14, 13, 6, 2$.

Theorem 1.3. *Each sequence $\boldsymbol{a}$ of real numbers can be written as a unique maximum shuffle (up to permutation)*

$$\boldsymbol{a} = \mathrm{ms}\left(\boldsymbol{q}_1^{\delta_1}, \boldsymbol{q}_2^{\delta_2}, \dots, \boldsymbol{q}_s^{\delta_s}\right)$$

where the δ_j are positive integers and the $\boldsymbol{q}_j$ are prime.

Here, $\boldsymbol{q}^{\delta}$ denotes $\mathrm{ms}(\underbrace{\boldsymbol{q}, \dots, \boldsymbol{q}}_{\delta \text{ times}})$.

For example, the sequence $18, 17, 19, 18, 17, 19, 5, 18$ can be written as the product of prime sequences $\mathrm{ms}(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4)$ with $\boldsymbol{p}_1 = 18$, $\boldsymbol{p}_2 = 17, 19, 18$, $\boldsymbol{p}_3 = 17, 19$ and $\boldsymbol{p}_4 = 5, 18$.

1.3. **Notations.**

1.3.1. *Hypergraphs.* A hypergraph is denoted $H = (V, E)$. The number n is usually its number of vertices. Two vertices u and v are *neighbors* is there is an edge $e \in E$ such that $\{u, v\} \subseteq e$. By $N(A)$, for A a subset of vertices, we denote the set of vertices in $V \setminus A$ having at least one neighbor in A. By $E[X]$, for a subset $X \subseteq V$ of vertices, we mean the subset of edges having all their vertices in X.

1.3.2. *Combinatorics.* Let A be an additive semi-group and w be any map from a finite set X into A. For F a subset of X, we denote $\sum_{x \in F} w(x)$ by $w(F)$.

Given a set E endowed with a linear order $\leq$, we recall that the *lexicographic order* $\preceq$ on the set of all finite sequences of elements of E is defined as follows:

- $\boldsymbol{\epsilon} \preceq \boldsymbol{a}$ with $\boldsymbol{\epsilon}$ being the empty sequence and $\boldsymbol{a}$ being any finite sequence,
- $a_1 a_2 \dots a_m \preceq b_1 b_2 \dots b_n$ if $a_1 < b_1$ or if $a_1 = b_1$ and $a_2 a_3 \dots a_m \preceq b_2 b_3 \dots b_n$.

1.3.3. *Scheduling.* Along the paper, we use sometimes equivalence with scheduling problems. We use the classical $\alpha|\beta|\gamma$ notation in scheduling where α denotes the properties of the machines (or the processors), β indicates the constraints on the tasks and γ describes the objective function. In this paper, α is equal to 1 since we consider one machine problems. The field β contains information on the precedence graph for the tasks (where 'prec' means that the precedence graph can be any acyclic graph). The objective in the field γ is $\sum w_i C_i$ if we want to minimize the total weighed completion time where w_i is the weight of task i and C_i is its completion time. It is $\sum w_i U_i$ when we want to minimize the weighted number of late jobs (where $U_i = 1$ if task i completes after its dead line, $U_i = 0$ otherwise).

1.4. **Plan.** Each of the Sections 2, 3, 4 and 5 presents various resultsfor one of the 4 problems defined above. In particular, Theorem 1.2 is proved in Section 4 and Proposition 1.1 is proved in Section 5. Section 6 is devoted to the proof of the analog of the fundamental theorem of arithmetic for sequences and shuffles (Theorem 1.3).

2. Testing Cost

We show that the problem Testing Cost is equivalent to a scheduling problem on one machine with a specific precedence graph and the total weighted completion time as the objective function: $1|prec|\sum w_i C_i$ in the usual scheduling notations.

We now define the tasks and the precedence graph.

- To each vertex v in the hypergraph is associated a (vertex-)task of duration $t_v = c(v)$ and of weight $w_v = 0$;
- To each edge e in the hypergraph is associated a (edge-)task of duration $t_e = 0$ and of weight $w_e = p(e)$;

The precedence graph naturally follows: a edge-task can be executed once all the vertex-tasks coming from vertices in the edge are completed. Figure 1 presents the scheduling problem associated to the model from Table 1.

	Test 1	Test 2	Test 3	Test 4
Costs	1	5	15	8

Table 2. Costs for the test of Table 1

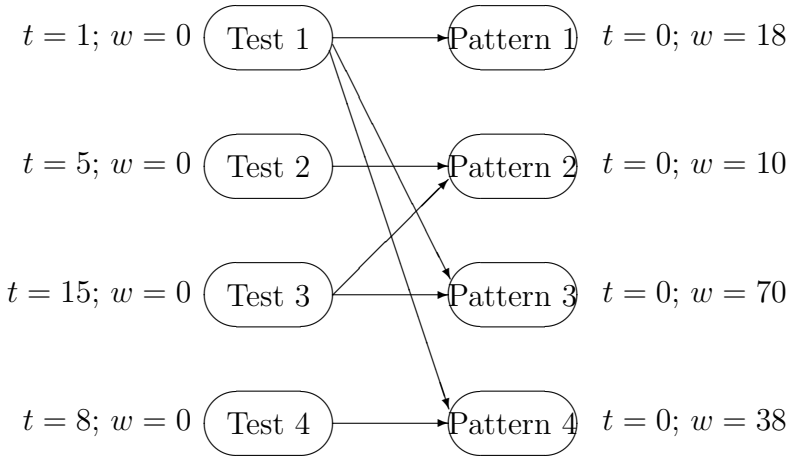

FIGURE 1. The precedence graph associated to the model of Table 1 and the costs of Table 2

Proposition 2.1. *Problem* TESTING COST *is equivalent to the scheduling problem* $1|prec|\sum w_iC_i$ *with the tasks and the precedence graph defined as above.*

Proof. Given a solution of the problem TESTING COST, we get a feasible solution for problem $1|prec|\sum w_iC_i$ of same cost while starting a edge-task as soon as all vertex-tasks it requires are finished. Conversely, an optimal solution of problem $1|prec|\sum w_iC_i$ is such that a edge-task starts as soon as all vertex-tasks it requires are finished and the sequence of vertices the solution induces is a solution of same cost for TESTING COST. □

The precedence graph has the following property: if there is a precedence constraint between tasks i and j of the form $i \to j$, then $w_i = 0$ and $t_j = 0$. This type of graphs is known as *red-blue bipartite graphs* [7].

The scheduling problem $1|prec|\sum w_iC_i$ is NP-hard for a general precedence graph [8]. We consider the special case of red-blue bipartite graphs for the precedence constraints with the restriction $p_i \in \{0, 1\}$ and $w_i \in \{0, 1\}$. This theoretical situation corresponds to the very special case where each pattern covers exactly one patient and each test has a unitary cost. It has been proved in [12] that this special case has the same approximation ratio that the general problem. TESTING COST is therefore NP-hard.

3. EXTRACTION OF A SUB-MODEL

3.1. **The densest subgraph.** The special case of the problem EXTRACTION OF A SUB-MODEL when H is a graph has been widely studied and is better known under the name DENSEST k-SUBGRAPH PROBLEM. It is an NP-hard problem since when $p(e) = 1$ for all e it becomes the problem of selecting the subset X of vertices with $|X| = k$ such that $E[X]$ has maximal cardinality, which contains the maximum clique problem. It is known to be polynomial on trees, with a complexity $O(k^2n)$ [10].

3.2. **As a scheduling problem.** The problem EXTRACTION OF A SUB-MODEL can also be linked to a scheduling problem similar to the one of Section 2. There is still one machine and the precedence graph is the same. The tasks are defined as follows :

- To each vertex v is associated a (vertex-)task of duration $t_v = 1$ and of weight $w_v = 0$ with no deadline $d_v = +\infty$;
- To each edge e is associated a (edge-)task of duration $t_e = 0$ and of weight $w_e = p(e)$ and of deadline $d_e = k$.

The objective function is now the minimization of the number of weighted tasks that finish late, $\sum w_i U_i$ where U_i indicates whether task i finishes after its deadline. The edge-tasks that are not late compose the submodel S.

We have therefore the following proposition. The proof is omitted since it is very similar to the one of Proposition 2.1.

Proposition 3.1. *Problem* EXTRACTION OF A SUB-MODEL *is equivalent to the scheduling problem* $1|prec|\sum w_i U_i$ *with the tasks and the precedence graph defined as above.*

This problem has been widely studied under different forms, see e.g. [2]. For instance, EXTRACTION OF A SUB-MODEL is equivalent to the tool magazine problem where the tests are the tools, the patterns are the parts to be produced and the prevalence is the demand for the parts. The part-tools matrix indicating for each part the tools needed for its production is then the incidence matrix of the hypergraph H. The value k defines the capacity of the tool magazine and the objective is to select k tools for the magazine that allow the satisfaction of the largest demand. This problem of the tool magazine is known to be NP-hard even if the number of tools for each part is less than 2 [4]. Crama [2] has given linear and non-linear formulations of the problem, proposed solutions methods and listed other applications of this model like repair kit selection or allocation of memory space in a database. It is mentioned in this paper that those practical problems are solved with ad-hoc heuristics. In [3], the authors present a worst case analysis of the greedy heuristics usually developed for this problem. They also conjecture that there is no polynomial time algorithm with a constant performance ratio.

4. NON-DOMINATED ORDERING

4.1. **First properties.** Defining $p(e)$ to be 1 for all $e \in E$, a maximal ordering for the lexicographic order necessarily starts with the vertices of a maximum clique. We have therefore the following complexity result.

Proposition 4.1. NON-DOMINATED ORDERING *is NP-hard even if the hypergraph H is a graph.*

As a preliminary remark on the problem, we can notice that

Lemma 4.2. *Let $v_1, \ldots, v_n$ be a solution of* NON-DOMINATED ORDERING. *For all $i < n$, if $N(v_1, \ldots, v_i) \neq \emptyset$, then $v_{i+1} \in N(v_1, \ldots, v_i)$.*

Proof. Assume that we have i such that $N(v_1, \ldots, v_i) \neq \emptyset$. Choosing v_{i+1} not in $N(v_1, \ldots, v_i)$ leads to $h_{i+1} = h_i$ whereas choosing it in $N(v_1, \ldots, v_i)$ leads $h_{i+1} > h_i$, without changing the values of h_j for $j < i$. □

4.2. **A greedy algorithm.** We propose now a greedy approach.

At each step, add the minimal number of vertices such that the induced subhypergraph contains a new edge. If several new edges can be obtained in this way, select the one with maximal value p. At the end of each iteration, the hypergraph is updated by merging the edges with the same sets of missing vertices. The new p is the sum of the values of p of the merged edges.

Greedy algorithm (a hypergraph H)
// Create an order with a greedy method
$Ordering \leftarrow \emptyset$ and $ActiveEdges \leftarrow \emptyset$
While $Ordering$ does not contain all the vertices do
    $ActiveEdges \leftarrow$ edges from hypergraph H that have the smallest number of vertices not in $Ordering$
    Add to $Ordering$ the vertices completing an edge in $ActiveEdges$ with the maximal p.
    Merge the edges with identical sets of vertices not in $Ordering$ and update the values of p.
end While
Return $Ordering$

Unfortunately, this greedy algorithm does not always lead to an ordering maximal for the lexicographic order $\preceq$. Consider, for example, the model presented in Table 3. The order Test 1, Test 2, Test 3, Test 4, Test 5 can be obtained by the greedy algorithm. It is dominated by the order Test 4, Test 3, Test 5, Test 2, and Test 1. The following proposition gives a sufficient condition for the order produced by the greedy algorithm to be non-dominated.

Patterns	Test 1	Test 2	Test 3	Test 4	Test 5	Cover
P_1	1	1	0	0	0	1
P_2	0	0	1	1	0	1
P_3	0	0	0	1	1	1

TABLE 3. A model for which the greedy algorithm does not give a non-dominated order

Proposition 4.3. *If, at each iteration, there is only one active edge of maximum prevalence, then the result of the greedy algorithm is an optimal solution of* NON-DOMINATED ORDERING.

Proof. The proof is achieved by induction on the iterations of the algorithm. At each step, we can not do better than selecting the vertices of this active edge. □

The sufficient condition presented in Proposition 4.3 is realistic in practice. The probability is low that there exist, at some time in the execution of the algorithm, two distinct updated edges that have the same prevalence and that need the realization of a minimal number of tests. Indeed, each of such updated edges comes from a distinct set of edges of the original hypergraph and, generically, these two sets of edges do not have the same total prevalence or, in other words, generically, the hyperplane $\sum_{e\in E} p(e)x_e = 0$ does not contain any vertex of the hypercube $[-1, 1]^{|E|}$. An interesting question could be to characterize the models that satisfy the condition of Proposition 4.3.

4.3. **Tree.** The purpose of this subsection is to prove Theorem 1.2. As a by-product, we get some properties of the *shuffle* and the ms operation (defined in the introduction). In Section 6, it will lead to the analog of the fundamental theorem of arithmetic we have mentioned.

4.3.1. *Preliminary discussion.*

Lemma 4.4. *Let $a_1 \dots a_n$ and $b_1 \dots b_n$ be two sequences of real numbers (of same length). $a_1 \dots a_n \preceq b_1 \dots b_n$ if and only if $a_1(a_2 - a_1)(a_3 - a_2) \dots (a_n - a_{n-1}) \preceq b_1(b_2 - b_1)(b_3 - b_2) \dots (b_n - b_{n-1})$.*

Proof. Let i^* be the first index i such that $b_i \neq a_i$. We have $b_{i^*} > a_{i^*}$ if and only if $b_{i^*} - b_{i^*-1} > a_{i^*} - a_{i^*-1}$. □

Lemma 4.4 combined with Lemma 4.2 shows that finding a maximal ordering for the lexicographic order when the hypergraph H is a tree T reduces to the following problem. Given r a special vertex in T, and a weight function $w : V(T) \to \mathbb{R}_+ \setminus \{0\}$ (the weight of a vertex v being the original weight of the unique edge emanating from v toward r), compute an ordering $r = v_1, \dots, v_n$ of the vertices such that $v_{i+1} \in N(v_1, \dots, v_i)$ for all i and such that $w(v_2), \dots, w(v_n)$ is maximal for the lexicographic order. Doing the same computation for each vertex r of T, and keeping the maximal one gives the solution of the original problem.

Note that the question of finding such an ordering starting from vertex r is easy if all weights in the tree are distinct. Indeed, at each time, selecting the edge with maximal weight among the edges leaving the set of vertices already reached clearly provides the maximal ordering (Proposition 4.3). The difficulty arises precisely when there are many weights that are equal. Choosing then an edge with maximal weight could eventually lead to lower weights later than those that would have been obtained by choosing another edge of the same maximal weight.

Deleting r from T gives raise to say s subtrees $T_1, T_2, \dots, T_s$. As we will prove below, once we have a maximal ordering of the vertices for each T_i, it is quite easy to "mix" these orderings to get the optimal one for T itself. This motivates the following study of sequences and the definition of the shuffle.

4.3.2. *Sequences, lexicographic order and shuffles.* We work with sequence of real numbers. For $A \subseteq \{1, 2, \dots, n\}$, we denote by $\boldsymbol{c}|_A$ the subsequence $(c_i)_{i \in A}$. Given two sequences $\boldsymbol{a} = a_1 \dots a_m$ and $\boldsymbol{b} = b_1 \dots b_n$, we denote by $\boldsymbol{a} \bullet \boldsymbol{b}$ the sequence $a_1 \dots a_m b_1 \dots b_n$. The empty sequence is denoted $\boldsymbol{\epsilon}$.

We endow the set of sequences with the lexicographic order $\preceq$.

We have the following proposition, which seems to be interesting for its own sake.

Proposition 4.5. *Given s sequences $\boldsymbol{a}_i$, $i = 1, \dots, s$, of real numbers, it is possible to compute $\mathrm{ms}(\boldsymbol{a}_1, \dots, \boldsymbol{a}_s)$ in $O(n^2 \log s)$, where n is the sum of the lengths of the $\boldsymbol{a}_i$.*

Before proving it, we will study some properties of ms. We will use the properties stated in the following lemma without explicit mention. Indeed, there are all more or less obvious.

Lemma 4.6. *We have the following properties.*

(1)
$$\mathrm{ms}(\boldsymbol{a}, \boldsymbol{b}) = \mathrm{ms}(\boldsymbol{b}, \boldsymbol{a}).$$

(2) *For a fixed sequence $\boldsymbol{a}$, the map $\boldsymbol{x} \mapsto \mathrm{ms}(\boldsymbol{a}, \boldsymbol{x})$ is an increasing map.*

(3) $\mathrm{ms}(\boldsymbol{a}, \boldsymbol{b}) \bullet \mathrm{ms}(\boldsymbol{c}, \boldsymbol{d}) \preceq \mathrm{ms}(\boldsymbol{a} \bullet \boldsymbol{c}, \boldsymbol{b} \bullet \boldsymbol{d})$.
(4) *if* $\boldsymbol{a} \preceq \boldsymbol{b}$ *then for any sequence* $\boldsymbol{c}$ *we have*

$$\boldsymbol{c} \bullet \boldsymbol{a} \preceq \boldsymbol{c} \bullet \boldsymbol{b}.$$

(5)

$$\mathrm{ms}(\boldsymbol{a}, \mathrm{ms}(\boldsymbol{b}, \boldsymbol{c})) = \mathrm{ms}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}).$$

Proof. Except maybe Points (3) and (5), the other points are straightforward.

Point (3): it is a consequence of the following fact: a shuffle of $\boldsymbol{a}$ and $\boldsymbol{b}$ followed by a shuffle of $\boldsymbol{c}$ and $\boldsymbol{d}$ is a shuffle of $\boldsymbol{a} \bullet \boldsymbol{c}$ and $\boldsymbol{b} \bullet \boldsymbol{d}$.

Point (5): We obviously have

$$\mathrm{ms}(\boldsymbol{a}, \mathrm{ms}(\boldsymbol{b}, \boldsymbol{c})) \preceq \mathrm{ms}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}).$$

Hence, we want to prove the reverse inequality. Let $\boldsymbol{d} = \mathrm{ms}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ and $\boldsymbol{s} = \mathrm{ms}(\boldsymbol{b}, \boldsymbol{c})$. Define β and γ to be respectively the prints of $\boldsymbol{b}$ and $\boldsymbol{c}$ in $\boldsymbol{d}$. Denote B the image of β and C the image of γ. Then $\boldsymbol{d}|_{B \cup C} \preceq \boldsymbol{s}$. Since, $\boldsymbol{d} = \mathrm{ms}(\boldsymbol{a}, \boldsymbol{d}|_{B \cup C})$, we conclude with the help of Point (2). □

Lemma 4.6 shows that the set of sequences endowed with ms as a product is a monoid.

Lemma 4.7. *Let* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *be two sequences such that* $\boldsymbol{a} \preceq \boldsymbol{b}$*. Let* $\boldsymbol{c}$ *be another sequence. Then*

$$\mathrm{ms}(\boldsymbol{a}, \boldsymbol{c} \bullet \boldsymbol{b}) \preceq \mathrm{ms}(\boldsymbol{c} \bullet \boldsymbol{a}, \boldsymbol{b}).$$

Proof. Write $\boldsymbol{a} = a_1 \dots a_m$, $\boldsymbol{b} = b_1 \dots b_n$ and $\boldsymbol{c} = c_1 \dots c_r$.

The proof works by induction on $m + n$. If $m + n = 1$, the statement is straightforward.

Let $\boldsymbol{a}' = a_1 \dots a_k$ and $\boldsymbol{a}'' = a_{k+1} \dots a_m$ be two sequences such that

$$\mathrm{ms}(\boldsymbol{a}, \boldsymbol{c} \bullet \boldsymbol{b}) = \mathrm{ms}(\boldsymbol{a}', \boldsymbol{c}) \bullet \mathrm{ms}(\boldsymbol{a}'', \boldsymbol{b}).$$

If $\boldsymbol{a}'' = \boldsymbol{\epsilon}$, we define k to be m.

Now, define $\boldsymbol{b}' := b_1 \dots b_{\min(k,n)}$ and $\boldsymbol{b}''$ such that $\boldsymbol{b} = \boldsymbol{b}' \bullet \boldsymbol{b}''$. If k or n equals 0, we define $\boldsymbol{b}' := \boldsymbol{\epsilon}$. If $\boldsymbol{b}' \succ \boldsymbol{a}'$, then we have

$$\mathrm{ms}(\boldsymbol{a}', \boldsymbol{c}) \bullet \mathrm{ms}(\boldsymbol{a}'', \boldsymbol{b}) \preceq \mathrm{ms}(\boldsymbol{b}', \boldsymbol{c}) \preceq \mathrm{ms}(\boldsymbol{c}, \boldsymbol{b}') \bullet \mathrm{ms}(\boldsymbol{a}, \boldsymbol{b}'') \preceq \mathrm{ms}(\boldsymbol{c} \bullet \boldsymbol{a}, \boldsymbol{b}).$$

Hence, we can assume that $\boldsymbol{b}' = \boldsymbol{a}'$. Thus, $\boldsymbol{b}'' \succeq \boldsymbol{a}''$. By induction

$$\mathrm{ms}(\boldsymbol{b}' \bullet \boldsymbol{a}'', \boldsymbol{b}'') \succeq \mathrm{ms}(\boldsymbol{a}'', \boldsymbol{b}' \bullet \boldsymbol{b}'').$$

Using the equality $\boldsymbol{b}' = \boldsymbol{a}'$ and Point (3) of Lemma 4.6, we get finally

$$\begin{array}{rcl} \mathrm{ms}(\boldsymbol{a}, \boldsymbol{c} \bullet \boldsymbol{b}) & = & \mathrm{ms}(\boldsymbol{a}', \boldsymbol{c}) \bullet \mathrm{ms}(\boldsymbol{a}'', \boldsymbol{b}' \bullet \boldsymbol{b}'') \\ & \preceq & \mathrm{ms}(\boldsymbol{b}', \boldsymbol{c}) \bullet \mathrm{ms}(\boldsymbol{b}' \bullet \boldsymbol{a}'', \boldsymbol{b}'') \\ & = & \mathrm{ms}(\boldsymbol{b}', \boldsymbol{c}) \bullet \mathrm{ms}(\boldsymbol{a}' \bullet \boldsymbol{a}'', \boldsymbol{b}'') \\ & = & \mathrm{ms}(\boldsymbol{c}, \boldsymbol{b}') \bullet \mathrm{ms}(\boldsymbol{a}, \boldsymbol{b}'') \\ & \preceq & \mathrm{ms}(\boldsymbol{c} \bullet \boldsymbol{a}, \boldsymbol{b}). \end{array}$$

□

Proof of Proposition 4.5. Lemma 4.7 implies that if $\boldsymbol{d} = d_1 \dots d_m$ and $\boldsymbol{e} = e_1 \dots e_{m'}$ are two sequences such that $\boldsymbol{e} \succeq \boldsymbol{d}$, then

$$\mathrm{ms}(\boldsymbol{e}, \boldsymbol{d}) = e_1 \bullet \mathrm{ms}(\boldsymbol{d}, e_2 \dots e_{m'}). \tag{1}$$

Indeed, if $e_1 > d_1$, it is obvious, and if not, Lemma 4.7 with $\boldsymbol{c} := e_1$, $\boldsymbol{a} := d_2 \dots d_m$ and $\boldsymbol{b} := e_2 \dots e_m$ implies it.

Assume now that $\boldsymbol{a}_1 \preceq \dots \preceq \boldsymbol{a}_s$. Note than necessarily $\mathrm{ms}(\boldsymbol{a}_{s-1}, \boldsymbol{a}_s) \succeq \boldsymbol{a}_i$ for all $i \notin \{s-1, s\}$. Combining Point (5) of Lemma 4.6 and the fact summarized by Equation (1), we get by an induction on s that

$$\mathrm{ms}(\boldsymbol{a}_1, \dots, \boldsymbol{a}_s) = a_{s1} \bullet \mathrm{ms}(\boldsymbol{a}_1, \dots, \boldsymbol{a}_{s-1}, a_{s2} \dots a_{sn_s}).$$

This leads directly to an algorithm whose time complexity is $O(n^2 \log s)$:

- sort all sequences: $O(sn \log s)$,
- let $\boldsymbol{m} = \boldsymbol{\epsilon}$,
- repeat n times the following operations: append to $\boldsymbol{m}$ the first term of the maximal sequence $\boldsymbol{a}$ in the list, delete this first term from $\boldsymbol{a}$ and put the new $\boldsymbol{a}$ at the right place in the list (this last operation takes $O(n \log s)$).
- return $\boldsymbol{m}$.

□

4.3.3. *Proof of Theorem 1.2.*

Proof. According to the previous discussion, it remains to prove that the optimal ordering starting at r can be obtained by doing ms on the optimal orderings of the subtrees T_i of T. It is straightforward.

Let r_i be the root of T_i for each i. The optimal ordering for T is a shuffle of the orderings of the T_i. According to the preliminary discussion in Subsection 4.3.1, each of these orderings starts with r_i.

For each i, consider now an ordering of the vertices of T_i starting with r_i, and denote by $\boldsymbol{m}_i$ the corresponding sequence of weights. The best shuffle we can get as an ordering of the vertices of T is the one obtained by doing the maximum shuffle product

$$\mathrm{ms}(w(r_1) \bullet \boldsymbol{m}_1, \dots, w(r_s) \bullet \boldsymbol{m}_s).$$

For each T_i, the best is to choose the ordering that maximizes $\boldsymbol{m}_i$ with respect to the lexicographic order (ms is increasing).

The whole complexity is computed as follows: for a fixed root r, using the inequalities

$$n^2 \log n \geq \log n \sum_i n_i^2 \geq \sum_i n_i^2 \log n_i \quad \text{when } \textstyle\sum_i n_i = n-1 \text{ and } n_i \geq 0 \text{ for all } i,$$

we get by induction $O(n^2 \log n \log d)$ where d is the maximum degree of the tree. Doing the computation for each possible root provides the solution. □

5. Pareto-optimal ordering

Finding a Pareto-optimal ordering seems to be a difficult task. It is even unclear whether it is possible to check in polynomial time that a given ordering is Pareto-optimal.

In some cases, the following criterion will be used for $q = 1$.

Lemma 5.1. *Let $f : \mathbb{R}^n \to \mathbb{R}^q$ be a strictly increasing map (ie $\boldsymbol{x} \neq \boldsymbol{x}'$ are such that $x_i \leq x'_i$ for all i, then $f(\boldsymbol{x}) \neq f(\boldsymbol{x}')$ and $f(\boldsymbol{x})_k \leq f(\boldsymbol{x}')_k$ for all $k = 1, \ldots, q$). Let $\mathcal{S}$ be a subset of $\mathbb{R}^n$. If $f(\boldsymbol{x}^*)$ is Pareto-optimal on $f(\mathcal{S})$, then $\boldsymbol{x}^*$ is Pareto-optimal on $\mathcal{S}$.*

Proof. Let $\boldsymbol{y} \in \mathbb{R}^n$ distinct from $\boldsymbol{x}^*$. We have $f(\boldsymbol{y})_k \leq f(\boldsymbol{x}^*)_k$ for a certain k. Since f is strictly increasing, we cannot have $x_i^* \leq y_i$ for all i. Therefore, $\boldsymbol{x}^*$ is Pareto-optimal. □

In the following proposition, the search for a Pareto-optimal ordering is related to the testing cost problem.

Proposition 5.2. *Any optimal solution of* Testing Cost *with $c(v) = 1$ for all $v \in V$ is a Pareto-optimal solution of* Pareto-optimal ordering.

Proof. Given an ordering $v_1, \ldots, v_n$, we have the following equality

$$\sum_{i=1}^{n} \sum_{e \in E \setminus E[v_1, \ldots, v_{i-1}]} p(e) = (|E| + 1)p(E) - \sum_{i=1}^{n} h_i.$$

Using the map $f(x_1, \ldots, x_n) = \sum_{i=1}^n x_i$ in Lemma 5.1, we get the required conclusion. □

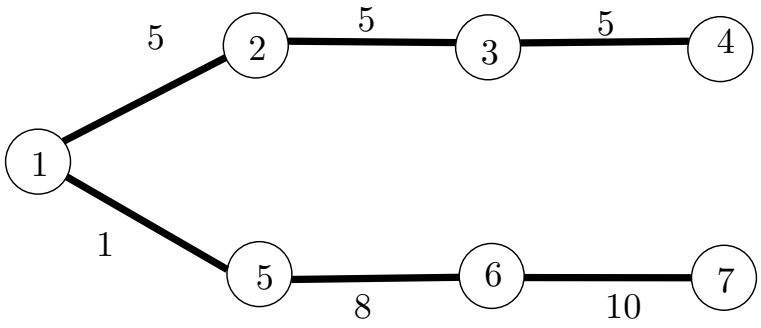

Figure 2. A rooted tree

5.1. **Pareto-optimality in trees.** Proposition 1.1 claims that when H is a tree, finding a Pareto-optimal ordering can be done in polynomial time. We prove now this proposition. The proof uses Lemma 5.1 and builds a special function f to transform the search of Pareto-optimality into a classical optimization problem.

Proof of Proposition 1.1. Denote by $T = (V, E)$ the tree and assume that there is a weight function $w : E \to \mathbb{R}_+$.

Fix a root r. Define $w' : V \to \mathbb{R}_+$ to be such that $w'(v) := w(uv)$ where u is the neighbor of v on the path linking r and v and $w'(r) = 0$. Consider the problem – which we call P – of finding an ordering $v_1, \ldots, v_n$ of the vertices of T, starting at a fixed vertex $v_1 = r$, such that

(1) $v_{i+1} \in N(v_1, \ldots, v_i)$ for all $i = 1, \ldots, n$
(2) the quantity $\sum_{i=1}^n (i-1)w'(v_i)$ is minimal.

Note that minimizing $\sum_{i=1}^n (i-1)w'(v_i)$ is equivalent to maximizing $\sum_{i=1}^n (n-i+1)w'(v_i)$, which is equal to $\sum_{i=1}^n h_i$ with $h_i = \sum_{\ell=1}^i w'(v_\ell)$. The sequence (h_i) is the one that has to be Pareto-optimal. Since choosing $v_{i+1} \notin N(v_1, \ldots, v_i)$ for a certain i provides $h_{i+1} = h_i$,

i	1	2	3	4	5	6	7
v_i	1	2	3	4	5	6	7
h_i	0	5	10	15	16	24	34

TABLE 4. Evaluation vector of an ordering for the graph of Figure 2

i	1	2	3	4	5	6	7
v_i	1	5	6	7	2	3	4
h_i	0	1	9	19	24	29	34

TABLE 5. Evaluation vector of an ordering for the graph of Figure 2

we can restrict our search for a Pareto-optimal sequence to the sequences satisfying $v_{i+1} \in N(v_1, \ldots, v_i)$ for all $i = 1, \ldots, n$.

If the ordering $v_1, v_2, \ldots, v_n$ is a solution of problem P, Lemma 5.1 with $f(x_2, \ldots, x_n) = \sum_{i=2}^{n}(n-i+1)x_i$ ensures that the ordering is Pareto-optimal.

Horn [6] proved that P can be solved in polynomial time (see [1] for a fast algorithm, in $O(n \log n)$).

Doing it for each possible root $r \in V$ eventually leads to the Pareto-optimal ordering. □

Note that the ordering obtained with this method is not necessarily a maximal one for the lexicographic order. Indeed, consider the tree of Figure 2 and the orderings from Tables 4 and 5. The ordering in Table 4 is obtained by applying the greedy algorithm presented earlier. Moreover, since the sufficient condition of Proposition 4.3 is satisfied, this ordering is maximal for the lexicographic order. The value of the objective function $\sum_{i=1}^{n} h_i$ for this ordering is 104. For the ordering presented in Table 5 the value is 116. The ordering obtained by the method described in the proof is therefore not necessarily maximal for the lexicographic order.

6. A combinatorial analog of the fundamental theorem of arithmetic

We prove now the combinatorial analog of the fundamental theorem of arithmetic (Theorem 1.3). As in Subsection 4.3.2, we forget the initial motivation and deal only with sequences. Recall that a sequence $\boldsymbol{p} = p_1 \ldots p_n$ is prime if $p_i p_{i+1} \ldots p_n \succeq \boldsymbol{p}$ for each $i \geq 1$, where $\preceq$ is the lexicographic order on the sequences.

We have an easy lemma

Lemma 6.1. *Let $\boldsymbol{p}$ and $\boldsymbol{p}'$ be two prime sequences such that $\boldsymbol{p} \preceq \boldsymbol{p}'$. Then* $\mathrm{ms}(\boldsymbol{p}, \boldsymbol{p}') = \boldsymbol{p}' \bullet \boldsymbol{p}$.

Proof. It is a consequence of the algorithm described in the proof of Proposition 4.5. □

Proof of Theorem 1.3. Let us write $\boldsymbol{a} = a_1 \ldots a_n$. Define $i_1 := 1$ and $\boldsymbol{a}_1 := \boldsymbol{a}$. Then define i_2 to be the first index $i > i_1$ such that $a_i a_{i+1} \ldots a_n \prec \boldsymbol{a}_1$, let $\boldsymbol{a}_2 := a_{i_2} a_{i_2+1} \ldots a_n$. Define i_3 to be the first index $i > i_2$ such that $a_i a_{i+1} \ldots a_n \prec \boldsymbol{a}_2$ and let $\boldsymbol{a}_3 := a_{i_3} a_{i_3+1} \ldots a_n$. And so on. Let r be the index of the last $\boldsymbol{a}_j$ defined (and by convention $i_{r+1} := n+1$).

Defining $\boldsymbol{p}_j$ to be $a_{i_j} a_{i_j+1} \ldots a_{i_{j+1}-1}$, we get

$$\boldsymbol{a} = \boldsymbol{p}_1 \bullet \boldsymbol{p}_2 \bullet \ldots \bullet \boldsymbol{p}_r.$$

Now, we have the following claim.

CLAIM. *The $\boldsymbol{p}_j$ are all prime and we have*

$$\boldsymbol{p}_1 \succeq \boldsymbol{p}_2 \succeq \ldots \succeq \boldsymbol{p}_r.$$

Let us first prove that $\boldsymbol{p}_j$ is prime for any j. Actually, we will prove it for $\boldsymbol{p}_1$, the proof being exactly the same for the other indices.

Let us suppose that $\boldsymbol{p}_1$ is not prime. Then there is an $i \in \{2, \ldots, i_2 - 1\}$ such that

$$a_i a_{i+1} \ldots a_{i_2-1} \prec a_1 a_2 \ldots a_{i_2-1}. \tag{2}$$

By definition of i_2, we have also

$$a_i a_{i+1} \ldots a_{i_2-1} a_{i_2} a_{i_2+1} \ldots a_n \succeq a_1 a_2 \ldots a_{i_2-1} a_{i_2} a_{i_2+1} \ldots a_n. \tag{3}$$

Equation (2) implies that $a_i \leq a_1$. Equation (3) implies that $a_i \geq a_1$. Together, they imply that $a_i = a_1$. We can therefore delete a_1 and a_i from their first places. Again, Equation (2) implies that $a_{i+1} \leq a_2$ and Equation (3) implies that $a_{i+1} \geq a_2$, which in turn leads to $a_{i+1} = a_2$. Going on in the same way, we get eventually:

$$a_i = a_1,\ a_{i+1} = a_2, \ldots,\ a_{i_2-1} = a_{i_2-i}.$$

Using these equalities in Equation (3), we get that

$$a_{i_2} a_{i_2+1} \ldots a_n \succeq a_{i_2-i+1} a_{i_2-i+2} \ldots a_n.$$

But, by definition of i_2, we have $\boldsymbol{a} \succ a_{i_2} a_{i_2+1} \ldots a_n$ and $\boldsymbol{a} \preceq a_{i_2-i+1} a_{i_2-i+2} \ldots a_n$. A contradiction.

Let us now prove the chain of inequalities. Again we will only prove that $\boldsymbol{p}_1 \succeq \boldsymbol{p}_2$, since the proof of the other inequalities is strictly the same.

Denote $\boldsymbol{p}_1 = p_{11} p_{12} \ldots p_{1n_1}$ and $\boldsymbol{p}_2 = p_{21} p_{22} \ldots p_{2n_2}$.

Assume for a contradiction that $\boldsymbol{p}_1 \prec \boldsymbol{p}_2$. By construction, we have

$$\boldsymbol{p}_1 \bullet \boldsymbol{p}_2 \bullet \ldots \bullet \boldsymbol{p}_r \succ \boldsymbol{p}_2 \bullet \ldots \bullet \boldsymbol{p}_r.$$

Thus, if $n_2 \leq n_1$, we would have $p_{11} = p_{21}$, $p_{12} = p_{22}$, ..., $p_{1n_2} = p_{2n_2}$, which contradicts $\boldsymbol{p}_1 \prec \boldsymbol{p}_2$. Thus $n_2 > n_1$. But then we have $p_{11} = p_{21}$, $p_{12} = p_{22}$, ..., $p_{1n_1} = p_{2n_1}$ and

$$\boldsymbol{p}_2 \bullet \ldots \bullet \boldsymbol{p}_r \succ p_{2n_1+1} p_{2n_1+2} \ldots p_{2n_2} \bullet \boldsymbol{p}_3 \bullet \ldots \bullet \boldsymbol{p}_r,$$

which is in contradiction with the definition of i_3.

The claim is proved.

Thus, we have according to Lemma 6.1, $\boldsymbol{a} = \mathrm{ms}(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_r)$. The existence claimed by the theorem is proved. It remains to prove the uniqueness of the decomposition.

Assume for a contradiction that there are also r' prime sequences

$$\boldsymbol{p}'_1 \succeq \boldsymbol{p}'_2 \succeq \ldots \succeq \boldsymbol{p}'_{r'}$$

such that $\boldsymbol{a} = \boldsymbol{p}'_1 \bullet \boldsymbol{p}'_2 \bullet \ldots \bullet \boldsymbol{p}'_{r'}$. Without loss of generality, we can assume that the length of $\boldsymbol{p}'_1$ is different from the length of $\boldsymbol{p}_1$.

If the length of $\boldsymbol{p}'_1$ is strictly smaller than the length of $\boldsymbol{p}_1$, then let k be such that $\boldsymbol{p}'_k$ contains a_{i_2-1} (the last term of $\boldsymbol{p}_1$). Since $\boldsymbol{p}_1$ is prime, we have $\boldsymbol{p}'_k \succeq \boldsymbol{p}_1$, which contradicts the fact that $\boldsymbol{p}'_1 \prec \boldsymbol{p}_1$.

If the length of $\boldsymbol{p}'_1$ is strictly larger than the length of $\boldsymbol{p}_1$, we can write

$$\boldsymbol{p}'_1 = \boldsymbol{p}_1 \bullet a_{i_2} a_{i_2+1} \ldots a_i,$$

for some index $i \geq i_2$. Since $\boldsymbol{p}'_1$ is prime, we have

$$a_{i_2}a_{i_2+1}\dots a_i \succeq \boldsymbol{p}_1 \bullet a_{i_2}a_{i_2+1}\dots a_i,$$

and since the length of the sequence on the left side is smaller than the length of the right's one, we have

$$a_{i_2}a_{i_2+1}\dots a_i a_{i+1}\dots a_n \succeq \boldsymbol{p}_1 \bullet a_{i_2}a_{i_2+1}\dots a_i a_{i+1}\dots a_n,$$

which contradicts the definition of i_2. □

Remark. We can extend the definition of prime sequences to infinite sequences. However, if we define an infinite sequence $\boldsymbol{p}$ to be prime if all its suffixes $\boldsymbol{q}$ are such that $\boldsymbol{q} \succeq \boldsymbol{p}$, we loose the uniqueness of the decomposition: for instance, the sequence $1, 2, 1, 2, 1, 2, \dots$ is prime according to this definition, but can also be written as $(1, 2)^\infty$, where $1, 2$ is a prime sequence.

The right way to extend the definition consists in defining a sequence $\boldsymbol{p} = p_1 p_2 \dots$ to be prime if all its strict suffixes $\boldsymbol{q}$ are such that $\boldsymbol{q} \succ \boldsymbol{p}$, a strict suffix being obtained by deleting at least p_1. Note that for finite sequences, it does not change anything, while it settles the problem for infinite sequences: Theorem 1.3 remains true for infinite sequences $\boldsymbol{a}$. The proof is almost the same and shows moreover that only two situations may arise: either s (the number of distinct prime factors) is infinite and then all prime sequences in the decomposition are finite, or s is finite and then only one prime sequence in the decomposition is infinite and its exponent is equal to 1.

7. Conclusion and open questions

Starting from a practical motivation, we have defined several challenging problems, which mix several features: scheduling, special orders, graphs... Polynomial algorithms have been proposed in some special cases. One of these algorithms has motivated the notion of a new product – the *maximum shuffle product* – defined on the set of sequences of real numbers. It appeared that this product has a nice property, namely an analog of the fundamental theorem of arithmetic.

Open questions have already been outlined in the paper. But other questions remain open. For instance, there is the question of larger special cases for which the problems remain polynomial. Polynomial algorithms have been proposed when the hypergraph is a tree graph, but there may exist larger classes of hypergraphs for which these problems are polynomial.

It would also be interesting to find a suitable framework for the maximum shuffle product ms, in which we would have more than a monoid. For instance, if ms is the product, what is the addition ?

References

[1] D. Adolphson and T. C. Hu. Optimal linear ordering. *SIAM Journal on Applied Mathematics*, 25(3):403–423, 1973.

[2] Y. Crama. Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Operational Research*, 99(1):136 – 153, 1997.

[3] Y. Crama and J. van de Klundert. Worst-case performance of approximation algorithms for tool management problems. *Naval Research Logistics*, 46(5):445–462, 1999.

[4] G. Gallo, P. Hammer, and B. Simeone. Quadratic knapsack problems. *Mathematical Programming Study*, 12:132–149, 1980.

[5] P. Hammer and T. Bonates. Logical analysis of data – an overview: From combinatorial optimization to medical applications. *Annals of Operations Research*, 148:203–225, 2006. 10.1007/s10479-006-0075-y.
[6] W. A. Horn. Single-machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM Journal on Applied Mathematics*, 23(2):189–202, 1972.
[7] S. G. Kolliopoulos and G. Steiner. Partially ordered knapsack and applications to scheduling. *Discrete Applied Mathematics*, 155(8):889 – 897, 2007.
[8] E. L. Lawler. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, 2:75–90, 1978.
[9] P. Lemaire, N. Brauner, P. Hammer, C. Trivin, J.-C. Souberbielle, and R. Brauner. Improved screening for growth hormone deficiency using logical analysis data. *Medical Science Monitor*, 15(1):MT5–10, 02 2009.
[10] D.J. Rader and G. J. Woeginger. The quadratic 0–1 knapsack problem with series-parallel support. *Operations Research Letters*, 30:159–166, 2002.
[11] V. A. Testov. An analog of the fundamental theorem of arithmetic in ordered groupoids. *Mathematical notes*, 62:762–766, 1997.
[12] G. J. Woeginger. On the approximability of average completion time scheduling under precedence constraints. *Discrete Applied Mathematics*, 131(1):237 – 252, 2003.

(GSCOP) G-SCOP, INPGrenoble, UJF, CNRS, 46 avenue Félix Viallet 38031 Grenoble cedex, France
E-mail address: nadia.brauner@grenoble-inp.fr

(IF) Institut Fourier 100, rue des Maths, BP 74, 38402 St Martin d'Hères cedex, France
E-mail address: sylvain.gravier@ujf-grenoble.fr

(ARTELYS) Artelys SA, 12 rue du Quatre Septembre, 75002 Paris, France
E-mail address: louisphilippe.kronek@gmail.com

(ENPC) Université Paris Est, CERMICS, 6-8 avenue Blaise Pascal, 77455 Marne-la-Vallée cedex, France
E-mail address: frederic.meunier@cermics.enpc.fr