Interpolation by lattice polynomial functions: a polynomial time algorithm

Quentin Brabant, Miguel Couceiro, José Figueira

To cite this version:

Quentin Brabant, Miguel Couceiro, José Figueira. Interpolation by lattice polynomial functions: a polynomial time algorithm. Fuzzy Sets and Systems, Elsevier, 2019, 368, pp.101-118. 10.1016/j.fss.2018.12.009 . hal-01958903v2

HAL Id: hal-01958903

https://hal.archives-ouvertes.fr/hal-01958903v2

Submitted on 26 Apr 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Interpolation by lattice polynomial functions: a polynomial time algorithm

Quentin Brabant♣, Miguel Couceiro♣, and José Rui Figueira♠

♣Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
name.surname@inria.fr
♠CEG-IST, Instituto Superior Técnico, Universidade de Lisboa, Portugal
surname@tecnico.ulisboa.pt

Abstract

This paper deals with the problem of interpolating partial functions over finite distributive lattices by lattice polynomial functions. More precisely, this problem can be formulated as follows: Given a finite distributive lattice L and a partial function f from $D \subseteq L^n$ to L, find all the lattice polynomial functions that interpolate f on D. If the set of lattice polynomials interpolating a function f is not empty, then it has a unique upper bound and a unique lower bound. This paper presents a new description of these bounds and proposes an algorithm for computing them that runs in polynomial time, thus improving existing methods. Furthermore, we present an empirical study on randomly generated datasets that illustrates our theoretical results.

Keywords: Distributive lattices, lattice polynomials, interpolation, multiple criteria evaluation, non-additive measures

The first and second named authors were partially supported by the Labex ANR-11-LABX-0040-CIMI (Centre International de Mathématiques et d'Informa-tique) in the setting of the program ANR-11-IDEX-0002-02, subproject ISIPA (Interpolation, Sugeno Integral, Proportional Analogy). The third named author acknowledges the support from the FCT grant SFRH/BSAB/139892/2018 under POCH program.

1 Introduction

Lattice polynomial functions (LPFs) constitute a noteworthy class of functions that enable meaningful aggregation of qualitative data [17]. Informally, they are functions on lattices that can be expressed by formulas involving only variables, constants, and the lattice operators of supremum $\vee$ and infimum $\wedge$. This paper focuses on *finite distributive lattices* and considers the problem of interpolating partial functions by LPFs. More precisely, we consider the following problem.

Interpolation Problem. Let L be a finite distributive lattice, and n a positive integer. Given an arbitrary set $D \subseteq L^n$ and a partial function $f : D \to L$, decide whether f can be interpolated by an LPF $p : L^n \to L$ (that is, $p(\mathbf{x}) = f(\mathbf{x})$, for all $\mathbf{x} \in D$) and, if this is the case, then describe all such interpolating LPFs.

Throughout this paper, we denote the set of all LPFs that interpolate a function f by $\mathcal{IP}(f)$. If $\mathcal{IP}(f)$ is nonempty, then we say that f is a partial LPF. It was shown in [27] that $\mathcal{IP}(f)$ has a unique upper bound and a unique lower bound when L is totally ordered. The authors also provided a method for constructing these bounds. In fact [7], the set $\mathcal{IP}(f)$ has uniques upper and lower bounds even in the case where L is a distributive lattice. However, to the best of our knowledge, no polynomial time method existed for computing these bounds.

In this paper we present a polynomial time algorithm that computes the bounds of $\mathcal{IP}(f)$ and checks whether they specify a nonempty set. As a by-product, we provide a characterization of partial LFPs.

This paper is organized as follows. In Section 2 we recall the basic background on distributive lattices and LPFs, and in Section 3 we explain how the interpolation problem for LPFs is related to Multiple Criteria Decision Aid. More precisely, we show that LPFs can help handling problems involving unknown values by enabling the use of distributive lattices as intermediary scales. In Section 4 we extend previous results [11, 26, 27] by showing that the upper and lower bounds of $\mathcal{IP}(f)$ can be described by a polynomial number of constraints. In Section 5 we make use of the latter result to characterize the class of partial LPFs through the notion of semi-congruence. In Section 6 we provide the algorithms for computing the set of constraints describing $\mathcal{IP}(f)$ and for checking whether $\mathcal{IP}(f)$ is nonempty. We also present the worst case complexity of these algorithms, which is attested empirically in Section 7 through an application to randomly generated datasets of various dimensions. Some conclusions and directions for future research are then discussed in the last section.

2 Preliminaries

In this section we recall basic concepts and preliminary results that will be needed throughout the paper. For further background see, e.g., [12, 18].

2.1 Basic background on distributive lattices

For any positive integer n, let $[n] = \{1, \ldots, n\}$. For any finite set X, we will denote by 2^X the set of all subsets of X. Note that 2^X is ordered by inclusion and constitutes a partially ordered set (or a poset for short).

A *lattice* is a poset $(L, \leq)$ in which any two elements $a, b \in L$ have exactly one least upper bound, called the *supremum* of a and b and denoted by $a \vee b$, and exactly one greatest lower bound, called the *infimum* of a and b and denoted by $a \wedge b$. Hence, a lattice $(L, \leq)$ can also be regarded as the algebraic structure $(L, \vee, \wedge)$. When there is no danger of ambiguity, we will denote a lattice simply by its universe L. Also, if L is a finite lattice, then it is necessarily *bounded*, that is, it has a greatest element and a least element that are denoted by $\top_L$ and $\bot_L$, respectively. For example, for any set X, the poset 2^X is a lattice where $\bot_L = \emptyset$, $\top_L = X$, and the operations $\vee$ and $\wedge$ correspond respectively to the set union and the set intersection.

A lattice $(L_1, \vee_1, \wedge_1)$ is said to be a *sublattice* of $(L_2, \vee_2, \wedge_2)$ if $L_1 \subseteq L_2$ and for any $a, b \in L_1$,

$$a \vee_1 b = a \vee_2 b \quad \text{and} \quad a \wedge_1 b = a \wedge_2 b.$$

In this paper, we will only consider finite distributive lattices. A lattice is said to be *distributive* if the following condition holds: for every $a, b, c \in L$,

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \text{ or, equivalently, } a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c).$$

Equivalently, distributive lattices can be characterized as lattices that do not have any of the two lattices shown in Figure 1 as a sublattice [1].

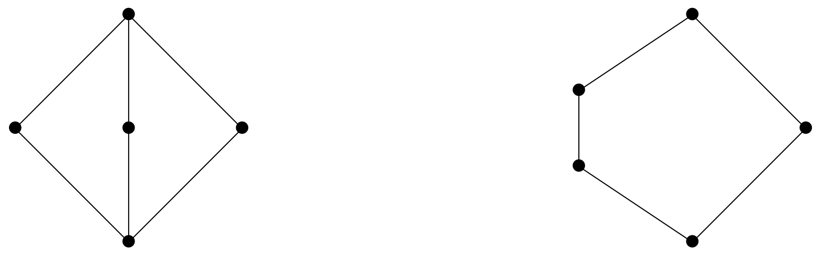


Figure 1: The two forbidden substructures in a distributive lattice.

Two partially ordered sets $(X_1, \leq_1)$ and $(X_2, \leq_2)$ are said to be *isomorphic* if there is an order isomorphism between $(X_1, \leq_1)$ and $(X_2, \leq_2)$, i.e., if there is a bijection $h : X_1 \to X_2$ such that, for every $a, b \in X_1$, $a \leq_1 b$ if and only if $h(a) \leq_2 h(b)$. In the case of lattices, this relation becomes more stringent. Two lattices $(L_1, \vee_1, \wedge_1)$ and $(L_2, \vee_2, \wedge_2)$ are said to be *isomorphic* (as lattices) if there exists a bijection $h : L_1 \to L_2$ such that, for all $a, b \in L_1$,

$$h(a \vee_1 b) = h(a) \vee_2 h(b) \quad \text{and} \quad h(a \wedge_1 b) = h(a) \wedge_2 h(b).$$

An element $x \in L$ is said to be *join-irreducible* if $x \neq \bot_L$ and if it is not the supremum of two other elements of L. In a lattice, any element is equal to the supremum of all join-irreducible elements lower than or equal to it. The set of join-irreducible elements of L is denoted by $\mathcal{J}(L)$. Note that $\mathcal{J}(L)$ is a poset whose order is inherited from L. Each distributive lattice is characterized by the poset of its join-irreducible elements. Therefore, two distributive lattices L_1 and L_2 are isomorphic if and only if the posets $\mathcal{J}(L_1)$ and $\mathcal{J}(L_2)$ are isomorphic [12].

Moreover, every finite poset X gives rise to a distributive lattice. Let $(X, \leq)$ denote a finite poset and let $x \in X$. The *downset* of x in X is denoted by $\downarrow x$ and defined by $\downarrow x = \{y \in X \mid y \leq x\}$. Consider the set

$$\mathcal{O}(X) = \left\{ \bigcup_{x \in A} \downarrow x \;\middle|\; A \subseteq X \right\}$$

of unions of downsets of X. The set $\mathcal{O}(X)$ ordered by inclusion is a distributive lattice whose poset of join-irreducible elements is

$$\mathcal{J}(\mathcal{O}(X)) = \{\downarrow x \mid x \in X\}.$$

It is easy to see that $\mathcal{J}(\mathcal{O}(X))$ is (order-)isomorphic to X.

Similarly, any finite distributive lattice L is isomorphic to $\mathcal{O}(\mathcal{J}(L))$. This establishes a two way correspondence between distributive lattices and posets that culminated in Birkhoff's representation theorem for finite distributive lattices, from which it follows that every finite distributive lattice L can be thought of as a sublattice of $2^{\mathcal{J}(L)}$.

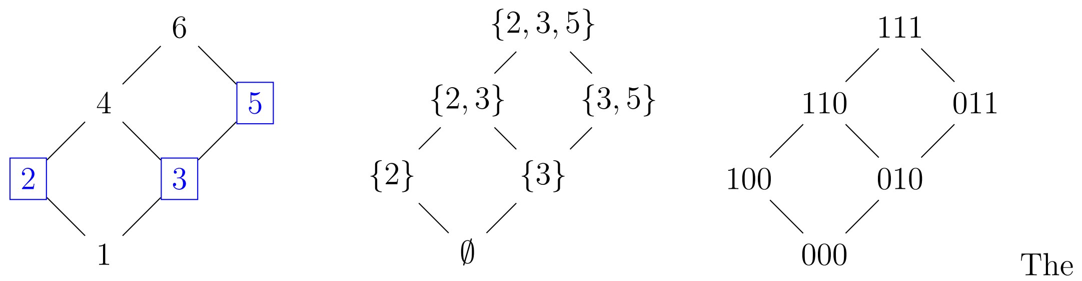
Thus, each element a of L can be seen as a subset of $\mathcal{J}(L)$, namely, as the set of join-irreducible elements that are lower than or equal to a, or as a binary tuple of size k, where $k = |\mathcal{J}(L)|$. Let $\mathcal{J}(L) = \{a_1, \ldots, a_k\}$, and let h denote the bijection from $\mathcal{O}(\mathcal{J}(L))$ to $\{0,1\}^k$ that associates each element of $\mathcal{O}(\mathcal{J}(L))$ to a binary representation. Formally, for each $A \in \mathcal{O}(\mathcal{J}(L))$, let

$$h(A) = (b_1, \ldots, b_k), \quad \text{where } b_i = \begin{cases} 1 \text{ if } a_i \in A, \\ 0 \text{ otherwise.} \end{cases}$$

We consider the component-wise order on $\{0,1\}^k$ defined by

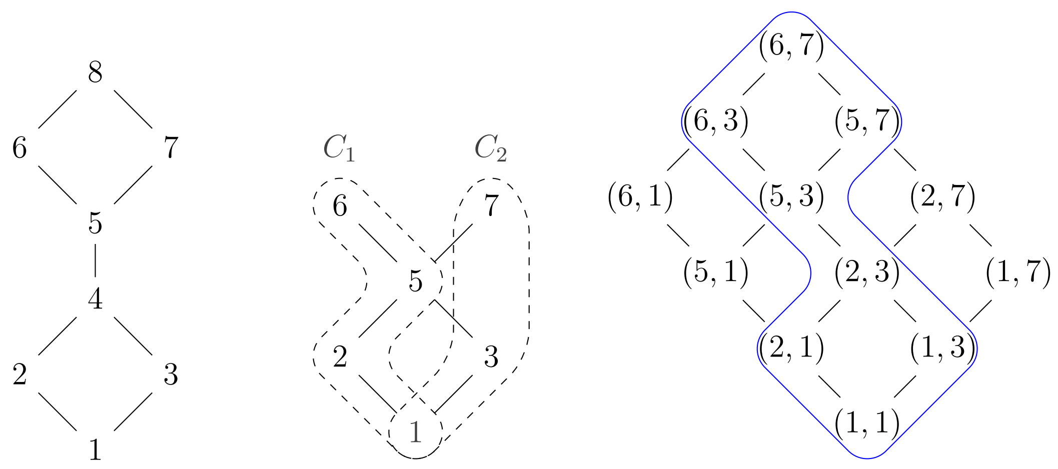
$$(b_1, \ldots, b_k) \leq (b'_1, \ldots, b'_k) \quad \text{if} \quad \forall i \in [k], b_i \leq b'_i,$$

for all $(b_1, \ldots, b_k), (b'_1, \ldots, b'_k) \in \{0,1\}^k$. It is then easy to verify that h is in fact an isomorphism. Figure 2 illustrates the transformations from L to $\mathcal{O}(\mathcal{J}(L))$ and from $\mathcal{O}(\mathcal{J}(L))$ to $\{0,1\}^{|\mathcal{J}(L)|}$. The leftmost figure is the distributive lattice L, and its join-irreducible elements are marked with blue squares. The figure in the middle is the lattice $\mathcal{O}(\mathcal{J}(L))$, where the elements of L are represented as subsets of join-irreducible elements of L. The rightmost figure is the image of $\mathcal{O}(\mathcal{J}(L))$ by h: a sublattice of $\{0,1\}^k$, where elements of L are thought of as binary tuples. For each $A \in \mathcal{O}(\mathcal{J}(L))$, the presence or absence of the join-irreducible elements 2, 3 and 5 of L are respectively indicated by the value of the 1st, 2nd, and 3rd component of $h(A)$.



A distributive lattice L. The lattice $\mathcal{O}(\mathcal{J}(L))$. The sublattice of $\{0,1\}^{|\mathcal{J}(L)|}$ given by h.

Figure 2: Three isomorphic distributive lattices.



A finite distributive lattice L.

The poset $\mathcal{J}(L) \cup \{1\}$ of join-irreducibles elements in L.

The lattice $C_1 \times C_2$. The image of L by the embedding is marked out.

Figure 3: An example of a Dilworth's embedding.

Alternatively, L can always be seen as a sublattice of a product of chains [22]. Let k be a positive integer and $C_1, \ldots, C_k$ be totally ordered sets such that

$$C_1 \cup \cdots \cup C_k = \mathcal{J}(L) \cup \{\bot_L\}$$

and such that

$$\forall i, j \in [k] : \quad i \neq j \quad \Longrightarrow \quad C_i \cap C_j = \{\bot_L\}.$$

For each $x \in L$ and $i \in [k]$, we define

$$c_i^x = \bigvee \{y \in C_i \mid y \leq x\}.$$

The function $\gamma : L \to \prod_{i=1}^{k} C_i$ defined by $\gamma(x) = (c_1^x, \ldots, c_k^x)$ is called a *Dilworth's embedding*. It associates the greatest and least elements of L to the greatest and least elements of $\prod_{i=1}^{k} C_i$, respectively. Note that, since any element of a distributive lattice equals the supremum of the join-irreducible elements lower than or equal to it, we have that

$$x = \bigvee \{y \in \mathcal{J}(L) \mid y \leq x\} = \bigvee_{i \in [k]} \bigvee \{y \in C_i \mid y \leq x\} = c_1^x \vee \cdots \vee c_k^x.$$

Any distributive lattice has at least one Dilworth's embedding. In this paper, any distributive lattice L will be associated to its image by some Dilworth's embedding $\gamma : L \to \prod_{i=1}^{k} C_i$. For any $x \in L$, we will denote by $x|_i$ the i^{th} component of $\gamma(x)$. Note that, for any $a, b \in L$ we have that

$$a \leq b \quad \text{if and only if} \quad \forall i \in [k] : a|_i \leq b|_i. \tag{1}$$

and that

$$\gamma(a \wedge b) = \big(a|_1 \wedge b|_1, \ldots, a|_k \wedge b|_k\big), \quad \gamma(a \vee b) = \big(a|_1 \vee b|_1, \ldots, a|_k \vee b|_k\big). \tag{2}$$

Remark 1. Any finite chain is isomorphic to an interval of natural numbers. Thus, the Dilworth's embedding of L in $\prod_{i=1}^{k} C_i$ can be seen as a way to represent each element of L by a k-tuple of integer values. Also note that in the case where $k = |\mathcal{J}(L)|$, each chain C_i (where $i \in [k]$) contains only two elements: one join-irreducible element and $\perp_L$. In this case, we say that the Dilworth's embedding of L in $\prod_{i=1}^{k} C_i$ is *trivial*. This is essentially the Birkhoff's representation of L as a sublattice of $2^{\mathcal{J}(L)}$ or of $\{0,1\}^{|\mathcal{J}(L)|}$.

2.2 Basic background on LPFs

Let L denote a distributive lattice and let n denote a positive integer. The class of lattice polynomial functions (LPFs) from L^n to L will be denoted by $\mathcal{P}_L^n$, and it can be defined recursively by finitely many applications of the following rules:

1. For any $k \in [n]$, the projection $(x_1, \ldots, x_n) \mapsto x_k$ is an LPF from L^n to L.
2. For any $c \in L$, the constant function $(x_1, \ldots, x_n) \mapsto c$ is an LPF from L^n to L.
3. If $p, q \in \mathcal{P}_L^n$, then $f : L^n \to L$ such that $f(\mathbf{x}) = p(\mathbf{x}) \vee q(\mathbf{x})$ for all $\mathbf{x} \in L^n$ or such that $f(\mathbf{x}) = p(\mathbf{x}) \wedge q(\mathbf{x})$ for all $\mathbf{x} \in L^n$ is an LPF from L^n to L.

It is well-known that any LPF over a distributive lattice can be represented by a disjunctive normal form (DNF) [15]:

$$p(\mathbf{x}) = \bigvee_{I \subseteq [n]} \Big(\alpha_p(I) \wedge \bigwedge_{i \in I} x_i\Big), \tag{3}$$

where α_p is a mapping from $2^{[n]}$ to L. In fact, α_p can be canonically defined by $\alpha_p(I) = p(\mathbf{e}_I)$, for every $I \subseteq [n]$, where $\mathbf{e}_I$ is the element of L^n whose ith component is $\top_L$ if $i \in I$ and $\perp_L$ otherwise. Clearly, α_p thus defined is order-preserving, i.e., $\alpha_p(I) \leq \alpha_p(J)$, whenever $I \subseteq J$.

Remark 2. Note that the canonical DNF representation of an LPF p, i.e., given in terms of α_p as defined above, may include redundant terms. To avoid redundancy, we may consider instead the mapping $\alpha^* \colon 2^{[n]} \to L$ defined by $\alpha^*(I) = \alpha_p(I)$ if $\alpha_p(I) > \bigvee_{J \subsetneq I} \alpha_p(J)$, and $\perp_L$ otherwise. The mapping α^*, introduced in [23, 25], is sometimes called the *ordinal Möbius transform* of α_p. It is not difficult to see that

$$p(\mathbf{x}) = \bigvee_{I \subseteq [n]} \Big(\alpha^*(I) \wedge \bigwedge_{i \in I} x_i\Big).$$

In fact, any function in the interval $[\alpha^*, \alpha_p]$ gives rise to a DNF representation of p, and any DNF representation

$$p(\mathbf{x}) = \bigvee_{I \subseteq [n]} \Big(\alpha(I) \wedge \bigwedge_{i \in I} x_i\Big),$$

of p implies that $\alpha \in [\alpha^*, \alpha_p]$; see [9].

Remark 3. The Sugeno integral can be thought as being a particular type of LPF, namely, those defined with respect to a *capacity* (i.e., a set-function $\mu : 2^{[n]} \to L$ that is order-preserving and that satisfies the boundary conditions $\mu(\emptyset) = \bot_L$ and $\mu([n]) = \top_L$). Even though it was originally introduced over the real unit interval $L = [0, 1]$ [28], the Sugeno integral was later extended to bounded distributive lattices in [24], where the Sugeno integral S_μ w.r.t. a capacity μ is defined by:

$$S_\mu(\mathbf{x}) = \bigvee_{I \subseteq [n]} \Big(\mu(I) \wedge \bigwedge_{i \in I} x_i \Big).$$

It follows from this definition that Sugeno integrals coincide exactly with idempotent LPFs, i.e., LPFs that preserve constant tuples.

We now recall the notion of partial LPF that will play a key role in what follows.

Definition 1. Let $D \subseteq L^n$. A function $f \colon D \to L$ is a partial LPF if there exists $p \in \mathcal{P}_L^n$ such that $p(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in D$.

From a computational point of view, the space required to encode a mapping $\alpha : 2^{[n]} \to L$ is exponential w.r.t. n. As hinted in Remark 2, it is sometimes possible to avoid redundancies. A subset $I \subseteq [n]$ is said to be a *focal set* of α if $\alpha(I) = \alpha^*(I)$, that is, if

$$\alpha(I) > \bigvee_{J \subsetneq I} \alpha(J).$$

Note that, since for any non-focal set $I \subseteq [n]$ we have $\alpha(I) = \bigvee_{J \subset I} \alpha(J)$, α is entirely specified by its values on focal sets.

3 LPFs in Multiple Criteria Decision Aid

Multiple Criteria Decision Aid (MCDA) is a field of operations research concerned with decision problems involving several *criteria*. Problem settings of that field typically consider a set of *alternatives* that are evaluated according to each criterion.

The *range* of a criterion is the set of possible evaluations of an alternative with respect to that criterion. Let X be a set of alternatives, n be the number of considered criteria and $X_1, \ldots, X_n$ the ranges of those criteria. Each alternative $x \in X$ can then be described by a tuple $(x_1, \ldots, x_n) \in X_1 \times \cdots \times X_n$. The values $x_1, \ldots, x_n$ are called the *local evaluations* of x. One problem in MCDA is to find a model that assigns, to each instance, an overall evaluation (that we call the *utility value* of the instance) depending on its local evaluations. In the rest of this section, we discuss an example where this model has to be chosen in a supervised setting.

We consider students that obtained grades in several courses. The possible grades are: "very bad", "bad", "average", "good", "very good". For a matter of simplicity, they are denoted by 1, 2, 3, 4, and 5 respectively. Moreover let $T = \{1, 2, 3, 4, 5\}$ be a totally ordered set where $1 < 2 < 3 < 4 < 5$. Each student also gets an overall grade, which is decided by a committee according to his/her grades in each course. The grades of the students are displayed in Table 1. A few

	physics	literature	economics	algebra	overall
student 1	5	5	?	4	4
student 2	3	3	3	3	3
student 3	5	3	3	5	5
student 4	?	1	2	4	3

Table 1: Toy example: grades attributed to several students.

grades were lost before the overall grade of each student was decided. These missing values are indicated in the table by question marks.

We want to find a model that, for each student, aggregates the grades in each course (i.e., the local evaluations) into an overall grade (i.e. the utility value), according to the examples given in the table. Aggregation functions involving numerical operations, such as a weighted sum or a Choquet integral [5, 16], cannot aggregate non-numerical values. Moreover, our model has to handle the missing values. We propose to use an LPF as a model. Note that, since some values are missing, Table 1 cannot be regarded as a partial function from T^4 to T, and thus the model cannot be inferred directly from the table by solving the interpolation problem. Instead, we propose to proceed as follows.

We will make use of an intermediary scale $\mathbf{T}$ in which partial information about grades is handled in a meaningful way. This scale $\mathbf{T}$ will be defined as a distributive lattice, and an LPF on $\mathbf{T}$ will then be used as a model for aggregating local evaluations into a utility value. We now detail how $\mathbf{T}$ and the LPF are chosen.

We can see T as an interval $[1, 5]$. Let $\mathbf{T}$ be the lattice whose elements are all sub-intervals of $[1, 5]$, and where the supremum and infimum are defined by

$$[a, b] \vee [c, d] = [a \vee c, b \vee d] \quad \text{and} \quad [a, b] \wedge [c, d] = [a \wedge c, b \wedge d],$$

for all $[a, b], [c, d] \in \mathbf{T}$. The lattice $\mathbf{T}$ (depicted in Figure 4) is indeed distributive. Moreover, for any $x \in [a, b]$ and $y \in [c, d]$, the intervals $[a, b] \vee [c, d]$ and $[a, b] \wedge [c, d]$ are in fact the intervals of possible values for $x \vee y$ and for $x \wedge y$, respectively. We identify each grade $a \in T$ to the interval $[a, a]$ and each missing value to the interval $[1, 5]$. We now want to chose an LPF on $\mathbf{T}$ that generalizes the examples given in Table 1. To do so, we regard Table 1 as a partial function f from $\mathbf{T}^4$ to $\mathbf{T}$, we compute the set $\mathcal{IP}(f)$ (i.e., we solve the interpolation problem) and we pick an LPF from this set if it is not empty.

One of the motivations for using LPFs it that they can be seen as rule-based models. Indeed, any LPF can be translated into several single-threshold rules [13, 19]. More precisely, for any distributive lattice L, any LPF $p : L^n \to L$ and all $\mathbf{x} \in L^n$, $p(\mathbf{x})$ is the smallest value allowed by all rules of the form

$$\textbf{if } \forall i \in A, x_i \geq \delta \textbf{ then } p(\mathbf{x}) \geq \delta,$$

where $A \subseteq [n]$ is a focal set of α_p, and $\delta \leq \alpha_p(A)$. One advantage of rule-based models is to be easier to interpret by humans (each prediction of the model can be justified by one or several simple rules). However, it is easy to see that some

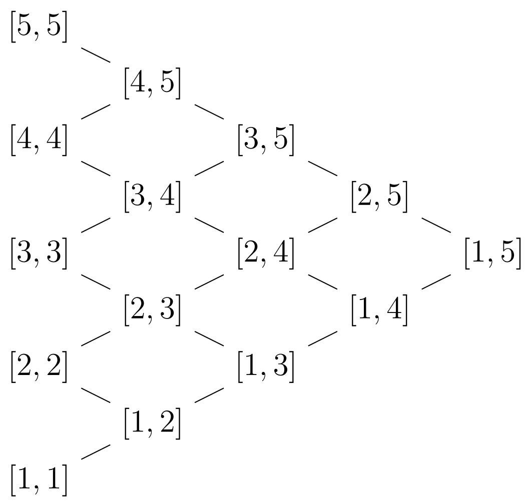


Figure 4: The distributive lattice $\mathbf{T}$ of subintervals of $[1, 5]$.

datasets cannot be modeled by an LPF, even if they can be modeled by a non-decreasing function. One way to deal with some of those datasets can be to divide them into subsets that can be interpolated by an LPF (see, e.g., [26]).

However, in many cases, the ranges of the criteria differ from the set of utility values. To overcome these limitations, the authors of [8, 10] introduced generalizations of Sugeno integrals, such as Sugeno Utility Functionals (SUF) or maxima of SUFs. These models allow to merge local evaluations when the ranges of criteria differ from each other. Moreover, a SUF or a maximum of SUFs $f : X_1 \times \cdots \times X_n \to L$ can always be represented by multi-threshold rules, of the form

$$\textbf{if } \forall i \in A, x_i \geq \delta_i \textbf{ then } f(\mathbf{x}) \geq \delta,$$

where $A \subseteq [n]$, $\delta \in L$, and each δ_i is a threshold belonging to the range of the i^{th} criterion. Note that certain sets of multi-threshold rules do not correspond to any SUF (see [2] and [19]), but always correspond to at least one maximum of SUFs [3]. Moreover, it was shown [6] that maxima of SUFs can be used for extracting rules from data. The rules thus obtained were shown to have a predictive accuracy that is competitive with rules extracted via state of the art methods for rule extraction.

4 Constraint stacking

In this section we show that the upper and lower bounds of $\mathcal{IP}(f)$ can always be characterized by a set of constraints, thus generalizing the result of [27] where L is assumed to be totally ordered, and providing an alternative description to that of [7]. More precisely, all LPFs $p \in \mathcal{IP}(f)$ are characterized by upper and lower constraints, which are respectively of the form $\alpha_p(I) \leq y$ and $\alpha_p(I) \geq y$, where $I \subseteq [n]$ and $y \in L$.

We first consider the particular case when the domain of f only has one element.

Lemma 1. Let $x_1, \ldots, x_n, y \in L$, and let p be an LPF. We have $p(x_1, \ldots, x_n) = y$ if and only if, for every $t \in [k]$

$$\alpha_p\big(G_{\mathbf{x},y,t}\big) \leq s^+_{y,t} \quad \text{and} \quad \alpha_p\big(G'_{\mathbf{x},y,t}\big) \geq s^-_{y,t}$$

where

$$\begin{aligned} s^+_{y,t} &= \bigvee \big\{a \in L \;\big|\; a|_t \leq y|_t\big\}, & s^-_{y,t} &= \bigwedge \big\{a \in L \;\big|\; a|_t \geq y|_t\big\}, \\ G_{\mathbf{x},y,t} &= \big\{i \;\big|\; x_i|_t > y|_t\big\} & G'_{\mathbf{x},y,t} &= \big\{i \;\big|\; x_i|_t \geq y|_t\big\}. \end{aligned}$$

Proof. First observe that

$$\bigvee_{I \subseteq [n]} \alpha_p(I) \wedge \bigwedge_{i \in I} x_i = y$$

holds if and only if for all $t \in [k]$ we have

$$\bigvee_{I \subseteq [n]} \alpha_p(I)|_t \wedge \bigwedge_{i \in I} x_i|_t = y|_t. \tag{4}$$

Notice that the left-hand side of (4) is the DNF of an LPF from $(C_t)^n$ to C_t (recall that we consider a Dilworth embedding of L in $C_1 \times \cdots \times C_k$). Denote this LPF by p^t, where $\alpha_{p^t} : 2^{[n]} \to C_t$ is defined by $\alpha_{p^t}(I) = \alpha_p(I)|_t$ for all $I \subseteq [n]$. Since p^t is defined on the chain C_t and since (4) is equivalent to $p^t(x_1|_t, \ldots, x_n|_t) = y|_t$, it then follows from [27] (Theorem 1) that (4) is equivalent to

$$\alpha_p\big(G_{\mathbf{x},y,t}\big)|_t \leq y|_t, \tag{5}$$

$$\alpha_p\big(G'_{\mathbf{x},y,t}\big)|_t \geq y|_t. \tag{6}$$

Since $s^+_{y,t}$ is the greatest element of L whose tth component is lower than or equal to $y|_t$, (5) is equivalent to $\alpha_p(G_{\mathbf{x},y,t}) \leq s^+_{y,t}$. By duality, since $s^-_{y,t}$ is the least element of L whose tth component is greater than or equal to $y|_t$, (6) is equivalent to $\alpha_p(G'_{\mathbf{x},y,t}) \geq s^-_{y,t}$. □

Using the notation of Lemma 1, we introduce the two following sets:

$$C^+_{\mathbf{x},y} = \Big\{\big(G_{\mathbf{x},y,t}, s^+_{y,t}\big) \;\Big|\; t \in [k]\Big\} \quad \text{and} \quad C^-_{\mathbf{x},y} = \Big\{\big(G'_{\mathbf{x},y,t}, s^-_{y,t}\big) \;\Big|\; t \in [k]\Big\}.$$

for any given $\mathbf{x} \in L^n$ and $y \in L$. From Lemma 1 it follows that for any LPF p we have that $p(\mathbf{x}) = y$ if and only if

$$\forall \big(G_{\mathbf{x},y,t}, s^+_{y,t}\big) \in C^+_{\mathbf{x},y} : \quad \alpha_p\big(G_{\mathbf{x},y,t}\big) \leq s^+_{y,t}$$

and

$$\forall \big(G'_{\mathbf{x},y,t}, s^-_{y,t}\big) \in C^-_{\mathbf{x},y} : \quad \alpha_p\big(G'_{\mathbf{x},y,t}\big) \geq s^-_{y,t}.$$

Example 1. Recall Table 1 in Section 3, where we considered students and their grades in several courses. Local evaluations and utility values are elements from the lattice $\mathbf{T}$. Thus, we are searching for an LPF $p : \mathbf{T}^4 \to \mathbf{T}$ that predicts exactly the utility value of each student. First, let us define a suitable Dilworth's embedding of $\mathbf{T}$. Notice that the lattice $\mathbf{T}$ is isomorphic to the sublattice of T^2 whose universe is the order simplex $\{(a,b) \in T^2 \mid a \leq b\}$. Thus, let $C_1 = C_2 = T$, and consider the Dilworth's embedding of $\mathbf{T}$ in $C_1 \times C_2$ defined by

$$[a,b]|_1 = a \quad \text{and} \quad [a,b]|_2 = b,$$

for all $[a,b] \in \mathbf{T}$.

Recall that each grade $a \in T$ is represented by the interval $[a,a]$ and that a missing grade is represented by the interval $[1,5]$. We denote the tuple of local evaluations of a student by $\mathbf{x}^i$ where i is the number of its row in Table 1. Thus, for the first student we must have

$$p(\mathbf{x}^1) = [4,4] \quad \text{where } \mathbf{x}^1 = ([5,5],[5,5],[1,5],[4,4]).$$

Applying Lemma 1, we get

$$\begin{array}{llll} s^+_{[4,4],1} = [4,5] & s^-_{[4,4],1} = [4,4] & G_{\mathbf{x}^1,[4,4],1} = \{1,2\} & G'_{\mathbf{x}^1,[4,4],1} = \{1,2,4\} \\ s^+_{[4,4],2} = [4,4] & s^-_{[4,4],2} = [1,4] & G_{\mathbf{x}^1,[4,4],2} = \{1,2,3\} & G'_{\mathbf{x}^1,[4,4],2} = \{1,2,3,4\} \end{array}$$

and the corresponding constraints

$$\alpha_p(\{1,2,3\}) \leq [4,4] \quad \text{and} \quad \alpha_p(\{1,2,4\}) \geq [4,4].$$

For the second student we must have

$$p(\mathbf{x}^2) = [3,3] \quad \text{where } \mathbf{x}^2 = ([3,3],[3,3],[3,3],[3,3]).$$

We get

$$\begin{array}{llll} s^+_{[3,3],1} = [3,5] & s^-_{[3,3],1} = [3,3] & G_{\mathbf{x}^2,[3,3],1} = \emptyset & G'_{\mathbf{x}^2,[3,3],1} = \{1,2,3,4\} \\ s^+_{[3,3],2} = [3,3] & s^-_{[3,3],2} = [1,3] & G_{\mathbf{x}^2,[3,3],2} = \emptyset & G'_{\mathbf{x}^2,[3,3],2} = \{1,2,3,4\} \end{array}$$

and the corresponding constraints:

$$\alpha_p(\emptyset) \leq [3,3] \quad \text{and} \quad \alpha_p(\{1,2,3,4\}) \geq [3,3].$$

For the third student we must have

$$p(\mathbf{x}^3) = [5,5] \quad \text{where } \mathbf{x}^3 = ([5,5],[3,3],[3,3],[5,5]).$$

We get

$$\begin{array}{llll} s^+_{[5,5],1} = [5,5] & s^-_{[5,5],1} = [5,5] & G_{\mathbf{x}^3,[5,5],1} = \emptyset & G'_{\mathbf{x}^3,[5,5],1} = \{1,4\} \\ s^+_{[5,5],2} = [5,5] & s^-_{[5,5],2} = [1,5] & G_{\mathbf{x}^3,[5,5],2} = \emptyset & G'_{\mathbf{x}^3,[5,5],2} = \{1,4\} \end{array}$$

and the corresponding constraints:

$$\alpha_p(\emptyset) \leq [5,5] \quad \text{and} \quad \alpha_p(\{1,4\}) \geq [5,5].$$

For the fourth student we must have

$$p(\mathbf{x}^4) = [3,3] \quad \text{where } \mathbf{x}^4 = ([1,5],[1,1],[2,2],[4,4]).$$

We get

$$\begin{array}{llll} s^+_{[3,3],1} = [3,5] & s^-_{[3,3],1} = [3,3] & G_{\mathbf{x}^4,[3,3],1} = \{4\} & G'_{\mathbf{x}^4,[3,3],1} = \{4\} \\ s^+_{[3,3],2} = [3,3] & s^-_{[3,3],2} = [1,3] & G_{\mathbf{x}^4,[3,3],2} = \{1,4\} & G'_{\mathbf{x}^4,[3,3],2} = \{1,4\} \end{array}$$

and the corresponding constraints:

$$\alpha_p(\{1,4\}) \le [3,3] \quad \text{and} \quad \alpha_p(\{4\}) \ge [3,3].$$

For a given $f : D \to L$, the constraints that characterize the upper and lower bounds of $\mathcal{IP}(f)$ are obtained by the union of the constraints given in Lemma 1 for each $\mathbf{x} \in D$. This leads us to the following proposition.

Proposition 1. Let $D \subseteq L^n$, $f : D \to L$, and $p \in \mathcal{P}^n_L$. The following two conditions are equivalent

1.a For all $\mathbf{x} = (x_1, \ldots, x_n) \in D$: $p(\mathbf{x}) = f(\mathbf{x})$.

1.b For all $t \in [k]$ and all $\mathbf{x} \in D$

$$\alpha_p\big(G_{\mathbf{x},y,t}\big) \le s^+_{y,t} \quad \text{and} \quad \alpha_p\big(G'_{\mathbf{x},y,t}\big) \ge s^-_{y,t},$$

where $y = f(\mathbf{x})$ and

$$\begin{array}{rlrl} s^+_{y,t} &= \bigvee \big\{a \in L \mid a|_t \le f(\mathbf{x})|_t\big\}, & s^-_{y,t} &= \bigwedge \big\{a \in L \mid a|_t \ge f(\mathbf{x})|_t\big\}, \\ G_{\mathbf{x},y,t} &= \big\{i \mid x_i|_t > f(\mathbf{x})|_t\big\}, & G'_{\mathbf{x},y,t} &= \big\{i \mid x_i|_t \ge f(\mathbf{x})|_t\big\}. \end{array}$$

The constraints given in Proposition 1 completely describe the upper and lower bounds of $\mathcal{IP}(f)$, whenever $\mathcal{IP}(f)$ is nonempty. Let C^+_f and C^-_f denote the sets upper and lower constraints, respectively. That is,

$$C^+_f = \Big\{\big(G_{\mathbf{x},f(\mathbf{x}),t}, s^+_{y,t}\big) \;\Big|\; t \in [k],\ \mathbf{x} \in D\Big\} = \bigcup_{\mathbf{x}\in D} C^+_{\mathbf{x},f(\mathbf{x})}, \tag{7}$$

$$C^-_f = \Big\{\big(G'_{\mathbf{x},f(\mathbf{x}),t}, s^-_{y,t}\big) \;\Big|\; t \in [k],\ \mathbf{x} \in D\Big\} = \bigcup_{\mathbf{x}\in D} C^-_{\mathbf{x},f(\mathbf{x})}. \tag{8}$$

The natural question is then how to check whether there is an LPF satisfying the conditions presented in Proposition 1, i.e., whether $\mathcal{IP}(f)$ is nonempty. The following proposition provides necessary and sufficient conditions for $\mathcal{IP}(f)$ to be nonempty.

Proposition 2. Let $D \subseteq L^n$ and $f : D \to L$. The three following assertions are equivalent.

2.a. There exists $p \in \mathcal{P}^n_L$ such that $p(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in D$.

2.b. There exists an order-preserving $\alpha : 2^{[n]} \to L$ such that

$$\forall (G, s^+) \in C_f^+, \quad \alpha(G) \le s^+ \quad \text{and} \quad \forall (G', s^-) \in C_f^-, \quad \alpha(G') \ge s^-.$$

2.c. For every $(G, s^+) \in C_f^+$ and $(G', s^-) \in C_f^-$, if $G' \subseteq G$, then $s^- \le s^+$.

Proof. The fact that 2.b is equivalent to 2.a follows immediately from Proposition 1 and from the definitions of C_f^+ and C_f^-.

To prove that 2.c and 2.b are also equivalent, we first show that if 2.c does not hold, then neither does 2.b. Suppose that there is $(G, s^+) \in C_f^+$ and $(G', s^-) \in C_f^-$ such that $G' \subseteq G$ and $s^- \not\le s^+$. In this case, for any $\alpha : 2^{[n]} \to L$ verifying

$$\alpha(G) \le s^+ \quad \text{and} \quad \alpha(G') \ge s^-,$$

we have $\alpha(G') \not\le \alpha(G)$ and thus α is not order preserving.

We now show that 2.c implies 2.b. So suppose that 2.c holds. Take $\alpha : 2^{[n]} \to L$ such that for all $I \subseteq [n]$,

$$\alpha(I) = \bigvee \{s^- \mid (G', s^-) \in C_f^-,\ G' \subseteq I\}. \tag{9}$$

It is not difficult to verify that α is order-preserving, and that for every $(G', s^-) \in C_f^-$, we have that $\alpha(G') \ge s^-$. Let $(G, s^+) \in C_f^+$. From (9) it follows that

$$\alpha(G) = \bigvee \{s^- \mid (G', s^-) \in C_f^-, G' \subseteq G\},$$

and since for all $(G', s^-) \in C_f^-$ such that $G' \subseteq G$ we have $s^- \le s^+$, we have $\alpha(G) \le s^+$. Therefore, 2.c implies 2.b. □

While Proposition 1 defines the constraints that characterize all LPFs interpolating a given partial function, Proposition 2 provides a way of checking if there exists an LPF satisfying these constraints. Even though equivalent to the description provided in [7], this new description enables algorithms to solve the interpolation problem in polynomial time, as we will see in Subsection 6.1.

Example 2. The four students described in Table 1 yield the following sets of constraints.

$$\begin{aligned} C_f^+ = \{&(\{1,2,3\},[4,4]),\\ &(\emptyset,[3,3]),\\ &(\emptyset,[5,5]),\\ &(\{1,4\},[3,3])\} \end{aligned} \qquad \begin{aligned} C_f^- = \{&(\{1,2,4\},[4,4]),\\ &(\{1,2,3,4\},[3,3]),\\ &(\{1,4\},[5,5]),\\ &(\{4\},[3,3])\}. \end{aligned}$$

These two sets contain a few redundancies. For example, in C_f^+, the couple $(\{1,4\},[3,3])$ represents a stronger constraint than $(\emptyset,[3,3])$ and than $(\emptyset,[5,5])$. In C_f^-, the couple $(\{1,4\},[5,5])$ represents a stronger constraint than $(\{1,2,4\},[4,4])$ and than $(\{1,2,3,4\},[3,3])$.

Moreover, notice that no LPF can satisfy both $(\{1,4\},[3,3])$ in C_f^+ and $(\{1,4\},[5,5])$ in C_f^-, since $\{1,4\} \subseteq \{1,4\}$ and $[5,5] > [3,3]$. This can also be seen from the fact that no monotonic function $\alpha : 2^{[n]} \to L$ can verify

$$\mu(\{1,4\}) \leq [3,3] \quad \text{and} \quad \mu(\{1,4\}) \geq [5,5].$$

In other words, there exists no FLP that models exactly the data of Table 1, because the third row is incompatible with the fourth row.

If we decide to remove the last row from the table, then we have that $(\{1,4\},[3,3]) \notin C_f^+$ and $(\{4\},[3,3]) \notin C_f^-$, and the set of interpolating LPFs becomes nonempty. A straightforward model is then the smallest LPF p that satisfies the remaining lower constraints, given by the couples $(\{1,2,4\},[4,4])$, $(\{1,2,3,4\},[3,3])$, and $(\{1,4\},[5,5])$ in C_f^-. This LPF is defined by

$$\alpha_p(I) = \begin{cases} [5,5] \text{ if } I = \{1,4\}, \\ \bigvee_{J \subset I} \alpha(J) \text{ otherwise.} \end{cases}$$

5 Characterization of partial LPFs

Recall that a partial function $f\colon D \to L$ for $D \subseteq L^n$ is said to be a partial LPF if there exists an LPF $p \in \mathcal{P}_L^n$ that interpolates f. In this section we provide several theoretical results that will culminate in a characterization of partial LPFs in terms of so-called *semi-congruences*.

The first result essentially states that we can check whether $\mathcal{IP}(f)$ is nonempty simply by looking at pairs $(\mathbf{x}, f(\mathbf{x})), (\mathbf{x'}, f(\mathbf{x'}))$ for each $\mathbf{x}, \mathbf{x'} \in D$.

Lemma 2. *Let $D \subseteq L^n$ and $f : D \to L$. There exists $p \in \mathcal{P}_L^n$ such that $p(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in D$, if and only if, for all $\mathbf{x}, \mathbf{x'} \in D$, there exists $p \in \mathcal{P}_L^n$ such that $p(\mathbf{x}) = f(\mathbf{x})$ and $p(\mathbf{x'}) = f(\mathbf{x'})$.*

Proof. Clearly, the condition is necessary. To show that it is sufficient, suppose that, for all $\mathbf{x}, \mathbf{x'} \in D$, there exists $p \in \mathcal{P}_L^n$ such that $p(\mathbf{x}) = f(\mathbf{x})$ and $p(\mathbf{x'}) = f(\mathbf{x'})$. From Proposition 2 it follows that for all $\mathbf{x}, \mathbf{x'} \in D$, if $(G, s^+) \in C^+_{\mathbf{x},f(\mathbf{x})} \cup C^+_{\mathbf{x'},f(\mathbf{x'})}$ and $(G', s^-) \in C^-_{\mathbf{x},f(\mathbf{x})} \cup C^-_{\mathbf{x'},f(\mathbf{x'})}$, we have that $G' \subseteq G$ implies $s^- \leq s^+$. From (7) and (8) it follows that for all $(G, s^+) \in C_f^+$ and all $(G', s^-) \in C_f^-$, $G' \subseteq G$ implies $s^- \leq s^+$, and by Proposition 2, we then conclude that there is $p \in \mathcal{P}_L^n$ such that $p(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in D$. □

Let Θ denote an equivalence relation on L. The relation Θ is said to be a *congruence* [1] on L if it is compatible with the lattice operations $\vee$ and $\wedge$, i.e., for any $(a,b), (c,d) \in \Theta$,

$$(a \vee c, b \vee d) \in \Theta \quad \text{and} \quad (a \wedge c, b \wedge d) \in \Theta.$$

A function $f : D \to L$ (where $D \subseteq L^n$) is said to be *congruence-preserving* if for any congruence Θ on L and any $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in D$ such that $(a_1, b_1), \ldots, (a_n, b_n) \in \Theta$, we have that

$$(f(a_1, \ldots, a_n), f(b_1, \ldots, b_n)) \in \Theta.$$

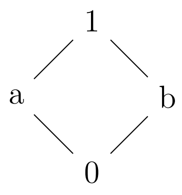


Figure 5: Lattice of Example 3.

It is well-known that the LPFs are congruence preserving functions (see for example [21]). Moreover, it is shown in [20] that Sugeno integrals correspond exactly to congruence-preserving aggregation functions. Since any partial LPF coincides with at least one LPF on each point of its domain, it follows that any partial LPF is congruence-preserving. However, there are partial congruence-preserving functions that are not partial LPF, as illustrated in the following example.

Example 3. Let L denote the lattice depicted in Figure 5, and consider the partial function $f\colon \{a,b\} \to L$ defined by $f(a) = b$ and $f(b) = a$. It is easy to verify that f is congruence-preserving, but that there is no LPF interpolating it.

In order to characterize *partial* LPFs, we introduce the following generalization of the notion of congruence.

Definition 2. We say that a binary relation $R \subseteq L^2$ is a *semi-congruence* if it is reflexive and compatible with $\vee$ and $\wedge$.

We say that a partial function $f\colon D \to L$, $D \subseteq L^n$, is *semi-congruence-preserving* if for any semi-congruence R on L and any

$$(a_1,\dots,a_n),(b_1,\dots,b_n) \in D$$

such that $(a_1,b_1),\dots,(a_n,b_n) \in R$ we have that

$$(f(a_1,\dots,a_n), f(b_1,\dots,b_n)) \in R.$$

For any $\mathbf{x},\mathbf{x'} \in L^n$, we define $R_{(\mathbf{x},\mathbf{x'})}$ as the intersection of all semi-congruences R such that

$$\forall i \in [n] : (x_i, x_i') \in R. \tag{10}$$

In other words, $R_{(\mathbf{x},\mathbf{x'})}$ is the smallest semi-congruence that satisfies (10).

Note that the elements of $R_{(\mathbf{x},\mathbf{x'})}$ can be obtained by finitely many applications of the following rules:

1. $(a,a) \in R_{(\mathbf{x},\mathbf{x'})}$ for all $a \in L$, and $(x_i,x_i') \in R_{(\mathbf{x},\mathbf{x'})}$ for all $i \in [n]$,
2. if $(a,b),(c,d) \in R_{(\mathbf{x},\mathbf{x'})}$, then $(a \vee c, b \vee d) \in R_{(\mathbf{x},\mathbf{x'})}$,
3. if $(a,b),(c,d) \in R_{(\mathbf{x},\mathbf{x'})}$, then $(a \wedge c, b \wedge d) \in R_{(\mathbf{x},\mathbf{x'})}$.

From this observation we immediately have the following auxiliary lemma.

Lemma 3. Let $\mathbf{x},\mathbf{x'} \in L^n$ and $y,y' \in L$. The following assertions are equivalent.

3.a. There is an LPF $p : L^n \to L$ such that $p(\mathbf{x}) = y$ and $p(\mathbf{x'}) = y'$.

3.b. For every semi-congruence R :

$$\big[\forall i \in [n] : (x_i, x'_i) \in R\big] \Rightarrow (y, y') \in R. \tag{11}$$

3.c. $(y, y') \in R_{(\mathbf{x},\mathbf{x'})}$.

Lemmas 2 and 3 together yield the characterization of partial LPFs as semi-congruence preserving functions.

Theorem 4. Let $f : D \to L$ with $D \subseteq L^n$. The following assertions are equivalent.

4.a. The function f is a partial LPF.

4.b. For every $\mathbf{x}, \mathbf{x'} \in D$ and every semi-congruence R :

$$\big[\forall i \in [n] : (x_i, x'_i) \in R\big] \Rightarrow (f(\mathbf{x}), f(\mathbf{x'})) \in R. \tag{12}$$

4.c. For every $\mathbf{x}, \mathbf{x'} \in D$, $(f(\mathbf{x}), f(\mathbf{x'})) \in R_{(\mathbf{x},\mathbf{x'})}$.

6 Computing the set of solutions

In this section we propose a novel approach for solving the interpolation problem and we make some theoretical observations concerning its computational complexity.

6.1 Algorithms

This subsection is devoted to the algorithms that solve the interpolation problem for any given $D \subseteq L^n$ and $f : D \to L$. Algorithm 1 computes the sets of constraints that define the upper and lower bounds of $\mathcal{IP}(f)$, whereas Algorithm 3 checks whether the induced set is empty. The function f is specified by a set

$$\mathcal{D} = \{d^1, \ldots, d^m\} \subseteq L^n \times L,$$

where each d^j is of the form $(\mathbf{x}^j, y^j)$, for $j \in \{1, \ldots, m\}$, and specifies the value $f(\mathbf{x}^j) = y^j$ of f.

Algorithm 1 : Computation of constraints sets C^- and C^+

```
1: function COMPUTE_CONSTRAINTS($\mathcal{D}$)
2:     initalize $C^+$ as an empty map
3:     initalize $C^-$ as an empty map
4:     for $(\mathbf{x}, y) \in \mathcal{D}$ do
5:         for $t \in \{1, \ldots, k\}$ do
6:             $G \leftarrow \{i \mid x_i|_t > y|_t\}$
7:             $G' \leftarrow \{i \mid x_i|_t \geq y|_t\}$
8:             $s^+ \leftarrow \bigvee\{a \in L \mid a|_t \leq y|_t\}$
9:             $s^- \leftarrow \bigwedge\{a \in L \mid a|_t \geq y|_t\}$
10:            UPDATE$^+(C^+, G, s^+)$
11:            UPDATE$^-(C^-, G', s^-)$
12:        end for
13:    end for
14:    return $C^-, C^+$
15: end function
```

The sets C^+_f and C^-_f introduced in Section 4, are thought of as maps $C^+ : 2^{[n]} \to L$ and $C^- : 2^{[n]} \to L$, and their respective values on a key $I \subseteq [n]$ are denoted by $C^+[I]$ and $C^-[I]$. The role of the functions UPDATE$^+$ and UPDATE$^-$ is to add new constraints to C^+ and to C^-, respectively.

Algorithm 2 : Update of maps

```
1: function UPDATE$^+$($C^+$, $G$, $s^+$)
2:     if $G$ is a key in $C^+$ then
3:         $C^+[G] \leftarrow C^+[G] \wedge s^+$
4:     else
5:         $C^+[G] \leftarrow s^+$
6:     end if
7: end function
```

```
1: function UPDATE$^-$($C^-$, $G'$, $s^-$)
2:     if $G'$ is a key in $C^-$ then
3:         $C^-[G'] \leftarrow C^-[G'] \vee s^-$
4:     else
5:         $C^-[G'] \leftarrow s^-$
6:     end if
7: end function
```

The function CHECK_CONSISTENCY returns a Boolean value indicating whether the set of solutions specified by C^+ and C^- is empty.

Algorithm 3 : Checking constraint consistency

```
1: function CHECK_CONSISTENCY($C^+$,$C^-$)
2:     for $(G, s^+) \in C^+$ do
3:         for $(G', s^-) \in C^-$ do
4:             if $G' \subseteq G$ and $s^- \not\leq s^+$ then
5:                 return false
6:             end if
7:         end for
8:     end for
9:     return true
10: end function
```

6.2 Computational complexity

In this section we regard L as a sublattice of a product of chains $C_1 \times \cdots \times C_k$. Recall that $k \leq |\mathcal{J}(L)|$. Here the objective is to provide an upper bound on the worst case complexity of Algorithms 1 and 3 w.r.t. the dimensions of the input, i.e., the arity n of f, the size m of $\mathcal{D}$, and the parameter k.

We start by looking at the worst case complexity of evaluating any of the following expressions: $a \vee b$, $a \wedge b$, $a \leq b$, and $a < b$ (we regard $a \leq b$ and $a < b$ as predicates that can be either 0 or 1 depending on whether they are true or false). By (1) and (2) in Section 2, any of these four expressions can be computed in $O(k)$. However, since these expressions are evaluated many times in our algorithms, it is convenient to precompute their values for each couple $(a, b) \in L \times L$. After indexing each element of L by an integer, the precomputed values can be stored in four two-dimensional arrays of size $|L|^2$. Since each of these arrays contains at most $|L|^2$ values, $O(|L|^2)$ values have to be computed. Hence, the overall worst time complexity of these precomputations is $O(|L|^2 k)$. Once the arrays are created and filled, any value can be retrieved in $O(1)$.

Observe that the values of the variables s^+ and s^- (which are set, respectively to $s^+_{y,t}$ and $s^-_{y,t}$ during each loop) are determined by the value of $y|_t$. Since $y|_t \in \mathcal{J}(L) \cup \{\perp_L\}$, both s^+ and s^- can take $|\mathcal{J}(L)| + 1$ different values. Again, these values can be precomputed and stored in arrays. For a given $y|_t \in \mathcal{J}(L) \cup \{\perp_L\}$, $s^+_{y,t}$ and $s^-_{y,t}$ can be computed in $O(|L|)$. Thus, computing all possible values of s^+ or s^- can be done in $O(|L||\mathcal{J}(L)|)$. In what follows, it is assumed that the values $s^+_{y,t}$ and $s^-_{y,t}$ are computed and stored for each $y|_t \in \mathcal{J}(L) \cup \{\perp_L\}$, and thus that the values of s^+ and s^- can be be retrieved in $O(1)$.To implement the maps C^+ and C^- we can make use of *tries* (see, e.g., [4], Section 8.1). Through tries, finding and inserting an item in C^+ or C^- can be performed with a complexity $O(n)$. Consequently, the functions UPDATE$^+$ and UPDATE$^-$ run in $O(n)$.

Now consider Algorithm 1. The code from lines 6 to 11 will not be executed more than mk times. Moreover, G and G' can be computed in $O(n)$. Therefore, Algorithm 1 runs in $O(mkn)$.

The number of loops performed in Algorithm 3 depends on $|C^+|$ and $|C^-|$. From Algorithm 1, it can be seen that this number is at most mk. Thus, line 4 will be executed at most m^2k^2 times. Since $G' \subseteq G$ can be checked in $O(n)$, Algorithm 3 runs in $O(m^2k^2n)$. This is, to the best of our knowledge, the only algorithm that computes $\mathcal{IP}(f)$ in polynomial time, for any partial function from $D \subseteq L^n$ to L.

Remark 4. Corollary 2 of [7] also allows to compute the greatest and the least LPFs that interpolate a set of data instances. However, any naive implementation of this theoretical result would lead to an algorithm running in $\Omega(2^n mn)$. This exponential complexity is due to the fact that the values of the capacities have to be computed explicitly for each subset of $[n]$.

7 Empirical study

In this section we perform an empirical study to see the behaviour of our algorithms with respect to the dimensions of datasets and compare the results with the theoretical complexity results obtained in the previous section. We will conduct our study on randomly generated data as described below.

Note that our empirical study relies on an implementation of algorithms that makes use of the trivial Dilworth's embedding, i.e., Birkhoff's representation where the elements of the lattice are represented by binary tuples of size $|\mathcal{J}(L)|$. Therefore, in this section we assume that $k = |\mathcal{J}(L)|$.

7.1 Random generation of instances

In order to get an idea of how the running time of our algorithm varies w.r.t. different parameters, we randomly generated datasets of different sizes. The main steps of the data generation can be summarized as follows:

1. Randomly generate a distributive lattice L,
2. Randomly generate an LPF $p : L^n \to L$,
3. Randomly generate a set of input-output instances. Each input is a randomly generated element $\mathbf{x} \in L^n$, and the corresponding output is given by $p(\mathbf{x})$.

When generating the data in this way, it is ensured that there will always be at least one LPF that will coincide with all input-output instances. Each step depends on several parameters and is detailed in what follows.

7.1.1 Random generation of distributive lattices

Each distributive lattice is characterized by the ordering of its join-irreducible elements (see [12]). Therefore, when the number of join-irreducible elements is given, we can reformulate the problem of randomly generating a distributive lattice as the one of randomly generating a partial order over a set. This is not a trivial problem, and several solutions have been proposed in the literature, see, e.g., [14]. Each of them induces a different probability distribution over the space of outcomes, i.e., orders.

The method that we consider in this paper can be decomposed into three main steps

1. Randomly generating an acyclic relation over the set of joint-irreducible elements. For this we assign an arbitrary index $i \in [k]$ to each join-irreducible element of L, so that $\mathcal{J}(L) = \{a_1, \ldots, a_k\}$. Then for each $i, j \in [k]$ such that $i < j$, we set $a_i < a_j$ with probability ρ.
2. Taking the transitive closure of the relation.
3. Building the lattice from the poset of join-irreducible elements.

Summing up, the generation of each lattice considers two parameters, namely, the number k of join-irreducible elements, and the probability ρ.

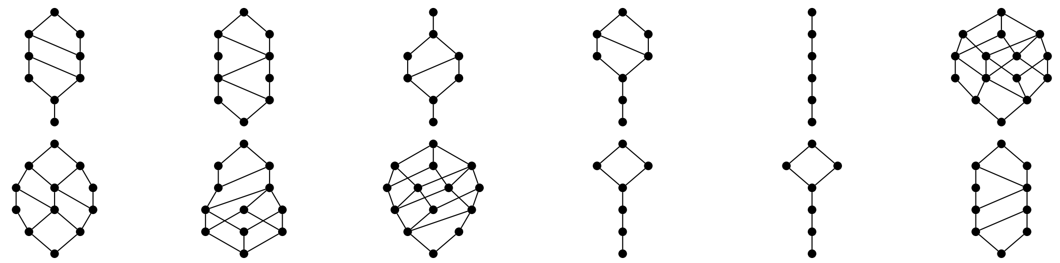


Figure 6: Sample of randomly generated distributive lattices ($k = 5$, $p = 0.6$).

7.1.2 Random generation of LPFs

Given a distributive lattice L, and a positive integer n, we randomly generate an LPF $p : L^n \to L$. We introduce a third parameter, which is the maximal number g of focal sets (see Section 2). The generation process can then be decomposed as follows.

1. Pick g different sets $I_1, \dots, I_g$ in $2^{[n]}$, with uniform probability. Let $F = \{I_1, \dots, I_g\}$. The function α_p will be defined is such a way that all its focal sets are in F.

2. For i from 1 to g, randomly pick $a_i \in L$ such that

$$\bigvee_{j<i,\ I_j \subset I_i} a_j \leq a_i \leq \bigwedge_{j<i,\ I_i \subset I_j} a_j$$

   and set $\alpha_p(I_i) = a_i$. Note that a_1 is picked such that $\bot_L \leq a_1 \leq \top_L$. For every $J \in 2^{[n]} \backslash F$, we set

$$\alpha_p(J) = \bigvee_{I_i \in F,\ I_i \subseteq J} \alpha_p(I^i).$$

   Thus J is not a focal set of α_p. These choices guarantee that α_p is order-preserving. Note that, by making use of the ordinal Möbius transform of α_p (see Section 2), we can avoid to set explicitly the values of each $J \in 2^{[n]} \backslash F$.

We then return the LPF p defined by $p = \bigvee_{I \in F} \alpha_p(I) \wedge \bigwedge_{i \in I} x_i$. The generation of p thus depends on two parameters, namely the arity n of p and the number of focal sets g.

7.1.3 Random generation of a dataset

In order to randomly generate a dataset of m instances, m elements of L^n are chosen randomly (with uniform probability). Let

$$D = \{\mathbf{x}^1, \dots, \mathbf{x}^m\} \subseteq L^n,$$

be the set of these randomly chosen elements. The dataset is then defined as

$$\mathcal{D} = \{(\mathbf{x}^i, p(\mathbf{x}^i)) \mid \mathbf{x}^i \in D\}.$$

		Maximal number of focal sets (g)					
		10	50	100	200	500	1000
ρ	0.0	151.0	159.6	147.0	171.5	142.3	147.8
	0.2	153.2	149.1	132.3	135.5	132.9	146.8
	0.4	134.1	139.7	161.2	139.4	136.7	136.3
	0.6	147.3	140.2	131.3	154.2	138.5	132.2
	0.8	130.2	135.0	140.4	132.5	132.4	132.9
	1.0	133.6	137.8	125.8	127.1	137.1	139.3

Table 2: Running time of Algorithms 1 and 3 (in milliseconds) depending on the maximal number g of focal sets and probability ρ (with $n = 30$, $m = 10000$, $k = 10$, $p = 0.6$).

7.2 Test procedure

We aim at observing how the running time of our algorithms varies depending on the five parameters used in the generation of a dataset, namely: the number k of joint-irreducible elements of L, the probability ρ used in the generation of L, the arity n of p, the maximal number g of focal sets in α_p, and the number m of instances in $\mathcal{D}$.

To conduct this empirical study, we run several tests. Each test consists in the random generation of $\mathcal{D}$, followed by the execution of Algorithms 1 and 3 on $\mathcal{D}$. A test was run for different parameter values. The result of one test is the time of execution of Algorithms 1 and 3. The algorithms were implemented in Java and their implementation is available in `https://github.com/QGBrabant/javaggregation`. The tests were run on an Intel Core i7-4600U CPU. Times displayed in Tables 2 and 3 are an average result of 10 tests.

Table 2 displays the times obtained, for varying values of ρ and g. We see that these two parameters have a minor impact on the running time. Table 3 describes the running times for fixed values of ρ and g, and varying arity, dataset size, and lattice dimension. The running times presented in Table 3 seem to be in line with the worst case complexity $O(m^2k^2n)$ given in Subsection 6.2. It shows that the set of interpolating LPFs can be found in reasonable time for data with substantial dimensionality. Moreover, we see that the reported running times do not increase as fast with the dimensions of the data as their worst case complexity would have suggested. A rough approximation would be that the running time increases linearly on k, and slightly more than linearly on m and n, while being almost constant on ρ and g.

8 Future work

In this paper we provided theoretical results regarding the interpolation of functions by an LPF. These results show that the interpolation problem is tractable even in the case of distributive lattices, and we proposed an algorithm that solves it in polynomial time. However, it is common that empirical data contain errors; these errors can prevent from finding any interpolating LPF. As future research, we aim to adapt the current algorithms to make them robust and capable of dealing with noise.

Furthermore we intend to extend our interpolation framework to SUFs (see Section 3). In fact a solution to the interpolation problem generalized to a certain class of restricted SUFs was already proposed in [10]. However it was shown that the interpolation problem in that case is NP-complete. On the other hand, it is still unknown whether the interpolation problem is tractable in the case of unrestricted SUFs. Finally, the interpolation of a function by a minimal number of SUFs in a max-SUF is a problem that remains open and constitutes a topic of ongoing research.

References

[1] G. Birkhoff. *Lattice Theory*, volume 25. American Mathematical Society, New York, USA, 1940.

[2] D. Bouyssou, T. Marchant, and M. Pirlot. A Conjoint Measurement Approach to the Discrete Sugeno Integral. In *The Mathematics of Preference, Choice and Order*, Studies in Choice and Welfare, pages 85–109. Springer, Berlin, Heidelberg, 2009.

[3] Q. Brabant, M. Couceiro, D. Dubois, H. Prade, and A. Rico. Extracting Decision Rules from Qualitative Data via Sugeno Utility Functionals. In *International Conference on Information Processing and Management of Uncertainty (IPMU)*, pages 253–265, Cadiz, Spain, 2018.

[4] P. Brass. *Advanced Data Structures.* Cambridge University Press, Leiden, 2008.

[5] G. Choquet. Theory of capacities. In *Annales de l'institut Fourier*, volume 5, pages 131–295, 1954.

[6] M. Couceiro, D. Dubois, H. Prade, and A. Rico. Enhancing the Expressive Power of Sugeno Integrals for Qualitative Data Analysis. In *Advances in Fuzzy Logic and Technology 2017*, Advances in Intelligent Systems and Computing, pages 534–547. Springer, Cham, 2017.

[7] M. Couceiro, D. Dubois, H. Prade, A. Rico, and T. Waldhauser. General interpolation by polynomial functions of distributive lattices. In S. Greco, B. Bouchon-Meunier, G. Coletti, M. Fedrizzi, B. Matarazzo, and R. R. Yager, editors, *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, volume 298 of *Communications in Computer and Information Science*, Catania, Italy, 2012.

[8] M. Couceiro, D. Dubois, H. Prade, and T. Waldhauser. Decision-Making with sugeno integrals - bridging the gap between multicriteria evaluation and decision under uncertainty. *Order*, 33:517–535, 2016.

[9] M. Couceiro and J.-L. Marichal. Characterizations of discrete Sugeno integrals as polynomial functions over distributive lattices. *Fuzzy Sets and Systems*, 161(5):694–707, 2010.

[10] M. Couceiro, M. Maróti, T. Waldhauser, and L. Zadori. Computing version spaces in the qualitative approach to multicriteria decision aid. To appear in *International Journal of Foundations of Computer Science.*

[11] M. Couceiro and T. Waldhauser. Interpolation by polynomial functions of distributive lattices: a generalization of a theorem of R. L. Goodstein. *Algebra universalis*, 69(3):287–299, 2013.

[12] B. A. Davey and H. A. Priestly. *Introduction to Lattices and Order.* Cambridge University Press, Cambridge, UK, 2002.

[13] D. Dubois, H. Prade, and A. Rico. The logical encoding of Sugeno integrals. *Fuzzy Sets and Systems*, 241:61–75, 2014.

[14] W. V. Gehrlein. On methods for generating random partial orders. *Operations research letters*, 5(6):285–291, 1986.

[15] R. L. Goodstein. The solution of equations in a lattice. *Proceedings of the Royal Society of Edinburgh. Section A. Mathematical and Physical Sciences*, 67(3):231–242, 1967.

[16] M. Grabisch and C. Labreuche. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 175(1):247–290, 2010.

[17] M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap. *Aggregation Functions*, volume 127 of *Encyclopedia of Mathematics and Its Applications.* Cambridge University Press, New York, NY, USA, 2009.

[18] G. Grätzer. *General Lattice Theory.* Springer Science & Business Media, Berlin, Germany, 2003.

[19] S. Greco, B. Matarazzo, and R. Slowinski. Axiomatic characterization of a general utility function and its particular cases in terms of conjoint measurement and rough-set decision rules. *European Journal of Operational Research*, 158(2):271–292, 2004.

[20] R. Halaš, R. Mesiar, and J. Pócs. Congruences and the discrete Sugeno integrals on bounded distributive lattices. *Information Sciences*, 367-368:443–448, 2016.

[21] K. Kaarli and A. F. Pixley. *Polynomial Completeness in Algebraic Systems.* Chapman & Hall/CRC, Boca Raton, USA, 2000.

[22] R. N. Larson. Embeddings of Finite Distributive Lattices into Products of Chains. *Semigroup Forum*, 56(1):70–77, 1998.

[23] J.-L. Marichal. *Aggregation Operators for Multicriteria Decision Aid.* PhD thesis, Institute of Mathematics, University of Liège, Liège, Belgium, 1998.

[24] J.-L. Marichal. Weighted lattice polynomials. *Discrete Mathematics*, 309(4):814–820, 2009.

[25] R. Mesiar. k-order pan-additive discrete fuzzy measures. In *7th International Fuzzy Systems Association World Congress (IFSA)*, pages 488–490, Prague, Czech Republic, 1997.

[26] H. Prade, A. Rico, M. Serrurier, and E. Raufaste. Elicitating Sugeno integrals: Methodology and a case study. In C. Sossai and G. Chemello, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertain*, volume 5590 of *Lecture Notes in Computer Science*, pages 712–723, Verona, Italy, 2009. Springer-Verlag, Berlin, Germany.

[27] A. Rico, M. Grabisch, C. Labreuche, and A. Chateauneuf. Preference modeling on totally ordered sets by the Sugeno integral. *Discrete Applied Mathematics*, 147(1):113–124, 2005.

[28] M. Sugeno. *Theory of fuzzy integrals and its applications.* Tokyo Institute of Technology, 1974.

A Detailed running times

	Dataset size				
Arity	10	100	1000	10000	50000
1	0.2	0.1	0.3	1.9	10.8
2	0.0	0.0	0.3	2.5	15.1
3	0.0	0.0	0.3	3.1	17.6
4	0.0	0.1	0.3	3.4	21.8
5	0.0	0.1	0.4	4.1	25.0
6	0.0	0.1	0.4	5.4	25.7
10	0.0	0.1	0.7	6.7	31.8
15	0.0	0.1	0.7	8.2	44.3
20	0.0	0.2	1.1	9.5	50.5
30	0.0	0.2	1.3	13.4	66.3

$k = 1$

	Dataset size				
Arity	10	100	1000	10000	50000
1	0.1	0.2	0.5	3.4	15.7
2	0.0	0.1	0.5	4.3	23.6
3	0.0	0.1	0.5	5.1	29.0
4	0.0	0.1	0.6	6.5	32.9
5	0.0	0.1	0.7	6.9	36.7
6	0.0	0.1	0.8	8.4	38.6
10	0.0	0.2	1.2	11.7	60.8
15	0.0	0.2	1.4	16.2	91.2
20	0.0	0.2	1.8	19.3	96.6
30	0.1	0.3	2.6	24.3	134.4

$k = 2$

	Dataset size				
Arity	10	100	1000	10000	50000
1	0.1	0.3	1.0	7.7	34.7
2	0.0	0.1	1.2	9.8	45.1
3	0.0	0.1	1.1	11.0	55.8
4	0.0	0.2	1.4	12.9	64.4
5	0.0	0.2	1.7	16.6	76.9
6	0.1	0.2	1.7	17.3	87.3
10	0.1	0.3	3.2	25.9	133.0
15	0.1	0.3	3.6	39.6	215.1
20	0.1	0.5	3.9	46.1	250.8
30	0.1	0.7	5.6	60.5	334.1

$k = 5$

	Dataset size				
Arity	10	100	1000	10000	50000
1	0.1	0.4	1.5	13.8	62.2
2	0.0	0.2	2.0	19.3	82.2
3	0.0	0.2	2.2	19.9	98.6
4	0.1	0.3	2.6	23.3	121.0
5	0.1	0.4	3.3	29.5	143.8
6	0.1	0.4	3.6	32.5	159.0
10	0.1	0.7	5.5	50.6	249.7
15	0.1	0.7	6.9	73.4	396.7
20	0.1	0.9	9.3	93.4	510
30	0.2	1.3	13.0	131.3	696.3

$k = 10$

	Dataset size				
Arity	10	100	1000	10000	50000
1	0.1	0.4	2.5	22.6	113.9
2	0.1	0.4	3.3	29.7	148.9
3	0.1	0.4	3.6	37.4	188.6
4	0.1	0.5	5.0	44.9	228.7
5	0.1	0.7	6.4	54.8	271.9
6	0.1	0.7	6.5	59.2	306.6
10	0.2	1.1	10.8	96.0	485.6
15	0.2	1.4	13.5	145.1	769.2
20	0.2	1.8	16.3	180.6	995.5
30	0.3	2.4	24.2	249.9	1387.8

$k = 20$

	Dataset size				
Arity	10	100	1000	10000	50000
1	0.2	0.6	3.6	32.8	166.9
2	0.1	0.6	5.0	44.4	217.8
3	0.1	0.6	5.9	55.6	279.9
4	0.1	0.8	7.0	68.1	333.4
5	0.1	0.9	8.4	85.1	401.4
6	0.2	1.0	9.7	89.4	444.9
10	0.2	1.8	14.4	137.3	691.6
15	0.2	1.9	21.2	213.2	1099.7
20	0.4	2.6	26.5	266.4	1435.6
30	0.4	3.6	38.1	374.4	2323.8

$k = 30$

Table 3: Time required (in milliseconds), for varying n, k, and m ($g = 200$, $p = 0.6$).