Computational Statistics and Data Analysis 173 (2022) 107520

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

www.elsevier.com/locate/csda

Complexity reduction and approximation of multidomain systems of partially ordered data

Alberto Arcagni [a,*], Alessandro Avellone [b], Marco Fattore [b]

[a] MEMOTEF Department, Sapienza University of Rome, Via Del Castro Laurenziano 9, 00161, Rome, Italy
[b] Department of Statistics and Quantitative Methods, University of Milano-Bicocca, Piazza dell'Ateneo Nuovo, 1, 20126, Milan, Italy

ARTICLE INFO

Article history:
Received 24 July 2021
Received in revised form 22 April 2022
Accepted 23 April 2022
Available online 29 April 2022

Keywords:
Bucket order
Complexity reduction
Multi-indicator system
Multidimensional ordinal data
Partially ordered set
Ranking

ABSTRACT

Two greedy algorithms for the synthesis and approximation of multidomain systems of partially ordered data are proposed. Given k input partially ordered sets (posets) on the same elements, the algorithms search for the optimally approximating partial orders, minimizing the dissimilarity between the generated and input posets, based on their matrices of mutual ranking probabilities. A general approximation algorithm is developed, together with a specific procedure for approximation over bucket orders, which are the natural choice when the goal is to "condense" the inputs into rankings, possibly with ties. Different loss functions are also employed, and their outputs are compared. A real example pertaining to regional well-being in Italy motivates the algorithms and shows them in action.

1. Introduction and a motivating example

This paper introduces two new algorithms for the synthesis and approximation of multidomain systems of ordinal indicators and partially ordered data, a type of statistical structure that is increasingly often encountered in multicriteria decision making, synthetic indicator construction, and evaluation studies. There, one often deals with a variety of ordinal indicators, usually organized in subsystems, which describe the different facets of the trait under investigation and typically score statistical units in conflicting ways, making it impossible to totally order the statistical population or sample. Remarkably, the existence of units that cannot be compared and placed on a "low–high" axis is not primarily due to missing information; rather, it is a manifestation of the complexity and the multifaceted nature of the underlying trait (Comim, 2021), which is reflected in the partial order structure of the data, as is made clear by the following example pertaining to subjective well-being in Italy, used in the text to motivate the algorithms and show them in action.

Motivating example. We consider the subjective well-being of the Italian population at a regional level, in three domains: satisfaction in *personal economic situation*, satisfaction in *health and family relations*, and satisfaction in *leisure time and friendship*. Data have been extracted from the "Multipurpose Survey: Aspects of Daily Life," carried out in 2015 by the Italian National Statistical Bureau (Istat, 2015), on 45 204 individuals (reduced to 38 711 after removing records with missing entries). The regional distributions of individual scores in the three domains have been compared, using a multidimensional fuzzy extension of the first-order dominance criterion, described in Fattore and Arcagni (2019) (computations not reported here). In the end, it turns out that only some pairs of regions can be ordered in terms of lower or higher well-being, while

* Corresponding author.
E-mail addresses: alberto.arcagni@uniroma1.it (A. Arcagni), alessandro.avellone@unimib.it (A. Avellone), marco.fattore@unimib.it (M. Fattore).

https://doi.org/10.1016/j.csda.2022.107520
0167-9473/

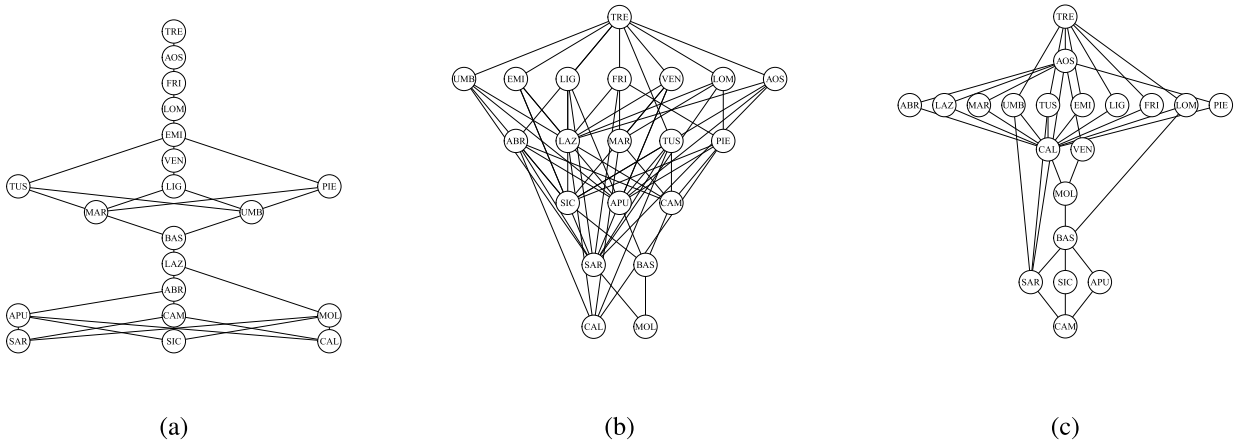

Fig. 1. Input posets for the well-being motivating example: (a) personal economic situation; (b) health and family relations; (c) leisure time and friendship (region labels are given in Table 5). These so-called *Hasse diagrams* are to be read from top to bottom: if a downward path of edges links region p to region q, then the two regions are *comparable* and the first *dominates* the second, in well-being terms; if no path exists between them, then the two regions are *incomparable*.

Table 1
Basic structural indicators for the well-being posets of the motivating example ("Inc": number of incomparabilities; "Comp": number of strict comparabilities).

Poset	Inc	Comp	Inc/Comp
a	13	117	0.073
b	74	116	0.638
c	76	114	0.667

others cannot, owing to the existence of structural differences that make their satisfaction distributions inherently incomparable. The patterns of regional comparabilities/incomparabilities in the three domains are represented by the partially ordered sets (*posets*) depicted in Fig. 1. Notice the different structures of the three posets: although all of them neatly reveal the typical Italian North–South polarization, with Trentino – Alto Adige at the top, posets (b) and (c) have more incomparabilities than poset (a), which orders its elements with less ambiguity, showing that Italian regions can be compared along the economic perspective in a much neater way than along the other dimensions of well-being considered in the example (see Table 1).

In the contexts of well-being assessment and of policy-making, the question naturally arises as to whether and how the three well-being posets can be "condensed" into one, reducing the complexity of the input data, still preserving their essential comparability/incomparability patterns, and providing a synthetic picture of the level and diversity of subjective well-being across different areas of the country. And since, for example for public communication, regional rankings can also be of interest, one may also ask how to order Italian regions, possibly arranged into classes (later called *buckets*), on a subjective well-being axis, minimizing the information loss given the input posets.

Cast in more general terms, the issue addressed in this paper is *designing algorithms to find out the "best approximating" poset and the "optimal" ranking (possibly with ties), given a set of k input posets on the same elements*. This complexity reduction task is mathematically and computationally nontrivial and requires exploiting the tools of order theory, a branch of discrete mathematics providing the "grammar" of order relations and multidimensional ordinal data. We address it by first developing a greedy "linearization" algorithm that generates a sequence of progressively "more linear" posets, searching for the partially ordered pattern that *structurally* best approximates the inputs. We then restrict the greedy search to a class of simpler posets, called *bucket orders*, and design a specific "bucketization" algorithm for the approximation of the inputs through rankings with ties. We implement both algorithms with various loss functions, to explore different ways of driving the optimization process.

The remainder of the paper is organized as follows. Section 2 briefly discusses the increasing relevance of partially ordered sets in data analysis; Section 3 provides a review of the literature; Section 4 presents some basic notions about partially ordered sets, needed in the paper; Section 5 describes the linearization algorithm with L_1 loss function and works out a toy example; Section 6 applies the linearization algorithm to the Italian data on regional subjective well-being; Section 7 proposes some alternative loss functions based on entropy and the Jensen–Shannon divergence; Section 8 introduces and exemplifies bucket orders and the "bucketization" algorithm; Section 9 briefly discusses computational issues; and Section 10 concludes the paper.

2. A few remarks on order theory and data analysis

Before going into the technical details of the paper, we briefly illustrate why partially ordered sets and the theory of order relations are gaining relevance in multivariate data analysis, particularly, but not exclusively, in socio-economic statistics. In brief, posets and order theory matter since they provide the natural data structures for many multidimensional statistical problems, together with the "mathematical grammar" to properly address them. Such a "naturality" traces back to two main roots. First, many topics in applied statistics and multicriteria decision making (e.g., the evaluation and measurement of well-being, of non-monetary deprivation, of skills and competencies, of service or product quality…) are often and naturally addressed through multidimensional systems of *ordinal* indicators, which cannot be consistently treated with metric tools. Ordinal data can only be compared in terms of *greater than* | *lower than*, and since the different indicators typically order statistical units in conflicting ways, these cannot be globally ordered and only a partial order relation can be defined on them. Ordinal data analysis then becomes the study of and the extraction of information from *partially* and *quasi-ordered* structures, with the aid of *order theory* (Davey and Priestley, 2002) and the mathematics of binary relations. Second, and perhaps more deeply, applied statistics increasingly often deals with the description and synthesis of multifaceted and multishaped issues and phenomena, where the impossibility of comparing all of the respective manifestations is intrinsic to their essential complexity. Consider, to take a prototypical example, the description and evaluation of multidimensional poverty, in view of policy-making. Individuals can be poor to different extents, but also in many different and incomparable ways, each calling for its proper contrasting action. Reducing all of the shapes and the patterns of poverty to the same "evaluation axis,"making all of them comparable, would lose a great deal of information on deprivation, distorting its picture. Preserving and reproducing these kinds of structural differences is therefore essential for obtaining faithful syntheses of complex socio-economic issues, and posets, which can account for both comparabilities and incomparabilities, become key tools for capturing their multifaceted nature (see, e.g., Fattore (2016) and Arcagni et al. (2019) on deprivation, Comim (2021) on human development, and the masterful discussion by Sen (1992) of the relevance of preserving "partiality" of orderings in inequality and deprivation studies). From this point of view, posets are not confined to ordinal data, and indeed they have been used to describe the structure of numerical indicator systems, whenever the focus is on multidimensional comparisons and the construction of synthetic indicators, overcoming the conceptual and practical difficulties of compensative aggregations, typical of composite indicators (see, e.g., Fattore et al. (2012), Di Bella et al. (2018), and in particular Brüggemann and Patil (2011) and Fattore and Brüggemann (2017), where various applications of partial order theory to data analysis in environmental and ecological sciences, socio-economics, engineering sciences, metrology, genetics, and biology can be found). Notwithstanding the naturality of partially and quasi-ordered sets as data structures for the description and analysis of many empirical relations across a variety of disciplines, it must be acknowledged that their use in data analysis is still at an early stage and that the "abstraction leap" of considering ordered structures as statistically interesting in themselves has not yet been fully accepted. This can be partly explained by a historical tendency, for example in the socio-economic statistical literature, to bend toward hard modeling and considering ordinal data and soft models as not completely satisfactory ways of representing reality. This, combined with order theory not being in the typical cultural and technical toolbox of theoretical and applied statisticians, has led to ordinal data often being looked at as rough manifestations of "truly" hidden numerical variables and to the habit of treating them metrically, thereby hindering the development of alternative approaches. (For a deep criticism of the application of metric models to ordinal data, with a discussion of the problematic consequences, see, e.g., Liddell and Kruschke (2018); see also Madden (2010) on similar problems with self-reported health data.) Interestingly, this obstacle to development is less of an issue in other fields of application, where the role of ordered structures has been acknowledged for a longer time, namely, in artificial intelligence and knowledge discovery, in mathematical psychology, in social choice theory, and in the analysis of preference data (see also Section 3 below). Finally, it should be noted that the ordered structures used in data analysis are combinatoric in nature and pose nontrivial computational issues that, in the past, limited their range of applicability. The growth in computational power and the development of effective algorithms are progressively overcoming these limitations, making posetic tools increasingly attractive to the data analysis community.

3. Poset approximation and reconstruction: literature review

The literature related to poset approximation and poset "reconstruction" is still limited, although it has recently been expanding at a faster pace. Broadly speaking, this body of research can be divided into three main areas. The first addresses the problem of approximating and reducing the complexity of an input partial order, in the spirit of *dimensionality reduction* procedures. To the best of our knowledge, the main attempt in this direction, with the principal aim of optimally scaling partially ordered data to real coordinates and visualizing them in the Cartesian plane, is so-called *partial order Scalogram Analysis by Base Coordinates* (*POSAC*), proposed by Shye in the context of *Facet theory* (Shye, 1978; Shye and Amar, 1985; Shye, 1985a,b, 2009). The POSAC algorithm directly searches for a bidimensional configuration of the partially ordered units, trying to reproduce their comparabilities and incomparabilities, and minimizing a suitable loss function. Somewhat related to this problem are the attempts to extend classical dimensionality reduction procedures, such as principal component analysis, to multidimensional ordinal data (Korhonen and Siljamäki, 1998), to develop clustering algorithms (Jacques and Biernacki, 2018; Zhang et al., 2020), or to work out binary matrix decompositions for binary data reduction (Belohlávek and Vychodil, 2010), in connection with *Formal Concept Analysis* (Ganter and Wille, 1999). However, in these cases, input data are not explicitly structured into posets, and the proposed algorithms are not explicitly intended as tools for approximating

multidimensional ordered structures. The second research area, mostly developed in the context of multicriteria decision making and multi-indicator system analysis, focuses on ranking extraction from multidimensional and partially ordered data. The problem is usually addressed by computing some synthetic score, to capture the "degree of dominance" of the poset elements and to order them according to it (see also Section 8). Dominance scores are derived from structural analysis of the dominances between pairs of elements, as in the case of *average heights* (Brüggemann and Patil, 2011) or *SVD-based scores* (Fattore and Arcagni, 2020); alternatively, rankings can be extracted through the iterative comparison of suitable frequency distributions associated with the elements of the poset, as in Patil and Taillie (2004). Although here the input data are posets, these procedures do not address the extraction of rankings as a poset approximation problem but mostly as an issue of synthetic indicator construction. The third research area deals with the problem of the reconstruction of posets from some limited knowledge of their structure. Most reconstruction algorithms are non-inferential (Mannila and Meek, 2000; Gionis et al., 2003; Ukkonen et al., 2005; Garriga, 2005), but some model-based approaches do exist (Beerenwinkel et al., 2007; Watt, 2015). Often, the target unknown poset is supposed to have a simple shape, as in the case of so-called *bucket orders* (i.e., informally speaking, of rankings with ties; see Section 8), which are relevant in many fields, for example in connection with the seriation problem in paleontology (Puolamäki et al., 2006). Algorithms for the reconstruction of bucket orders (or their subclasses) are available in Fernandez et al. (2013) Feng et al. (2008), Ukkonen et al. (2009), Aledo et al. (2017), and D'Ambrosio et al. (2019). Somewhat related to this research is the problem of reconstructing preferences, from partial information, usually in the context of the Mallows models (Lu and Boutilier, 2014) and the Plackett–Luce models (Liu et al., 2019; Zhao and Xia, 2019, 2020).

All in all, the problem of poset approximation (or reconstruction) appears in a variety of disciplines and has been addressed in various ways. Still, to the best of our knowledge, no algorithms have yet been constructed to explicitly approximate *many known* posets, as we do in the present paper.

4. Elements of partial order theory

In the following, we provide some essential definitions and results about partially ordered sets, needed in the development of the approximation algorithms.

Basic definitions. A partially ordered set (poset) $\pi = (X, \trianglelefteq)$ is a set X equipped with a partial order relation $\trianglelefteq$, that is, a binary relation satisfying the properties of *reflexivity*, *antisymmetry*, and *transitivity* (Davey and Priestley, 2002; Schröder, 2003). In general, many different partial order relations can be imposed on a set, and, in the following, we write Π_X for the set of posets over X. Let $\pi \in \Pi_X$ and let x_i and x_j be elements of π. If $x_i \trianglelefteq x_j$ or $x_j \trianglelefteq x_i$, then x_i and x_j are called *comparable*, otherwise they are called *incomparable* (written $x_i || x_j$). We write $x_i \triangleleft x_j$ when $x_i \trianglelefteq x_j$ *and* $x_i \neq x_j$; in this case, x_j is said to *strictly dominate* x_i. In practice, a finite poset (i.e., a poset defined on a finite set) is completely defined when one specifies, for each pair of different elements x_i and x_j, whether $x_i \trianglelefteq x_j$, or $x_j \trianglelefteq x_i$, or $x_i || x_j$. In the following, we will consider only finite posets and denote their cardinality, that is, the cardinality of the underlying set X, by n. A poset π where any two elements are comparable is called a *linear* (or *complete*, or *total*) *order*, or a *chain*. On the contrary, if any two elements of π are incomparable, then π is called an *antichain*. An element $x \in \pi$ that does not dominate any other element is called *minimal*, while an element that is dominated by all of the other elements is called (the) *minimum*. *Maximal* elements and the *maximum* are defined dually. An element x_j is said to *cover* x_i (written $x_i \prec x_j$) if $x_i \triangleleft x_j$ and there is no other element $x_k \in \pi$ such that $x_i \triangleleft x_k \triangleleft x_j$. In a finite poset, the cover relation determines the partial order relation, since it can be easily proved that $x_i \triangleleft x_j$ if and only if there exists a collection of elements $x_{k_0}, x_{k_1}, \ldots, x_{k_s}$ in π such that $x_i = x_{k_0} \prec x_{k_1} \prec \ldots \prec x_{k_s} = x_j$. More generally, given a cover relation $\prec$ on a set X, the corresponding partial order relation $\trianglelefteq$ is obtained by "completing" $\prec$ with all of the comparabilities implied by transitivity. This defines $\trianglelefteq$ as the *transitive closure* of $\prec$. The graph associated with the cover relation is called the *Hasse diagram* of the corresponding poset (see Fig. 1).

Intersection of two posets. As for any binary relation, a partial order on set X is a subset of the Cartesian product X^2, that is, a list of pairs of elements of X; as such, it is meaningful to define the intersection of two posets $\pi_a = (X, \trianglelefteq_a)$ and $\pi_b = (X, \trianglelefteq_b)$, which turns out to be itself a partially ordered set, namely, the poset $\pi_\cap = (X, \trianglelefteq_\cap)$, where $x_i \trianglelefteq_\cap x_j$ if and only if $x_i \trianglelefteq_a x_j$ *and* $x_i \trianglelefteq_b x_j$ $(x_i, x_j \in X)$. The definition extends in the obvious way to the intersection of a finite family of posets.

Extensions of a poset. Given two posets $\pi_a = (X, \trianglelefteq_a)$ and $\pi_b = (X, \trianglelefteq_b)$ in Π_X, we say that π_b is a proper *extension* of π_a (written $\pi_a \subset \pi_b$) if $x_i \trianglelefteq_a x_j$ in π_a implies $x_i \trianglelefteq_b x_j$ in π_b and there exist elements x_p and x_q in X such that $x_p ||_a x_q$ *and* $x_p \triangleleft_b x_q$. In other words, π_b is an extension of π_a if it can be obtained from the latter by turning some incomparabilities into comparabilities. If π_b is an extension of π_a and, in addition, it is a linear order, then it is called a *linear extension* of π_a. In the finite case, it can be proved that the set $\Omega(\pi)$ of the linear extensions of π uniquely determines it, by intersection, that is, $\pi = \cap_{\ell \in \Omega(\pi)} \ell$ (Schröder, 2003). This property motivates the key role of linear extensions in applications of partial order theory to data analysis (e.g. Arcagni et al., 2019; Fattore et al., 2011; Fattore, 2016; Fattore and Arcagni, 2019; Iglesias et al., 2017), making it possible to reduce one complex partially ordered structure to many simple linear structures and to decompose the original problem into "more elementary pieces."

Matrix representations of finite posets. Finite posets can be conveniently represented using square matrices that convey, in different ways, essential information on their structure. Here, we focus on the so-called *incidence*, *cover*, and *mutual ranking probability* matrices, the last of which plays a key role in the approximation algorithms.

1. *The incidence matrix* $\boldsymbol{Z}$. The most natural algebraic representation of a finite poset π is the so-called Z matrix, whose entries are defined as $Z_{ij} = 1$ if $x_i \trianglelefteq x_j$ in π and $Z_{ij} = 0$ otherwise. The matrix Z is essentially the characteristic function of the partial order relation and simply lists the pairs of elements belonging to it, that is, the *dominances* of the poset.
2. *The cover matrix* $\boldsymbol{C}$. Since by transitivity the cover relation $\prec$ associated with $\trianglelefteq$ determines it, the binary matrix C, which is defined as $C_{ij} = 1$ if $x_i \prec x_j$ in π and $C_{ij} = 0$ otherwise, implicitly provides an equivalent representation of π.
3. *The matrix of mutual ranking probability* $\boldsymbol{P}$. Given $x_i, x_j \in \pi$, by the *mutual ranking probability* (MRP) of x_j with respect to x_i we mean the fraction of linear extensions of the input poset π, where x_j is ordered higher than x_i, that is, the probability of picking uniformly at random, out of $\Omega(\pi)$, a linear extension ℓ such that $x_i \trianglelefteq_\ell x_j$ (De Loof, 2009). By collecting all of these probabilities into a matrix, we get the MRP matrix P, whose entries are formally defined as

$$P_{ij} = \frac{|\{\ell \in \Omega(\pi) : x_i \trianglelefteq_\ell x_j\}|}{|\Omega(\pi)|} \quad (i, j = 1, \ldots, |X| = n). \tag{1}$$

   By construction, the diagonal of P is composed of 1s and $P_{ij} + P_{ji} = 1$ $(i \neq j)$. Notice that $P_{ij} = 1$ if and only if, in all of the linear extensions of π, x_j dominates x_i, and this, in turn, is true if and only if $Z_{ij} = 1$; this shows that the MRP matrix determines the Z matrix and so uniquely identifies the input poset.

Distances and dissimilarity measures between finite posets. A crucial step, in view of the approximation algorithms, is to learn how to compare the structures of two or more posets, by defining some distance or dissimilarity measure between them. This task can be addressed in different ways, as briefly outlined below.

1. One can turn the set Π_X, considered as a poset itself under inclusion of binary relations, into a metric space (Monjardet, 1981; Foldes and Radeleczki, 2001) and then compare two posets π_1 and π_2 on X by computing their distance as elements of Π_X.
2. Alternatively, one can directly define some metric between posets, as in Zelinka (1993), where a distance between isomorphism classes of finite posets is introduced as a measure of their structural dissimilarity.
3. Owing to the bijection between finite posets and their matrix representations, a natural idea is to measure the distance between two partial order relations as the distance between their respective C, Z, or P matrices. Whereas C and Z are binary matrices, and classify pairs of elements just in terms of covering or dominance, the matrix P resolves the relation between incomparable elements in a much finer way and allows posets to be more effectively compared. Measuring poset dissimilarity through their MRP matrices requires that some matrix metric or norm be chosen. The most popular choices are the L_1 and the L_2 (or *Frobenius*) norms, respectively defined as

$$\|P\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{n}|P_{ij}|, \qquad \|P\|_2 = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}P_{ij}^2}. \tag{2}$$

   Given the probabilistic nature of their entries, MRP matrices can also be compared based on the *Jensen–Shannon* divergence, which is a symmetric measure of dissimilarity between probability distributions derived from the Kullback–Liebler divergence (see Section 7 for technical details and explicit formulas).

Among the three approaches to poset comparison, the "matrix" one is adopted here; in particular, we employ both the L_1 norm and the Jensen–Shannon divergence to construct the loss functions used to drive the linearization and bucketization algorithms.

5. The linearization algorithm

In this section, we describe and formalize the linearization algorithm for general multi-poset approximation and ranking extraction, providing a toy example to clarify its functioning. Starting from k input posets $\pi_1, \ldots, \pi_k \in \Pi_X$, the linearization algorithm produces a nested sequence of posets $\hat{\pi}_0 \subset \hat{\pi}_1 \subset \cdots \subset \hat{\pi}_m = \hat{\ell}$ in Π_X, each of which extends the previous one, until a linear order $\hat{\ell}$ is reached. At each step, $\hat{\pi}_{i+1}$ is obtained from $\hat{\pi}_i$ by searching for the poset that best approximates the k input posets jointly, among the extensions of $\hat{\pi}_i$, generated by adding just one edge to its Hasse diagram. This constructs a path between the starting point $\hat{\pi}_0$ and the final candidate linear order $\hat{\ell}$, passing through a sequence of progressively extended posets, the best-approximating one of which is picked up by minimization of a suitable loss function. To operationalize the algorithm, the initial poset $\hat{\pi}_0$ and the loss function used to drive the linearization process are specified as follows:

1. $\hat{\pi}_0$ is set equal to the intersection of the input posets, that is, $\hat{\pi}_0 = \pi_\cap = \pi_1 \cap \ldots \cap \pi_k$. Indeed, if the input posets share a set of dominances, any sensible approximating poset must share such dominances as well and so must be an extension of $\pi_\cap$. Notice that $\hat{\pi}_0 = \pi_\cap$ is not to be meant as an initial guess, but is simply needed to start the greedy search; being the smallest poset containing all the dominances shared by the inputs, the intersection is the *most informative* poset to start with and defines the space in which to perform the search (i.e., the subset of Π_X composed of extensions of $\hat{\pi}_0$).
2. At step i, the loss function $\mathcal{L}_i$ is obtained by averaging the relative distances between the generated $\hat{\pi}_i$ and the input posets $\pi_1, \ldots, \pi_k$. These are here computed based on the L_1 distances between the matrices of mutual ranking probabilities $\hat{P}_i$ and P_j of $\hat{\pi}_i$ and π_j $(j = 1, \ldots, k)$, respectively. Formally, the loss function at step i is defined as

$$\mathcal{L}_i = \frac{1}{k} \sum_{j=1}^{k} \frac{\|\hat{P}_i - P_j\|_1}{\|P_j\|_1}. \tag{3}$$

A few remarks on the linearization algorithm are now in order.

1. Since, for any MRP matrix P, both the diagonal elements and the sums of elements P_{ij} and P_{ji} $(i \neq j)$ are equal to 1, the L_1 norm $\|P_j\|_1$ in the denominator of (3) equals $n(n+1)/2$ (n being the number of elements of π_j). Since all of the k input posets have the same number of elements, the denominator can be factored out, becoming just a scale coefficient. It has been inserted into (3) to explicitly normalize the loss with respect to the number of poset elements and to make it possible to compare the losses of linearization processes applied to different poset systems.
2. In addition to the value of the overall loss function, the quality of the outputs can be evaluated by comparing the MRP matrix of the selected final poset with each of the MRP matrices of the k input posets, by columns, by rows, or entry-wise. In this way, one can control for the distribution and the types of "errors" in the final approximation. Notice that comparing MRP matrices by columns means comparing the degrees of dominance of each statistical unit over the others; comparing them by rows means comparing the degrees by which each unit is dominated by the others. These two points of view are equally valuable, but they are not mathematically equivalent.
3. In the case of a single input poset, the algorithm reduces to a tool for optimal linear approximation or, otherwise stated, for optimal ranking extraction, as an alternative to the use of average heights (Brüggemann and Patil, 2011) or SVD-dominance scores (Fattore and Arcagni, 2020).
4. The linearization algorithm, as introduced above, requires the input posets to share the same ground set X. In some applications, however, it may well be that they share just a subset X_s of X, for example in the case of missing data in some of the inputs. In this case, the algorithm can be adapted and modified along the following lines:
   - Consistently with the concept that the dominances in the initial poset must be shared by all of the inputs, the poset $\hat{\pi}_0$ is constructed restricting $\pi_\cap$ to X_s and declaring each element in the complement X_s^c of X_s in X as incomparable with all of the other elements in X (in practice, the elements of X_s^c are *isolated* in the Hasse diagram of the initial poset $\hat{\pi}_0$).
   - At step i of the linearization process, the comparison between the MRP matrix $\hat{P}_i$ of $\hat{\pi}_i$ and the MRP matrix P_j of input π_j $(j = 1, \ldots, k)$ is restricted to the elements shared by the two posets.
   - The corresponding loss function $\mathcal{L}_i$ is constructed as an average of the L_1 distances to the k input MRP matrices, weighted based on the fraction of elements shared by each input poset π_j $(j = 1, \ldots, k)$ and the generated poset $\hat{\pi}_i$.

5.1. Algorithm formalization

After introducing the linearization algorithm in an informal way, we now provide its pseudocode description.

A – Computation of input incidence and MRP matrices

for all input posets π_j, $j = 1, \ldots, k$ **do**
    $Z_j \leftarrow$ incidence matrix of π_j
    $P_j \leftarrow$ MRP matrix of π_j
end for

B – Initialization

$i \leftarrow 0$
$\hat{Z}_0 \leftarrow Z_1 \circ \cdots \circ Z_k$ (computation of the incidence matrix of $\hat{\pi}_0 = \pi_1 \cap \cdots \cap \pi_k$; the operator $\circ$ stands for entry-wise matrix multiplication)
$m \leftarrow$ number of rows and columns of $\hat{Z}_0$

C – Main loop

repeat
    $i \leftarrow i + 1$

step i: search

for all $s, t \in 1 \dots m, s \neq t$ **do**
  if $\hat{Z}_{st} = \hat{Z}_{ts} = 0$ **then**
    $\hat{Z}^{(s,t)} \leftarrow \hat{Z}_{i-1}$
    $\hat{Z}^{(s,t)}_{st} \leftarrow 1$
    $\hat{Z}^{(s,t)}_{*} \leftarrow$ transitive closure of $\hat{Z}^{(s,t)}$.
    $\hat{P}^{(s,t)} \leftarrow$ MRP matrix of the poset represented by $\hat{Z}^{(s,t)}_{*}$
    $\mathcal{L}^{(s,t)} \leftarrow \frac{1}{k} \sum_{j=1}^{k} \frac{\|\hat{P}^{(s,t)} - P_j\|}{\|P_j\|}$
  end if
end for
$(u, v) \leftarrow \underset{s,t}{\operatorname{argmin}}(\mathcal{L}^{(s,t)})$

step i: poset extension

$\hat{Z}_i \leftarrow \hat{Z}^{(u,v)}_{*}$
$\mathcal{L}_i \leftarrow \mathcal{L}^{(u,v)}$

conclusion

until $\hat{Z}_i$ is the incidence matrix of a complete order, i.e., no incomparabilities remain

5.2. Toy example

We now exemplify the linearization procedure on a very simple toy poset system. The input posets π_1, π_2 and π_3 are depicted in Fig. 2, and their cover, incidence, and MRP matrices are as follows:

$C_1 =$

	A	B	C	D	E
A	0	0	0	0	0
B	0	0	1	0	0
C	1	0	0	0	0
D	1	0	0	0	0
E	0	0	0	1	0

$C_2 =$

	A	B	C	D	E
A	0	0	1	0	0
B	1	0	0	0	0
C	0	0	0	1	0
D	0	0	0	0	0
E	0	0	0	1	0

$C_3 =$

	A	B	C	D	E
A	0	0	0	1	0
B	0	0	1	0	0
C	1	0	0	0	0
D	0	0	0	0	0
E	0	0	1	0	0

$Z_1 =$

	A	B	C	D	E
A	1	0	0	0	0
B	1	1	1	0	0
C	1	0	1	0	0
D	1	0	0	1	0
E	1	0	0	1	1

$Z_2 =$

	A	B	C	D	E
A	1	0	1	1	0
B	1	1	1	1	0
C	0	0	1	1	0
D	0	0	0	1	0
E	0	0	0	1	1

$Z_3 =$

	A	B	C	D	E
A	1	0	0	1	0
B	1	1	1	1	0
C	1	0	1	1	0
D	0	0	0	1	0
E	1	0	1	1	1

$P_1 =$

	A	B	C	D	E
A	1.00	0.00	0.00	0.00	0.00
B	1.00	1.00	1.00	0.83	0.50
C	1.00	0.00	1.00	0.50	0.17
D	1.00	0.17	0.50	1.00	0.00
E	1.00	0.50	0.83	1.00	1.00

$P_2 =$

	A	B	C	D	E
A	1.00	0.00	1.00	1.00	0.50
B	1.00	1.00	1.00	1.00	0.75
C	0.00	0.00	1.00	1.00	0.25
D	0.00	0.00	0.00	1.00	0.00
E	0.50	0.25	0.75	1.00	1.00

$P_3 =$

	A	B	C	D	E
A	1.00	0.00	0.00	1.00	0.00
B	1.00	1.00	1.00	1.00	0.50
C	1.00	0.00	1.00	1.00	0.00
D	0.00	0.00	0.00	1.00	0.00
E	1.00	0.50	1.00	1.00	1.00

.

The procedure is initialized by taking the intersection of the input posets $\hat{\pi}_0 = \pi_1 \cap \pi_2 \cap \pi_3$ (see Fig. 2). In step 1, one of the seven incomparabilities $A||C$, $A||D$, $A||E$, $B||D$, $B||E$, $C||D$, $C||E$ must be turned into comparability. From the loss values reported in Table 2, the best choice is to add to $\hat{\pi}_0$ the dominance $C \lhd A$, giving the poset $\hat{\pi}_1$ depicted in Fig. 3. Notice that no other dominances are induced by transitivity and that $\mathcal{L}_1$ (0.2267) is lower than $\mathcal{L}_0$ (0.2444), that is, the first extension produces a better approximation to the input posets than their intersection. To get to the final linear order, just four more iterations are needed, since the third iteration, introducing the dominance $A \lhd D$, induces the dominances $B \lhd D$ and $C \lhd D$, by transitivity. The sequence of generated posets is depicted in Fig. 3, and Fig. 4 depicts a plot of the loss function and of

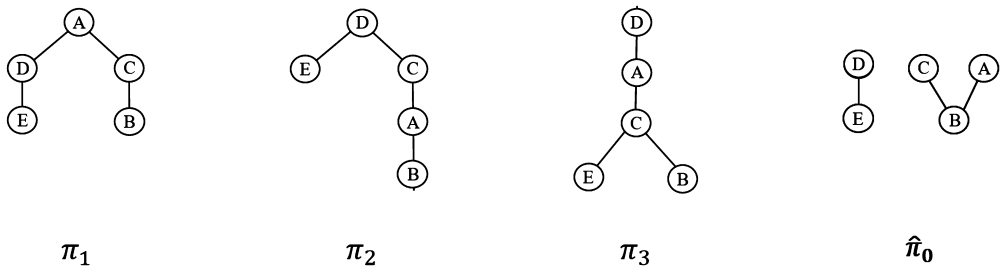

Fig. 2. Toy example: Hasse diagrams of the input posets and of their intersection $\hat{\pi}_0 = \pi_1 \cap \pi_2 \cap \pi_3$.

Table 2
Toy example: loss values for different added covers, at step 1.

$\lhd$	A	B	C	D	E
A			0.281	0.236	0.378
B				0.240	0.289
C	**0.227**			0.232	0.356
D	0.391	0.511	0.346		
E	0.237	0.345	0.259		

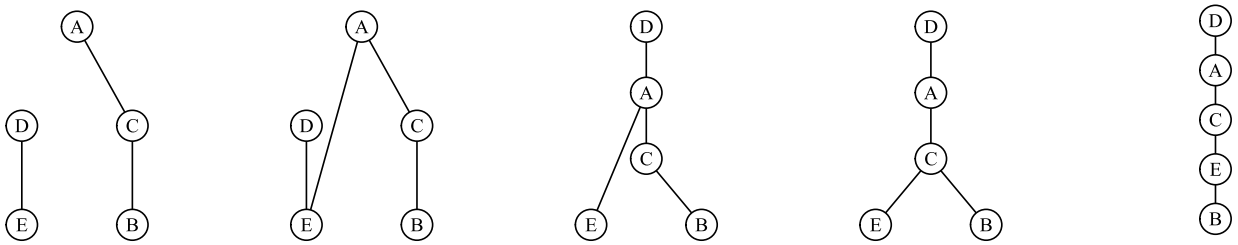

Fig. 3. From left to right, the posets $\hat{\pi}_1$, $\hat{\pi}_2$, $\hat{\pi}_3$, $\hat{\pi}_4$, and $\hat{\pi}_5$ generated by the linearization algorithm, with corresponding loss values $\mathcal{L}_1 = 0.2267$, $\mathcal{L}_2 = 0.2173$, $\mathcal{L}_3 = 0.1852$, $\mathcal{L}_4 = 0.1704$, and $\mathcal{L}_5 = 0.2148$.

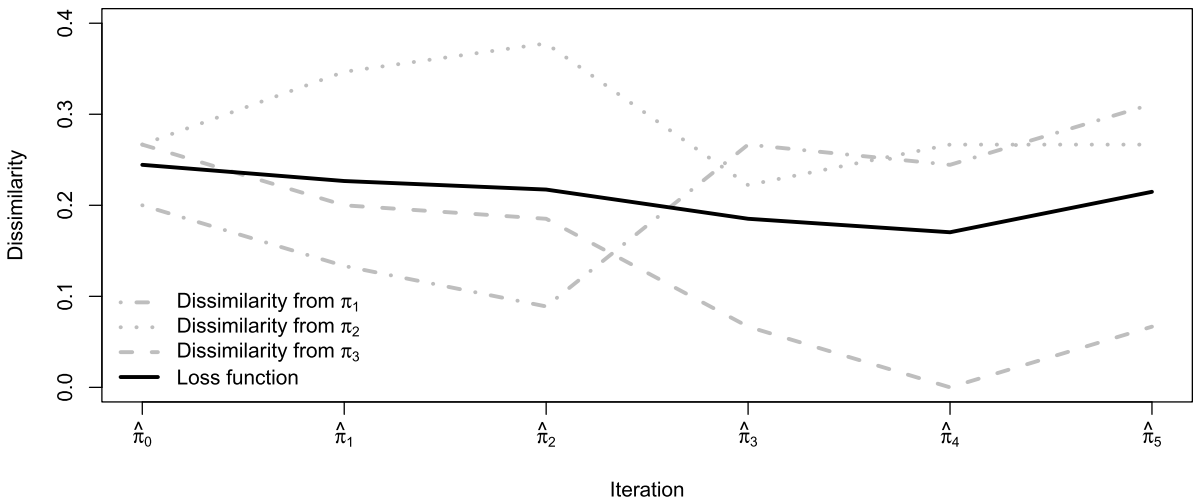

Fig. 4. Loss function, and relative dissimilarities from the input posets, for the toy example.

the normalized distances to the single input posets. Notice that π_2 is the worst approximated poset all along the entire process and that, in the final iterations, a better approximation to π_3 corresponds to a worse approximation to the other two posets. The loss function is minimized by $\hat{\pi}_4$ (0.1704), and getting to the final linear order increases it over the previous iteration (+0.0444, i.e., +26%).

Table 3
Basic structural indicators for the initial (intersection) poset, the optimal poset, and the final linear order of the subjective well-being example and corresponding dissimilarities from the input posets ("Inc": number of incomparabilities; "Comp": number of strict comparabilities).

Poset	Inc	Comp	Inc/Comp	Diss. from (a)	Diss. from (b)	Diss. from (c)
Initial poset	102	88	1.159	0.363	0.184	0.144
Optimal poset	70	120	0.583	0.235	0.165	0.174
Final linear order	0	190	0	0.121	0.342	0.332

6. Subjective well-being of Italian regions

To show the linearization algorithm in action on a more interesting case, we apply it in this section to the motivating example on regional subjective well-being, that is, to the three posets depicted in Fig. 1. The procedure is initialized by the intersection poset represented in Fig. 5 and requires 95 iterations to get to the final linear order. The loss function, plotted in Fig. 6, attains its minimum (0.1915) at iteration 25; the corresponding optimal[1] approximating poset $\hat{\pi}_{25}$ is represented in Fig. 7. Table 3 provides some basic structural indicators, showing the effect of the linearization process, in terms of added comparabilities, and reporting the dissimilarities with each of the input posets. The optimal poset, whose incidence matrix is shown in Table 4, has a top region (Trentino – Alto Adige, which is the top of each input poset), but no bottom region; there are instead many minimal regions, namely Apulia, Basilicata, Calabria, Campania, Molise, Sardinia, and Sicily, all belonging to the South of Italy. The final ranking is reported in Table 5, with a loss equal to 0.2648 (38% higher than the minimum). Fig. 8 shows the decomposition of the loss function by regions, that is, the degree of approximation to the input vectors of mutual ranking probabilities of the single regions, as iterations proceed, and Table 6 provides the same information at the optimal poset and the final ranking. Notice that Trentino – Alto Adige does not contribute to the loss function, being the top of the three input posets and thus of their intersection and of each poset generated by the procedure; notice also that the contribution of Abruzzo increases rapidly after iteration 53, when the dominance LAZ $\lhd$ ABR is inserted. The above computations took 173.168 seconds on an iMac Pro 1.1, with an eight-core Intel Xeon W processor, 3.2 GHz, 32 GB RAM, and macOS 11.6 (mutual ranking probabilities have been computed exactly, using the lattice of ideals approach mentioned later, in Section 9).

Remark – Solution stability. The selection of the "minimum dissimilarity" poset as the best (greedy) approximation/synthesis to the input partial orders raises a natural question as to the stability of the solution around the minimum loss. Indeed, if structurally dissimilar posets achieved similar losses, then the minimization criterion would lead to a somewhat artificial choice, different partial orders being essentially "equivalent" in terms of approximation power. To check for the stability of the solution, the following two numerical/graphical tools can then be employed:

1. Compute and plot the ratio I/C between the number of incomparabilities and the number of strict comparabilities of each poset in the linearization chain. I/C equals its minimum, namely, 0, in linear orders (which have no incomparabilities) and grows as the poset approaches an antichain (at which one puts $I/C = +\infty$). Since, in the linearization algorithm, each poset of the sequence extends the preceding one, that is, dominances can only be added, I/C is strictly decreasing, and structural changes will be revealed by "big jumps" of the indicator at some points of the linearization process.
2. Using the formula for the loss function (restricted to $k = 1$), compute and plot the dissimilarity D between the MRP matrix of the optimal poset and those of the other posets in the linearization sequence; again, "big jumps" along the path will reveal global structural changes between the posets.

Fig. 9 depicts the shapes of the above measures for the well-being example. As can be seen, I/C decreases smoothly, and the variations of D around the optimum, where trivially $D = 0$, are very small (notwithstanding the downward peak, visible in the figure), showing that the posets around the optimum have negligible structural differences.

7. Entropy-based loss functions

In the basic implementation of the linearization algorithm, introduced in the preceding paragraphs, the loss function was constructed on the basis of the simple L_1 distance between MRP matrices. Since the entries of such matrices are probabilities, an appropriate alternative is to base the loss function upon the *Jensen–Shannon (JS) divergence*, a measure of similarity specifically designed to compare probability distributions. As we shall see below, this not only introduces a different approximation criterion, but also makes it possible to introduce, in a natural way, a weighting system into the loss function, to account for the different "informative powers" of the input posets.

[1] To be meant in a greedy sense, since we are not optimizing on the set of all posets defined on Italian regions.

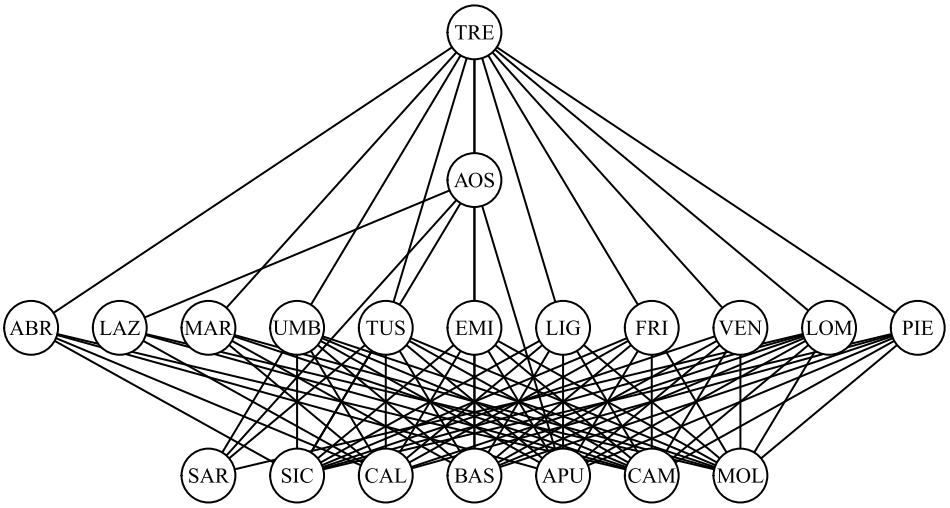

Fig. 5. Initialization poset for the well-being example (region labels are given in Table 5).

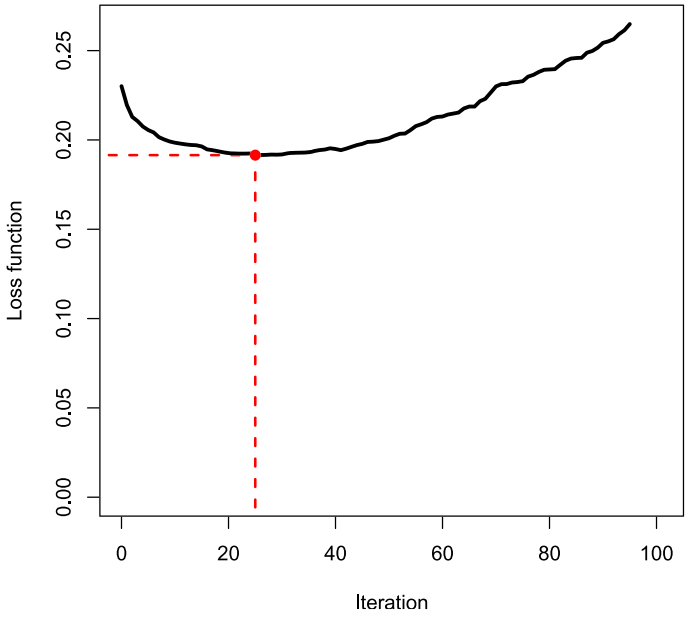

Fig. 6. $\mathcal{L}$-loss function for the regional well-being example (with the minimum being attained at iteration 25).

The unweighted JS-loss function. Given two probability distributions $\boldsymbol{p}_1(y)$ and $\boldsymbol{p}_2(y)$ on the same finite alphabet A, the *Jensen–Shannon divergence* (Lin, 1991) is defined as

$$JS(\boldsymbol{p}_1, \boldsymbol{p}_2) = \tfrac{1}{2}[D_{KL}(\boldsymbol{p}_1||\boldsymbol{q}) + D_{KL}(\boldsymbol{p}_2||\boldsymbol{q})], \tag{4}$$

where $\boldsymbol{q} = \frac{1}{2}(\boldsymbol{p}_1 + \boldsymbol{p}_2)$ and $D_{KL}(\cdot||\cdot)$ is the Kullback–Liebler divergence:

$$D_{KL}(\boldsymbol{p}_{1,2}||\boldsymbol{q}) = \sum_{y \in A} \boldsymbol{p}_{1,2}(y) \log_2 \frac{\boldsymbol{q}(y)}{\boldsymbol{p}_{1,2}(y)}. \tag{5}$$

The Jensen–Shannon divergence is bounded between 0 and 1, and its square root provides a metric between probability distributions. From it, we can derive a dissimilarity measure between MRP matrices involved in the linearization process, by observing that any MRP matrix P can be seen as the collection of $n(n-1)/2$ Bernoulli distributions B_{rs}, each corresponding to a pair of nondiagonal entries (P_{rs}, P_{sr}) (recall that, for $r \neq s$, $P_{sr} = 1 - P_{rs}$ and that P_{rs} is the probability of picking up, uniformly at random, a linear extension of the underlying poset, such that x_s dominates x_r). The MRP matrix $\hat{P}_i$ associated

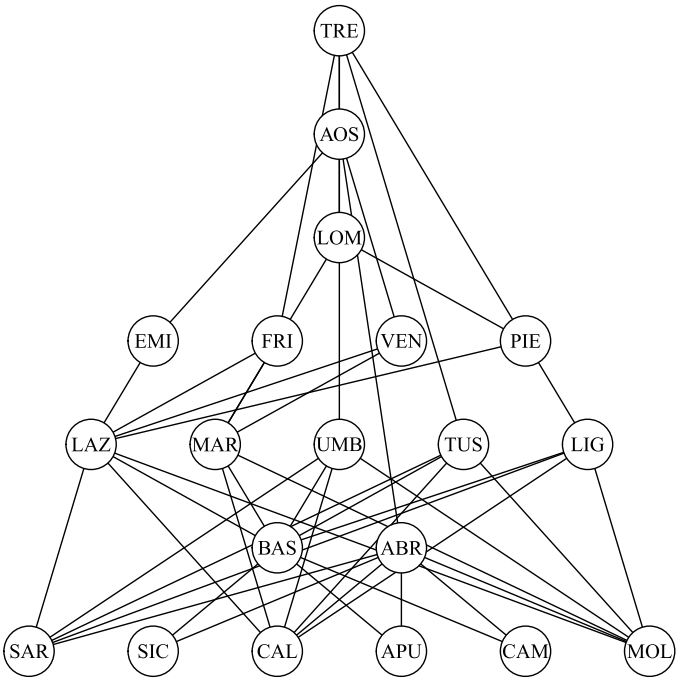

Fig. 7. Optimal poset $\hat{\pi}_{25}$ for the regional well-being example (region labels are given in Table 5).

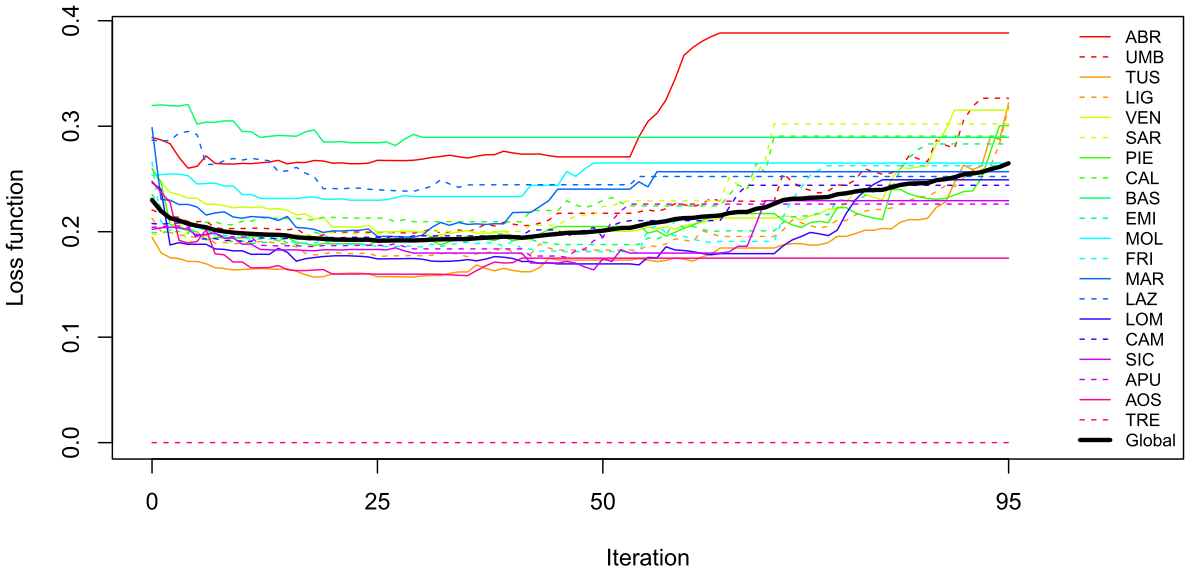

Fig. 8. $\mathcal{L}$-loss function for the regional well-being example: decomposition by regions (region labels are given in Table 5). The global loss function of Fig. 6 is shown in black.

with the ith step in the linearization algorithm can then be compared with the MRP matrix P_j of the jth input poset, by computing and averaging the Jensen–Shannon divergences of the corresponding Bernoulli distributions. Formally, we define the dissimilarity between $\hat{P}_i$ and P_j by

$$\Delta_{ij}^{JS} = \frac{2}{n(n-1)} \sum_{r=1}^{n} \sum_{1<s<r} JS(B_{rs}^{(j)}, \hat{B}_{rs}^{(i)}) \tag{6}$$

and the induced unweighted global loss function by

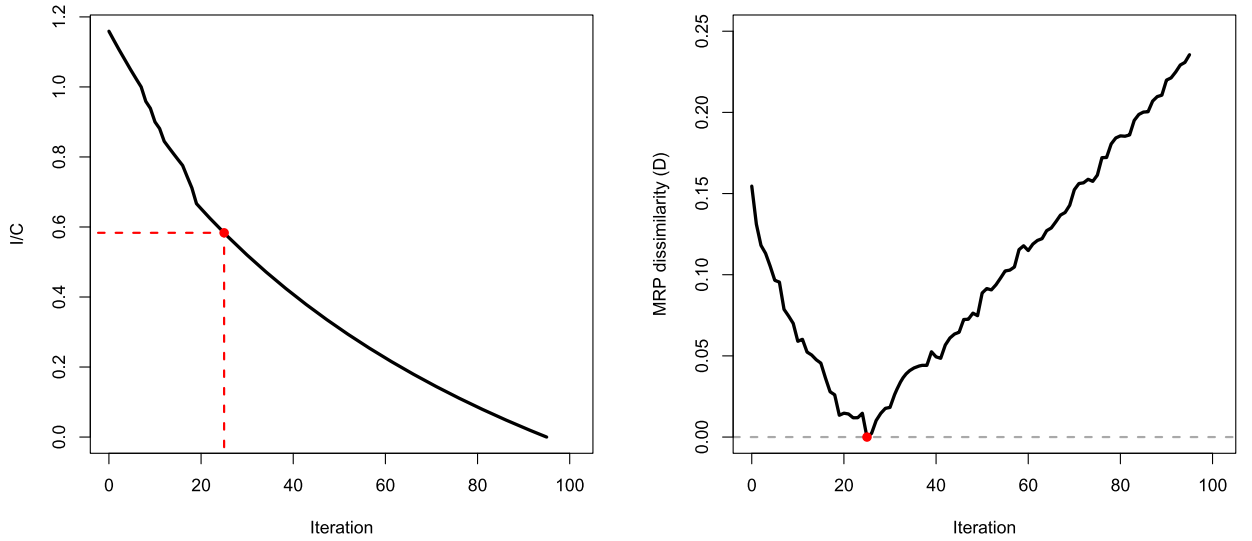

Fig. 9. Shape of ratio I/C (left panel) and dissimilarity D (right panel) along the linearization process for the well-being data and the $\mathcal{L}$ loss function.

Table 4
Incidence matrix of the optimal poset for the regional well-being example with $\mathcal{L}$-loss function (region labels are given in Table 5).

	TRE	AOS	FRI	LOM	EMI	VEN	TUS	LIG	PIE	UMB	ABR	MAR	LAZ	BAS	MOL	APU	SAR	CAL	SIC	CAM
TRE	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AOS	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FRI	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LOM	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EMI	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VEN	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TUS	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
LIG	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
PIE	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
UMB	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
ABR	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
MAR	1	1	1	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
LAZ	1	1	1	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
BAS	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	0	0	0	0
MOL	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0
APU	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0
SAR	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	0
CAL	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	0
SIC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0
CAM	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1

Table 5
Final ranking for the regional well-being example, with $\mathcal{L}$-loss function.

1	TRE	Trentino – Alto Adige	11	ABR	Abruzzo
2	AOS	Aosta Valley	12	MAR	Marche
3	FRI	Friuli Venezia Giulia	13	LAZ	Lazio
4	LOM	Lombardy	14	BAS	Basilicata
5	EMI	Emilia – Romagna	15	MOL	Molise
6	VEN	Veneto	16	APU	Apulia
7	TUS	Tuscany	17	SAR	Sardinia
8	LIG	Liguria	18	CAL	Calabria
9	PIE	Piedmont	19	SIC	Sicily
10	UMB	Umbria	20	CAM	Campania

Table 6
$\mathcal{L}$-loss function contributions for the regional well-being example: decomposition by regions at the optimal poset and the final linear order (region labels are given in Table 5).

Region	Optimal poset	Linear order	Region	Optimal poset	Linear order
TRE	0.000	0.000	ABR	0.268	0.388
AOS	0.160	0.175	MAR	0.195	0.257
FRI	0.186	0.263	LAZ	0.240	0.252
LOM	0.174	0.249	BAS	0.284	0.289
EMI	0.193	0.283	MOL	0.230	0.265
VEN	0.199	0.315	APU	0.187	0.226
TUS	0.157	0.322	SAR	0.198	0.302
LIG	0.177	0.318	CAL	0.213	0.291
PIE	0.192	0.301	SIC	0.183	0.229
UMB	0.199	0.327	CAM	0.195	0.244

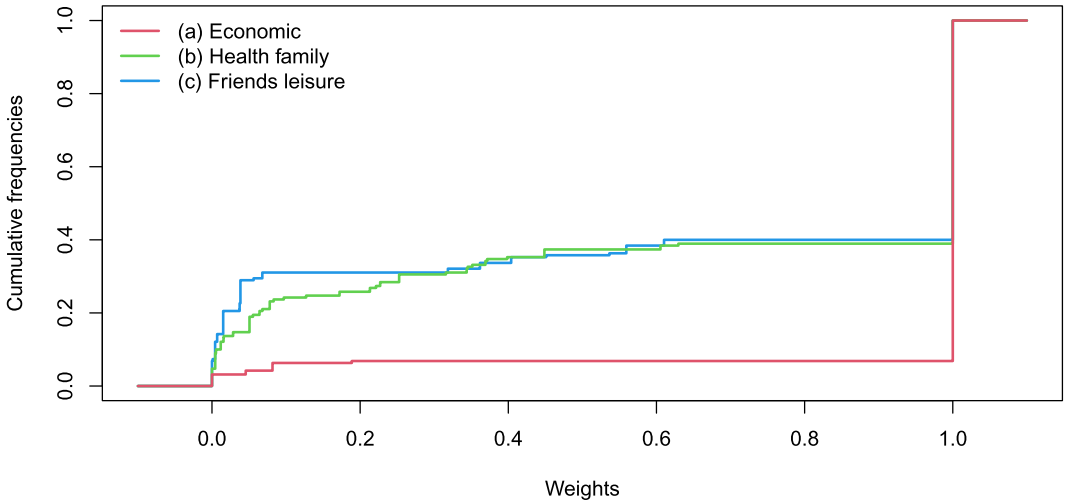

Fig. 10. Cumulative distributions of $1 - H_{rs}^{(j)}$ $(r \neq s,\ j = a, b, c)$, over all element pairs, for the three well-being posets.

$$\mathcal{L}_i^{JS} = \frac{1}{k}\sum_{j=1}^{k} \Delta_{ij}^{JS} = \frac{2}{kn(n-1)} \sum_{j=1}^{k}\sum_{r=1}^{n}\sum_{1<s<r} JS(B_{rs}^{(j)}, \hat{B}_{rs}^{(i)}), \tag{7}$$

which is just the average over the k input posets of the average dissimilarity of $\hat{P}_i$ from each of them.

The weighted JS-loss functions. In the L_1- and JS-loss functions, each input poset enters into the average with the same weight. However, referring to the well-being example, it may be observed that the three posets provide different amounts of information on the ordering of Italian regions, as is clear from Fig. 1 and the structural indices presented in Section 1. Poset (a), which is "almost linear," is indeed less ambiguous with regard to region ordering than posets (b) and (c), which contain fewer comparabilities and wider antichains. This leads quite naturally to the idea of weighting the loss function with the *informative power* of the input posets, to more efficiently allocate the approximation costs.

Formally, the amount of information conveyed by the input poset π_j with regard to the mutual ranking of the elements x_r and x_s is here measured by a decreasing function of the entropy $H_{rs}^{(j)}$ of the associated Bernoulli distribution $B_{rs}^{(j)}$, namely, by $1 - H_{rs}^{(j)}$, where

$$H_{rs}^{(j)} = -[P_{j(rs)} \log_2 P_{j(rs)} + (1 - P_{j(rs)}) \log_2 (1 - P_{j(rs)})] \tag{8}$$

(notice that $0 \leq H_{rs}^{(j)} \leq 1$). Fig. 10 compares the cumulative distributions of $1 - H_{rs}^{(j)}$ $(j = a, b, c)$, for the three well-being posets, showing quantitatively how poset (a) is definitely more informative than posets (b) and (c), as is also confirmed by the corresponding average values $1 - \overline{H}^a = 0.935$, $1 - \overline{H}^b = 0.667$, and $1 - \overline{H}^c = 0.648$, with

$$\overline{H}^{(j)} = \frac{2}{n(n-1)} \sum_{r=1}^{n}\sum_{1<s<r} H_{rs}^{(j)}. \tag{9}$$

We are thus led to weight $\mathcal{L}_i^{JS}$ in two alternative ways:

Table 7
Linearization algorithm with JS-loss functions: basic results ("Inc": number of incomparabilities; "Comp": number of strict comparabilities). Computation times (seconds): 495.831 (JS), 305.894 (JS1), and 513.553 (JS2). Mutual ranking probabilities have been computed "exactly," as in the L_1 case.

Loss function	Iterations	Initial loss	Minimum loss	Final loss	Inc	Comp	Inc/Comp
JS	101	0.064	0.052	0.102	88	102	0.863
JS1	68	0.073	0.035	0.045	34	156	0.218
JS2	101	0.071	0.055	0.093	76	114	0.667

(i) by weighting the divergences between each pair of distributions $B_{rs}^{(j)}$ and $\hat{B}_{rs}^{(i)}$ with the corresponding factor $1 - H_{rs}^{(j)}$:

$$\mathcal{L}_i^{JS1} = \frac{\sum_{j=1}^{k} \sum_{r=1}^{n} \sum_{1<s<r} JS(B_{rs}^{(j)}, \hat{B}_{rs}^{(i)})(1 - H_{rs}^{(j)})}{\sum_{j=1}^{k} \sum_{1<s<r} \sum_{r=1}^{n} (1 - H_{rs}^{(j)})}; \tag{10}$$

(ii) by weighting each input poset "globally" with the average $1 - \overline{H}^{(j)}$:

$$\mathcal{L}_i^{JS2} = \frac{2}{n(n-1)} \frac{\sum_{j=1}^{k} (1 - \overline{H}^{(j)}) \sum_{r=1}^{n} \sum_{1<s<r} JS(B_{rs}^{(j)}, \hat{B}_{rs}^{(i)})}{\sum_{j=1}^{k} (1 - \overline{H}^{(j)})}. \tag{11}$$

Table 7 and Figs. 11 and 12 report the results of the application of the linearization algorithm, with the unweighted and weighted JS-loss functions, to the subjective well-being data. We can observe the following:

1. The loss functions of the three linearization sequences have similar "U" shapes, but the number of iterations in the JS1 case is much less than with the JS and JS2 losses; noticeably, only with JS1 is the loss at the final order lower than that at the starting intersection poset.
2. The shapes of the I/C ratio and of the structural dissimilarity measure D reveal that for each of the JS-losses, the generated posets are quite stable around their respective optima. Interestingly, however, the shape of D for the JS1-loss is quite different from the others. In the JS1 case, the structure of the generated posets changes "quickly" until the optimal poset is reached, and then it stabilizes. Looking at the shape of the loss function and at the Hasse diagram of the optimal poset reveals that the JS1-loss drives the algorithm toward an "almost linear" optimal solution and, in the last part of the sequence, tunes it to the final linear order, just slightly modifying the order structure at the minimum loss.
3. All of the optimal posets reflect the North–South (and, partly, East–West) Italian polarization, placing Northern regions on the upper and Southern regions on the lower levels of the optimal Hasse diagrams. The "degree of linearity" of the diagrams is, however, different, and it is higher for the weighted cases. In particular, as already noticed, a neat leap toward linearity is achieved with the JS1-loss function; so, even if the three synthetic posets share some basic features, they reproduce different comparability/incomparability patterns, providing different pictures of the diversification of subjective well-being across Italy.
4. The final orders (Table 9) produced by the three loss functions are quite similar (and are also quite similar to the L_1-based ranking reported in Table 5), the only differences being in the relative positions of some Southern regions, which are nevertheless invariably ranked in the lower part of the linear extensions, consistently with the historical North–South polarization of Italy.

The effect of the different weighting systems on the linearization process is clarified in Table 8, which reports the dissimilarities of the optimal posets and the final linear orders from each of the three input posets for the different Jensen–Shannon losses. The absolute loss values cannot be compared directly, being computed using different formulas; however, the figures show that when JS1 is employed, the optimal poset is nearly "equidistant" from the input ones, and also the final linear order has more balanced dissimilarities from the inputs. Given the complex structure of the loss functions, it is not possible to provide an analytical interpretation of these results, but some interesting hints can nevertheless be obtained. In the JS1 case, at each step, the optimization criterion selects the incomparable pair to be ordered, based on weights that depend upon the information power of each input poset *for that pair*. This establishes a sort of competition among the inputs for the identification of the cover to add. From time to time, a different poset may "win" such a competition and be more determinant than the others in the selection of the cover. At different steps, the algorithm may extract information from different input posets; as a consequence, at least on the well-being data, the optimal and the final linear order emerge by mixing the information contributions (i.e., the MRP structures) of the input posets, in a more or less balanced way. In the JS2 case, the weights attached to pairs are constant within each input poset, and so globally more informative posets attach larger weights even to pairs on which they are scarcely informative. As the linearization process goes on, increasingly "more incomparable" pairs become ordered and, as the final linear order is approached, the higher global weight of poset (a) blends the sequence toward it, to minimize the optimization costs. A similar effect occurs in the unweighted case, where the intrinsic higher information power of this poset is sufficient to attract the linearization sequence to it.

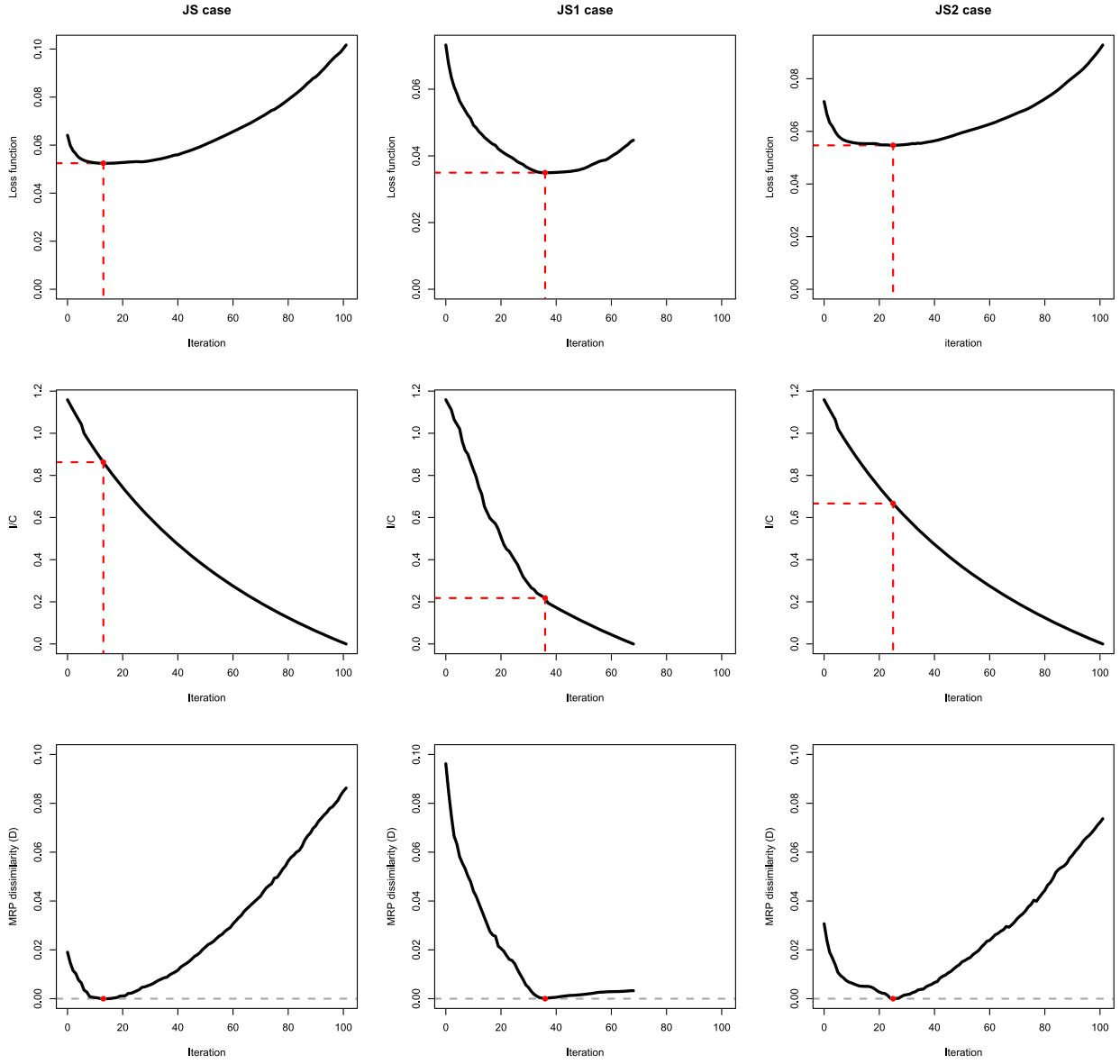

Fig. 11. Upper panels: Jensen–Shannon loss functions for the well-being example (notice that the number of iterations to achieve the final linear order is different for the different loss functions: 101 for JS and JS2 and 68 for JS1); middle and lower panels: I/C and structural difference D as functions of the algorithm iterations (notice that the dissimilarity measures have been computed using the same loss functions employed in the linearization processes, namely, the JS, JS1, and JS2 formulas, respectively).

All in all, it is not possible to state whether one JS-loss function is "better" than the others and than the $\mathcal{L}$ (i.e., L_1-based) loss. However, the results above and the previous discussion suggest that the JS1 formula, weighting "locally" the Jensen–Shannon divergences, is more effective in exploiting the information power on dominances contained in the inputs, thereby shortening the linearization process and delivering "more linear" optima. Notice, however, that in some circumstances, we could also be interested in reproducing incomparabilities, considered as structural features of the phenomenon/concept under investigation, rather than as being due to a lack of information on dominance.

8. Bucket orders

According to the preceding sections, the best greedy approximating poset is only required to be an extension of the intersection of the inputs, and no other constraint is imposed on its structure. If required, the linearization algorithm can also deliver a final ranking, which is often useful for prioritization purposes or for communication goals. Admittedly, however, compressing the input posets into a single linear order can be somewhat artificial and misleading, since linearity forces dominance between units, even when these are essentially tied. When the interest leans toward ranking, a more realistic

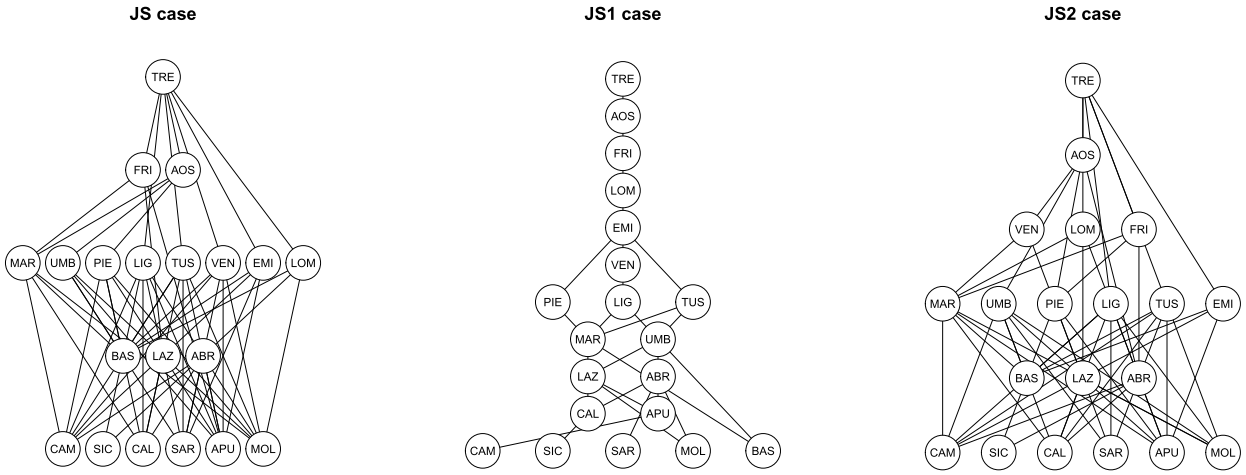

Fig. 12. Optimal posets for the well-being example, with Jensen–Shannon loss functions.

Table 8
Dissimilarity from the input well-being posets, for the optimal posets and the final linear orders, with different weighted and unweighted JS-loss functions.

Poset	Diss. from (a)	Diss. from (b)	Diss. from (c)
Optimal poset (JS)	0.079	0.037	0.042
Final linear order (JS)	0.045	0.139	0.121
Optimal poset (JS1)	0.038	0.033	0.033
Final linear order (JS1)	0.051	0.059	0.024
Optimal poset (JS2)	0.066	0.043	0.051
Final linear order (JS2)	0.029	0.139	0.137

Table 9
Well-being example: final rankings according to JS-, JS1-, and JS2-loss functions (region labels are given in Table 5).

	JS	JS1	JS2
1	TRE	TRE	TRE
2	AOS	AOS	AOS
3	FRI	FRI	FRI
4	LOM	LOM	LOM
5	EMI	EMI	EMI
6	VEN	VEN	VEN
7	TUS	TUS	TUS
8	LIG	LIG	LIG
9	PIE	PIE	PIE
10	UMB	UMB	UMB
11	MAR	MAR	MAR
12	LAZ	LAZ	LAZ
13	ABR	ABR	ABR
14	BAS	CAL	BAS
15	MOL	BAS	MOL
16	APU	APU	APU
17	SAR	SAR	CAM
18	CAL	MOL	SAR
19	SIC	SIC	CAL
20	CAM	CAM	SIC

compromise would be to organize statistical units into equivalence classes and to order these, without imposing comparability of all the elements. This leads to restricting the search for an optimal approximating poset to so-called *bucket orders*,[2] a class of partial order relations that have already attracted some attention in the computational and applied literature on partially ordered data analysis, for example in connection with the seriation problem in paleontology (Kenkre et al., 2011). Below, we introduce a novel greedy "bucketization" algorithm, specifically designed for this purpose.

[2] We thank an anonymous referee for pointing out the relevance of bucket orders and for suggesting the development of an optimization algorithm for this kind of poset.

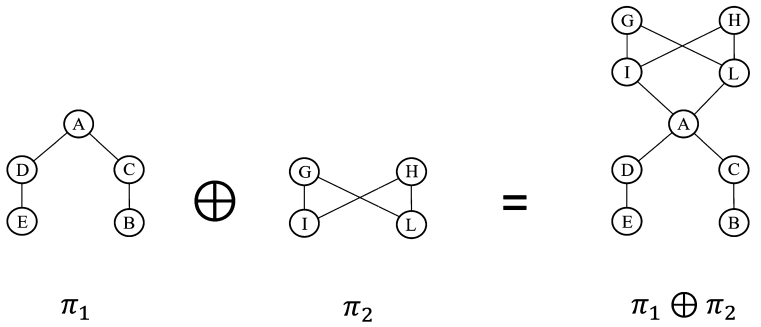

Fig. 13. A linear sum of two small posets.

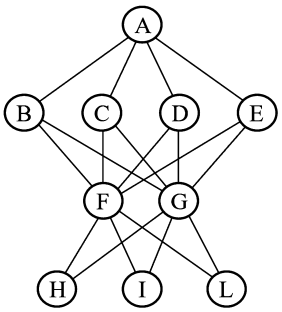

Fig. 14. A small bucket order with four buckets.

8.1. Definition and main properties of bucket orders

To introduce bucket orders, the definition of the *linear sum* of two posets is first needed. Let $\pi_1 = (X_1, \trianglelefteq_1)$ and $\pi_2 = (X_2, \trianglelefteq_2)$ be two posets over disjoint ground sets X_1 and X_2. Then the *linear sum* $\pi_1 \oplus \pi_2$ of π_1 and π_2 (Davey and Priestley, 2002) is defined to be the poset $\pi_{1\oplus 2} = (X_1 \cup X_2, \trianglelefteq_{1\oplus 2})$, where the partial order relation $\trianglelefteq_{1\oplus 2}$ is given by

$$u \trianglelefteq_{1\oplus 2} v \quad \text{if and only if} \quad (u \trianglelefteq_1 v) \text{ or } (u \trianglelefteq_2 v) \text{ or } (u \in X_1 \text{ and } v \in X_2). \tag{12}$$

In practice, the partial order relation $\trianglelefteq_{1\oplus 2}$ coincides with $\trianglelefteq_1$ and $\trianglelefteq_2$ on X_1 and X_2 respectively and orders any element of X_2 above any element of X_1; thus, in the finite case, the Hasse diagram of $\pi_{1\oplus 2}$ is simply obtained by putting the diagram of π_2 above that of π_1 and adding edges between all of the minimal elements of the former and all of the maximal elements of the latter, as exemplified in Fig. 13. The definition of linear sum extends in the natural way to more than two posets. A *bucket order* (BO) π_{BO} with h *buckets* is simply a linear sum of h posets, where the summands are antichains; formally,

$$\pi_{BO} = \bigoplus_{i=1}^{h} b_i, \tag{13}$$

with $b_1, \ldots, b_h$ antichains (see Fig. 14). Given their simple structure, bucket orders enjoy the following useful properties:

1. The number of linear extensions of a BO with h buckets is simply $n_1! \cdots n_h!$, where $n_1, \ldots, n_h$ are the cardinalities of the buckets $b_1, \ldots, b_h$.
2. By symmetry, the mutual ranking probability of two different elements u and v belonging to the same bucket is 0.5, and that of elements belonging to different buckets is either 1 or 0. The MRP matrix of a BO is thus trivial to obtain, leading to great computational simplifications in the development of optimization algorithms.

In the following, the class of bucket orders over the ground set X will be denoted by BO_X.

8.2. The "bucketization" algorithm

Given k posets on the same ground set X, we want to perform a greedy search for a bucket order β^*, best approximating the inputs under a given dissimilarity measure. As in the linearization procedure, we want β^* to extend the intersection $\pi_\cap$ of the k input posets but, differently from that case, we relax the requirement for the sequence of generated posets to be nested and allow for the possibility that, sometimes along the process, dominances are removed, making it possible

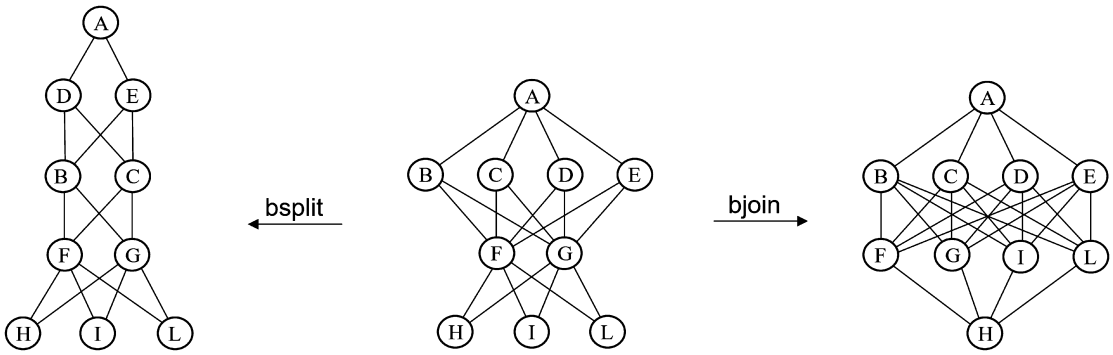

Fig. 15. Examples of *bsplit* and *bjoin* moves, on the bucket order of Fig. 14. On the left, bucket $\{B, C, D, E\}$ is split as $\{B, C\} \oplus \{D, E\}$; on the right, the subset $\{I, L\}$ of the bottom bucket $\{H, I, L\}$ is joined to the bucket $\{F, G\}$.

for the algorithm to explore the "bucket space" both "vertically" (i.e., along chains of progressively extended posets) and "horizontally" (i.e., producing posets that do not extend all the previous ones). Globally, however, we impose the algorithm to "move upward" by employing a "driving score" to select the moves to be performed, finally reaching a linear order (which is indeed a special case of *BO*). The most natural way to achieve all of this, is (i) to design an iterative procedure whose iterations preserve the dominances of the initial poset and are "bucketness-preserving", that is, generate a *BO* whenever the input is a *BO*, and (ii) to feed it with an initial bucket order β_0 extending $\pi_\cap$. The algorithm then outputs a sequence $\beta_0, \beta_1, \ldots, \beta_m = \ell$ of *BO*s, whose last element is a linear order. At each step, the chosen function loss is computed and the *BO* minimizing it within the sequence is, in the end, selected. To pursue this line, we first introduce two bucketness-preserving operators, namely, *bsplit* and *bjoin*, used to modify the structure of the bucket orders along the process, then explain how the driving scores are built, how they are employed in the splitting and joining transformations, and, finally, specify how the initial bucket order β_0 can be set. Some remarks and an example end this subsection.

A. The bsplit and bjoin operators. Let $\beta = b_1 \oplus b_2 \oplus \ldots \oplus b_h$ be a bucket order with h buckets. Consider any bucket b_j $(1 \leq j < h)$ and a partition $b_j = b_{j_1} \cup b_{j_2}$ into two disjoint and nonempty subsets. Then the action of *bsplit* and *bjoin* is described as follows (see Fig. 15):

1. *bsplit* turns b_j into $b_{j_1} \oplus b_{j_2}$, that is, it "explodes" an input bucket into the linear sum of two buckets;
2. *bjoin* removes b_{j_2} from b_j and adds it to the next bucket b_{j+1}, thus replacing $b_j \oplus b_{j+1}$ with $b_{j_1} \oplus (b_{j+1} \cup b_{j_2})$.

Clearly, *bsplit* increases the number of buckets of the poset by one and extends it, adding dominances between the new buckets b_{j_2} and b_{j_1}; on the contrary, *bjoin* leaves the number of buckets unchanged and does not extend the input poset, since while some dominances may be added (those between the elements of b_{j_2} and b_{j_1}), others are removed (those between the elements of b_{j+1} and b_{j_2}).

The *bsplit* and *bjoin* operations are repeatedly used in the algorithm, at each iteration searching for the "best" move to do. However, to reduce the computational burden, the "admissible" partitions $b_j = b_{j_1} \cup b_{j_2}$ are restricted to those where the elements of b_{j_2} are likely to dominate those of b_{j_1} in the optimal (unknown) bucket order. The identification of such partitions employs a *driving score*, attached to each element of the initial poset β_0 and computed as described below.

B. Driving scores. Informally speaking, in the unknown optimal bucket order, elements should be intuitively arranged in such a way that those "placed higher" in the input posets (and so that "tend to dominate the other elements") are put in higher buckets; as a consequence, informally speaking, it is scarcely useful for the algorithm to test splits and joins where "weakly dominating" elements are moved upward. To prevent the bucketization algorithm from performing these operations, we introduce a *global dominance score*, assessing the degree of dominance of each element in the ground set X, based on its "relational position" in the input posets and used to identify the most sensible moves, that is, those that move upward the most dominating elements. The global dominance score $s(x)$ of element $x \in X$ can be obtained in different ways, for example (i) by summing or averaging its so-called *average heights*[3] in each of the k input posets, or (ii) by doing the same, but replacing the average height with its *SVD-based score* (Fattore and Arcagni, 2020), or (iii) by directly taking its *height* in the final linear order delivered by the linearization algorithm (notice that the idea of driving the bucketization process through a dominance score, or an external linear order, is not new and has already been employed, along similar lines, in Gionis et al. (2006)).

[3] The *height* of x in a linear order ℓ is 1+ the number of elements below x in ℓ; the *average height* of x in a poset π is the average of its heights in the linear extensions of π. The average height of x can be easily computed from its mutual ranking probabilities (De Loof, 2009).

Algorithm 1 Selection of the bucket order in the BST trees minimizing the loss function.

function SEARCHBSPLIT($b_1 \oplus \cdots \oplus b_h$: BucketOrder, p: int)
  parameters: l : LossFunction
  if $h = n$ or $p > h$ **then**
    return $b_1 \oplus \cdots \oplus b_h$
  else
    $r \leftarrow$ SEARCHBSPLIT($b_1 \oplus \cdots \oplus b_h, p+1$)
    $\mathcal{B} \leftarrow \left[b_1 \oplus \cdots \oplus b_{p_1} \oplus b_{p_2} \oplus \cdots \oplus b_h \textbf{ foreach } \textit{bsplit-admissible } b_{p_1} \cup b_{p_2} = b_p\right]$
    $\mathcal{R} \leftarrow$ [SEARCHBSPLIT($\beta, p+2$) **foreach** β in $\mathcal{B}$]
    return $\underset{\beta \in (\{r\} \cup \mathcal{B} \cup \mathcal{R})}{\operatorname{argmin}} \{x : x = l(\beta)\}$
  end if
end function

Algorithm 2 Selection of the bucket order in the BJT trees minimizing the loss function.

function SEARCHBJOIN($b_1 \oplus \cdots \oplus b_h$: BucketOrder, p: int)
  parameters: l : LossFunction
  if $p \geq h$ or $p = 0$ **then**
    return $b_1 \oplus \cdots \oplus b_h$
  else
    $r \leftarrow$ SEARCHBJOIN($b_1 \oplus \cdots \oplus b_h, p-1$)
    $\mathcal{B} \leftarrow \left[b_1 \oplus \cdots \oplus b_{p_1} \oplus (b_{p+1} \cup b_{p_2}) \oplus \cdots \oplus b_h \right.$
      $\left. \textbf{foreach } \textit{bjoin-admissible } b_{p_1} \oplus (b_{p+1} \cup b_{p_2}) = b_p \oplus b_{p+1}\right]$
    $\mathcal{R} \leftarrow$ [SEARCHBJOIN($\beta, p-1$) **foreach** β in $\mathcal{B}$]
    return $\underset{\beta \in (\{r\} \cup \mathcal{B} \cup \mathcal{R})}{\operatorname{argmin}} \{x : x = l(\beta)\}$
  end if
end function

C. Admissible partitions. Driving scores are then used to define the set of admissible partitions, as follows. Given a bucket b_j, we call a partition $b_j = b_{j_1} \cup b_{j_2}$ *bsplit-admissible* if and only if all the dominance scores of the elements of b_{j_2} are not lower than all of the dominance scores of the elements of b_{j_1}; we call it *bjoin-admissible* if and only if the following two conditions are met: (i) it is *bsplit-admissible* and (ii) the elements in b_{j_2} are incomparable, in $\pi_\cap$, with those in the next bucket b_{j+1} (this condition assures the *bjoin* operator to preserve the dominances of the intersection $\pi_\cap$).

D. The core of the bucketization algorithm. Based on the above definitions, at iteration $i \geq 1$, the bucketization algorithm splits and joins the buckets $b_1(i-1), b_2(i-1), \ldots$ of the bucket order β_{i-1} produced at iteration $i-1$, searching for the move that minimizes the loss function, among all possible moves. However, being unfeasible to compare them all, the tested moves are restricted as follows. Let $BST_j(i)$ be the tree whose elements are bucket orders generated by performing all admissible *bsplits* of all the buckets $b_j(i-1), b_{j+1}(i-1), \ldots$ and combining them in all possible ways and let $BJT_j(i)$ be the analogous tree generated through *bjoins* instead of *bsplits*. Then the algorithm (see Algorithms 1 and 2)

1. builds the collection of bucket split trees $BST_1(i), BST_2(i), BST_3(i), \ldots$;
2. builds the collection of bucket join trees $BJT_1(i), BJT_2(i), BJT_3(i), \ldots$;
3. for each bucket order in each of the above trees, computes the loss function and selects the one that minimizes it.

The above operations are repeated until no further move is possible, that is, when a linear order is reached. The optimal bucket order β^* is then chosen to be the one (if unique) globally minimizing the loss function, along the entire sequence $\beta_0, \beta_1, \ldots, \beta_m$, generated by the algorithm.

Remark. Notice that the $BST_j(i)$ tree is composed of bucket orders whose elements have the buckets up to the $j-1$th equal to those of the bucket order β_{i-1}. So the collection of BST trees can be seen as an "upper " diagonal sequence of trees. An analogous remark holds for the collection of BJT trees.

E. Algorithm initialization. To run the algorithm, an initial bucket order β_0 is to be set; however, differently from the linearization case, here there is no natural unique choice for it. The essential requirements are that β_0 must extend the intersection of the input posets $\pi_\cap$ and, very informally stated, that it should do this by modifying it in a "minimal" way. Denoting by $Ext(\pi_\cap)$ the set of extensions of $\pi_\cap$, partially ordered by inclusion (see Section 4), the candidate initial bucket orders should then be the minimal elements of $Ext(\pi_\cap) \cap BO_X$, that is, those posets that are (i) extensions of the intersection, (ii) bucket orders, and (iii) do not extend any other poset with the first two properties. Unfortunately, at least to our knowledge, there is no easy way to generate this minimal set, and one has to guess sensible starting points, possibly choosing more than one. There is, however, a simple procedure to generate an element of $Ext(\pi_\cap) \cap BO_X$, namely, take the minimal elements of $\pi_\cap$ and put them into the first (from bottom) bucket; remove these elements from $\pi_\cap$, take the minimal elements of the resulting poset and form the second bucket; goes on this way, until no elements are left. In this case, β_0 is formally defined by

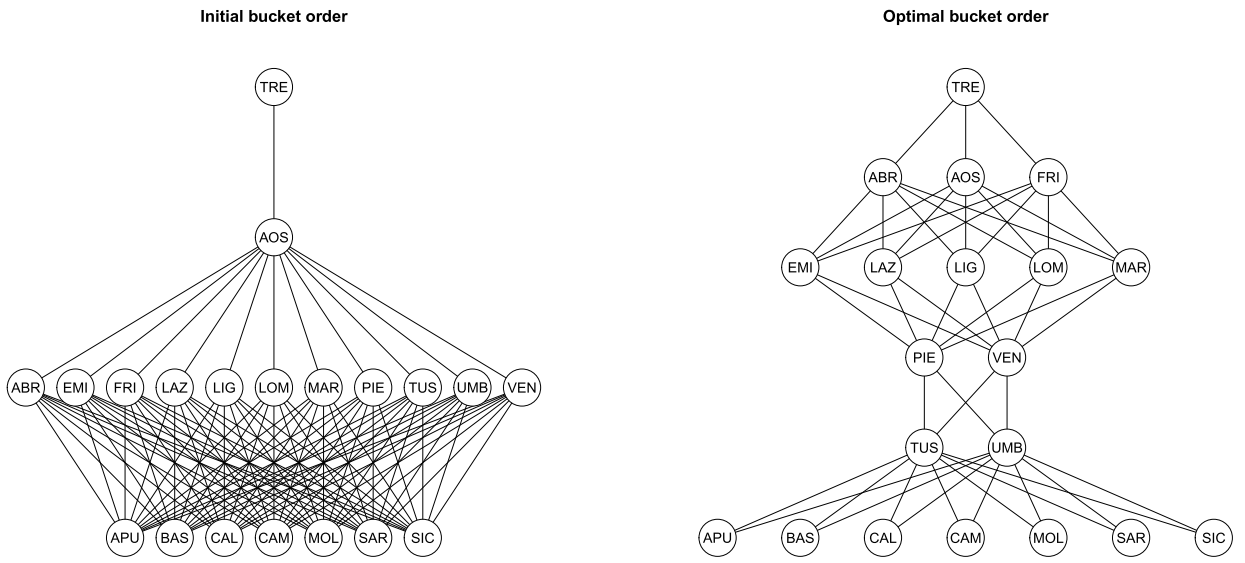

Fig. 16. Hasse diagrams of the initial bucket order β_0 (left) and the optimal bucket order β^* (right) for the well-being example under the L_1-loss function.

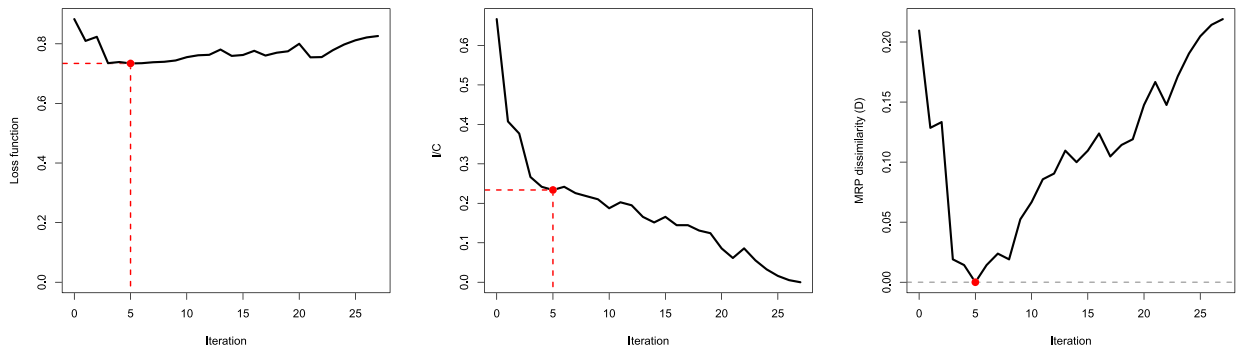

Fig. 17. Bucketization algorithm: the L_1-loss function (left), the I/C ratio (middle), and the dissimilarity measure D (right) as functions of the number of iterations for the well-being example.

$$\beta_0 = \bigoplus_i min(\pi_\cap^i) \quad \text{with} \quad \pi_\cap^i = \pi_\cap^{i-1} \setminus min(\pi_\cap^{i-1}) \ (i > 1) \quad \text{and} \quad \pi_\cap^1 = \pi_\cap, \tag{14}$$

where $min(\pi)$ is the set of minimal elements of π. By construction, β_0 is an extension of $\pi_\cap$. Moreover, it is minimal in $Ext(\pi_\cap) \cap BO_X$: indeed, if β belongs to $Ext(\pi_\cap) \cap BO_X$ and is extended by β_0, then at least one of its buckets must be the union of some buckets of the latter, but this is impossible since any pair of buckets in β_0 must contain at least a pair of elements (one in each bucket) comparable in $\pi_\cap$.

Some additional comments are now in order:

1. Since *bsplit* and *bjoin* preserve both bucketness and the dominances of $\pi_\cap$, the sequence $\beta_0, \beta_1, \ldots, \beta_m$ lies within BO_X and is composed of extensions of the intersection $\pi_\cap$, as desired.
2. The restriction to bucket partitions with nonempty subsets prevents entire buckets from being *bjoined*; as a consequence, the bucketization algorithm cannot produce the same bucket order twice and necessarily terminates. Moreover, such a condition is not that restrictive as it may seem at first; indeed, apart when $\pi_\cap$ is an antichain (and so $\beta_0 = \pi_\cap$), or when the algorithm is near to the final linear order, two different buckets are likely to contain at least a pair of elements that are comparable in $\pi_\cap$, and therefore the whole bucket, seen as a trivial partition, would not satisfy condition (ii) either, and so would not be join-admissible anyway.
3. The bucketization algorithm can be implemented with any of the loss functions discussed in the preceding sections (and possibly others). In the example below, we employ the L_1 one.

Figs. 16 and 17 exemplify the bucketization algorithm on the well-being data, with the L_1-loss function and the average height driving scores. The algorithm took 27 iterations to get to the final linear order. The optimal bucket is generated at

Table 10
Computation times (seconds) for the linearization algorithm, for different numbers of elements and I/C ratios of the input posets and for different loss functions. Mutual ranking probabilities have been computed exactly, with the lattice-of-ideals approach (the algorithm was run on an iMac Pro 1.1, with eight-core Intel Xeon W, 3.2 GHz, 32 GB RAM, and macOS 11.6).

I/C	$n = 20$			$n = 30$			$n = 40$		
	0.40	0.50	0.75	0.4	0.5	0.75	0.4	0.5	0.75
L_1	10.26	10.12	30.40	125.50	207.46	610.300	917.30	2549.80	4706.60
JS	252.96	330.49	595.48	2748.90	3839.00	4275.40	15858.50	23813.90	39993.00
JS1	251.60	309.60	636.64	2093.00	2658.03	3747.70	15150.20	23674.50	40361.00
JS2	308.22	421.91	738.69	3190.51	4600.00	5075.00	19038.00	30211.00	46814.00

the fifth iteration, with a loss value equal to 0.7341 (higher than the corresponding loss for the linearization algorithm, as expected). The computation times were negligible, given the simple structure of the MRP matrices of bucket orders.

9. Computational issues

In this section, we present a brief discussion of the computational issues raised by the linearization and, to a lesser extent, bucketization algorithms. It must be noted that the computation times for both procedures depend heavily on the structure of the input posets and on the selected loss function, which together determine the path followed to search for the best solution within the poset space. Thoroughly investigating the behaviors of the algorithms requires a dedicated and systematic study, which is outside the scope of this section; nevertheless, the discussion below and the values presented provide a first assessment of the range of applicability of the algorithms, also suggesting how the computational burden can be lightened. In general terms, the main source of computational complexity in the algorithms is the construction of the MRP matrices. For posets with a relatively small number of incomparabilities, this is usually achieved by generating all the linear extensions (LEs) and counting, for each poset element, the fraction of LEs over which it dominates the other elements. Various algorithms for the generation of LEs exist in the literature, from simple recursive procedures to more efficient ones, like the loopless algorithm of Korsh and LaFollette (2002) or the approach based on the so-called *lattice of ideals*, proposed in De Loof et al. (2006). Interestingly, mutual ranking probabilities can also be obtained by properly counting the number of LEs and not by listing them all, thereby lessening the computational burden; however, apart from very simple cases (e.g., bucket orders), there are no closed formulas for the number of LEs, and one must employ specific algorithms to compute it, as we do in our C++ implementations, following the approach of De Loof (2009). When the exact computation of the mutual ranking probabilities is not feasible, LE sampling procedures need to be employed. Among these, the most widely used is the MCMC algorithm of Bubley and Dyer (1999) for the quasi-uniform sampling of linear extensions, although other procedures have been proposed more recently (Talvitie et al., 2018). Let us finally mention that some rough approximation formulas for mutual ranking probabilities have also been proposed (Bruggemann and Carlsen, 2011).

With regard to computational performance, we discuss the two algorithms separately. In the linearization case, the computational complexity is mainly due to (i) the computation of the input MRP matrices and (ii) the computation of the MRP matrices of all of the tested posets at each iteration. We have run the algorithm on posets composed of 20, 30, and 40 elements with different I/C ratios, to consider both "more linear" (lower I/C) and less linear (higher I/C) cases, and with all of the four loss functions introduced in this paper. In all tests, the number of input posets has been set to 10. As can be seen from Table 10, the computation times are very sensitive to the parameter I/C, increasing rapidly when this gets larger. They also depend significantly on the loss functions, as revealed by the large differences between the L_1-based and JS-based losses. We even tested a poset with $I/C = 1.00$ and 40 elements, getting quite large computation times (e.g., 52 734.00 seconds for the L_1-based loss function and 126 583.00 seconds for the JS case). With 30 elements, we could also test the $I/C = 1.5$ case, with computation times of 4600.00 seconds (L_1), 15 542.70 seconds (JS), 12 988.00 seconds (JS1), and 16 563.00 seconds (JS2).

In the bucketization algorithm, apart from the computation of the input MRP matrices, most of the computational effort is due to the number of *bsplit/bjoin* operations to be performed at each iteration (the computation of the MRPs of the single bucket orders being trivial), to the depths of the BST and BJT trees, and to the number of iterations to be done, which depends in a complex way upon the structure of the initial bucket order β_0 and upon the kind of loss function. To give an overall idea of the performance of the bucketization algorithm, independently of the computation of the input MRPs, we have considered the "worst" case, where β_0 is an antichain and have run the algorithm for $n = 50$, 60, 65, and 75, with 10 linear input posets. Table 11 presents the computation times using the L_1-, JS-, JS1-, and JS2-loss functions. As expected, their order of magnitude is definitely lower than that of the linearization algorithm.

The values reported above neatly show the computational burden of the algorithms, particularly the linearization one. To overcome this problem and widen the range of applicability of the two procedures, some strategies can be pursued. Indeed, the logical thread of the two procedures can be also implemented in a lighter way, by substituting the MRP matrices with other coarser representations of poset structure. For example, the linearization algorithm with L_1-based loss could be implemented substituting MRP matrices with dominance matrices or, to retain more structural information, with matrices consisting of the graph distances in the Hasse diagram between each element and the elements it dominates (e.g., computed by Dijkstra's algorithm). In the case where the JS-based losses are used, MRP matrices can be approximated by the formula

Table 11
Computation times (seconds) for the bucketization algorithm, for different numbers of elements for the input posets and different loss functions (the algorithm was run on an iMac Pro 1.1, with eight-core Intel Xeon W, 3.2 GHz, 32 GB RAM, and macOS 11.6).

n	50	60	65	75
L_1	0.658	16.282	18.778	322.653
JS	7.396	28.208	54.764	92.330
JS1	7.951	30.019	57.779	97.036
JS2	7.602	29.292	56.807	95.547

given in Bruggemann and Carlsen (2011), which, although quite rough, can nevertheless be used to drive the algorithm. The bucketization algorithm can instead be made lighter by reducing the number of bucket orders tested at each iteration (e.g., by constructing new tentative bucket orders by combining only the best *bsplits* or *bjoins* for each bucket, or even by just *bsplitting* or *bjoining* one bucket at a time). Implementing and assessing these alternatives is, however, outside the scope of this paper, whose aim is to outline and exemplify a general approach to multi-poset approximation.

10. Conclusion and open issues

In this paper, we have introduced novel "linearization" and "bucketization" algorithms for the approximation of multidomain systems of partially ordered sets (posets), a relevant problem in multi-indicator system analysis, multicriteria decision-making, and indicator construction. Given k posets on the same ground set, the linearization algorithm generates a greedy nested sequence of posets, searching for the optimal approximating one and, eventually, for a final linear order; the bucketization algorithm performs a similar process, restricting the search to the class of bucket orders. The algorithms have been demonstrated in action on real data pertaining to subjective well-being in Italian regions, with different loss functions based on the classical L_1 distance or on unweighted and weighted Jensen–Shannon divergences.

Some issues remain to be further investigated, both from a computational and a mathematical point of view. Ways to lessen the computational burden of the linearization algorithm must be explored, for example by computing or approximating mutual ranking probabilities more efficiently, or by using other structural matrices in place of the MRP ones, thereby reducing the computational costs without significantly affecting the search power of the procedure. The role of the different loss functions also deserves further investigation, particularly given the distinctive behavior of the JS1 dissimilarity measure. This is relevant from both a theoretical and a practical point of view, determining the kind of information extracted by the approximation process and consequently the structure and interpretation of the outputs. With regard to the bucketization algorithm, two major open issues are (i) to design alternative implementations, reducing the number of *bsplit* and *bjoin* operations to be performed and considered, when generating the bucket order trees and (ii) to develop an algorithm for identifying alternative initial guesses, satisfying the conditions discussed in Section 8. Both problems require a deeper study of the mathematics of the bucket order space, with the aim of better understanding the properties of the *bsplit/bjoin* moves and the structure of the paths followed by the algorithm within the search space. Finally, both the linearization and bucketization algorithms can be generalized and made more flexible, to allow for input posets sharing just a common subset of elements (see the remarks in Section 5) or when weights are attached to the elements of the inputs.

Finally, the reader may have noticed that, in the title and in the main text, the expression "complexity reduction" has been adopted, instead of the more usual "dimensionality reduction." This is not by chance. Although it is possible to define the *dimension* of a poset as the minimum number of linear extensions that generate it by intersection (Trotter, 1992), this concept is not that useful in data analysis, not being properly related to the intuitive idea that a poset becomes simpler as it approximates a complete order. In fact, along the sequence of extensions leading from a k-dimensional poset to a linear order (which has dimension 1), the dimensionality of the intermediate posets need not decrease monotonically. From a mathematical point of view, the issue is that there is no clear definition of "poset simplification"; namely, both linear orders (posets without incomparabilities) and antichains (posets without comparabilities) can be considered as "simple," even if they are structurally opposites. Independently of the number of elements, any antichain has dimension two (see Trotter, 1992), and thus both a linear order and an antichain can be seen as the result of dimensionality reduction processes, which start from a k-dimensional poset and, in a sense, point in different directions. This leads to an interesting foundational issue: since in evaluation and prioritization studies one is often concerned with rankings, a more useful definition of poset dimension should be worked out, to drive dimensionality reduction processes toward linearity, analogously to what is done by Euclidean dimension in classical multidimensional data analysis.

References

Aledo, J.A., Gámez, J.A., Rosete, A., 2017. Utopia in the solution of the bucket order problem. Decis. Support Syst. 97, 69–80.
Arcagni, A., Barbiano di Belgiojoso, E., Fattore, M., Rimoldi, S.M.L., 2019. Multidimensional analysis of deprivation and fragility patterns of migrants in Lombardy, using partially ordered sets and self-organizing maps. Soc. Indic. Res. 141, 551–579.
Beerenwinkel, N., Eriksson, N., Sturmfels, B., 2007. Conjunctive Bayesian networks. Bernoulli 13, 893–909.
Belohlávek, R., Vychodil, V., 2010. Discovery of optimal factors in binary data via a novel method of matrix decomposition. J. Comput. Syst. Sci. 76, 3–20.
Bruggemann, R., Carlsen, L., 2011. An improved estimation of averaged ranks of partial orders. MATCH Commun. Math. Comput. Chem. 65, 383–414.
Brüggemann, R., Patil, G.P., 2011. Ranking and Prioritization for Multi-indicator Systems: Introduction to Partial Order Applications. Springer.

Bubley, R., Dyer, M., 1999. Faster random generation of linear extensions. Discrete Math. 201, 81–88.
Comim, F., 2021. A poset-generalizability method for human development indicators. Soc. Indic. Res., 1–20. https://doi.org/10.1007/s11205-021-02737-0.
D'Ambrosio, A., Iorio, C., Staiano, M., Siciliano, R., 2019. Median constrained bucket order rank aggregation. Comput. Stat. 34, 787–802.
Davey, B.A., Priestley, H.A., 2002. Introduction to Lattices and Order. Cambridge University Press.
De Loof, K., 2009. Efficient computation of rank probabilities in posets. Ph.D. thesis. Ghent University.
De Loof, K., De Meyer, H., De Baets, B., 2006. Exploiting the lattice of ideals representation of a poset. Fundam. Inform. 71, 309–321.
Di Bella, E., Gandullia, L., Leporatti, L., Montefiori, M., Orcamo, P., 2018. Ranking and prioritization of emergency departments based on multi-indicator systems. Soc. Indic. Res. 136, 1089–1107.
Fattore, M., 2016. Partially ordered sets and the measurement of multidimensional ordinal deprivation. Soc. Indic. Res. 128, 835–858.
Fattore, M., Arcagni, A., 2019. F-FOD: fuzzy first order dominance analysis and populations ranking over ordinal multi-indicator systems. Soc. Indic. Res. 144, 1–29.
Fattore, M., Arcagni, A., 2020. Ranking extraction in ordinal multi-indicator systems. In: Book of Short Papers – SIS 2020. Pearson.
Fattore, M., Brüggemann, R., 2017. Partial Order Concepts in Applied Sciences. Springer.
Fattore, M., Brüggemann, R., Owsiński, J., 2011. Using poset theory to compare fuzzy multidimensional material deprivation across regions. In: New Perspectives in Statistical Modeling and Data Analysis. Springer, pp. 49–56.
Fattore, M., Maggino, F., Colombo, E., 2012. From composite indicators to partial orders: evaluating socio-economic phenomena through ordinal data. In: Quality of Life in Italy: Researches and Reflections. Springer.
Feng, J., Fang, Q., Ng, W., 2008. Discovering bucket orders from full rankings. In: SIGMOD '08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data.
Fernandez, P.L., Heath, L.S., Ramakrishnan, N., Tan, M., Vergara, J.P.C., 2013. Mining posets from linear orders. Discrete Math. Algorithms Appl. 5, 1350030.
Foldes, S., Radeleczki, S., 2001. On distances and metrics in discrete ordered sets. Math. Bohem. 146, 251–262.
Ganter, B., Wille, R., 1999. Formal Concept Analysis: Mathematical Foundations. Springer.
Garriga, G.C., 2005. Summarizing sequential data with closed partial orders. In: SIAM International Conference on Data Mining, pp. 80–91.
Gionis, A., Kujala, T., Mannila, H., 2003. Fragments of order. In: KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
Gionis, A., Mannila, H., Puolamäki, K., Ukkonen, A., 2006. Algorithms for discovering bucket orders from data. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 561–566.
Iglesias, K., Suter, C., Beycan, T., Vani, B.P., 2017. Exploring multidimensional well-being in Switzerland: comparing three synthesizing approaches. Soc. Indic. Res. 134, 847–875.
Istat, 2015. Indagine multiscopo – Aspetti della vita quotidiana.
Jacques, J., Biernacki, C., 2018. Model-based co-clustering for ordinal data. Comput. Stat. Data Anal. 123, 101–115.
Kenkre, S., Khan, A., Pandit, V., 2011. On discovering bucket orders from preference data. In: Proceedings of the 2011 SIAM International Conference on Data Mining. SIAM.
Korhonen, P., Siljamäki, A., 1998. Ordinal principal component analysis theory and an application. Comput. Stat. Data Anal. 26, 411–424.
Korsh, J.F., LaFollette, P.S., 2002. Loopless generation of linear extensions of a poset. Order 19, 115–126.
Liddell, T.M., Kruschke, J.K., 2018. Analyzing ordinal data with metric models: what could possibly go wrong? J. Exp. Soc. Psychol. 79, 328–348.
Lin, J., 1991. Divergence measures based on the Shannon entropy. IEEE Trans. Inf. Theory 37, 145–151.
Liu, A., Zhao, Z., Liaom, C., Lu, P., Xia, L., 2019. Learning Plackett–Luce mixtures from partial preferences. In: Proceedings of the AAAI Conference on Artificial Intelligence.
Lu, T., Boutilier, C., 2014. Effective sampling and learning for Mallows models with pairwise-preference data. J. Mach. Learn. Res. 15, 3783–3829.
Madden, D., 2010. Ordinal and cardinal measures of health inequality: an empirical comparison. Health Econ. 19, 243–250.
Mannila, H., Meek, C., 2000. Global partial orders from sequential data. In: KDD '00: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
Monjardet, B., 1981. Metrics on partially ordered sets – a survey. Discrete Math. 35, 173–184.
Patil, G.P., Taillie, C., 2004. Multiple indicators, partially ordered sets, and linear extensions: multi-criterion ranking and prioritization. Environ. Ecol. Stat. 11, 199–228.
Puolamäki, K., Fortelius, M., Mannila, H., 2006. Seriation in paleontological data using Markov chain Monte Carlo methods. PLoS Comput. Biol. 2 (2), e6.
Schröder, B.S.W., 2003. Ordered Sets. An Introduction. Birkäuser.
Sen, A., 1992. Inequality Reexamined. Harvard University Press.
Shye, S., 1978. Partial order scalogram analysis. In: Theory Construction and Data Analysis in Behavioural Sciences. Jossey Bass.
Shye, S., 1985a. Multiple Scaling. The Theory and Application of Partial Order Scalogram Analysis. Technical Report. Israel Institute of Applied Social Research, Jerusalem.
Shye, S., 1985b. Multiple Scaling: The Theory and Application of Partial Order Scalogram Analysis. North-Holland.
Shye, S., 2009. Partial order scalogram analysis by coordinates (POSAC) as a facet theory measurement procedure: how to do POSAC in four simple steps. In: Cohen, A. (Ed.), Facet Theory and Scaling: In Search for Structure in Behavioral and Social Sciences Facet Theory. Facet Theory Association Press, pp. 295–310.
Shye, S., Amar, R., 1985. Partial-order scalogram analysis by base coordinates and lattice mapping of the items by their scalogram roles. In: Facet Theory. Springer.
Talvitie, T., Kangas, K., Niinimäki, T., Koivisto, M., 2018. Counting linear extensions in practice: MCMC versus exponential Monte Carlo. In: Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18). Association for the Advancement of Artificial Intelligence.
Trotter, W.T., 1992. Combinatorics and Partially Ordered Sets: Dimension Theory. Johns Hopkins University Press.
Ukkonen, A., Fortelius, M., Mannila, H., 2005. Finding partial orders from unordered 0-1 data. In: KDD '05: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining.
Ukkonen, A., Puolamäki, K., Gionis, A., Mannila, H., 2009. A randomized approximation algorithm for computing bucket orders. Inf. Process. Lett. 109, 356–359.
Watt, A.M., 2015. Inference for partial orders from random linear extensions. Ph.D. thesis. University of Oxford.
Zelinka, B., 1993. Distances between partially ordered sets. Math. Bohem. 118, 167–170.
Zhang, Y., Cheung, Y.m., 2020. An ordinal data clustering algorithm with automated distance learning. In: Proceedings of the AAAI Conference on Artificial Intelligence.
Zhao, Z., Xia, L., 2019. Learning mixtures of Plackett–Luce models from structured partial orders. In: Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19).
Zhao, Z., Xia, L., 2020. Learning mixtures of Plackett–Luce models with features from top-*l* orders. arXiv:2006.03869v2 [cs.LG].