TOP (2020) 28:271–339
https://doi.org/10.1007/s11750-020-00563-0

INVITED PAPER

Distance geometry and data science

Leo Liberti[1]

Published online: 5 June 2020

Abstract
Data are often represented as graphs. Many common tasks in data science are based on distances between entities. While some data science methodologies natively take graphs as their input, there are many more that take their input in vectorial form. In this survey, we discuss the fundamental problem of mapping graphs to vectors, and its relation with mathematical programming. We discuss applications, solution methods, dimensional reduction techniques, and some of their limits. We then present an application of some of these ideas to neural networks, showing that distance geometry techniques can give competitive performance with respect to more traditional graph-to-vector mappings.

Keywords Euclidean distance · Isometric embedding · Random projection · Mathematical programming · Machine learning · Artificial neural networks

Mathematics Subject Classification 51Kxx · 90Cxx · 68Pxx

1 Introduction

This survey is about the application of distance geometry (DG) techniques to problems in data science (DS). More specifically, data are often represented as graphs, and many methodologies in data science require vectors as input. We look at the fundamental problem in DG, namely that of reconstructing vertex positions from given edge lengths, in view of using its solution methods to produce vector input for further data processing.

Dedicated to the memory of Mariano Bellasio (1943–2019).

This invited paper is discussed in the comments available at https://doi.org/10.1007/s11750-020-00560-3, https://doi.org/10.1007/s11750-020-00561-2, https://doi.org/10.1007/s11750-020-00562-1.

✉ Leo Liberti
liberti@lix.polytechnique.fr

[1] LIX CNRS, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

The organization of this survey is based on a "storyline". In summary, we want to exhibit alternative competitive methods for mapping graphs to vectors to analyse graphs using machine learning (ML) methodologies that take vectorial input. This storyline will take us through fairly different subfields of mathematics, operations research, and computer science. This survey does not provide exhaustive literature reviews in all these fields. Its purpose (and usefulness) rests in communicating the main idea sketched above, rather than serving as a reference for a field of knowledge. It is nonetheless a survey because, limited to the scope of its purpose, it aims at being informative and also partly educational, rather than just giving the minimal notions required to support its goal.

Here is a more detailed account of our storyline. We first introduce DG, some of its history, its fundamental problem, and its applications. Then, we motivate the use of graph representations for several types of data. Next, we discuss some of the most common tasks in data science (e.g., classification and clustering) and the related methodologies (unsupervised and supervised learning). We introduce robust and efficient algorithms used for embedding general graphs in vector spaces. We present some dimensional reduction operations, which are techniques for replacing sets X of high-dimensional vectors by lower dimensional ones X', so that some of the properties of X are preserved at least approximately in X'. We discuss the instability of distances on randomly generated vectors and its impact on distance-based algorithms. Finally, we present an application of much of the foregoing theory: we train an artificial neural network (ANN) on many training sets, so as to learn several given clusterings on sentences in natural language. Some training sets are generated using the traditional methods, namely incidence vectors of short sequences of consecutive words in the corpus dictionary. Other training sets are generated by representing sentences by graphs and then using a DG method to encode these graphs into vectors. It turns out that some of the DG-generated training sets have competitive performances with the traditional methods. While the empirical evidence is too limited to support any general conclusion, it might invite more research on this topic.

The survey is interspersed with eight theorems with proofs. Aside from Theorem 8 about distance instability, the proof of which is taken almost verbatim from the original source (Beyer et al. 1998), the proofs from the other well-known theorems are not taken from specific sources (this does not mean that the theorems or their proofs are original). The presented proofs are reasonably short, and, we hope, easy to follow. There are several reasons for the presence of these theorems in this survey: (a) we have not found them stated and proved clearly anywhere else, and we wish we had during our research work (Theorems 1–4); (b) their proofs showcase some point we deem important about the underlying theory (Theorems 7–8); (c) they give some indication of the proof techniques involved in the overarching field (Theorem 6–7); (d) they justify a mathematical statement for which we found no citation (Theorem 5). While there may be some original mathematical results in this survey, e.g., Eq. (35) and the corresponding Theorem 5 (though something similar might be found in Henry Wolkowicz' work) as well as the computational comparison in Sect. 7.3.2, we believe that the only truly original part is the application of DG techniques to constructing training sets of ANNs in Sect. 9. Section 4, about representing data by graphs, may also contain some new ideas to Mathematical

Programming (MP) readers, although everything that we wrote can be easily reconstructed from existing literature.

In the following, we use formal notations from different fields, which may be confusing to some readers. The underlying assumption is that sentences are written as is customary in axiomatic set theory: existential ($\exists$) or universal ($\forall$) quantification on the left of the sentence by default, brackets for operator priority disambiguation, standard arithmetic/transcendental operators/functions, $\vee$ to denote disjunction ("or"), $\wedge$ to denote conjunction ("and") of two sentences, and $\neg$ to denote negation of a sentence. Some shortcuts are used to decrease the number of formal symbols and improve readability: "$\forall a \in A \, \forall b \in B$" is shortened to "$\forall a \in A, b \in B$", and similarly for $\exists$; if K is an integer and k is an index, $k \leq K$ means $k \in \{1, \ldots, K\}$; specifically, this is used in the arguments of $\forall, \exists, \sum, \prod$ quantifiers. The character $\rightarrow$ is used formally in the definition of functions (e.g., $f : A \rightarrow B$ denotes a function mapping elements of the set A to elements of the set B) or as the relation "implies" between to logical sentences within a formal language (i.e., $A \rightarrow B$ means $\neg(A \wedge \neg B)$); the same relation in the meta-language is denoted $\Rightarrow$ (i.e., $A \Rightarrow B$ means "from A one can deduce that B", where the formal deduction is not specified).

The rest of this paper is organized as follows. In Sect. 2, we give a brief introduction to the field of MP, considered as a formal language for optimization. In Sect. 3, we introduce the field of DG. In Sect. 4, we give details on how to represent four types of data as graphs. In Sect. 5, we introduce methods for clustering on vectors as well as directly on graphs. In Sect. 6, we present many methods for realizing graphs in Euclidean spaces, most of which are based on MP. In Sect. 7, we introduce some dimensional reduction techniques. In Sect. 8, we discuss the distance instability phenomenon, which may have a serious negative impact on distance-based algorithms. In Sect. 9, we present an application of clustering in natural language by means of an ANN, and discuss how the aforementioned DG techniques can help to construct the input part of the training set.

2 Mathematical programming

Many of the methods discussed in this survey are optimization methods. Specifically, they belong to MP, which is a field of optimization science and operation research. While most of the readers of this paper should be familiar with MP, the interpretation which we give to this term is more formal than most other treatments, and we therefore discuss it in this section.

2.1 Syntax

MP is a formal language for describing optimization problems. The valid sentences of this language are the MP formulations. Each formulation consists of an array p of parameter symbols (which encode the problem input), an array x of n decision variable symbols (which will contain the solution), an objective function $f(p, x)$ with an

optimization direction (either min or max), a set of explicit constraints $g_i(p,x) \leq 0$ for all $i \leq m$, and some implicit constraints, which impose that x should belong to some implicitly described set X. For example, some of the variables might be constrained to take integer values only, or to belong to the non-negative orthant, or to a positive semidefinite (psd) cone. The standard MP formulation is as follows:

$$\left.\begin{array}{l} \underset{x\in\mathbb{R}^n}{\text{opt}}\ f(p,x) \\ \forall i \leq m\ g_i(p,x) \leq 0 \\ x \in X. \end{array}\right\} \tag{1}$$

We note that indices, or sets thereof, appearing in the arguments of quantifiers such as $\forall, \sum, \prod$ cannot depend on the values of decision variables.

It is customary to define MP formulations over explicitly closed feasible sets, to prevent issues with feasible formulations which have infima or suprema but no optima. This forbids the use of strict inequality symbols in the MP language.

2.2 Taxonomy

MP formulations are classified according to syntactical properties. We list the most important classes:

- if f, g_i are linear in x and X is the whole space, Eq. (1) is a linear program (LP);
- if f, g_i are linear in x and $X = \{0,1\}^n$, Eq. (1) is a binary linear program (BLP);
- if f, g_i are linear in x and X is the whole space intersected with an integer lattice (possibly defined on a subset of the spatial dimensions), Eq. (1) is a mixed-integer linear program (MILP);
- if f is quadratic in x, g_i are linear in x, and X is the whole space, Eq. (1) is a quadratic program (QP); if f is convex, then it is a convex QP (cQP);
- if f is linear in x, g_i are quadratic in x, and X is the whole space or a polyhedron, Eq. (1) is a quadratically constrained program (QCP); if g_i are convex, it is a convex QCP (cQCP);
- if f and g_i are quadratic in x, and X is the whole space or a polyhedron, Eq. (1) is a quadratically constrained quadratic program (QCQP); if f, g_i are convex, it is a convex QCQP (cQCQP);
- if f, g_i are (possibly) nonlinear functions in x, and X is the whole space or a polyhedron, Eq. (1) is a nonlinear program (NLP); if f, g_i are convex, it is a convex NLP (cNLP);
- if x is a symmetric matrix of decision variables, f, g_i are linear, and X is the set of all psd matrices, Eq. (1) is a semidefinite program (SDP);
- if we impose some integrality constraints on any decision variable on formulations from the classes QP, QCQP, NLP, and SDP, we obtain their respective mixed-integer variants MIQP, MIQCQP, MINLP, and MISDP.

This taxonomy is by no means complete (see Liberti 2009, §3.2 and Williams 1999).

2.3 Semantics

As in all formal languages, sentences are given a meaning by replacing variable symbols with other mathematical entities. In the case of MP, semantics are assigned by an algorithm, called solver, which looks for a numerical solution $x^* \in \mathbb{R}^n$ having some optimality properties and satisfying the constraints. For example, BLPs such as Eq. (19) can be solved by the CPLEX solver (IBM 2017). This allows users to solve optimization problems just by "modelling" them (i.e., describing them as an MP formulation) instead of having to invent a specific solution algorithm. As a formal descriptive language, MP was shown to be Turing-complete (Liberti 2019; Liberti and Marinelli 2014).

2.4 Reformulations

It is always the case that infinitely many formulations have the same semantics: this can be seen in a number of trivial ways, such as, e.g., multiplying some constraint $g_i \leq 0$ by any positive scalar in Eq. (1). This will produce an uncountable number of different formulations with the same feasible and optimal set.

Less trivially, this property is precious insofar as solvers perform more or less efficiently on different (but semantically equivalent) formulations. More generally, a symbolic transformation on an MP formulation for which one can provide some guarantees on the extent of the engendered modifications of the feasible and/or optimal set is called a reformulation (Liberti 2009; Liberti et al. 2009, 2010).

Three types of reformulation guarantees will appear in this survey:

- the exact reformulation: the optima of the reformulated problem can be mapped efficiently back to those of the original problem;
- the relaxation: the optimal objective function value of the reformulated problem provides a bound (in the optimization direction) on the optimal objective function value of the original problem;
- the approximating reformulation: a sequence of formulations based on a parameter which also appears in a "guarantee statement" (e.g., an inequality providing a bound on the optimal objective function value of the original problem); an additional desirable property is that, when the parameter tends to infinity, the guarantee proves that formulations in the sequence tend to an exact reformulation or to a relaxation.

Reformulations are only useful when they can be solved more efficiently than the original problem. Exact reformulations are important, because the optima of the original formulation can be retrieved easily. Relaxations are important to evaluate the quality of solutions of heuristic methods which provide solutions without any optimality guarantee; moreover, they are crucial in branch-and-bound (BB) type solvers (such as, e.g., CPLEX). Approximating reformulations are important to devise approximate solution methods for MP problems.

There are some trivial exact reformulations which guarantee that Eq. (1) is much more general than it would appear at first sight: for example, inequality constraints can be turned into equality constraints by the addition of slack or surplus variables; equality constraints can be turned to inequality constraints by listing the constraint twice, once with $\leq$ sense and once with $\geq$ sense; minimization can be turned to maximization by the equation $\min f = -\max -f$ (Liberti et al. 2009, §3.2).

2.4.1 Linearization

We note two easy, but very important types of reformulations.

- The linearization consists in identifying a nonlinear term $t(x)$ appearing in f or g_i, replacing it with an added variable y_t, and then adjoining the defining constraint $y_t = t(x)$ to the formulation.
- The constraint relaxation consists in removing a constraint: since this means that the feasible region becomes larger, the optima can only improve with respect to those of the original problem. Thus, relaxing constraints yields a relaxation of the problem.

These two reformulation techniques are often used in sequence: one identifies problematic nonlinear terms, linearizes them, and then relaxes the defining constraints. Carrying this out recursively for every term in an NLP (McCormick 1976) and only relaxing the nonlinear defining constraints yield an LP relaxation of an NLP (Smith and Pantelides 1999; Tawarmalani and Sahinidis 2004; Belotti et al. 2009).

3 Distance geometry

DG refers to a foundation of geometry based on the concept of distances instead of those of points and lines (Euclid) or point coordinates (Descartes). The axiomatic foundations of DG were first laid out in full generality by Menger (1928), and later organized and systematized by Blumenthal (1953). A metric space is a pair $(\mathbb{X}, d)$, where $\mathbb{X}$ is an abstract set and d is a binary relation $d : \mathbb{X} \times \mathbb{X} \to \mathbb{R}_+$ obeying the metric axioms:

1. $\forall x, y \in \mathbb{X} \quad d(x, y) = 0 \leftrightarrow x = y$ (identity);
2. $\forall x, y \in \mathbb{X} \quad d(x, y) = d(y, x)$ (symmetry);
3. $\forall x, y, z \in \mathbb{X} \quad d(x, y) + d(y, z) \geq d(x, z)$ (triangle inequality).

Based on these notions, one can define sequences and limits (through converging distances), as well as open and closed sets (through membership of limit points in sets). For any triplet x, y, z of distinct elements in $\mathbb{X}$, y is between x and z if $d(x, y) + d(y, z) = d(x, z)$. This notion of metric betweenness can be used to characterize convexity: a subset $\mathbb{Y} \subseteq \mathbb{X}$ is metrically convex if, for any two points $x, z \in \mathbb{Y}$, there is at least one point $y \in \mathbb{Y}$ between x and z. The fundamental notion

of invariance in metric spaces is that of congruence: two metric spaces $\mathbb{X}, \mathbb{Y}$ are congruent if there is a mapping $\mu : \mathbb{X} \to \mathbb{Y}$ such that for all $x, y \in \mathbb{X}$ we have $d(x, y) = d(\mu(x), \mu(y))$.

The word "isometric" is often used as a synonym of "congruent" in many contexts, e.g., with isometric embeddings (Sect. 6.2.2). In this survey, we mostly use "isometric" in relation to mappings from graphs to sets of vectors, such that the weights of the edges are the same as the length of the segments between the vectors corresponding to the adjacent vertices. In other words, "isometric" is mostly used for partially defined metric spaces—only the distances corresponding to the graph edges must be preserved.

While a systematization of the axioms of DG was only formulated in the twentieth century, DG is pervasive throughout the history of mathematics, starting with Heron's theorem (computing the area of a triangle given the side lengths) (Heron 50AD), going on to Euler's conjecture on the rigidity of (combinatorial) polyhedra (Euler 1862), Cauchy's creative proof of Euler's conjecture for strictly convex polyhedra (Cauchy 1813), Cayley's theorem for inferring point positions from determinants of distance matrices (Cayley 1841), Maxwell's analysis of the stiffness of frames (Maxwell 1864), Henneberg's investigations on rigidity of structures (Henneberg 1911), Gödel's fixed point theorem for showing that a tetrahedron with nonzero volume can be embedded isometrically (with geodetic distances) on the surface of a sphere (Gödel 1986), Menger's axiomatization of DG (Menger 1931), yielding, in particular, the concept of the Cayley–Menger determinant (an extension of Heron's theorem to any dimension, which was used in many proofs of DG theorems), up to Connelly's disproof of Euler's conjecture (Connelly 1978) in its most general form. A more detailed account of many of these achievements is given in Liberti and Lavor (2016). An extension of Gödel's theorem on the sphere embedding in any finite dimension appears in Liberti et al. (2016).

3.1 The distance geometry problem

Before the widespread use of computers, the main applied problem of DG was to congruently embed finite metric spaces (i.e., with all known distances) in some vector space. The first mention of the need for isometric embeddings using only a partial set of distances probably appeared in Yemini (1978). This need arose from wireless sensor networks: by estimating a set of distances for pairs of sensors which are close enough to establish peer-to-peer communication, is it possible to recover the position for all sensors in the network? Note that (a) distances can be recovered from peer-to-peer communicating pairs by monitoring the amount of battery required to exchange data; and (b) the positions for the sensors are in $\mathbb{R}^K$, with $K = 2$ (usually) or $K = 3$ (sometimes).

Thus, we can formulate the main problem in DG.

> Distance geometry problem (DGP): given an integer $K > 0$ and a simple undirected graph $G = (V, E)$ with an edge weight function $d : E \to \mathbb{R}_+$, determine whether there exists a realization $x : V \to \mathbb{R}^K$ such that:

$$\forall \{u, v\} \in E \quad \|x(u) - x(v)\| = d(u, v). \tag{2}$$

We let $n = |V|$ and $m = |E|$ in the following.

We can re-state the DGP as follows: given a weighted graph G and the dimension K of a vector space, draw G in $\mathbb{R}^K$ in such a way that each edge is drawn as a straight segment of length equal to its weight. We remark that the realization x, defined as a function, is usually represented as an $n \times K$ matrix $x = (x_{uk} \mid u \in V \wedge k \leq K)$, which may also be seen as an element of $\mathbb{R}^{nK}$.

Note that we usually write x_u, x_v and d_{uv} for $x(u), x(v)$ and $d(u, v)$. If the norm used in Eq. (2) is the Euclidean (ℓ_2) norm, then the above equation is usually squared, so it becomes a multivariate polynomial of degree two:

$$\forall \{u, v\} \in E \quad \|x_u - x_v\|_2^2 = d_{uv}^2. \tag{3}$$

While most of the distances in this paper will be Euclidean, we shall also mention the so-called *linearizable norms* (D'Ambrosio and Liberti 2017), i.e. ℓ_1 and ℓ_∞, because they can be described using piecewise affine functions. We also remark that the input of the DGP can also be represented by a *partial* $n \times n$ *distance matrix* D where only the entries d_{uv} corresponding to $\{u, v\} \in E$ are specified.

Many more notions about the DGP can be found in Liberti et al. (2014), Liberti and Lavor (2017). Recent results on the DGP related to graph theory are given in Lavor et al. (2019), Lavor et al. (2019); for recent results on the application to protein conformation, see Malliavin et al. (2019).

3.2 Number of solutions

A DGP instance may have no solutions if the given distances do not define a metric, a finite number of solutions if the graph is rigid, or uncountably many solutions if the graph is flexible.

Restricted to the ℓ_2 norm, there are several different notions of rigidity. We only define the simplest, which is easiest to explain intuitively: if we consider the graph as a representation of a joint-and-bar framework, a graph is flexible if the framework can move (excluding translations and rotations) and rigid otherwise. The formal definition of rigidity of a graph $G = (V, E)$ involves: (a) a mapping D from a realization $x \in \mathbb{R}^{nK}$ to the partial distance matrix:

$$\mathsf{D}(x) = (\|x_u - x_v\| \mid \{u, v\} \in E);$$

and (b) the completion $\mathsf{K}(G)$ of G, defined as the complete graph on V. We want to say that G is rigid if, were we to move x ever so slightly (excluding translations and rotations), $\mathsf{D}(x)$ would also vary accordingly. We formalize this idea indirectly: a graph is rigid if the realizations in a neighbourhood χ of x corresponding to changes in $\mathsf{D}(x)$ are equal to those in the neighbourhood $\bar{\chi}$ of a realization $\bar{x}$ of $\mathsf{K}(G)$ (Liberti and Lavor 2017, Ch. 7). We note that realizations $\bar{x} \in \bar{\chi}$ correspond to small variations in $\mathsf{D}(\mathsf{K}(G))$: this definition makes sense, because $\mathsf{K}(G)$ is a complete graph,

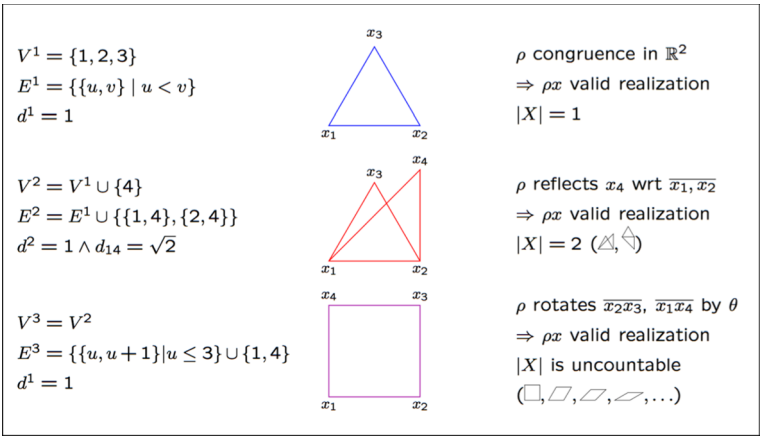

Fig. 1 Instances with one, two, and uncountably many realizations

which implies that its distance matrix is invariant, and hence, $\bar{\chi}$ may only contain congruences.

We thus obtain the following formal characterization of rigidity (Asimow and Roth 1978):

$$\mathsf{D}^{-1}(\mathsf{D}(x)) \cap \chi = \mathsf{D}^{-1}(\mathsf{D}(\bar{x})) \cap \bar{\chi}. \tag{4}$$

Let us parse Eq. (4): for a partial distance matrix Y, $\mathsf{D}^{-1}(Y)$ corresponds to all of the realizations that give rise to Y (which are uncountably many because of congruences). Now, let x be a realization of the partial distance matrix Y, and $\bar{x}$ a realization of the metric completion $\bar{Y}$ of Y (if it exists). Moreover, χ is a neighbourhood of x and $\bar{\chi}$ is a neighbourhood of $\bar{x}$ (in the vector space $\mathbb{R}^{nK}$). Since we know that $\bar{Y}$ corresponds to a realizable complete graph, its framework is rigid. Therefore, the set $\mathsf{D}^{-1}(\mathsf{D}(\bar{x})) \cap \bar{\chi}$ only contains realizations obtained from $\bar{x}$ by means of congruences. Equation (4) states that the framework realized by x is rigid if the realizations of the partial distance matrix of x can be obtained from x only from congruences: in other words, if it "behaves like" the framework of a complete graph.

Uniqueness of solution (modulo congruences) is sometimes a necessary feature in applications. Many different sufficient conditions to uniqueness have been found (Liberti et al. 2014, §4.1.1). By way of example as concerns the number of DGP solutions in graphs, a complete graph has at most one solution modulo congruences, as remarked above. It was proved in Liberti et al. (2013) that protein backbone graphs have a realization set having power of two cardinality with probability 1. As shown in Fig. 1 (bottom row), a cycle graph on four vertices has uncountably many solutions.

On the other hand, the remaining possibility of a countably infinite set of realizations of a DGP instance cannot happen, as shown in Theorem 1. This result is a simple corollary of a well-known theorem of Milnor (1964). It was noted informally in Liberti et al. (2014, p. 27) without details; we provide a proof here.

Theorem 1 *No DGP instance may have an infinite but countable number of solutions.*

Proof Equation (3) is a system of m quadratic equations associated with the instance graph G. Let $X \subseteq \mathbb{R}^{nK}$ be the variety associated to Eq. (3). Now, suppose X is countable: then, no connected component of X may contain uncountably many elements. By the notion of connectedness, this implies that every connected component is an isolated point in X. Since X is countable, it must contain a countable numbers of connected components. By Milnor (1964), the number of connected components of X is finite; in particular, it is bounded by $O(3^{nK})$. Hence, the number of connected components of X is finite. Since each is an isolated point, i.e., a single realization of G, $|X|$ is finite. □

3.3 Applications

The DGP is an inverse problem with many applications to science and engineering.

3.3.1 Engineering

When $K = 1$, a typical application is that of clock synchronization (Singer 2011). Network protocols for wireless sensor networks are designed so as to save power in communication. When synchronization and battery usage are key, the peer-to-peer communications needed to exchange the timestamp can be limited to the exchange of a single scalar, i.e., the time (or phase) difference. The problem is then to retrieve the absolute times of all of the clocks, given some of the phase differences. This is equivalent to a DGP on the time line, i.e., in a single dimension. We already sketched above the problem of sensor network localization (SNL) in $K \in \{2, 3\}$ dimensions. In $K = 3$, we also have the problem of controlling fleets of underwater autonomous vehicles (UAV), which requires the (fast) localization of each UAV (Bahr et al. 2009; Tabaghi et al. 2019).

3.3.2 Science

An altogether different application in $K = 3$ is the determination of protein structure from nuclear magnetic resonance (NMR) experiments (Wüthrich 1989): proteins are composed of a linear backbone and some side-chains. The backbone determines a total order on the backbone atoms, by which follow some properties of the protein backbone graph. Namely, the distances from vertex i to vertices $i - 1$ and $i - 2$ in the order are known almost exactly because of chemical information, and the distance between vertex i and vertex $i - 3$ is known approximately because of NMR output. Moreover, some other distances (with larger index difference) may also be known because of NMR—typically, when the protein folds and two atoms from different folds happen to be close to each other. If we suppose all of these distances are known exactly, we obtain a subclass of DGP which is called discretizable molecular DGP (DMDGP). The structure of the graph of a DMDGP instance is such that

vertex i is adjacent to its three immediate predecessors in the order: this yields a graph which consists of a sequence of embedded cliques on 4 vertices, the edges of which are called discretization edges, with possibly some extra edges called pruning edges.

If we had to realize this graph with $K = 2$, we could use trilateration (Eren et al. 2004): given three points in the plane, compute the position of a fourth point at known distance from the three given points. Trilateration gives rise to a system of equations which has either no solution (if the distance values are not a metric) or a unique solution, since three distances in two dimensions are enough to disambiguate translations, rotations, and reflections. Due to the specific nature of the DMDGP graph structure, it would suffice to know the positions of the first three vertices in the order to be able to recursively compute the positions of all other vertices. With $K = 3$, however, there remains one degree of freedom which yields an uncertainty: the reflection.

We can still devise a combinatorial algorithm which, instead of finding a unique solution in $n - K$ trilateration steps, is endowed with back-tracking over reflections. Thus, the DMDGP can be solved completely (meaning that all incongruent solutions can be found) in worst-case exponential time using the branch-and-prune (BP) algorithm (Liberti et al. 2008). The DMDGP has other very interesting symmetry properties (Liberti et al. 2014), which allow for an a priori computation of its number of solutions (Liberti et al. 2013), as well as for generating all of the incongruent solutions from any one of them (Mucherino et al. 2012); moreover, it turns out that BP is a fixed-parameter tractable (FPT) algorithm, which makes the DMDGP a FPT problem (Liberti et al. 2013).

3.3.3 Machine learning

So far, we have only listed applications where K is fixed by the constraints of physical space. The focus of this survey, however, is a case where K may vary according to the data: if we need to map graphs to vectors in view of preprocessing the input of an ML methodology, we may choose a dimension K appropriate to the methodology and application at hand. See Sect. 9 for an example.

3.4 Complexity

3.4.1 Membership in NP

The DGP is clearly a decision problem, so one may ask whether it is in NP. As stated above, with real number input in the edge weight function, it is clear that it is not, since the Turing computation model cannot be applied. We therefore consider its rational equivalent, where $d : E \to \mathbb{Q}_+$, and ask the same question. It turns out that, for $K > 1$, we do not know whether the DGP is in NP: the issue is that the solutions of sets of quadratic polynomials over $\mathbb{Q}$ may well be (algebraic) irrational. We, therefore, have the problem of establishing that a realization matrix x with algebraic components satisfies Eq. (3) in polynomial time. While some compact representations of

algebraic numbers exist (Liberti 2019, §2.3), it is not known how to employ them in the polynomial time verification of Eq. (3). Negative results for the most basic representations of algebraic numbers were derived in Beeker et al. (2013).

On the other hand, it is known that the DGP is in NP for $K = 1$: as this case reduces to realizing graphs on a single real line, the fact that all of the given distances are in $\mathbb{Q}$ means that the distance between any two points on the line is rational: therefore, if one point is rational, then all the others can be obtained as sums and differences of this one point and a set of rational values, which implies that there is always a rational realization. Naturally, verifying whether a rational realization satisfies Eq. (3) can be carried out in polynomial time.

3.4.2 NP-hardness

It was proved in Saxe (1979) that the DGP is NP-hard, even for $K = 1$ by reduction from Partition to the DGP on simple cycle graphs, see a detailed proof in Liberti and Lavor (2017, §2.4.2). Hence it is actually NP-complete for $K = 1$. In the same paper (Saxe 1979), using more complicated gadgets, it was also shown that the DGP is NP-hard for each fixed K and with edge weights restricted to taking values in $\{1, 2\}$ (reduction from 3SAT).

A sketch of an adaptation of the reduction to cycle graphs is given in Yemini (1979) for DMDGP graphs, showing that they are an NP-hard subclass of the DGP. A full proof following a similar idea can be found in Lavor et al. (2012).

4 Representing data by graphs

It may be obvious to most readers that data can be naturally represented by graphs. This is immediately evident whenever data represent similarities or dissimilarities between entities in a vertex set V. In this section we make this intuition more explicit for a number of other relevant cases.

An unweighted graph represents a binary relation on the entities represented by vertices: u, v are related if and only if the edge $\{u, v\}$ is in the graph. Scalar weights assigned to edges can measure the strength or weakness of the relation; edge colors encode a discrete attribute of the relation; other numeric or symbolic weight types are used to encode other relation attributes. Parallel edges can be used to define different relations on the same set of entities. Edge weights are often used to represent distance (as in the DGP), similarity (the larger the weight, the more similar), and dissimilarity (the larger the weight, the less similar) between pairs of entities. Similarity/dissimilarity weights are often normalized to range in [0, 1].

4.1 Processes

The description of a process, be it chemical, electric/electronic, mechanical, computational, logical, or otherwise, is practically always based on a directed graph, or

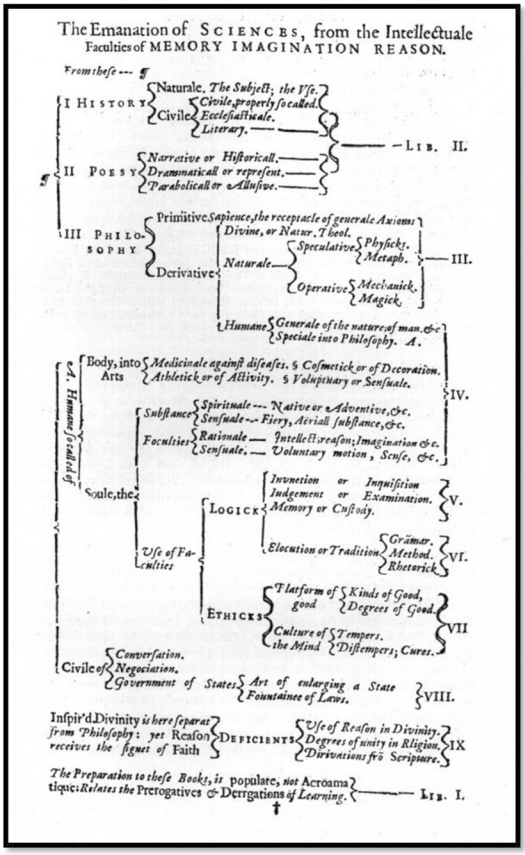

Fig. 2 A tree diagram from F. Bacon's *Advancement of Learning*, Oxford 1640

digraph, $G = (N, A)$. The set of nodes N represents the various stages of the process, while the arcs in A represent transitions between stages.

Formalizations of this concept may possibly be first ascribed to the organization of knowledge proposed by Aristotle into genera and differences, commonly represented with a tree (a class of digraphs). While no graphical representation of this tree ever came to us from Aristotelian times, the commentator Porphyry of Tyre (third century AD) did refer to a representation which was actually drawn as a tree (at least since the tenth century Verboon 2014). Many interesting images can be found in http://last-tree.scottbot.net/illustrations/, see e.g. Fig. 2.

A general treatment of process diagrams in mechanical engineering is given in Gilbreth and Gilbreth (1921). Bipartite graphs with two node classes representing operations and materials have been used in process network synthesis in chemical engineering (Friedler et al. 1992). Circuit diagrams are a necessary design tool for any electrical and electronic circuit (Seshu and Reed 1961). Software flowcharts (i.e., graphical description of computer programs) have been used in the design of software so pervasively that one of the most important results in computer science, namely the Böhm–Jacopini's theorem on the expressiveness of universal computer

languages, is based on a formalization of the concept of flowchart (Böhm and Jacopini 1966). The American National Standards Institute (ANSI) set standards for flowcharts and their symbols in the 1960s. The International Organization for Standardization (ISO) adopted the ANSI symbols in 1970 (Wikipedia: Flowchart 2019). The cyclomatic number $|E| - |V| + 1$ of a graph, namely the size of a cycle basis of the cycle space, was adopted as a measure of process graph complexity very early (see Paton 1969; Deo et al. 1982; Brambilla and Premoli 2001; Amaldi et al. 2009 and Knuth 1997, §2.3.4.1).

An evaluation of flowcharts to process design is the unified modelling language (UML) (Object Management Group 2005), which was mainly conceived to aid the design of software-based systems, but was soon extended to much more general processes. With respect to flowcharts, UML also models interactions between software systems and hardware systems, as well as with system users and stakeholders. When it is applied to software, UML is a semi-formal language, in the sense that it can automatically produce a set of header files with the description of classes and other objects, ready for code development in a variety of programming languages (Liberti 2010).

4.2 Text

One of the foremost issues in linguistics is the formalization of the rules of grammar in natural languages. On one hand, text is scanned linearly, word by word. On the other hand, the sense of a sentence becomes apparent only when sentences are organized as trees (Chomsky 1965). This is immediately evident in the computer parsing of formal languages, with a "lexer" which carries out the linear scanning, and a "parser" which organizes the lexical tokens in a parsing tree (Levine et al. 1995). The situation is much more complicated for natural languages, where no rule of grammar is ever absolute, and any proposal for overarching principles has so many exceptions that it is hard to argue in their favor (Moro 2008).

The study of natural languages is usually split into syntax (how the sentence is organized), semantics (the sense conveyed by the sentence), and pragmatics (how the context when the sentence is uttered influences the meaning, and the impact that the uttered sentence has on the context itself) (Morris 1946). The current situation is that we have been able to formalize rules for natural language syntax (namely turning a linear text string into a parsing tree) fairly well, using probabilistic parsers (Manning and Schütze 1999) as well as supervised ML (Collobert et al. 2011). We are still far from being able to successfully formalize semantics. Semiotics suggested many ways to assign semantics to sentences (Eco 1984), but none of these is immediately and easily implementable as a computer program.

Two particularly promising suggestions are the organization of knowledge into an evolving encyclopedia, and the representation of the sense of words in a "space" with "semantic axes" (e.g., "good/bad", "white/black", and "left/right"...). The first suggestion yielded organized corpora such as Miller (1995), which is a tree representation of words, synonyms, and their semantical relations, not unrelated to a Porphyrian tree (Sect. 4.1). There is still a long way to go before the second is

successfully implemented, but we see in the Google Word Vectors (Mikolov et al. 2013) the start of a promising path. On the other hand some easy semantical interpretations, such as analogies, are apparently not so well preserved in these word vectors despite the publicity (Khalife et al. 2019).

For pragmatics, the situation is even more dire; some suggestions for representing knowledge and cognition w.r.t. the state of the world are given in Minsky (1986). See Wikipedia: Computational pragmatics (2019) for more information.

Insofar as graphs are concerned, syntax is organized into tree graphs, and semantics is often organized in corpora that are also trees, or directed acyclic graphs (DAGs), e.g., WordNet and similar.

4.2.1 Graph-of-words

In Sect. 9 we will consider a graph representation of sentences known as the *graph-of-words* (Rousseau and Vazirgiannis 2013). Given a sentence s represented as a sequence of words $s = (s_1, \dots, s_m)$, an n-gram is a subsequence of n consecutive words of s. Each sentence obviously has at most $(m - n + 1)$ n-grams. In a graph-of-words $G = (V, E)$ of order n, V is the set of words in s; two words have an edge only if they appear in the same n-gram; the weight of the edge is equal to the number of n-grams in which the two words appear. This graph may also be enriched with semantic relations between the words, obtained, e.g., from WordNet.

4.3 Databases

The most common form of data collection is a database; among the existing database representations, one of the most popular is the tabular form used in spreadsheets and relational databases.

A *table* is a rectangular array A with n rows (the records) and m columns (the features), which is (possibly only partially) filled with values. Specifically, each feature column must have values of the same type (when present). If A_{rf} is filled with a value, we denote this $\mathsf{def}(r, f)$, for each record index r and feature index f. We can represent this array via a bipartite graph $B = (R, F, E)$ where R is the set of record indices, F is the set of feature indices, and there is an edge $\{r, f\} \in E$ if the (r, f) th component A_{rf} of A is filled. A label function ℓ assigns the value A_{rf} to the edge $\{r, f\}$. While this is an edge-labelled graph, the labels (i.e., the contents of A) may not always be interpretable as edge weights—so this representation is not yet what we are looking for.

We now assume that there is a symmetric function $d_f : A_{\cdot f} \times A_{\cdot f} \to \mathbb{R}_+$ defined over elements of the column $A_{\cdot f}$: since all elements in a column have the same type, such functions can always be defined in practice. We note that d_f is undefined whenever one of the two arguments is not filled with a value. We can then define a composite function $d : R \times R \to \mathbb{R}_+$ as follows:

$$\forall r \neq s \in R \quad d(r,s) = \begin{cases} \sum\limits_{f \in F} d_f(A_{rf}, A_{sf}) \\ \mathsf{def}(r,f) \wedge \mathsf{def}(s,f) \\ \text{undefined if } \exists f \in F\ (\neg\mathsf{def}(r,f) \vee \neg\mathsf{def}(s,f)). \end{cases} \tag{5}$$

Next, we define a graph $G = (R, E')$ over the records R, where

$$E' = \{\{r,s\} \mid r \neq s \in R \wedge d(r,s) \text{ is defined}\},$$

weighted by the function $d : E' \to \mathbb{R}_+$ defined in Eq. (5). We call G the database distance graph. Analysing this graph yields insights about record distributions, similarity, and differences.

4.4 Abductive inference

According to Eco (1983), there are three main modes of rational thought, corresponding to three different permutations of the concepts "hypothesis" (call this H), "prediction" (call this P), and "observation" (call this O). Each of the three permutations singles out a pair of concepts and a remaining concept. Specifically:

1. deduction: H $\wedge$ P $\to$ O;
2. (scientific) induction: O $\wedge$ P $\to$ H;
3. abduction: H $\wedge$ O $\to$ P.

Take, for example, the most famous syllogism about Socrates being mortal:

– H: "all humans are mortal";
– P: "Socrates is human";
– O: "Socrates is mortal".

The syllogism is an example of deduction: we are given H and P, and deduce O. Note also that deduction is related to modus ponens: if we let $A(x)$ be the sentence "x is human" and $B(x)$ be the sentence "x is mortal", and let s be the constant denoting Socrates, the syllogism can be restated as:

$$[\forall x\ (A(x) \to B(x)) \wedge A(s)] \to B(s).$$

Deduction infers truths (propositional logic) or provable sentences (first-order and higher order logic), and is mostly used by logicians and mathematicians.

Scientific induction[1] exploits observations and verifies predictions to derive a general hypothesis: if a large quantity of predictions is verified, a general hypothesis can be formulated. In other words, given O and P we infer H. Scientific induction can never provide proofs in sufficiently expressive logical universes, no matter the amount of observations and verified predictions. Any false prediction, however,

[1] Not to be confused with mathematical induction.

disproves the hypothesis (Popper 1968). Scientific induction is about causality; it is mostly used by physicists and other natural scientists.

Abduction (Douven 2017) infers educated guesses about a likely state of a known universe from observed facts: given H and O, we infer P. Following (McCulloch 1961),

> Deductions lead from rules and cases to facts—the conclusions. Inductions lead toward truth, with less assurance, from cases and facts, toward rules as generalizations, valid for bound cases, not for accidents. Abductions, the apagoge of Aristotle, lead from rules and facts to the hypothesis that the fact is a case under the rule.

According to Eco (1983), abduction can be traced back to Peirce (1878), who cited Aristotle as a source. The author of Proni (2016) argues that the precise Aristotelian source cited by Peirce fails to make a valid reference to abduction; however, he also concedes that there are some forms of abduction foreshadowed by Aristotle in the texts where he defines definitions.

Let us see an example of abduction. Sherlock Holmes is called on a crime scene where Socrates lies dead on his bed. After much evidence is collected and a full-scale investigation is launched, Holmes ponders some possible hypotheses: for example, all rocks are dead. The prediction that is logically consistent with this hypothesis and the observation that Socrates is dead would be that Socrates is a rock. After some unsuccessful tests using Socrates' remains as a typical rock, Holmes eliminates this possibility. Following a few more untenable suggestions by Dr. Watson, Holmes considers the hypothesis that all humans are mortal. The logically consistent prediction is that Socrates is a man, which, in a dazzling display of investigative abilities, Holmes finds it to be exactly the case. Thus, Holmes brilliantly solves the mystery, while Inspector Lestrade was just about ready to give up in despair. Abduction is about plausibility; it is the most common type of human inference.

Abduction and scientific induction are the basis of learning: after witnessing a set of facts, and postulating hypotheses for relate them, we are able to make and then verify predictions about the future. Obviously, abductions can, and in fact often turn out to, be wrong, e.g.:

- H: all beans in the bag are white;
- O: there is a white bean next to the bag;
- P: the bean was in the bag.

The white bean next to the bag, however, might have been placed there before the bag was even in sight. With this last example, we note that abductions are inferences often used in statistics. For an observation O, a set $\mathcal{H}$ of hypotheses and a set of possible predictions $\mathcal{P}$, we must evaluate the probability,

$$\forall \, \mathrm{H} \in \mathcal{H}, \; \mathrm{P} \in \mathcal{P} \quad p_{\,\mathrm{HP}} = \mathsf{P}(\, \mathrm{O} \mid \mathrm{O}, \mathrm{H} \;\; \text{abduce} \;\; \mathrm{P}\,),$$

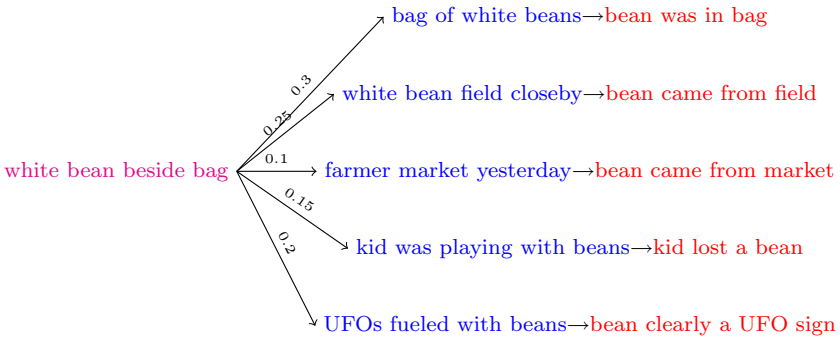

Fig. 3 Evaluating probabilities in abduction. From left to right, observation O abduces the inference H→P

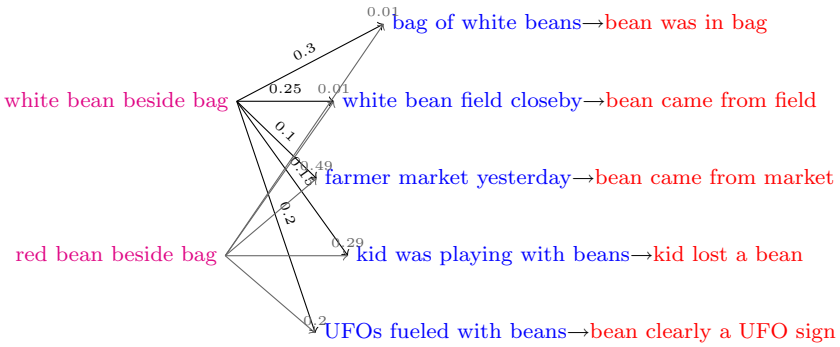

Fig. 4 Probability distributions over abduction inferences assigned to observations

and then choose the pair (H,P) having largest probability p_{HP} (see a simplified example in Fig. 3).

When more than one observation is collected, one can also compare distributions to make more plausible predictions, see Fig. 4. Abduction appears close to the kind of analysis often required by data scientists.

4.4.1 The abduction graph

We now propose a protocol for modelling good predictions from data, by means of an abduction graph. We consider:

– a set $\mathcal{O}$ of observations O;
– a set $\mathcal{I} \subseteq \mathcal{H} \times \mathcal{P}$ of abductive premises, namely pairs (H, P).

First, we note that different elements of $\mathcal{I}$ might be logically incompatible (e.g., there may be contradictory sets of hypotheses or predictions). We must therefore extract

a large set of logically compatible subsets of $\mathcal{I}$. Consider the relation $\sim$ on $\mathcal{I}$ with $h \sim k$, meaning that $h, k \in \mathcal{I}$ are logically compatible. This defines a graph $(\mathcal{I}, \sim)$. We then find the largest (or at least large enough) clique $\bar{\mathcal{I}}$ in $(\mathcal{I}, \sim)$.

Next, we define probability distributions p^{O} on $\bar{\mathcal{I}}$ for each $\mathrm{O} \in \mathcal{O}$. We let $E = \{\{\mathrm{O}, \mathrm{O}'\} \mid \delta(p^{\mathrm{O}}, p^{\mathrm{O}'}) \leq \delta_0\}$, where δ evaluates dissimilarities between probability distributions, e.g., δ could be the Kullback–Leibler (KL) divergence (Kullback and Leibler 1951), and δ_0 a given threshold. Thus, E defines a relation on $\mathcal{O}$ if $p^{\mathrm{O}}, p^{\mathrm{O}'}$ are sufficiently similar. We can finally define the graph $\mathcal{F} = (\mathcal{O}, E)$, with edges weighted by δ.

If we think of Sherlock Holmes again, the abduction graph encodes sets of clues compatible with the most likely consistent explanations.

5 Common data science tasks

DS refers to practically every task or problem defined over large amounts of data. Even problems in P, and sometimes even those for which there exist linear time algorithms, may take too long when confronted with large-scale instances. We are not going to concern ourselves here with evaluation problems (such as, e.g., computing means or variances—which can be a daunting task for extremely large datasets), but rather with decision problems. In particular, it appears that a very common family of decision problems solved on large masses of data are those that help people make sense of the data themselves: in other words, classification and clustering.

There is no real functional distinction between the two, as both aim at partitioning the data into a relatively small number of subsets. However, "classification" usually refers to the problem of assigning class labels to data elements, while "clustering" indicates a classification based on the concept of similarity or distance, meaning that similar data elements should be in the same class. This difference is usually more evident in the algorithmic description: classification methods tend to exploit information inherent to elements, while clustering methods consider information relative to pairs of elements. It also appears that the term "clustering" is used in unsupervised learning, whereas "classification" is more often used in supervised learning. In the rest of this paper, we shall adopt a functional view, and refer to either interchangeably.

Given a set P of n entities and some pairwise similarity function $\delta : P \times P \to \mathbb{R}_+$, clustering aims at finding a set of k subsets $C_1, \ldots, C_k \subseteq P$ (with their union covering P) such that each cluster contains as many similar entities, and as few dissimilar entities, as possible. Cluster analysis—as a field—grew out of statistics in the course of the second half of the 20th century, encouraged by the advances in computing power. However, some early forms of cluster analysis may also be attributed to earlier scientists, e.g. Aristotle, Buffon, Cuvier, and Linné (Hansen and Jaumard 1997).

We note that "clustering on graphs" may refer to two separate tasks.

A. Cluster the vertices of a given graph.
B. Cluster the graphs in a given set.

Both may arise depending on the application at hand. The proposed DG techniques for realizing graphs into vector spaces apply to both of these tasks (see Sect. 9.4.2).

As mentioned above, this paper focuses on transforming graphs into vectors so as to be able to use vector-based methods for classification and clustering. We shall first survey some of these methods. We shall then mention some methods for classifying/clustering graphs directly (i.e., without needing to transform them into vectors first).

5.1 Clustering on vectors

Methods for classification and clustering on vectors are usually seen as part of ML. They are partitioned into unsupervised and supervised learning methods. The former are usually based on some measure of similarity or dissimilarity defined over pairs of elements. The latter require a training set, which they exploit to find a set of optimal parameter values for a parametrized "model" of the data.

5.1.1 The k-means algorithm

The k-means algorithm is a well-known heuristic for solving the following problem (Aloise et al. 2012).

Minimum sum-of-squares clustering (MSSC). Given an integer $k > 0$ and a set $P \subset \mathbb{R}^m$ of n vectors, find a cover $\mathcal{C} = \{C_1, \ldots, C_k\}$ of P such that the function

$$f(\mathcal{C}) = \sum_{j \le k} \sum_{x \in C_j} \|x - \mathsf{centroid}(C_j)\|_2^2 \tag{6}$$

is minimum, where

$$\mathsf{centroid}(C_j) = \frac{1}{|C_j|} \sum_{x \in C_j} x. \tag{7}$$

It is interesting to note that the MSSC problem can also be seen as a discrete analogue of the problem of partitioning a body into smaller bodies having minimum sum of moments of inertia (Steinhaus 1956).

The k-means algorithm improves a given initial clustering $\mathcal{C}$ by means of the two following operations:

1. compute centroids $c_j = \mathsf{centroid}(C_j)$ for each $j \le k$;
2. for any pair of clusters $C_h, C_j \in \mathcal{C}$ and any point $x \in C_h$, if x is closer to c_j than to c_h, move x from C_h to C_j.

These two operations are repeated until the clustering $\mathcal{C}$ no longer changes. Since the only decision operation (i.e., operation 2) is effective only if it decreases $f(\mathcal{C})$, it follows that k-means is a local descent algorithm. In particular, this very simple analysis offers no guarantee on the approximation of the objective function. For more information on the k-means algorithm, see (Blömer et al. 2016).

The k-means algorithm is an unsupervised learning technique (Jain et al. 1999), insofar as it does not rest on a data model with parameters to be estimated prior to actually finding clusters. Moreover, the number "k" of clusters must be known a priori.

5.1.2 Artificial neural networks

An ANN is a parametrized model for representing an unknown function. Like all such models, it needs data to estimate suitable values for the parameters: this puts ANNs in the category of supervised ML. An ANN consists of two MP formulations defined over a graph and a training set.

An ANN is formally defined as a triplet $\mathcal{N} = (G, T, \phi)$, where:

- $G = (V, A)$ is a directed graph, with a node weight function $b : V \to \mathbb{R}$ (threshold at a node), and an edge weight function $w : A \to \mathbb{R}$ (weight on an arc); moreover, a subset $I \subset V$ of input nodes with $|I| = n$ and a subset $O \subset V$ of output nodes with $|O| = k$ are given in G;
- $T = (X, Y)$ is the training set, where $X \subset \mathbb{R}^n$ (input set), $Y \subset \mathbb{R}^k$ (output set), and $|X| = |Y|$;
- $\phi = (\phi_j \mid j \in V \setminus I)$ is a sequence of activation functions $\phi_j : \mathbb{R} \to \mathbb{R}$ (many common activation functions map injectively into [0, 1]).

The two MP formulations assigned to an ANN describe the training problem and the evaluation problem. In the training problem, appropriate values for b, w are found using T. In the evaluation problem, a given input vector in $\mathbb{R}^n$ (usually not part of the input training set X) is mapped to an output vector in $\mathbb{R}^k$. The training problem decides values for the ANN parameters when seen as a model for an unknown function mapping the training input X to the training output Y. After the model is trained, it can be evaluated on new (unseen) input.

For a node $i \in V$, we let $N^-(i) = \{j \in V \mid (j, i) \in A\}$ be the inward star of i. For a tensor $s_{i_1,\ldots,i_r}$, where $i_j \in I_j$ for each $j \leq r$, we denote a slice of s, defined by subsets $J_j \subseteq I_j$ for some $j \leq r$, by $s[J_1] \cdots [J_r]$.

We discuss the evaluation phase first. Given values for w, b and an input vector $x \in \mathbb{R}^n$, we decide a node weight function u over V as follows:

$$u_I = x \tag{8}$$

$$\forall j \in V \setminus I \quad u_j = \phi_j\left(\sum_{i \in N^-(j)} w_{ij} u_i + b_j\right). \tag{9}$$

We remark that Eq. (9) is not an optimization but a decision problem. Nonetheless, it is an MP formulation (formally with zero objective function). After solving Eq. (9), one retrieves in particular $u[O]$, which correspond to an output vector in $u[O] = y \in \mathbb{R}^k$. When G is acyclic, this decision problem reduces to a simple computation, which "propagates" the values of u from the input nodes and forward

through the network until they reach the output nodes. If G is not acyclic, different solution methods must be used (Anderson 1995; Floreano 1996; Goodfellow et al. 2016).

The training problem is given in Eq. (10). We let N be the index set for the training pairs (x, y) in T (we recall that $|X| = |Y|$), and introduce a two-dimensional tensor v of decision variables indexed by N and V:

$$\left.\begin{array}{l} \min\limits_{w,b,v} \mathsf{dist}(v[N][O], Y) \\ v[N][I] = X \\ \forall t \in N, j \in V \smallsetminus I \qquad v_{tj} = \phi_j\left(\sum\limits_{i \in N^-(j)} w_{ij} v_{ti} + b_j\right), \end{array}\right\} \tag{10}$$

where $\mathsf{dist}(A, B)$ is a dissimilarity function taking dimensionally consistent tensor arguments A, B, which becomes closer to zero as A and B get closer. The solution of the training problem yields optimal values w^*, b^* for the arc weights and node biases.

The training problem is, in general, a nonconvex optimization problem (because of the products between w and v, and of the ϕ functions occurring in equations), which may have multiple global optima: finding them with state-of-the-art methods might require exponential time. For specific types of graphs and choices of objective function $\mathsf{dist}(\cdot, \cdot)$, the training problem may turn out to be convex. For example, if: (a) G is a DAG, (b) $V = I \dot\cup O$ is the disjoint union of I and O, (c) the induced subgraphs $G[I]$ and $G[O]$ are empty (i.e., they have no arcs), (d) the activation functions are all sigmoids $\phi(z) = (1 + \exp(-z))^{-1}$, and (e) $\mathsf{dist}(\cdot, \cdot)$ is the negative logarithm of the likelihood function:

$$\prod_{t \in N} \phi\left(w^\top x_t + b_i\right)^{y_t} \left(1 - \phi\left(w^\top x_t + b_i\right)\right)^{1 - y_t},$$

(where $X = (x_t \mid t \in N)$ is the list of input training vectors) summed over all output nodes $i \in O$, then it can be shown that the training problem is convex (Jordan 1995; Schumacher et al. 1996).

In contemporary treatments of ANNs, the underlying graph G is almost always assumed to be a DAG. In modern application programming interfaces (API), the acyclicity of G is enforced by recursively replacing v_{tj} with the corresponding expression in $\phi(\cdot)$.

Most algorithms usually solve Eq. (10) only locally and approximately. Usually, they employ a technique called stochastic gradient descent (SGD) (Bottou 2012), which can be applied after the constraints of Eq. (10) have been relaxed and added as penalty terms to the objective function. This is a form of gradient descent where, at each iteration, the gradient of a multivariate function is estimated by partial gradients with respect to a randomly chosen subset of variables (Moitra 2018, p. 100).

The functional definition of an optimum for the training problem Eq. (10) is poorly understood, as finding precise local (or global) optima is considered "overfitting". In other words, global or almost global optima of Eq. (10) lead to evaluations

which are possibly perfect for pairs in the training set, but unsatisfactory for yet unseen input. Currently, finding "good" optima of ANN training problems is mostly based on experience, although a considerable effort is under way to reach a sound definition of optimum (Dauphin et al. 2014; Yun et al. 2018; Haeffele and Vidal 2017; Choromanska et al. 2015).

The main reason why ANNs are so popular today is that they have proven hugely successful at image recognition (Goodfellow et al. 2016), and also extremely good at accomplishing other tasks, including natural language processing (Collobert et al. 2011). Many efficient applications of ANNs to complex tasks involve interconnected networks of ANNs of many different types (Bengio et al. 2007).

ANNs originated from an attempt to simulate neuronal activity in the brain: should the attempt prove successful, it would realize the old human dream of endowing a machine with human intelligence (ben Judah of Worms XII-XIII Century). While ANNs today display higher precision than humans in some image recognition tasks, they may also be easily fooled by a few appropriately positioned pixels of different colors, which places the realization of "human machine intelligence" still rather far in the future—or even unreachable, e.g., if Penrose's hypothesis of quantum activity in the brain influencing intelligence at a macroscopic level holds (Penrose 1989). For more information about ANNs, see Schmidhuber (2015) and Goodfellow et al. (2016).

5.2 Clustering on graphs

While we argue in this paper that DG techniques allow the use of vector clustering methods to graph clustering, there also exist methods for clustering on graphs directly. We discuss two of them, both applicable to the task of clustering vertices of a given graph (Task A on p. 20).

5.2.1 Spectral clustering

Consider a connected graph $G = (V, E)$ with an edge weight function $w : E \to \mathbb{R}_+$. Let A be the adjacency matrix of G, with $A_{ij} = w_{ij}$ for all $\{i, j\} \in E$, and $A_{ij} = 0$ otherwise. Let Δ be the diagonal weighted degree matrix of G, with $\Delta_{ii} = \sum_{j \neq i} A_{ij}$ and $\Delta_{ij} = 0$ for all $i \neq j$. The Laplacian of G is defined as $L = \Delta - A$.

Spectral clustering aims at finding a minimum balanced cut $U \subset V$ in G by looking at the spectrum of the Laplacian of G. For now, we give the word "balanced" only an informal meaning: it indicates the fact that we would like clusters to have approximately the same cardinality (we shall be more precise below). Removing the cutset $\delta(U)$ (i.e., the set of edges between U and $V \setminus U$) from G yields a two-way partitioning of V. If $|\delta(U)|$ is minimum over all possible cuts U, then the two sets $U, V \setminus U$ should both intuitively induce subgraphs $G[U]$ and $G[V \setminus U]$ having more edges than those in $\delta(U)$. In other words, the criterion which we are interested in maximizes the intra-cluster edges of the subgraphs of G induced by the cluster while minimizing the inter-cluster edges of the corresponding cutsets.

We remark that each of the two partitions can be recursively partitioned again. A recursive clustering by two-way partitioning is a general methodology which is part of a family of hierarchical clustering methods (Schaeffer 2007). Therefore, the scope of this section is not limited to generating two clusters only.

For simplicity, we only discuss the case with unit edge weights, although the generalization to general weights is not problematic. Thus, Δ_{ii} is the degree of vertex $i \in V$. We model a balanced partition $\{B, C\}$ corresponding to a minimum cut by means of decision variables $x_i = 1$ if $i \in B$ and $x_i = -1$ if $i \in C$, for each $i \leq n$, with $n = |V|$. Then $f(x) = \frac{1}{4}\sum_{\{i,j\}\in E}(x_i - x_j)^2$ counts the number of intercluster edges between B and C. We have:

$$\begin{aligned} 4f(x) &= \sum_{\{i,j\}\in E}(x_i^2 + x_j^2) - 2\sum_{\{i,j\}\in E} x_i x_j = \sum_{\{i,j\}\in E} 2 - \sum_{i,j\leq n} x_i A_{ij} x_j \\ &= 2|E| - x^\top A x = \sum_{i\leq n} x_i \Delta_{ii} x_i - x^\top A x = x^\top(\Delta - A)x = x^\top L x, \end{aligned}$$

whence $f(x) = \frac{1}{4}x^\top L x$. We can therefore obtain cuts with minimum $|\delta(B)|$ by minimizing $f(x)$.

We can now give a more precise meaning to the requirement that partitions are balanced: we require that x must satisfy the constraint:

$$\sum_{i\leq n} x_i = 0. \tag{11}$$

Obviously, Eq. (11) only ensures equal cardinality partitions on graphs having an even number of vertices. However, we relax the integrality constraints $x \in \{-1, 1\}^n$ to $x \in [-1, 1]^n$, so $\sum_{i\leq n} x_i = 0$ is applicable to any graph. With this relaxation, the values of x might be fractional. We shall deal with this issue by rounding them to $\{-1, 1\}$ after obtaining the solution. We also note that the constraint:

$$x^\top x = \|x\|_2^2 = n \tag{12}$$

holds for $x \in \{-1, 1\}^n$, and so it provides a strengthening of the continuous relaxation to $x \in [-1, 1]^n$. We therefore obtain a relaxed formulation of the minimum balanced two-way partitioning problem as follows:

$$\left.\begin{array}{ll} \min\limits_{x\in[-1,1]^n} & \frac{1}{4}x^\top L x \\ \text{s.t.} & \mathbf{1}^\top x = 0 \\ & \|x\|_2^2 = n, \end{array}\right\} \tag{13}$$

where $\mathbf{1}$ is the all-one vector. We remark that, by construction, L is a *diagonally dominant* (dd) symmetric matrix with non-negative diagonal, namely it satisfies:

$$\forall i \leq n \quad L_{ii} \geq \sum_{j\neq i} |L_{ij}|; \tag{14}$$

(in fact, L satisfies Eq. (14) at equality). Since all dd matrices are also psd (Wikipedia: Diagonally dominant matrix 2019), $f(x)$ is a convex function. This means that Eq. (13) is a cQP, which can be solved at global optimality in polynomial time (Vavasis 1991).

By Fiedler (1973), there is another polynomial time method for solving Eq. (14), which is generally more efficient than solving a cQP in polynomial time using a nonlinear programming (NLP) solver. This method concerns the second-smallest eigenvalue of L (called algebraic connectivity) and its corresponding eigenvector. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the ordered eigenvalues of L and $u_1, \ldots, u_n$ be the corresponding eigenvectors, normalized so that $\|u_i\|_2^2 = n$ for all $i \leq n$. It is known that $u_1 = \mathbf{1}$, $\lambda_1 = 0$ and, if G is connected, $\lambda_2 > 0$ (Merris 1994; Bollobás 1998). By the definition of eigenvalue and eigenvector, we have:

$$\forall i \leq n \quad Lu_i = \lambda_i u_i \quad \Rightarrow \quad u_i{}^\top L u_i = \lambda_i u_i{}^\top u_i = \lambda_i \|u_i\|_2^2 = \lambda_i n. \tag{15}$$

Because of the orthogonality of the eigenvectors, if $i \geq 2$ we have $u_i u_1 = 0$, which implies $u_2 \mathbf{1} = 0$ (i.e., u_2 satisfies Eq. (11)). We recall that eigenvectors are normalized, so that $\|u_i\|_2^2 = n$ for all $i \leq n$ (in particular, u_2 satisfies Eq. (12)). By Eq. (15), since $\lambda_1 = 0$, λ_2 yields the smallest nontrivial objective function value $\frac{n}{4}\lambda_2$ with solution $\bar{x} = u_2$, which is therefore a solution of Eq. (13).

Theorem 2 *The eigenvector u_2 corresponding to the second smallest eigenvalue λ_2 of the graph Laplacian L is an optimal solution to Eq. (13).*

Proof Since the eigenvectors $u_1, \ldots, u_n$ are an orthogonal basis of $\mathbb{R}^n$, we can express an optimal solution as $\bar{x} = \sum_i c_i u_i$. Thus:

$$\bar{x}^\top L \bar{x} = \sum_{i,j} c_i c_j u_i^\top L u_j = \sum_{i,j} c_i c_j \lambda_j u_i^\top u_j = n \sum_{i>1} c_i^2 \lambda_i. \tag{16}$$

The last equality in Eq. (16) follows, because $Lu_i = \lambda_i u_i$ for all $i \leq n$, $u_i^\top u_j = 0$ for each $i \neq j$, and $\lambda_1 = 0$. Since $u_1 = \mathbf{1}$ and by eigenvector orthogonality, letting $\mathbf{1}^\top \bar{x} = 0$ yields $c_1 = 0$. Finally, requiring $\|\bar{x}\|_2 = n$, again by eigenvector orthogonality, yields:

$$\begin{aligned} \left\| \sum_{i>1} c_i u_i \right\|_2^2 &= \left\langle \sum_{i>1} c_i u_i, \sum_{j>1} c_j u_j \right\rangle = \sum_{i,j>1} c_i c_j \langle u_i, u_j \rangle \\ &= \sum_{i>1} c_i^2 \|u_i\|_2^2 = n \sum_{i>1} c_i^2 = n. \end{aligned} \tag{17}$$

After replacing c_i^2 by y_i in Eqs. (16) and (17), we can reformulate Eq. (13) as:

$$n \min \left\{ \sum_{i>1} \lambda_i y_i \mid \sum_{i>1} y_i = 1 \wedge y \geq 0 \right\},$$

which is equivalent to finding the convex combination of $\lambda_2, \ldots, \lambda_n$ with smallest value. Since $\lambda_2 \leq \lambda_i$ for all $i > 2$, the smallest value is achieved at $y_2 = 1$ and $y_i = 0$ for all $i > 2$. Hence, $\bar{x} = u_2$ as claimed. □

Normally, the components of $\bar{x}$ obtained this way are not in $\{-1, 1\}$. We round $\bar{x}_i$ to its closest value in $\{-1, 1\}$, breaking ties in such a way as to keep the bisection balanced. We then obtain a practically efficient approximation of the minimum balanced cut.

5.2.2 Modularity clustering

Modularity, first introduced in Newman and Girvan (2004), is a measure for evaluating the quality of a clustering of vertices in a graph $G = (V, E)$ with a weight function $w : E \to \mathbb{R}_+$ on the edges. We let $n = |V|$ and $m = |E|$. Given a vertex clustering $\mathcal{C} = (C_1, \ldots, C_k)$, where each $C_i \subseteq V$, $C_i \cap C_j = \emptyset$ for each $i \neq j$, and $\bigcup_i C_i = V$, the modularity of $\mathcal{C}$ is the proportion of edges in E that fall within a cluster minus the expected proportion of the same quantity if edges were distributed at random while keeping the vertex degrees constant. This definition is not so easy to understand, so we shall assume for simplicity that $w_{uv} = 1$ for all $\{u, v\} \in E$ and $w_{uv} = 0$ otherwise. We give a more formal definition of modularity, and comment on its construction.

The "fraction of the edges that fall within a cluster" is:

$$\frac{1}{m} \sum_{i \leq k} \sum_{\substack{u, v \in C_i \\ \{u, v\} \in E}} 1 = \frac{1}{2m} \sum_{\substack{i \leq k \\ (u, v) \in (C_i)^2}} w_{uv},$$

where $w_{uv} = w_{vu}$ turns out to be the (u, v)th component of the $n \times n$ symmetric incidence matrix of the edge set E in $V \times V$—thus, we divide by $2m$ rather than m in the right hand side (RHS) of the above equation. The "same quantity if edges were distributed at random while keeping the vertex degrees constant" is the probability that a pair of vertices u, v belongs to the edge set of a random graph on V. If we were computing this probability over random graphs sampled uniformly over all graphs on V with m edges, this probability would be $1/m$; but since we only want to consider graphs with the same degree sequence as G, the probability is $\frac{|N(u)|\,|N(v)|}{2m}$ (Lehmann and Hansen 2007). Here is an informal explanation: given vertices u, v, there are $k_u = |N(u)|$ "half-edges" out of u, and $k_v = |N(v)|$ out of v, which could come together to form an edge between u and v (over a total of $2m$ "half-edges"). Thus, we obtain a modularity:

$$\mu(\mathcal{C}) = \frac{1}{2m} \sum_{\substack{(u, v) \in C^2 \\ C \in \mathcal{C}}} \left(w_{uv} - k_u k_v / (2m)\right)$$

for the clustering $\mathcal{C}$.

We now introduce binary variables x_{uv} which have value 1 if $u, v \in V$ are in the same cluster, and 0 otherwise. This allows us to rewrite the modularity as:

$$\begin{aligned}\mu(x) =& \frac{1}{2m} \sum_{u \neq v \in V} \big(w_{uv} - k_u k_v/(2m)\big) x_{uv} \\ =& \frac{1}{m} \sum_{u < v \in V} \big(w_{uv} - k_u k_v/(2m)\big) x_{uv}.\end{aligned} \tag{18}$$

Following Aloise et al. (2010), we can reformulate the modularity maximization problem to a clique partitioning problem with the following formulation:

$$\left.\begin{array}{rrl} \max & \mu(x) & \\ \forall 1 \leq i < j < k \leq n & x_{ij} + x_{jk} - x_{ik} & \leq 1 \\ \forall 1 \leq i < j < k \leq n & x_{ij} - x_{jk} + x_{ik} & \leq 1 \\ \forall 1 \leq i < j < k \leq n & -x_{ij} + x_{jk} + x_{ik} & \leq 1 \\ \forall 1 \leq i < j \leq n & x_{ij} & \in \{0, 1\}, \end{array}\right\} \tag{19}$$

which is a BLP formulation. The weighted variant of this problem yields a formulation like Eq. (19) where w are the edge weights and $k_u = \sum_{\{u,v\} \in E} w_{uv}$ for all $v \neq u$ in V. Another variant for graphs including loops and multiple edges is described in Cafieri et al. (2010). We note that, by Eq. (19), maximizing modularity does not require the number of clusters to be known a priori.

There is a large literature about modularity maximization and its solution methods: for a survey, see (Fortunato 2010, §VI). Solution methods based on MP are of particular interest to the topics of this survey. A BLP formulation similar to Eq. (19) was proposed in Brandes et al. (2008). Another BLP formulation with different sets of decision variables (requiring the number of clusters to be known a priori) was proposed in Xu et al. (2007). Some column generation approaches, which scale better in size with respect to previous formulations, were proposed in Aloise et al. (2010). Some MP-based heuristics are discussed in Cafieri et al. (2011), Cafieri et al. (2014), and Aloise et al. (2013).

6 Robust solution methods for the DGP

In this section, we discuss some solution methods for the DGP which can be extended to deal with cases where distances are uncertain, noisy or wrong. Most of the methods which we present are based on MP. We also discuss a different (non-MP based) class of methods in Sect. 6.2, in view of their computational efficiency.

6.1 Mathematical programming-based methods

DGP solution methods based on MP are robust to noisy or wrong data because MP allows for: (a) modification of the objective and constraints; (b) adjoining of side constraints. Moreover, although we do not review these here, there are MP-based methodologies for ensuring robustness of solutions (Ben-Tal et al. 2009), probabilistic constraints (Pfeffer 2016), and scenario-based stochasticity (Birge and Louveaux 2011), which can be applied to the formulations in this section.

6.1.1 Unconstrained quartic formulation

A system of equations such as Eq. (3) is itself an MP formulation with objective function identically equal to zero, and $X = \mathbb{R}^{nK}$. It therefore belongs to the QCP class. In practice, solvers for this class perform rather poorly when given Eq. (3) as input (Lavor et al. 2006). Much better performances can be obtained by solving the following unconstrained formulation:

$$\min \sum_{\{u,v\}\in E} \left(\|x_u - x_v\|_2^2 - d_{uv}^2\right)^2. \tag{20}$$

We note that Eq. (20) consists in the minimization of a polynomial of degree four. It belongs to the class of nonconvex NLP formulations. In general, this is an NP-hard class (Liberti 2019), which is not surprising, as it formulates the DGP which is itself an NP-hard problem. Very good empirical results can be obtained on the DGP by solving Eq. (20) with a local NLP solver such as IPOPT (COIN-OR 2006) or (SNOPT Gill 2006) from a good starting point (Lavor et al. 2006). This is the reason why Eq. (20) is very important: it can be used to "refine" solutions obtained with other methods, as it suffices to let such solutions be starting points given to a local solver acting on Eq. (20).

Even if the distances d_{uv} are noisy or wrong, optimizing Eq. (20) can yield good approximate realizations. If the uncertainty on the distance values is modelled using an interval $[d_{uv}^L, d_{uv}^U]$ for each edge $\{u, v\}$, the following function (Liberti et al. 2010) can be optimized instead of Eq. (20):

$$\min \sum_{\{u,v\}\in E} \left(\max\left(0, (d_{uv}^L)^2 - \|x_u - x_v\|_2^2\right) + \max\left(0, \|x_u - x_v\|_2^2 - (d_{uv}^U)^2\right)\right). \tag{21}$$

The DGP variant where distances are intervals instead of values is known as the INTERVAL DGP (iDGP) (Gonçalves et al. 2017; Lavor et al. 2013). We remark that, with interval distances, the formulations proposed in this section are no longer exact reformulations of Eq. (3).

Note that Eq. (21) involves binary max functions with two arguments. Relatively a few MP user interfaces/solvers would accept this function. To overcome this issue, we linearize (see Sect. 2.4.1) the two max terms by two sets of added decision variables y, z, and obtain:

$$\left.\begin{array}{rrcl} \min & \sum_{\{u,v\}\in E} (y_{uv} + z_{uv}) & & \\ \forall \{u,v\} \in E & \|x_u - x_v\|_2^2 & \geq & (d_{uv}^L)^2 - y_{uv} \\ \forall \{u,v\} \in E & \|x_u - x_v\|_2^2 & \leq & (d_{uv}^U)^2 + z_{uv} \\ & y, z & \geq & 0, \end{array}\right\} \tag{22}$$

which follows from Eq. (21) because of the objective function direction, and because $a \geq \max(b, c)$ is equivalent to $a \geq b \wedge a \geq c$. We note that Eq. (22) is no longer an unconstrained quartic, however, but a QCP. It expresses a minimization of penalty variables to the quadratic inequality system:

$$\forall \{u,v\} \in E \quad (d^L_{uv})^2 \leq \|x_u - x_v\|_2^2 \leq (d^U_{uv})^2. \tag{23}$$

We also note that many local NLP solvers take very arbitrary functions in input (such as functions expressed by computer code), so the reformulation Eq. (22) may be unnecessary when only locally optimal solutions of Eq. (21) are needed.

6.1.2 Constrained quadratic formulations

We propose two formulations in this section. The first is derived directly from Eq. (3):

$$\left.\begin{array}{rl} \min & \sum\limits_{\{u,v\}\in E} s_{uv}^2 \\ \forall \{u,v\} \in E & \|x_u - x_v\|_2^2 = d_{uv}^2 + s_{uv}. \end{array}\right\} \tag{24}$$

We note that Eq. (24) is a QCQP formulation. Similarly to Eq. (22), it uses additional variables to penalize feasibility errors with respect to (3). Differently from Eq. (22), however, it removes the need for two separate variables to model slack and surplus errors. Instead, s_{uv} is unconstrained, and can therefore take any value. The objective, however, minimizes the sum of the squares of the components of s. In practice, Eq. (24) performs much better than Eq. (3); on average, the performance is comparable to that of Eq. (20). We remark that Eq. (24) has a convex objective function but nonconvex constraints.

The second formulation which we propose is an exact reformulation of Eq. (20). First, we replace the minimization of squared errors by absolute values, yielding:

$$\min \sum_{\{u,v\}\in E} \left| \|x_u - x_v\|_2^2 - d_{uv}^2 \right|,$$

which clearly has the same set of global optima as Eq. (20). We then rewrite this similarly to Eq. (22) as follows:

$$\left.\begin{array}{rrl} \min & \sum\limits_{\{u,v\}\in E} (y_{uv} + z_{uv}) & \\ \forall \{u,v\} \in E & \|x_u - x_v\|_2^2 & \geq d_{uv}^2 - y_{uv} \\ \forall \{u,v\} \in E & \|x_u - x_v\|_2^2 & \leq d_{uv}^2 + z_{uv} \\ & y, z & \geq 0, \end{array}\right\}$$

which, again, does not change the global optima. Next, we note that we can fix $z_{uv} = 0$ without changing global optima, since they all have the property that $z_{uv} = 0$. Now, we replace y_{uv} in the objective function by $d_{uv}^2 - \|x_u - x_v\|_2^2$, which we can do without changing the optima since the first set of constraints reads $y_{uv} \geq d_{uv}^2 - \|x_u - x_v\|_2^2$. We can discard the constant d_{uv}^2 from the objective, since adding constants to the objective does not change optima, and change $\min -f$ to $-\max f$, yielding:

$$\left.\begin{array}{rl} \max & \sum\limits_{\{u,v\}\in E} \|x_u - x_v\|_2^2 \\ \forall \{u,v\} \in E & \|x_u - x_v\|_2^2 \ \leq d_{uv}^2, \end{array}\right\} \tag{25}$$

which is a QCQP known as the "push-and-pull" formulation of the DGP, since the constraints ensure that x_u, x_v are pushed closer together, while the objective attempts to pull them apart (Mencarelli et al. 2017, §2.2.1).

Contrariwise to Eq. (24), Eq. (25) has a nonconvex (in fact, concave) objective function and convex constraints. Empirically, this often turns out to be somewhat easier than tackling the reverse situation. The theoretical justification is that finding a feasible solution in a nonconvex set is a hard task in general, whereas finding local optima of a nonconvex function in a convex set is tractable: the same cannot be said for global optima, but in practice one is often satisfied with "good" local optima.

6.1.3 Semidefinite programming

SDP is linear optimization over the cone of psd matrices, which is convex: if A, B are two psd matrices, $C = \alpha A + (1-\alpha)B$ is psd for $\alpha \in [0,1]$. Suppose that there is $x \in \mathbb{R}^n$, such that $x^\top C x < 0$. Then, $\alpha x^\top A x + (1-\alpha) x^\top B x < 0$, so $0 \leq \alpha x^\top A x < -(1-\alpha) x^\top B x \leq 0$, i.e., $0 < 0$, which is a contradiction, and hence C is also psd as claimed. Therefore, SDP is a subclass of cNLP.

The SDP formulation which we propose is a relaxation of Eq. (3). First, we write $\|x_u - x_v\|_2^2 = \langle x_u, x_u\rangle + \langle x_v, x_v\rangle - 2\langle x_u, x_v\rangle$. Then, we linearize all of the scalar products by means of additional variables X_{uv}:

$$\begin{aligned} \forall \{u,v\} \in E \quad X_{uu} + X_{vv} - 2X_{uv} &= d_{uv}^2 \\ X &= xx^\top. \end{aligned}$$

We note that $X = xx^\top$ constitutes the whole set of defining constraints $X_{uv} = \langle x_u, x_v\rangle$ (for each $u, v \leq n$) introduced by the linearization procedure (Sect. 2.4.1).

The relaxation which we envisage does not entirely drop the defining constraints, as in Sect. 2.4.1. Instead, it relaxes them from $X - xx^\top = 0$ to $X - xx^\top \succeq 0$. In other words, instead of requiring that all of the eigenvalues of the matrix $X - xx^\top$ are zero, we simply require that they should be ≥ 0. Moreover, since the original variables x do not appear anywhere else, we can simply require $X \succeq 0$, obtaining:

$$\left.\begin{array}{rl} \forall \{u,v\} \in E \ X_{uu} + X_{vv} - 2X_{uv} & = d_{uv}^2 \\ X & \succeq 0. \end{array}\right\} \tag{26}$$

The SDP relaxation in Eq. (26) has the property that it provides a solution $\bar{X}$, which is an $n \times n$ symmetric matrix. Spectral decomposition of $\bar{X}$ yields $P\Lambda P^\top$, where P is a matrix of eigenvectors and $\Lambda = \mathsf{diag}(\lambda)$, where λ is a vector of eigenvalues of $\bar{X}$. Since $\bar{X}$ is psd, $\lambda \geq 0$, which means that $\sqrt{\Lambda}$ is a real matrix. Therefore, by setting $Y = P\sqrt{\Lambda}$, we have that:

$$YY^\top = (P\sqrt{\Lambda})(P\sqrt{\Lambda})^\top = P\sqrt{\Lambda}\sqrt{\Lambda}P^\top = P\Lambda P^\top = \bar{X},$$

which implies that $\bar{X}$ is the Gram matrix of Y. Thus, we can take Y to be a realization satisfying Eq. (3). The only issue is that Y, as an $n \times n$ matrix, is a realization in n dimensions rather than K. Naturally, $\mathsf{rk}(Y) = \mathsf{rk}(\bar{X})$ need not be equal to n, but could be lower; in fact, to find a realization of the given graph, we would like to find a solution $\bar{X}$ with rank at most K. Imposing this constraint is equivalent to asking that $X = xx^\top$ (which have been relaxed in Eq. (26)).

We note that Eq. (26) is a pure feasibility problem. Every SDP solver, however, also accepts an objective function as input. In absence of a "natural" objective in a pure feasibility problem, we can devise one to heuristically direct the search towards parts of the psd cone which we believe might contain "good" solutions. A popular choice is:

$$\begin{aligned}\min \mathsf{tr}(X) &= \min \mathsf{tr}(P\Lambda P^\top) = \min \mathsf{tr}(PP^\top \Lambda) \\ &= \min \mathsf{tr}(PP^{-1}\Lambda) = \min \lambda_1 + \cdots + \lambda_n,\end{aligned}$$

where tr is the trace, the first equality follows by spectral decomposition (with P a matrix of eigenvectors and Λ a diagonal matrix of eigenvalues of X), the second by commutativity of matrix products under the trace, the third by orthogonality of eigenvectors, and the last by definition of trace. This aims at minimizing the sum of the eigenvalues of X, hoping this will decrease the rank of $\bar{X}$.

For the DGP applied to protein conformation (Sect. 3.3.2), the objective function:

$$\min \sum_{\{u,v\} \in E} (X_{uu} + X_{vv} - 2X_{uv})$$

was empirically found to be a good choice (Dias and Liberti 2016, §2.1). We remark that the equality constraints in Eq. (26) can be used to reformulate the function in Eq. (6.1.3) to the constant $\sum_{\{u,v\} \in E} d_{ij}^2$. The reason why Eq. (6.1.3) did not behave like a constant function in empirical testing must be related to the fact the current iterate is not precisely feasible at every step of the solution algorithm. More (unpublished) experimentation showed that the scalarization of the two objectives:

$$\min \sum_{\{u,v\} \in E} (X_{uu} + X_{vv} - 2X_{uv}) + \gamma \mathsf{tr}(X), \tag{27}$$

with γ in the range $O(10^{-2})$–$O(10^{-3})$, is a good objective function for solving Eq. (26) when it is applied to protein conformation.

In the majority of cases, solving SDP relaxations does not yield solution matrices with rank K, even with objective functions such as Eq. (27). We discuss methods for constructing an approximate rank K realization from $\bar{X}$ in Sect. 7.

SDP is one of those problems which is not known to be in P (nor NP-complete) in the Turing machine model. It is, however, known that SDPs can be solved in polynomial time up to a desired error tolerance $\epsilon > 0$, with the complexity depending on $\frac{1}{\epsilon}$ as well as the instance size. Currently, however, the main issue with SDP is technological: state-of-the art solvers do not scale all that well with size. One of the reasons is that K is usually fixed (and small) with respect to n, so the while the original problem has $O(n)$ variables, the SDP relaxation has $O(n^2)$. Another reason is that

the Interior Point Method (IPM), which often features as a "state of the art" SDP solver, has a relatively high computational complexity (Potra and Wright 2000): a "big oh" notation estimate of $O(\max(m,n)mn^{2.5})$ is given in Bubeck's blog at ORFE, Princeton.[2]

6.1.4 Diagonally dominant programming

To address the size limitations of SDP, we employ some interesting linear approximations of the psd cone proposed in Majumdar et al. (2014) and Ahmadi and Majumdar (2019). An $n \times n$ real symmetric matrix X is diagonally dominant (dd) if:

$$\forall i \leq n \quad \sum_{j \neq i} |X_{ij}| \leq X_{ii}. \tag{28}$$

As remarked in Sect. 5.2.1, it is well known that every dd matrix is also psd, while the converse may not hold. Specifically, the set of dd matrices form a sub-cone of the cone of psd matrices (Barker and Carlson 1975).

The interest of dd matrices is that, by linearization of the absolute value terms, Eq. (28) can be reformulated, so it becomes linear: we introduce an added matrix T of decision variables, then write:

$$\forall i \leq n \quad \sum_{j \neq i} T_{ij} \leq X_{ii} \tag{29}$$

$$-T \leq X \leq T, \tag{30}$$

which are linear constraints equivalent to Eq. (28) (Ahmadi and Majumdar 2019, Thm. 10). One can see this easily whenever $X \geq 0$ or $X \leq 0$. Note that

$$\forall i \leq n \quad X_{ii} \geq \sum_{j \neq i} T_{ij} \geq \sum_{j \neq i} X_{ij}$$
$$\forall i \leq n \quad X_{ii} \geq \sum_{j \neq i} T_{ij} \geq \sum_{j \neq i} -X_{ij}$$

follow directly from Eqs. (29) and (30). Now one of the RHSs is equal to $\sum_{j \neq i} |X_{ij}|$, which implies Eq. (28). For the general case, the argument uses the extreme points of Eqs. (29) and (30) and elimination of T by projection.

We can now approximate Eq. (26) by the pure feasibility LP:

$$\left.\begin{array}{rr} \forall \{u,v\} \in E \; X_{uu} + X_{vv} - 2X_{uv} & = d_{uv}^2 \\ \forall i \leq n \qquad \sum_{j \neq i} T_{ij} & \leq X_{ii} \\ -T \leq X & \leq T, \end{array}\right\} \tag{31}$$

[2] http://blogs.princeton.edu/imabandit/2013/02/19/orf523-ipms-for-lps-and-sdps/.

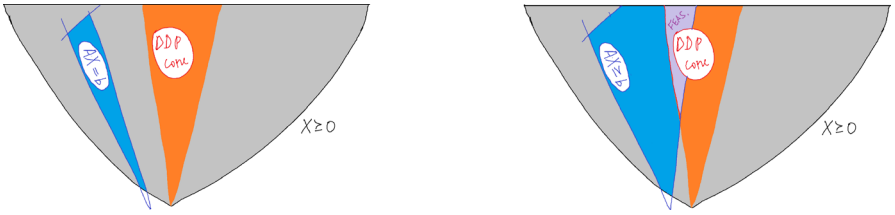

Fig. 5 On the left, the DDP is infeasible even if the SDP is not; on the right, a relaxed set of constraints makes the DDP feasible

which we call a diagonally dominant program (DDP). As in Eq. (26), we do not explicitly give an objective function, since it depends on the application. Since the DDP in Eq. (31) is an inner approximation of the corresponding SDP in Eq. (26), the DDP feasible set is a subset of that of the SDP. This situation yields both an advantage and a disadvantage: any solution $\tilde{X}$ of the DDP is psd, and can be obtained at a smaller computational cost; however, the DDP might be infeasible even if the corresponding SDP is feasible (see Fig. 5, left). To decrease the risk of infeasibility of Eq. (31), we relax the equation constraints to inequality, and impose an objective as in the push-and-pull formulation Eq. (25):

$$\left.\begin{array}{rrl} \max & \sum\limits_{\{u,v\}\in E} (X_{uu} + X_{vv} - 2X_{uv}) & \\ \forall \{u,v\} \in E & X_{uu} + X_{vv} - 2X_{uv} & \leq d_{uv}^2 \\ \forall i \leq n & \sum\limits_{j\neq i} T_{ij} & \leq X_{ii} \\ & -T \leq X & \leq T. \end{array}\right\} \tag{32}$$

This makes the DDP feasible set larger, which means that it is more likely to be feasible (see Fig. 5, right). Equation (32) was successfully tested on protein graphs in Dias and Liberti (2016).

If C is any cone in $\mathbb{R}^n$, the *dual cone* C^* is defined as:

$$C^* = \{y \in \mathbb{R}^n \mid \forall x \in C \ \langle x, y\rangle \geq 0\}.$$

Note that the dual cone contains the set of vectors making a non-obtuse angle with all of the vectors in the original (primal) cone. We can exploit the dual dd cone to provide another DDP formulation for the DGP which turns out to be an outer approximation. Outer approximations have symmetric advantages and disadvantages with respect to the inner ones: if the original SDP is feasible, then the outer DDP approximation is also feasible; however, the solution $\tilde{X}$ which we obtain from the outer DDP need not be a psd matrix. Some computational experience related to Salgado et al. (2018) showed that it often happens that more or less half of the eigenvalues of $\tilde{X}$ are negative.

We now turn to the actual DDP formulation related to the dual dd cone. A cone C of $n \times n$ real symmetric matrices is *finitely generated* by a set $\mathcal{X}$ of matrices if:

$$\forall X \in C \; \exists \delta \in \mathbb{R}_+^{|\mathcal{X}|} \quad X = \sum_{x \in \mathcal{X}} \delta_x x x^\top.$$

It turns out (Barker and Carlson 1975) that the dd cone is finitely generated by:

$$\mathcal{X}_{\mathsf{dd}} = \{e_i \mid i \leq n\} \cup \{e_i \pm e_j \mid i < j \leq n\},$$

where $e_1, \ldots, e_n$ is the standard orthogonal basis of $\mathbb{R}^n$. This is proved in Barker and Carlson (1975) by showing that the following rank-one matrices are extreme rays of the dd cone:

- $E_{ii} = \mathsf{diag}(e_i)$, where $e_i = (0, \ldots, 0, 1_i, 0, \ldots, 0)^\top$;
- E_{ij}^+ has a minor $\begin{pmatrix} 1_{ii} & 1_{ij} \\ 1_{ji} & 1_{jj} \end{pmatrix}$ and is zero elsewhere;
- E_{ij}^- has a minor $\begin{pmatrix} 1_{ii} & -1_{ij} \\ -1_{ji} & 1_{jj} \end{pmatrix}$ and is zero elsewhere,

and, moreover, that the extreme rays are generated by the standard basis vectors as follows:

$$\begin{aligned} \forall i \leq n \quad E_{ii} &= e_i e_i^\top \\ \forall i < j \leq n \quad E_{ij}^+ &= (e_i + e_j)(e_i + e_j)^\top \\ \forall i < j \leq n \quad E_{ij}^- &= (e_i - e_j)(e_i - e_j)^\top. \end{aligned}$$

This observation allowed Ahmadi and his co-authors to write the DDP formulation equation [Eq. (32)] in terms of the extreme rays $E_{ii}, E_{ij}^\pm$ (Ahmadi and Majumdar 2019), and also to define a column generation algorithms over them (Ahmadi et al. 2020).

If a matrix cone is finitely generated, the dual cone has the same property. Let $\mathbb{S}_n$ be the set of real symmetric $n \times n$ matrices; for $A, B \in \mathbb{S}_n$ we define an inner product $\langle A, B \rangle = A \bullet B \triangleq \mathsf{tr}(AB^\top)$.

Theorem 3 *Assume C is finitely generated by $\mathcal{X}$. Then C^* is also finitely generated. Specifically, $C^* = \{Y \in \mathbb{S}_n \mid \forall x \in \mathcal{X} \, (Y \bullet x x^\top \geq 0)\}$.*

Proof By assumption, $C = \{X \in \mathbb{S}_n \mid \exists \delta \in \mathbb{R}_+^{|\mathcal{X}|} \; X = \sum_{x \in \mathcal{X}} \delta_x x x^\top\}$.

($\Rightarrow$) Let $Y \in \mathbb{S}_n$ be such that, for each $x \in \mathcal{X}$, we have $Y \bullet x x^\top \geq 0$. We are going to show that $Y \in C^*$, which, by definition, consists of all matrices Y, such that for all $X \in C$, $Y \bullet X \geq 0$. Note that, for all $X \in C$, we have $X = \sum_{x \in \mathcal{X}} \delta_x x x^\top$ (by finite generation). Hence, $Y \bullet X = \sum_x \delta_x Y \bullet x x^\top \geq 0$ (by definition of Y), whence $Y \in C^*$.

($\Leftarrow$) Suppose $Z \in C^* \setminus \{Y \mid \forall x \in \mathcal{X} \, (Y \bullet x x^\top \geq 0)\}$. Then, there is $\mathcal{X}' \subset \mathcal{X}$, such that for any $x \in \mathcal{X}'$, we have $Z \bullet x x^\top < 0$. Consider any $Y = \sum_{x \in \mathcal{X}'} \delta_x x x^\top \in C$ with $\delta \geq 0$. Then, $Z \bullet Y = \sum_{x \in \mathcal{X}'} \delta_x Z \bullet x x^\top < 0$, so $Z \notin C^*$, which is a contradiction. Therefore, $C^* = \{Y \mid \forall x \in \mathcal{X} \, (Y \bullet x x^\top \geq 0)\}$ as claimed. □

We are going to exploit Theorem 3 to derive an explicit formulation of the following DDP formulation based on the dual cone C_{dd}^* of the dd cone C_{dd} finitely generated by $\mathcal{X}_{\mathsf{dd}}$:

$$\left.\begin{array}{rl} \forall \{u,v\} \in E \; X_{uu} + X_{vv} - 2X_{uv} & = d_{uv}^2 \\ X & \in C_{\mathsf{dd}}^*. \end{array}\right\}$$

We remark that $X \bullet vv^\top = v^\top X v$ for each $v \in \mathbb{R}^n$. By Theorem 3, $X \in C_{\mathsf{dd}}^*$ can be restated as $\forall v \in \mathcal{X}_{\mathsf{dd}} \; v^\top X v \geq 0$. We obtain the following LP formulation:

$$\left.\begin{array}{rll} \max & \sum\limits_{\{u,v\} \in E} \left(X_{uu} + X_{vv} - 2X_{uv}\right) & \\ \forall \{u,v\} \in E & X_{uu} + X_{vv} - 2X_{uv} & = d_{uv}^2 \\ \forall v \in \mathcal{X}_{\mathsf{dd}} & v^\top X v & \geq 0. \end{array}\right\} \tag{33}$$

With respect to the primal DDP, the dual DDP formulation in Eq. (33) provides a very tight bound to the objective function value of the push-and-pull SDP formulation Eq. (25). On the other hand, the solution $\bar{X}$ is usually far from being a psd matrix.

6.2 Fast high-dimensional methods

In Sect. 6.1, we surveyed methods based on MP, which are very flexible, insofar as they can accommodate side constraints and noisy data, but computationally demanding. In this section we discuss two very fast, yet robust, methods for embeddings graphs in Euclidean spaces.

6.2.1 Incidence vectors

The simplest, and most naive methods for mapping graphs into vectors are given by exploiting various incidence information in the graph structure. By contrast, the resulting embeddings are unrelated to Eq. (3).

Given a simple graph $G = (V, E)$ with $|V| = n$, $|E| = m$ and edge weight function $w : E \to \mathbb{R}_+$, we present two approaches: one which outputs an $n \times n$ matrix, and one which outputs a single vector in $\mathbb{R}^K$ with $K = \frac{1}{2}n(n-1)$.

1. For each $u \in V$, let $x_u = (x_{uv} \mid v \in V) \in \mathbb{R}^n$ be the incidence vector of $N(u)$ on V, that is:

$$\forall u \in V \quad x_{uv} = \begin{cases} w_{uv} & \text{if } \{u,v\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

2. Let $K = \frac{1}{2}n(n-1)$, and $x^E = (x_e \mid e \in E) \in \mathbb{R}^K$ be the incidence vector of the edge set E into the set $\{\{i,j\} \mid i < j \leq n\}$, that is:

$$x_e = \begin{cases} w_e & \text{if } e \in E \\ 0 & \text{otherwise.} \end{cases}$$

Both embeddings can be obtained in $O(n^2)$ time. Both embeddings are very high dimensional. For practical usefulness, it is necessary to post-process them using dimensional reduction techniques (see Sect. 7).

6.2.2 The universal isometric embedding

This method, also called Fréchet embedding, is remarkable in that it maps any finite metric space congruently into a set of vectors in the ℓ_∞ norm (Kuratowski 1935, §6). No other norm allows exact congruent embeddings in vector spaces (Matoušek 2013). The Fréchet embedding provided the foundational idea for several other probabilistic approximate embeddings in various other norms and dimensions (Bourgain 1985; Linial et al. 1995).

Theorem 4 *Given any finite metric space (X, d), where $|X| = n$ and d is a distance function defined on X, there exists an embedding $\rho : X \to \mathbb{R}^n$ such that $(\rho(X), \ell_\infty)$ is congruent to (X, d).*

This theorem is surprising because of its generality in conjunction with the exactness of the result: it works on any (finite) metric space. The "magic hat" out of which we shall pull the vectors in $\rho(X)$ is simply the only piece of data which we are given, namely the distance matrix of X. More precisely, the ith element of X is mapped to the vector corresponding to the ith column of the distance matrix.

Proof Let $\mathsf{D}(X)$ be the distance matrix of (X, d), namely $\mathsf{D}_{ij}(X) = (d(x_i, x_j))$, where $X = \{x_1, \ldots, x_n\}$. We denote $d(x_i, x_j) = d_{ij}$ for brevity. For any $j \leq n$, we let $\rho(x_j) = \delta_j$, where δ_j is the jth column of $\mathsf{D}(X)$. We have to show that $\|\rho(x_i) - \rho(x_j)\|_\infty = d_{ij}$ for each $i < j \leq n$. By definition of the ℓ_∞ norm, for each $i < j \leq n$, we have:

$$\|\rho(x_i) - \rho(x_j)\|_\infty = \|\delta_i - \delta_j\|_\infty = \max_{k \leq n} |\delta_{ik} - \delta_{jk}| = \max_{k \leq n} |d_{ik} - d_{jk}|. \quad (*)$$

By the triangular inequality on (X, d), for $i < j \leq n$ and $k \leq n$, we have:

$$\begin{aligned} d_{ik} \leq & d_{ij} + d_{jk} \quad \wedge \quad d_{jk} \leq d_{ij} + d_{ik} \\ \Rightarrow & d_{ik} - d_{jk} \leq d_{ij} \quad \wedge \quad d_{jk} - d_{ik} \leq d_{ij} \\ \Rightarrow & |d_{ik} - d_{jk}| \leq d_{ij}; \end{aligned}$$

since these inequalities are valid for each k, by $(*)$, we have:

$$\|\rho(x_i) - \rho(x_j)\|_\infty \leq \max_k d_{ij} = d_{ij}, \quad (\dagger),$$

where the last equality follows because d_{ij} does not depend on k. Now, we note that the maximum of $|d_{ik} - d_{jk}|$ over k must exceed the value of the same expression

when either of the terms d_{ik} or d_{jk} is zero, i.e. when $k \in \{i,j\}$ since, when $k = i$, then $|d_{ik} - d_{jk}| = |d_{ii} - d_{ji}| = d_{ij}$, and the same holds when $k = j$. Hence:

$$\max_{k \leq n} |d_{ik} - d_{jk}| \geq d_{ij}. \quad (\ddagger).$$

By $(*)$, $(\dagger)$ and $(\ddagger)$, we finally have:

$$\forall i < j \leq n \quad \|\rho(x_i) - \rho(x_j)\|_\infty = d_{ij}$$

as claimed. □

We remark that Theorem 4 is only applicable when $\mathsf{D}(X)$ is a distance matrix, which corresponds to the case of a graph G edge-weighed by d being a complete graph. We address the more general case of any (connected) simple graph $G = (V, E)$, corresponding to a partially defined distance matrix, by completing the matrix using the shortest path metric (this distance matrix completion method was used for the isomap heuristic, see Tenenbaum et al. 2000; Liberti and D'Ambrosio 2017 and Sect. 7.1.1):

$$\forall \{i,j\} \notin E \quad d_{ij} = \mathsf{shortest\_path\_length}_G(i,j). \quad (34)$$

In practice, we can compute the lengths of all shortest paths in G using the Floyd–Warshall algorithm, which runs in $O(n^3)$ time (but there exist reasonably fast implementations).

This method yields a realization of G in ℓ_∞^n, which is a high-dimensional embedding. It is necessary to post-process it using dimensional reduction techniques (see Sect. 7).

6.2.3 Multidimensional scaling

The literature on multidimensional scaling (MDS) is extensive (Cox and Cox 2001; Borg and Groenen 2010), and many variants exist. The basic version, called classic MDS, aims at finding an approximate realization of a partial distance matrix. In other words, it is a heuristic solution method for the

> Euclidean distance matrix completion problem (EDMCP). Given a simple undirected graph $G = (V, E)$ with an edge weight function $w : E \to \mathbb{R}_+$, determine whether there exists an integer $K > 0$ and a realization $x : V \to \mathbb{R}^K$, such that Eq. (3) holds.

The difference between EDMCP and DGP may appear diminutive, but it is in fact very important. In the DGP, the integer K is part of the input, whereas in the EDMCP it is part of the output. This has a large effect on worst-case complexity: while the DGP is NP-hard even when only an ε-approximate realization is sought (Saxe 1979, §5), ε-approximate realizations of EDMCPs can be found in polynomial time by solving an SDP (Alfakih et al. 1999). See Liberti and Lavor (2013) and Sánchez and Lavor (2020) for more information about the relationship between EDMCP and DGP.

Consider the following matrix:

$$\Delta(E,d) = \begin{cases} w_{ij}^2 & \text{if } \{i,j\} \in E \\ d_{ij} & \text{otherwise,} \end{cases}$$

where $d = (d_{ij} \mid \{i,j\} \notin E)$ is a vector of decision variables, and $J = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$. Then the following formulation is valid for the EDMCP:

$$\left.\begin{array}{rll} \min\limits_{d,T,X} & \mathbf{1}\mathbf{1}^\top \bullet T & \\ -T & \leq X + \frac{1}{2} J\,\Delta(E,d)\,J \leq T \\ X & \succeq 0. & \end{array}\right\} \tag{35}$$

Theorem 5 *The SDP in Eq. (35) correctly models the EDMCP.*

By "correctly models" we mean that the solution of the EDMCP can be obtained in polynomial time from the solution of the SDP in Eq. (35).

Proof First, we remark that, given a realization $x : V \to \mathbb{R}^n$, its Gram matrix is $X = xx^\top$, and its squared Euclidean distance matrix (EDM) is:

$$D^2 = (\|x_u - x_v\|_2^2 \mid u \leq n \wedge v \leq n) \in \mathbb{R}^{n \times n}.$$

Next, we recall that:

$$X = -\frac{1}{2} J D^2 J \tag{36}$$

by Dattorro (2015) after Schoenberg 1935—see (Liberti and Lavor 2016, §7) for a direct proof[3]. Now, we note that minimizing $\mathbf{1}\mathbf{1}^\top \bullet T$ subject to $-T \leq X + \frac{1}{2} J\Delta(E,d)J \leq T$ is an exact reformulation of:

$$\min_{G,d} \|X - (-1/2)J\Delta(E,d)J)\|_1, \quad (*),$$

since $\mathbf{1}\mathbf{1}^\top \bullet T = \sum_{i,j} T_{ij}$, and T is used to "sandwich" the argument of the ℓ_1 norm in $(*)$. This implies that $X = -\frac{1}{2} J\Delta(E,d)J$ iff $T = 0$ iff $\mathbf{1}\mathbf{1}^\top \bullet T = 0$. Consequently, if the optimal objective function value of Eq. (35) is zero with corresponding solution d^*, T^*, X^*, then $\mathsf{tr}(\mathbf{1}\mathbf{1}^\top \bullet T^*) = 0 \Rightarrow T^* = 0 \Rightarrow (*) = 0$. We also recall another basic fact of linear algebra: a matrix is Gram if and only if it is psd: hence, requiring $X \succeq 0$ forces X to be a Gram matrix. Therefore, X^* is a Gram matrix and $\Delta(E,d^*) = D^2$ is its corresponding EDM by Eq. (36). Finally, the realization x^* corresponding to the Gram matrix X^* can be obtained by spectral decomposition of $X^* = P\Lambda P^\top$, which yields $x^* = P\sqrt{\Lambda}$: this implies that the EDMCP instance is YES. Otherwise, if the optimal objective function value of Eq. (35) is nonzero, then $T^* \neq 0$, which means

[3] Also see http://math.stackexchange.com/questions/1882130/ for a compact derivation.

that the EDMCP instance is NO (assuming it was YES would contradict optimality). □

The practically useful corollary to Thm. (5) is that solving Eq. (35) provides an approximate solution x^* even if $\Delta(E, d)$ cannot be completed to an EDM.

Classic MDS is an efficient heuristic method for finding an approximate realization of a partial distance matrix $\Delta(E, d)$. It works as follows:

1. complete $\Delta(E, d)$ to an approximate EDM $\tilde{D}^2$ using the shortest-path metric (Eq. (34));
2. let $\tilde{X} = -\frac{1}{n} J \tilde{D}^2 J$;
3. let $P \tilde{\Lambda} P^\top$ be the spectral decomposition of $\tilde{X}$;
4. if $\tilde{\Lambda} \geq 0$ then, by Eq. (36), $\tilde{D}^2$ is a EDM, with corresponding (exact) realization $\tilde{x} = P\sqrt{\Lambda}$;
5. otherwise, let $\Lambda^+ = \mathsf{diag}((\max(\lambda, 0) \mid \lambda \in \Lambda))$: then $\tilde{x} = P\sqrt{\Lambda^+}$ is an approximate realization of $\tilde{D}^2$.

Note that both Eq.(35) and classic MDS determine K as part of the output, i.e. K is the rank of the realizations x^* and $\tilde{x}$.

7 Dimensional reduction techniques

Dimensional reduction techniques reduce the dimensionality of a set of vectors according to different criteria, which may be heuristic, or give some (possibly probabilistic) guarantee of keeping some quantity approximately invariant. They are necessary to make many of the methods in Sect. 6 useful in practice.

7.1 Principal component analysis

Principal component analysis (PCA) is one of the foremost dimensional reduction techniques. It is ascribed to Harold Hotelling[4] (Hotelling 1933).

Consider an $n \times m$ matrix X consisting of n data row vectors in $\mathbb{R}^m$, and let $K < m$ be a given integer. We want to find a change of coordinates for X, such that the first component has largest variance over the transformed vectors, the second component has second-largest variance, and so on, until the Kth component. The other components can be neglected, as the variance of the data in those directions is low.

The usual geometric interpretation of PCA is to take the smallest enclosing ellipsoid $\mathcal{E}$ for X: then the required coordinate change maps component 1 to the line parallel to the largest radius of $\mathcal{E}$, component 2 to the line parallel to the second-largest

[4] A young and unknown George Dantzig had just finished his presentation of LP to an audience of "big shots", including Koopmans and Von Neumann. Harold Hotelling raised his hand, and stated: "but we all know that the world is nonlinear!", thereby obliterating the simplex method as a mathematical curiosity. Luckily, Von Neumann answered on Dantzig's behalf and in his defence (Dantzig 1983).

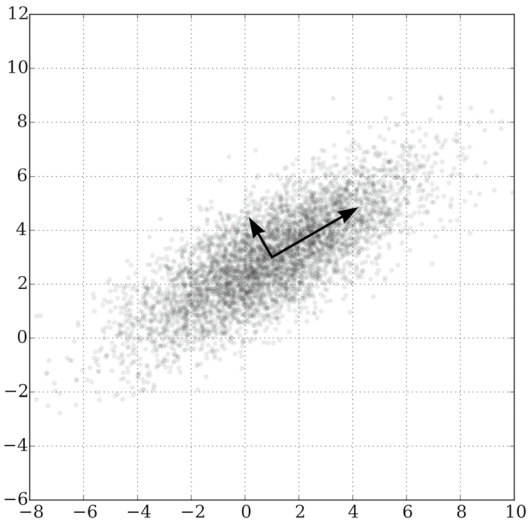

Fig. 6 Geometric interpretation of PCA (image from Wikipedia: Principal component analysis (2019))

radius of $\mathcal{E}$, and so on until component K (see Fig. 6). The statistical interpretation of PCA looks for the change of coordinates which makes the data vectors be uncorrelated in their components. Figure 6 should give an intuitive idea about why this interpretation corresponds with the ellipsoid of the geometric interpretation. The cartesian coordinates in Fig. 6 are certainly correlated, while the rotated coordinates look far less (linearly) correlated. The zero correlation situation corresponds to a perfect ellipsoid. An ellipsoid is described by the equation $\sum_{j\leq n}\left(\frac{x_j}{r_j}\right)^2 = 1$, which has no mixed terms $x_i x_j$ contributing to correlation. Both interpretations are well (and formally) argued in Vidal et al. (2016, §2.1).

The interpretation given here is motivated by DG, and related to MDS (Sect. 6.2.3). PCA can be seen as a modification of MDS which only takes into account the K (nonnegative) principal components. Instead of Λ^+ (step 5 of the MDS algorithm), PCA uses a different diagonal matrix Λ^{pca}: the ith diagonal component is:

$$\Lambda^{\mathsf{pca}}_{ii} = \begin{cases} \max(\Lambda_{ii}, 0) & \text{if } i \leq K \\ 0 & \text{otherwise,} \end{cases} \tag{37}$$

where $P\Lambda P^\top$ is the spectral decomposition of $\tilde{G}$. In this interpretation, when given a partial distance matrix and the integer K as input, PCA can be used as an approximate solution method for the DGP.

On the other hand, the PCA algorithm is most usually considered as a method for dimensionality reduction, so it has a data matrix X and an integer K as input. It is as follows:

1. let $\tilde{G} = XX^\top$ be the $n \times n$ Gram matrix of the data matrix X;
2. let $P\tilde{\Lambda}P^\top$ be the spectral decomposition of $\tilde{G}$;
3. return $\tilde{x} = P\sqrt{\Lambda^{\mathsf{pca}}}$.

Then $\tilde{x}$ is an $n \times K$ matrix, where $K < n$. The ith row vector in $\tilde{x}$ is a dimensionally reduced representation of the ith row vector in X.

There is an extensive literature on PCA, ranging over many research papers, dedicated monographs, and textbooks (Wikipedia: Principal component analysis 2019; Jolliffe 2010; Vidal et al. 2016). Among the variants and extensions, see (Demartines and Hérault 1997; Saerens et al. 2004; D'Aspremont et al. 2014; Allen 2012; Dey et al. 2017).

7.1.1 Isomap

One of the most interesting applications of PCA is possibly the Isomap algorithm (Tenenbaum et al. 2000), already mentioned above in Sect. 6.2.2, which is able to use PCA to perform a nonlinear dimensional reduction from the original dimension m to a given target dimension K, as follows.

1. Form a connected graph $H = (V, E)$ with the column indices $1, \dots, n$ of X as vertex set V: determine a threshold value τ, such that, for each column vector x_i in X (for $i \leq n$), and for each x_j in X, such that $\|x_i - x_j\|_2 \leq \tau$, the edge $\{i, j\}$ is in the edge set E; the graph H should be as sparse as possible but also connected.
2. Complete H using the shortest path metric (Eq. (34)).
3. Use PCA in the MDS interpretation mentioned above: interpret the completion of (V, E) as a metric space, construct its (approximate) EDM $\tilde{D}$, compute the corresponding (approximate) Gram matrix $\tilde{G}$, compute the spectral decomposition of $\tilde{G}$, replace its diagonal eigenvalue matrix Λ as in Eq. (37), and return the corresponding K-dimensional vectors.

Intuitively, Isomap works well because in many practical situations where a set X of points in $\mathbb{R}^m$ are close to a (lower) K-dimensional manifold, the shortest-path metric is likely to be a better estimation of the Euclidean distance in $\mathbb{R}^K$ than the Euclidean distance in $\mathbb{R}^m$, see (Tenenbaum et al. 2000, Fig. 3).

7.2 Barvinok's naive algorithm

By Eq. (26), we can solve an SDP relaxation of the DGP and obtain an $n \times n$ psd matrix solution $\bar{X}$ which, in general, will not have rank K (i.e., it will not yield an $n \times K$ realization matrix, but rather an $n \times n$ one). In this section, we shall derive a dimensionality reduction algorithm to obtain an approximation of $\bar{X}$ which has the correct rank K.

7.2.1 Quadratic programming feasibility

Barvinok's naive algorithm (Barvinok 1997, §5.3) is a probabilistic algorithm which finds an approximate vector solution $x' \in \mathbb{R}^n$ to a system of quadratic equations:

$$\forall i \leq m \quad x^\top Q^i x = a_i, \tag{38}$$

where the Q^i are $n \times n$ symmetric matrices, $a \in \mathbb{R}^m$, $x \in \mathbb{R}^n$, and m is polynomial in n. The analysis of this algorithm provides a probabilistic bound on the maximum distance that x' can have from the set of solutions of Eq. (38). Thereafter, one can run a local NLP solver with x' as a starting point, and obtain a hopefully good (approximate) solution to Eq. (38). We note that this algorithm is still not immediately applicable to our setting where K might be different from 1: we shall address this issue in Sect. 7.2.4.

Barvinok's naive algorithm solves an SDP relaxation of Eq. (38), and then retrieves a certain randomized vector from the solution:

1. form the SDP relaxation:
$$\forall i \leq m \; (Q^i \bullet X = a_i) \wedge X \succeq 0 \tag{39}$$
of Eq. (38) and solve it to obtain $\bar{X} \in \mathbb{R}^{n \times n}$;
2. let $T = \sqrt{\bar{X}}$, which is a real matrix, since $\bar{X} \succeq 0$ (T can be obtained by spectral decomposition, i.e., $\bar{X} = P \Lambda P^\top$ and $T = P\sqrt{\Lambda}$);
3. let y be a vector sampled from the multivariate normal distribution $\mathsf{N}^n(0, 1)$;
4. compute and return $x' = Ty$.

The analysis provided in Barvinok (1997) shows that $\exists c > 0$ and an integer $n_0 \in \mathbb{N}$, such that $\forall n \geq n_0$:

$$\mathsf{P}\left(\forall i \leq m \quad \mathsf{dist}(x', \mathcal{X}_i) \leq c \sqrt{\|\bar{X}\|_2 \ln n}\right) \geq 0.9. \tag{40}$$

In Eq. (40), $\mathsf{dist}(b, B) = \inf_{\beta \in B} \|b - \beta\|_2$ is the Euclidean distance between the point b and the set B, and c is a constant that only depends on $\log_n m$. We recall that $\mathsf{P}(\cdot)$ denotes the probability of an event. We note that the term $\sqrt{\|\bar{X}\|_2}$ in Eq. (40) arises from T being a factor of $\bar{X}$. We note also that 0.9 follows from assigning some arbitrary value to some parameter—i.e., the constant 0.9 can be increased as long as the problem size is large enough.

For cases of Eq. (38) where one of the quadratic equations is $\|x\|_2^2 = 1$ (namely, the solutions of Eq. (38) must belong to the unit sphere), it is noted in Barvinok (1997, Eg. 5.5) that, if $\bar{X}$ is "sufficiently generic", then $\|\bar{X}\|_2 = O(1/n)$, which implies that the bounding function $c\sqrt{\bar{X}_2 \ln n} \to 0$ as $n \to \infty$. This, in turn, means that x' converges towards a feasible solution of the original problem in the limit.

7.2.2 Concentration of measure

The term $\ln n$ in Eq. (40) arises from a phenomenon of high-dimensional geometry called "concentration of measure".

We first give an example of concentration of measure around the median value of a Lipschitz function. We recall that a function $f : \mathcal{X} \to \mathbb{R}$ is Lipschitz if there is a constant $M > 0$, s.t. for any $x, y \in \mathcal{X}$, we have $|f(x) - f(y)| < M\|x - y\|_2$. A measure

space $(\mathcal{X}, \mu)$ has the concentration of measure property if, for any Lipschitz function f, there are constants $C, c > 0$, such that:

$$\forall \varepsilon > 0 \quad \mathsf{P}(|f(x) - \mathsf{M}_{\mu}(f)| > \varepsilon \mid x \in \mathcal{X}) \leq C\, e^{-c\varepsilon^2}, \tag{41}$$

where $\mathsf{M}_{\mu}(f)$ is the median value of f with respect to μ. In other words, $\mathcal{X}$ has measure concentration if, for any Lipschitz function f, its discrepancy from its median value is small with arbitrarily high probability. It turns out that the Euclidean space $\mathbb{R}^n$ with the Gaussian density measure $\phi(x) = (2\pi)^{n/2} e^{-\|x\|_2^2/2}$ has measure concentration around the mean (Barvinok 2002, §5.3).

Measure concentration is interesting in view of applications, since, given any large enough closed subset A of $\mathcal{X}$, its ε-neighbourhood:

$$A(\varepsilon) = \{x \in \mathcal{X} \mid \mathsf{dist}(x, A) \leq \varepsilon\} \tag{42}$$

contains almost the whole measure of $\mathcal{X}$. More precisely, if $(\mathcal{X}, \mu)$ has measure concentration and $A \subset \mathcal{X}$ is closed, then for any $p \in (0, 1)$ there is a $\varepsilon_0(p) > 0$ such that (Liberti and Vu 2018, Prop. 2):

$$\forall \varepsilon \geq \varepsilon_0(p) \quad \mu(A(\varepsilon)) > 1 - p. \tag{43}$$

Equation (43) is useful for applications, because it defines a way to analyse probabilistic algorithms. For a random point sampled in $(\mathcal{X}, \mu)$ that happens to be in A on average, Eq. (43) ensures that it is unlikely that it should be far from A. This can be used to bound errors, as Barvinok did with his naive algorithm. Concentration of measure is fundamental in data science, insofar as it may provide algorithmic analyses to the effect that some approximation errors decrease in function of the increasing instance size.

7.2.3 Analysis of Barvinok's algorithm

We sketch the main lines of the analysis of Barvinok's algorithm—see (Barvinok 1995, Thm. 5.4) or (Liberti and Vu 2018, §3.2) for a more detailed proof. We let $\mathcal{X} = \mathbb{R}^n$ and $\mu(x) = \phi(x)$ be the Gaussian density measure. It is easy to show that:

$$\mathsf{E}_{\mu}(x^\top Q^i x \mid x \in \mathbb{R}^n) = \mathsf{tr}(Q^i)$$

for each $i \leq m$. From this fact and the factorization $\bar{X} = TT^\top$, one obtains:

$$\mathsf{E}_{\mu}\big(x^\top T^\top Q^i\, Tx \mid x \in \mathcal{X}\big) = \mathsf{tr}(T^\top Q^i\, T) = \mathsf{tr}(Q^i \bar{X}) = Q^i \bullet \bar{X} = a_i.$$

This shows that, for any $y \sim \mathsf{N}^n(0, 1)$, the average of $y^\top T^\top Q^i\, Ty$ is a_i.

The analysis then goes on to show that, for some $y \sim \mathsf{N}^n(0, 1)$, it is unlikely that $y^\top T^\top Q^i\, Ty$ should be far from a_i. It achieves this result by defining the sets $A_i^+ = \{x \in \mathbb{R}^n \mid x^\top Q^i x \geq a_i\}$, $A_i^- = \{x \in \mathbb{R}^n \mid x^\top Q^i x \leq a_i\}$, and their respective neighbourhoods $A_i^+(\varepsilon)$, $A_i^-(\varepsilon)$. Using a technical lemma (Liberti and Vu 2018, Lemma 4), it is possible to apply Eq. (43) to $A_i^+(\varepsilon)$ and $A_i^-(\varepsilon)$ to argue for concentration of measure. Applying the union bound, it can be shown that their

intersection $A_i(\varepsilon)$ is the neighbourhood of $A_i = \{x \in \mathbb{R}^n \mid x^\top Q^i x = a_i\}$. Another application of the union bound to all the sets $A_i(\varepsilon)$ yields the result (Liberti and Vu 2018, Thm. 5).

We note that concentration of measure proofs often have this structure: (a) prove that a certain event holds on average; (b) prove that the discrepancy from average gets smaller and/or more unlikely with increasing size. Usually proving (a) is easier than proving (b).

7.2.4 Applicability to the DGP

The issue with trying to apply Barvinok's naive algorithm to the DGP is that we should always assume $K = 1$ by Eq. (38). To circumvent this issue, we might represent an $n \times K$ realization matrix as a vector in $\mathbb{R}^{nK}$ by stacking its columns (or concatenating its rows). This, on the other hand, would require solving SDPs with $nK \times nK$ matrices, which is prohibitive because of size.

Luckily, Barvinok's naive algorithm can be very easily extended to arbitrary values of K. We replace Step 3 by:

3*b*. let y be an $n \times K$ matrix sampled from $\mathsf{N}^{n\times K}(0, 1)$.

The corresponding analysis needs some technical changes (Liberti and Vu 2018), but the overall structure is the same as the case $K = 1$. The obtained bound replaces $\sqrt{\ln n}$ in Eq. (40) with $\sqrt{\ln nK}$.

In the DGP case, the special structure of the matrices Q^i (for i ranging over the edge set E) makes it possible to remove the factor K, so we retrieve the exact bound of Eq. (40). As noted in Sect. 7.2.1, if the DGP instance is on a sphere (Liberti et al. 2016), this means that $x' = Ty$ converges to an exact realization with probability 1 in the limit of $n \to \infty$. Similar bounds to Eq. (40) were also derived for the iDGP case (Liberti and Vu 2018).

Barvinok also described concentration of measure-based techniques for finding low-ranking solutions of the SDP in Eq. (39) (see Barvinok 1995 and Barvinok 2002, §6.2), but these do not allow the user to specify an arbitrary rank K, so they only apply to the EDMCP.

7.3 Random projections

Random projections (RPs) are another dimensionality reduction technique exploiting high-dimensional geometry properties and, in particular, the concentration of measure phenomenon (Sect. 7.2.2). They are more general than Barvinok's naive algorithm (Sect. 7.2) in that they apply to sets of vectors in some high-dimensional Euclidean space $\mathbb{R}^n$ (with $n \gg 1$). These sets are usually finite and growing polynomially with instance sizes (Vempala 2004), but they may also be infinite (Woodruff 2014), in which case the technical name used is subspace embeddings.

7.3.1 The Johnson–Lindenstrauss lemma

The foremost result in RPs is the celebrated Johnson–Lindenstrauss lemma (JLL) (Johnson and Lindenstrauss 1984). For a set of vectors $\mathcal{X} \subset \mathbb{R}^n$ with $|\mathcal{X}| = \ell$, and an $\varepsilon \in (0, 1)$, there is a $k = O(\frac{1}{\varepsilon^2} \ln \ell)$ and a mapping $f : \mathcal{X} \to \mathbb{R}^k$, such that:

$$\forall x, y \in \mathcal{X} \quad (1 - \varepsilon)\|x - y\|_2 \leq \|f(x) - f(y)\|_2 \leq (1 + \varepsilon)\|x - y\|_2. \tag{44}$$

The proof of this result (Johnson and Lindenstrauss 1984, Lemma 1) is probabilistic: it shows that an f satisfying Eq. (44) exists with some nonzero probability.

Later and more modern proofs (e.g., Dasgupta and Gupta 2002) clearly point out that f can be a linear operator represented by a $k \times n$ matrix T, each component of which can be sampled from a subgaussian distribution. This term refers to a random variable $\mathfrak{B}$ for which there are constants C, c, s.t. for each $t > 0$, we have:

$$\mathsf{P}(|\mathfrak{B}| > t) \leq C\, e^{-ct^2}.$$

In particular, the Gaussian distribution is also subgaussian. Then the probability that a randomly sampled T satisfies Eq. (44) can be shown to exceed $1/\ell$. The union bound then provides an estimate on the number of samplings of T necessary to guarantee Eq. (44) with a desired probability.

Some remarks are in order.

1. Computationally, Eq. (44) is applied to some given data as follows: given a set $\mathcal{X}$ of ℓ vectors in $\mathbb{R}^n$ and some error tolerance $\varepsilon \in (0, 1)$, find an appropriate $k = O(\frac{1}{\varepsilon^2} \ln \ell)$, construct the $k \times n$ RP T by sampling each of its components from $\mathsf{N}(0, \frac{1}{\sqrt{k}})$, and then define the set $T\mathcal{X} = \{Tx \mid x \in X\}$. By the JLL, $T\mathcal{X}$ is approximately congruent to $\mathcal{X}$ in the sense of Eq. (44); however, $T\mathcal{X} \subset \mathbb{R}^k$, whereas $\mathcal{X} \subset \mathbb{R}^n$, and, typically, $k \ll n$.
2. The computation of an appropriate k would appear to require an estimation of the constant in the expression $O(\frac{1}{\varepsilon^2} \ln \ell)$. Values computed theoretically are often so large as to make the technique useless in practice. As far as we know, this constant has only been computed empirically in some cases (Venkatasubramanian and Wang 2011), ending up with an estimation of the constant at 1.8 (which is the value we employed in most of our experiments).
3. The term $\frac{1}{\sqrt{k}}$ is the standard deviation of the normal distribution from which the components of T must be sampled. It corresponds to a scaling of the vectors in $T\mathcal{X}$ induced by the loss in dimensions (see Theorem. 6).
4. In the expression $O(\frac{1}{\varepsilon^2} \ln \ell)$, the logarithmic term is the one that counts for analysis purposes, but in practice ε^{-2} can be large. Our advice is to take $\varepsilon \in (0.1, 0.2)$ and then fine-tune ε according to results.
5. Surprisingly, the target dimension k is independent of the original dimension n.
6. Even if the data in $\mathcal{X}$ are sparse, $T\mathcal{X}$ ends up being dense. Different classes of sparse RPs have been investigated (Achlioptas 2003; Kane and Nelson 2014) to tackle this issue. A simple algorithm (D'Ambrosio et al. 2019, §5.1) consists in initializing T as the $k \times n$ zero matrix, and then only fill components using sam-

ples from $\mathsf{N}(0, \frac{1}{\sqrt{kp}})$ with some given probability p. The value of p corresponds to the density of T. In general, and empirically, it appears that the larger n and ℓ are, the sparser T can be.

7. Obviously, a Euclidean space of dimension k can embed at most k orthogonal vectors. An easy but surprising corollary of the JLL is that as many as $O(2^k)$ approximately orthogonal vectors can fit in $\mathbb{R}^k$. This follows by Vu et al. (2018, Prop. 1) applied to the standard basis $S = \{e_1, \ldots, e_n\}$ of $\mathbb{R}^n$: we obtain $\forall i < j \leq n\ (-\varepsilon \leq \langle Te_i, Te_j\rangle - e_i e_j \leq \varepsilon)$, which implies $|\langle Te_i, Te_j\rangle| \leq \varepsilon$ with $TS \subset \mathbb{R}^k$ and $k = O(\ln n)$. Therefore TS is a set of $O(2^k)$ almost orthogonal vectors in $\mathbb{R}^k$, as claimed.
8. Typical applications of RPs arise in clustering databases of large files (e.g., e-mails, images, songs, and videos), performing basic tasks in ML (e.g., k-means (Boutsidis et al. 2010), k-nearest neighbors (k-NN) (Indyk and Naor 2007), robust learning (Arriaga and Vempala 2006) and more (Indyk 2001), and approximating large MP formulations (e.g., LP, QP, see Sect. 7.3.3).
9. The JLL seems to suggest that most of the information encoded by the congruence of a set of vectors can be maintained up to an ε tolerance in much smaller dimensional spaces. This is not true for sets of vectors in low dimensions. For example, with $n \in \{2, 3\}$ a few attempts immediately show that RPs yield sets of projected vectors which are necessarily incongruent with the original vectors.

In this paper, we do not give a complete proof of the JLL, since many different ones have already been provided in research articles (Johnson and Lindenstrauss 1984; Dasgupta and Gupta 2002; Indyk and Motwani 1998; Ailon et al. 2006; Kane and Nelson 2014; Matoušek 2008; Allen-Zhu et al. 2014) and textbooks (Vempala 2004; Matoušek 2013; Kantor et al. 2015; Vershynin 2018). We only prove the first part of the proof, namely the easy result that RPs preserve norms on average. This provides an explanation for the variance $1/k$ of the distribution from which the components of T are sampled.

Theorem 6 *Let T be a $k \times n$ RP sampled from $\mathsf{N}(0, \frac{1}{\sqrt{k}})$, and $u \in \mathbb{R}^n$; then $\mathsf{E}(\|Tu\|_2^2) = \|u\|_2^2$.*

Proof We prove the claim for $\|u\|_2 = 1$; the result will follow by scaling. For each $i \leq k$, we define $v_i = \sum_{j \leq n} T_{ij} u_j$. Then $\mathsf{E}(v_i) = \mathsf{E}\big(\sum_{j \leq m} T_{ij} u_j\big) = \sum_{j \leq m} \mathsf{E}(T_{ij}) u_j = 0$. Moreover:

$$\mathsf{Var}(v_i) = \sum_{j \leq m} \mathsf{Var}(T_{ij} u_j) = \sum_{j \leq m} \mathsf{Var}(T_{ij}) u_j^2 = \sum_{j \leq m} \frac{u_j^2}{k} = \frac{1}{k}\|u\|^2 = \frac{1}{k}.$$

Now: $\frac{1}{k} = \mathsf{Var}(v_i) = \mathsf{E}(v_i^2 - (\mathsf{E}(v_i))^2) = \mathsf{E}(v_i^2 - 0) = \mathsf{E}(v_i^2)$. Hence:

Table 1 Values of $\|sTT^\top - I_d\|$ in function of s, n

s	n									
	1e3	2e3	3e3	4e3	5e3	e3	7e3	8e3	9e3	1e4
$1/n$	9.72	7.53	6.55	5.85	5.36	5.01	4.71	4.44	4.26	4.09
$1/d$	5e1	1e2	1.5e2	2e2	2.5e2	3e2	3.5e2	3.9e2	4.4e2	4.8e2
1	2e5	4e5	6e5	8e5	1e6	1.2e6	1.4e6	1.6e6	1.8e6	2e6

$$\mathsf{E}(\|Tu\|^2) = \mathsf{E}(\|v\|^2) = \mathsf{E}\left(\sum_{i\le k} v_i^2\right) = \sum_{i\le k} \mathsf{E}(v_i^2) = \sum_{i\le k} \frac{1}{k} = 1,$$

as claimed. □

7.3.2 Approximating the identity

If T is a $k \times n$ RP where $k = O(\varepsilon^{-2} \ln n)$, both $TT^\top$ and $T^\top T$ have some relation with the identity matrices I_k and I_n. This is a lesser known phenomenon, so it is worth discussing it here in some detail.

We look at $TT^\top$ first. By Zhang et al. (2013, Cor. 7) for any $\epsilon \in (0, \frac{1}{2})$, we have:

$$\|\frac{1}{n} T\, T^\top - I_k\|_2 \le \varepsilon$$

with probability at least $1 - \delta$ as long as $n \ge \frac{(k+1)\ln(2k/\delta)}{\mathcal{C}\epsilon^2}$, where $\mathcal{C} \ge \frac{1}{4}$ is a constant.

In Table 1 we give values of $\|s\,TT^\top - I_d\|_2$ for $s \in \{1/n, 1/d, 1\}$, $n \in \{1000, 2000, \ldots, 10{,}000\}$ and $d = \lceil \ln(n)/\epsilon^2 \rceil$ where $\epsilon = 0.15$. It is clear that the error decreases as the size increases only in the case $s = \frac{1}{n}$. This seems to indicate that the scaling is a key parameter in approximating the identity.

Let us now consider the product $T^\top T$. It turns out that, for each fixed vector x not depending on T, the matrix $T^\top T$ behaves like the identity with respect to x.

Theorem 7 *Given any fixed $x \in \mathbb{R}^n$, $\epsilon \in (0, 1)$ and an RP $T \in \mathbb{R}^{d\times n}$, there is a universal constant $\mathcal{C}$, such that*:

$$-\mathbf{1}\varepsilon \le T^\top Tx - x \le \mathbf{1}\varepsilon \tag{45}$$

with probability at least $1 - 4e^{\mathcal{C}\epsilon^2 d}$.

Proof By definition, for each $i \le n$ we have $x_i = \langle e_i, x\rangle$, where e_i is the ith unit coordinate vector. By elementary linear algebra, we have $\langle e_i, T^\top Tx\rangle = \langle Te_i, Tx\rangle$. By D'Ambrosio et al. (2019, Lemma 3.1), for $i \le n$ we have:

$$\langle e_i, x\rangle - \epsilon\|x\|_2 \le \langle Te_i, Tx\rangle \le \langle e_i, x\rangle + \epsilon\|x\|$$

with arbitrarily high probability, which implies the result. □

Table 2 Average values of diagonal and off-diagonal components of $T^\top T$ in function of n, where T is a $k \times n$ RP with $k = O(\epsilon^{-2} \ln n)$ and $\epsilon = 0.15$

n	Diagonal	Off-diag
500	1.00085	0.00014
1000	1.00069	0.00008
1500	0.99991	− 0.00006
2000	1.00194	0.00005
2500	0.99920	− 0.00004
3000	0.99986	− 0.00000
3500	1.00044	0.00000
4000	0.99693	0.00000

One might be tempted to infer from Theorem 7 that $T^\top T$ "behaves like the identity matrix" (independently of x). This is generally false: Theorem 7 only holds for a given (fixed) x.

In fact, since T is a $k \times n$ matrix with $k < n$, $T^\top T$ is a square symmetric psd $n \times n$ matrix with rank k, hence $n - k$ of its eigenvalues are zero—and the nonzero eigenvalues need not have value one. On the other hand, $T^\top T$ looks very much like a slightly perturbed identity, on average, as shown in Table 2.

7.3.3 Using RPs in MP

Random projections have mostly been applied to probabilistic approximation algorithms. By randomly projecting their (vector) input, one can execute algorithms with lower dimensional vector more efficiently. The approximation guarantee is usually derived from the JLL or similar results.

A line of research about applying RPs to MP formulations was started in Vu et al. (2019), Vu et al. (2018), Vu et al. (2019), and D'Ambrosio et al. (2019). Whichever algorithm one may choose to solve the MP, the RP properties guarantee an approximation on optimality and/or feasibility. Thus, this approach leads to stronger/more robust results with respect to applying RPs to algorithmic input.

Linear and integer feasibility problems (i.e. LP and MILP formulations without objective function) are investigated in Vu et al. (2019) from a purely theoretical point of view. The effect of RPs on LPs (with nonzero objective) is investigated in Vu et al. (2018), both theoretically and computationally. Specifically, the randomly projected LP formulation is shown to have bounded feasibility error and an approximation guarantee on optimality. The computational results suggest that the range of practical application of this technique starts with relatively small LPs (thousands of variables/constraints). In both Vu et al. (2018, 2019) we start from a (MI)LP in standard form:

$$\mathcal{P} \equiv \min\{c^\top x \mid Ax = b \wedge x \geq 0 \wedge x \in X\},$$

(where $X = \mathbb{R}^n$ or $\mathbb{Z}^n$, respectively), and obtain a randomly projected formulation under the RP $T \sim \mathsf{N}^{n \times k}(0, \frac{1}{\sqrt{k}})$ with the form:

$$T\mathcal{P} \equiv \min\left\{c^\top x \mid TAx = Tb \wedge x \geq 0 \wedge x \in X\right\},$$

i.e., T reduces the number of constraints in $\mathcal{P}$ to $O(\ln n)$, which can therefore be solved more efficiently.

The RP technique in Vu et al. (2019), D'Ambrosio et al. (2019) is different, insofar as it targets the number of variables. In D'Ambrosio et al. (2019), we consider a QP of the form:

$$\mathcal{Q} \equiv \max\left\{x^\top Qx + c^\top x \mid Ax \leq b\right\},$$

where Q is $n \times n$, $c \in \mathbb{R}^n$, A is $m \times n$, and $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$. This is projected as follows:

$$T\mathcal{Q} \equiv \max\left\{u^\top \bar{Q}x + \bar{c}^\top u \mid \bar{A}u \leq b\right\},$$

where $\bar{Q} = TQT^\top$ is $k \times k$, $\bar{A} = AT^\top$ is $m \times k$, $\bar{c} = Tc$ is in $\mathbb{R}^k$, and $u \in \mathbb{R}^k$. In Vu et al. (2019) we consider a QCQP $\mathcal{Q}'$ like $\mathcal{Q}$ but subject to a ball constraint $\|x\|_2 \leq 1$. In the projected problem $T\mathcal{Q}'$, this is replaced by a ball constraint $\|u\|_2 \leq 1$. Both (D'Ambrosio et al. 2019; Vu et al. 2019) are both theoretical and computational. In both cases, the number of variables of the projected problem is $O(\ln n)$.

In applying RPs to MPs, one solves the smaller projected problems to obtain an answer concerning the corresponding original problems. In most cases, one has to devise a way to retrieve a solution for the original problem using the solution of the projected problem. This may be easy or difficult depending on the structure of the formulation and the nature of the RP.

8 Distance instability

Most of the models and methods in this survey are based on the concept of distance: usually Euclidean, occasionally with other norms. The k-means algorithm (Sect. 5.1.1) is heavily based on Euclidean distances in Step 2 (p. 20), where the reassignment of a point to a cluster is carried out based on proximity: in particular, one way to implement Step 2 is to solve a 1-nearest neighbor problem. The training of an ANN (Sect. 5.1.2) repeatedly solves a minimum-distance subproblem in Eq. (10). In spectral clustering (Sect. 5.2.1) we have a Euclidean norm constraint in Eq. (12). All DGP solution methods (Sect. 6), with the exception of incidence vectors (Sect. 6.2.1), are concerned with distances by definition. PCA (Sect. 7.1), in its interpretation of a modified MDS, can be seen as another solution method for the DGP. Barvinok's naive algorithm (Sect. 7.2) is a dimensional reduction method for SDPs the analysis of which is based on a distance bound; moreover, it was successfully applied to the DGP (Liberti and Vu 2018). The RP-based methods discussed in Sect. 7.3 have all been derived from the JLL (Sect. 7.3.1), which is a statement about the Euclidean distance. We also note that the focus of this survey is on typical DS problems, which are usually high-dimensional.

It is therefore absolutely essential that all of these methods should be able to take robust decisions based on comparing distance values computed on pairs of high-dimensional vectors. It turns out, however, that smallest and largest distances $D_{\min}, D_{\max}$ of a random point $Z \in \mathbb{R}^n$ to a set of random points $X_1, \ldots, X_\ell \subset \mathbb{R}^n$ are almost equal (and, hence, difficult to compare) as $n \to \infty$ under some reasonable conditions. This holds for any distribution used to sample Z, X_i. This result, first presented in Beyer et al. (1998) and subsequently discussed in a number of papers (Hinneburg et al. 2000; Aggarwal et al. 2001; François et al. 2007; Durrant and Kabán 2009; Radovanović et al. 2010; Mansouri and Khademi 2015; Flexer and Schnitzer 2015), appears to jeopardize all of the material presented in this survey, and much more beyond. The phenomenon leading to the result is known as distance instability and concentration of distances.

8.1 Statement of the result

Let us look at the exact statement of the distance instability result.

First, we note that the points $Z, X_1, \ldots, X_\ell$ are not given points in $\mathbb{R}^n$ but rather multivariate random variables with n components, so distance instability is a purely statistical statement rather than a geometric one. We consider:

$$\begin{aligned} Z &= (Z_1, \ldots, Z_n) \\ \forall i \leq \ell \quad X_i &= (X_{i1}, \ldots, X_{in}), \end{aligned}$$

where $Z_1, \ldots, Z_n$ are random variables with distribution $\mathcal{D}_1$; $X_{11}, \ldots, X_{\ell n}$ are random variables with distribution $\mathcal{D}_2$; and all of these random variables are independently distributed.

Second, $D_{\min}, D_{\max}$ are functions of random variables:

$$D_{\min} = \min\{\mathsf{dist}(Z, X_i) \mid i \leq \ell\} \tag{46}$$

$$D_{\max} = \max\{\mathsf{dist}(Z, X_i) \mid i \leq \ell\}, \tag{47}$$

and are therefore random variables themselves. In the above, dist denotes a function mapping pairs of points in $\mathbb{R}^n$ to a non-negative real number, which makes distance instability a very general phenomenon. Specifically, dist need not be a distance at all.

Third, we now label every symbol with an index m, which will be used to compute limits for $m \to \infty$: Z^m, X^m, $\mathcal{D}_1^m$, $\mathcal{D}_2^m$, $D_{\min}^m$, $D_{\max}^m$, dist^m. We shall see that the proof of the distance instability result is wholly syntactical: its steps are very simple and follow from basic statistical results. In particular, we can see m as an abstract parameter under which we shall take limits, and the proof will hold. Since the proof holds independently of the value of n, it also holds if we assume that $m = n$, i.e., if we give m the interpretation of dimensionality of the Euclidean space embedding the points. While this assumption is not necessary for the proof to hold, it may simplify its understanding: $m = n$ makes the proof somewhat less general, but it gives the above indexing a more concrete meaning. Specifically, $Z, X, \mathcal{D}, D, \mathsf{dist}$ are points,

distributions, extreme distance values and a distance function in dimension m, and the limit $m \to \infty$ is a limit taken on increasing dimension.

Fourth, the "reasonable conditions" referred to above for the distance instability result to hold are that there is a constant $p > 0$ such that:

$$\exists i \leq \ell \quad \lim_{m\to\infty} \mathsf{Var}\left(\frac{(\mathsf{dist}(Z^m, X_i^m))^p}{\mathsf{E}((\mathsf{dist}(Z^m, X_i^m))^p)}\right) = 0. \tag{48}$$

A few remarks on Eq. (48) are in order.

(a) The existential quantifier encodes the fact that the X_i are all identically distributed, so a statement involving variance and expectation of quantities depending on the X_i random variables holds for all $i \leq \ell$ if it holds for just one X_i.
(b) The constant p simply gives more generality to the result, but plays no role whatsoever in the proof; it can be used to simplify computations when dist is an ℓ_p norm.
(c) The fraction term in Eq. (48) measures a spread relative to an expectation. Requiring that the limit of this relative spread goes to zero for increasing dimensions looks like an asymptotic concentration requirement (hence the alternative name "distance concentration" for the distance instability phenomenon). Considering the effect of concentration of measure phenomena in high dimensions (Sect. 7.2.2), distance instability might now appear somewhat less surprising.

With these premises, we can state the distance instability result.

Theorem 8 *If D^m_{min} and D^m_{max} are as in Eq. (46) and (47) and satisfy Eq. (48), then, for any $\varepsilon > 0$, we have*:

$$\lim_{m\to\infty} \mathsf{P}\left(D^m_{\mathsf{max}} \leq (1+\varepsilon) D^m_{\mathsf{min}}\right) = 1. \tag{49}$$

Theorem 8 basically states that closest and farthest neighbors of Z are indistinguishable up to an ε. If the closest and farthest are indistinguishable, trying to discriminate between the closest and the second closest neighbors of a given point might well be hopeless due to floating point errors (note that this discrimination occurs at each iteration of the well known k-means algorithm). This is why distance instability is sometimes cited as a reason for convergence issues in k-means (Gayraud 2017).

8.2 Related results

In Beyer et al. (1998), several scenarios are analyzed to see where distance instability occurs—even if some of the requirements of distance instability are relaxed (Beyer et al. 1998, §3.5)—and where it does not (Beyer et al. 1998, §4). Among the cases where distance instability does not apply, we find the case where the data points X are well separated and the case where the dimensionality is implicitly low. Among the cases where it does apply, we find k-NN: in their experiments, the

authors of Beyer et al. (1998) find that k-NN becomes unstable already in the range $n \in \{10, 20\}$ dimensions. Obviously, the instability of k-NN propagates to any algorithm using k-NN, such as k-means.

Among later studies, Hinneburg et al. (2000) proposes an alternative definition of dist where high-dimensional points are projected into lower dimensional spaces. In Hinneburg et al. (2000), the authors study the impact of distance instability on different ℓ_p norms, and concludes that smallest values of p lead to more stable norms; in particular, quasinorms with $0 < p < 1$ are considered. Some counterexamples are given against a generalization of this claim for quasinorms in François et al. (2007). In Durrant and Kabán (2009), the converse of Theorem 8 is proved, namely that Eq. (48) follows from Eq. (49): from this fact, the authors find practically relevant cases where Eq. (48) is not verified, and propose them as "good" examples of where k-means can help. In Mansouri and Khademi (2015), the authors propose multiplicative functions dist and show that they are robust with respect to distance instability. In Radovanović et al. (2010), distance instability is related to "hubness", i.e., the number of times a point appears among the k nearest neighbors of other points. In Flexer and Schnitzer (2015), an empirical study is provided which shows how to show an appropriate ℓ_p norm that should avoid distance instability with respect to hubness.

8.3 The proof

The proof of the instability theorem can be found in Beyer et al. (1998). We repeat it here to demonstrate the fact that it is "syntactical": every step follows from the previous ones by simple logical inference. There is no appeal to any results other than convergence in probability, Slutsky's theorem, and a simple corollary as shown below. The proof does not pass from object language to meta-language, nor does it require exotic interpretations of symbols in complicated contexts. Although one may find this result surprising, there appears to be no reason to doubt it, and no complication in the proof warranting sophisticated interpretations. The only point worth re-stating is that this is a result about probability distributions, not about actual instances of real data.

Lemma 1 *Let* $\{B^m \mid m \in \mathbb{N}\}$ *be a sequence of random variables with finite variance. Assume that* $\lim_{m\to\infty} \mathsf{E}(B^m) = b$ *and that* $\lim_{m\to\infty} \mathsf{Var}(B^m) = 0$. *Then*:

$$\forall \varepsilon > 0 \quad \lim_{m\to\infty} \mathsf{P}(\|B^m - b\| \leq \varepsilon) = 1. \tag{50}$$

A random variable sequence satisfying Eq. (50) is said to converge in probability to b. This is denoted $B^m \to_{\mathsf{P}} b$.

Lemma 2 (Slutsky's theorem, Wikipedia: Slutsky's theorem 2019) *Let* $\{B^m \mid m \in \mathbb{N}\}$ *be a sequence of random variables, and* $g : \mathbb{R} \to \mathbb{R}$ *be a continuous function. If* $B^m \to_{\mathsf{P}} b$ *and* $g(b)$ *exists, then* $g(B^m) \to_{\mathsf{P}} g(b)$.

Corollary 1 *If* $\{A^m \mid m \in \mathbb{N}\}$ *and* $\{B^m \mid m \in \mathbb{N}\}$ *are sequences of random variables, such that* $A^m \to_{\mathsf{P}} a$ *and* $B^m \to_{\mathsf{P}} b \neq 0$, *then* $\frac{A^m}{B^m} \to_{\mathsf{P}} \frac{a}{b}$.

Proof of Theorem 8 Let $\mu_m = \mathsf{E}((d^m(Z^m, X_i^m))^p)$. We note that μ_m is independent of i, since all X_i^m are identically distributed.

We claim $V_m = \frac{(d^m(Z^m, X_i^m))^p}{\mu_m} \to_{\mathsf{P}} 1$:

- we have $\mathsf{E}(V_m) = 1$, since it is a random variable over its mean: hence, trivially, $\lim_m \mathsf{E}(V_m) = 1$;
- by the hypothesis of the theorem (Eq. (48)), $\lim_m \mathsf{Var}(V_m) = 0$;
- by Lemma 1, $V_m \to_{\mathsf{P}} 1$, which establishes the claim.

Now, let $\mathbf{V}^m = (V_m \mid i \leq \ell)$. By the claim above, we have $\mathbf{V}^m \to_{\mathsf{P}} \mathbf{1}$. Now, by Lemma 2, we obtain $\min(\mathbf{V}^m) \to_{\mathsf{P}} \min(\mathbf{1}) = 1$ and, similarly, $\max(\mathbf{V}^m) \to_{\mathsf{P}} 1$. By Corollary 1, $\frac{\max(\mathbf{V}^m)}{\min(\mathbf{V}^m)} \to_{\mathsf{P}} 1$. Therefore:

$$\frac{D_{\mathsf{max}}^m}{D_{\mathsf{min}}^m} = \frac{\mu_m \max(\mathbf{V}^m)}{\mu_m \min(\mathbf{V}^m)} \to_{\mathsf{P}} 1.$$

By definition of convergence in probability, we have:

$$\forall \varepsilon > 0 \quad \lim_{m \to \infty} \mathsf{P}\big(|D_{\mathsf{max}}^m / D_{\mathsf{min}}^m - 1| \leq \varepsilon\big) = 1.$$

Moreover, since $\mathsf{P}(D_{\mathsf{max}}^m \geq D_{\mathsf{min}}^m) = 1$, we have:

$$\mathsf{P}(D_{\mathsf{max}}^m \leq (1+\varepsilon) D_{\mathsf{min}}^m) = \mathsf{P}(D_{\mathsf{max}}^m / D_{\mathsf{min}}^m - 1 \leq \varepsilon) = \mathsf{P}(|D_{\mathsf{max}}^m / D_{\mathsf{min}}^m - 1| \leq \varepsilon) = 1.$$

The result follows by taking the limit as $m \to \infty$. □

8.4 In practice

In Fig. 7, we show how ε (Eq. (49)) varies with increasing dimension n (recall we assume $m = n$) between 1 and 10,000. It is clear that ε decreases very rapidly towards zero, and then reaches its asymptotic value more slowly. On the other hand, ε is the distortion between minimum and maximum distance values; most algorithms need to discriminate between smallest and second smallest distance values.

Most of the papers listed in Sect. 8.2 include empirical tests which illustrate the impact and limits of the distance instability phenomenon.

9 An application to neural networks

In this last section, we finally show how several concepts explained in this survey can be used conjunctively. We shall consider a natural language processing task (Sect. 4) where we cluster some sentences (Sect. 5) using an ANN (Sect. 5.1.2) with

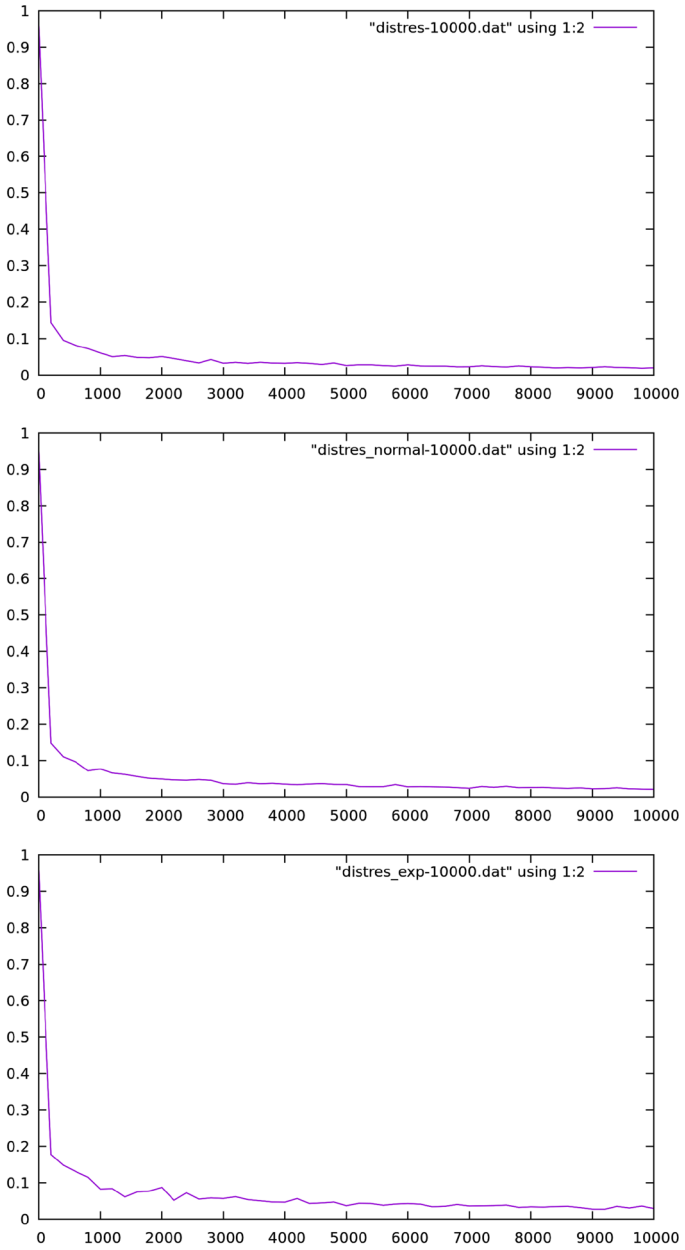

Fig. 7 Plots of ϵ versus n for the uniform distribution on $[0, 1]$ (above), $\mathsf{N}(0, 1)$ (center), and the exponential distribution with parameter 1 (below)

different training sets $T = (X, Y)$. We compare ANN performances depending on the training set used.

The input set X is a vector representation of the input sentences. The output set Y is a vectorial representation of cluster labels: we experiment with (a) clusterings obtained by running k-means (Sect. 5.1.1) on the input sets, and (b) a clustering found by a modularity maximization heuristic (Sect. 5.2.2). All of these clusterings are considered "ground truth" sets Y; we would like our ANN to learn to associate with various types of input vector sets X representing the sentences. The sentences to be clustered are first transformed into graphs (Sect. 4.2), and then into vectors (Sect. 6), which then undergo dimensionality reduction (Sect. 7).

Our goal is to compare the results obtained by the same ANN with different vector representations for the same text: most notably, the comparison will establish how well or poorly different input vector sets can predict corresponding ground truth outputs. We will focus specifically on a comparison of the well-known incidence vectors (Sect. 6.2.1) embeddings with respect to the newly proposed DGP methods which we surveyed in Sect. 6.

In our implementations, all our code was developed using Python 3 (van Rossum 2019).

9.1 Performance measure

We are going to measure the performance quality of the error of an ANN, which is based on a comparison of its output with the ground truth that the ANN is supposed to learn. Using the notation of Sect. 5.1.2, if the ANN output for a given input $x \in \mathbb{R}^n$ consists of a vector $y \in \mathbb{R}^k$, and if the ground truth corresponding to x is $z \in \mathbb{R}^k$, then we define the error as the *loss* function:

$$\mathsf{loss}(y, z) = \|y - z\|_2. \tag{51}$$

An ANN $\mathcal{N} = (G, T, \phi)$ is usually evaluated over many (input,output) pairs. Let $\hat{X} \subset \mathbb{R}^n$ and $\hat{Y} \subset \mathbb{R}^k$ be, respectively, a set of input vectors and the corresponding set of output vectors evaluated by the trained ANN. Let $\hat{Z}$ be a set of ground truth vectors corresponding to $\hat{X}$, and assume $|\hat{X}| = |\hat{Y}| = |\hat{Z}| = q$. The cumulative loss measure evaluated on the *test set* $(\hat{X}, \hat{Z})$ is then:

$$\mathsf{loss}(\mathcal{N}) = \frac{1}{q} \sum_{i \leq q} \mathsf{loss}(y_i, z_i), \tag{52}$$

where $\hat{Y} = \{y_i \mid i \leq q\}$ and $\hat{Z} = \{z_i \mid i \leq q\}$.

9.2 A natural language processing task

Clustering of sentences in a text is a common task in Natural Language Processing. We considered "On the duty of civil disobedience" by H.D. Thoreau (Thoreau 1849; Wikipedia: Civil disobedience 2019). This text is stored in an ASCII file which can be obtained from `archive.org`. The file that we used for testing is 661146 bytes long, organized in 10108 lines and 116608 words. The text was parsed into sentences using basic methods from NLTK (Bird et al. 2009) under Python 3. Common words,

stopwords, punctuation, and unusual characters were removed, which reduced the text to 4083 sentences over a set of 11,431 "significant" words (see Sect. 9.2.1).

As mentioned above, we want to train our ANN to learn different types of clusterings:

- (k-means) obtained by running the k-means unsupervised clustering algorithm (Sect. 5.1.1) over the different vector representations of the sentences in the text;
- (sentence graph) obtained by running a modularity clustering heuristic (Sect. 5.2.2) on a graph representation of the sentences in the document (see Sect. 9.2.2).

These clusterings are used as ground truths, and provide the output part of the training sets to be used by the ANN, as well as of the test sets for measuring purposes (Sect. 9.1). See Sect. 9.4.1 for more information on the construction of these clusterings.

9.2.1 Selecting the sentences

We constructed two sets of sentences.

- **The large sentence set.** Each sentence in the text file was mapped to an incidence vector of 3-grams in $\{0,1\}^{48,087}$, i.e. a dictionary of 48,087 3-grams over the text. In other words, 48,087 3-grams were found in the text, and then, each sentence was mapped to a vector having 1 at component i iff the ith 3-grams was present in the sentence. Since some sentences had fewer than 3 significant words, only 3940 sentences remained in the sentence set S, which was, therefore, represented as a $3940 \times 48{,}087$ matrix $\bar{S}$ with components in $\{0,1\}$.
- **The small sentence set.** It turns out that most of the 3-grams in the set S only appear a single time. We selected a subset $S' \subset S$ of sentences having 3-grams appearing in at least two sentences. It turns out that $|S'| = 245$, and the total number of 3-grams appearing more than once is 160. S' is, therefore, naturally represented as a 245×160 matrix $\bar{S}'$ with components in $\{0,1\}$.

We constructed training sets (Sect. 9.4) for each of these two sets. Specifically, each sentence in the text was encoded into a weighted graph-of-word (see Sect. 4.2.1) over 3-grams, with edges $\{u,v\}$ weighted by the number c_{uv} of 3-grams where the two words u, v appear. Then, each graph was mapped into a realization using DG methods (see Sect. 9.4).

9.2.2 Construction of a sentence graph

In this section, we describe the method used to construct a sentence graph $G^{\mathrm{s}} = (S,E)$ from the text, which is used to produce a ground truth for the (sentence graph) type. G^{s} is then clustered using the greedy modularity clustering heuristic in the Python library `networkX` (Hagberg et al. 2008).

Each sentence in the text is encoded into a weighted graph-of-word (see Sect. 4.2.1) over 3-grams, with edges $\{u, v\}$ weighted by the number c_{uv} of 3-grams where the two words u, v appear. The union of the graph-of-words for the sentences (contracting repeated words to a single vertex) yields a weighted graph-of-word G^{w} for the whole text.

The graph $G^{\mathrm{w}} = (W, F)$ is then "projected" onto the set S of sentences as follows. We define the logical proposition $P(u, v, s, t)$ to mean $(u \in s \wedge v \in t) \vee (v \in s \wedge u \in t)$ for words u, v and sentences s, t. The edge set E of G^{s} is then defined by the following implication:

$$\forall \{u, v\} \in F,\ s, t \in S \quad P(u, v, s, t) \rightarrow \{s, t\} \in E.$$

In other words, s, t form an edge in E if two words u, v in s, t (respectively) or t, s form an edge in F. For each edge $\{s, t\} \in E$, the weight w_{st} is given by:

$$w_{st} = \sum_{\substack{\{u, v\} \in F \\ P(u, v, s, t)}} c_{uv},$$

with edge weights meaning similarity.

9.3 The ANN

We consider a very simple ANN $\mathcal{N} = (G, T, \phi)$. In the terminology of Sect. 5.1.2, the underlying digraph $G = (V, A)$ is tripartite with $V = V_1 \dot{\cup} V_2 \dot{\cup} V_3$. The "input layer" V_1 has n nodes, where n is the dimensionality of the input vector set X. The "output layer" V_3 has a single node. The "hidden layer" V_2 has a constant number of nodes (20 in our experiments). The training set T is discussed in Sect. 9.4. We adopt the piecewise-linear mapping known as *rectified linear unit* (ReLu) (Wikipedia: Rectifier 2019) for the activation functions ϕ in V_2, and a traditional sigmoid function for the single node in V_3. Both types of activation functions map to [0, 1].

We implemented $\mathcal{N}$ using the Python library `keras` (Chollet et al. 2015), which is a high-level API running over TensorFlow (Abadi et al. 2015). The default configuration was chosen for all layers. We used the Adam solver (Kingma and Ba 2015) to train the network. Each training set was split in three parts: 35% of the vectors were used for training, 35% for validation (a training phase used for deciding values of any model parameter aside from v, b, w, if any exist, and/or for deciding when to stop the training phase), and 30% for testing. The performance of the ANN is measured using the loss function in Eq. (52).

9.4 Training sets

Our goal is to compare training sets $T = (X, Y)$ where the vectors in X are constructed in different ways. In particular, we consider input sets $X(\sigma, \mu, \rho)$ where:

- $\sigma \in \Sigma = \{S', S\}$ is the sentence set: $\sigma = S'$ corresponds to the small set with 245 sentences; $\sigma = S$ corresponds to the large set with 3940 sentences;
- $\mu \in M = \{\mathsf{inc}, \mathsf{uie}, \mathsf{qrt}, \mathsf{sdp}\}$ is the method used to map sentences to vectors: inc are the incidence vectors (Sect. 6.2.1), uie is the universal isometric embedding (Sect. 6.2.2), qrt is the unconstrained quartic (Sect. 6.1.1), sdp is the SDP (Sect. 6.1.3);
- $\rho \in R = \{\mathsf{pca}, \mathsf{rp}\}$ is the dimensional reduction method used: pca is PCA (Sect. 7.1), rp are RPs (Sect. 7.3).

The methods in M were all implemented using Python 3 with some well known external libraries (e.g., `numpy`, `scipy`). Specifically, qrt was implemented using the Ipopt [47] NLP solver, and sdp was implemented using the SCS (O'Donoghue et al. 2016) SDP solver. As for the dimensional reduction methods in R, the PCA implementation of choice was the probabilistic PCA algorithm implemented in the Python library `scikit-learn` (Pedregosa et al. 2011). The chosen RPs were the simplest: each component of the RP matrices was sampled from an appropriately scaled zero-mean Gaussian distribution (Theorem 6).

9.4.1 The output set

The output set Y should naturally contain discrete values, namely the labels of the h clusters $\{1, 2, \ldots, h\}$ in the ground truth clusterings. We map these values to scalars in [0, 1] (or, according to Sect. 5.1.2, to k-dimensional vectors with $k = 1$) as follows. We divide the range [0, 1] into $h - 1$ equal sub-intervals of length $1/(h-1)$, and hence h discrete values in [0, 1]. Then we assign labels to sub-intervals endpoints: label j is mapped to $(j-1)/(h-1)$ (for $1 \leq j \leq h$).

As mentioned above, we consider two types of output sets:

- (k-means) for each input set $X(\sigma, \mu, \rho)$, we obtained an output set $Y(\sigma, \mu, \rho)$ using k-means (Sect. 5.1.1) implementation in `scikit-learn` (Pedregosa et al. 2011) on the vectors in X, for each sentence set $\sigma \in \Sigma$, method $\mu \in M$, and dimensional reduction method $\rho \in R$;
- (sentence graph) for each sentence set $\sigma \in \Sigma$, we constructed a sentence graph as detailed in Sect. 9.2.2.

9.4.2 Realizations to vectors

The inc method (Sect. 6.2.1) is the only one (in our benchmark) that can natively map sentences of various lengths into vectors all having the same number of components.

For all other methods in $M \setminus \{\mathsf{inc}\}$, we loop over sentences (in small/large sets S', S). For each sentence, we construct its graph-of-words (Sect. 4.2.1). We then realize it in some arbitrary dimensional Euclidean space $\mathbb{R}^K$ (specifically, we chose $K = 10$) using uie, qrt, sdp. At this point, we are confronted with the following difficulty: a realization of a graph G with p vertices in $\mathbb{R}^K$ is a $p \times K$ matrix, and we have as many graphs G as we have sentences, with p varying over the number

Table 3 Training set statistics for $X(\sigma, \mu, \rho)$ and corresponding output sets in the (k-means) class

μ	$\lvert\sigma\rvert = 245$				$\lvert\sigma\rvert = 3940$			
ρ	inc	uie	qrt	sdp	inc	uie	qrt	sdp
Dimensionality of input vectors								
pca	3	159	244	200	3	10	400	400
rp	100	248	248	248	373	373	373	373
Original	160	1140	1140	1140	48,087	1460	1460	1460
Number of clusters to learn								
pca	4	3	11	6	3	8	9	14
rp	4	3	7	5	3	9	16	14

of unique words in the sentences (i.e., the cardinalities of the vertex sets of the graphs-of-words).

To reduce all of these differently sized realizations to vectors having the same dimension, we employ the following procedure. Given realizations $\{x^i \in \mathbb{R}^{p_i \times K} \mid i \in \sigma\}$, where σ is the set of sentences (for $\sigma \in \Sigma$) and x^i realizes the graph-of-word of sentence $i \in \sigma$:

1. we stack the columns of x^i so as to obtain a single vector $\hat{x}^i \in \mathbb{R}^{p_i K}$ for each $i \in \sigma$;
2. we let $\hat{n} = \max_i p_i K$ be the maximum dimensionality of the stacked realizations;
3. we pad every realization vector $\hat{x}^i$ shorter than $\hat{n}$ with zeros to achieve dimension $\hat{n}$ for stacked realization vectors;
4. we form the $s \times \hat{n}$ matrix $\hat{X}$ having $\hat{x}^i$ as its ith row (for $i \in \sigma$ and with $s = |\sigma|$);
5. we reduce the dimensionality of $\hat{X}$ to an $s \times n$ matrix X with pca or rp.

9.5 Computational comparison

We discuss the details of our training sets, a validation test, and the comparison tests.

9.5.1 Training set statistics

In Table 3, we report the dimensionalities of the vectors in the input parts $X(\sigma, \mu, \rho)$ of the training sets, as well as the number of clusters in the output sets $Y(\sigma, \mu, \rho)$ of the (k-means) class. We recall that the number of clusters was found with k-means in the `scikit-learn` implementation. The choice of 'k' corresponds to the smallest number of clusters giving a nontrivial clustering (with "trivial" meaning having a cluster of zero cardinality, or too close to zero relative to the set size, only possibly allowing some outlier clusters with a single element). Some more remarks follow.

– For $\rho =$ pca, we employed the smallest dimension, such that the residual variance in the neglected components was almost zero; this ranges from 3 to 244 in

Table 3. For the two cases where the dimensionality reduction was set to 400 (qrt and sdp in the large sentence set S), the residual variance was nonzero.

- It is interesting that for $\mu = \mathsf{uie}$ we have higher projected dimensionality (248) in the small set S' than in the large set S (10): this depends on the fact that the large set has more easily distinguishable clusters (8 found by k-means) than the small set (only 3 found by k-means). The dimension of $X(\mathsf{inc}, \mathsf{pca}, S)$ is smaller (3) than that of $X(\mathsf{uie}, \mathsf{pca}, S)$ (10), even though the original number of dimensions of the former (48,087) vastly exceeds that of the latter (1460) for the same reason.
- The training sets $X(\sigma, \mathsf{inc}, \mathsf{pca})$ are the smallest-dimensional ones (for $\sigma \in \{S', S\}$): they are also "degenerate", in the sense that the vectors in a given clusters are all equal; the co-occurrence patterns of the incidence vectors conveyed relatively little information to this vectorial sentence representation.
- The RP-based dimensionality reduction method yields the same dimensionality (373) of $X(\mu, \mathsf{rp}, S)$ for $\mu \in M$. This occurs because the target dimensionality in RP depends on the number of vectors, which is the same for all methods (3940), rather than on the number of dimensions (see Sect. 7.3).

There is one output set in the (sentence graph) class for each $\sigma \in \Sigma$. For $\sigma = S'$, we have $|V| = 245$, $|E| = 28{,}519$, and 230 clusters, with the first 5 clusters having 6, 5, 4, 3, 2 elements, and the rest having a single element. For $\sigma = S$ we have $|V| = 3940, |E| = 7{,}173{,}633$, and 3402 clusters, with the first 10 clusters having 161 , 115, 62, 38, 34, 29, 19, 16, 14, 11 elements, and the rest having fewer than 10 elements.

9.5.2 Comparison tests

We first report the comparative results of the ANN on:

$$T = (X(\sigma, \mu_1, \rho_1), Y(\sigma, \mu_2, \rho_2))$$

for $\sigma \in \Sigma$, $\mu_1, \mu_2 \in M$, $\rho_1, \rho_2 \in R$. The sums in the rightmost columns of Table 4 are only carried out on terms obtained with an input vector generation method μ_1 different from the method μ_2 used to obtain the ground truth clustering via k-means (since we want to compare methods). The results corresponding to cases where $\mu_1 = \mu_2$ are emphasized in italics in the table. The best performance sums are emphasized in boldface, and the worst are shown in grey.

According to Table 4, for the small sentence set, the best method is inc, but qrt and sdp are not far behind; the only really imprecise method is uie. For the large sentence set, the best method is qrt, with sdp not far behind; both inc, uie are imprecise.

In Table 5, which has a similar format as Table 4, we report results on training sets:

$$\bar{T} = (X(\sigma, \mu, \rho), \bar{Y}(\sigma))$$

for $\sigma \in \Sigma$, $\mu \in M$, $\rho \in R$, where $\bar{Y}(\sigma)$ are output sets of the (sentence graph) class. For the small set, inc is the best method (independently of ρ), with ($\mu = \mathsf{sdp}$, $\rho = \mathsf{pca}$)

Table 4 Comparison tests on output sets of (k-means) class

Training set outputs									
μ	inc	inc	uie	uie	qrt	qrt	sdp	sdp	sum
ρ	pca	rp	pca	rp	pca	rp	pca	rp	$\mu' \neq \mu$
Training set inputs									
$\|\sigma\|$	245								
inc pca	*0.061*	*0.042*	0.059	0.013	0.094	0.108	0.064	0.025	**0.363**
inc rp	*0.005*	*0.010*	0.055	0.015	0.104	0.109	0.065	0.025	0.373
uie pca	0.271	0.052	*0.070*	*0.169*	0.233	0.201	0.127	0.111	0.995
uie rp	0.093	0.026	*0.094*	*0.076*	0.191	0.236	0.079	0.117	0.976
qrt pca	0.082	0.067	0.105	0.047	*0.084*	*0.133*	0.071	0.087	0.459
qrt rp	0.057	0.068	0.059	0.053	*0.162*	*0.073*	0.095	0.055	0.387
sdp pca	0.106	0.063	0.067	0.022	0.106	0.135	*0.058*	*0.034*	0.499
sdp rp	0.095	0.065	0.093	0.021	0.103	0.139	*0.074*	*0.018*	0.516
$\|\sigma\|$	3940								
inc pca	*0.052*	*0.013*	0.068	0.027	0.106	0.164	0.079	0.161	0.605
inc rp	*0.001*	*0.000*	0.067	0.028	0.106	0.167	0.080	0.159	0.607
uie pca	0.063	0.022	*0.020*	*0.016*	0.124	0.201	0.070	0.127	0.607
uie rp	0.061	0.023	*0.024*	*0.023*	0.131	0.190	0.072	0.126	0.603
qrt pca	0.063	0.022	0.36	0.023	*0.038*	*0.218*	0.079	0.159	**0.382**
qrt rp	0.062	0.024	0.047	0.025	*0.120*	*0.035*	0.076	0.164	0.398
sdp pca	0.063	0.021	0.023	0.024	0.126	0.195	*0.033*	*0.149*	0.452
sdp rp	0.059	0.021	0.025	0.024	0.121	0.176	*0.083*	*0.037*	0.426

following very closely, and, in general, sdp and qrt still being acceptable; uie is the most imprecise method. For the large set, inc is against the best method, with ($\mu = \mathsf{sdp}, \rho = \mathsf{rp}$) following closely. While the other methods do not excel, the performance difference between all methods is less remarkable than with the small set.

Table 5 Comparison tests on output sets of (sentence graph) class

Training set outputs								
μ	inc	inc	uie	uie	qrt	qrt	sdp	sdp
ρ	pca	rp	pca	rp	pca	rp	pca	rp
Training inputs								
$\|\sigma\|$	245							
	0.107	0.108	0.196	0.184	0.129	0.151	0.109	0.122
$\|\sigma\|$	3940							
	0.097	0.098	0.124	0.119	0.136	0.113	0.114	0.106

10 Conclusion

We have surveyed some of the concepts and methodologies of distance geometry which are used in data science. More specifically, we have looked at algorithms (mostly based on mathematical programming) for representing graphs as vectors as a pre-processing step to performing some machine learning task requiring vectorial input.

We started with brief introductions to mathematical programming and distance geometry. We then showed some ways to represent data by graphs, and introduced clustering on vectors and graphs. Following, we surveyed robust algorithms for realizing weighted graphs in Euclidean spaces, where the robustness is with respect to errors or noise in the input data. It turns out that most of these algorithms are based on mathematical programming. Since some of these algorithms output high-dimensional vectors and/or high-rank matrices, we also surveyed some dimensional reduction techniques. We then discussed a result about the instability of distances with respect to randomly generated points.

The guiding idea in this survey is that distance geometry allows the application many supervised and unsupervised clustering techniques based on vectors to the problem of clustering on graphs. To demonstrate the applicability of this idea, we showed that vectorial representations of graphs obtained using distance geometry offer competitive performances when training an artificial neural network. While we do not think that our limited empirical analysis allows any definite conclusion, we hope that it will entice more research in this area.

Acknowledgements I am grateful to J.J. Salazar, the Editor-in-Chief of TOP, for inviting me to write this survey. This work would not have been possible without the numerous co-authors with whom I pursued my investigations in distance geometry, among which I will single out the longest-standing: C. Lavor, N. Maculan, and A. Mucherino. I have first heard of concentration of measure as I passed by D. Malioutov's office at the T.J. Watson IBM Research laboratory: the door was open, the Johnson-Lindenstrauss lemma was mentioned, and I could not refrain from interrupting the conversation and asking for clarification, as I thought that there must surely be a mistake; incredibly, the result was true, and I am grateful to Dr. Malioutov for hosting the conversation I eavesdropped on. I am very thankful to the co-authors who helped me investigate random projections, in particular P.L. Poirion and K. Vu, without whom none of our papers would have been possible. I learned about the existence of the distance instability result thanks to N. Gayraud, who was in the audience during a talk I gave, and suggested it to me as I expressed puzzlement at the poor quality of k-means clusterings. João Fontes Gonçalves, a student in my M.Sc. course, first made the remark following Eq. (6.1.3) ("why are you optimizing a constant?"). I am very grateful to

S. Khalife, D. Gonçalves, and M. Escobar for reading the manuscript and making insightful comments. This research was partly funded by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement n. 764759 ETN "MINOA".

References

Abadi M et al (2015) TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. http://tensorflow.org/

Achlioptas D (2003) Database-friendly random projections: Johnson-Lindenstrauss with binary coins. J Comput Syst Sci 66:671–687

Aggarwal C, Hinneburg A, Keim D (2001) On the surprising behavior of distance metrics in high dimensional space. In: den Bussche JV, Vianu V (eds) Proceedings of ICDT, LNCS, vol 1973. Springer, Berlin, pp 420–434

Ahmadi A, Majumdar A (2019) DSOS and SDSOS optimization: more tractable alternatives to sum of squares and semidefinite optimization. SIAM J Appl Algebra Geom 3(2):193–230

Ahmadi A, Jungers R, Parrilo P, Roozbehani M (2014) Joint spectral radius and path-complete graph Lyapunov functions. SIAM J Control Optim 52(1):687–717

Ailon N, Chazelle B (2006) Approximate nearest neighbors and fast Johnson–Lindenstrauss lemma. In: Proceedings of the symposium on the theory of computing, STOC, vol. '06. ACM, Seattle

Alfakih A, Khandani A, Wolkowicz H (1999) Solving Euclidean distance matrix completion problems via semidefinite programming. Comput Optim Appl 12:13–30

Allen G (2012) Sparse higher-order principal components analysis. In: N. Lawrence, M. Girolami (eds) Proceedings of the international conference on Artificial intelligence and Statistics, vol 22, pp 27–36. PMLR, La Palma

Allen-Zhu Z, Gelashvili R, Micali S, Shavit N (2014) Sparse sign-consistent Johnson-Lindenstrauss matrices: Compression with neuroscience-based constraints. Proc Natl Acad Sci 111(47):16872–16876

Aloise D, Cafieri S, Caporossi G, Hansen P, Perron S, Liberti L (2010) Column generation algorithms for exact modularity maximization in networks. Phys Rev E 82(4):046112

Aloise D, Hansen P, Liberti L (2012) An improved column generation algorithm for minimum sum-of-squares clustering. Math Program A 131:195–220

Aloise D, Caporossi G, Hansen P, Liberti L, Perron S, Ruiz M (2013) Modularity maximization in networks by variable neighbourhood search. In: Bader D, Sanders P, Wagner D (eds) Graph partitioning and graph clustering, contemporary mathematics, vol 588. AMS, Providence, pp 113–127

Amaldi E, Liberti L, Maffioli F, Maculan N (2009) Edge-swapping algorithms for the minimum fundamental cycle basis problem. Math Methods Oper Res 69:205–223

Anderson J (1995) An introduction to neural networks. MIT Press, Cambridge

Arriaga R, Vempala S (2006) An algorithmic theory of learning: Robust concepts and random projection. Mach Learn 63:161–182

Asimow L, Roth B (1978) The rigidity of graphs. Trans AMS 245:279–289

Bahr A, Leonard J, Fallon M (2009) Cooperative localization for autonomous underwater vehicles. Int J Robot Res 28(6):714–728

Barker G, Carlson D (1975) Cones of diagonally dominant matrices. Pac J Math 57(1):15–32

Barvinok A (2002) A course in convexity, No. 54 in graduate studies in mathematics. AMS, Providence

Barvinok A (1995) Problems of distance geometry and convex properties of quadratic maps. Discrete Comput Geom 13:189–202

Barvinok A (1997) Measure concentration in optimization. Math Program 79:33–53

Beeker N, Gaubert S, Glusa C, Liberti L (2013) Is the distance geometry problem in NP? In: Mucherino A., Lavor C., Liberti L., Maculan N. (eds) Distance geometry. Springer, New York, NY, pp 85–94

Belotti P, Lee J, Liberti L, Margot F, Wächter A (2009) Branching and bounds tightening techniques for non-convex MINLP. Optim Methods Softw 24(4):597–634

ben Judah of Worms E (XII-XIII Century) Sodei Razayya

Bengio Y, Lamblin P, Popovici D, Larochelle H (2007) Greedy layer-wise training of deep networks. Advances in neural information processing systems. NIPS, vol 19. MIT Press, Cambridge, pp 153–160

Ben-Tal A, Ghaoui LE, Nemirovski A (2009) Robust optimization. Princeton University Press, Princeton
Beyer K, Goldstein J, Ramakrishnan R, Shaft U (1998) When is "nearest neighbor" meaningful? In: Beeri C, Buneman P (eds) Proceedings of ICDT, LNCS, vol 1540. Springer, Heidelberg, pp 217–235
Bird S, Klein E, Loper E (2009) Natural language processing with Python. O'Reilly, Cambridge
Birge J, Louveaux F (2011) Introduction to stochastic programming. Springer, New York
Blömer J, Lammersen C, Schmidt M, Sohler C (2016) Theoretical analysis of the k-means algorithm: a survey. In: Kliemann L, Sanders P (eds) Algorithm engineering, LNCS, vol 9220. Springer, Cham, pp 81–116
Blumenthal L (1953) Theory and applications of distance geometry. Oxford University Press, Oxford
Böhm C, Jacopini G (1966) Flow diagrams, Turing machines and languages with only two formation rules. Commun ACM 9(5):366–371
Bollobás B (1998) Modern graph theory. Springer, New York
Borg I, Groenen P (2010) Modern multidimensional scaling, 2nd edn. Springer, New York
Bottou L (2012) Stochastic gradient descent tricks. In: Montavon G et al (eds) Neural networks: tricks of the trade, LNCS, vol 7700. Springer, Berlin, pp 421–436
Bourgain J (1985) On Lipschitz embeddings of finite metric spaces in Hilbert space. Isr J Math 52(1–2):46–52
Boutsidis C, Zouzias A, Drineas P (2010) Random projections for k-means clustering. Advances in neural information processing systems. NIPS. NIPS Foundation, La Jolla, pp 298–306
Brambilla A, Premoli A (2001) Rigorous event-driven (red) analysis of large-scale nonlinear rc circuits. IEEE Trans Circ Syst I Fundam Theory Appl 48(8):938–946
Brandes U, Delling D, Gaertler M, Görke R, Hoefer M, Nikoloski Z, Wagner D (2008) On modularity clustering. IEEE Trans Knowl Data Eng 20(2):172–188
Cafieri S, Hansen P, Liberti L (2010) Loops and multiple edges in modularity maximization of networks. Phys Rev E 81(4):46102
Cafieri S, Hansen P, Liberti L (2011) Locally optimal heuristic for modularity maximization of networks. Phys Rev E 83(056105):1–8
Cafieri S, Hansen P, Liberti L (2014) Improving heuristics for network modularity maximization using an exact algorithm. Discrete Appl Math 163:65–72
Cauchy AL (1813) Sur les polygones et les polyèdres. Journal de l'École Polytechnique 16(9):87–99
Cayley A (1841) A theorem in the geometry of position. Camb Math J II:267–271
Chollet F et al (2015) Keras. https://keras.io
Chomsky N (1965) Aspects of the theory of syntax. MIT Press, Cambridge
Choromanska A, Henaff M, Mathieu M, Arous GB, LeCun Y (2015) The loss surfaces of multilayer networks. In: Proceedings of the international conference on artificial intelligence and statistics, AISTATS, vol 18. JMLR, San Diego
COIN-OR (2006) Introduction to IPOPT: a tutorial for downloading, installing, and using IPOPT
Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2461–2505
Connelly R (1978) A counterexample to the rigidity conjecture for polyhedra. Publications Mathématiques de l'IHES 47:333–338
Cox T, Cox M (2001) Multidimensional scaling. Chapman & Hall, Boca Raton
D'Ambrosio C, Liberti L (2017) Distance geometry in linearizable norms. In: Nielsen F, Barbaresco F (eds) Geometric science of information, LNCS, vol 10589. Springer, Berlin, pp 830–838
D'Ambrosio C, Liberti L, Poirion PL, Vu K (2019) Random projections for quadratic programming. Math Program B (in revision)
D'Ambrosio C, Liberti L, Poirion PL, Vu K (2019) Random projections for quadratic programming. Tech. Rep. 2019-7-7322, Optimization Online
Dantzig G (1983) Reminiscences about the origins of linear programming. In: Bachem A, Grötschel M, Korte B (eds) Mathematical programming: the state of the art. Springer, Berlin
Dasgupta S, Gupta A (2002) An elementary proof of a theorem by Johnson and Lindenstrauss. Random Struct Algorithms 22:60–65
D'Aspremont A, Bach F, Ghaoui LE (2014) Approximation bounds for sparse principal component analysis. Math Program B 148:89–110
Dattorro J (2015) Convex optimization and Euclidean distance geometry. $M\epsilon\beta oo$, Palo Alto
Dauphin Y, Pascanu R, Gulcehre C, Cho K, Ganguli S, Bengio Y (2014) Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. Advances in neural information processing systems. NIPS. NIPS Foundation, La Jolla, pp 2933–2941

Demartines P, Hérault J (1997) Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets. IEEE Trans Neural Netw 8(1):148–154
Deo N, Prabhu G, Krishnamoorthy M (1982) Algorithms for generating fundamental cycles in a graph. ACM Trans Math Softw 8(1):26–42
Dey S, Mazumder R, Molinaro M, Wang G (2017) Sparse principal component analysis and its ℓ_1-relaxation. Tech. Rep. arXiv:1712.00800v1
Dias G, Liberti L (2016) Diagonally dominant programming in distance geometry. In: Cerulli R, Fujishige S, Mahjoub R (eds) International symposium in combinatorial optimization, LNCS, vol 9849. Springer, New York, pp 225–236
Douven I (2017) Abduction. In: Zalta E (ed) The Stanford encyclopedia of philosophy. Stanford University, Stanford
Durrant R, Kabán A (2009) When is 'nearest neighbour' meaningful: a converse theorem and implications. J Complex 25:385–397
Eco U (1983) Horns, hooves, insteps. Some hypotheses on three kinds of abduction. In: Eco U, Sebeok T (eds) Dupin, Holmes. Peirce. The Sign of Three. Indiana University Press, Bloomington
Eco U (1984) Semiotics and the philosophy of language. Indiana University Press, Bloomington
Eren T, Goldenberg D, Whiteley W, Yang Y, Morse A, Anderson B, Belhumeur P (2004) Rigidity, computation, and randomization in network localization. IEEE, pp 2673–2684
Euler L (1862) Continuatio fragmentorum ex adversariis mathematicis depromptorum: II Geometria, 97. In: Fuss P, Fuss N (eds) Opera postuma mathematica et physica anno 1844 detecta, vol I. Eggers & C, Petropolis, pp 494–496
Fiedler M (1973) Algebraic connectivity of graphs. Czechoslov Math J 23(2):298–305
Flexer A, Schnitzer D (2015) Choosing ℓ_p norms in high-dimensional spaces based on hub analysis. Neurocomputing 169:281–287
Floreano D (1996) Manuale sulle Reti Neurali Il. Mulino, Bologna
Fortunato S (2010) Community detection in graphs. Phys Rep 486(3–5):75–174
François D, Wertz V, Verleysen M (2007) The concentration of fractional distances. IEEE Trans Knowl Data Eng 19(7):873–886
Friedler F, Huang Y, Fan L (1992) Combinatorial algorithms for process synthesis. Comput Chem Eng 16(1):313–320
Gayraud N (2017) Public remark. Le Monde des Mathématiques Industrielles at INRIA Sophia-Antipolis (MOMI17)
Gilbreth F, Gilbreth L (1921) Process charts: first steps in finding the one best way to do work. In: Proceedings of the annual meeting. American Society of Mechanical Engineers, New York
Gill P (2006) User's guide for SNOPT version 7.2. Systems Optimization Laboratory, Stanford University, California
Gödel K (1986) On the isometric embeddability of quadruples of points of r_3 in the surface of a sphere. In: Feferman S, Dawson J, Kleene S, Moore G, Solovay R, van Heijenoort J (eds) Kurt Gödel: collected works, vol I, pp (1933b) 276–279. Oxford University Press, Oxford
Gonçalves D, Mucherino A, Lavor C, Liberti L (2017) Recent advances on the interval distance geometry problem. J Glob Optim 69:525–545
Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
Haeffele B, Vidal R (2017) Global optimality in neural network training. In: Proceedings of the conference in computer vision and pattern recognition, CVPR. IEEE, Piscataway, pp 4390–4398
Hagberg A, Schult D, Swart P (2008) Exploring network structure, dynamics, and function using `NetworkX`. In: Varoquaux G, Vaught T, Millman J (eds) Proceedings of the 7th Python in science conference (SciPy2008), Pasadena, pp 11–15
Hansen P, Jaumard B (1997) Cluster analysis and mathematical programming. Math Program 79:191–215
Henneberg L (1911) Die Graphische Statik der starren Systeme. Teubner, Leipzig
Heron (50AD) Metrica, vol I. Alexandria
Hinneburg A, Aggarwal C, Keim D (2000) What is the nearest neighbor in high dimensional spaces? In: Proceedings of the conference on very large databases, VLDB, vol 26. Morgan Kaufman, San Francisco, pp. 506–515
Hotelling H (1933) Analysis of a complex of statistical variables into principal components. J Educ Psychol 24(6):417–441
IBM (2017) ILOG CPLEX 12.8 User's Manual. IBM
Indyk P (2001) Algorithmic applications of low-distortion geometric embeddings. Foundations of computer science. FOCS, vol 42. IEEE, Washington, DC, pp 10–33

Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the symposium on the theory of computing, STOC, vol 30. ACM, New York, pp 604–613
Indyk P, Naor A (2007) Nearest neighbor preserving embeddings. ACM Trans Algorithms 3(3), Art. 31
Jain A, Murty M, Flynn P (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323
Johnson W, Lindenstrauss J (1984) Extensions of Lipschitz mappings into a Hilbert space. In: Hedlund G (ed) Conference in modern analysis and probability, contemporary mathematics, vol 26. AMS, Providence, pp 189–206
Jolliffe I (2010) Principal component analysis, 2nd edn. Springer, Berlin
Jordan M (1995) Why the logistic function? A tutorial discussion on probabilities and neural networks. Tech. Rep. Computational Cognitive Science TR 9503, MIT
Kane D, Nelson J (2014) Sparser Johnson–Lindenstrauss transforms. J ACM 61(1):4
Kantor I, Matoušek J, Šámal R (2015) Mathematics++: selected topics beyond the basic courses. No. 75 in Student Mathematical Library. AMS, Providence
Khalife S, Liberti L, Vazirgiannis M (2019) Geometry and analogies: a study and propagation method for word representation. In: Statistical language and speech processing, SLSP, vol. 7
Kingma D, Ba J (2015)Adam: A method for stochastic optimization. In: Proceedings of ICLR. San Diego
Knuth D (1997) The art of computer programming, part I: fundamental algorithms, 3rd edn. Addison-Wesley, Reading
Kullback S, Leibler R (1951) On information and sufficiency. Ann Math Stat 22(1):79–86
Kuratowski C (1935) Quelques problèmes concernant les espaces métriques non-séparables. Fundam Math 25:534–545
Lavor C, Liberti L, Maculan N (2006) Computational experience with the molecular distance geometry problem. In: Pintér J (ed) Global optimization: scientific and engineering case studies. Springer, Berlin, pp 213–225
Lavor C, Liberti L, Maculan N, Mucherino A (2012) The discretizable molecular distance geometry problem. Comput Optim Appl 52:115–146
Lavor C, Liberti L, Mucherino A (2013) The interval Branch-and-Prune algorithm for the discretizable molecular distance geometry problem with inexact distances. J Glob Optim 56:855–871
Lavor C, Liberti L, Donald B, Worley B, Bardiaux B, Malliavin T, Nilges M (2019) Minimal NMR distance information for rigidity of protein graphs. Discrete Appl Math 256:91–104
Lavor C, Souza M, Carvalho L, Liberti L (2019) On the polynomiality of finding ^{K}DMDGP re-orders. Discrete Appl Math 267:190–194
Lehmann S, Hansen L (2007) Deterministic modularity optimization. Eur Phys J B 60:83–88
Levine R, Mason T, Brown D (1995) Lex and Yacc, 2nd edn. O'Reilly, Cambridge
Liberti L (2010) Software modelling and architecture: exercises. Ecole Polytechnique. https://www.lix.polytechnique.fr/~liberti/swarchex.pdf
Liberti L (2009) Reformulations in mathematical programming: definitions and systematics. RAIRO-RO 43(1):55–86
Liberti L (2019) Undecidability and hardness in mixed-integer nonlinear programming. RAIRO-Oper Res 53:81–109
Liberti L, Lavor C (2013) On a relationship between graph realizability and distance matrix completion. In: Migdalas A, Sifaleras A, Georgiadis C, Papathanaiou J, Stiakakis E (eds) Optimization theory, decision making, and operational research applications, proceedings in mathematics & statistics, vol 31. Springer, Berlin, pp 39–48
Liberti L, Lavor C (2016) Six mathematical gems in the history of distance geometry. Int Trans Oper Res 23:897–920
Liberti L, Lavor C (2017) Euclidean distance geometry: an introduction. Springer, New York
Liberti L, Marinelli F (2014) Mathematical programming: Turing completeness and applications to software analysis. J Comb Optim 28(1):82–104
Liberti L, Vu K (2018) Barvinok's naive algorithm in distance geometry. Oper Res Lett 46:476–481
Liberti L, Lavor C, Maculan N (2008) A branch-and-prune algorithm for the molecular distance geometry problem. Int Trans Oper Res 15:1–17
Liberti L, Cafieri S, Tarissan F (2009) Reformulations in mathematical programming: a computational approach. In: Abraham A, Hassanien AE, Siarry P, Engelbrecht A (eds) Foundations of computational intelligence, vol 3. no 203 in Studies in Computational Intelligence. Springer, Berlin, pp 153–234

Liberti L, Cafieri S, Savourey D (2010) Reformulation optimization software engine. In: Fukuda K, van der Hoeven J, Joswig M, Takayama N (eds) Mathematical software, LNCS, vol 6327. Springer, New York, pp 303–314
Liberti L, Lavor C, Mucherino A, Maculan N (2010) Molecular distance geometry methods: from continuous to discrete. Int Trans Oper Res 18:33–51
Liberti L, Lavor C, Alencar J, Abud G (2013) Counting the number of solutions of ^{k}DMDGP instances. In: Nielsen F, Barbaresco F (eds) Geometric science of information, LNCS, vol 8085. Springer, New York, pp 224–230
Liberti L, Lavor C, Maculan N, Mucherino A (2014) Euclidean distance geometry and applications. SIAM Rev 56(1):3–69
Liberti L, Masson B, Lavor C, Lee J, Mucherino A (2014) On the number of realizations of certain Henneberg graphs arising in protein conformation. Discrete Appl Math 165:213–232
Liberti L, Swirszcz G, Lavor C (2016) Distance geometry on the sphere. In: Akiyama J et al (eds) JCDCG^2, LNCS, vol 9943. Springer, New York, pp 204–215
Liberti L, D'Ambrosio C (2017) The Isomap algorithm in distance geometry. In: Iliopoulos C, Pissis S, Puglisi S, Raman R (eds) Proceedings of 16th international symposium on experimental algorithms (SEA), LIPICS, vol 75. Dagstuhl Publishing, Schloss Dagstuhl, pp 5:1–5:13
Liberti L, Lavor C, Mucherino A (2013) The discretizable molecular distance geometry problem seems easier on proteins. In: Mucherino A, Lavor C, Liberti L, Maculan N (eds) Distance geometry: theory, methods and applications. Springer, New York, pp 47–60
Linial N, London E, Rabinovich Y (1995) The geometry of graphs and some of its algorithmic applications. Combinatorica 15(2):215–245
Majumdar A, Ahmadi A, Tedrake R (2014) Control and verification of high-dimensional systems with dsos and sdsos programming. Conference on decision and control, vol 53. Piscataway, IEEE, pp 394–401
Malliavin T, Mucherino A, Lavor C, Liberti L (2019) Systematic exploration of protein conformational space using a distance geometry approach. J Chem Inf Model 59:4486–4503
Manning C, Schütze H (1999) Foundations of statistical natural language processing. MIT Press, Cambridge
Mansouri J, Khademi M (2015) Multiplicative distance: a method to alleviate distance instability for high-dimensional data. Knowl Inf Syst 45:783–805
Matoušek J (2013) Lecture notes on metric embeddings. Tech. rep, ETH Zürich
Matoušek J (2008) On variants of the Johnson-Lindenstrauss lemma. Random Struct Algorithms 33:142–156
Maxwell J (1864) On the calculation of the equilibrium and stiffness of frames. Philos Mag 27(182):294–299
McCormick G (1976) Computability of global solutions to factorable nonconvex programs: Part I-Convex underestimating problems. Math Program 10:146–175
McCulloch W (1961) What is a number, that a man may know it, and a man, that he may know a number? Gen Semant Bull 26–27:7–18
Mencarelli L, Sahraoui Y, Liberti L (2017) A multiplicative weights update algorithm for MINLP. EURO J Comput Optim 5:31–86
Menger K (1928) Untersuchungen über allgemeine Metrik. Math Ann 100:75–163
Menger K (1931) New foundation of Euclidean geometry. Am J Math 53(4):721–745
Merris R (1994) Laplacian matrices of graphs: a survey. Linear Algebra Appl 198:143–176
Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Burges C, Bottou L, Welling M, Ghahramani Z, Weinberger K (eds) Advances in neural information processing systems, NIPS, vol 26. NIPS Foundation, La Jolla, pp 3111–3119
Miller G (1995) Wordnet: a lexical database for English. Commun ACM 38(11):39–41
Milnor J (1964) On the Betti numbers of real varieties. Proc AMS 15:275–280
Minsky M (1986) The society of mind. Simon & Schuster, New York
Moitra A (2018) Algorithmic aspects of machine learning. CUP, Cambridge
Moro A (2008) The boundaries of Babel. MIT Press, Cambridge
Morris C (1946) Signs. Language and behavior. Prentice-Hall, New York
Mucherino A, Lavor C, Liberti L (2012) Exploiting symmetry properties of the discretizable molecular distance geometry problem. J Bioinform Comput Biol 10(1–15):1242009

Mucherino A, Lavor C, Liberti L, Maculan N (eds) (2013) Distance geometry: theory, methods, and applications. Springer, New York
Newman M, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69:026113
Object Management Group (2005) Unified modelling language: superstructure, v. 2.0. Tech. Rep. formal/05-07-04, OMG
O'Donoghue B, Chu E, Parikh N, Boyd S (2016) Operator splitting for conic optimization via homogeneous self-dual embedding. J Optim Theory Appl 169(3):1042–1068
Paton K (1969) An algorithm for finding a fundamental set of cycles of a graph. Commun ACM 12(9):514–518
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
Peirce C (1878) Illustrations of the logic of science, part 6: induction, deduction, and hypothesis. Popul Sci Mon 13:470–482
Penrose R (1989) The emperor's new mind. Penguin, New York
Pfeffer A (2016) Practical probabilistic programming. Manning Publications, Shelter Island
Popper K (1968) The logic of scientific discovery. Hutchinson, London
Potra F, Wright S (2000) Interior-point methods. J Comput Appl Math 124:281–302
Proni G (2016) Is there abduction in Aristotle? Peirce, Eco, and some further remarks. Ocula 17:1–14
Radovanović M, Nanopoulos A, Ivanović M (2010) Hubs in space: Popular nearest neighbors in high-dimensional data. J Mach Learn Res 11:2487–2531
Rousseau F, Vazirgiannis M (2013) Graph-of-word and TW-IDF: new approach to ad hoc IR. In: Proceedings of CIKM. ACM, New York
Saerens M, Fouss F, Yen L, Dupont P (2004) The principal components analysis of a graph, and its relationships to spectral clustering. In: Boulicaut JF, Esposito F, Giannotti F, Pedreschi D (eds) Proceedings of the European conference in machine learning (ECML), LNAI, vol 3201. Springer, Berlin, pp 371–383
Salgado E, Scozzari A, Tardella F, Liberti L (2018) Alternating current optimal power flow with generator selection. In: Lee J, Rinaldi G, Mahjoub R (eds) Combinatorial optimization (Proceedings of ISCO 2018), LNCS, vol 10856, pp 364–375
Sánchez AB, Lavor C (2020) On the estimation of unknown distances for a class of Euclidean distance matrix completion problems with interval data. Linear Algebra Appl 592:287–305
Saxe J (1979) Embeddability of weighted graphs in k-space is strongly NP-hard. In: Proceedings of 17th Allerton conference in communications, control and computing, pp 480–489
Schaeffer S (2007) Graph clustering. Comput Sci Rev 1:27–64
Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003. arXiv:1404.7828 [cs.NE]
Schoenberg I (1935) Remarks to Maurice Fréchet's article Sur la définition axiomatique d'une classe d'espaces distanciés vectoriellement applicable sur l'espace de Hilbert. Ann Math 36(3):724–732
Schumacher M, Roßner R, Vach W (1996) Neural networks and logistic regression: part I. Comput Stat Data Anal 21:661–682
Seshu S, Reed M (1961) Linear graphs and electrical networks. Addison-Wesley, Reading
Singer A (2011) Angular synchronization by eigenvectors and semidefinite programming. Appl Comput Harmon Anal 30:20–36
Smith E, Pantelides C (1999) A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. Comput Chem Eng 23:457–478
Steinhaus H (1956) Sur la division des corps matériels en parties. Bulletin de l'Académie Polonaise des Sciences Cl. III 4(12):801–804
Tabaghi P, Dokmanić I, Vetterli M (2019) On the move: localization with kinetic Euclidean distance matrices. In: International conference on acoustics, speech and signal processing (ICASSP). IEEE, Piscataway
Tawarmalani M, Sahinidis N (2004) Global optimization of mixed integer nonlinear programs: a theoretical and computational study. Math Program 99:563–591
Tenenbaum J, de Silva V, Langford J (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290:2319–2322
Thoreau H (1849) Resistance to civil government. In: Peabody E (ed) Æsthetic papers. J. Wilson, Boston
van Rossum G et al (2019) Python language reference, version 3. Python Software Foundation

Vavasis S (1991) Nonlinear optimization: complexity issues. Oxford University Press, Oxford
Vempala S (2004) The Random projection method. No. 65 in DIMACS series in discrete mathematics and theoretical computer science. AMS, Providence
Venkatasubramanian S, Wang Q (2011) The Johnson–Lindenstrauss transform: an empirical study. Algorithm engineering and experiments. ALENEX, vol 13. SIAM, Providence, pp 164–173
Verboon A (2014) The medieval tree of Porphyry: an organic structure of logic. In: Worm A, Salonis P (eds) The Tree. Symbol, allegory and structural device in medieval art and thought, international medieval research, vol 20. Brepols, Turnhout, pp 83–101
Vershynin R (2018) High-dimensional probability. CUP, Cambridge
Vidal R, Ma Y, Sastry S (2016) Generalized principal component analysis. Springer, New York
Vu K, Poirion PL, Liberti L (2018) Random projections for linear programming. Math Oper Res 43(4):1051–1071
Vu K, Poirion PL, D'Ambrosio C, Liberti L (2019) Random projections for quadratic programs over a Euclidean ball. In: Lodi A et al (eds) Integer programming and combinatorial optimization (IPCO), LNCS, vol 11480. Springer, New York, pp 442–452
Vu K, Poirion PL, Liberti L (2019) Gaussian random projections for Euclidean membership problems. Discrete Appl Math 253:93–102
Wikipedia: Civil disobedience (thoreau) (2019). http://en.wikipedia.org/wiki/Civil_Disobedience_(Thoreau). [Online; accessed 190804]
Wikipedia: Computational pragmatics (2019). http://en.wikipedia.org/wiki/Computational_pragmatics. [Online; accessed 190802]
Wikipedia: Diagonally dominant matrix (2019). http://en.wikipedia.org/wiki/Diagonally_dominant_matrix. [Online; accessed 190716]
Wikipedia: Flowchart (2019). http://en.wikipedia.org/wiki/Flochart. [Online; accessed 190802]
Wikipedia: Principal component analysis (2019). http://en.wikipedia.org/wiki/Principal_component_analysis. [Online; accessed 190726]
Wikipedia: Rectifier (neurl networks) (2019). http://en.wikipedia.org/wiki/Rectifier_(neural_networks). [Online; accessed 190807]
Wikipedia: Slutsky's theorem (2019). http://en.wikipedia.org/wiki/Slutsky%27s_theorem. [Online; accessed 190802]
Williams H (1999) Model building in mathematical programming, 4th edn. Wiley, Chichester
Woodruff D (2014) Sketching as a tool for linear algebra. Found Trends Theor Comput Sci 10(1–2):1–157
Wüthrich K (1989) Protein structure determination in solution by nuclear magnetic resonance spectroscopy. Science 243:45–50
Xu G, Tsoka S, Papageorgiou L (2007) Finding community structures in complex networks using mixed integer optimisation. Eur Phys J B 60:231–239
Yemini Y (1978) The positioning problem—a draft of an intermediate summary. In: Proceedings of the conference on distributed sensor networks. Carnegie-Mellon University, Pittsburgh, pp 137–145
Yemini Y (1979) Some theoretical aspects of position-location problems. In: Proceedings of the 20th annual symposium on the foundations of computer science, pp. 1–8. IEEE, Piscataway
Yun C, Sra S, Jadbabaie A (2018) Global optimality conditions for deep neural networks. In: Proceedings of the 6th international conference on learning representations. ICLR, La Jolla, CA
Zhang L, Mahdavi M, Jin R, Yang T, Zhu S (2013) Recovering the optimal solution by dual random projection. In: Shalev-Shwartz S, Steinwart I (eds) Conference on learning theory (COLT), Proceedings of machine learning research, vol 30, pp 135–157. ⟨http://mlr.org⟩