Cycle-based formulations in Distance Geometry

Leo Liberti[1], G. Iommazzo[1,2], C. Lavor[3], N. Maculan[4]

1. *CNRS LIX, École Polytechnique, F-91128 Palaiseau, France*
   Email:liberti@lix.polytechnique.fr
2. *DI, Università di Pisa, Italy*
   Email:giommazz@lix.polytechnique.fr
3. *IMECC, University of Campinas, Brazil*
   Email:clavor@ime.unicamp.br
4. *COPPE, Federal Univ. Rio de Janeiro (UFRJ), Brazil*
   Email:maculan@cos.ufrj.br

June 20, 2020

Abstract

The distance geometry problem asks to find a realization of a given simple edge-weighted graph in a Euclidean space of given dimension K, where the edges are realized as straight segments of lengths equal (or as close as possible) to the edge weights. The problem is often modelled as a mathematical programming formulation involving decision variables that determine the position of the vertices in the given Euclidean space. Solution algorithms are generally constructed using local or global nonlinear optimization techniques. We present a new modelling technique for this problem where, instead of deciding vertex positions, formulations decide the length of the segments representing the edges in each cycle in the graph, projected in every dimension. We propose an exact formulation and a relaxation based on a Eulerian cycle. We then compare computational results from protein conformation instances obtained with stochastic global optimization techniques on the new cycle-based formulation and on the existing edge-based formulation. While edge-based formulations take less time to reach termination, cycle-based formulations are generally better on solution quality measures.

1 Introduction

We consider the fundamental problem in Distance Geometry (DG):

> Distance Geometry Problem (DGP). Given a positive integer K and a simple undirected graph $G = (V, E)$ with an edge weight function $d : E \to \mathbb{R}_{\geq 0}$, establish whether there exists a *realization* $x : V \to \mathbb{R}^K$ of the vertices such that Eq. (1) below is satisfied:
>
> $$\forall \{i, j\} \in E \qquad \|x_i - x_j\| = d_{ij}, \tag{1}$$
>
> where $x_i \in \mathbb{R}^K$ for each $i \in V$ and d_{ij} is the weight on edge $\{i, j\} \in E$.

Although the DGP is given above in the canonical decision form, we consider the corresponding search problem, where one has to actually find the realization x. The DGP is also known as the *graph realization problem* in geometric rigidity [25, 6, 17]. It belongs to a more general class of metric completion and embedding problems [7, 21, 51].

In its most general form, the DGP might be parametrized over any norm [11]. In practice, the ℓ_2 norm is the most usual choice [40], and will also be employed in this paper. The DGP with the ℓ_2 norm is sometimes called the EUCLIDEAN DGP (EDGP). For the EDGP, Eq. (1) is often reformulated to:

$$\forall \{i,j\} \in E \qquad \|x_i - x_j\|_2^2 = d_{ij}^2, \tag{2}$$

which is a system of quadratic polynomial equations with no linear terms [35, §2.4].

The EDGP is motivated by many scientific and technological applications. The clock synchronization problem, for example, aims at establishing the absolute time of a set of clocks when only the time difference between subsets of clocks can be exchanged [53]. The sensor network localization problem aims at finding the positions of moving wireless sensor on a 2D manifold given an estimation of some of the pairwise Euclidean distances [17, 2, 15]. The MOLECULAR DGP (MDGP) aims at finding the positions of atoms in a protein, given some of the pairwise Euclidean distances [27, 29, 40, 35, 8, 46]. The position of autonomous underwater vehicles cannot be determined via GPS (since the GPS signal does not reach under water), but must rely on distances estimated using sonars: a DGP can then be solved in order to localize the fleet [3]. Applications of the DGP to data science are described in [32]; see [31] for an application to natural language processing. In general, the DGP is an inverse problem which occurs every time one can measure some of the pairwise distances in a set of entities, and needs to establish their position.

The DGP is weakly **NP**-hard even when restricted to simple cycle graphs (by reduction from PARTITION) and strongly **NP**-hard even when restricted to integer edge weights in $\{1, 2\}$ in general graphs (by reduction from 3SAT) [50]. It is in **NP** if $K = 1$ but not known to be in **NP** if $K > 1$ for general graphs [4], which is an interesting open question [36].

There are many approaches to solving the DGP. Generally speaking, application-specific solution algorithms exploit some of the graph structure, whenever it is induced by the application. For example, a condition often asked when reconstructing the positions of sensor networks is that the realization should be unique (as one would not know how to choose between multiple realizations), a condition called *global rigidity* [10]. This condition can, at least generically, be ensured by a specific graph rigidity structure of the unweighted input graph. For protein structures, on the other hand, which are found in nature in several isomers, one is sometimes interested in finding all (incongruent) realizations of the given protein graph [28, 48, 37]. Since such graphs are rigid, one can devise an algorithm (called Branch-and-Prune) which, following a given vertex order, branches on reflections of the position of the next vertex, which is computed using trilateration [38, 35]. In absence of any information on the graph structure, however, one can resort to Mathematical Programming (MP) formulations and corresponding solvers [41, 12, 14].

The MP formulation which is most often used reformulates Eq. (2) to the minimization of the sum of squared error terms:

$$\min_x \sum_{\{i,j\} \in E} (\|x_i - x_j\|_2^2 - d_{ij}^2)^2. \tag{3}$$

This formulation describes an unconstrained polynomial minimization problem. The polynomial in question has degree 4, is always nonnegative, and generally nonconvex and multimodal. The decision variables are represented by a $n \times K$ rectangular matrix x such that x_{ik} is the k-th component of the vector x_i, which gives the position in $\mathbb{R}^K$ of vertex $i \in V$. Each solution $x^* \in \mathbb{R}^{nK}$ having global minimum value equal to zero is a realization of the given graph. Solutions with small objective function value represent approximate solutions. Because of the nonconvexity of the formulation and the hardness of the problem, Eq. (3) is not usually solved to guaranteed ε-optimality (e.g. using a spatial Branch-and-Bound approach [5]); rather, heuristic approaches, such as MultiStart (MS) [34, 26], Variable Neighbourhood Search (VNS) [39], or relaxation-based heuristics [14, 43] may be used.

As far as we know, all existing MP formulations for the EDGP are based on the incidence of edges and vertices. In this paper we discuss a new MP formulation for the EDGP based on the incidence of cycles and edges instead, a relaxation based on Eulerian cycles, and a computational comparison with Eq. (3).

2 Some existing MP formulations

In this short section we give a minimal list of typical variants of Eq. (3) in order to motivate the claim that the cycle-based formulation of the DGP discussed in this paper is new. Of course, only a complete enumeration of DGP formulations in the literature could substantiate this claim. But even this short list shows that the typical modelling approach for the DGP is direct: namely, decision variables encode the realization of each vertex as a vector in $\mathbb{R}^K$. Many more formulations of the DGP and its variants, all corresponding to this criterion, are given in [26, 41, 12].

The closest variant of Eq. (3) simply adds a constraint ensuring that the centroid of all of the points in the realization is at the origin. This removes the degrees of freedom given by translations:

$$\left.\begin{array}{rl} \min\limits_{x} & \sum\limits_{\{i,j\}\in E} (\|x_i - x_j\|_2^2 - d_{ij}^2)^2 \\ \forall k \leq K & \sum\limits_{i\in V} x_{ik} = 0. \end{array}\right\} \tag{4}$$

This formulation describes a linearly constrained polynomial minimization problem. Like Eq. (3), the polynomial in Eq. (4) has degree 4, is always nonnegative, and is generally nonconvex and multimodal.

Another small variant of Eq. (4) is achieved by adding range bounds to the the realization variables x; generally valid (but slack) bound values can be set to $\pm\frac{1}{2}\sum_{\{u,v\}\in E} d_{uv}$. This corresponds to the worst case of a single path being arranged in a straight line with unknown orientation.

Another possible formulation, derived again from Eq. (3), is obtained by replacing the squared error with absolute value errors (whose positive and negative parts are encoded by s^+, s^-). This yields the following formulation:

$$\left.\begin{array}{rl} \min\limits_{s,x} & \sum\limits_{\{i,j\}\in E} (s_{ij}^+ + s_{ij}^-) \\ \forall \{i,j\} \in E & \|x_i - x_j\|_2^2 = d_{ij}^2 + s_{ij}^+ - s_{ij}^- \\ \forall \{i,j\} \in E & s_{ij}^+, s_{ij}^- \geq 0. \end{array}\right\} \tag{5}$$

Note that, again, each solution s^*, x^* with zero optimal objective value makes x^* an encoding of a realization of the given graph. Thus, global optima are preserved by this reformulation, while local optimal may differ.

Yet another reformulation derived from replacing squared errors with absolute values consists in observing that the "plus" and "minus" parts of each absolute value term correspond to a convex and concave function. This yields a formulation called *push-and-pull*, since the objective pulls adjacent vertices apart, while the constraint push them back together:

$$\left.\begin{array}{rcl} \max\limits_{x} & \sum\limits_{\{i,j\}\in E} \|x_i - x_j\|_2^2 & \\ \forall \{i,j\} \in E & \|x_i - x_j\|_2^2 & \leq \quad d_{ij}^2. \end{array}\right\} \tag{6}$$

Eq. (6) is a Quadratically Constrained Quadratic Program with concave objective and convex constraints. It was used within a Multiplicative Weights Update algorithm for the DGP in [12, 47], as well as a basis for Semidefinite Programming and Diagonally Dominant Programming relaxations [14, 43]. It can be shown that all constraints are active at global optima, which therefore correspond to realizations of the given graph [47].

3 A new formulation based on cycles

In this section we propose a new formulation for the EDGP, based on the fact that the quantities $x_{ik} - x_{jk}$ sum up to zero over all edges of any cycle in the given graph for each dimensional index $k \leq K$. This idea was used in [50] for proving weak **NP**-hardness of the DGP on cycle graphs. For a subgraph H of

a graph $G = (V, E)$, we use $V(H)$ and $E(H)$ to denote vertex and edge set of H explicitly; given a set F of edges we use $V(F)$ to denote the set of incident vertices. Let $m = |E|$ and $n = |V|$. For a mapping $x : V \to \mathbb{R}^K$ we denote by $x[U]$ the restriction of x to a subset $U \subseteq V$.

3.1 Lemma
Given an integer $K > 0$, a simple undirected weighted graph $G = (V, E, d)$ and a mapping $x : V \to \mathbb{R}^K$, then for each cycle C in G, each orientation of the edges in C given by a closed trail $W(C)$ in the cycle, and each $k \leq K$ we have:

$$\sum_{(i,j)\in W(C)} (x_{ik} - x_{jk}) = 0. \tag{7}$$

Proof. We renumber the vertices in $V(C)$ to $1, 2, \ldots, \gamma = |V(C)|$ following the walk order in $W(C)$. Then Eq. (7) can be explicitly written as:

$$\begin{aligned} (x_{1k} - x_{2k}) + (x_{2k} - x_{3k}) + \cdots + (x_{\gamma k} - x_{1k}) &= \\ = x_{1k} - (x_{2k} - x_{2k}) - \cdots - (x_{\gamma k} - x_{\gamma k}) - x_{1k} &= 0, \end{aligned}$$

as claimed. □ □

We introduce new decision variables y_{ijk} replacing the terms $x_{ik} - x_{jk}$ for each $\{i, j\} \in E$ and $k \leq K$. Eq. (2) then becomes:

$$\forall \{i, j\} \in E \qquad \sum_{k\leq K} y_{ijk}^2 = d_{ij}^2. \tag{8}$$

We remark that for the DGP with other norms this constraint changes. For the ℓ_1 or ℓ_∞ norms, for example, we would have:

$$\forall \{i, j\} \in E \quad \sum_{k\leq K} |y_{ijk}| = d_{ij} \quad \text{or} \quad \max_{k\leq K} |y_{ijk}| = d_{ij}. \tag{9}$$

Next, we adjoin the constraints on cycles:

$$\forall k \leq K, C \subset G \quad \Big(C \text{ is a cycle} \Rightarrow \sum_{\{i,j\}\in E(C)} y_{ijk} = 0\Big). \tag{10}$$

We also note that the feasible value of a y_{ijk} variable is the (oriented) length of the segment representing the edge $\{i, j\}$ projected on the k-th coordinate. We can therefore infer bounds for y as follows:

$$\forall k \leq K, \{i, j\} \in E \quad -d_{ij} \leq y_{ijk} \leq d_{ij}. \tag{11}$$

Although Eq. (11) are not necessary to solve the cycle formulation, they may improve performance of spatial Branch-and-Bound (sBB) algorithms [54, 5] as well as of various "matheuristics" [42], as well as allow an exact linearization of variable products, should a y variable occur in a product with a binary variable in some DGP variant.

We now state our main result, i.e. that Eq. (8) and (10) are a valid MP formulation for the EDGP.

3.2 Theorem
There exists a vector $y^* \in \mathbb{R}^{Km}$ which satisfies Eq. (8) and (10), parametrized on K, G, if and only if (K, G) is a YES instance of the EDGP.

The proof argues by recursion on a graph decomposition of G that a certain linear system related to the cycles of G (see Eq. (12) below) has a solution if and only if the given DGP instance is YES. We shall construct the proof by steps. The first step defines the graph decomposition.

Given a graph $G = (V, E)$ and a subset $U \subset V$, the subgraph $G[U]$ *induced* by U is the graph $(U, \{\{u, v\} \in E \mid u, v \in U\})$. With a slight abuse of notation we denote the vertices of a graph G' by $V(G')$ and its edges by $E(G')$. We let $\gamma(G)$ be the number of connected components of G. A vertex v of G with the property that $\gamma(G[V \setminus \{v\}]) > \gamma(G)$ is called a *cut vertex*. A graph G is *biconnected* if there is a simple cycle in G incident to any pair of distinct vertices of G.

3.3 Lemma
$G = (V, E)$ is biconnected if and only if it is connected and has no cut vertices.

Proof. Suppose G is biconnected with a cut vertex v: then the removal of v from G yields two separate connected components G_1, G_2. Let $u_1 \in V(G_1)$ and $u_2 \in V(G_2)$. Since $u_1, u_2 \in V$ and G is biconnected, there is a simple cycle in G incident to u_1, u_2, consisting of two vertex-disjoint simple paths p_1 and p_2 from u_1 to u_2. Since the removal of v can break at most one of these paths (by vertex disjointness), the other path shows that G_1, G_2 are not disconnected, against the assumption. So G cannot have cut vertices. Conversely, if G is connected and has a cut vertex v, then any pair of paths from u_1 to u_2 must necessarily pass through v, which means that they are not vertex disjoint, which implies that there is no cycle between u_1 and u_2 in G, which in turn implies that G is not biconnected. □ □

We now define a graph decomposition based on removal of a single cut vertex.

3.4 Definition
A 1-decomposition of a graph $G = (V, E)$ is a set of subgraphs $G_1, \ldots, G_r$ (where $r \in \mathbb{N}$ with $r \geq 1$) of G such that:

(a) G_i is either biconnected or a tree for all $i \leq r$;

(b) $\bigcup_{i \leq r} E(G_i) = E$;

(c) for any $i < j \leq r$ the intersection $V(G_i) \cap V(G_j)$ either has zero cardinality or it consists of a single cut vertex of G.

A 1-decomposition of G is nontrivial if $r > 1$. A graph G is 1-decomposable if it has a nontrivial 1-decomposition.

3.5 Lemma
A connected graph $G = (V, E)$ is 1-decomposable if and only if it has a cut vertex.

Proof. Suppose $\mathcal{C} = \{C_1, \ldots, C_t\}$ is a nontrivial 1-decomposition of G. By Defn. 3.4 and since G is connected, it follows that G must have at least one cut vertex and as many as $|\mathcal{C}| - 1$. Conversely, supposing that G has a cut vertex would yield a nontrivial 1-decomposition by Defn. 3.4, i.e. G is 1-decomposable. □ □

3.6 Corollary
No simple graph consisting of a single cycle is 1-decomposable.

Proof. Since a cycle is biconnected, by Lemma 3.3 it cannot have a cut vertex, hence its only possible 1-decomposition is trivial. □ □

3.7 Corollary
Let G be 1-decomposable, with decomposition $\mathcal{G} = \{G_1, \ldots, G_r\}$, and C be a cycle in G. Then there is an index $i \leq r$ s.t. C is a subgraph of G_i.

Proof. Consider there were two subgraphs G_i, G_j in $\mathcal{G}$ both incident to the edges of C. Then there is a nontrivial path p in C, with at least two edges, joining a vertex u in G_i to a vertex v in G_j. Therefore there must be a cut vertex of G on p, which implies that there is a cut vertex in C, which is impossible by Cor. 3.6. □ □

3.8 Corollary
No biconnected graph G is 1-decomposable.

Proof. By Lemma 3.3, if G is biconnected it cannot have a cut vertex, therefore any 1-decomposition must necessarily be trivial. □ □

3.9 Proposition
Any simple graph $G = (V, E)$ has a 1-decomposition consisting of biconnected subgraphs and tree subgraphs.

Proof. We prove this result by induction on the number β of biconnected subgraphs in a 1-decomposition $\mathcal{C} = \{G_1, \ldots, G_r\}$ of G for some $r \in \mathbb{N}$. We first deal with the base case, where $\beta = 0$. We claim that G must be a tree: supposing G has a cycle G', by Cor. 3.6 and part (c) of Defn. 3.4, G' must be one of the $G_1, \ldots, G_r$. But then $\beta = 1$ against the assumption. Therefore, the trivial 1-decomposition $\mathcal{C} = \{G\}$ is a valid 1-decomposition of G. We now tackle the induction step. Consider the largest biconnected subgraph B of G: then $\tilde{G} = G[V \setminus V(B)]$ has one fewer biconnected components than G, so, by induction, $\tilde{G}$ has a 1-decomposition $\mathcal{D}' = \{G'_1, \ldots, G'_{t-1}\}$ for some $t \in \mathbb{N}$ with $t > 1$. We prove that $\mathcal{D} = \mathcal{D}' \cup \{B\}$ is a valid 1-decomposition of G. Condition (a) is verified since $\mathcal{D}'$ is a valid 1-decomposition by induction, and B is biconnected; condition (b) is verified since the union of the graph in $\mathcal{D}$ is $\mathcal{G}$ by construction; for condition (c), suppose there is $i < t$ s.t. $|V(G_i) \cap V(B)| \geq 2$: this means there are two distinct vertices u, v in both $V(G_i)$ and $V(B)$. Since G_i is connected, there must be a path p from u to v in G_i, hence $G[B \cup V(p)]$ is a biconnected graph larger than B. But B was assumed to be largest, so this is not possible, and (c) holds, which concludes the proof. □ □

The second step proves the easier ($\Leftarrow$) direction of Thm. 3.2.

3.10 Proposition
For any YES instance (K, G) of the EDGP there is a vector $y^* \in \mathbb{R}^{Km}$ which satisfies Eq. (8) and (10).

Proof. Assume that (K, G) is a YES instance of the EDGP. Then G has a realization $x^* \in \mathbb{R}^{nK}$ in $\mathbb{R}^K$. We define $y^*_{ijk} = x^*_{ik} - x^*_{jk}$ for all $\{i, j\} \in E$ and $k \leq K$. Since x^* is a realization of G, by definition it satisfies Eq. (2), and, by substitution, Eq. (8). Moreover, any realization of G satisfies Eq. (7) over each cycle by Lemma 3.1. Hence, by replacement, it also satisfies Eq. (10). □ □

In the third step, we lay the groundwork towards the more difficult ($\Rightarrow$) direction of Thm. 3.2. We proceed by contradiction: we assume that (K, G) is a NO instance of the EDGP, and suppose that Eq. (8) and (10) have a non-empty feasible set Y. For every $y \in Y$ we consider the K linear systems

$$\forall \{i, j\} \in E \quad x_{ik} - x_{jk} = y_{ijk}, \tag{12}$$

for each $k \leq K$, each with n variables and m equations. We square both sides then sum over $k \leq K$ to obtain

$$\forall \{i, j\} \in E \quad \sum_{k \leq K} (x_{ik} - x_{jk})^2 = \sum_{k \leq K} y^2_{ijk}. \tag{13}$$

By Eq. (8) we have

$$\sum_{k \leq K} y^2_{ijk} = d^2_{ij}, \tag{14}$$

whence follows Eq. (2), contradicting the assumption that the EDGP is NO. So we need only show that there is a solution x to Eq. (12) for any given $y \in Y$. To this effect, we shall exploit the 1-decomposition of G into biconnected graphs and trees derived in Prop. 3.9. First, though, we have to show that Eq. (12) has a solution if $Y \neq \varnothing$ in the "base cases" of the 1-decomposition, namely trees and biconnected graphs.

3.11 Lemma
Let $G = (V, E)$ be a tree, and $Y \neq \varnothing$ satisfying Eq. (8) and (10). Then Eq. (12) has a solution for every $k \leq K$.

Proof. Let M^k be the matrix of each system Eq. (12), for $k \leq K$; we aim at proving that M^k and (M^k, y^k) have the same rank, where $y^k = (y_{uvk} \mid \{u, v\} \in E)$, and that this rank is full. We proceed by induction on the size $|E|$ of the tree. The base case, where $|E| = 1$ and G consists of a single edge $\{u, v\}$, yields $M^k = (1, -1)$ with rank 1 for each $k \leq K$. By inspection, (M^k, y_{uvk}) also has rank 1 for any y_{uvk}. Consider a tree G' with one fewer edge (say, $\{u, v\}$) than G, such that $V \smallsetminus V(G') = \{v\}$. Let the corresponding system Eq. (12) $\tilde{M}^k = \tilde{y}^k$ satisfy $\mathsf{rank}(\tilde{M}^k) = \mathsf{rank}(\tilde{M}^k, \tilde{y}^k)$, for all $k \leq K$. Then the shape of M^k is:

$$M^k = \begin{pmatrix} \tilde{M}^k & 0 \\ e_u & -1 \end{pmatrix},$$

where $e_u = (0, \ldots, 0, 1_u, 0, \ldots, 0)$. This shows that $\mathsf{rank}(M^k) = \mathsf{rank}(\tilde{M}^k) + 1$, that this rank is full, and hence also that $\mathsf{rank}(M^k) = \mathsf{rank}((M^k, y^k))$, as claimed. □ □

3.12 Lemma
Let $G = (V, E)$ be biconnected, and $Y \neq \varnothing$ satisfying Eq. (8) and (10). Then Eq. (12) has a solution for every $k \leq K$.

Proof. We proceed by induction on the simple cycles of G. For the base case, we consider G to be a graph consisting of a single cycle, with corresponding y satisfying Eq. (8) and (10). Since G is a cycle, it has the same number of vertices and edges, say q. This implies that, for any fixed $k \leq K$, Eq. (12) is a linear system $M^k x = y^k$ (where $y^k = (y_{uvk} \mid \{u, v\} \in E)$ with a $q \times q$ matrix:

$$M^k = \begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & 1 & \ddots & \\ & & & \ddots & -1 \\ -1 & & & & 1 \end{pmatrix}. \tag{15}$$

By Eq. (7) and by inspection of Eq. (15) it is clear that $\mathsf{rank}(M^k) = q - 1$: then Eq. (10) ensures that $\mathsf{rank}((M^k, y^k)) = \mathsf{rank}(M^k)$, and therefore that Eq. (12) has a solution.

We now tackle the induction step. The incidence vectors in E of the cycles of any graph are a vector space of dimension $m - n + 1$ over the finite field $\mathbb{F}_2 = \{0, 1\}$ [52]. We consider a fundamental cycle basis $\mathcal{B}$ of G (see Sect. 4). We assume that (a) G' is a union of fundamental cycles in $\mathcal{B}' \subsetneq \mathcal{B}$, for which Eq. (12) has a solution x' by the induction hypothesis, and (b) that C is another fundamental cycle in $\mathcal{B} \smallsetminus \mathcal{B}'$, with a solution x^C of Eq. (12) which exists by the base case. We aim at proving that Eq. (12) has a solution for $G' \cup C$. Since G is biconnected, the induction can proceed by ear decomposition [45], which means that G' is also biconnected, and that C is such that $E(G') \cap E(C) = F$ is a non-empty path in G'.

By Eq. (10) applied to C, we have

$$\forall k \leq K \quad \sum_{\{i,j\} \in C} y_{ijk} = 0. \tag{16}$$

Since x' satisfies Eq. (12) by the induction hypothesis,

$$\forall k \leq K, \{i,j\} \in F \quad x'_{ik} - x'_{jk} = y_{ijk}. \tag{17}$$

We replace Eq. (17) in Eq. (16), obtaining

$$\forall k \leq K \quad \sum_{\{i,j\} \in F} (x'_{ik} - x'_{jk}) = - \sum_{\{i,j\} \in E(C) \smallsetminus F} y_{ijk}. \tag{18}$$

Moreover, x^C also satisfies Eq. (12) over C, hence we can replace the right hand side of Eq. (18) with the corresponding terms in $x^C_{ik} - x^C_{jk}$ to get:

$$\forall k \leq K \quad \sum_{\{i,j\} \in F} (x'_{ik} - x'_{jk}) + \sum_{\{i,j\} \in E(C) \smallsetminus F} (x^C_{ik} - x^C_{jk}) = 0. \tag{19}$$

We now fix x', and aim at modifying x^C so that: (a) x^C matches x' on $V(F)$, (b) the modified x^C is still a solution of Eq. (12) on C. We set x^C_{ik} to x'_{ik} for each $i \in V(F)$, and consider the resulting linear system Eq. (12) given by M^k, as in Eq. (15), for each $k \leq K$, where we assume without loss of generality that $V(F) = \{1, \ldots, r\}$ and $V(C) = \{r+1, \ldots, s\}$:

$$\left.\begin{array}{rrrrrrrrrl}
x'_{1k} & - x'_{2k} & & & & & & = & y_{12k} & (1) \\
 & x'_{2k} & - x'_{3k} & & & & & = & y_{23k} & (2) \\
 & \ddots & \ddots & & & & & \vdots & \vdots & \vdots \\
 & & x'_{rk} & - x^C_{r+1,k} & & & & = & y_{r,r+1,k} & (r) \\
 & & & x^C_{r+1,k} & - x^C_{r+2,k} & & & = & y_{r+1,r+2,k} & (r{+}1) \\
 & & & & \ddots & \ddots & & \vdots & \vdots & \vdots \\
 & & & & & x^C_{s-1,k} & - x^C_{sk} & = & y_{s-1,s,k} & (s{-}1) \\
- x'_{1k} & & & & & & x^C_{sk} & = & y_{1sk}. & (s)
\end{array}\right\} \tag{20}$$

The equations from (1) to $(r-1)$ in Eq. (20) are satisfied by the induction hypothesis since they only depend on x', so we can remove them from the system and assume x' to be constant. We are left with:

$$\left.\begin{array}{rrrrrl}
- x^C_{r+1,k} & & & = & y_{r,r+1,k} - x'_{rk} & (r) \\
x^C_{r+1,k} & - x^C_{r+2,k} & & = & y_{r+1,r+2,k} & (r{+}1) \\
\ddots & \ddots & & \vdots & \vdots & \vdots \\
 & x^C_{s-1,k} & - x^C_{sk} & = & y_{s-1,s,k} & (s{-}1) \\
 & & x^C_{sk} & = & y_{1sk} + x'_{1k}. & (s)
\end{array}\right\} \tag{21}$$

Summing up the left hand sides of Eq. (21), we obtain:

$$\begin{array}{rl}
 & -x^C_{r+1,k} + (x^C_{r+1,k} - x^C_{r+2,k}) + \cdots + (x^C_{s-1,k} - x^C_{sk}) + x^C_{sk} \\
= & (-x^C_{r+1,k} + x^C_{r+1,k}) + \cdots + (-x^C_{sk} + x^C_{sk}) = 0
\end{array}$$

for all $k \leq K$, so the $(s-r+1) \times (s-r+1)$ matrix $\bar{M}^k$ of the k-th linear system Eq. (21) has rank $\leq s-r$. On the other hand, eliminating the first or last row makes it clear by inspection that the rest of the rows are linearly independent; therefore the rank of $\bar{M}^k$ is exactly $s-r$. Summing up the components of the right hand side vector $\bar{y}^k$ of Eq. (21), we obtain:

$$\begin{array}{rcl}
\chi & = & -x'_{rk} + y_{r,r+1,k} + y_{r+1,r+2,k} + \cdots + y_{s-1,s,k} + y_{1sk} + x'_{1k} \\
 & = & (x'_{1k} - x'_{rk}) + \displaystyle\sum_{\{i,j\} \in E(C) \smallsetminus F} y_{ijk}.
\end{array}$$

We remark that

$$\begin{array}{rcl}
x'_{1k} - x'_{rk} & = & (x'_{1k} - x'_{2k}) + (x'_{2k} - x'_{3k}) + \cdots + (x'_{r-1,k} + x'_{rk}) \\
 & = & \displaystyle\sum_{\{i,j\} \in F} (x'_{ik} - x'_{jk}) = \sum_{\{i,j\} \in F} y_{ijk}
\end{array}$$

since x' satisfies Eq. (12) by the induction hypothesis. Therefore

$$\chi = \sum_{\{i,j\}\in F} y_{ijk} + \sum_{\{i,j\}\in E(C)\smallsetminus F} y_{ijk} = \sum_{\{i,j\}\in E(C)} y_{ijk},$$

whence $\chi = 0$ by Eq. (16). This implies that $\mathsf{rank}((\bar{M}^k, \bar{y}^k)) = \mathsf{rank}(\bar{M}^k) = s - r$. Therefore, Eq. (21) has a solution, which yields the modified x^C with properties (a) and (b) given above. This concludes the induction step and the proof. □ □

We can finally give the proof of Thm. 3.2.

Proof of Thm. 3.2. The ($\Leftarrow$) part follows by Prop. 3.10. For the ($\Rightarrow$) part, we exploit a 1-decomposition of G into trees and biconnected subgraphs, derive solutions to Eq. (12) for each subgraph, and show that the solutions can be easily combined to yield a solution to Eq. (12) for the whole graph G.

We assume without loss of generality that G is connected (otherwise each connected component can be treated separately), and consider a 1-decomposition $\mathcal{D} = \{G_1, \ldots, G_r\}$ of G. By Lemmata 3.11 and 3.12, there exist solutions $x^1, \ldots x^r$ to Eq. (12) applied to $G_1, \ldots, G_r$ respectively. Consider the graph

$$\mathscr{D} = (\mathcal{D},\ \{\{i,j\} \mid 1 \leq i \neq j \leq r \wedge |V(G_i) \cap V(G_j)| = 1\}).$$

By Cor. 3.7, $\mathscr{D}$ is a tree: otherwise, a cycle in $\mathcal{D}$ would be a contraction of a cycle in G not included in a single G_i, against Cor. 3.7. This allows us to reorder $\mathcal{D}$ so that, for each $j > 1$, there is a unique $i < i$ such that $\{i,j\} \in E(\mathscr{D})$.

We remark that, for each $i \leq r$, x^i is a realization of G_i in $\mathbb{R}^K$ by Eq. (12)-(14). More precisely, x^i is a $|V(G_i)| \times K$ matrix $x^i = (x^i_{\ell k})$ so that $x^i_\ell = (x^i_{\ell 1}, \ldots, x^i_{\ell K})$ is the position of vertex $\ell \in V(G_i)$ in $\mathbb{R}^K$. Note that the realizations $x^1, \ldots, x^r$ can be modified by translations without changing the values of y (by inspection of Eq. (12)).

We now construct a solution $\bar{x}$ of Eq. (12) for G by induction on $\mathcal{D}$ ordered as described above. For the base case $i = 1$, we fix x^1 in any way (e.g. by taking the centroid of the rows of x^1 to be the origin), and initialize the first $|V(G_1)|$ rows of $\bar{x}$ with those of x^1. For any $i > 1$, we identify the unique predecessor j of i in the order on $\mathcal{D}$. The induction hypothesis ensures the existence of a solution $\bar{x}$ of the union of $G_1, \ldots, G_j$. Consider the cut vertex v in $V(G_j) \cap V(G_i)$ guaranteed by definition of the order on $\mathcal{D}$, and let $\bar{x}_v \in \mathbb{R}^K$ be its position. Then the translation $\tilde{x}^i = x^i - \mathbf{1}(x^i_v - \bar{x}_v)^\top$ yields another valid solution of Eq. (12) applied to G^i by translation invariance, and this solution is such that $\tilde{x}^i_v = \bar{x}_v$. Therefore, using the rows of $\tilde{x}^i$, $\bar{x}$ can be extended to a solution of Eq. (12) applied to the union of $G_1, \ldots, G_j$ and G^i, as claimed. □

Thm. 3.2 can also be interpreted as a polynomial reduction of the EDGP to the problem of finding a solution of Eq. (8) and (10).

3.13 Corollary
Deciding feasibility of Eq. (8) *and* (10) *is* **NP**-*hard.*

Proof. By reduction from EDGP using Thm. 3.2. □ □

A remarkable consequence of Thm. 3.2 is that it allows a decomposition of the computation of the realization x into two stages: first, solve Eq. (8)-(10) to find a feasible y^*; then solve

$$\forall k \leq K, \{i,j\} \in E \quad x_{ik} - x_{jk} = y^*_{ijk} \tag{22}$$

to find a realization x^*. We note that Eq. (22) is just a restatement of Eq. (12) universally quantified over k.

3.14 Corollary
Given a solution y^* solving Eq. (8) and Eq. (10), any solution x^* of Eq. (22) is a valid realization of the EDGP instance (K, G).

Proof. The feasibility of Eq. (22) with the right hand side replaced by a solution y^* of Eq. (8) and (10) follows directly from Thm. 3.2, since if such a y^* exists then the EDGP is feasible. □ □

The first stage is **NP**-hard by Cor. 3.13, while the second stage is tractable, since solving linear systems can be done in polynomial time.

3.15 Remark
Note that Eq. (22) has Km equations, but its rank may be lower, since there are only Kn variables: in particular, Eq. (22) may be an overdetermined linear system. The feasibility of this system is guaranteed by Cor. 3.14; in particular, the steps of the proof of Thm. 3.2 imply that Eq. (22) loses rank w.r.t. Km according to the incidence of the edges in the cycles of G. In other words, any solution y' to Eq. (10) provides a right hand side to Eq. (22) that makes the system feasible.

The issue with Thm. (3.2) is that it relies on the exponentially large family of constraints Eq. (10). While this is sometimes addressed by algorithmic techniques such as row generation, we shall see in the following that it suffices to consider a polynomial set of cycles (which, moreover, can be found in polynomial time) in the quantifier of Eq. (10).

4 The cycle vector space and its bases

We recall that incidence vectors of cycles (in a Euclidean space having $|E|$ dimensions) form a vector space over a field $\mathbb{F}$, which means that every cycle can be expressed as a weighted sum of cycles in a basis. In this interpretation, a *cycle* in G is simply a subgraph of G where each vertex has even degree: we denote their set by $\mathcal{C}$. This means that Eq. (10) is actually quantified over a subset of $\mathcal{C}$, namely the simple connected cycles. Every basis has cardinality $m - n + a$, where a is the number of connected components of G. If G is connected, cycle bases have cardinality $m - n + 1$ [52].

Our interest in introducing cycle bases is that we would like to quantify Eq. (10) polynomially rather than exponentially in the size of G. Our goal is to replace "C is any simple connected cycle in $\mathcal{C}$" by "C is a cycle in a cycle basis of G". In order to show that this limited quantification is enough to imply every constraint in Eq. (10), we have to show that, for each simple connected cycle $C \in \mathcal{C}$, the corresponding constraint in Eq. (10) can be obtained as a weighted sum of constraints corresponding to the basis elements.

Another feature of Eq. (10) to keep in mind is that edges are implicitly given a direction: for each cycle, the term for the *undirected* edge $\{i, j\}$ in Eq. (10) is $(x_{ik} - x_{jk})$. Note that while $\{i, j\}$ is exactly the same vertex set as $\{j, i\}$, the corresponding term is either positive or not, depending on the direction (i, j) or (j, i). We deal with this issue by arbitrarily directing the edges in E to obtain a set A of arcs, and considering *directed* cycles in the directed graph $\bar{G} = (V, A)$. In this interpretation, the incidence vector of a directed cycle C of $\bar{G}$ is a vector $c^C \in \mathbb{R}^m$ satisfying [24, §2, p. 201]:

$$\forall j \in V(C) \quad \sum_{(i,j)\in A} c^C_{ij} = \sum_{(j,\ell)\in A} c^C_{j\ell}. \tag{23}$$

A directed circuit D of $\bar{G}$ is obtained by applying the edge directions from $\bar{G}$ to a connected subgraph of G where each vertex has degree exactly 2 (note that a directed circuit need not be strongly connected,

although its undirected version is connected). Its incidence vector $c^D \in \{-1,0,1\}^m$ is defined as follows:

$$\forall (i,j) \in A \qquad c^D_{ij} \triangleq \begin{cases} 1 & \text{if} \quad (i,j) \in A(D) \\ -1 & \text{if} \quad (j,i) \in A(D) \\ 0 & \text{otherwise} \end{cases}$$

where we have used $A(D)$ to mean the arcs in the subgraph D. In other words, whenever we walk over an arc (i,j) in the natural direction $i \to j$ we let the (i,j)-th component of c^D be 1; if we walk over (i,j) in the direction $j \to i$ we assign a -1, and otherwise a zero.

4.1 Constraints over cycle bases

The properties of undirected and directed cycle bases have been investigated in a sequence of papers by many authors, culminating with [24]. We now prove that it suffices to quantify Eq. (10) over a directed cycle basis.

4.1 Proposition
Let $\mathcal{B}$ be a directed cycle basis of $\bar{G}$ over $\mathbb{Q}$. Then Eq. (10) holds if and only if:

$$\forall k \leq K, B \in \mathcal{B} \qquad \sum_{(i,j) \in A(B)} c^B_{ij} y_{ijk} = 0. \tag{24}$$

Proof. Necessity (10) $\Rightarrow$ (24) follows because Eq. (10) is quantified over all cycles: in particular, it follows for any undirected cycle in any undirected cycle basis. Moreover, the signs of all terms in the sum of Eq. (24) are consistent, by definition, with the arbitrary edge direction chosen for $\bar{G}$.

Next, we claim sufficiency (24) $\Rightarrow$ (10). Let $C \in \mathcal{C}$ be a simple cycle, and $\bar{C}$ be its directed version with the directions inherited from $\bar{G}$. Since $\mathcal{B}$ is a cycle basis, we know that there is a coefficient vector $(\gamma_B \mid B \in \mathcal{B}) \in \mathbb{R}^{|\mathcal{B}|}$ such that:

$$c^{\bar{C}} = \sum_{B \in \mathcal{B}} \gamma_B c^B. \tag{25}$$

We now consider the expression:

$$\forall k \leq K \quad \sum_{B \in \mathcal{B}} \gamma_B \sum_{(i,j) \in A(B)} c^B_{ij} y_{ijk}. \tag{26}$$

On the one hand, by Eq. (25), Eq. (26) is identically equal to $\sum_{(i,j) \in A(\bar{C})} c^{\bar{C}}_{ij} y_{ijk}$ for each $k \leq K$; on the other hand, each inner sum in Eq. (26) is equal to zero by Eq. (24). This implies $\sum_{(i,j) \in A(\bar{C})} c^{\bar{C}}_{ij} y_{ijk} = 0$ for each $k \leq K$. Since C is simple and connected, $\bar{C}$ is a directed circuit. This implies that $c^{\bar{C}} \in \{-1,0,1\}$. Now it suffices to replace $-y_{ijk}$ with y_{jik} to obtain

$$\forall k \leq K \qquad \sum_{\{i,j\} \in E(C)} y_{ijk} = 0,$$

where the edges on C are indexed in such a way as to ensure they appear in order of consecutive adjacency. □ □

Obviously, if $\mathcal{B}$ has minimum (or just small) cardinality, Eq. (24) will be sparsest (or just sparse), which is often a desirable property of linear constraints occurring in MP formulations. Hence we should attempt to find short cycle bases $\mathcal{B}$.

In summary, given a basis $\mathcal{B}$ of the directed cycle space of $\bar{G}$ where c^B is the incidence vector of a cycle $B \in \mathcal{B}$, the following:

$$\left.\begin{array}{rll} \min\limits_{s\geq 0, y} & \sum\limits_{\{i,j\}\in E} (s^+_{ij} + s^-_{ij}) & \\ \forall (i,j) \in A(\bar{G}) & \sum\limits_{k\leq K} y^2_{ijk} - d^2_{ij} & = \quad s^+_{ij} - s^-_{ij} \\ \forall k \leq K, B \in \mathcal{B} & \sum\limits_{(i,j)\in A(B)} c^B_{ij} y_{ijk} & = \quad 0 \end{array}\right\} \tag{27}$$

is a valid formulation for the EDGP. The solution of Eq. (27) yields a feasible vector y^*. As pointed out in Cor. 3.14, we must then solve Eq. (22) to obtain a realization x^* for G.

4.2 How to find directed cycle bases

We require directed cycle bases over $\mathbb{Q}$. By [24, Thm. 2.4], each undirected cycle basis gives rise to a directed cycle basis (so it suffices to find a cycle basis of G and then direct the cycles using the directions in $\bar{G}$). Horton's algorithm [22] and its variants [19, 44] find a minimum cost cycle basis in polynomial time. The most efficient deterministic variant is $O(m^3 n)$ [44], and the most efficient randomized variant has the complexity of matrix multiplication. Existing approximation algorithms have marginally better complexity.

It is not clear, however, that the provably sparsest constraint system will make the DGP actually easier to solve. We therefore consider a much simpler algorithm: starting from a spanning tree, we pick the $m-n+1$ circuits that each *chord* (i.e., non-tree) edge defines with the rest of the tree. This algorithm [49] yields a *fundamental* cycle basis (FCB). Finding the minimum FCB is known to be **NP**-hard [13], but heuristics based on spanning trees prove to be very easy to implement and work reasonably well [13] (optionally, their cost can be improved by an edge-swapping phase [1, 30]).

5 The Eulerian cycle relaxation

In this section we construct a relaxation of Eq. (27) that decreases the number of constraints in Eq. (24), which occurs as the last line in Eq. (27), from $|\mathcal{B}|$ to 1.

We let G' be the multigraph obtained from G by adding sufficiently many parallel edges to G, so that the degree of each vertex in G' is even. This can always be done by [16], which implies that G' is Eulerian, i.e. it has a cycle incident with every edge in G' exactly once. We let $\mathscr{E}$ be a Eulerian cycle in G', and let $\bar{\mathscr{E}}$ be either of the two orientations of $\mathscr{E}$ obtained by walking over the cycle. We let $\bar{G}'$ be the digraph induced by the Eulerian circuit $\bar{\mathscr{E}}$. For each $\{i,j\} \in E$ let H_{ij} be the number of parallel edges between i, j in G'.

We note that $\bar{G}'$ might have parallel and antiparallel arcs. Consider the family of arc subset $\mathcal{H}_{ij} = \{(i', j', h) \mid h \leq H_{ij} \wedge \{i', j'\} = \{i, j\}\}$ of $A(\bar{G}')$. We replace each arc $(i', j', h) \in \mathcal{H}_{ij}$ having $h > 1$ by an oriented 2-path $p_{i'j'h} = \{(i', v_{ijh}), (v_{ijh}, j')\}$ involving a new added vertex v_{ijh}. Call $\tilde{G}$ the digraph obtained from $\bar{G}'$ with this replacement. We remark that $\tilde{G}$ is simple (it has no parallel/antiparallel arcs) by construction. Moreover, $\tilde{G}$ is a Eulerian digraph: take the Eulerian circuit $\bar{\mathscr{E}}$ in $\bar{G}'$, and, every time it traverses a parallel/antiparallel arc $(i', j', h) \in \mathcal{H}_{ij}$ with $h > 1$, let it traverse the oriented 2-path replacement $p_{i'j'h}$ instead: this is clearly a Euclidean circuit in $\tilde{G}$, which we call $\mathscr{C}$.

Next we consider the simple graph $\hat{G}$ obtained by replacing each arc in $\tilde{G}$ with an edge. Let $\hat{V} = \{v_{ijh} \mid \{i,j\} \in E \wedge h > 1\}$, and $\hat{E}$ be the subset of edges of $E(\hat{G})$ obtained from losing the orientation of the arcs in the union

$$\bigcup_{\substack{(i',j',h)\in\mathcal{H}_{ij} \\ \{i,j\}\in E\wedge h>1}} p_{i'j'h}$$

of all the arcs from the 2-path replacements. We note that, by construction,

$$\hat{V} = V(\hat{G}) \smallsetminus V \quad \wedge \quad \hat{E} = E(\hat{G}) \smallsetminus E. \tag{28}$$

Let $c^{\mathscr{C}}_{ij} \in \{1, -1\}$ be the orientation of (i, j) in $\mathscr{C}$ w.r.t. $\hat{G}$; let $\hat{\mathscr{C}}$ be the simple Eulerian cycle in $\hat{G}$ corresponding to $\mathscr{C}$.

We can now prove the main result of this section.

5.1 Proposition
The formulation

$$\left.\begin{array}{rrcl} \min\limits_{s \geq 0, y} & \sum\limits_{\{i,j\} \in E} (s^+_{ij} + s^-_{ij}) & & \\ \forall (i,j) \in A(\bar{G}) & \sum\limits_{k \leq K} y^2_{ijk} - d^2_{ij} & = & s^+_{ij} - s^-_{ij} \\ \forall k \leq K & \sum\limits_{(i,j) \in \mathscr{C}} c^{\mathscr{C}}_{ij} y_{ijk} & = & 0 \quad (\dagger) \end{array}\right\} \tag{29}$$

is a relaxation of Eq. (27)*.*

Proof. We form a variant of the cycle formulation Eq. (27) applied to $\hat{G}$, where, from the constraints corresponding to Eq. (8) (second line of Eq. (27)), we omit those indexed by $\hat{E}$. We call this variant $(\star)$. We claim that $(\star)$ is an exact reformulation of Eq. (27) applied to G. The claim holds because $E(\hat{G}) \smallsetminus \hat{E} = E$ by Eq. (28), and because the signs of the y variables are irrelevant in Eq. (8) since they are squared. Now, since $\hat{\mathscr{C}}$ is a Eulerian cycle in $\hat{G}$, Eq. (†) is an aggregation of constraints in Eq. (24), which occur within the reformulation $(\star)$. So Eq. (29) is a relaxation of $(\star)$. The proposition follows because of the claim. □ □

Note that Eq. (29) provides a solution $\bar{y}$ which may not satisfy Eq. (24), which also guarantee feasibility in Eq. (10) by Prop. 4.1. By Remark 3.15, this implies that Cor. 3.14 is no longer applicable. In other words, the realization x of G cannot in general be retrieved from $\bar{y}$ using the linear system in Eq. (22), since $\bar{y}$ might well make Eq. (22) infeasibile. Eq. (22), however, can instead be integrated into Eq. (29) as additional constraints. This invalidates the decomposition property of Cor. 3.14, but allows the relaxation to yield a valid realization.

We therefore define the Eulerian cycle-based relaxation formulation as follows:

$$\left.\begin{array}{rrcl} \min\limits_{s \geq 0, x, y} & \sum\limits_{\{i,j\} \in E} (s^+_{ij} + s^-_{ij}) & & \\ \forall (i,j) \in A(\bar{G}) & \sum\limits_{k \leq K} y^2_{ijk} - d^2_{ij} & = & s^+_{ij} - s^-_{ij} \\ \forall k \leq K & \sum\limits_{(i,j) \in A(\hat{\mathscr{E}})} c^{\hat{\mathscr{E}}}_{ij} y_{ijk} & = & 0 \\ \forall \{i,j\} \in A(\bar{G}) & x_{ik} - x_{jk} & = & y_{ijk} \\ \forall k \leq K & \sum\limits_{i \in V} x_{ik} & = & 0. \end{array}\right\} \tag{30}$$

For a formulation P, we denote by $\mathsf{val}(P)$ its optimal objective function value. Since Eq. (30) has additional constraints w.r.t. Eq. (29), we naturally have $\mathsf{val}(30) \geq \mathsf{val}(29)$. Moreover, for every instance for which a solution $\bar{y}$ of Eq. (29) yields an infeasible system Eq. (22), by inspection $\bar{y}$ must be infeasible in Eq. (30), which implies that there are cases where Eq. (30) is a strictly tighter relaxation than Eq. (29). The very last constraint in Eq. (30) fixes the centroid of the points at the origin, as in Eq. (4).

6 Computational experiments

The aim of this section is to compare the computational performance of the following EDGP formulations:

(i) the cycle-based formulation in Eq. (27), where the realization is retrieved as a post-processing stage using (22) according to Cor. 3.14;

(ii) the Eulerian cycle-based relaxation in Eq. (30);

(iii) the classic edge-based formulation in Eq. (4).

All of these formulations are nonconvex Nonlinear Programs (NLP), which are generally NP-hard to solve. Specifically, formulations (i) and (iii) are as hard to solve as the EDGP, which is NP-hard. No specific NP-hardness proof exists for formulation (ii) yet.

As a solution algorithm, we used a very simple MultiStart (MS) heuristic based on calling a local NLP solver from a random initial starting point at each iteration, and updating the best solution found so far as needed: although there are better heuristics around [39, 12, 47], MS is the best trade-off between implementation simplicity and efficiency. Moreover, more efficient heuristic often change the formulation during their execution, which may hinder the meaning of this computational comparison between formulations.

We evaluate the quality of a realization x of a graph G according to mean (MDE) and largest distance error (LDE), defined this way:

$$\begin{aligned} \mathsf{mde}(x, G) &= \frac{1}{|E|} \sum_{\{i,j\} \in E} \big| \|x_i - x_j\|_2 - d_{ij} \big| \\ \mathsf{lde}(x, G) &= \max_{\{i,j\} \in E} \big| \|x_i - x_j\|_2 - d_{ij} \big|. \end{aligned}$$

The CPU time taken to find the solution may also be important, depending on the application. In the control of underwater vehicles [3], for example, DGP instances might need to be solved in real time. In other applications, such as finding protein structure from distance data [8, 46] (our application of choice), the CPU time is not so important.

Our tests were carried out on a single CPU of a 2.1GHz 4-CPU 8-core-per-CPU machine with 64GB RAM running Linux. The local NLP solver used within the MS heuristic was the IPOpt solver [9]. We remarked in some preliminary tests that IPOpt was considerably slowed down by variants of Eq. (3) such as Eq. (5), which essentially move a nonconvexity on the objective to one in the constraints. The same holds for the cycle-based formulation in Eq. (27). We therefore reformulated Eq. (27) as follows:

$$\left.\begin{aligned} \min_{y} \quad & \sum_{\{i,j\} \in A(\bar{G})} \Big(\sum_{k \leq K} y_{ijk}^2 - d_{ij}^2 \Big)^2 \\ \forall k \leq K, B \in \mathcal{B} \quad & \sum_{(i,j) \in A(B)} c_{ij}^B y_{ijk} = 0, \end{aligned} \right\} \tag{31}$$

and Eq. (30) similarly.

Our implementation consists of a mixture of Python 3 [55] and AMPL [18] interfaced through `amplpy`. Cycle bases and Eulerian cycles are found using `networkX` [20]. Solutions to the feasible but possibly overdetermined linear systems in Eq. (22) are obtained using an ℓ_1 error minimization approach reformulated as a Linear Programming problem solved with CPLEX [23].

6.1 Results

A benchmark on a diverse collection of randomly generated weighted graphs of small size and many different types, with a very similar set-up to the one discussed here, is presented in [33]. It was found that the cycle formulation finds better MDE values, while the edge formulation generally finds better LDE values and is faster. Some results on proteins, obtained with only 3 MS iterations, were also presented in [33].

The benchmark we consider here contains medium to large scale protein graph instances realized in $\mathbb{R}^3$. W.r.t. the protein results presented in [33], we integrated one more instance, `1tii`, which, at 69800 edges and 5684 vertices, is considerably larger than all the others.

The results are given in Table 1. We report instance name, instance sizes m and n, then performance measures MDE, LDE and CPU for cycle, Eulerian and edge-based formulations. In the last three lines we report average, standard deviation, and number of instances where the formulation performed best, for all performance measures. In all tested cases, finding the cycle basis, the Eulerian cycles, and solving Eq. (22) took a small fraction of the total solution time. The missing result for instance `100d` on the Eulerian cycle reformulation is due to a failure occurred in the `networkX` module because the graph of `100d` is not connected.

			MDE			LDE			CPU		
Instance	m	n	cycle	Eul	edge	cycle	Eul	edge	cycle	Eul	edge
`1guu`	955	150	0.086	0.069	**0.053**	1.234	1.068	**1.037**	**7.90**	553.76	290.21
`1guu-1`	959	150	0.080	0.082	**0.059**	1.013	1.069	**0.980**	9.67	23.03	**1.72**
`1guu-4000`	968	150	0.112	0.106	**0.092**	1.073	1.431	**0.936**	8.68	10.77	**1.56**
`pept`	999	107	**0.144**	0.239	0.179	2.862	**1.847**	1.943	5.52	4.72	**1.4**
`2kxa`	2711	177	**0.051**	0.119	0.172	3.705	**2.826**	3.813	21.53	25.54	**7.35**
`res_2kxa`	2627	177	**0.055**	0.237	0.156	**2.949**	3.570	3.054	20.84	21.20	**12.44**
`C0030pkl`	3247	198	**0.000**	0.145	0.211	**0.000**	3.537	3.829	29.50	26.69	**7.36**
`cassioli`	4871	281	0.146	0.113	**0.057**	3.914	3.616	**3.185**	47.23	48.44	**14.51**
`100d`	5741	488	**0.201**	-	0.251	**3.038**	-	3.987	387.32	-	**29.42**
`hlx_amb`	6265	392	**0.105**	0.214	0.119	3.836	3.888	**3.485**	120.25	80.27	**20.54**
`water`	11939	648	**0.146**	0.490	0.243	**3.579**	4.196	4.281	1346.69	399.42	**224.66**
`3al1`	17417	678	**0.062**	0.126	0.216	3.451	**3.175**	4.059	835.10	433.69	**123.45**
`1hpv`	18512	1629	**0.385**	0.402	0.416	3.847	**3.831**	4.015	10138.00	2387.29	**442.70**
`il2`	45251	2084	0.385	**0.049**	0.107	4.422	**4.204**	4.583	18141.22	9904.81	**5255.76**
`1tii`	69800	5684	0.620	0.436	**0.434**	6.755	4.492	**3.854**	18846.37	38230.21	**9039.28**
avg			**0.172**	0.202	0.184	**3.045**	3.054	3.136	3331.05	3724.99	**1031.49**
stdev			0.167	0.144	**0.118**	1.673	**1.204**	1.272	6672.49	10272.3	**2587.33**
$\|best\|$			**9**	1	5	4	5	**6**	1	0	**14**

Table 1: Cycle formulation vs. edge formulation performance on protein graphs (realizations in $K = 3$ dimensions).

It appears that, on average, there is relatively little difference between the quality performances of these three DGP formulations on protein graphs of medium and large sizes. CPU-time wise, of course, the edge formulation is best. Cycle formulations, taken together, are definitely better than the edge formulation on quality measures. The cycle-based formulation Eq. (27) is slightly better than the other formulations for both MDE and LDE. The number of instances on which Eq. (27) is best on quality measures is 13, against 11 for the edge-based formulation. Eq. (27) was the only formulation by which a global optimum was found (that of `C0030pkl`). All in all, we believe that our results show that cycle formulations are credible competitors w.r.t. the well established edge-based formulations, especially when the CPU time is not an important performance measure (which is generally the case in the protein conformation application).

Acknowledgements

While the seminal idea for considering DGPs over cycles dates from Saxe's **NP**-hardness proof [50], the "cycle formulation" concept occurred to us as one of the authors (LL) attended a talk by Matteo Gallet given at the Erwin Schrödinger Institute (ESI), Vienna, during the Geometric Rigidity workshop 2018. LL has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement n. 764759 "MINOA". CL is grateful to the Brazilian research agencies FAPESP and CNPq for support. NM is grateful to the Brazilian research agencies COPPETEC Foundation and CNPq for support.

References

[1] E. Amaldi, L. Liberti, F. Maffioli, and N. Maculan. Edge-swapping algorithms for the minimum fundamental cycle basis problem. *Mathematical Methods of Operations Research*, 69:205–223, 2009.

[2] J. Aspnes, T. Eren, D. Goldenberg, S. Morse, W. Whiteley, R. Yang, B. Anderson, and P. Belhumeur. A theory of network localization. *IEEE Transactions on Mobile Computing*, 5(12):1663–1678, 2006.

[3] A. Bahr, J. Leonard, and M. Fallon. Cooperative localization for autonomous underwater vehicles. *International Journal of Robotics Research*, 28(6):714–728, 2009.

[4] N. Beeker, S. Gaubert, C. Glusa, and L. Liberti. Is the distance geometry problem in NP? In A. Mucherino, C. Lavor, L. Liberti, and N. Maculan, editors, *Distance Geometry: Theory, Methods, and Applications*, pages 85–94. Springer, New York, 2013.

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

[6] A. Berg and T. Jordán. Algorithms for graph rigidity and scene analysis. In G. Di Battista and U. Zwick, editors, *Algorithms: Proceedings of the European Symposium on Algorithms*, volume 2832 of *LNCS*, pages 78–89, Berlin, 2003. Springer.

[7] M. Bukatin, R. Kopperman, S. Matthews, and H. Pajoohesh. Partial metric spaces. *American Mathematical Monthly*, 116(8):708–718, 2009.

[8] A. Cassioli, B. Bordeaux, G. Bouvier, A. Mucherino, R. Alves, L. Liberti, M. Nilges, C. Lavor, and T. Malliavin. An algorithm to enumerate all possible protein conformations verifying a set of distance constraints. *BMC Bioinformatics*, 16:23–38, 2015.

[9] COIN-OR. *Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT*, 2006.

[10] R. Connelly. Generic global rigidity. *Discrete Computational Geometry*, 33:549–563, 2005.

[11] C. D'Ambrosio and L. Liberti. Distance geometry in linearizable norms. In F. Nielsen and F. Barbaresco, editors, *Geometric Science of Information*, volume 10589 of *LNCS*, pages 830–838, Berlin, 2017. Springer.

[12] C. D'Ambrosio, Ky Vu, C. Lavor, L. Liberti, and N. Maculan. New error measures and methods for realizing protein graphs from distance data. *Discrete and Computational Geometry*, 57(2):371–418, 2017.

[13] N. Deo, G.M. Prabhu, and M.S. Krishnamoorthy. Algorithms for generating fundamental cycles in a graph. *ACM Transactions on Mathematical Software*, 8(1):26–42, March 1982.

[14] G. Dias and L. Liberti. Diagonally dominant programming in distance geometry. In R. Cerulli, S. Fujishige, and R. Mahjoub, editors, *International Symposium in Combinatorial Optimization*, volume 9849 of *LNCS*, pages 225–236, New York, 2016. Springer.

[15] Y. Ding, N. Krislock, J. Qian, and H. Wolkowicz. Sensor network localization, Euclidean distance matrix completions, and graph realization. *Optimization and Engineering*, 11:45–66, 2010.

[16] J. Edmonds and E. Johnson. Matching, Euler tours, and the Chinese postman. *Mathematical Programming*, 5:88–124, 1973.

[17] T. Eren, D. Goldenberg, W. Whiteley, Y. Yang, A. Morse, B. Anderson, and P. Belhumeur. Rigidity, computation, and randomization in network localization. *IEEE*, pages 2673–2684, 2004.

[18] R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.

[19] A. Golynski and J.D. Horton. A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In *8th Scandinavian Workshop on Algorithm Theory*, 2002.

[20] A. Hagberg, D. Schult, and P. Swart. Exploring network structure, dynamics, and function using `NetworkX`. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA, 2008.

[21] P. Hoffman and B. Richter. Embedding graphs in surfaces. *Journal of Combinatorial Theory B*, 36:65–84, 1984.

[22] J.D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal of Computing*, 16(2):358–366, 1987.

[23] IBM. *ILOG CPLEX 12.9 User's Manual*. IBM, 2019.

[24] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, and K. Zweig. Cycle bases in graphs: characterization, algorithms, complexity, and applications. *Computer Science Review*, 3:199–243, 2009.

[25] M. Laurent. Cuts, matrix completions and graph rigidity. *Mathematical Programming*, 79:255–283, 1997.

[26] C. Lavor, L. Liberti, and N. Maculan. Computational experience with the molecular distance geometry problem. In J. Pintér, editor, *Global Optimization: Scientific and Engineering Case Studies*, pages 213–225. Springer, Berlin, 2006.

[27] C. Lavor, L. Liberti, and N. Maculan. Molecular distance geometry problem. In C. Floudas and P. Pardalos, editors, *Encyclopedia of Optimization*, pages 2305–2311. Springer, New York, second edition, 2009.

[28] C. Lavor, L. Liberti, N. Maculan, and A. Mucherino. The discretizable molecular distance geometry problem. *Computational Optimization and Applications*, 52:115–146, 2012.

[29] C. Lavor, L. Liberti, N. Maculan, and A. Mucherino. Recent advances on the discretizable molecular distance geometry problem. *European Journal of Operational Research*, 219:698–706, 2012.

[30] J. Lee and L. Liberti. A matroid view of key theorems for edge-swapping algorithms. *Mathematical Methods of Operations Research*, 76:125–127, 2012.

[31] L. Liberti. A new distance geometry method for constructing word and sentence vectors. In *Companion Proceedings of the Web Conference (DL4G Workshop)*, volume 20 of *WWW*, New York, 2020. ACM.

[32] L. Liberti. Distance geometry and data science. *TOP*, to appear.

[33] L. Liberti, G. Iommazzo, C. Lavor, and N. Maculan. A cycle-based formulation of the Distance Geometry Problem. In C. Gentile et al., editor, *Proceedings of 18th Cologne-Twente Workshop*, volume 4 of *AIRO*, New York, 2020. Springer.

[34] L. Liberti and S. Kucherenko. Comparison of deterministic and stochastic approaches to global optimization. *International Transactions in Operational Research*, 12:263–285, 2005.

[35] L. Liberti and C. Lavor. *Euclidean Distance Geometry: An Introduction*. Springer, New York, 2017.

[36] L. Liberti and C. Lavor. Open research areas in distance geometry. In A. Migalas and P. Pardalos, editors, *Open Problems in Optimization and Data Analysis*, volume 141 of *SOIA*, pages 183–223. Springer, New York, 2018.

[37] L. Liberti, C. Lavor, J. Alencar, and G. Abud. Counting the number of solutions of ^{k}DMDGP instances. In F. Nielsen and F. Barbaresco, editors, *Geometric Science of Information*, volume 8085 of *LNCS*, pages 224–230, New York, 2013. Springer.

[38] L. Liberti, C. Lavor, and N. Maculan. A branch-and-prune algorithm for the molecular distance geometry problem. *International Transactions in Operational Research*, 15:1–17, 2008.

[39] L. Liberti, C. Lavor, N. Maculan, and F. Marinelli. Double variable neighbourhood search with smoothing for the molecular distance geometry problem. *Journal of Global Optimization*, 43:207–218, 2009.

[40] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino. Euclidean distance geometry and applications. *SIAM Review*, 56(1):3–69, 2014.

[41] L. Liberti, C. Lavor, A. Mucherino, and N. Maculan. Molecular distance geometry methods: from continuous to discrete. *International Transactions in Operational Research*, 18:33–51, 2010.

[42] L. Liberti, N. Mladenović, and G. Nannicini. A good recipe for solving MINLPs. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Hybridizing metaheuristics and mathematical programming*, volume 10 of *Annals of Information Systems*, pages 231–244, New York, 2009. Springer.

[43] L. Liberti and K. Vu. Barvinok's naive algorithm in distance geometry. *Operations Research Letters*, 46:476–481, 2018.

[44] C. Liebchen and R. Rizzi. A greedy approach to compute a minimum cycle basis of a directed graph. *Information Processing Letters*, 94:107–112, 2005.

[45] L. Lovász and M. Plummer. On minimal elementary bipartite graphs. *Journal of Combinatorial Theory B*, 23:127–138, 1977.

[46] T. Malliavin, A. Mucherino, C. Lavor, and L. Liberti. Systematic exploration of protein conformational space using a distance geometry approach. *Journal of Chemical Information and Modeling*, 59:4486–4503, 2019.

[47] L. Mencarelli, Y. Sahraoui, and L. Liberti. A multiplicative weights update algorithm for MINLP. *EURO Journal on Computational Optimization*, 5:31–86, 2017.

[48] A. Mucherino, C. Lavor, and L. Liberti. Exploiting symmetry properties of the discretizable molecular distance geometry problem. *Journal of Bioinformatics and Computational Biology*, 10:1242009(1–15), 2012.

[49] K. Paton. An algorithm for finding a fundamental set of cycles of a graph. *Communications of the ACM*, 12(9):514–518, 1969.

[50] J. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. *Proceedings of 17th Allerton Conference in Communications, Control and Computing*, pages 480–489, 1979.

[51] G. Schaeffer. Random sampling of large planar maps and convex polyhedra. In *Proceedings of the 31st Annual ACM Symposium on the Theory of Computing*, STOC, pages 760–769, New York, 1999. ACM.

[52] S. Seshu and M.B. Reed. *Linear Graphs and Electrical Networks*. Addison-Wesley, Reading, MA, 1961.

[53] A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis*, 30:20–36, 2011.

[54] M. Tawarmalani and N.V. Sahinidis. Global optimization of mixed integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.

[55] G. van Rossum and *et al. Python Language Reference, version 3*. Python Software Foundation, 2019.