A cycle-based formulation for the Distance Geometry Problem

Leo Liberti, Gabriele Iommazzo, Carlile Lavor, and Nelson Maculan

Abstract The distance geometry problem consists in finding a realization of a weighed graph in a Euclidean space of given dimension, where the edges are realized as straight segments of length equal to the edge weight. We propose and test a new mathematical programming formulation based on the incidence between cycles and edges in the given graph.

1 Introduction

The Distance Geometry Problem (DGP), also known as the *realization problem* in geometric rigidity, belongs to a more general class of metric completion and embedding problems.

> 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 $\{i, j\} \in E$.

L. Liberti
LIX CNRS Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France, e-mail: liberti@lix.polytechnique.fr

G. Iommazzo
LIX Ecole Polytechnique, France and DI Università di Pisa, Italy, e-mail: giommazz@lix.polytechnique.fr

C. Lavor
IMECC, University of Campinas, Brazil, e-mail: clavor@ime.unicamp.br

N. Maculan
COPPE, Federal University of Rio de Janeiro (UFRJ), Brazil, e-mail: maculan@cos.ufrj.br

In its most general form, the DGP might be parametrized over any norm. In practice, the ℓ_2 norm is the most usual choice. 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.

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 [29]. 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 [2]. The Molecular DGP (MDGP) aims at finding the positions of atoms in a protein, given some of the pairwise Euclidean distances [15, 16]. 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 and strongly **NP**-hard even when restricted to integer edge weights in $\{1, 2\}$ in general graphs [27]. 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 [19]. More information about the DGP can be found in [22].

There are many approaches to solving the DGP. Generally speaking, application-specific solution algorithms exploit some of the graph structure, if 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* [7] which can, at least generically, be imposed directly on the unweighted input graph. For protein structures, on the other hand, which are found in nature in several isomers, one is often interested in finding all (incongruent) realizations of the given protein graph [20]. 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 [21, 18]. In absence of any information on the graph structure, however, one can resort to Mathematical Programming (MP) formulations and corresponding solvers [23, 8].

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. Each solution x^* having global minimum value equal to zero is a realization of the given graph.

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, some variants, and a computational comparison with a well-known edge-based formulation.

2 A new formulation based on cycles

In this section we propose a new formulation for the EDGP. The basic idea stems from 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 [27] for proving weak **NP**-hardness of the DGP on cycle graphs, by reduction from PARTITION. 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 $x : V \longrightarrow \mathbb{R}^K$ we denote by $x[U]$ the restriction of x to a subset $U \subseteq V$.

Lemma 1 *Given an integer $K > 0$, a simple undirected weighted graph $G = (V, E, d)$ and a mapping $x : V \longrightarrow \mathbb{R}^K$, 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:*

$$\sum_{(i,j)\in W(C)} (x_{ik} - x_{jk}) = 0. \tag{4}$$

Proof We renumber the vertices in $V(C)$ to $1, 2, \ldots, \gamma = |V(C)|$ following the walk order in $W(C)$. Then Eq. (4) 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:

$$\forall \{i, j\} \in E \qquad \sum_{k \leq K} y_{ijk}^2 = d_{ij}^2. \tag{5}$$

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

$$\forall \{i, j\} \in E \quad \sum_{k \leq K} |y_{ijk}| = d_{ij} \quad \text{or} \quad \max_{k \leq K} |y_{ijk}| = d_{ij}. \tag{6}$$

Next, we adjoin the constraints on cycles:

$$\forall k \leq K, C \subset G \quad \left(C \text{ is a cycle} \Rightarrow \sum_{\{i,j\} \in E(C)} y_{ijk} = 0 \right). \tag{7}$$

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:

$$\forall k \leq K, \{i, j\} \in E \quad -d_{ij} \leq y_{ijk} \leq d_{ij}. \tag{8}$$

We now prove our main result, i.e. that Eq. (5) and (7) are a valid MP formulation for the EDGP.

Theorem 1 *There exists a vector $y^* \in \mathbb{R}^{Km}$ which satisfies Eq.* (5) *and* (7)*, parametrized on K, G, if and only if (K, G) is a YES instance of the EDGP.*

Proof ($\Leftarrow$) 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. (5). Moreover, any realization of G satisfies Eq. (4) over each cycle by Lemma 1. Hence, by replacement, it also satisfies Eq. (7).

($\Rightarrow$) Assume next that (K, G) is a NO instance of the EDGP, and suppose that Eq. (5) and (7) have a non-empty feasible set Y. For every $y \in Y$ we consider the K linear systems

$$\forall k \leq K \qquad \forall \{i, j\} \in E \quad x_{ik} - x_{jk} = y_{ijk}, \tag{9}$$

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

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

By Eq. (5) we have $\sum_{k \leq K} y^2_{ijk} = d^2_{ij}$, 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. (9) for any given $y \in Y$.

We first consider the case where G is a tree. In this case, for each fixed $k \leq K$ system Eq. (9) has n vertices and $n - 1$ edges. Let A be the set of vertices incident to a single edge and B the set of vertices incident to two edges (clearly $A \cup B = V$). If $i \in A$ then x_i occurs in a single equation; if $i \in B$ then x_i occurs in exactly two equations. Thus the linear dependence condition $\sum_{\{i,j\} \in E} \lambda_{ijk}(x_{ik} - x_{jk}) = 0$ (†) requires all of the λ_{ijk} involving $i \in A$ to be zero, which implies $j \in A$ too (if $j \in B$ there would be an x_j term left in (†)): this implies $\lambda = 0$, showing that the system has rank $n - 1$. Thus Eq. (9) has uncountably many solutions. This is repeated for every $k \leq K$ to yield a realization of the tree in $\mathbb{R}^K$.

Now we assume WLOG that G is biconnected, since any pendant trees can be easily treated separately as shown above, and proceed by induction on the simple cycles of G. For the base case, we consider a cycle C with corresponding y satisfying Eq. (5) and (7). Since C is a cycle, it has the same number of vertices and edges, say q. This implies that, for any fixed $k \leq K$, Eq. (9) is a linear system of equations $Mx = y$ with a $q \times q$ matrix M as follows:

$$M = \begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & 1 & \ddots & \\ & & & \ddots & -1 \\ -1 & & & & 1 \end{pmatrix}.$$

By Eq. (4) and by inspection it is clear that the rank of M is exactly $q-1$: then Eq. (7) ensures that Eq. (9) has uncountably many solutions. Repeating this for every $k \le K$ we obtain a realization x of C with K degrees of freedom.

Since any cycle basis generates the set of all cycles in a graph, for the induction step we consider a cycle basis $\mathcal{B}$ of G that is fundamental, see Sect. 3. We assume that G' is a union of fundamental cycles in $\mathcal{B}$ with realization x' satisfying Eq. (9), and that C is another fundamental cycle in $\mathcal{B}$ with realization x^C. We aim at proving that Eq. (9) has a solution for $G' \cup C$. Since G is biconnected, the induction can proceed by ear decomposition [25], 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'. We want to show that C can be realized so the edges in F are realized according to x': we argue that there is $\tilde{x} : V(C) \smallsetminus V(F) \to \mathbb{R}^K$ such that $\bar{x}^C = (x'[V(F)], \tilde{x})$ is a realization of C. It suffices to assume that $E(C) \smallsetminus F$ consists of a single edge, say $\{u, v\}$, since any more edges can be considered as a pendant path attached to G' (easily dealt with as we saw above since paths are trees) and a single edge. This means that $u, v \in V(F)$, i.e. x' already maps $V(G') \cup V(C)$ to $\mathbb{R}^K$. Thus we only need to check that $x'_{uk} - x'_{vk} = y_{uvk}$ for each $k \le K$.

By Eq. (4) applied to $C = F \cup \{\{u, v\}\}$ and the facts that (a) C is a cycle in G and (b) x' realizes G', which contains F, we have

$$\forall k \le K \quad \sum_{\{i,j\} \in C} (x'_{ik} - x'_{jk}) = 0. \tag{10}$$

By Eq. (7) applied to C and the fact that $Y \neq \varnothing$ we have

$$\forall k \le K \quad \sum_{\{i,j\} \in C} y_{ijk} = 0. \tag{11}$$

By induction hypothesis x' satisfies Eq. (9), whence

$$\forall k \le K, \{i, j\} \in F \quad x'_{ik} - x'_{jk} = y_{ijk}. \tag{12}$$

We replace Eq. (12) in Eq. (11), obtaining

$$\forall k \le K \quad \sum_{\{i,j\} \in F} (x'_{ik} - x'_{jk}) = -y_{uvk}. \tag{13}$$

Subtracting Eq. (13) from Eq. (10) finally yields $x'_{uk} - x'_{vk} = y_{uvk}$ for all $k \le K$, which concludes the proof. □

The issue with Thm. (1) is that it relies on the exponentially large family of constraints Eq. (7). 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. (7).

3 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. (7) 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 $m-n+1$ [28].

Our interest in introducing cycle bases is that we would like to quantify Eq. (7) polynomially rather than exponentially in the size of G. Our goal is to replace "C a simple connected cycle in $\mathcal{C}$" by "C in a cycle basis of G". In order to show that this limited quantification is enough to imply every constraint in Eq. (7), we have to show that, for each simple connected cycle $C \in \mathcal{C}$, the corresponding constraint in Eq. (7) can be obtained as a weighted sum of constraints corresponding to the basis elements.

Another feature of Eq. (7) 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. (7) 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 [14]:

$$\forall j \in V(C) \qquad \sum_{(i,j)\in A} c^C_{ij} = \sum_{(j,\ell)\in A} c^C_{j\ell}. \tag{14}$$

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 $i \longrightarrow j$ we let the (i, j)-th component of c^D be 1; if we walk over (i, j) in the direction $j \longrightarrow i$ we assign a -1, and otherwise a zero.

3.1 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 [14]. We now prove that it suffices to quantify Eq. (7) over a directed circuit basis of the cycle space.

Proposition 1 *Let $\mathcal{B}$ be a directed cycle basis of $\bar{G}$ over $\mathbb{Q}$. Then Eq.* (7) *holds if and only if:*

$$\forall k \le K, B \in \mathcal{B} \qquad \sum_{(i,j)\in A(B)} c^B_{ij} y_{ijk} = 0. \tag{15}$$

Proof Necessity (7) ⇒ (15) follows because Eq. (7) 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. (15) are consistent, by definition, with the arbitrary edge direction chosen for $\bar{G}$.

Next, we claim sufficiency (15) ⇒ (7). Let $C \in \mathcal{C}$, 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:

$$c^{\bar{C}} = \sum_{B\in\mathcal{B}} \gamma_B c^B. \tag{16}$$

We now consider the expression:

$$\forall k \le K \quad \sum_{B\in\mathcal{B}} \gamma_B \sum_{(i,j)\in A(B)} c^B_{ij} y_{ijk}. \tag{17}$$

On the one hand, by Eq. (16), Eq. (17) is identically equal to $\sum_{(i,j)\in A(\bar{C})} c^{\bar{C}}_{ij} y_{ijk}$ for each $k \le K$; on the other hand, each inner sum in Eq. (17) is equal to zero by Eq. (15). This implies $\sum_{(i,j)\in A(\bar{C})} c^{\bar{C}}_{ij} y_{ijk} = 0$ for each $k \le K$. Since C is simple and connected $\bar{C}$ is a directed circuit, which implies that $c^{\bar{C}} \in \{-1, 0, 1\}$. Now it suffices to replace $-y_{ijk}$ with y_{jik} to obtain

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

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. (15) 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:

$$\left.\begin{array}{rl} \min\limits_{s\geq 0,y} & \sum\limits_{\{i,j\}\in E} (s_{ij}^+ + s_{ij}^-) \\ \forall (i,j) \in A(\bar{G}) & \sum\limits_{k\leq K} y_{ijk}^2 - d_{ij}^2 = s_{ij}^+ - s_{ij}^- \\ \forall k \leq K, B \in \mathcal{B} & \sum\limits_{(i,j)\in A(B)} c_{ij}^B y_{ijk} = 0 \end{array}\right\} \tag{18}$$

is a valid formulation for the EDGP. The solution of Eq. (18) yields a feasible vector y^*. We must then exploit Eq. (9) to obtain a realization x^* for G.

3.2 How to find directed cycle bases

We require directed cycle bases over $\mathbb{Q}$. By [14, 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 [12] and its variants [11, 24] find a minimum cost cycle basis in polynomial time. The most efficient deterministic variant is $O(m^3n)$ [24], 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 [26] yields a *fundamental* cycle basis (FCB). Finding the minimum FCB is known to be **NP**-hard [9], but heuristics based on spanning trees prove to be very easy to implement and work reasonably well [9] (optionally, their cost can be improved by an edge-swapping phase [1, 17]).

4 Computational results

The aim of this section is to compare the computational performance of the new "cycle formulation" Eqns. (18) and (9) with the standard "edge formulation" Eq. (3). We note that both formulations are nonconvex Nonlinear Programs (NLP), which are generally hard to solve. We therefore used a very simple 3-iteration multi-start 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.

We remark that we added the centroid constraints:

$$\forall k \le K \quad \sum_{i \le n} x_{ik} = 0$$

to the edge formulation Eq. (3). In our experience, these constraints (which simply remove the degrees of translation freedom) give a slight stability advantage to the edge formulation when solved with most local NLP solvers.

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

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

We remark that these realization quality measures are formally different from the objective functions of the formulations we benchmarked.

The CPU time taken to find the solution may also be important, depending on the application. In real-time control of underwater vehicles [3], for example, DGP instances might need to be solved every second. In other applications, such as finding protein structure from distance data [5], 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. We used AMPL [10] to implement our formulations and solution algorithms, and the local NLP IpOpt solver [6] to solve each formulation locally.

Our first benchmark contains a diverse collection of randomly generated weighted graphs of small size and many different types (Table 2), realized in $\mathbb{R}^2$. The cycle formulation finds better MDE values, while the edge formulation generally finds better LDE values and is faster. The instance names in Table 2 label the graph type and some random generation parameters: `almostreg`-k-n are almost k-regular graphs on n vertices, `bipartite`-n-p are bipartite graphs on $2n$ vertices with edge density p, `cluster`-n-k-p-q are k-clustered n-graphs with intercluster density p and intracluster density q, `euclid`-n-p are graphs on n random points in the plane with density p, `flowersnark`-n are flower snark graphs [13] of order n, `hypercube`-n are graphs on 2^n vertices connected with a hypercube topology, `powerlaw`-n-t-a are $\mathsf{deg}_i = ani^{-t}$ power law graphs on n vertices with biconnectedness guaranteed by the addition of a Hamiltonian cycle, `random`-n-p are Erdős-Renyi graphs on n vertices with density p, `rnddegdist`-n are biconnected random graphs on n vertices with a randomly generated degree distribution, `tripartite`-n-p are tripartite graphs on $3n$ vertices with edge density p.

Our second benchmark contains medium to large scale protein graph instances (Table 1), realized in $\mathbb{R}^3$. It turns out that the cycle formulation gives generally better quality solutions (the MDE is better on all instances but two, the LDE is better a little less than half of the times), but takes more time in order to find them. In our largest tested instance (`il2`) the trend is reversed, meaning that the cycle formulation found a bad quality solution but in a tenth of the time.

Instance	*m*	*n*	mdeC	mdeE	ldeC	ldeE	cpuC	cpuE
1guu	955	150	**0.057**	0.061	1.913	**1.884**	**18.18**	37.14
1guu-1	959	150	**0.035**	0.038	2.025	**1.824**	24.27	**5.48**
1guu-4000	968	150	0.061	**0.060**	2.324	**2.121**	24.24	**6.97**
pept	999	107	**0.104**	0.161	3.367	**2.963**	34.67	**10.89**
2kxa	2711	177	**0.053**	0.155	**3.613**	3.936	169.95	**35.44**
res_2kxa	2627	177	0.131	**0.045**	**3.197**	3.442	153.00	**32.40**
C0030pkl	3247	198	**0.009**	0.059	**2.761**	3.965	156.09	**76.58**
cassioli-130731	4871	281	**0.005**	0.060	**3.447**	3.963	376.33	**143.31**
100d	5741	488	**0.146**	0.246	4.295	**4.090**	3024.67	**253.56**
helix_amber	6265	392	**0.038**	0.059	**3.528**	4.578	1573.10	**212.68**
water	11939	648	**0.222**	0.422	4.557	**4.322**	9384.08	**3836.23**
3al1	17417	678	**0.084**	0.124	4.165	**4.087**	4785.91	**1467.74**
1hpv	18512	1629	**0.334**	0.338	**4.256**	4.619	53848.33	**6620.70**
il2	45251	2084	1.481	**0.248**	9.510	**4.415**	**2323.90**	24321.25

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

In all cases, finding the cycle basis and solving the auxiliary retrieval problem Eq. (9) takes a tiny fraction of the total solution time.

Acknowledgements

While the seminal idea for considering DGPs over cycles dates from Saxe's **NP**-hardness proof [27], 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.

References

1. Amaldi, E., Liberti, L., Maffioli, F., Maculan, N.: Edge-swapping algorithms for the minimum fundamental cycle basis problem. Mathematical Methods of Operations Research **69**, 205–223 (2009)
2. Aspnes, J., Eren, T., Goldenberg, D., Morse, S., Whiteley, W., Yang, R., Anderson, B., Belhumeur, P.: A theory of network localization. IEEE Transactions on Mobile Computing **5**(12), 1663–1678 (2006)
3. Bahr, A., Leonard, J., Fallon, M.: Cooperative localization for autonomous underwater vehicles. International Journal of Robotics Research **28**(6), 714–728 (2009)
4. Beeker, N., Gaubert, S., Glusa, C., Liberti, L.: Is the distance geometry problem in NP? In: A. Mucherino, C. Lavor, L. Liberti, N. Maculan (eds.) Distance Geometry: Theory, Methods, and Applications, pp. 85–94. Springer, New York (2013)

5. Cassioli, A., Bordeaux, B., Bouvier, G., Mucherino, A., Alves, R., Liberti, L., Nilges, M., Lavor, C., Malliavin, T.: An algorithm to enumerate all possible protein conformations verifying a set of distance constraints. BMC Bioinformatics **16**, 23–38 (2015)
6. COIN-OR: Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT (2006)
7. Connelly, R.: Generic global rigidity. Discrete Computational Geometry **33**, 549–563 (2005)
8. D'Ambrosio, C., Vu, K., Lavor, C., Liberti, L., Maculan, N.: New error measures and methods for realizing protein graphs from distance data. Discrete and Computational Geometry **57**(2), 371–418 (2017)
9. Deo, N., Prabhu, G., Krishnamoorthy, M.: Algorithms for generating fundamental cycles in a graph. ACM Transactions on Mathematical Software **8**(1), 26–42 (1982)
10. Fourer, R., Gay, D.: The AMPL Book. Duxbury Press, Pacific Grove (2002)
11. Golynski, A., Horton, J.: A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In: 8th Scandinavian Workshop on Algorithm Theory (2002)
12. Horton, J.: A polynomial-time algorithm to find the shortest cycle basis of a graph. SIAM Journal of Computing **16**(2), 358–366 (1987)
13. Isaacs, R.: Infinite families of nontrivial trivalent graphs which are not Tait colorable. American Mathematical Monthly **82**(3), 221–239 (1975)
14. Kavitha, T., Liebchen, C., Mehlhorn, K., Michail, D., Rizzi, R., Ueckerdt, T., Zweig, K.: Cycle bases in graphs: characterization, algorithms, complexity, and applications. Computer Science Review **3**, 199–243 (2009)
15. Lavor, C., Liberti, L., Maculan, N.: Molecular distance geometry problem. In: C. Floudas, P. Pardalos (eds.) Encyclopedia of Optimization, second edn., pp. 2305–2311. Springer, New York (2009)
16. Lavor, C., Liberti, L., Maculan, N., Mucherino, A.: Recent advances on the discretizable molecular distance geometry problem. European Journal of Operational Research **219**, 698–706 (2012)
17. Lee, J., Liberti, L.: A matroid view of key theorems for edge-swapping algorithms. Mathematical Methods of Operations Research **76**, 125–127 (2012)
18. Liberti, L., Lavor, C.: Euclidean Distance Geometry: An Introduction. Springer, New York (2017)
19. Liberti, L., Lavor, C.: Open research areas in distance geometry. In: A. Migalas, P. Pardalos (eds.) Open Problems in Optimization and Data Analysis, *SOIA*, vol. 141, pp. 183–223. Springer, New York (2018)
20. Liberti, L., Lavor, C., Alencar, J., Abud, G.: Counting the number of solutions of ^{k}DMDGP instances. In: F. Nielsen, F. Barbaresco (eds.) Geometric Science of Information, *LNCS*, vol. 8085, pp. 224–230. Springer, New York (2013)
21. Liberti, L., Lavor, C., Maculan, N.: A branch-and-prune algorithm for the molecular distance geometry problem. International Transactions in Operational Research **15**, 1–17 (2008)
22. Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. SIAM Review **56**(1), 3–69 (2014)
23. Liberti, L., Lavor, C., Mucherino, A., Maculan, N.: Molecular distance geometry methods: from continuous to discrete. International Transactions in Operational Research **18**, 33–51 (2010)
24. Liebchen, C., Rizzi, R.: A greedy approach to compute a minimum cycle basis of a directed graph. Information Processing Letters **94**, 107–112 (2005)
25. Lovász, L., Plummer, M.: On minimal elementary bipartite graphs. Journal of Combinatorial Theory B **23**, 127–138 (1977)
26. Paton, K.: An algorithm for finding a fundamental set of cycles of a graph. Communications of the ACM **12**(9), 514–518 (1969)
27. Saxe, J.: Embeddability of weighted graphs in k-space is strongly NP-hard. Proceedings of 17th Allerton Conference in Communications, Control and Computing pp. 480–489 (1979)
28. Seshu, S., Reed, M.: Linear Graphs and Electrical Networks. Addison-Wesley, Reading, MA (1961)
29. Singer, A.: Angular synchronization by eigenvectors and semidefinite programming. Applied and Computational Harmonic Analysis **30**, 20–36 (2011)

Instance	*m*	*n*	mdeC	mdeE	ldeC	ldeE	cpuC	cpuE
`almostreg-3-100`	298	100	**0**	**0**	0.048	**0.041**	0.88	**0.23**
`almostreg-3-150`	448	150	**0**	**0**	0.330	**0.282**	1.29	**0.30**
`almostreg-3-200`	598	200	**0**	**0**	0.030	**0.020**	2.15	**0.44**
`almostreg-3-50`	146	50	**0**	**0**	**0**	**0**	0.31	**0.11**
`almostreg-6-100`	591	100	**0.077**	0.093	0.740	**0.410**	6.85	**0.35**
`almostreg-6-150`	893	150	**0.085**	0.099	1.030	**0.485**	16.52	**0.68**
`almostreg-6-200`	1192	200	**0.076**	0.098	0.729	**0.501**	34.07	**1.35**
`almostreg-6-50`	292	50	**0.082**	0.099	0.648	**0.471**	1.80	**0.13**
`almostreg-8-100`	777	100	**0.105**	0.131	0.846	**0.577**	8.89	**0.42**
`almostreg-8-150`	1189	150	**0.104**	0.121	0.805	**0.528**	34.84	**0.83**
`almostreg-8-200`	1581	200	**0.104**	0.125	0.974	**0.654**	48.10	**1.79**
`almostreg-8-50`	387	50	**0.104**	0.113	0.670	**0.520**	2.46	**0.13**
`bipartite-100-03`	3044	200	**0.206**	0.218	0.931	**0.790**	209.15	**7.86**
`bipartite-100-06`	6024	200	**0.225**	0.234	0.978	**0.753**	439.74	**8.00**
`bipartite-150-03`	6708	300	**0.220**	0.232	0.951	**0.724**	582.71	**14.37**
`bipartite-150-06`	13466	300	**0.231**	0.240	0.852	**0.808**	1904.18	**30.79**
`bipartite-200-03`	11906	400	**0.223**	0.235	0.936	**0.812**	3183.43	**33.06**
`bipartite-200-06`	23963	400	**0.235**	0.244	0.888	**0.741**	4885.52	**64.03**
`bipartite-50-03`	744	100	**0.166**	0.185	0.936	**0.787**	29.27	**1.11**
`bipartite-50-06`	1468	100	**0.201**	0.217	1.011	**0.754**	80.80	**1.38**
`cluster-120-4-05-01`	1495	120	**0.191**	0.206	0.873	**0.838**	98.67	**1.69**
`cluster-120-8-05-01`	1149	120	**0.181**	0.196	0.892	**0.740**	62.29	**1.04**
`cluster-150-2-05-01`	3337	150	**0.218**	0.230	**0.901**	0.936	605.00	**3.66**
`cluster-150-8-05-01`	1750	150	**0.190**	0.205	0.886	**0.831**	70.66	**2.44**
`cluster-200-2-05-01`	5957	200	**0.231**	0.241	**0.931**	0.952	612.82	**8.01**
`cluster-200-4-05-01`	4155	200	**0.221**	0.233	0.924	**0.906**	397.45	**7.67**
`cluster-200-8-05-01`	3046	200	**0.206**	0.220	0.988	**0.851**	462.46	**5.61**
`cluster-50-2-05-01`	361	50	**0.159**	0.171	0.742	**0.679**	7.52	**0.20**
`cluster-50-4-05-01`	242	50	**0.145**	0.167	0.899	**0.588**	3.63	**0.18**
`cluster-50-8-05-01`	187	50	**0.113**	0.133	0.716	**0.500**	2.73	**0.16**
`euclid-150-02`	2341	150	**0**	**0**	**0**	**0**	286.09	**2.69**
`euclid-150-05`	5678	150	**0**	**0**	**0**	**0**	991.87	**2.86**
`euclid-150-08`	8915	150	**0**	**0**	**0**	**0**	1507.94	**3.88**
`euclid-200-05`	10037	200	**0**	**0**	**0**	**0**	1881.40	**5.47**
`euclid-200-08`	15877	200	**0**	**0**	**0**	**0**	3114.95	**7.96**
`flowersnark120`	720	480	**0**	**0**	0.151	**0.109**	**7.86**	8.21
`flowersnark-150`	900	600	**0**	**0**	0.101	**0.086**	36.53	**15.50**
`flowersnark-200`	1200	800	**0**	**0**	0.141	**0.123**	**18.02**	31.04
`flowersnark40`	240	160	**0**	**0**	0.016	**0.005**	1.92	**0.35**
`flowersnark80`	480	320	**0**	**0**	0.068	**0.059**	3.18	**1.08**
`hypercube-10`	5120	1024	**0.128**	0.152	1.004	**0.653**	4965.30	**133.93**
`hypercube-5`	80	32	**0.054**	0.058	0.401	**0.321**	0.95	**0.10**
`hypercube-6`	192	64	**0.075**	0.087	0.774	**0.426**	4.20	**0.20**
`hypercube-8`	1024	256	**0.104**	0.127	0.876	**0.631**	81.68	**2.59**
`powerlaw-100-2-05`	148	100	**0.024**	0.025	0.338	**0.309**	1.24	**0.38**
`powerlaw-100-2-08`	178	100	**0.042**	**0.042**	0.464	**0.398**	1.64	**0.59**
`powerlaw-150-2-05`	223	150	**0.034**	0.035	0.404	**0.360**	**1.37**	1.94
`powerlaw-150-2-08`	268	150	**0.047**	**0.047**	0.471	**0.404**	2.44	**1.73**
`powerlaw-200-2-05`	298	200	**0.025**	0.026	0.581	**0.443**	2.64	**1.27**
`powerlaw-200-2-08`	358	200	**0.037**	0.038	0.454	**0.376**	3.75	**1.78**
`random-100-02`	1093	100	**0.193**	0.203	0.874	**0.742**	48.43	**0.67**
`random-100-05`	2479	100	**0.224**	0.234	0.938	**0.855**	168.40	**1.48**
`random-150-02`	2394	150	**0.209**	0.223	0.932	**0.809**	226.60	**3.98**
`random-150-05`	5675	150	**0.241**	0.250	0.965	**0.953**	580.59	**6.10**
`random-200-02`	4097	200	**0.218**	0.228	0.930	**0.887**	271.94	**7.68**
`random-200-05`	10023	200	**0.248**	0.255	**0.949**	0.952	1024.32	**11.43**
`random-50-02`	291	50	**0.143**	0.161	0.922	**0.638**	7.03	**0.17**
`random-50-05`	665	50	**0.195**	0.212	**0.836**	0.953	16.20	**0.23**
`rnddegdist-100`	2252	100	**0.223**	0.235	**0.929**	0.963	136.74	**1.48**
`rnddegdist-150`	5293	150	**0.240**	0.249	**0.939**	0.955	819.86	**3.91**
`rnddegdist-30`	174	30	**0.156**	0.179	0.767	**0.667**	2.26	**0.11**
`rnddegdist-40`	221	40	**0.156**	0.175	0.672	**0.628**	2.93	**0.17**
`tripartite-100-02`	4038	300	**0.198**	0.213	0.968	**0.737**	369.77	**10.39**
`tripartite-100-05`	10003	300	**0.227**	0.238	0.917	**0.729**	1150.35	**21.37**
`tripartite-150-02`	9061	450	**0.213**	0.227	0.956	**0.765**	2005.30	**32.43**
`tripartite-150-05`	22431	450	**0.235**	0.245	0.876	**0.751**	4687.28	**45.27**
`tripartite-30-02`	359	90	**0.106**	0.118	0.736	**0.547**	10.31	**0.37**
`tripartite-50-02`	995	150	**0.153**	0.173	0.958	**0.722**	38.55	**1.00**
`tripartite-50-05`	2519	150	**0.208**	0.220	0.849	**0.736**	160.43	**2.39**

Table 2 Cycle formulation vs. edge formulation performance on various small sized graphs (realizations in $K = 2$ dimensions.