

## Chapter 2

# The Distance Geometry Problem

The Distance Geometry Problem (DGP) is an *inverse problem*. The corresponding “direct problem” is to compute some pairwise distances of a given set of points. Whereas the direct problem is obviously trivial (just carry out the computation), the inverse problem is generally difficult to solve.

### 2.1 Computing all pairwise distances from points

Consider these four points in  $\mathbb{R}^2$ :  $A = (4, 0)$ ,  $B = (0, 3)$ ,  $C = (-4, 0)$ ,  $D = (0, -3)$ , as shown in Fig. 2.1. Their pairwise distances are easy to compute (see Fig. 2.2):

$$\overline{AB} = 5, \overline{AC} = 8, \overline{AD} = 5, \overline{BC} = 5, \overline{BD} = 6, \overline{CD} = 5. \quad (2.1)$$

### 2.2 Computing points from all pairwise distances

Now, consider the inverse problem: you are given all the distances as in (2.1), with each value assigned to a pair of point names, you are told that they refer to a 2D vector space, and you are asked to compute four points  $A, B, C, D$  in  $\mathbb{R}^2$  that give rise to those distances.

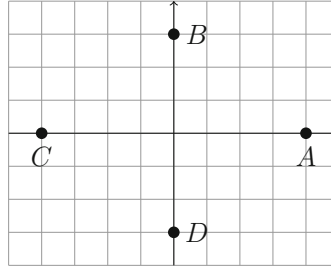
#### 2.2.1 Ill-posedness

Notice that the inverse problem is ill-defined: it may have one or more solutions or even have no solution at all. For example, the points  $A = (5, 1)$ ,  $B = (1, 4)$ ,  $C = (-3, 1)$ ,  $D = (1, -2)$  also yield the same distances as  $A = (4, 0)$ ,  $B = (0, 3)$ ,  $C = (-4, 0)$ ,  $D = (0, -3)$  (see Fig. 2.3). And so do all the infinitely many points

$$A = (4 + \alpha, 0 + \beta), B = (0 + \alpha, 3 + \beta), C = (-4 + \alpha, 0 + \beta), D = (0 + \alpha, -3 + \beta)$$

for any scalars  $\alpha, \beta \in \mathbb{R}$ .

We can also rotate the point configuration in Fig. 2.1 by any given angle  $\vartheta$  around any given center  $c$ , obtaining four different points always yielding the same distances as in (2.1).



**Fig. 2.1** Configuration of four points in  $\mathbb{R}^2$ .

```
x = {{4,0},{0,3},{-4,0},{0,-3}}; MatrixForm[EuclideanDistanceMatrix[x]]
```

$$\begin{pmatrix} 0 & 5 & 8 & 5 \\ 5 & 0 & 5 & 6 \\ 8 & 5 & 0 & 5 \\ 5 & 6 & 5 & 0 \end{pmatrix}$$

**Fig. 2.2** Computing a distance matrix from an array of 2D points.

```
x = {{4,0},{0,3},{-4,0},{0,-3}}; Dx = EuclideanDistanceMatrix[x];
y = Map[(# + {1,1})&, x]; Dy = EuclideanDistanceMatrix[y];
PartialEDMError[Dx,Dy]
```

0

**Fig. 2.3** Translations preserve pairwise distances. The `Map` function applies the function in the first argument  $((\# + \{1, 1\}) \&)$  to each of the elements of the list  $(x)$  in the second argument. In general, if  $f$  is a function and  $x = (x_1, \dots, x_n)$  is a list, then `Map`[( $f$  [ $\#$ ])  $\&$ ,  $x$ ] is the list  $(f(x_1), \dots, f(x_n))$ .

### 2.2.2 No solution

If we modify (2.1) so that  $\overline{AC} = 11$ , then the inverse problem has no solution at all: since  $\overline{AB} = \overline{BC} = 5$  but  $\overline{AC} = 11$ , and  $5 + 5$  is *not* greater than 11, these values contradict the triangle inequality.<sup>1</sup> This means that, no matter how hard we try, we shall never find three points  $A, B, C$  having those values as pairwise distances.

Ill-definedness notwithstanding, we shall see that it is possible to efficiently compute the correct point sets when all pairwise distances are given.

<sup>1</sup>See the metric axiom (Axiom 3) in Appendix A.6.

## 2.3 The fundamental problem of DG

In real problems, it very rarely happens that we are given all pairwise distances. The fundamental problem of DG is as follows: is there a set of points in a vector space of given dimension which yields a given subset pairwise distances, each with its assigned point name pair?

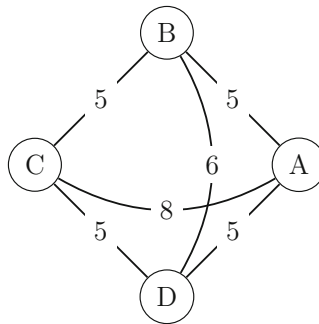
### 2.3.1 The input as a weighted graph

We formalize this problem using graphs to encode point names, their pairs, and the corresponding distances (see Fig. 2.4). Each point name is a vertex of the graph, and pairs of vertices are edges. Each edge is assigned to a distance value. This yields a weighted graph  $G = (V, E, d)$  where  $V$  is the set of vertices of the graph,  $E$  is the set of edges, and  $d$  is the edge weight function. Graphs which are inputs to DG problems are generally *simple*, i.e., they do not have loops or parallel edges (see Appendix A.8.2). The first metric axiom (see Appendix A.6) states that the distance between a point and itself is zero; so it would not make much sense to have loops in the graph. Also, since any metric is a function, it must be well-defined; it must map any given argument to a unique value. So, if we had parallel edges between two vertices, we would have to assign the same distance value to all of them. But then we can simply replace all parallel edges with just one edge, without loss of generality. Because of these two observations, we can require the graph  $G$  to be simple.

We can also assume the input graph to be connected; in case it is not, then the graph really consists of a set of different subgraphs, each of which is connected. In order to solve the DGP on a disconnected graph, it suffices to solve it separately for each connected component. This reduces the DGP on disconnected graphs to a set of DGPs on connected graphs, showing that the connectivity assumption on  $G$  yields no loss of generality.

### 2.3.2 Formalization of the DGP

A *realization* is a function which maps a set of vertices to a Euclidean space of some given dimension.



**Fig. 2.4** Graph encoding the distances (2.1).

```
x = {{4,0},{0,3},{-4,0},{0,-3}}; {MatrixForm[x], MatrixForm[Transpose[x]]}
```

$$\left\{ \begin{pmatrix} 4 & 0 \\ 0 & 3 \\ -4 & 0 \\ 0 & -3 \end{pmatrix}, \begin{pmatrix} 4 & 0 & -4 & 0 \\ 0 & 3 & 0 & -3 \end{pmatrix} \right\}$$

**Fig. 2.5** Since we wish  $x_i$  to be the column vector representing the  $i$ th point of the realization  $x$ , in *Mathematica* we write  $x$  as a list of lists: each of the  $n$  elements of  $x$  is itself a list with  $K$  elements, as shown in the code above. *Mathematica*, however, encodes matrices by row:  $x$  is shown (on the left) as an  $n \times K$  matrix; to make sure matrix computations are dimensionally correct, we have to either multiply on the right, as in the product `(SalesTable . Proj3D` in Fig. 1.3), or use the transpose, as above.

**DISTANCE GEOMETRY PROBLEM (DGP).** Given a simple, connected, weighted graph  $G = (V, E, d)$  and an integer  $K > 0$ , is there a realization  $x : V \rightarrow \mathbb{R}^K$  such that:

$$\forall \{u, v\} \in E \quad \|x(u) - x(v)\| = d_{uv} ? \quad (2.2)$$

Since a realization certifies a positive answer to a DGP problem instance, realizations are sometimes also called “solutions” or “certificates.”

We usually write  $x_u, x_v$  instead of  $x(u), x(v)$  and limit our attention to the Euclidean distance  $\|x_u - x_v\|_2$ . Moreover, since each  $x_v$  is in  $\mathbb{R}^K$  for any  $v \in V$ ,  $x$  can in fact be written as a rectangular  $K \times n$  array:

$$x = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{K1} & \cdots & x_{Kn} \end{pmatrix},$$

where  $n = |V|$ , and the  $v$ th column is the  $K$ -vector  $x_v$  (see Fig. 2.5). With a slight abuse of notation, we often write  $x \in \mathbb{R}^{Kn}$  to mean that  $x$  is a  $K \times n$  matrix. This means that elements of  $\mathbb{R}^{Kn}$  are not to be seen as linear sequences of  $Kn$  elements; as a consequence, the rank of  $x$  is to be considered as varying between 0 and  $K$  rather than between 0 and 1.

## 2.4 A quadratic system of equations

If  $\|\cdot\|$  is the Euclidean norm, then Eq. (2.2) becomes<sup>2</sup>:

$$\forall \{u, v\} \in E \quad \sqrt{\sum_{k=1}^K (x_{uk} - x_{vk})^2} = d_{uv}. \quad (2.3)$$

The fourth metric axiom (see Appendix A.6) states that distances are always nonnegative, so we can

<sup>2</sup>The symbol  $\sum_{k=1}^K$  stands for  $\sum_{k=1}^K$ .

square both sides of Eq. (2.3) and obtain:

$$\forall \{u, v\} \in E \quad \sum_{k \leq K} (x_{uk} - x_{vk})^2 = d_{uv}^2, \quad (2.4)$$

which is a multivariate polynomial system of equations of degree 2.

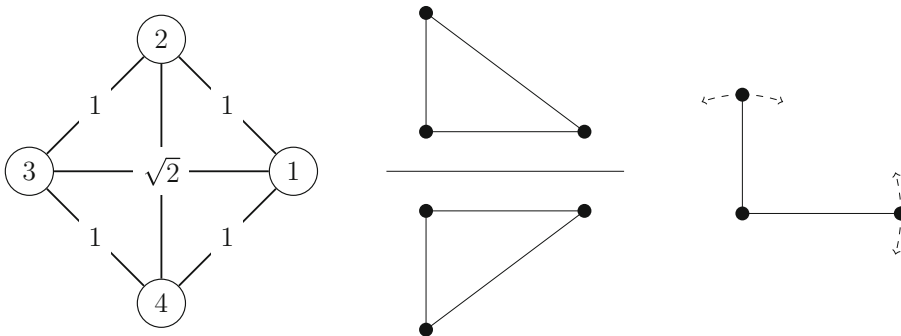
### 2.4.1 The number of solutions

Any realization  $x$  satisfying Eq. (2.3) can be translated and rotated in  $\mathbb{R}^K$  without any change to the pairwise distances, implying that the cardinality of the solution set is not only infinite, but uncountably so (see Appendix A.1).

However, what if we were to discount the effect of translations and rotations? How many distinct realizations would a given weighted graph have in  $\mathbb{R}^K$  then? The answer depends on the structure of the graph, as well as on the edge weight function: there could be no realization, a unique realization, finitely many, or uncountably many (see Fig. 2.6). A result in real algebraic geometry [10] precludes the only other possibility, i.e., a countable infinity of realizations.

#### 2.4.1 Example

Consider the complete graph shown in Fig. 2.6, left: there is no way to move a vertex while at least two other vertices remain fixed. In other words, aside from rotating or translating the whole drawing, there is a unique realization of the given graph (note that reflecting the drawing with respect to the axis given by the two fixed vertices yields a drawing which can be superposed on the original drawing by rotations and translations—only with two swapped vertex labels). Consider now the triangle graph given in Fig. 2.6, above middle: keeping two vertices fixed does not fix the third one, which can be reflected across the fixed edge, as shown in Fig. 2.6, below middle, yielding a new drawing which cannot be superposed on the original drawing using rotations and translations only.



**Fig. 2.6** In  $\mathbb{R}^2$ : a unique solution (left), two solutions (center), and uncountably many solutions (right). Finally, consider the graph in this figure, right the arrows show the directions of the continuous movements which the corresponding vertices can make without changing any of the given edge lengths, and while keeping the other two vertices fixed.

### 2.4.2 Computational complexity of the DGP

The DGP is **NP**-hard [107] (see Appendix A.9). The proof works as follows: we pick another **NP**-hard problem  $P$ , and we show that there is a polynomial transformation (also known as *reduction*, see Sect. A.9.6) from  $P$  to DGP such that  $P$  is a YES instance if and only if the reduced DGP instance is YES. This implies that if we could solve DGP, then  $P$  could be solved by means of a polynomial number of calls to a solution algorithm for DGP, which is another way of saying that DGP is at least as hard as  $P$  modulo a polynomial amount of computational effort. Since  $P$  is assumed to be **NP**-hard, then DGP must also be in the same complexity class. Saxe's proof reduces the PARTITION problem to the DGP for  $K = 1$ .

PARTITION. Given a sequence  $A = (a_1, \dots, a_n)$  of nonnegative integers, is there a subset  $I \subseteq \{1, \dots, n\}$  such that  $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$ ?

This is the first nontrivial proof we present. The reader should be familiar with basic notions of computational complexity, such as reduction and hardness. Read Appendix A.9 before moving on.

For a given instance  $(a_1, \dots, a_n)$  of PARTITION, consider a simple cycle  $C = (V, E)$  with  $|V| = |E| = n$  and an edge weight function  $d$  such that  $d_{i,i+1} = a_i$  for  $i \leq n-1$ , and  $d_{n1} = a_n$ : by arbitrarily setting  $K = 1$ , we make this weighted cycle graph into a DGP instance.

Assume that the given PARTITION instance is a YES instance: we show that the corresponding DGP instance is also a YES instance, i.e., there is a realization  $x$  of  $C$  in 1D (i.e., on a line). We construct a realization  $x$  of  $C$  in  $\mathbb{R}$  inductively, as follows:

1. we fix  $x_1 = 0$ ;
2. if we know the position  $x_i$  for  $i < n$  and  $i \in I$ , we let  $x_{i+1} = x_i + d_{i,i+1}$  (right placement), else  $x_{i+1} = x_i - d_{i,i+1}$  (left placement).

Following the same induction on  $i$ , we easily prove that  $x$  is a valid realization for  $C$ ; it suffices to assume it is correct as far as vertex  $i$  and conclude, by the induction hypothesis, that it is also correct for vertex  $i+1$ , since we place  $x_{i+1}$  at distance  $d_{i,i+1}$  from  $i$ . This induction holds as far as we are applying rule 2 above, i.e., until  $i = n-1$ , which yields position for  $i+1 = n$ . In order for the realization to be valid, however, we also need to ensure that the distance  $d_{1n}$  is preserved. To this end, we define a dummy index  $n+1$  to be equivalent to index 1: the position  $x_{n+1}$ , computed according to rule 2 where  $d_{n,n+1} = d_{n1} = d_{1n}$ , should turn out to be equal to  $x_1$ , which we still need to prove.

We have:

$$\begin{aligned}
 \sum_{i \in I} (x_{i+1} - x_i) &= \sum_{i \in I} d_{i,i+1} \quad [\text{by defn. of the cycle graph}] \\
 &= \sum_{i \in I} a_i = \sum_{i \notin I} a_i \quad [\text{Partition instance is YES}] \\
 &= \sum_{i \notin I} d_{i,i+1} = \sum_{i \notin I} (x_i - x_{i+1}).
 \end{aligned}$$

This implies that the first and the last terms are equal, hence:

$$\begin{aligned}
0 &= \sum_{i \in I} (x_{i+1} - x_i) - \sum_{i \notin I} (x_i - x_{i+1}) \\
&= \sum_{i \in I} (x_{i+1} - x_i) + \sum_{i \notin I} (x_{i+1} - x_i) \quad [\text{sign change in 2nd term}] \\
&= \sum_{i \leq n} (x_{i+1} - x_i) \quad [\text{grouping terms from both sums}] \\
&= (x_{n+1} - x_n) + (x_n - x_{n-1}) + \dots + (x_2 - x_1) \\
&= x_{n+1} + (x_n - x_n) + \dots + (x_2 - x_2) - x_1 = x_{n+1} - x_1,
\end{aligned}$$

implying  $x_{n+1} = x_1$  as claimed. So the DGP instance is YES.

Now, assume that the given PARTITION instance is a NO instance, and suppose, to get a contradiction, that the corresponding DGP instance is a YES instance. So we suppose there is a realization  $x$  of  $C$  in 1D. Since we are realizing on a line, for any two points  $x_u, x_v$ , we either have  $x_u \leq x_v$  or  $x_u > x_v$ . Let  $F = \{\{u, v\} \in E \mid x_u \leq x_v\}$ , so that  $E \setminus F$  will only contain edges  $\{u, v\}$  for which  $x_u > x_v$ . Because  $C$  is a cycle, starting with any vertex  $v$ , we must be able to walk the cycle from  $v$  back to itself passing through every vertex; for this to hold, the walk must have one direction over all edges in  $F$  and the opposite direction for all edges in  $E \setminus F$ . Since there is a unique position  $x_v$  for each vertex  $v$ , the distance walked in one direction must be equal to the distance walked in the opposite direction. Hence:

$$\begin{aligned}
&\sum_{\{u,v\} \in F} (x_v - x_u) = \sum_{\{u,v\} \in E \setminus F} (x_u - x_v) \\
\Rightarrow \quad &\sum_{\{u,v\} \in F} |x_u - x_v| = \sum_{\{u,v\} \in E \setminus F} |x_u - x_v| \\
\Rightarrow \quad &\sum_{\{u,v\} \in F} d_{uv} = \sum_{\{u,v\} \in E \setminus F} d_{uv}. \tag{2.5}
\end{aligned}$$

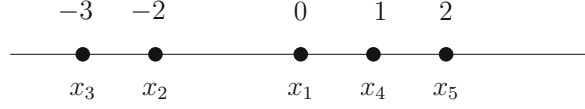
By definition, every edge in  $E$  has the form  $\{i, i+1\}$  for  $i < n$  or  $\{n, 1\}$ : let  $J$  be the set of all  $i < n$  such that  $\{i, i+1\}$  is in  $F$ , and let it also contain  $n$  if  $\{n, 1\}$  is in  $F$ . Then, Eq. (2.5) becomes:

$$\sum_{i \in J} a_i = \sum_{i \notin J} a_i,$$

which implies that  $J$  is a solution for the given PARTITION instance, against the assumption that it was a NO instance. Hence, the corresponding DGP instance must also be a NO instance, as claimed.

Lastly, it is easy to note that the transformation of a PARTITION instance into the corresponding DGP instance can be carried out in time bounded by a polynomial in  $n$ , since for each  $i \leq n$  we construct a vertex and an edge in the cycle graph.

This means that we have a polynomial-time (also known as *polytime*) transformation to turn PARTITION instances to DGP instances with  $K = 1$  so that YES instances map to YES instances and NO instances map to NO instances. In other words, if we could solve the DGP in polytime, then we could exploit this polytime transformation to solve PARTITION in polytime too. But since PARTITION is **NP**-hard [51], then DGP must also be **NP**-hard with  $K = 1$ . And since the case  $K = 1$  determines a subset of instances of the DGP, the DGP itself must be **NP**-hard.



**Fig. 2.7** Realization of a cycle in 1D.

### 2.4.2 Example

Consider the PARTITION instance  $a_1 = 2, a_2 = 1, a_3 = 4, a_4 = 1, a_5 = 2$ . We construct the cycle  $C$  over  $\{1, 2, 3, 4, 5\}$  with edges  $\{i, i+1\}$  for  $i \leq 4$  and  $\{5, 1\}$  (which closes the cycle), weighted by  $a_i$  for each  $i \leq 4$  and  $d_{51} = a_5$ . We realize  $C$  with  $x_1 = 0, x_2 = -2, x_3 = -3, x_4 = 1, x_5 = 2$ , as shown in Fig. 2.7. Now, the set  $F$  of edges  $\{u, v\}$  with  $u < v$  and  $x_u \leq x_v$  is  $\{3, 4\}$  and  $\{4, 5\}$ , so  $J = \{3, 4\}$ , and it is easy to verify that  $\sum_{i \in J} a_i = 4 + 1 = 5 = 2 + 1 + 2 = \sum_{i \notin J} a_i$ , showing that  $(a_3, a_4)$  and  $(a_1, a_2, a_5)$  is the desired partition.

Saxe also proved that the DGP is **NP**-hard for any fixed value of  $K$ , using a more complicated reduction from a different **NP**-hard problem.

## 2.5 Direct solution methods

The simplest approach to find the set of solutions of Eq. (2.4) is to attempt to solve the system of equations directly. In general, and for given  $K \geq 2$ , there is evidence that a closed-form solution where every component of  $x$  is expressed by radicals (i.e., using integers combined with sum, difference, product, fraction, power, and roots) is not possible [6, 9].

Numerically, Eq. (2.4) is difficult to solve directly.

### 2.5.1 A global optimization formulation

Instead, we can formulate the problem as a global optimization (GO) problem, where we minimize the sum of the errors over all the equations:

$$\min_{x \in \mathbb{R}^{Kn}} \sum_{\{u,v\} \in E} \left( \sum_{k \leq K} (x_{uk} - x_{vk})^2 - d_{uv}^2 \right)^2. \quad (2.6)$$

If the global optimum  $x^*$  has value zero, then every term in the sum has error zero, which implies that  $x^*$  is a realization of the given weighted graph  $G = (V, E, d)$ . Conversely, if  $x^*$  is a realization of  $G$ , then Eq. (2.6) has value zero. Moreover, Eq. (2.6) being a sum of nonnegative values, it cannot be negative itself. From all this, we conclude that this formulation of the problem identifies a realization if and only if the global optimum has value zero.

What if  $G$  cannot be realized in  $\mathbb{R}^K$ ? Then, the global optimum  $x^*$  of Eq. (2.6) will yield a function value strictly greater than zero. From a geometrical point of view,  $x^*$  provides the smallest change to the edge weighting of  $G$  that does yield a realization.

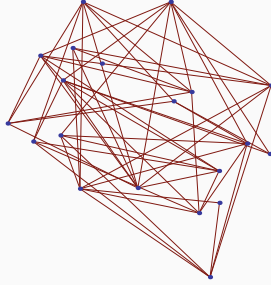
Equation (2.6) aims at minimizing a multivariate polynomial of degree four. It turns out that solving such problems is hard at a computational complexity point of view, as well as from a practical one.



```

y = Map[(DGPSystemApproxGlobal[G,2,#])&,
  {"NelderMead","DifferentialEvolution","SimulatedAnnealing","RandomSearch"}];
t = Map[(RelativePartialEDMError[A, EuclideanDistanceMatrix[y[[#]]]]&,Range[4]);
i = Flatten[Position[t,Min[t]]][[1]];
GraphPlot[G, VertexCoordinateRules->y[[i]] ]

```



**Fig. 2.8**  $G$  is a random graph created as in Fig. 2.9. We solve the corresponding DGP by means of Eq. (2.6) using each of the four GO methods in *Mathematica*, then evaluate their mean relative per-distance error with respect to the distances in  $G$ , pick the minimum, and plot the corresponding realization. To understand what the definition of  $i$  does, use *Mathematica* to print out `Min[t]`, `Position[t,Min[t]]`, and `Flatten[Position[t,Min[t]]]`. For any list  $L$ ,  $L[[i]]$  returns its  $i$ th element.

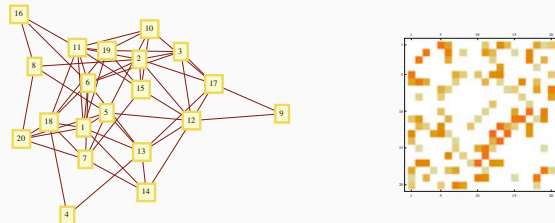
Some general-purpose GO approaches have been tested in [78]. Unfortunately, none of them scales well to medium or large instances. We test some of these methods (Fig. 2.8) on random graphs (Fig. 2.9).

One of the challenges of GO is that many of the existing methods—specially the most efficient ones—do not provide a guarantee of optimality. In other words, we may find a solution  $x'$  of Eq. (2.6) which is not a global optimum, but simply a local one. Such a solution can be considered as an

```

n = 20;
m = Round[N[0.3*n (n - 1)/2]];
G = RandomGraph[{n, m}, EdgeWeight -> RandomReal[{0, 10}, m]];
A = Normal[WeightedAdjacencyMatrix[G]];
GraphPlot[G, VertexLabeling -> True]
MatrixPlot[A]

```



**Fig. 2.9** Creating a random graph with  $n$  vertices and  $m$  edges in *Mathematica*. The pictures show a drawing of the graph in 2D (left, made by *Mathematica*), and the sparsity structure of the distance matrix (right).

approximate realization of  $G$ , the approximation error of which depends on the value of Eq. (2.6) at  $x'$ : the closer to zero, the better the approximation. This does not imply that there is a provable guarantee that the approximation error is bounded by the objective function value; it may happen occasionally that a very small but nonzero objective function value corresponds to a large error in the realization (e.g., if there exist two extremely different realizations having almost exactly the same edge lengths). But, empirically speaking, this is unlikely.

## 2.6 Exercises

### 2.6.1 Exercise

Consider a triangle graph ( $V = \{1, 2, 3\}$ ,  $E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$ ) with unit edge weights  $d_{12} = d_{13} = d_{23} = 1$ . Let the position of vertex  $v = 1$  be the origin. Write down the systems in Eq. (2.4) for  $K \in \{1, 2, 3\}$ , and solve these systems, either algebraically or numerically (e.g., using *Mathematica*). How many solutions are there for  $K = 1$ ? For  $K = 2$ ? And for  $K = 3$ ?

### 2.6.2 Exercise

Consider two disjoint triangle graphs with unit edge weights, with the position of vertex  $v = 1$  fixed at the origin. Write down the system in Eq. (2.4) for  $K = 2$  and provide a solution. How many solutions are there? Now, fix another vertex position, say  $v = 2$ : how many solutions?

### 2.6.3 Exercise

Consider two disjoint triangle graphs with unit edge weights sharing a vertex, as in Fig. 2.10. Write down the system in Eq. (2.4) for  $K = 2$  and provide a solution. How many solutions are there? Now, by fixing the position of vertex  $v = 3$ , how many solutions? And finally by fixing the position of vertex  $v = 5$ , how many solutions?

### 2.6.4 Exercise

Prove (from first principles) that a simple cycle on  $n$  vertices has  $n$  edges.

### 2.6.5 Exercise

Prove that any simple cycle can be decomposed into two simple paths with no common edges. How many such decompositions are there?

### 2.6.6 Exercise

Consider the procedure used in Sect. 2.4.2 to reduce a PARTITION instance into a DGP one. Apply it to the following PARTITION instances: (1, 1, 1, 3, 2) and (1, 2, 3, 4, 5, 6). Derive corresponding DGP instances in 1D and solve them. Use the solutions to decide whether the original PARTITION instances are YES or NO.

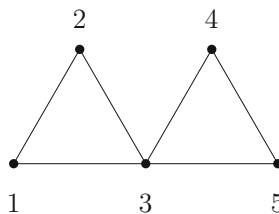


Fig. 2.10 Graph consisting of two triangle graphs sharing a vertex.

Euclidean Distance Geometry

An Introduction

Liberti, L.; Lavor, C.

2017, XIII, 133 p. 60 illus., 31 illus. in color., Hardcover

ISBN: 978-3-319-60791-7