O problema não rotulado de Geometria de Distâncias

Aluno: Gustavo Café de Miranda
Departamento de Física, CFM, UFSC

Orientador: Prof. Dr. Douglas Gonçalves
Departamento de Matemática, CFM, UFSC

Resumo

O problema fundamental de Geometria de Distâncias consiste em determinar posições para objetos em um espaço Euclidiano de dimensão apropriada conhecendo apenas algumas distâncias entre pares de objetos. O problema de geometria de distancias não rotulado é uma generalização deste problema, em que não sabemos a quais pares de objetos uma determinada distância está associada. Mostramos que alguns resultados válidos para o problema clássico não valem mais, como a unicidade quando todas as distâncias par-a-par são conhecidas, levando ao que chamamos de estruturas fracamente homométricas. Buscamos resolver o problema por meio de um algoritmo de reconstrução geométrica sequencial, exaustivo, que simultaneamente busca atribuições de distâncias e constrói a estrutura correspondente. Finalmente, apresentamos alguns resultados experimentais e algumas considerações sobre algoritmos mais gerais na literatura.

Palavras-chave: Problema não rotulado de geometria de distâncias, Grafos e estruturas, Rigidez, Algoritmos Build-up

1 Introdução

O estudo de geometria de distâncias tem uma longa trajetória na história da matemática, mas seu tratamento sistemático é recente. A primeira formulação explícita do problema fundamental (rotulado) de geometria de

distâncias é de 1978, e o primeiro livro integralmente dedicado ao seu estudo só foi publicado em 2013, segundo Lavor e Liberti [8, pg.1]. O interesse no estudo de geometria de distâncias tem aumentado nos últimos anos com o desenvolvimento de nanotecnologias e nanoestruturas. Pois que muitas das propriedades de materiais nanoestruturados ou de moléculas complexas são determinadas pelas suas estruturas atomicas [5, pg. 2]. Experimentalmente, usa-se de técnicas de cristalografia (espectroscopia) que medem no material as distâncias interatômicas, mas não a estrutura atômica. Por isso é de grande interesse buscarmos reconstruir a estrutura original sabendo as distâncias entre os átomos.

De maneira mais abstrata, trataremos o problema como sendo de vértices e arestas (isto é, como se nosso objeto fosse um grafo). De fato, não estaremos preocupados aqui com as aplicações do estudo, mesmo sendo diversas, mas sim com seu desenvolvimento por si só.

Imagine, então, que sabemos de antemão as distâncias $D = (l_1, l_2, ..., l_m)$ de uma certa estrutura, e sabemos a quais pares de vértices cada distância está associada. O nosso problema originário (chamado de problema rotulado de geometria de distâncias) é determinar, com estas informações, qual era a nossa estrutura original. Neste estudo, trataremos de uma versão generalizada deste problema, o problema *não-rotulado* de geometria de distâncias. Com "não-rotulado" queremos dizer que eliminamos a informação de qual par de vértices cada distância está associada. Então temos a tarefa de descobrir ambas coisas, a estrutura e a ordem certa de distâncias a serem escolhidas ao construirmos a estrutura. Ocorre que algumas considerações sobre o problema anterior não são mais válidas para esta nova formulação.

Uma abordagem para resolver ambos problemas é o uso de métodos computacionais. Veremos adiante que algoritmos do tipo *Build-up* [3, pg.6 e pg.10] são usados para o problema rotulado e não rotulado. Nós desenvolvemos um algoritmo recursivo e exaustivo (que no pior dos casos testa todas permutações de distâncias em D).

Na Seção 2 introduzimos o problema fundamental (rotulado) de geometria de distâncias, para isto damos algumas definições preliminares. Discutimos a relação entre o número de soluções para o problema e o que definimos adiante como rigidez. Na Seção 3 enunciamos o problema não rotulado de geometria de distâncias, explicamos as diferenças da sua versão rotulada e mostramos que algumas considerações para o problema rotulado não são válidas para a sua versão não rotulada, como é o caso para estruturas fracamente homométricas. Apresentamos na Seção 4.1 métodos Build-up de reconstrução de estruturas. Já na Seção 4.2 fazemos uma generalização recursiva do método apresentado na Seção 4.1, para tratar do caso não rotulado. Na Seção 4.3 apresentamos o pseudo algoritmo desenvolvido na pesquisa. Finalmente na

Seção 5 apresentamos alguns resultados experimentais do nosso algoritmo e avaliamos o custo computacional efetivo.

2 O problema rotulado de geometria de distâncias

Iremos adiante recorrentemente usar da noção de um grafo. Por isso apresentaremos, antes de mais nada, algumas definições preliminares.

2.1 Definições preliminares

Definição 2.1 *Definimos um grafo $G = (V, E)$ como um par de conjuntos, V de vértices, e $E \subseteq V \times V$ de arestas.*

Por exemplo, podemos ter um grafo $G = (V, E)$ tal que $V = \{a, b, c\}$ e $E = \{\{a, b\}, \{a, c\}\}$, isto é, um grafo com três vértices e duas arestas ligando o vértice a aos dois restantes. Veja que o conjunto de arestas é composto por pares não ordenados, afinal não nos é necessário (pelo menos aqui neste trabalho) diferenciar para um grafo $\{a, b\}$ de $\{b, a\}$, isto é, a aresta que liga a à b da que liga b à a [1]. Como também não trabalharemos com vértices conectados a si mesmos então vamos definir grafos que excluem essa ocorrência.

Definição 2.2 *Um grafo é dito simples quando $\forall\{x, y\} \in E$, $x \neq y$.*

2.2 Problema rotulado de geometria de distâncias

O problema fundamental (ou rotulado) de geometria de distâncias (DGP[2]) pode ser enunciado da seguinte forma: dado $k \in \mathbb{Z}_+$ e um grafo simples $G = (V, E, \ell)$, com uma função peso nas arestas $\ell : E \to (0, \infty)$, encontre um mapa $x : V \to \mathbb{R}^k$ tal que

$$\|x(u) - x(v)\| = \ell(u, v), \quad \forall\{u, v\} \in E, \tag{1}$$

onde $\|.\|$ denota a norma euclidiana.

Quer dizer, queremos achar um mapa que associe pontos (coordenadas) aos vértices do nosso grafo, cujas distâncias entre eles (os pontos) são as mesmas dos valores que estão definidos por ℓ.

[1] Neste caso teríamos um *grafo direcionado.*

[2] Do inglês *Distance Geometry Problem*

Definição 2.3 *Uma solução do DGP (i.e., uma solução do sistema de equações* (1)*), é chamada de realização. Além disso, o par* (G, x) *é denominado estrutura.*

Iremos mais adiante abordar o DGP sob o aspecto do que se conhece na literatura como a *rigidez* de estruturas. Veja que o conjunto de soluções para o DGP, se excetuarmos as soluções dadas por rotações e translações totais da estrutura (que preservam as distâncias trivialmente), pode ser de três tipos: vazio, finito e infinito não enumerável. O conjunto é vazio quando não existe solução para (1). Por exemplo, imagine que tenhamos o grafo $G = (V, E)$ dado por $V = \{u, v, r\}$ e $E = \{\{u, v\}, \{v, r\}, \{u, r\}\}$, e as restrições de distâncias do DGP são dadas por $\ell(u, v) = \ell(v, r) = 1$ e $\ell(u, r) = 5$. Vemos então que o nosso grafo é dado por três vértices todos conectados entre si, porém as distâncias aqui violam a desigualdade triangular, logo não há nenhuma realização que possa satisfazer estas condições, e então nosso conjunto solução é vazio.

Agora examinemos o caso em que as condições podem ser satisfeitas por alguma realização, i.e. que o conjunto solução é não vazio. Suponha duas estruturas (G_1, x_1) onde $G_1 = (V_1, E_1)$, $V_1 = \{u, v, r\}$ e $E_1 = \{\{u, v\}, \{v, r\}\}$, e a segunda estrutura é dada por (G_2, x_2), onde $G_2 = (V_2, E_2)$, $V_2 = V_1$ e $E_2 = \{\{u, v\}, \{v, r\}, \{u, r\}\}$. Então, dadas as restrições $\ell(u, v) = \ell(v, r) = 1$ para a primeira, o conjunto de soluções para o DGP admite um quantidade infinita não-enumerável de soluções, pois qualquer rotação parcial (por menor que seja) da estrutura ainda satisfaz as restrições de distâncias e existe então uma realização que é solução. Como exemplificado na Figura 1.

Agora, se adicionamos a distância $\ell(u, r) = 1$, então o número de soluções passa a ser finito. Na verdade, neste caso, excetuando-se translações totais, reflexões totais e rotações, só há uma solução (a saber: a estrutura de um triângulo).

Veja que grafos como o da Figura 2, com um conjunto conveniente de distâncias (isto é, que produza uma estrutura similar ao da figura) também tem um número finito de soluções, porém desta vez há mais de uma.

A cardinalidade do conjunto solução de um DGP pode, como dissemos acima, ser associada intuitivamente a ideia de quanto uma estrutura é deformável. A seguir, veremos sob quais condições uma estrutura admite flexões.

Definição 2.4 *Dizemos que duas estruturas* (G, x_0) *e* (G, x_1) *são equivalentes se* $\forall \{u, v\} \in E, \|x_0(u) - x_0(v)\| = \|x_1(u) - x_1(v)\|$.

Definição 2.5 *Duas estruturas* (G, x_0) *e* (G, x_1) *são congruentes se*

$$\|x_0(u) - x_0(v)\| = \|x_1(u) - x_1(v)\|, \forall \{u, v\} \in V \times V.$$

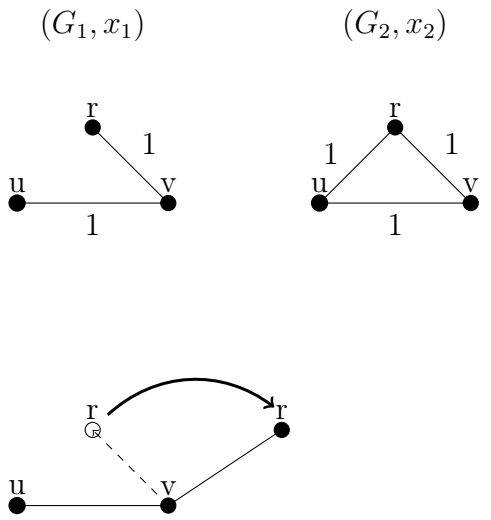

Figura 1: Soluções.

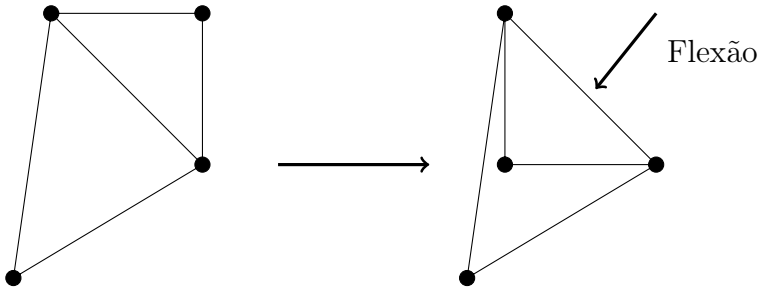

Figura 2: Flexão discreta.

Definição 2.6 *Dizemos que uma estrutura (G, x) é globalmente rígida se todas suas estruturas que são equivalentes são também congruentes.*

Por exemplo, toda estrutura de um grafo completo é globalmente rígida. Pois estando todos os vértices conectados por arestas, então qualquer movimento dos vértices está restrito ao tamanho das arestas, e logo a distância entre todos vértices é preservada. Veja por exemplo a estrutura (G_2, x_2) na Figura 1.

Outro conceito importante é o de *rigidez local*. Neste caso, se todas as estruturas equivalentes suficientemente próximas de (G, x_0) são congruentes, dizemos que a (G, x_0) é localmente rígida.

Definição 2.7 *Uma estrutura (G, x_0) é dita localmente rígida se $\exists \epsilon > 0$ tal que toda estrutura (G, x_1) que é equivalente a (G, x_0) e que satisfaz $\|x_0(v) - x_1(v)\| < \epsilon, \forall v \in V$, é também congruente a (G, x_0).*

Figura 3: Realizações equivalentes.

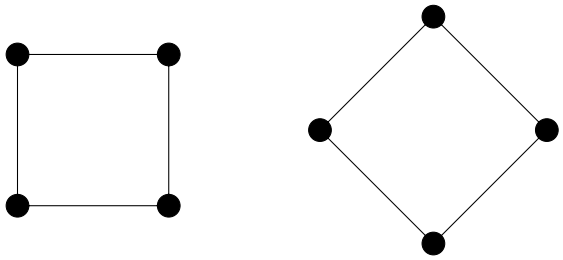

Figura 4: Realizações congruentes.

Do contrário, dizemos que a estrutura (G, x_0) é *localmente flexível*. Observe a Figura 2 com um exemplo de uma estrutura que é localmente rígida porém não é globalmente rígida. Nesta estrutura, podemos ver que há uma flexão discreta, isto é, há duas realizações que são equivalentes mas não congruentes, logo ela não é globalmente rígida. Porém é localmente rígida em $\mathbb{R}^2$, pois que as realizações que satisfazem $\|x_0(v) - x_1(v)\| < \epsilon, \forall v \in V$ que forem equivalentes serão congruentes também. Portanto vemos que estruturas podem ser rígidas (localmente) e não ter realização única.

Usaremos muito adiante a seguinte definição de rigidez.

Definição 2.8 *Uma estrutura (G, x_0) é dita Redundantemente Rígida, se retirada uma aresta, ela continua rígida.*

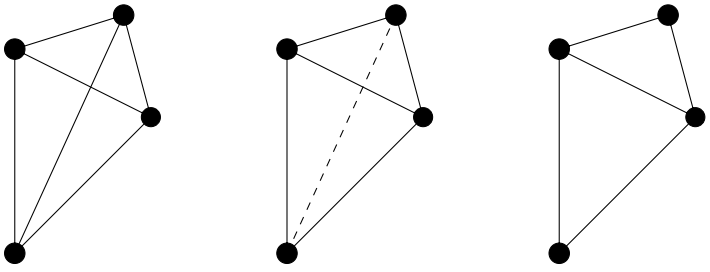

Figura 5: Estrutura redundantemente rígida.

3 Problema não rotulado de geometria de distâncias

Poderia se pensar que, em muitas situações experimentais, nós não temos acesso a todas as informações que o DGP pede. Por exemplo, poderíamos por meio de certos experimentos saber as distâncias entre os vértices (o tamanho das arestas) mas não saber a qual par de vértices cada distância está associada, e então teríamos uma nova complicação sobre o nosso problema inicial. Considere, por exemplo, o efeito de difração de raio X sobre um cristal. Neste experimento busca-se reconstruir a estrutura do cristal sabendo-se somente o comprimento de onda da luz antes e depois de interagir com o material [3, pg. 5]. Assim, sabemos algumas distâncias interatômicas no cristal, mas não sabemos a posição dos vértices (dos átomos), nem a que pares de átomos tais distâncias estão associadas, o que caracteriza um problema não rotulado de Geometria de Distâncias (uDGP [3]).

A informação que nos falta é o que chamamos de um *assinalamento* ou *atribuição*, isto é, uma função que associe cada distância, indexada por $j \in J = \{1, \ldots, m\}$, a um par de vértices $\{u, v\} \in V \times V$. Assim, buscamos uma função injetiva $\alpha : J \to V \times V$. Dado uma atribuição α, perceba que $\alpha(J) \subset V \times V$ define um conjunto de arestas para tal atribuição.

Iremos então formular o nosso problema de uma nova maneira. Podemos pensar que temos uma lista de distâncias, mas que não temos duas informações: a posição dos vértices, nem o assinalamento. E então buscamos reconstruir a estrutura com as informações que temos, isto é, somente as distâncias. Enuncia-se, então, o *Problema não rotulado de Geometria de Distâncias* (uDGP) da seguinte maneira

Dados $k \in \mathbb{Z}_+$, $J = \{1, ..., m\}$ e uma lista de distâncias $D = (l_1, l_2, ..., l_m)$, determine uma assinalamento $\alpha : J \to V \times V$, e uma imersão $x : V \to \mathbb{R}^k$ tal que:

$$\forall \{u, v\} \in \alpha(J) : \ell(u, v) = l_{\alpha^{-1}(\{u,v\})} \quad e \quad \|x_u - x_v\| = \ell(u, v). \tag{2}$$

Neste caso, para $j \in J$, escrevemos $l_j = l_{\alpha^{-1}(\{u,v\})}$, pois $\alpha^{-1}(\{u, v\}) = j$. Ora, então para resolver o uDGP precisamos encontrar α^{-1} que associe os pares de vértices certos às distâncias dadas, de forma que a estrutura final (G, x) seja condizente com a nossa lista de distâncias, e seja solução para o DGP consequente. Pois veja que, dada um assinalamento α, o nosso problema se reduz ao DGP clássico. Porém, visto que a α pode ter um caráter indeterminado (afinal, não conhecemos o assinalamento certo!), pode ser que obtenhamos uma atribuição cujo DGP não possui solução.

[3] do inglês, *unassigned Distance Geometry Problem*

Veremos mais adiante que certas condições podem influenciar a dificuldade do nosso uDGP e modificar a maneira de abordar o problema. Por exemplo, pode ocorrer que mais de uma estrutura seja solução de nosso problema (mesmo excluindo àquelas dadas por transformações rígidas). Afinal, temos a liberdade de alterar a atribuição e com isso encontrar outra estrutura com a mesma lista de distâncias. Neste caso, não sabemos se encontramos a estrutura que procurávamos a menos que continuemos procurando. Isto motiva a seguinte definição.

Definição 3.1 *Dizemos que duas estruturas* (G_1, p_1), (G_2, p_2) *são fracamente homométricas [9, pg. 1942] quando o conjunto de todas as distâncias (conjunto completo de distâncias) de* (G_1, p_1) *é igual ao de* (G_2, p_2).

Do contrário dizemos que são *não fracamente homométricas*.

Por exemplo, a lista de distâncias $\{4, 2, \sqrt{2}, \sqrt{2}, \sqrt{10}, \sqrt{10}\}$, associada à um grafo de 4 vértices, gera as seguintes possíveis estruturas rígidas (em $\mathbb{R}^2$):

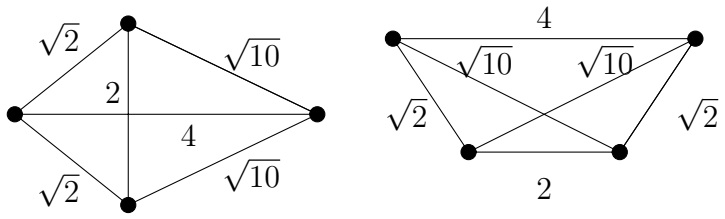

Figura 6: Estruturas Fracamente Homométricas (inspirado em [4, pg. 714]).

Com este exemplo, vê-se que o grafo ser completo não garante que a realização será única. Pois alterando o assinalamento, i.e., a ordem em que escolhemos as distâncias, obtemos duas estruturas, ambas globalmente rígidas, mas que diferem por mais do que rotações e translações totais. Vemos então que, alguns resultados do DGP não podem ser generalizados para o uDGP. Pois que no DGP o grafo ser completo era condição suficiente para que a solução fosse única [8, pg.11], enquanto que aqui, se houverem grafos fracamente homométricos, teremos mais de uma realização para a mesma lista de distâncias.

Veja que em aplicações como a de estudar a estrutura de cristais, é provável que encontraremos estruturas homométricas [3, pg.5]. Nestes casos, usa-se de argumentos físicos ou químicos (como a preferência do sistema para se organizar de maneira a minimizar a energia) para determinar qual das estruturas é a mais provável. Geralmente, quando mais de uma estrutura é possível, ambas coexistem no cristal, mesmo que não na mesma proporção.

4 Um algoritmo para o uDGP

Adiante discutiremos um método exaustivo para encontrar realizações redundantemente rígidas em $\mathbb{R}^k$, dada uma lista completa de distâncias D.

A princípio, poderíamos simplesmente listar todas as possíveis combinações de pares de vértices, todas atribuições, e verificar uma por uma, por meio de um DGP clássico. Porém, como discutiremos na Seção 4.2, o aumento do custo computacional é exponencial, o que torna tal algoritmo não muito atrativo.

Na literatura se destaca que, se o problema estiver restrito a estruturas não fracamente homométricas, e se soubermos o assinalamento correto (isto é, se nosso problema se reduzir a resolver um DGP rotulado), então o problema tem complexidade polinomial [3, pg. 5]. Isto nos motiva a buscar soluções que não deixem para verificar a rigidez no final, mas que a testem passo a passo. Por isso, esperamos reduzir o custo computacional, se verificarmos a rigidez redundante de subestruturas a cada passo intermediário do processo.

O algoritmo que desenvolvemos é fortemente inspirado no algoritmo TRIBOND de [3, pg. 6]. Porém, para maior simplicidade nos restringimos para $\mathbb{R}^2$, já que a ideia central do algoritmo não depende da dimensão, e pode ser generalizado para estruturas em $\mathbb{R}^k$. Também pedimos na nossa versão que a lista de distâncias seja completa, isto é, que todos vértices estivessem conectados. Assim, se temos uma estrutura com $|V|$ vértices [4], teríamos de ter uma lista de distâncias com cardinalidade $\frac{|V|(|V|-1)}{2}$.

4.1 Método Build-up

Existem métodos computacionais eficientes para certas classes do DGP rotulado, como por exemplo para a classe de grafos completos. O método que discutiremos nesta seção é conhecido como *Build-up* [5, pg. 4]. Neste método buscamos construir a estrutura um vértice por vez, de maneira que a cada passo se preserve a rigidez.

Dada uma lista de distâncias $D = (l_1, l_2, ..., l_m)$ completa e um uma atribuição α, construímos inicialmente uma estrutura redudantemente rígida. Neste caso, para $\mathbb{R}^2$ escolhemos uma estrutura como da Figura 5. Essa estrutura inicial é chamada de *Core*. Para o caso geral em $\mathbb{R}^k$, precisaríamos de uma estrutura com $K+2$ pontos, tais que todas as subestruturas de $K+1$ vértices sejam formadas por pontos afimente independentes.

Para construir o *Core*, escolhemos, arbitrariamente, que o primeiro ponto se encontre na origem, e que o segundo estivesse no eixo X. Com esta

[4] Aqui $|V|$ indica a cardinalidade do conjunto V

escolha, excluímos todas realizações dadas por rotações e translações totais da nossa estrutura. Ao escolhermos $x_0 = (0, 0)$ restringimos as translações, e ao escolhermos $x_1 = (l_1, 0)$ restringimos as rotações.

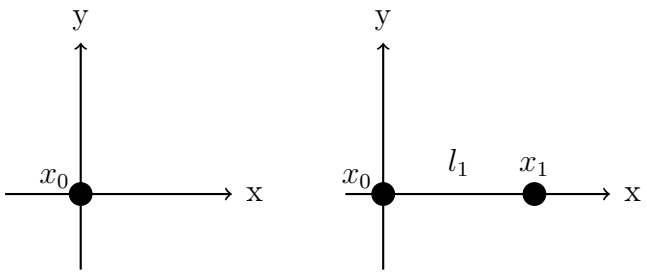

Figura 7: Primeiros vértices.

Agora, para o próximo ponto, consideramos um sistema de equações quadráticas da forma (1), mas em que as posições de x_0 e x_1 são conhecidas. Tomando o quadrado para facilitar a aritmética, temos

$$\begin{aligned} \|x_0 - x_2\|^2 &= l_2^2, \\ \|x_1 - x_2\|^2 &= l_3^2. \end{aligned} \tag{3}$$

Para cada equação temos

$$\|x_i\|^2 - 2\langle x_i, x_j\rangle + \|x_j\|^2 = \langle x_i - x_j, x_i - x_j\rangle = \|x_i - x_j\|^2 = l^2_{\alpha^{-1}(\{i,j\})}. \tag{4}$$

Subtraindo a primeira equação da segunda em (3):

$$\|x_1\|^2 - \|x_0\|^2 - 2\langle x_2, x_0 - x_1\rangle = l_3^2 - l_2^2,$$

e reorganizando e lembrando que $\|x_0\| = 0$ e $\|x_1\| = l_1$, obtemos

$$\langle x_0 - x_1, x_2\rangle = \frac{l_1^2 + l_2^2 - l_3^2}{2}. \tag{5}$$

A Equação (5), em conjunto com a primeira de (3), nos dão as posições de x_2. Porém, sendo uma das equações quadrática, temos uma arbitrariedade sobre o sinal. Geometricamente, estamos resolvendo um problema de intersecção de dois círculos, e que como representado na Figura 8, temos, em geral, dois pontos possíveis. A escolha de qual deles vamos usar é arbitrária. No nosso caso usamos a com valores positivos do eixo Y.

Para o terceiro e sucessivos pontos o raciocínio é análogo. Porém agora temos uma equação a mais.

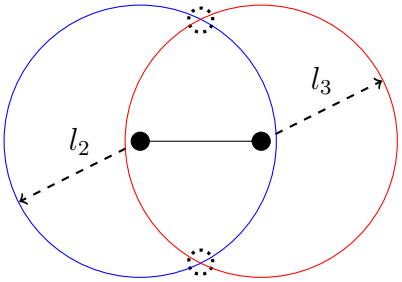

Figura 8: Definindo posição, Intersecção de 2 círculos.

$$\begin{aligned} \|x_0 - x_3\|^2 &= l_4^2, \\ \|x_1 - x_3\|^2 &= l_5^2, \\ \|x_2 - x_3\|^2 &= l_6^2. \end{aligned} \tag{6}$$

Novamente subtraindo a primeira das demais, temos as seguintes equações:

$$\begin{aligned} \langle x_3, x_1 - x_0 \rangle &= \tfrac{\|x_1\|^2 - \|x_0\|^2 + l_4^2 - l_5^2}{2}, \\ \langle x_3, x_2 - x_0 \rangle &= \tfrac{\|x_0\|^2 - \|x_2\|^2 + l_4^2 - l_6^2}{2}. \end{aligned} \tag{7}$$

Esta equação pode ser representada como um sistema linear

$$Ax_3 = b, \quad \text{onde } A = \begin{pmatrix} (x_1 - x_0)^T \\ (x_2 - x_0)^T \end{pmatrix} \tag{8}$$

em que A é uma matriz 2×2, e v^T denota o transposto de um vetor coluna.

É importante notar que, se os pontos x_0, x_1 e x_2 são afimente independentes, i.e., são não-colineares, então a matriz A é não singular e o sistema (8) tem solução única. Mesmo assim, é necessário verificar se a solução deste sistema satisfaz as equações (6), pois as distâncias envolvidas poderiam ser incompatíveis.

Resolvendo então este sistema linear e usando a primeira equação de (6) para validar o vetor x_3 encontrado, temos o novo ponto. Este problema está relacionado a encontrar a intersecção de 3 círculos.

Diferentemente do caso com 2 pontos, agora não há ambiguidade. Pois, sob as condições acima, a intersecção de 3 círculos (quando não vazia) é única.

Esta técnica usada para encontrar os pontos é chamada na literatura de *trilateração* [3, pg. 6].

Para os demais vértices o raciocínio é o mesmo. Portanto, uma vez montado nosso *Core*, vamos adicionar novos pontos um a um, utilizando sempre

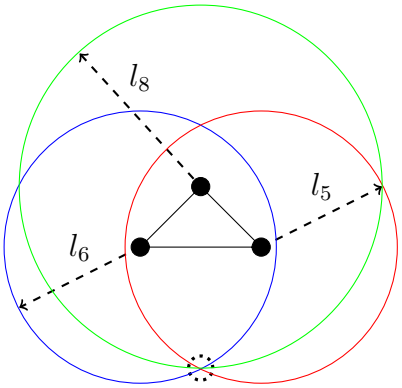

Figura 9: Definindo posição, Intersecção de 3 círculos.

três vértices de referências *não-colineares* na subestrutura já construída e três distâncias do novo ponto a estas referências.

No caso geral, pediríamos $K + 1$ vértices, e procuraríamos $K + 1$ distâncias, para conectar o novo vértice. Assim, o novo ponto é adicionado à subestrutura de modo a garantir que a mesma é rígida, como representado na Figura 11.

Como o grafo associado a nossa estrutura é completo, então, para o mais novo vértice, existem distâncias que o conectam ao resto da estrutura. Para o DGP, já temos a atribuição correta, por isso basta completar as arestas referentes as demais distâncias em D que ligam os demais vértices da estrutura ao novo.

Uma vez conectado a todos os outros vértices da estrutura podemos passar ao próximo vértice, e continuar construindo nossa estrutura. Ao final do processo, quando se depletarem as distâncias, devemos encontrar a estrutura final. E como verificamos a cada passo que tínhamos rigidez, então ela deve ser rígida.

4.2 Generalização para uDGP

Como vimos anteriormente, o problema não rotulado se diferencia por não sabermos qual o assinalamento verdadeiro. O algoritmo que desenvolvemos é também do tipo *Build-up*, porém tem novas propriedades, dedicadas a resolver o problema da falta de assinalamento.

Agora, dada uma lista[5] de distâncias $D = (l_1, l_2, ..., l_m)$, temos de descobrir a ordem certa para a atribuição de cada distância aos pares de vértices.

[5]também poderíamos chamar de *multiset*, pois queremos um conjunto que admita repetições de elementos

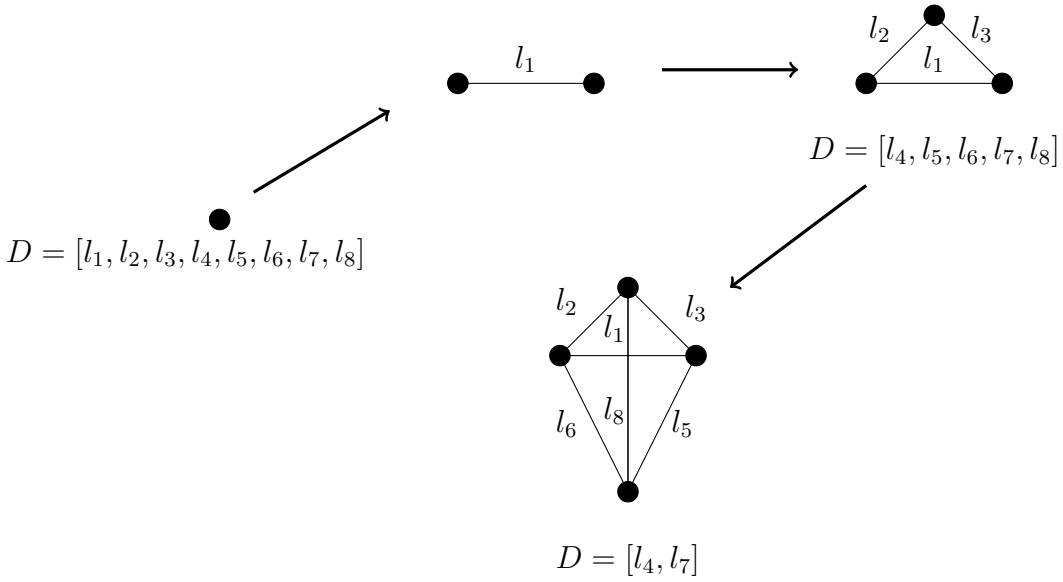

Figura 10: Sequência de construção do *Core*.

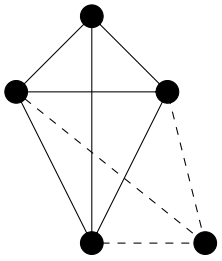

Figura 11: Novo vértice.

Poderíamos listar primeiro todas as atribuições possíveis, e então usar um método de Build-up para cada uma até encontrarmos aquela que nos dá a estrutura total. Este método reiniciaria a cada vez que se acha um caminho sem volta, ou seja, uma atribuição falsa. Porém, veja que se nossa estrutura desejada (G, x) de um grafo $G = (V, E)$ completo tem $|V|$ vértices, então tem $m = |V|(|V| - 1)/2$ distâncias. Logo ao listar todas as combinações possíveis estaremos lidando com $m!$ permutações da lista D. É claro que este algoritmo terá um custo computacional fatorial, o que nos motiva a procurar outra maneira de resolver o nosso problema.

O algoritmo que desenvolvemos também é exaustivo, isto é, testaremos *implicitamente* as atribuições uma a uma. No entanto, ele é recursivo e, em vez de trabalharmos separadamente os assinalamentos e o Build-up, o que faremos é resolver ambos paralelamente, da seguinte maneira. Primeiro, nosso

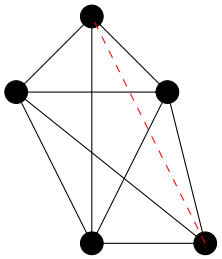

Figura 12: Checagem de distâncias remanescentes.

algoritmo escolhe uma combinação de 6 distâncias em $D = (l_1, l_2, ..., l_m)$, e constrói um *Core* da mesma maneira que na Seção 4.1. Se as distâncias escolhidas forem tais que o sistema quadrático (6) da Seção 4.1 não tiver solução, então já eliminamos as atribuições que começam com esta combinação e começamos novamente. Se a escolha for bem sucedida, então damos continuidade ao Build-up.

Para cada ponto novo, o algoritmo faz uma escolha de assinalamento (escolhe uma tripla de distâncias) e uma tripla de vértices na estrutura, e define a posição nova por trilateração. Novamente, se não houver solução para o sistema de equações (6), o algoritmo deve procurar outra tripla de distâncias e de vértices. Se a atribuição for bem sucedida até agora, então prosseguimos para a próxima etapa.

Como o grafo é completo, se a nossa atribuição for correta, existem distâncias em D que satisfazem a distância do vértice novo aos demais vértices da estrutura, como representado na Figura 12. Porém agora, não temos certeza se temos o assinalamento correto. Por isso precisamos verificar as distâncias remanescentes uma por uma, isto é, se usamos até agora as distâncias $D_u \subseteq D$, então temos de verificar em $D \setminus D_u$ se alguma distância satisfaz o valor desejado. Fazemos então uma busca entre as distâncias até encontrarmos aquela que satisfaz o valor desejado.

Se conseguimos conectar o vértice novo a todos os outros, então damos prosseguimento a construção. Para isso, o que fazemos é repetir o procedimento para um ponto novo, porém com a lista de distâncias atualizada $D \setminus D_u$ e com a nossa estrutura atual (isto é, com o Core e os pontos já fixados até o momento). Fazemos isto de maneira recursiva, isto é, a função, cuja tarefa é fixar um ponto novo, tem também como tarefa chamar a si mesma com as variáveis atualizadas para localizar o próximo ponto.

Sendo o algoritmo recursivo, toda vez que encontrar um caminho sem saída, ele deve retornar ao passo anterior e testar as possibilidades remanescentes. No caso, é como se a distância representada na Figura 12 não tivesse correspondência na lista, e então precisamos voltar ao que já havíamos cer-

teza e testar outra combinação. Veja a Figura 13.

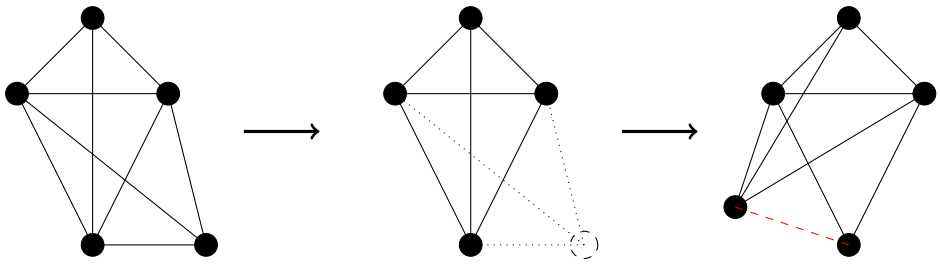

Figura 13: Caminho falso, e nova tentativa.

Assim, temos uma vantagem sobre aquele algoritmo que sugerimos anteriormente: de listar as possibilidades e testá-las uma a uma. Pois que, é possível que encontremos a nossa estrutura já nas primeiras iterações. Somente no pior dos casos é que a estrutura final está na última tentativa. O custo continua sendo exponencial, porém o número de tentativas pode ser menor.

De fato, encontrar uma estrutura não é conclusão do problema. Pois como vimos, estruturas homométricas existem, e pode ser que, se continuarmos a nossa busca, encontremos outras realizações. Porém, se nos basta uma, então o algoritmo pode parar assim que a encontrar.

4.3 Algoritmo para o uDGP

Abaixo rescrevemos as ideias principais discutidas nessa seção em forma de um pseudo-algoritmo.

O Algoritmo 1, é responsável pela construção do Core. Começamos com a lista de distâncias $D = (l_1, l_2, ...l_m)$ e garantimos que a cardinalidade de D é tal que a nossa estrutura seja de um grafo completo. Por isso pedimos que $|D| = |V|(|V| - 1)/2$. Este algoritmo faz combinações $\binom{|D|}{6}$ e constrói Cores se as distâncias escolhidas são compatíveis. Entendemos como *compatíveis* as distâncias serem tais que conseguimos resolver o sistema quadrático (6). Veja também que não escrevemos o conjunto de distâncias em notação de conjuntos, pois que aqui a ordem e a multiplicidade dos elementos importa, logo não é um conjunto propriamente dito.

Uma vez construído um Core, o Algoritmo 1 irá chamar outro, aqui denominado *Newpoint*, que corresponde ao nosso Algoritmo 2. Veja que ao montar o Core, o Algoritmo 1 atualiza a lista de distâncias, e chama a função Newpoint com os valores atualizados de D e com o Core: $Newpoint(D, Core)$.

A função *Newpoint*, tem dois argumentos, a lista de distâncias D e uma estrutura X. Para esta função fica a tarefa de adicionar um ponto à estrutura

Algoritmo 1 Dado um conjunto de distâncias completo $D = (l_1, l_2, ..., l_m)$, com $|D| = |V|(|V| - 1)/2$, encontrar estrutura rígida (G, p) em $\mathbb{R}^2$.

$X = \emptyset$, $D' \leftarrow D$
for cada combinação de 6 distâncias $D_u \subseteq D'$, das $\binom{|D|}{6}$ possíveis **do**
  if as distâncias em D_u forem compatíveis **then**
    $D \leftarrow D'$
    Monte um $Core$ com as distâncias D_u (por trilateração).
    $D \leftarrow D \setminus D_u$, $X \leftarrow Core$
    X= Newpoint (D, $Core$)
  end if
  if $|X| = |V|$ **then**
    return X
  end if
end for
return fail.

pelo método discutido na Seção 4.2, isto é, trilateração e depois checar se as distâncias remanescentes correspondem aos demais vértices da estrutura. Para isto, faz uma escolha dentro as combinações das distâncias $\binom{|D|}{3}$ e de vértices $\binom{|X|}{3}$ da estrutura. Também entende-se aqui as combinações de 3 distâncias $D'_u \subseteq D'$ como triplas de distâncias (l_{k1}, l_{k2}, l_{k3}) cujos elementos $l_{ki} \in D'$. Veja que aqui $(l_{k1}, l_{k2}, l_{k3}) \neq (l_{k2}, l_{k3}, l_{k1})$, por exemplo.

Uma vez fixado o novo vértice a função atualiza os valores de D e da estrutura X, e chama a si mesma com os valores atualizados, $Newpoint(D, X)$. Usamos a notação de conjunto $X \cup \{x\}$, para sinalizar que o novo vértice foi adicionado à estrutura.

Se a função Newpoint não for bem sucedida, então ela volta à sua instância anterior, onde buscará uma nova combinação de distâncias e vértices. Finalmente, um condicional em $|X| = |V|$, nos garante que encontramos a estrutura total.

Algoritmo 2 X = Newpoint(D,X)

if $D = \emptyset$ **then**
  return X
else
  $X' \leftarrow X$, $D' \leftarrow D$
  for cada combinação de 3 distâncias $D'_u \subseteq D'$ **do**
    for cada tripla de vértices *não colineares* $X_u \subseteq X'$ **do**
      if Se X_u e D'_u são compatíveis **then**
        $D \leftarrow D'$
        Encontre a posição de um novo ponto x por trilateração, usando as referências X_u e as distâncias D'_u
        $D \leftarrow D \setminus D'_u$
        for $v \in X \setminus X_u$ **do**
          for $d \in D$ **do**
            if $d = \|x - v\|$ **then**
              $D \leftarrow D \setminus \{d\}$
              Vá para o próximo v.
            end if
          end for
          Vá para o próximo X_u.
        end for
        $X \leftarrow X \cup \{x\}$
        X = Newpoint(D,X)
        if $|X| = |V|$ **then**
          return X
        else
          $X \leftarrow X'$
        end if
      end if
    end for
  end for
  return X'
end if

Pontos	Distâncias	Tempo (s)
5	10	0.2
6	15	0.86
10	45	10.5
15	105	67.2
20	190	525
25	300	4761

Tabela 1: Tempo (em segundos) para encontrar a primeira estrutura.

5 Experimentos computacionais

Implementamos nosso algoritmo em linguagem Matlab e executamos alguns testes com um número pequeno de vértices.

Para gerar as instâncias, primeiro geramos pontos aleatórios no quadrado unitário, por meio da função *rand(2,N)* do Matlab. A seguir, calculamos todas as $N(N-1)/2$ distâncias e criamos então a lista D, com estas distâncias embaralhadas aleatoriamente.

Interrompemos o algoritmo ao encontrar a primeira estrutura. Segundo a Tabela 1, os tempos de resolução (em segundos) parecem crescer de maneira exponencial, o que é confirmado pelo ajuste mostrado na Figura 14.

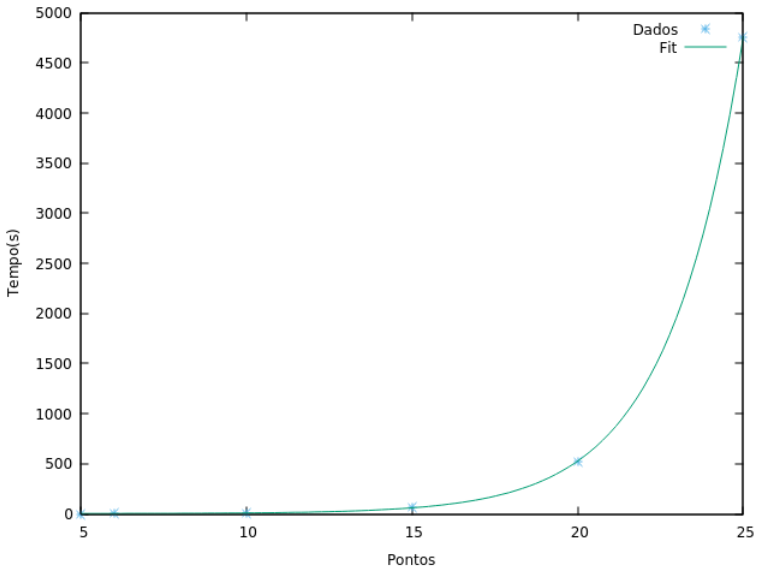

Figura 14: Gráfico de Pontos experimentais e da curva exponencial correspondente .

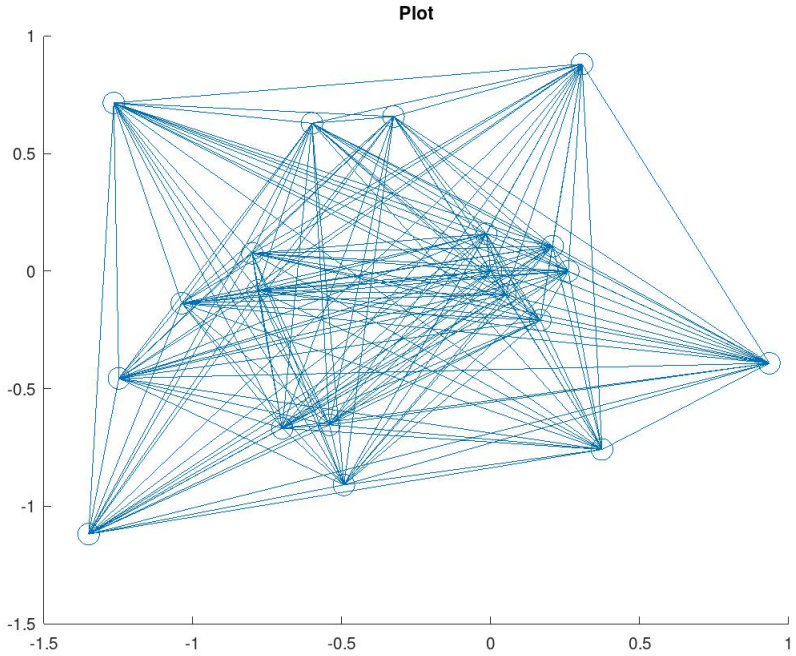

Figura 15: Estrutura para 20 pontos.

A estrutura da Figura 15, corresponde a uma das nossas estruturas encontradas para 20 pontos. Para facilitar a visualização usamos funções do Matlab destinadas a fazer gráficos de grafos.

A estrutura da Figura 16, é o resultado de um teste para 10 pontos.

6 Conclusão

O problema não rotulado de geometria de distâncias é uma área de estudo desafiadora. Vimos que em comparação ao rotulado tem muitas outras dificuldades que devem ser levadas em conta. A tarefa de encontrar o assinalamento correto torna os algoritmos muito mais custosos, e consequentemente, precisamos de mais tempo para encontrar as soluções.

Vimos que a informação do assinalamento altera o custo de maneira sensível. Se sabemos o assinalamento correto então é polinomial (para estruturas não fracamente homométricas), da ordem de $(|V|-4)\mathcal{O}(2^3)$ [6] . Se não sabemos o assinalamento, então o custo é exponencial, pois temos de

[6]Aqui o número 2 corresponde à dimensão, no nosso caso, $\mathbb{R}^2$, para o algoritmo generalizado teríamos $(|V|-k)\mathcal{O}(k^3)$, para $\mathbb{R}^k$.

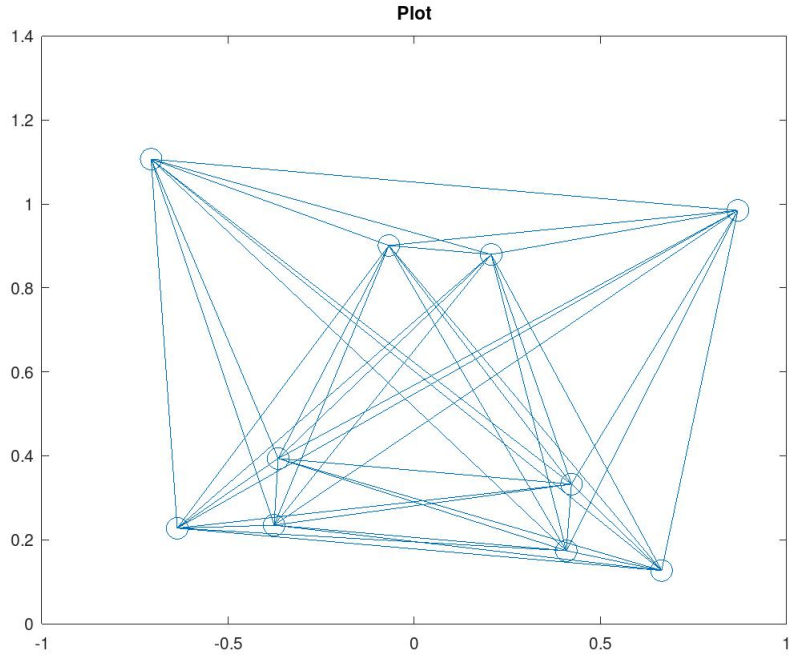

Figura 16: Estrutura para 10 pontos.

testar (possivelmente) todas atribuições possíveis, que é dado por $m!$, onde $m = \frac{|V|(|V|-2)}{2}$. Assim, é necessário recorrer a outras técnicas para diminuir o custo, como o uso de algoritmos recursivos.

O nosso algoritmo, parece corresponder a essas expectativas, como vemos na Figura 14. Nesta figura temos os pontos dos custos da tabela acima dados no plano (m, t), onde t é o tempo computacional dado em segundos. A curva $f(x)$ traçada é a melhor curva para o fitting dos dados, e é uma exponencial da forma $t = A.e^{(Bm)}$, em que t é dado em segundos.

Há generalizações do nosso algoritmo para maiores dimensões e para instâncias de grafos que não sejam completas. De fato, o que desenvolvemos aqui é um caso especial. Porém, sendo a ideia central do algoritmo a mesma para maiores dimensões, a generalização é feita de maneira a seguir o mesmo raciocínio.

Agradecimentos

Agradeço ao meu orientador pelo ensino, a paciência, e o cuidado; ao programa PIBIC do CNPq pelo financiamento; e a muitos outros pelo apoio

diário.

Referências

[1] L. ASIMOW and B. ROTH. The rigidity of graphs I. *Transactions of the American Mathematical Society*, 245:279–289, 1978.

[2] L. ASIMOW and B. ROTH. The rigidity of graphs II. *Journal of Mathematical Analysis and Applications*, 68:171–190, 1979.

[3] S. J. L. BILLINGE, DUXBURY P. M., GONÇALVES D. S., LAVOR C., and MUCHERINO A. Assigned and unassigned distance geometry: applications to biological molecules and nanostructures. *4OR*, 14(4):337–376, 2016.

[4] M. BOUTIN and G. KEMPER. On reconstructing n-point configurations from the distribution of distances or areas. *Advances in Applied Mathematics*, 32(4):709–735, 2004.

[5] P.M. DUXBURY, L. GRANLUND, S.R. GUJARATHI, P. JUHAS, and S.J.L. BILLINGE. The unassigned distance geometry problem. *Discrete Applied Mathematics*, 204:117–132, 2016.

[6] B. HENDRICKSON. Conditions for unique graph realizations. *SIAM Journal on Computing*, 21(1):65–84, 1992.

[7] B. JACKSON. Notes on the rigidity of graphs. Technical report, School of Mathematical Sciences, Queen Mary, University of London, 2007.

[8] C. LAVOR and L. LIBERTI. *Um convite à Geometria de Distâncias*, volume 71. SBMAC, São Carlos, SP, 2014. ISBN 978-85-8215-050-4. Notas em Matemática aplicada.

[9] M. SENECHAL. A point set puzzle revisited. *European Journal of Combinatorics*, 29(8):1933–1944, 2008.