Disciplina INF2604 – Geometria Computacional

RELATÓRIO DE EXERCÍCIO:
Detecção do círculo mínimo envolvente em pontos gerados aleatoriamente

Aluno: Gustavo Coelho

1. Objetivos

O objetivo deste relatório é detalhar os fundamentos teóricos e relatar as abordagens práticas para a resolução do problema de detecção do círculo mínimo envolvente em pontos gerados aleatoriamente. Serão consideradas duas abordagens, sendo a primeira baseada em um algoritmo heurístico que fornece uma aproximação da solução exata e que deve obrigatoriamente envolver todos os pontos. A segunda abordagem será baseada em um algoritmo de detecção do círculo mínimo envolvente exato. Apesar de se esperar algum grau de imprecisão em comparação à solução exata, a solução baseada no algoritmo heurístico deve apresentar um ganho de performance, o que será também mostrado.

2. Métodos

Algoritmo Heurístico:

O primeiro método utilizado (algoritmo heurístico) busca uma forma simples de determinar o centro do círculo envolvente $C = (C_x, C_y)$ e seu raio r. Ressalta-se que este método não resulta no exato círculo mínimo.

O primeiro passo do algoritmo é encontrar os pontos que possuam:

- Coordenada x mínima: $p_{x_{min}}$
- Coordenada x máxima: $p_{x_{max}}$
- Coordenada y mínima: $p_{y_{min}}$
- Coordenada y máxima: $p_{y_{max}}$

Para cada uma dessas coordenadas, é necessário percorrer o conjunto de dados por completo, a fim de localizar os valores máximos e mínimos. Portanto a complexidade temporal assintótica será $O_{(n)}$.

A heurística parte da hipótese de que o par de pontos mais distante no conjunto ofereçe uma boa aproximação inicial de um círculo envolvente, onde a distância entre este par forma o diâmetro deste círculo (algo que será verificado e eventualmente ajustado em um passo posterior). Portanto, o próximo passo é encontrar o par de pontos p_i e p_j no conjunto $\{p_{x_{min}}, p_{x_{max}}\}, \{p_{y_{min}}, p_{y_{max}}\}$, de distância máxima. Para isso, calculamos a seguinte métrica para cada par possível de pontos neste conjunto:

$$\|\overrightarrow{d_{i,j}}\| = \sqrt{\left(p_{i_x} - p_{j_x}\right)^2 + \left(p_{i_y} - p_{j_y}\right)^2}$$

O número de cálculos para a definição de $\|\overrightarrow{d_{i,j}}\|$ é constante e no máximo 4x4 = 16. Em cada iteração, verifica-se se o valor calculado é maior em relação ao valor armazenado (inicialmente nulo). Armazena-se então o maior valor, assim como o par de pontos correspondente $\{p_i, p_j\}$, e descarta-se o de menor valor, encontrando ao final da iteração a maior distância $\|\overrightarrow{d_{i,j}}\|_{max}$. Portanto, a complexidade será $O_{(1)}$. Em seguida, consideramos a primeira estimativa de C e r, onde:

$$C_x = \frac{p_{i_x} + p_{j_x}}{2}, \quad C_y = \frac{p_{i_y} + p_{j_y}}{2} \quad \text{e} \quad r = \frac{\|\overrightarrow{d_{i,j}}\|_{max}}{2}$$

Por fim, verifica-se a validade da hipótese tomada inicialmente e certifica-se de que todos os pontos estão envolvidos pelo círculo. Caso contrário, o centro e raio do círculo são incrementados na direção do ponto não envolvido e proporcionalmente à distância deste ponto à fronteira do círculo atual. Ou seja, primeiro verifica-se para cada ponto p_k e sua distância $\|\vec{d}\|$ em relação ao centro C do círculo:

$$\|\vec{d}\| = \sqrt{\left(p_{k_x} - C_x\right)^2 + \left(p_{k_y} - C_y\right)^2}$$

Caso $\|\vec{d}\| < r$, passa-se à iteração seguinte. Caso contrário, recalcula-se C e r da seguinte forma:

$$C_x = C_x + \left(\frac{\|\vec{d}\|-r}{2}\right)\frac{(p_{k_x}-C_x)}{\|\vec{d}\|},\ \ C_x = C_y + \left(\frac{\|\vec{d}\|-r}{2}\right)\frac{\left(p_{k_y}-C_y\right)}{\|\vec{d}\|} \quad \text{e} \quad r = \frac{\|\vec{d}\|+r}{2}$$

Esta iteração percorrerá novamente o conjunto de pontos inteiramente, resultando em uma complexidade $O_{(n)}$, que será o mesma complexidade do algoritmo como um todo.

Ao final desta sequência, o algoritmo retornará um círculo com garantia de envolver o conjunto de pontos, porém sem nenhuma constatação de que seja o círculo envolvente mínimo, como já mencionado.

Algoritmo de Círculo Mínimo envolvente:

O segundo algoritmo tem como propósito encontrar de fato o círculo mínimo envolvente. Primeiro, realizamos a permutação randômica do conjunto de pontos. Isso é feito para evitar a ordenação dos pontos em uma ordem que aumente a complexidade do algoritmo.

Em seguida, inicializamos a função “MinCircle”, que recebe o conjunto de dados randomizados. A função “MinCircle” inicializa C e r considerando os dois primeiros pontos do conjunto (após a permutação randômica), ou seja, os pontos p_0 e p_1 (como usaremos a linguagem Python para implementação, usaremos a indexação iniciada por zero):

$$C_x = \frac{p_{0_x}+p_{1_x}}{2},\quad C_y = \frac{p_{0_y}+p_{1_y}}{2} \quad \text{e} \quad r = \frac{\sqrt{\left(p_{0_x}-p_{1_x}\right)^2+\left(p_{0_y}-p_{1_y}\right)^2}}{2}$$

O próximo passo inicializa um loop que percorre o conjunto de dados a partir do terceiro elemento e verifica se cada um está contido no círculo. Ou seja, verifica-se a seguinte condição:

$$\sqrt{\left(p_{k_x}-C_x\right)^2+\left(p_{k_y}-C_y\right)^2} < r$$

Caso a condição seja verdadeira, passa-se para a seguinte iteração. Caso contrário, a função “MinCircWithPoint” é chamada. Esta função recebe o conjunto de pontos $\{p_0, \dots, p_{k-1}\}$ e o ponto p_k separadamente.

A função “MinCircWithPoint” por sua vez inicializa C e r de maneira análoga à “MinCircle”, considerando os pontos p_0 e p_k, iniciando um novo loop para verificar se os pontos estão envolvidos pelo novo círculo. Caso contrário, uma outra função “MinCircleWith2Points” é chamada considerando o conjunto $\{p_0, \dots, p_{j-1}\}$, p_j e p_k, onde *j* representa cada iteração deste loop.

A função “MinCircleWith2Points” por sua vez inicializa o novo círculo considerando p_j e p_k, entrando em um novo loop para verificar a envolvência dos pontos. Caso um ponto não esteja envolvido, então a única hipótese restante é de que o círculo mínimo passa por p_j, p_k e p_i, onde i representa a iteração deste loop.

Portanto o problema se resume na obtenção do círculo que contém os três pontos mencionados. Para encontrá-lo, primeiro deduzimos que as distâncias entre qualquer um dos três pontos e o centro do círculo são iguais, ou seja:

$$\|p_j - C\| = \|p_k - C\|$$
$$\|p_j - C\| = \|p_i - C\|$$

Desenvolvendo a equação, temos:

$$(p_{j_x} + C_x)^2 + (p_{j_y} + C_y)^2 = (p_{k_x} + C_x)^2 + (p_{k_y} + C_y)^2$$
$$(p_{j_x} + C_x)^2 + (p_{j_y} + C_y)^2 = (p_{i_x} + C_x)^2 + (p_{i_y} + C_y)^2$$

$${p_{j_x}}^2 + 2p_{j_x}C_x + {C_x}^2 + {p_{j_y}}^2 + 2p_{j_y}C_y + {p_y}^2 = {p_{k_x}}^2 + 2p_{k_x}C_x + {C_x}^2 + {p_{k_y}}^2 + 2p_{k_y}C_y + {C_y}^2$$
$${p_{j_x}}^2 + 2p_{j_x}C_x + {C_x}^2 + {p_{j_y}}^2 + 2p_{j_y}C_y + {p_y}^2 = {p_{i_x}}^2 + 2p_{i_x}C_x + {C_x}^2 + {p_{i_y}}^2 + 2p_{i_y}C_y + {C_y}^2$$

$$C_x(2p_{j_x} - 2p_{k_x}) + C_y\left(2p_{j_y} - 2p_{k_y}\right) = {p_{k_x}}^2 + {p_{k_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2$$
$$C_x(2p_{j_x} - 2p_{i_x}) + C_y\left(2p_{j_y} - 2p_{i_y}\right) = {p_{i_x}}^2 + {p_{i_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2$$

Usando a regra de Cramer, definimos D, D_{C_x} e D_{C_y} da seguinte forma:

$$D = \begin{vmatrix} 2p_{j_x} - 2p_{k_x} & 2p_{j_y} - 2p_{k_y} \\ 2p_{j_x} - 2p_{i_x} & 2p_{j_y} - 2p_{i_y} \end{vmatrix}$$

$$D_{C_x} = \begin{vmatrix} {p_{k_x}}^2 + {p_{k_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2 & 2p_{j_y} - 2p_{k_y} \\ {p_{i_x}}^2 + {p_{i_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2 & 2p_{j_y} - 2p_{i_y} \end{vmatrix}$$

$$D_{C_y} = \begin{vmatrix} 2p_{j_x} - 2p_{k_x} & {p_{k_x}}^2 + {p_{k_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2 \\ 2p_{j_x} - 2p_{i_x} & {p_{i_x}}^2 + {p_{i_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2 \end{vmatrix}$$

Onde $C_x = \frac{D_{C_x}}{D}$ e $C_y = \frac{D_{C_y}}{D}$. Portanto, temos:

$$C_x = \frac{\left({p_{k_x}}^2 + {p_{k_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2\right)\left(p_{j_y} - p_{i_y}\right) - \left({p_{i_x}}^2 + {p_{i_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2\right)\left(p_{j_y} - p_{k_y}\right)}{2\left[\left(p_{j_x} - p_{k_x}\right)\left(p_{j_y} - p_{i_y}\right) - \left(p_{j_x} - 2p_{i_x}\right)\left(p_{j_y} - p_{k_y}\right)\right]}$$

$$C_y = \frac{\left(p_{j_x} - p_{k_x}\right)\left({p_{i_x}}^2 + {p_{i_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2\right) - \left(p_{j_x} - p_{i_x}\right)\left({p_{k_x}}^2 + {p_{k_y}}^2 - {p_{j_x}}^2 - {p_{j_y}}^2\right)}{2\left[\left(p_{j_x} - p_{k_x}\right)\left(p_{j_y} - p_{i_y}\right) - \left(p_{j_x} - p_{i_x}\right)\left(p_{j_y} - p_{k_y}\right)\right]}$$

Após encontrar as coordenadas do centro do círculo, podemos então definir o raio como sendo a distância entre o centro e qualquer um dos três pontos, como por exemplo p_j:

$$r = \sqrt{\left(p_{j_x} - C_x\right)^2 + \left(p_{j_y} - C_y\right)^2}$$

A complexidade esperada para o algoritmo é $O_{(n)}$, como veremos na implementação.

3. Implementação

Ambos algoritmos foram implementados em linguagem Python, gerando uma massa randômica de pontos com distribuição normal e com diferentes tamanhos. As figuras abaixo ilustram os resultados gráficos para diferentes tamanhos de conjuntos de pontos:

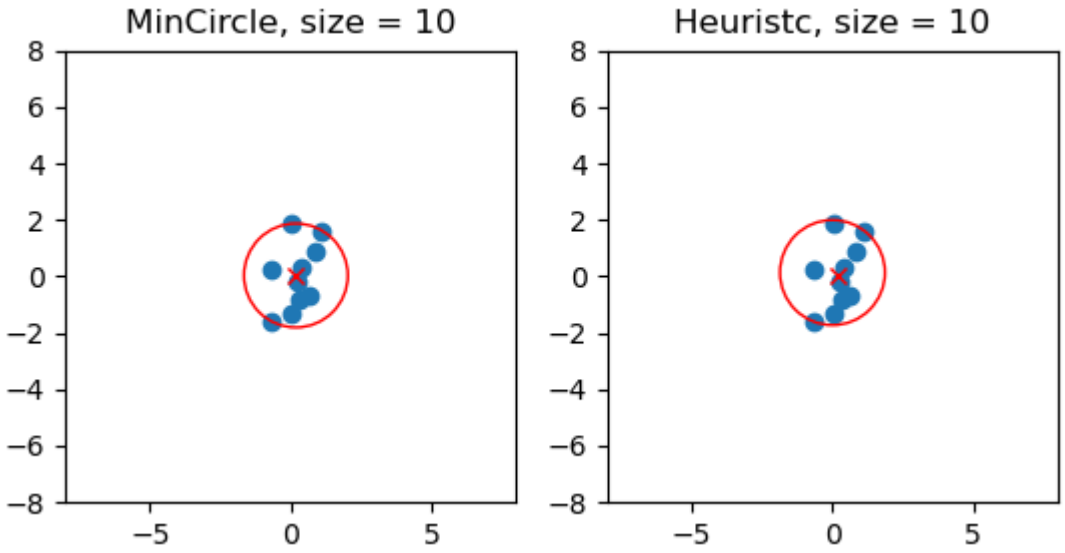

Figura 1: Comparação entre métodos com 10 pontos.

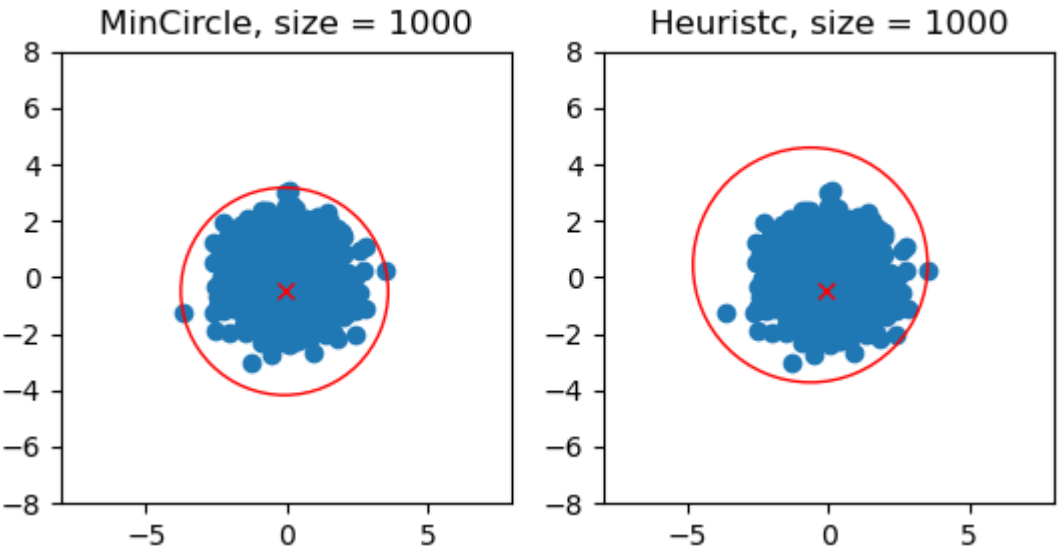

Figura 2: Comparação entre métodos com 1.000 pontos.

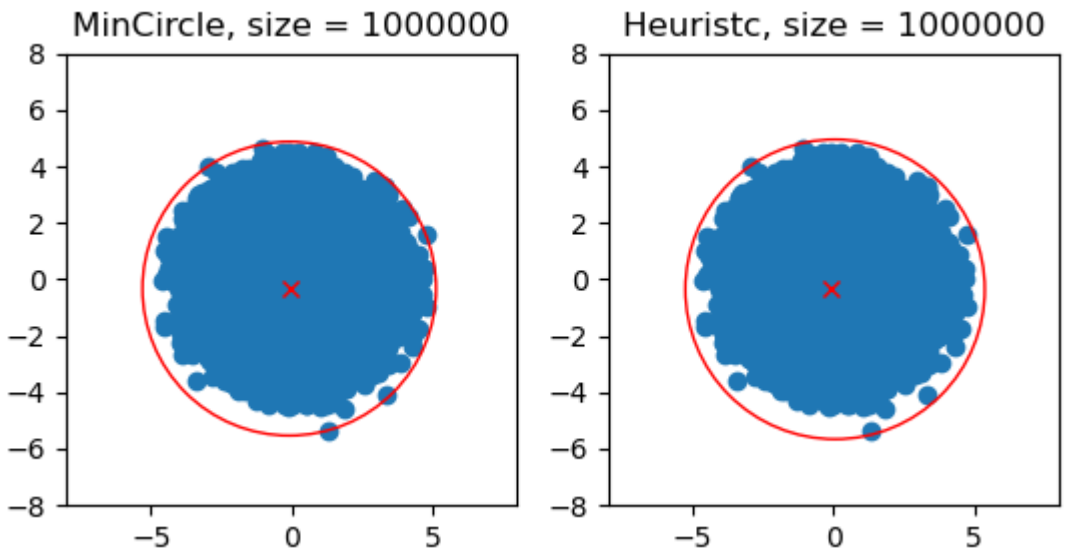

Figura 3: Comparação entre métodos com 1 milhão de pontos.

Conforme esperado, enquanto o algoritmo de cálculo do círculo mínimo envolvente é capaz de definir a solução exata, o algoritmo baseado na heurística definida retorna uma aproximação, que em alguns casos pode apresentar uma maior distorção, como mostrado na Figura 2.

Abaixo, podemos visualizar a representação gráfica da comparação entre os tempos de execução de ambos algoritmos:

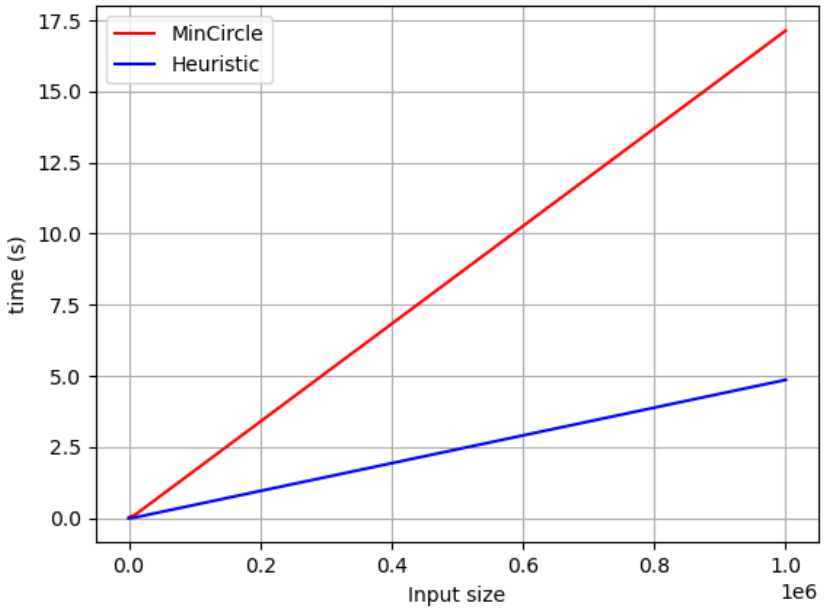

Figura 4: Comparação entre tempos de execução.

Como podemos observar, ambos apresentam uma complexidade linear, como esperado. Porém, o algoritmo baseado em heurística mostra uma constante de inclinação menor, fazendo com que haja um ganho de performance em relação ao método de mínimo círculo envolvente.

4. Conclusão

Mostrou-se neste relatório o detalhamento teórico dos dois métodos propostos, assim como a implementação dos algoritmos em linguagem Python e uma análise gráfica dos resultados, assim como uma comparação de performance computacional.

Ambos algoritmos explorados apresentam vantagens e desvantagens, a depender da aplicação. Enquanto o algoritmo de cálculo do círculo mínimo envolvente retorna a solução exata do problema, o algoritmo baseado em heurística retorna uma aproximação que pode ser suficiente em determinados casos, considerando que a aproximação não afeta a garantia de envolvência de 100% dos pontos.

Em contrapartida, apesar de ambos algoritmos apresentarem uma complexidade temporal assintótica proporcional a $O_{(n)}$, o método baseado em heurística mostra um ganho de desempenho em relação à alternativa de solução exata. Este ganho, por sua vez, pode não ser relevante para determinadas aplicações onde o conjunto de pontos seja pequeno e/ou onde o requerimento de tempo não seja altamente restrito.