

# 30º Colóquio Brasileiro de Matemática

IMPA, Rio de Janeiro, 26 a 31 de julho de 2015



## Aula 4

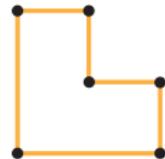
# Geração de malhas

Aula 4 - 29 de julho de 2015 - 12h30min às 14h

# Problema de Geração de Malhas

## Requisitos:

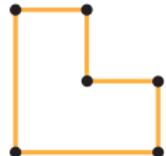
- Entrada: pontos e segmentos de reta



# Problema de Geração de Malhas

## Requisitos:

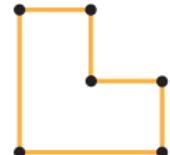
- ▶ Entrada: pontos e segmentos de reta
- ▶ Condições de optimização



# Problema de Geração de Malhas

## Requisitos:

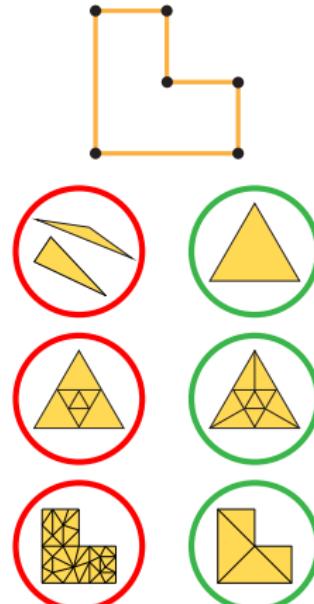
- ▶ Entrada: pontos e segmentos de reta
- ▶ Condições de optimalidade
- ▶ *Conformidade*



# Problema de Geração de Malhas

## Requisitos:

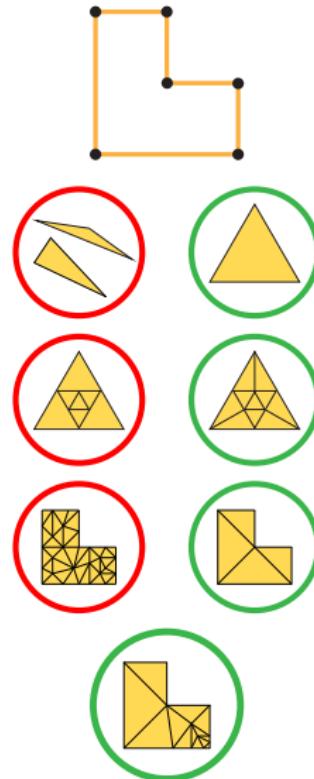
- ▶ Entrada: pontos e segmentos de reta
- ▶ Condições de optimalidade
- ▶ *Conformidade*
- ▶ Poucos elementos



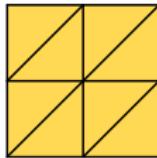
# Problema de Geração de Malhas

## Requisitos:

- ▶ Entrada: pontos e segmentos de reta
- ▶ Condições de optimalidade
- ▶ *Conformidade*
- ▶ Poucos elementos
- ▶ *Boa transição*



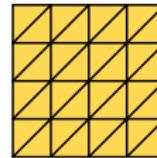
# Quantidade de elementos



poucos elementos



**solução mais rápida**

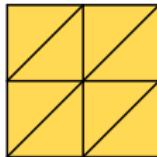


muitos elementos



**solução mais precisa**

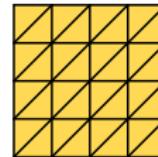
# Quantidade de elementos



poucos elementos



solução mais **rápida**



muitos elementos



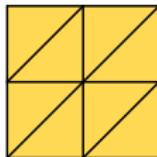
solução mais **precisa**

**Otimalidade × Poucos Elementos**



Lake Superior

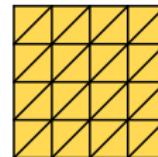
# Quantidade de elementos



poucos elementos



solução mais **rápida**

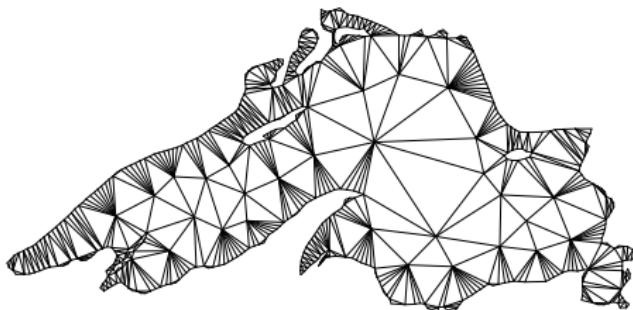


muitos elementos



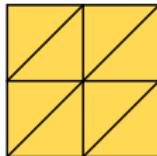
solução mais **precisa**

**Otimalidade × Poucos Elementos**



$$\theta_{\min} = 5^\circ \quad \# \triangle = 593$$

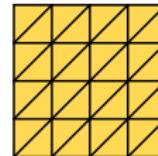
# Quantidade de elementos



poucos elementos



solução mais **rápida**

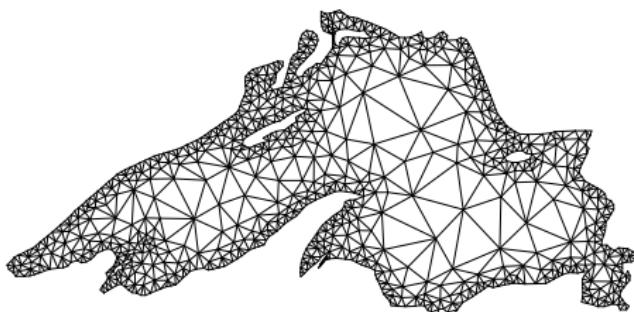


muitos elementos



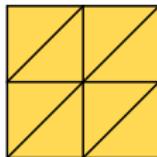
solução mais **precisa**

**Otimalidade × Poucos Elementos**



$$\theta_{\min} = 25^\circ \quad \# \triangle = 1427$$

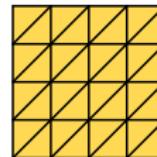
# Quantidade de elementos



poucos elementos



solução mais **rápida**

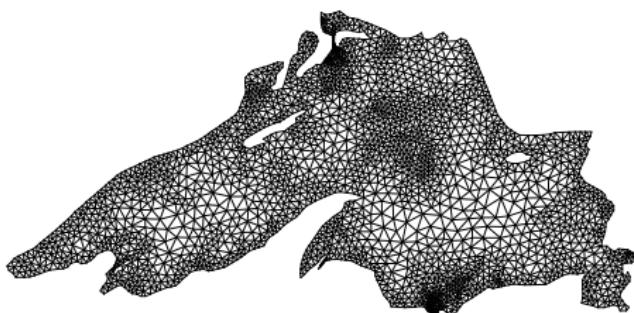


muitos elementos



solução mais **precisa**

**Otimalidade × Poucos Elementos**

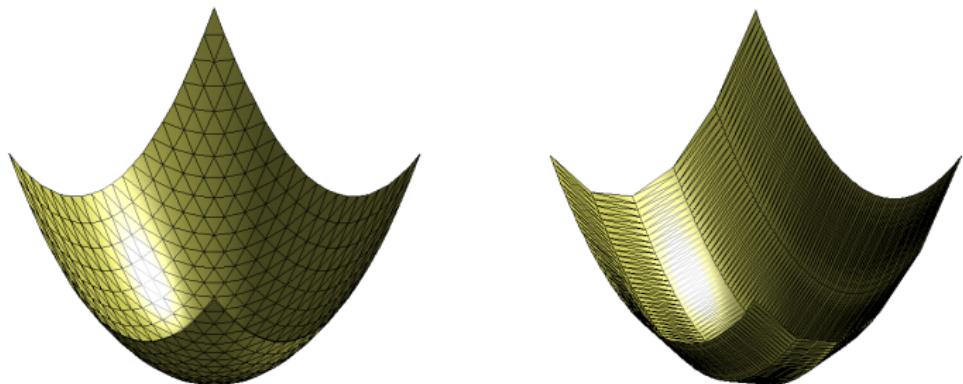


$$\theta_{\min} = 34,2^\circ$$

$$\#\triangle = 4886$$

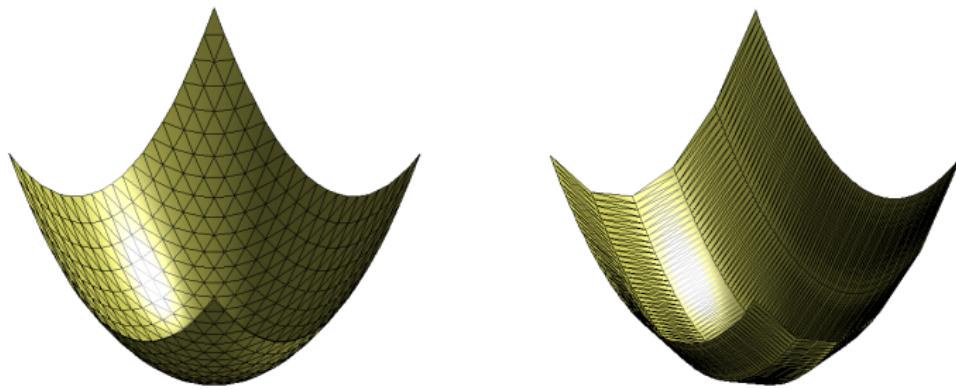
## Qualidade da malha importa

Dois parabolóides:  $z = x^2 + y^2$  com  $(x, y) \in [-1, 1]^2$  com 800 triângulos.



## Qualidade da malha importa

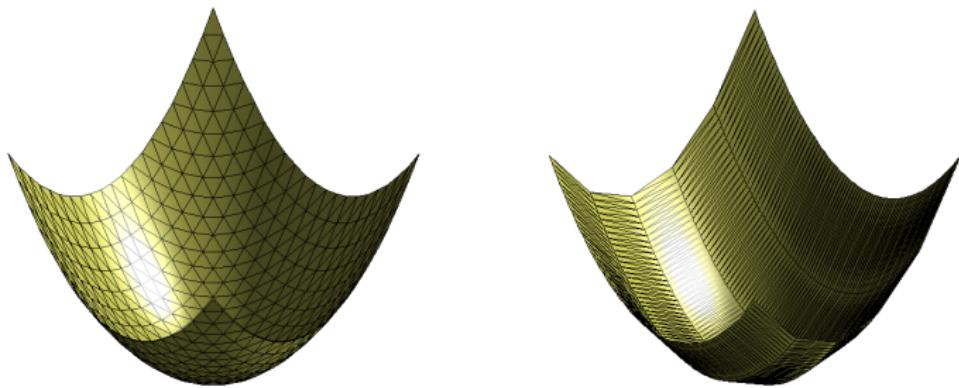
Dois parabolóides:  $z = x^2 + y^2$  com  $(x, y) \in [-1, 1]^2$  com 800 triângulos.



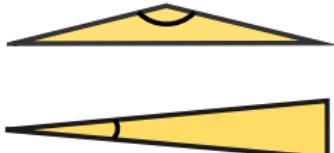
$$\|\nabla(f - \tilde{f})\|_\infty \leq \frac{3h}{\sin(\theta_{\min})} \|D^2 f\|_\infty$$

## Qualidade da malha importa

Dois parabolóides:  $z = x^2 + y^2$  com  $(x, y) \in [-1, 1]^2$  com 800 triângulos.



$$\|\nabla(f - \tilde{f})\|_\infty \leq \frac{3h}{\sin(\theta_{\min})} \|D^2 f\|_\infty$$

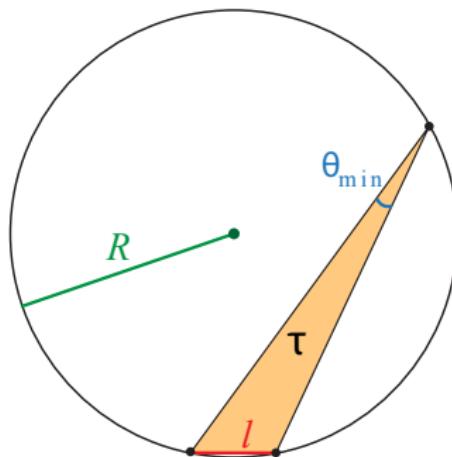


**Triângulos magros** causam erro de interpolação de derivadas.

# Medida de Qualidade

## Razão circunraio-aresta

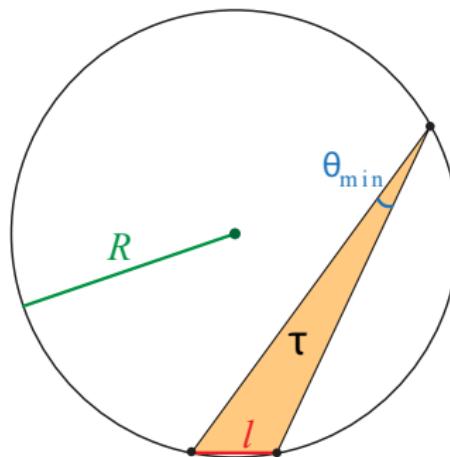
$$\rho(\tau) = \frac{R}{l} = \frac{1}{2 \operatorname{sen}(\theta_{min})}$$



# Medida de Qualidade

## Razão circunraio-aresta

$$\rho(\tau) = \frac{R}{l} = \frac{1}{2 \operatorname{sen}(\theta_{min})}$$

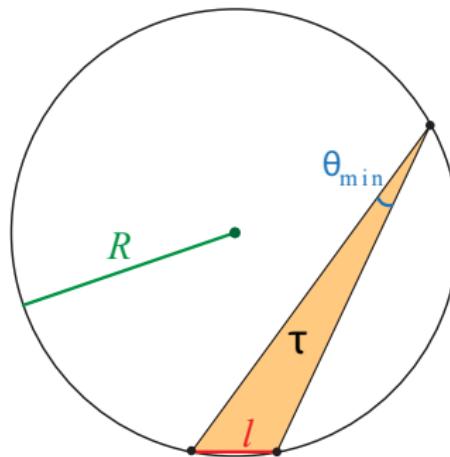


maior  $\theta_{min} \iff$  menor  $\rho(\tau)$

# Medida de Qualidade

## Razão circunraio-aresta

$$\rho(\tau) = \frac{R}{l} = \frac{1}{2 \operatorname{sen}(\theta_{min})}$$



maior  $\theta_{min} \iff$  menor  $\rho(\tau)$

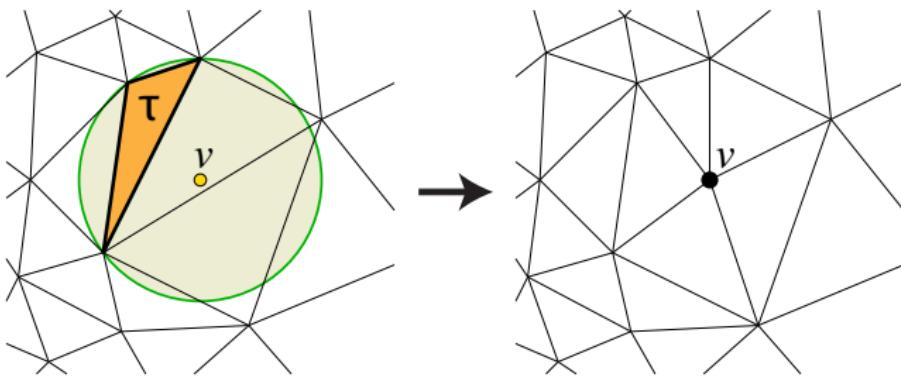
minimizar  $\rho \iff$  maximizar  $\theta_{min} \iff$  triangulação de Delaunay

## Refinamento de Delaunay

- ▶ P. Chew (1989), J. Ruppert (1995), J.R. Shewchuk (2001)

# Refinamento de Delaunay

- P. Chew (1989), J. Ruppert (1995), J.R. Shewchuk (2001)

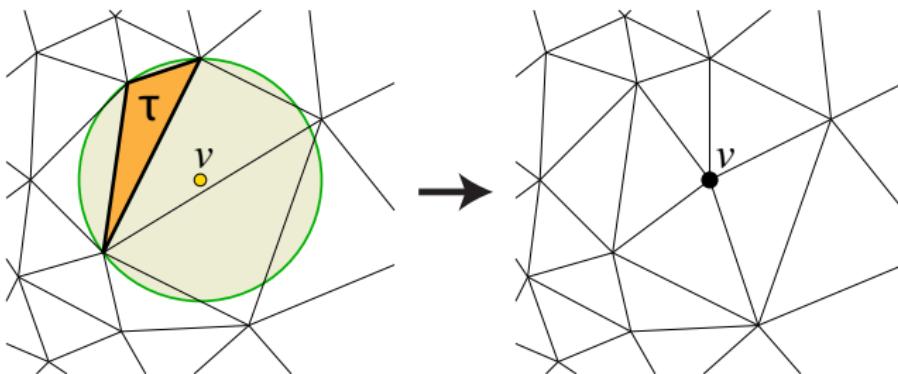


## Ideia Principal:

- Inserir vértices no circuncentro de triângulos magros.

# Refinamento de Delaunay

- P. Chew (1989), J. Ruppert (1995), J.R. Shewchuk (2001)

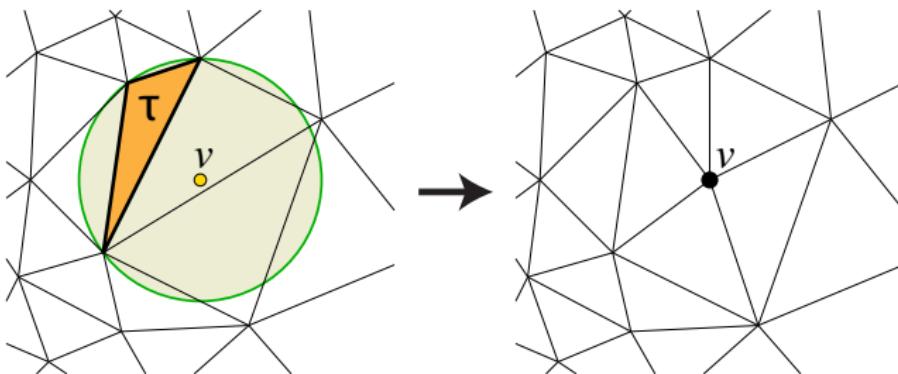


## Ideia Principal:

- Inserir vértices no circuncentro de triângulos magros.
- Algoritmo incremental de Delaunay.

# Refinamento de Delaunay

- ▶ P. Chew (1989), J. Ruppert (1995), J.R. Shewchuk (2001)

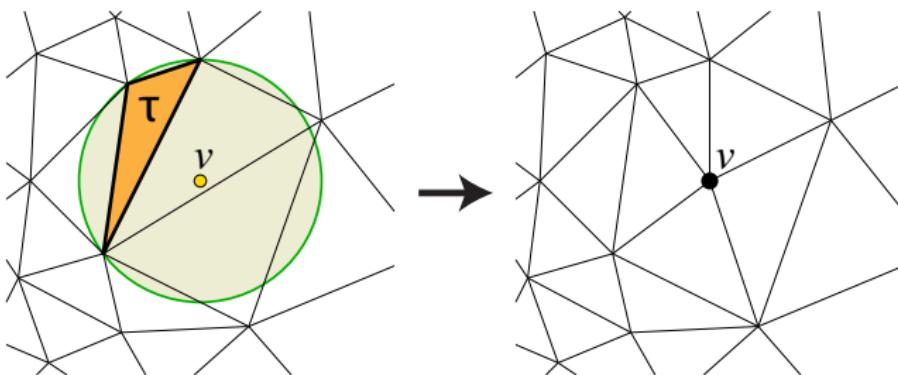


## Ideia Principal:

- ▶ Inserir vértices no circuncentro de triângulos magros.
- ▶ Algoritmo incremental de Delaunay.
- ▶ Quão magro? (medida de qualidade)

# Refinamento de Delaunay

- P. Chew (1989), J. Ruppert (1995), J.R. Shewchuk (2001)

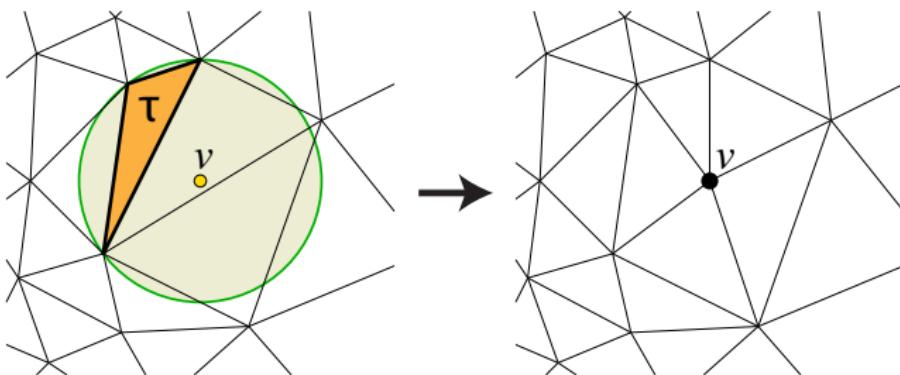


## Ideia Principal:

- Inserir vértices no circuncentro de triângulos magros.
- Algoritmo incremental de Delaunay.
- Quão magro? (medida de qualidade)
- Novas arestas têm pelo menos o comprimento do circunraio de  $\tau$

# Refinamento de Delaunay

- P. Chew (1989), J. Ruppert (1995), J.R. Shewchuk (2001)



## Ideia Principal:

- Inserir vértices no circuncentro de triângulos magros.
- Algoritmo incremental de Delaunay.
- Quão magro? (medida de qualidade)
- Novas arestas têm pelo menos o comprimento do circunraio de  $\tau$
- Garantia de término!

# Refinamento de Delaunay

Entrada

$$(\mathcal{G}, \bar{\rho})$$

# Refinamento de Delaunay

## Entrada

$$(\mathcal{G}, \bar{\rho})$$

- ▶  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$  é um *grafo planar de segmentos de reta* (GPRS) que satisfaçõas as seguintes propriedades:

# Refinamento de Delaunay

## Entrada

$$(\mathcal{G}, \bar{\rho})$$

- ▶  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$  é um *grafo planar de segmentos de reta* (GPRS) que satisfaz as seguintes propriedades:
  - (P1)  $\mathcal{G}$  é um complexo simplicial de dimensão 1

# Refinamento de Delaunay

## Entrada

$$(\mathcal{G}, \bar{\rho})$$

- ▶  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$  é um *grafo planar de segmentos de reta* (GPRS) que satisfaz as seguintes propriedades:
  - (P1)  $\mathcal{G}$  é um complexo simplicial de dimensão 1
  - (P2)  $\partial(FC(\mathcal{V}))$  é a união de um subconjunto de  $\mathcal{S}$

# Refinamento de Delaunay

## Entrada

$$(\mathcal{G}, \bar{\rho})$$

- ▶  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$  é um *grafo planar de segmentos de reta* (GPRS) que satisfaz as seguintes propriedades:
  - (P1)  $\mathcal{G}$  é um complexo simplicial de dimensão 1
  - (P2)  $\partial(FC(\mathcal{V}))$  é a união de um subconjunto de  $\mathcal{S}$
  - (R1) segmentos de  $\mathcal{S}$  não formam ângulos  $< 90^\circ$  (a remover!)

# Refinamento de Delaunay

Entrada

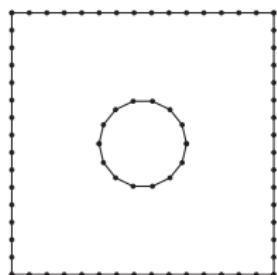
$$(\mathcal{G}, \bar{\rho})$$

- $\mathcal{G} = (\mathcal{V}, \mathcal{S})$  é um *grafo planar de segmentos de reta* (GPRS) que satisfaz as seguintes propriedades:

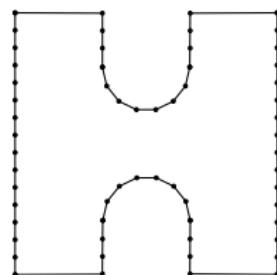
(P1)  $\mathcal{G}$  é um complexo simplicial de dimensão 1

(P2)  $\partial(FC(\mathcal{V}))$  é a união de um subconjunto de  $\mathcal{S}$

(R1) segmentos de  $\mathcal{S}$  não formam ângulos  $< 90^\circ$  (a remover!)



satisfaz (P2)



não satisfaz (P2)

# Refinamento de Delaunay

Entrada

$$(\mathcal{G}, \bar{\rho})$$

- ▶  $\tau$  é um triângulo magro  $\iff \rho(\tau) > \bar{\rho}$

# Refinamento de Delaunay

## Entrada

$$(\mathcal{G}, \bar{\rho})$$

- ▶  $\tau$  é um triângulo magro  $\iff \rho(\tau) > \bar{\rho}$
- ▶  $\bar{\rho}$  é um parâmetro de otimalidade  $\iff \rho(\tau) \leq \bar{\rho}$ , para todo  $\tau$

# Refinamento de Delaunay

## Entrada

$$(\mathcal{G}, \bar{\rho})$$

- ▶  $\tau$  é um triângulo magro  $\iff \rho(\tau) > \bar{\rho}$
- ▶  $\bar{\rho}$  é um parâmetro de otimalidade  $\iff \rho(\tau) \leq \bar{\rho}$ , para todo  $\tau$
- ▶ garantia de término:  $\bar{\rho} \geq \sqrt{2}$

# Refinamento de Delaunay

Entrada

$$(\mathcal{G}, \bar{\rho})$$

- ▶  $\tau$  é um triângulo magro  $\iff \rho(\tau) > \bar{\rho}$
- ▶  $\bar{\rho}$  é um parâmetro de otimalidade  $\iff \rho(\tau) \leq \bar{\rho}$ , para todo  $\tau$
- ▶ garantia de término:  $\bar{\rho} \geq \sqrt{2}$

**Cotas inferior e superior para os ângulos internos dos triângulos**

$$\theta_{\min} = \arcsen\left(\frac{1}{2\bar{\rho}}\right) \approx 20,7^\circ$$

# Refinamento de Delaunay

Entrada

$$(\mathcal{G}, \bar{\rho})$$

- ▶  $\tau$  é um triângulo magro  $\iff \rho(\tau) > \bar{\rho}$
- ▶  $\bar{\rho}$  é um parâmetro de otimalidade  $\iff \rho(\tau) \leq \bar{\rho}$ , para todo  $\tau$
- ▶ garantia de término:  $\bar{\rho} \geq \sqrt{2}$

**Cotas inferior e superior para os ângulos internos dos triângulos**

$$\theta_{\min} = \arcsen\left(\frac{1}{2\bar{\rho}}\right) \approx 20,7^\circ$$

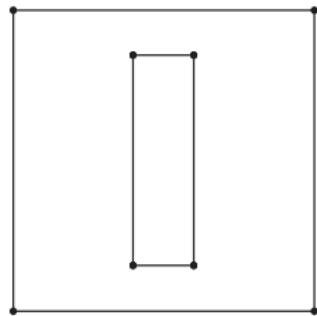
$$\theta_{\max} = 180^\circ - 2\theta_{\min} \approx 139^\circ$$

# Refinamento de Delaunay

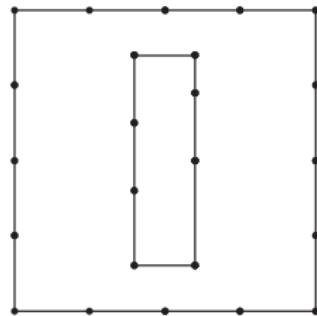
Saída

## Definição (refinamento)

Um GPSR  $\mathcal{G}' = (\mathcal{V}', \mathcal{S}')$  é um *refinamento* do GPSR,  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ , se, e somente se, temos que  $\mathcal{V} \subset \mathcal{V}'$  e que cada segmento em  $\mathcal{S}$  é um segmento em  $\mathcal{S}'$  ou a união de dois ou mais segmentos em  $\mathcal{S}'$ .



GPSR  $\mathcal{G}$



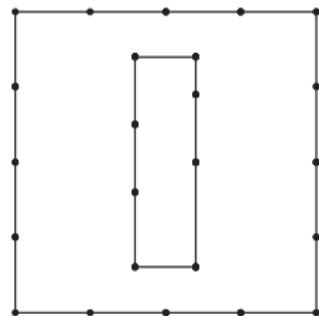
refinamento  $\mathcal{G}'$

# Refinamento de Delaunay

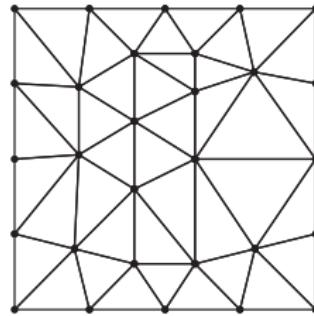
Saída

## Definição (triangulação restrita do GPSR)

Dado um GPSR  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ , uma *triangulação restrita*,  $\mathcal{T}(\mathcal{G})$ , de  $\mathcal{G}$  é uma triangulação do subconjunto,  $\mathcal{V}$ , de pontos de  $\mathbb{E}^2$  tal que todo segmento em  $\mathcal{S}$  é uma aresta de  $\mathcal{T}(\mathcal{G})$ .



GPSR  $\mathcal{G}$



$\mathcal{T}(\mathcal{G})$

# Refinamento de Delaunay

## Saída

Dado um GPSR  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ , a saída produzida pelo refinamento será:

# Refinamento de Delaunay

## Saída

Dado um GPSR  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ , a saída produzida pelo refinamento será:

- ▶  $\mathcal{TD}(P)$  de um subconjunto de pontos  $P \subset \mathbb{E}^2$

# Refinamento de Delaunay

## Saída

Dado um GPSR  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ , a saída produzida pelo refinamento será:

- ▶  $\mathcal{T}\mathcal{D}(P)$  de um subconjunto de pontos  $P \subset \mathbb{E}^2$
- ▶  $\mathcal{V} \subseteq P$

# Refinamento de Delaunay

## Saída

Dado um GPSR  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ , a saída produzida pelo refinamento será:

- ▶  $\mathcal{TD}(P)$  de um subconjunto de pontos  $P \subset \mathbb{E}^2$
- ▶  $\mathcal{V} \subseteq P$
- ▶ todo segmento de  $\mathcal{S}$  é uma aresta ou a união de arestas de  $\mathcal{TD}(P)$

# Refinamento de Delaunay

## Saída

Dado um GPSR  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ , a saída produzida pelo refinamento será:

- ▶  $\mathcal{T}\mathcal{D}(P)$  de um subconjunto de pontos  $P \subset \mathbb{E}^2$
- ▶  $\mathcal{V} \subseteq P$
- ▶ todo segmento de  $\mathcal{S}$  é uma aresta ou a união de arestas de  $\mathcal{T}\mathcal{D}(P)$
- ▶ o menor ângulo de qualquer triângulo de  $\mathcal{T}\mathcal{D}(P)$  é  $\geq \text{arcsen}\left(\frac{1}{2\bar{\rho}}\right)$

# Refinamento de Delaunay

## Saída

Dado um GPSR  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ , a saída produzida pelo refinamento será:

- ▶  $\mathcal{TD}(P)$  de um subconjunto de pontos  $P \subset \mathbb{E}^2$
- ▶  $\mathcal{V} \subseteq P$
- ▶ todo segmento de  $\mathcal{S}$  é uma aresta ou a união de arestas de  $\mathcal{TD}(P)$
- ▶ o menor ângulo de qualquer triângulo de  $\mathcal{TD}(P)$  é  $\geq \text{arcsen}\left(\frac{1}{2\rho}\right)$

A triangulação  $\mathcal{TD}(P)$  é dita ser uma **triangulação de Delaunay conforme (TDC) de  $\mathcal{G}$** , pois  $\mathcal{TD}(P)$  contém o GRPS  $\mathcal{G}$  ou um refinamento  $\mathcal{G}'$  de  $\mathcal{G}$ .

# Refinamento de Delaunay

## Saída

Dado um GPSR  $\mathcal{G} = (\mathcal{V}, \mathcal{S})$ , a saída produzida pelo refinamento será:

- ▶  $\mathcal{T}\mathcal{D}(P)$  de um subconjunto de pontos  $P \subset \mathbb{E}^2$
- ▶  $\mathcal{V} \subseteq P$
- ▶ todo segmento de  $\mathcal{S}$  é uma aresta ou a união de arestas de  $\mathcal{T}\mathcal{D}(P)$
- ▶ o menor ângulo de qualquer triângulo de  $\mathcal{T}\mathcal{D}(P)$  é  $\geq \text{arcsen}\left(\frac{1}{2\rho}\right)$

A triangulação  $\mathcal{T}\mathcal{D}(P)$  é dita ser uma **triangulação de Delaunay conforme (TDC) de  $\mathcal{G}$** , pois  $\mathcal{T}\mathcal{D}(P)$  contém o GRPS  $\mathcal{G}$  ou um refinamento  $\mathcal{G}'$  de  $\mathcal{G}$ .

Resumindo,  $\mathcal{T}\mathcal{D}(P)$  é uma triangulação restrita  $\mathcal{T}(\mathcal{G}')$ .

# Refinamento de Delaunay

## Algoritmo

---

**Algoritmo** REFINAMENTODELAUNAY( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$   $\{\mathcal{V} \text{ é o conjunto de vértices de } \mathcal{G}\}$
  - 2: calcule  $\mathcal{TD}(Q)$
  - 3: **enquanto**  $\mathcal{TD}(Q)$ : (i) não é uma TDC ou (ii) o seu menor ângulo interno é menor que  $\text{arcsen}\left(\frac{1}{2\bar{\rho}}\right)$  **faça**  
4:     escolha  $p \in FC(\mathcal{V})$   
5:     faça  $Q \leftarrow Q \cup \{p\}$   
6:     recalcule  $\mathcal{TD}(Q)$
  - 7: **fim enquanto**
-

# Refinamento de Delaunay

## Algoritmo

---

**Algoritmo** REFINAMENTODELAUNAY( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$   $\{\mathcal{V} \text{ é o conjunto de vértices de } \mathcal{G}\}$
  - 2: calcule  $\mathcal{TD}(Q)$
  - 3: **enquanto**  $\mathcal{TD}(Q)$ : (i) não é uma TDC ou (ii) o seu menor ângulo interno é menor que  $\text{arcsen}\left(\frac{1}{2\bar{\rho}}\right)$  **faça**  
4:     escolha  $p \in FC(\mathcal{V})$   
5:     faça  $Q \leftarrow Q \cup \{p\}$   
6:     recalcule  $\mathcal{TD}(Q)$
  - 7: **fim enquanto**
- 

- ▶ Como escolher  $p$ ?

# Refinamento de Delaunay

## Algoritmo

---

**Algoritmo** REFINAMENTODELAUNAY( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$   $\{\mathcal{V} \text{ é o conjunto de vértices de } \mathcal{G}\}$
  - 2: calcule  $\mathcal{TD}(Q)$
  - 3: **enquanto**  $\mathcal{TD}(Q)$ : (i) não é uma TDC ou (ii) o seu menor ângulo interno é menor que  $\text{arcsen}\left(\frac{1}{2\bar{\rho}}\right)$  **faça**  
    4:     escolha  $p \in FC(\mathcal{V})$   
    5:     faça  $Q \leftarrow Q \cup \{p\}$   
    6:     recalcule  $\mathcal{TD}(Q)$
  - 7: **fim enquanto**
- 

► Como escolher  $p$ ? Basta posicionar  $p$  no circuncentro do  $\triangle$ .

# Refinamento de Delaunay

## Algoritmo

---

**Algoritmo** REFINAMENTODELAUNAY( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$   $\{\mathcal{V} \text{ é o conjunto de vértices de } \mathcal{G}\}$
  - 2: calcule  $\mathcal{TD}(Q)$
  - 3: **enquanto**  $\mathcal{TD}(Q)$ : (i) não é uma TDC ou (ii) o seu menor ângulo interno é menor que  $\text{arcsen}\left(\frac{1}{2\bar{\rho}}\right)$  **faça**  
    4:     escolha  $p \in FC(\mathcal{V})$   
    5:     faça  $Q \leftarrow Q \cup \{p\}$   
    6:     recalcule  $\mathcal{TD}(Q)$
  - 7: **fim enquanto**
- 

- Como escolher  $p$ ? Basta posicionar  $p$  no circuncentro do  $\triangle$ .
- O algoritmo termina?

# Refinamento de Delaunay

## Algoritmo

---

**Algoritmo** REFINAMENTODELAUNAY( $\mathcal{G}, \bar{\rho}$ )

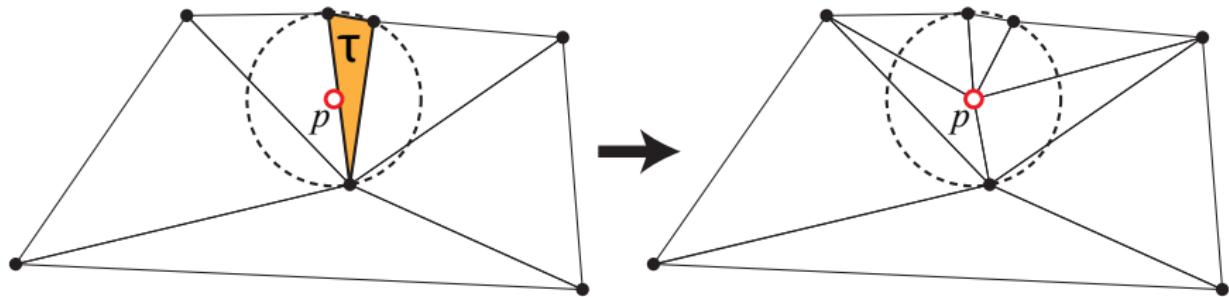
---

- 1:  $Q \leftarrow \mathcal{V}$   $\{\mathcal{V} \text{ é o conjunto de vértices de } \mathcal{G}\}$
  - 2: calcule  $\mathcal{TD}(Q)$
  - 3: **enquanto**  $\mathcal{TD}(Q)$ : (i) não é uma TDC ou (ii) o seu menor ângulo interno é menor que  $\text{arcsen}\left(\frac{1}{2\bar{\rho}}\right)$  **faça**  
4:     escolha  $p \in FC(\mathcal{V})$   
5:     faça  $Q \leftarrow Q \cup \{p\}$   
6:     recalcule  $\mathcal{TD}(Q)$
  - 7: **fim enquanto**
- 

- Como escolher  $p$ ? Basta posicionar  $p$  no circuncentro do  $\triangle$ .
- O algoritmo termina? Sim, pelo Lema do Empacotamento.

# Refinamento de Delaunay

## Inserção de Vértice



vértices são inseridos no circuncentro de triângulos magros

# Refinamento de Delaunay

## Término do Algoritmo

- ▶ Seja  $\alpha$  o comprimento da menor aresta de  $\mathcal{TD}(Q)$

# Refinamento de Delaunay

## Término do Algoritmo

- ▶ Seja  $\alpha$  o comprimento da menor aresta de  $\mathcal{TD}(Q)$
- ▶ Seja  $R$  o circunraio de  $\tau$

$$d(p, Q) \geq R = \rho(\tau) \cdot l > \bar{\rho} \cdot \alpha = \lambda$$

# Refinamento de Delaunay

## Término do Algoritmo

- ▶ Seja  $\alpha$  o comprimento da menor aresta de  $\mathcal{TD}(Q)$
- ▶ Seja  $R$  o circunraio de  $\tau$

$$d(p, Q) \geq R = \rho(\tau) \cdot l > \bar{\rho} \cdot \alpha = \lambda$$

- ▶ Lema do Empacotamento

$$d(u, v) > \lambda, \forall u, v \in Q \Rightarrow |Q| < \infty$$

# Refinamento de Delaunay

Erro no Algoritmo!?

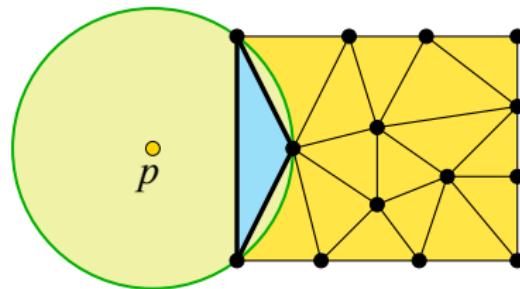
O que acontece se o circuncentro  $p \notin FC(\mathcal{V})$ ?

# Refinamento de Delaunay

Erro no Algoritmo!?

O que acontece se o circuncentro  $p \notin FC(\mathcal{V})$ ?

- Deixa de ser um TDC!

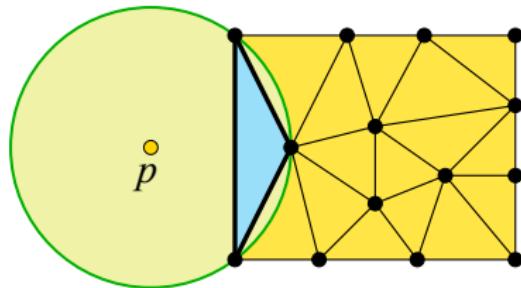


# Refinamento de Delaunay

Erro no Algoritmo!?

O que acontece se o circuncentro  $p \notin FC(\mathcal{V})$ ?

- Deixa de ser um TDC!



Precisamos fazer uma (simples) mudança no algoritmo para que as condições abaixo sejam satisfeitas:

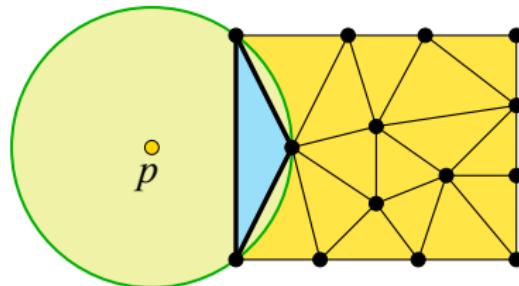
- (i)  $\mathcal{T}\mathcal{D}(Q)$  é uma TDC
- (ii) o menor ângulo interno de  $\mathcal{T}\mathcal{D}(Q)$  é  $\geq \text{arcsen} \left( \frac{1}{2\bar{\rho}} \right)$

# Refinamento de Delaunay

Erro no Algoritmo!?

O que acontece se o circuncentro  $p \notin FC(\mathcal{V})$ ?

- Deixa de ser um TDC!



Precisamos fazer uma (simples) mudança no algoritmo para que as condições abaixo sejam satisfeitas:

- (i)  $\mathcal{T}\mathcal{D}(Q)$  é uma TDC
- (ii) o menor ângulo interno de  $\mathcal{T}\mathcal{D}(Q)$  é  $\geq \text{arcsen} \left( \frac{1}{2\bar{\rho}} \right)$

Devemos garantir que a condição (i) seja satisfeita antes de (ii).

# Refinamento de Delaunay

## Invasão

### Definição (segmento invadido)

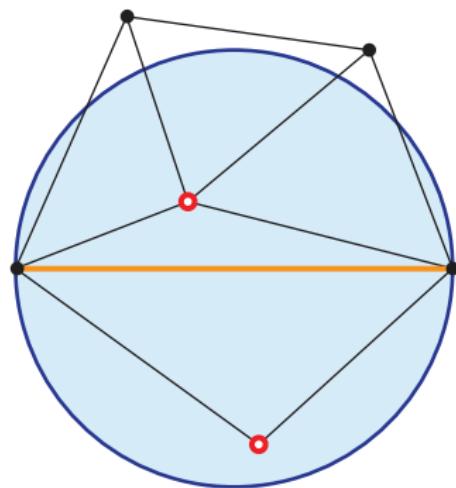
Um segmento  $s$  é *invadido* por um vértice  $v$  de  $\mathcal{TD}(Q)$ , se  $v$  reside no círculo que possui  $s$  como diâmetro, mas não é um ponto extremo de  $s$ .

# Refinamento de Delaunay

## Invasão

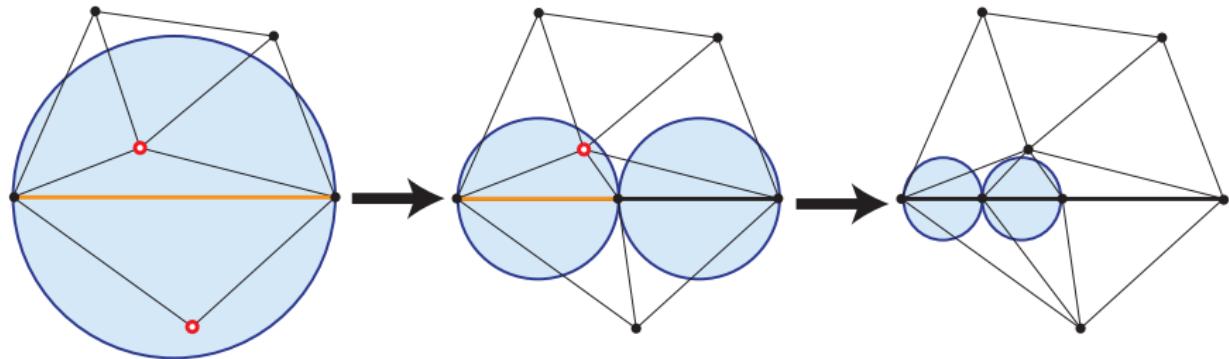
### Definição (segmento invadido)

Um segmento  $s$  é *invadido* por um vértice  $v$  de  $\mathcal{TD}(Q)$ , se  $v$  reside no círculo que possui  $s$  como diâmetro, mas não é um ponto extremo de  $s$ .



# Refinamento de Delaunay

## Inserção de Vértice Revisitada



um segmento invadido é dividido em subsegmentos adicionando um novo vértice em seu ponto médio

# Refinamento de Delaunay

## Inserção de Vértice Revisitada

- ▶  $E = \{\text{conjunto de subsegmentos}\}$

# Refinamento de Delaunay

## Inserção de Vértice Revisitada

- ▶  $E = \{\text{conjunto de subsegmentos}\}$
- ▶  $E \leftarrow S$

# Refinamento de Delaunay

## Inserção de Vértice Revisitada

- ▶  $E = \{\text{conjunto de subsegmentos}\}$
- ▶  $E \leftarrow S$
- ▶ se  $s \in E$  é um **segmento invadido** chame o algoritmo:

# Refinamento de Delaunay

## Inserção de Vértice Revisitada

- ▶  $E = \{\text{conjunto de subsegmentos}\}$
- ▶  $E \leftarrow S$
- ▶ se  $s \in E$  é um **segmento invadido** chame o algoritmo:

---

### **Algoritmo** DIVIDESUBSEGMENTO( $s, Q, E$ )

---

- 1: calcule o ponto médio  $p$  de  $s$
  - 2: troque  $s$  por seus dois subsegmentos (metades) em  $E$
  - 3: **devolva**  $p$
-

# Refinamento de Delaunay

## Inserção de Vértice Revisitada

- ▶ Lembra do problema em que o circuncentro  $p \notin FC(\mathcal{V})$ ?

# Refinamento de Delaunay

## Inserção de Vértice Revisitada

- ▶ Lembra do problema em que o circuncentro  $p \notin FC(\mathcal{V})$ ?
- ▶ Temos outro problema,  $p$  pode ficar tão próximo possível de  $FC(\mathcal{V})$ !  
(não vale o Lema do Empacotamento)

# Refinamento de Delaunay

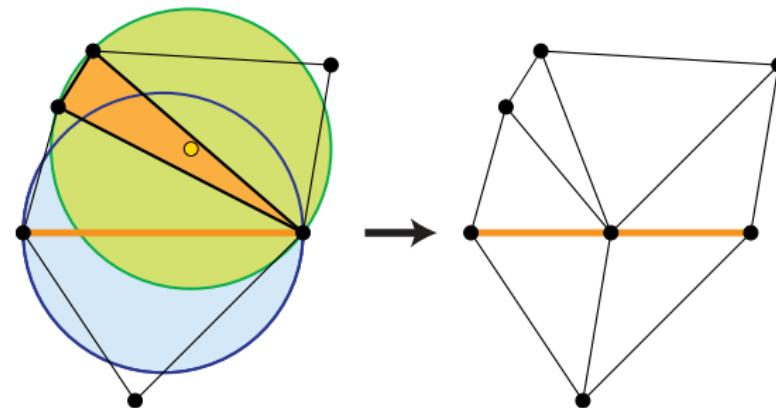
## Inserção de Vértice Revisitada

- ▶ Lembra do problema em que o circuncentro  $p \notin FC(\mathcal{V})$ ?
- ▶ Temos outro problema,  $p$  pode ficar tão próximo possível de  $FC(\mathcal{V})$ !  
(não vale o Lema do Empacotamento)
- ▶ Como resolver esses problemas?

# Refinamento de Delaunay

## Inserção de Vértice Revisitada

- ▶ Lembra do problema em que o circuncentro  $p \notin FC(\mathcal{V})$ ?
- ▶ Temos outro problema,  $p$  pode ficar tão próximo possível de  $FC(\mathcal{V})$ !  
(não vale o Lema do Empacotamento)
- ▶ Como resolver esses problemas?



segmentos invadidos têm prioridade sobre triângulos magros

# Refinamento de Delaunay

## Inserção de Vértice Revisitada

- se  $\tau$  for um **triângulo magro** então chame o algoritmo:

---

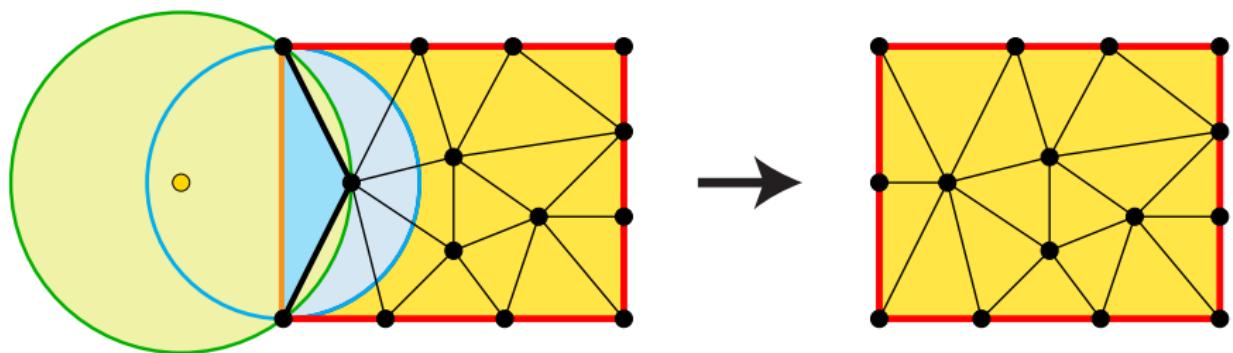
### **Algoritmo** DIVIDETRIÂNGULO( $\tau, Q, E$ )

---

- 1: calcule o circuncentro  $p$  de  $\tau$
  - 2: **se** existir um subsegmento  $s \in E$  invadido por  $p$  **então**
  - 3:    $p \leftarrow \text{DIVIDESUBSEGMENTO}(s, Q, E)$
  - 4: **fim se**
  - 5: **devolva**  $p$
-

# Refinamento de Delaunay

Inserção de Vértice Revisitada



# Algoritmo de Ruppert

---

**Algoritmo** GERA MALHA( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$
- 2: calcule  $\mathcal{TD}(Q)$
- 3:  $E \leftarrow$  todos os subsegmentos invadidos de  $\mathcal{G}$
- 4:  $R \leftarrow$  todos os triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 5: **enquanto**  $E \neq \emptyset$  **or**  $R \neq \emptyset$  **faça**
- 6:   **enquanto**  $E \neq \emptyset$  **faça**
- 7:     retire um subsegmento  $s$  de  $E$
- 8:      $p \leftarrow \text{DIVIDESUBSEGMENTO}(s, Q, E)$
- 9:     insira  $p$  em  $\mathcal{TD}(Q)$
- 10:     $Q \leftarrow Q \cup \{p\}$
- 11:   **fim enquanto**
- 12:   **se**  $R \neq \emptyset$  **então**
- 13:     retire um triângulo  $s$  de  $E$
- 14:      $p \leftarrow \text{DIVIDETRIÂNGULO}(\tau, Q, E)$
- 15:     insira  $p$  em  $\mathcal{TD}(Q)$
- 16:      $Q \leftarrow Q \cup \{p\}$
- 17:   **fim se**
- 18:   remova de  $R$  triângulos que não mais existem em  $\mathcal{TD}(Q)$
- 19:   insira em  $E$  novos subsegmentos invadidos por um vértice de  $Q$
- 20:   insira em  $R$  novos triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 21: **fim enquanto**
- 22: **devolva**  $\mathcal{TD}(Q)$

---

# Algoritmo de Ruppert

---

**Algoritmo** GERA MALHA( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$
- 2: calcule  $\mathcal{TD}(Q)$
- 3:  $E \leftarrow$  todos os subsegmentos invadidos de  $\mathcal{G}$
- 4:  $R \leftarrow$  todos os triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 5: enquanto  $E \neq \emptyset$  or  $R \neq \emptyset$  faça
- 6:   enquanto  $E \neq \emptyset$  faça
- 7:     retire um subsegmento  $s$  de  $E$
- 8:      $p \leftarrow \text{DIVIDESUBSEGMENTO}(s, Q, E)$
- 9:     insira  $p$  em  $\mathcal{TD}(Q)$
- 10:     $Q \leftarrow Q \cup \{p\}$
- 11:   fim enquanto
- 12:   se  $R \neq \emptyset$  então
- 13:     retire um triângulo  $s$  de  $E$
- 14:      $p \leftarrow \text{DIVIDETRIÂNGULO}(\tau, Q, E)$
- 15:     insira  $p$  em  $\mathcal{TD}(Q)$
- 16:      $Q \leftarrow Q \cup \{p\}$
- 17:   fim se
- 18:   remova de  $R$  triângulos que não mais existem em  $\mathcal{TD}(Q)$
- 19:   insira em  $E$  novos subsegmentos invadidos por um vértice de  $Q$
- 20:   insira em  $R$  novos triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 21: fim enquanto
- 22: devolva  $\mathcal{TD}(Q)$

---

# Algoritmo de Ruppert

---

**Algoritmo** GERA MALHA( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$
- 2: calcule  $\mathcal{TD}(Q)$
- 3:  $E \leftarrow$  todos os subsegmentos invadidos de  $\mathcal{G}$
- 4:  $R \leftarrow$  todos os triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 5: **enquanto**  $E \neq \emptyset$  or  $R \neq \emptyset$  **faça**
- 6:   **enquanto**  $E \neq \emptyset$  **faça**
- 7:     retire um subsegmento  $s$  de  $E$
- 8:      $p \leftarrow \text{DIVIDESUBSEGMENTO}(s, Q, E)$
- 9:     insira  $p$  em  $\mathcal{TD}(Q)$
- 10:     $Q \leftarrow Q \cup \{p\}$
- 11:   **fim enquanto**
- 12:   **se**  $R \neq \emptyset$  **então**
- 13:     retire um triângulo  $s$  de  $E$
- 14:      $p \leftarrow \text{DIVIDETRIÂNGULO}(\tau, Q, E)$
- 15:     insira  $p$  em  $\mathcal{TD}(Q)$
- 16:      $Q \leftarrow Q \cup \{p\}$
- 17:   **fim se**
- 18:   remova de  $R$  triângulos que não mais existem em  $\mathcal{TD}(Q)$
- 19:   insira em  $E$  novos subsegmentos invadidos por um vértice de  $Q$
- 20:   insira em  $R$  novos triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 21: **fim enquanto**
- 22: **devolva**  $\mathcal{TD}(Q)$

---

# Algoritmo de Ruppert

---

**Algoritmo** GERA MALHA( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$
- 2: calcule  $\mathcal{TD}(Q)$
- 3:  $E \leftarrow$  todos os subsegmentos invadidos de  $\mathcal{G}$
- 4:  $R \leftarrow$  todos os triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 5: **enquanto**  $E \neq \emptyset$  or  $R \neq \emptyset$  **faça**
- 6:   **enquanto**  $E \neq \emptyset$  **faça**
- 7:     retire um subsegmento  $s$  de  $E$
- 8:      $p \leftarrow \text{DIVIDESUBSEGMENTO}(s, Q, E)$
- 9:     insira  $p$  em  $\mathcal{TD}(Q)$
- 10:     $Q \leftarrow Q \cup \{p\}$
- 11:   **fim enquanto**
- 12:   se  $R \neq \emptyset$  então
- 13:     retire um triângulo  $s$  de  $E$
- 14:      $p \leftarrow \text{DIVIDETRIÂNGULO}(\tau, Q, E)$
- 15:     insira  $p$  em  $\mathcal{TD}(Q)$
- 16:      $Q \leftarrow Q \cup \{p\}$
- 17:   **fim se**
- 18:   remova de  $R$  triângulos que não mais existem em  $\mathcal{TD}(Q)$
- 19:   insira em  $E$  novos subsegmentos invadidos por um vértice de  $Q$
- 20:   insira em  $R$  novos triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 21: **fim enquanto**
- 22: devolva  $\mathcal{TD}(Q)$

---

# Algoritmo de Ruppert

---

**Algoritmo** GERA MALHA( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$
- 2: calcule  $\mathcal{TD}(Q)$
- 3:  $E \leftarrow$  todos os subsegmentos invadidos de  $\mathcal{G}$
- 4:  $R \leftarrow$  todos os triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 5: **enquanto**  $E \neq \emptyset$  **or**  $R \neq \emptyset$  **faça**
- 6:   **enquanto**  $E \neq \emptyset$  **faça**
- 7:     retire um subsegmento  $s$  de  $E$
- 8:      $p \leftarrow \text{DIVIDESUBSEGMENTO}(s, Q, E)$
- 9:     insira  $p$  em  $\mathcal{TD}(Q)$
- 10:     $Q \leftarrow Q \cup \{p\}$
- 11:   **fim enquanto**
- 12:   **se**  $R \neq \emptyset$  **então**
- 13:     retire um triângulo  $s$  de  $E$
- 14:      $p \leftarrow \text{DIVIDETRIÂNGULO}(\tau, Q, E)$
- 15:     insira  $p$  em  $\mathcal{TD}(Q)$
- 16:      $Q \leftarrow Q \cup \{p\}$
- 17:   **fim se**
- 18:   remova de  $R$  triângulos que não mais existem em  $\mathcal{TD}(Q)$
- 19:   insira em  $E$  novos subsegmentos invadidos por um vértice de  $Q$
- 20:   insira em  $R$  novos triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 21: **fim enquanto**
- 22: **devolva**  $\mathcal{TD}(Q)$

---

# Algoritmo de Ruppert

---

**Algoritmo** GERA MALHA( $\mathcal{G}, \bar{\rho}$ )

---

- 1:  $Q \leftarrow \mathcal{V}$
- 2: calcule  $\mathcal{TD}(Q)$
- 3:  $E \leftarrow$  todos os subsegmentos invadidos de  $\mathcal{G}$
- 4:  $R \leftarrow$  todos os triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 5: **enquanto**  $E \neq \emptyset$  **or**  $R \neq \emptyset$  **faça**
- 6:   **enquanto**  $E \neq \emptyset$  **faça**
- 7:     retire um subsegmento  $s$  de  $E$
- 8:      $p \leftarrow \text{DIVIDESUBSEGMENTO}(s, Q, E)$
- 9:     insira  $p$  em  $\mathcal{TD}(Q)$
- 10:     $Q \leftarrow Q \cup \{p\}$
- 11:   **fim enquanto**
- 12:   **se**  $R \neq \emptyset$  **então**
- 13:     retire um triângulo  $s$  de  $E$
- 14:      $p \leftarrow \text{DIVIDETRIÂNGULO}(\tau, Q, E)$
- 15:     insira  $p$  em  $\mathcal{TD}(Q)$
- 16:      $Q \leftarrow Q \cup \{p\}$
- 17:   **fim se**
- 18:   **remova de**  $R$  **triângulos que não mais existem em**  $\mathcal{TD}(Q)$
- 19:   **insira em**  $E$  **novos subsegmentos invadidos por um vértice de**  $Q$
- 20:   **insira em**  $R$  **novos triângulos**  $\tau$  **de**  $\mathcal{TD}(Q)$  **tais que**  $\rho(\tau) > \bar{\rho}$
- 21: **fim enquanto**
- 22: **devolva**  $\mathcal{TD}(Q)$

---

# Algoritmo de Ruppert

---

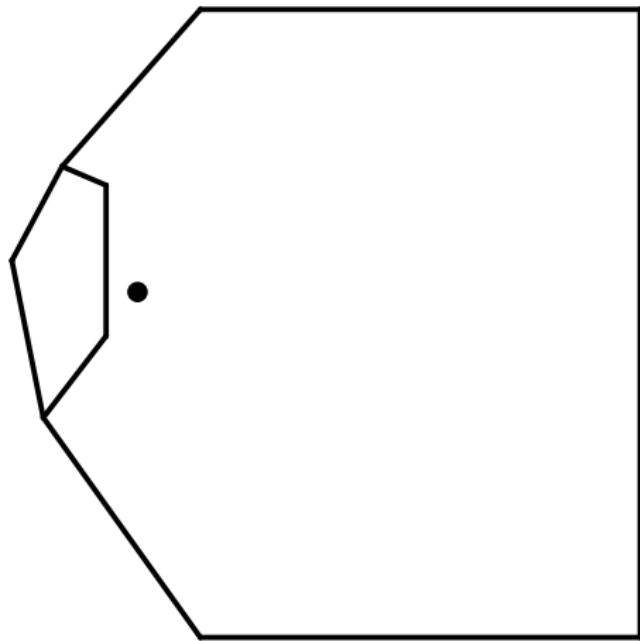
**Algoritmo** GERA MALHA( $\mathcal{G}, \bar{\rho}$ )

---

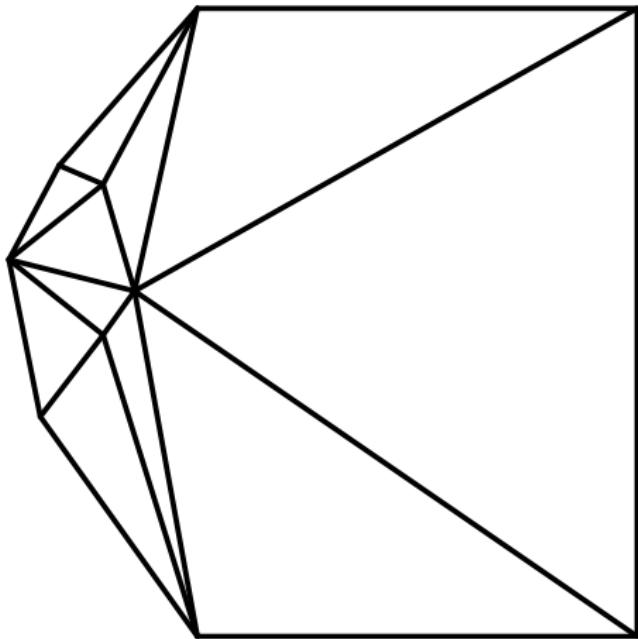
- 1:  $Q \leftarrow \mathcal{V}$
- 2: calcule  $\mathcal{TD}(Q)$
- 3:  $E \leftarrow$  todos os subsegmentos invadidos de  $\mathcal{G}$
- 4:  $R \leftarrow$  todos os triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 5: **enquanto**  $E \neq \emptyset$  **or**  $R \neq \emptyset$  **faça**
- 6:   **enquanto**  $E \neq \emptyset$  **faça**
- 7:     retire um subsegmento  $s$  de  $E$
- 8:      $p \leftarrow \text{DIVIDESUBSEGMENTO}(s, Q, E)$
- 9:     insira  $p$  em  $\mathcal{TD}(Q)$
- 10:     $Q \leftarrow Q \cup \{p\}$
- 11:   **fim enquanto**
- 12:   **se**  $R \neq \emptyset$  **então**
- 13:     retire um triângulo  $s$  de  $E$
- 14:      $p \leftarrow \text{DIVIDETRIÂNGULO}(\tau, Q, E)$
- 15:     insira  $p$  em  $\mathcal{TD}(Q)$
- 16:      $Q \leftarrow Q \cup \{p\}$
- 17:   **fim se**
- 18:   remova de  $R$  triângulos que não mais existem em  $\mathcal{TD}(Q)$
- 19:   insira em  $E$  novos subsegmentos invadidos por um vértice de  $Q$
- 20:   insira em  $R$  novos triângulos  $\tau$  de  $\mathcal{TD}(Q)$  tais que  $\rho(\tau) > \bar{\rho}$
- 21: **fim enquanto**
- 22: **devolva**  $\mathcal{TD}(Q)$

---

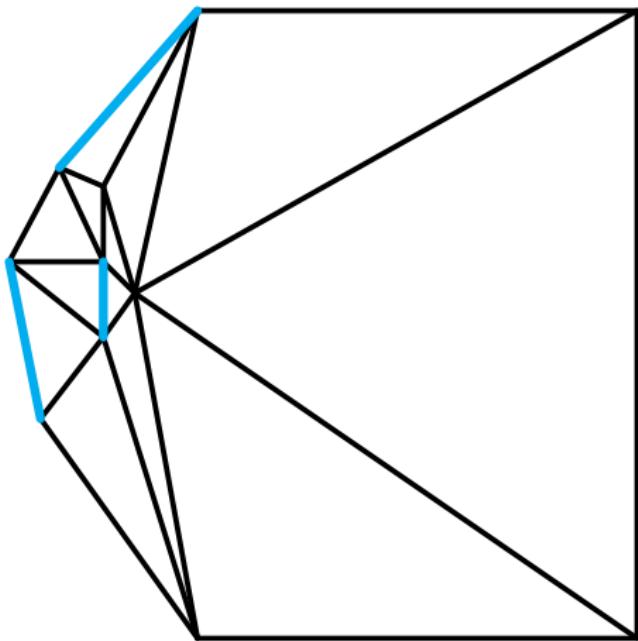
# Algoritmo de Ruppert



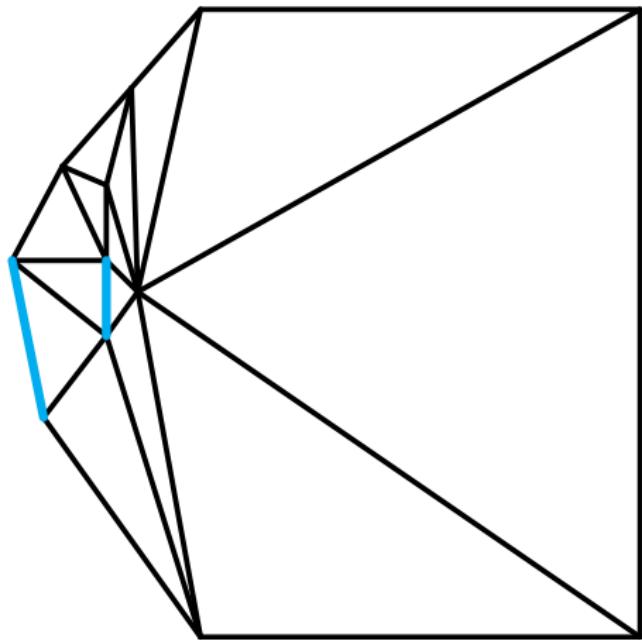
# Algoritmo de Ruppert



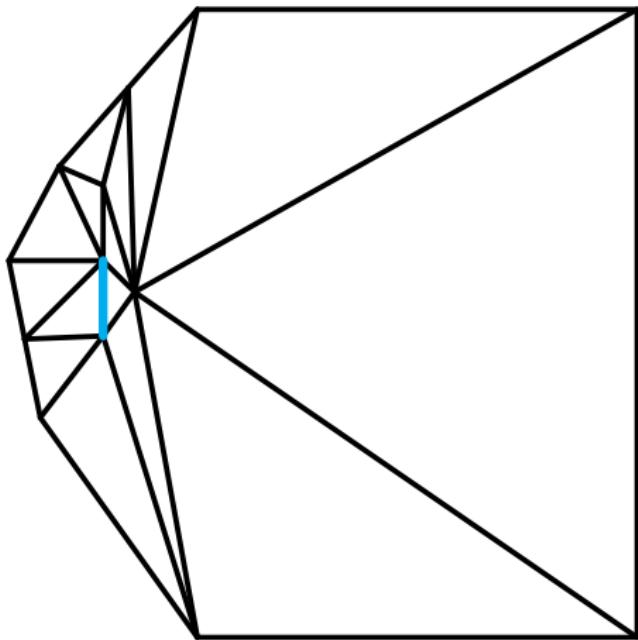
# Algoritmo de Ruppert



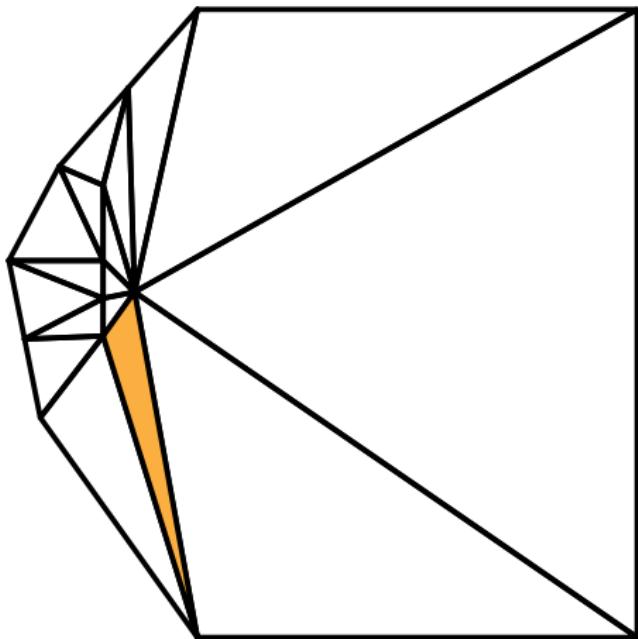
# Algoritmo de Ruppert



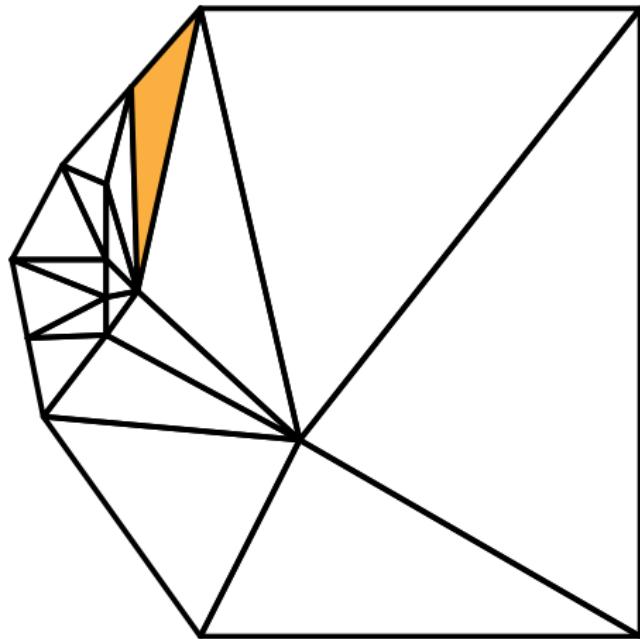
# Algoritmo de Ruppert



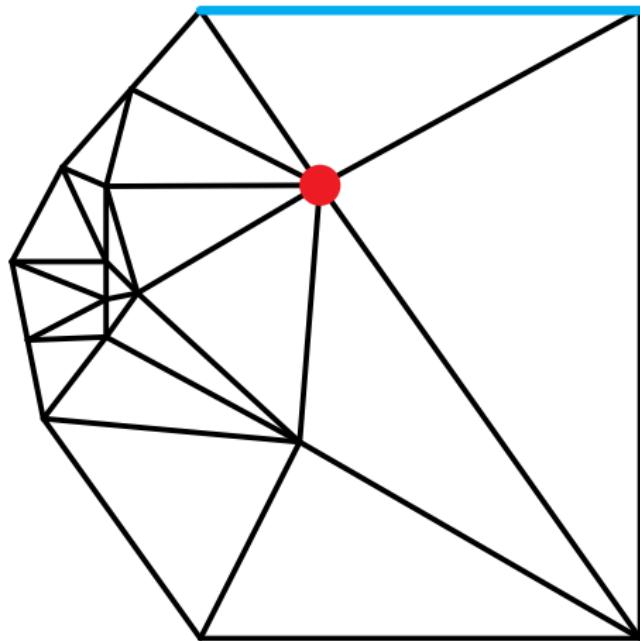
# Algoritmo de Ruppert



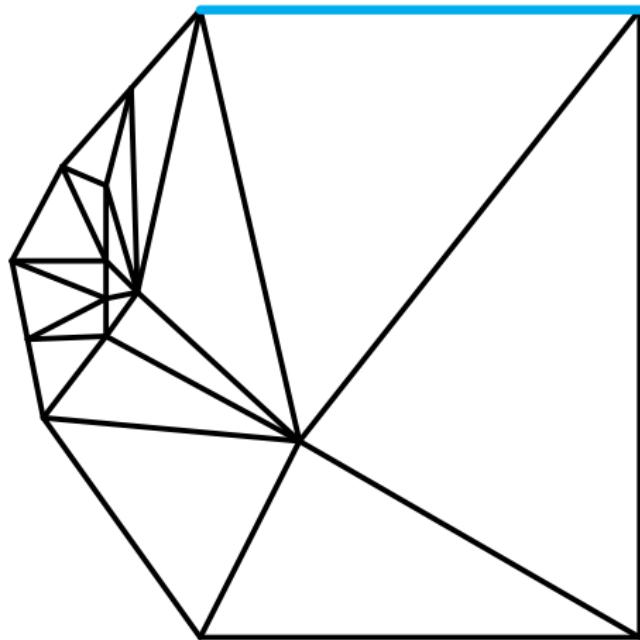
# Algoritmo de Ruppert



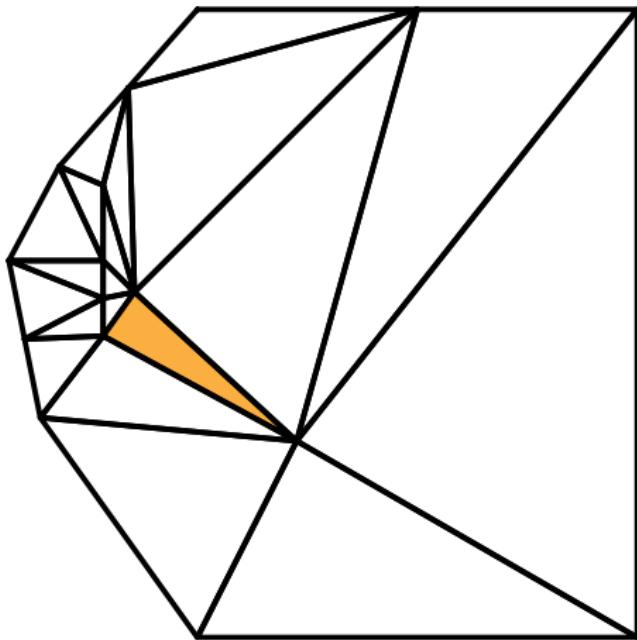
# Algoritmo de Ruppert



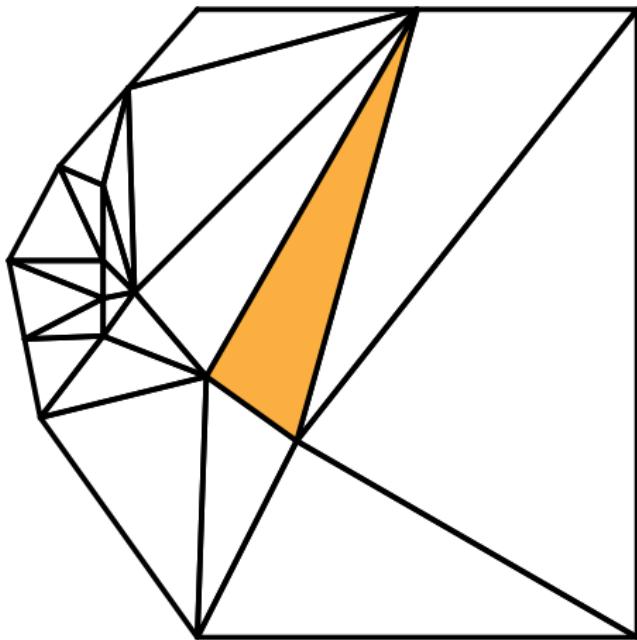
# Algoritmo de Ruppert



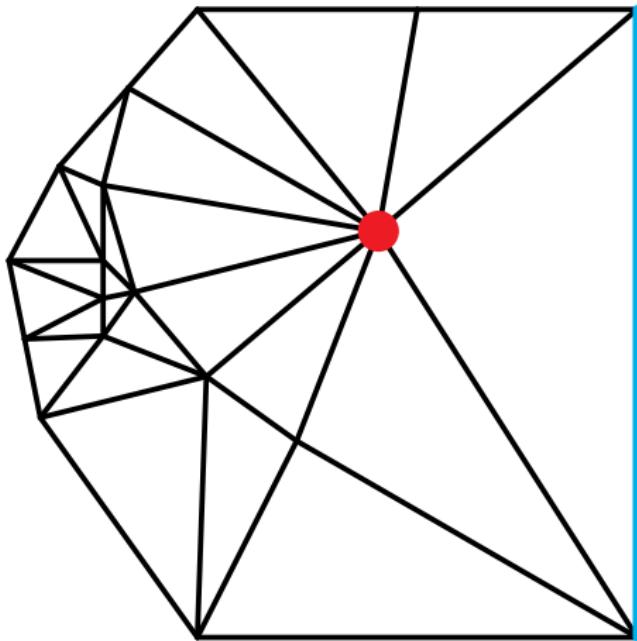
# Algoritmo de Ruppert



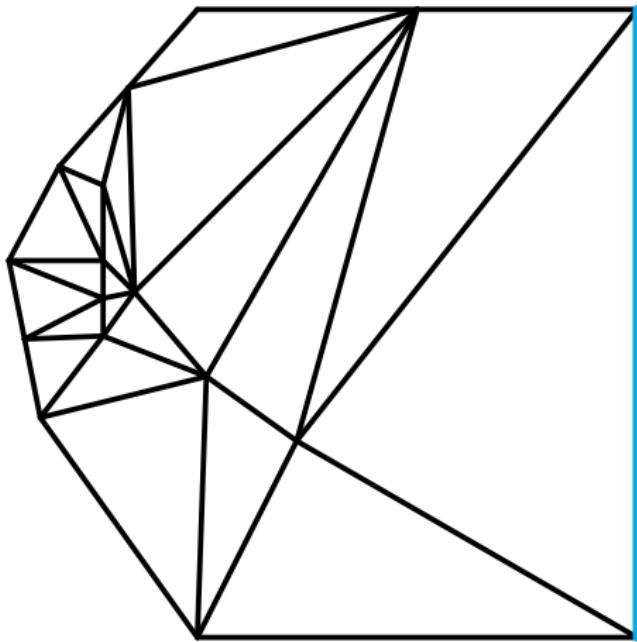
# Algoritmo de Ruppert



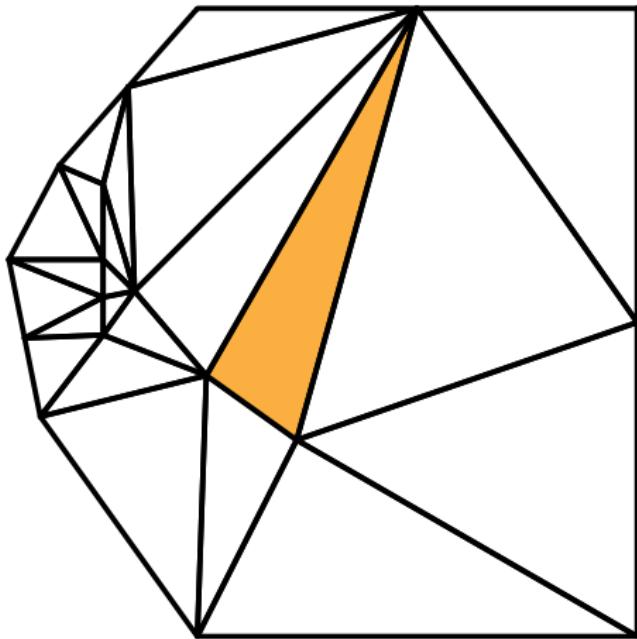
# Algoritmo de Ruppert



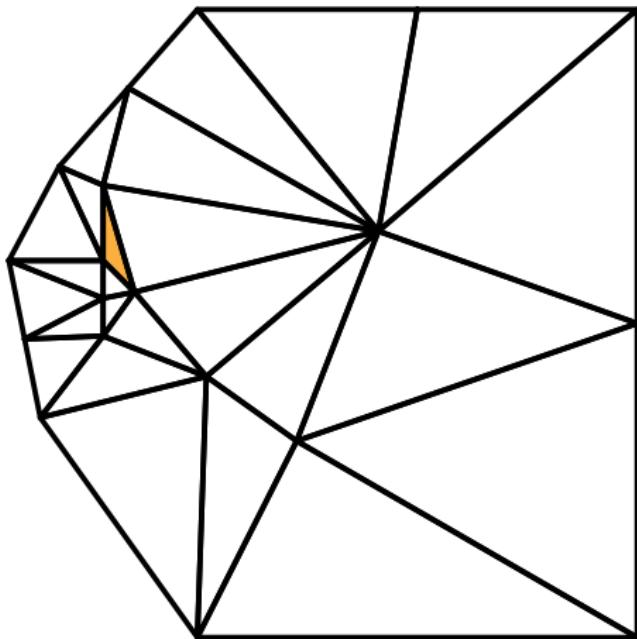
# Algoritmo de Ruppert



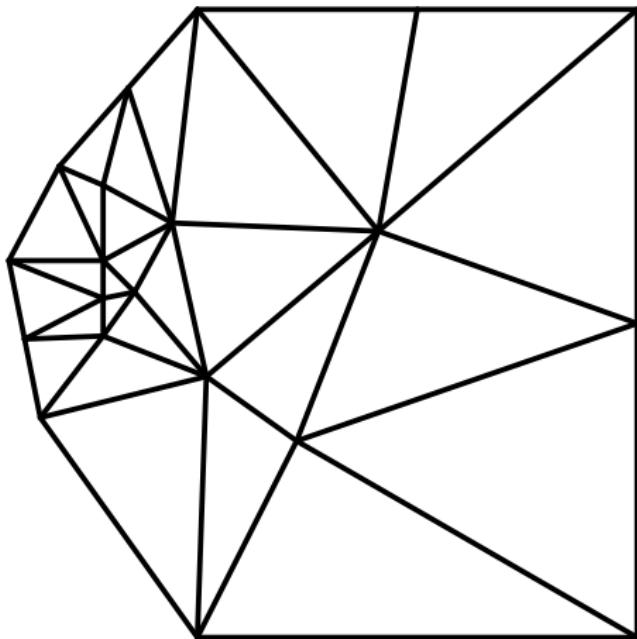
# Algoritmo de Ruppert



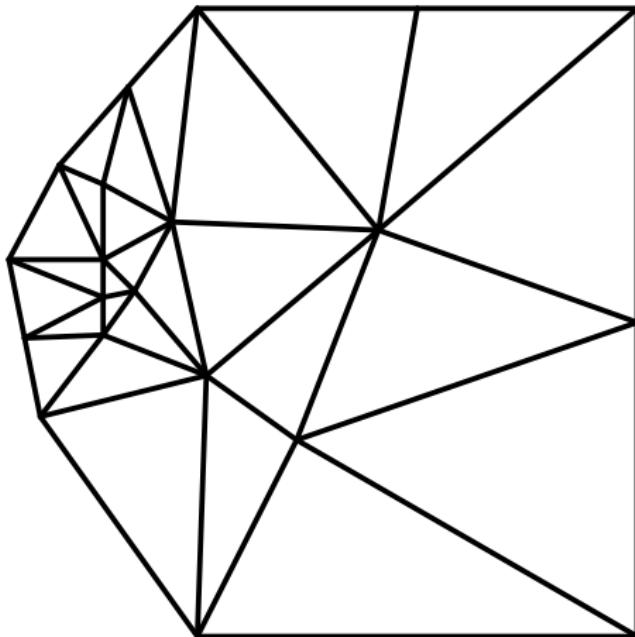
# Algoritmo de Ruppert



# Algoritmo de Ruppert



# Algoritmo de Ruppert



**Complexidade:**  $\mathcal{O}(n \lg n + N)$

# Análise do Algoritmo de Ruppert

## Raio de Inserção

### Definição (raio de inserção)

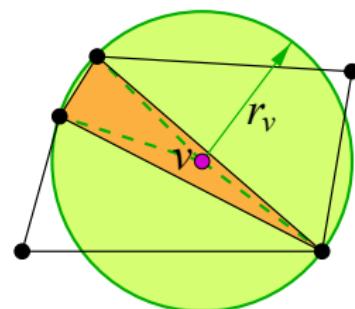
A cada ponto  $v \in \mathbb{E}^2$  inserido pelo Algoritmo de Ruppert em  $\mathcal{TD}(Q)$ , definimos o *raio de inserção* de  $v$ , denotado por  $r_v$ , como sendo a distância  $d(v, Q)$ .

# Análise do Algoritmo de Ruppert

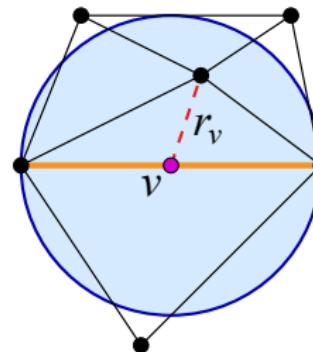
## Raio de Inserção

### Definição (raio de inserção)

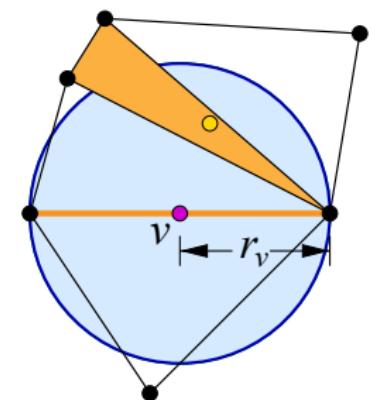
A cada ponto  $v \in \mathbb{E}^2$  inserido pelo Algoritmo de Ruppert em  $\mathcal{TD}(Q)$ , definimos o *raio de inserção* de  $v$ , denotado por  $r_v$ , como sendo a distância  $d(v, Q)$ .



inserindo  
circuncentro



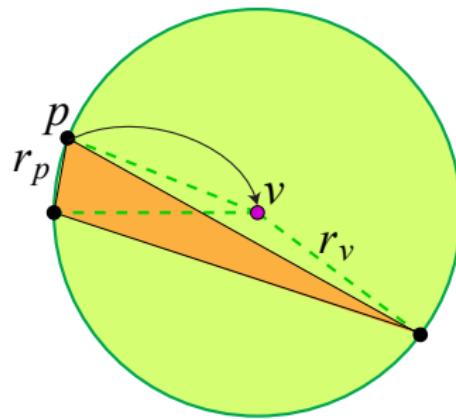
inserindo  
ponto médio



inserindo  
ponto médio

# Análise do Algoritmo de Ruppert

## Raio de Inserção de Circuncentros

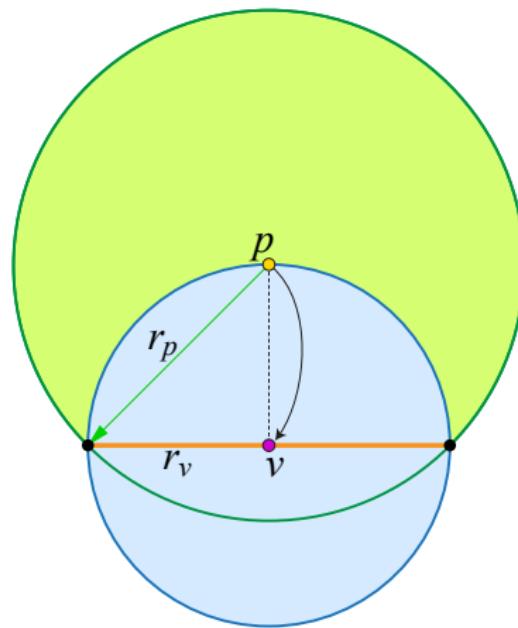


$p$  é pai do vértice  $v$

$$\frac{r_v}{r_p} > \bar{\rho}$$

# Análise do Algoritmo de Ruppert

Raio de Inserção de Ponto Médios de Subsegmentos

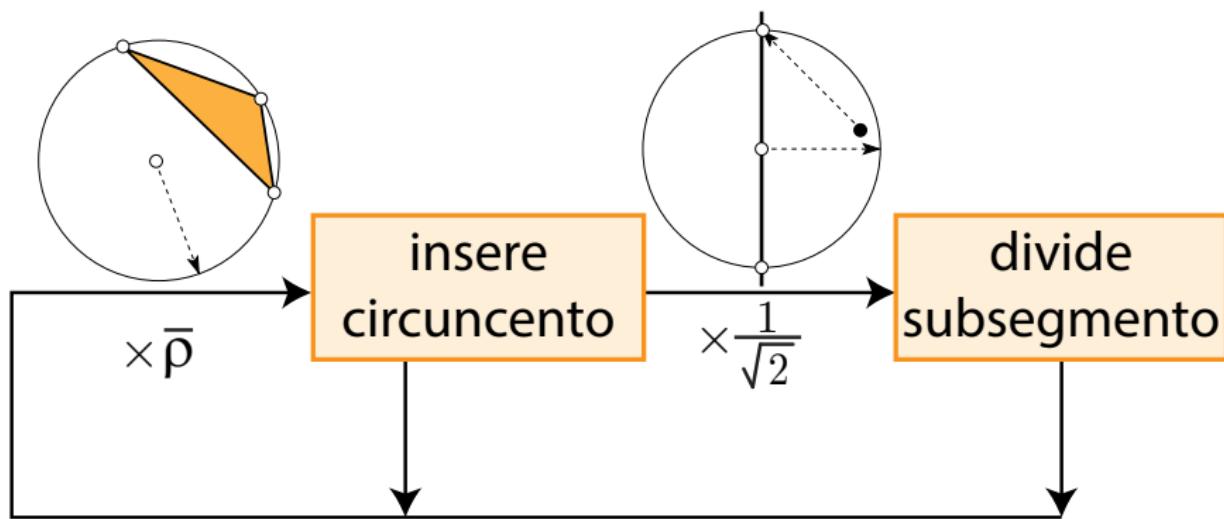


$p$  é pai do vértice  $v$   
( $p$  no pior caso)

$$\frac{r_v}{r_p} > \frac{1}{\sqrt{2}}$$

# Análise do Algoritmo de Ruppert

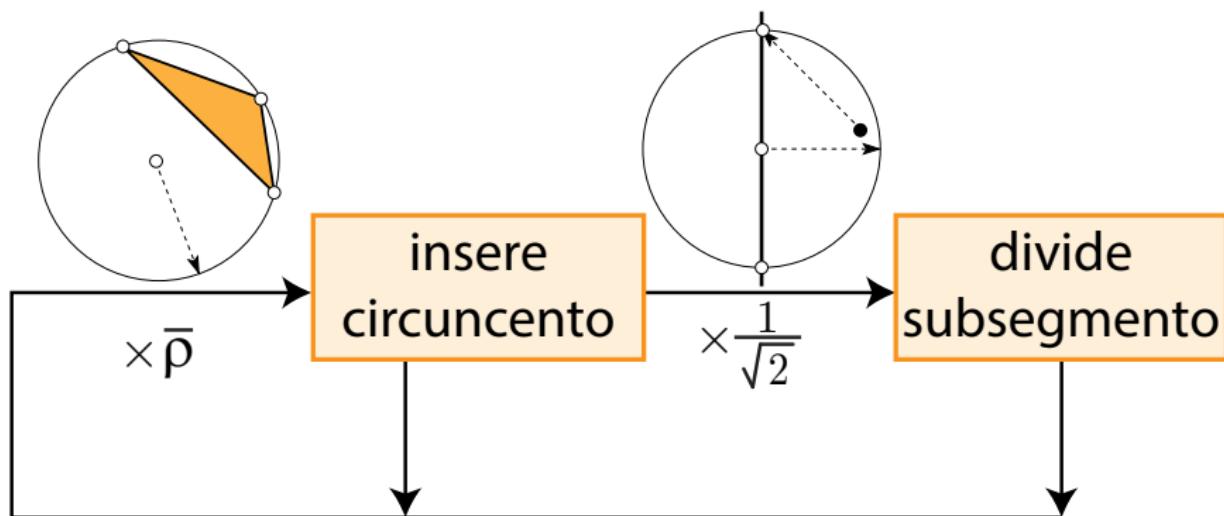
## Garantia de Término



algoritmo termina se não existir nenhum ciclo com o produto  $< 1$

# Análise do Algoritmo de Ruppert

## Garantia de Término



**algoritmo termina se não existir nenhum ciclo com o produto  $< 1$**

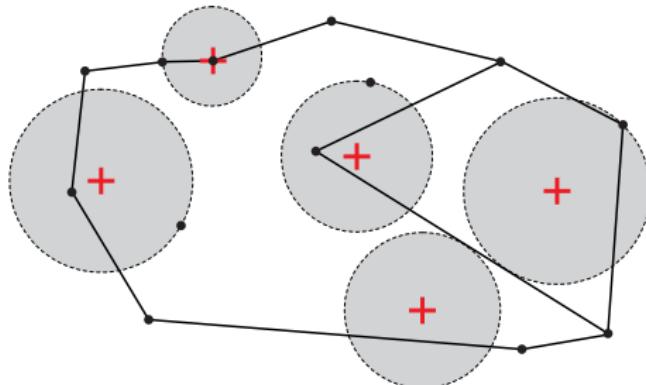
O que acontece quando um segmento é invadido por um vértice cujo pai não é um circuncentro?

# Análise do Algoritmo de Ruppert

## Garantia de Término

### Definição (*local feature size*)

A *local feature size* (LFS) de um GPSR  $\mathcal{G}$  é uma função,  $lfs : \mathbb{E}^2 \rightarrow \mathbb{R}$ , tal que para todo  $p \in \mathbb{E}^2$ , o valor de  $lfs(p)$  é igual ao raio do menor círculo centrado em  $p$  que intersecta dois elementos disjuntos (isto é, dois vértices, um vértice  $v$  e um segmento  $s$  tal que  $v \notin s$  ou dois segmentos,  $s_1$  e  $s_2$ , tais que  $s_1 \cap s_2 = \emptyset$ ) do GPSR  $\mathcal{G}$ .



# Análise do Algoritmo de Ruppert

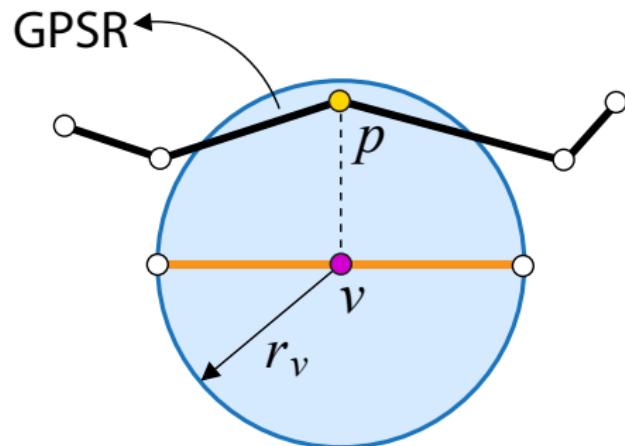
## Garantia de Término

- ▶ Os segmentos do GPSR não formam ângulos agudos

# Análise do Algoritmo de Ruppert

## Garantia de Término

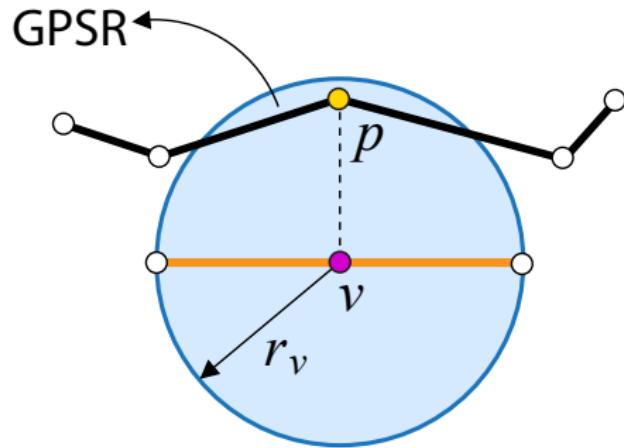
- ▶ Os segmentos do GPSR não formam ângulos agudos



# Análise do Algoritmo de Ruppert

## Garantia de Término

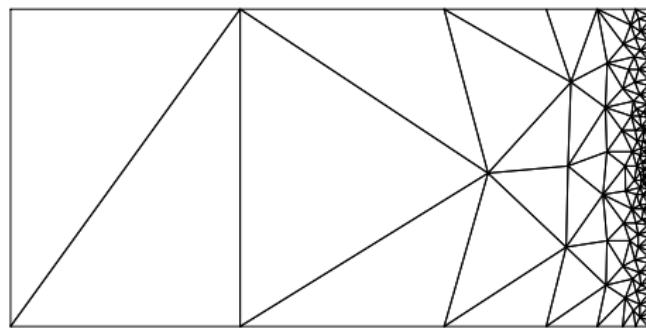
- ▶ Os segmentos do GPSR não formam ângulos agudos



$$r_v = d(p, v) \geqslant lfs(v)$$

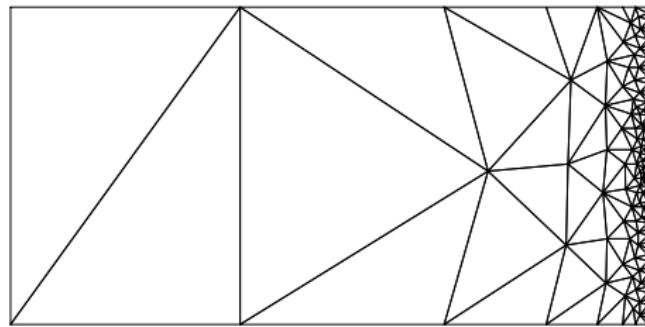
# Análise do Algoritmo de Ruppert

Malha Comprovadamente Boa



# Análise do Algoritmo de Ruppert

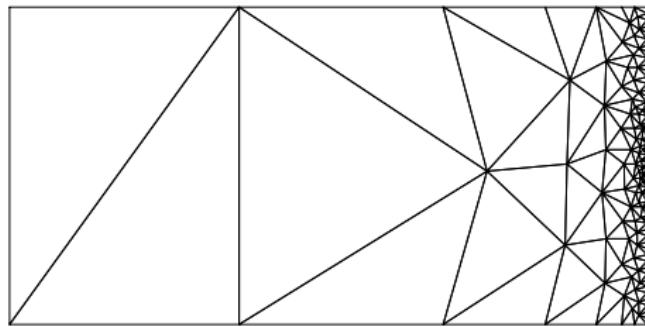
Malha Comprovadamente Boa



- Boa transição: todas as arestas possuem comprimento  $r_v \geq C_g \cdot lfs(v)$

# Análise do Algoritmo de Ruppert

Malha Comprovadamente Boa



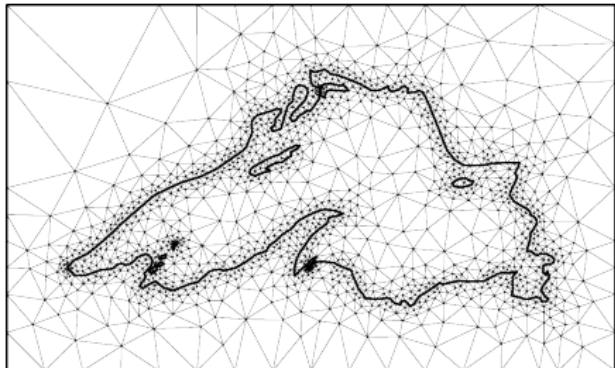
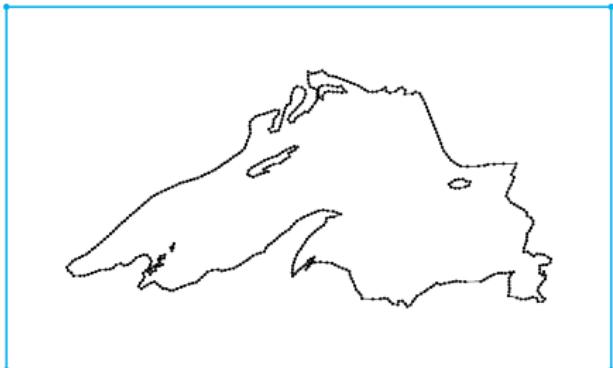
- ▶ Boa transição: todas as arestas possuem comprimento  $r_v \geq C_g \cdot lfs(v)$
- ▶ Otimalidade: o número de triângulos da malha é  $< C_m$ , onde  $C_m$  é uma constante que depende de  $\bar{\rho}$

## Algoritmo de Ruppert

- ▶ Como gerar uma malha de um GPSR que não satisfaz (P2)?

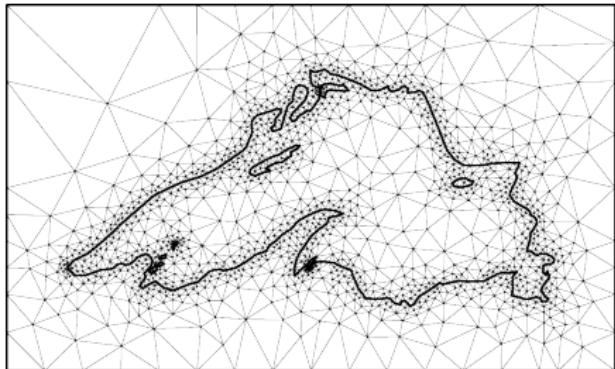
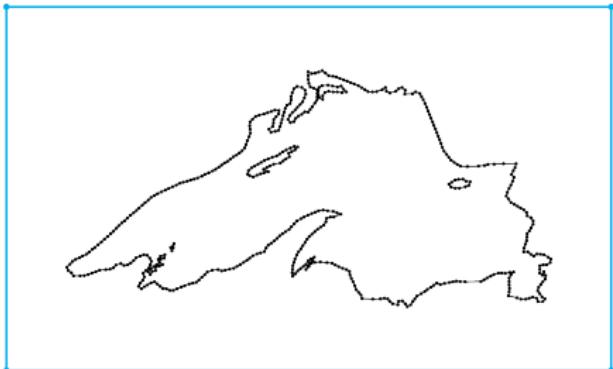
# Algoritmo de Ruppert

- ▶ Como gerar uma malha de um GPSR que não satisfaz (P2)?



# Algoritmo de Ruppert

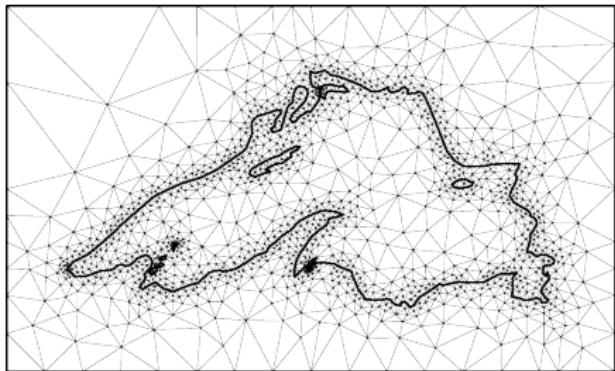
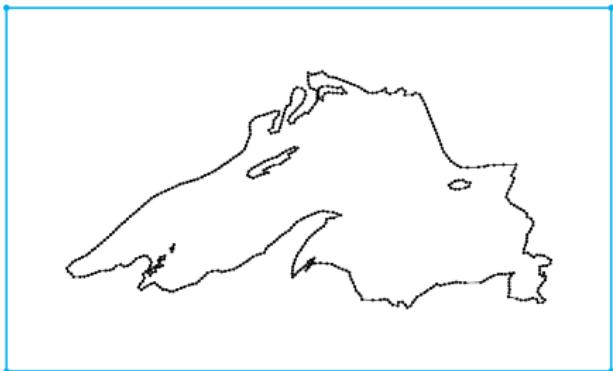
- ▶ Como gerar uma malha de um GPSR que não satisfaz (P2)?



esforço computacional aonde não precisa

# Algoritmo de Ruppert

- ▶ Como gerar uma malha de um GPSR que não satisfaz (P2)?



esforço computacional aonde não precisa

Na próxima aula:

- ▶ Como fazer de isso de forma eficiente?
- ▶ Como admitir um GPSR com ângulos agudos?