Konvexní obálka množiny bodů.

Graham Scan. Jarvis Scan. Quick Hull. Sweep Line. Divide and Conquer. Minimum Area Enclosing Rectangle. Výpočet hlavních směrů budov.

Tomáš Bayer | bayertom@natur.cuni.cz

Katedra aplikované geoinformatiky a kartografie. Přírodovědecká fakulta UK.

Obsah přednášky

1. Detekce obrysu budovy z klasifikovaných dat

2. Formulace problému

Dáno:	Množina n bodů $S = \{p_1, , p_2, ..., p_n\}$ v $\mathbb{R}^2$, kde $p_i = [x_i, y_i]$.
Hledáme:	Nejmenší konvexní množinu C, $\forall p_i \in C$. (Convex Hull Problem)

Definice 1:
Konvexní obálka množiny $\mathcal{H}$ konečné množiny S v $\mathbb{E}^2$ představuje nejmenší konvexní mnohoúhelník P obsahující S (tj. neexistuje $P' \subset P$ splňující definici).

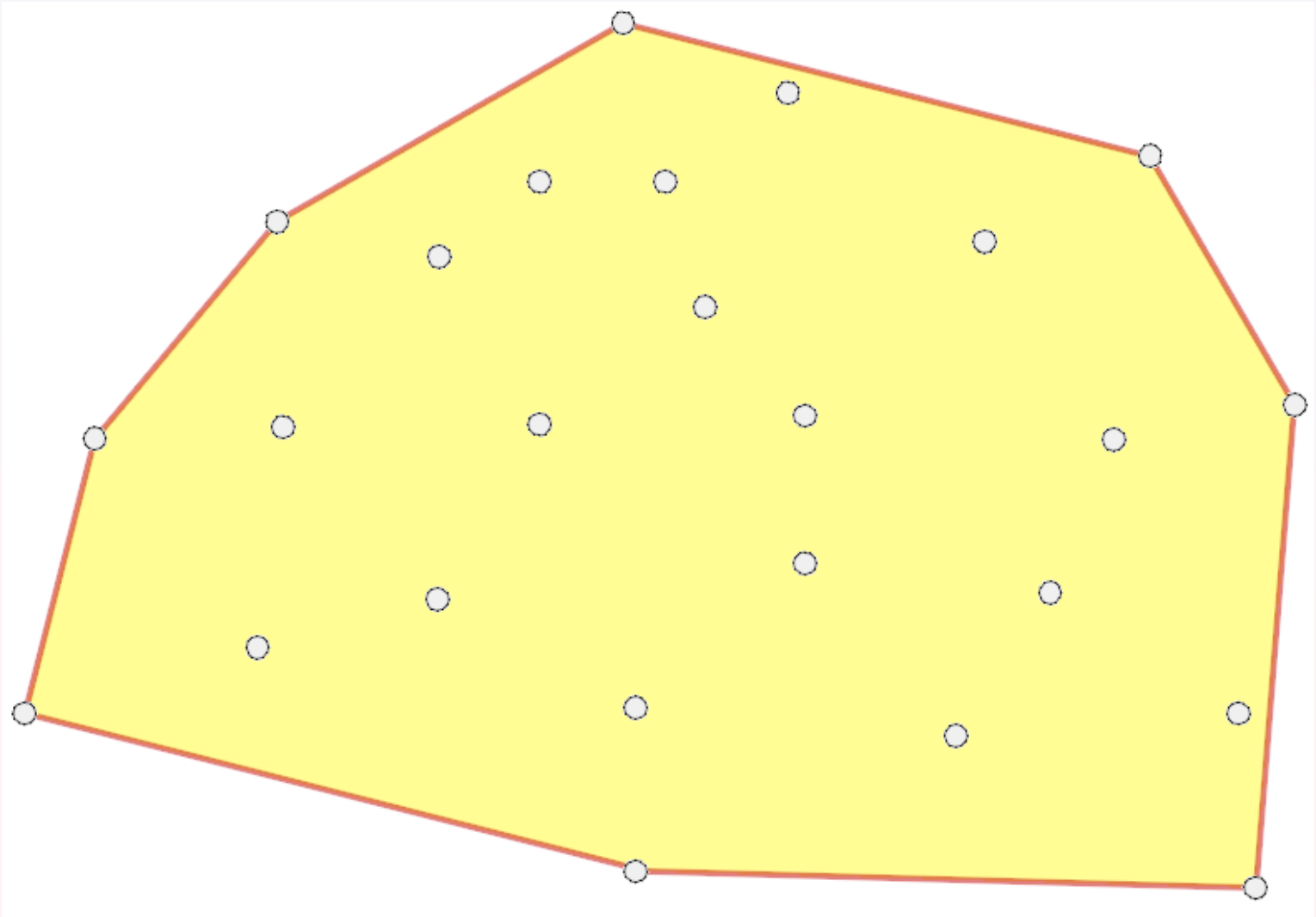
Definice 2:
Konvexní obálka $\mathcal{H}$ množiny konečné množiny S představuje konvexní mnohoúhelník P s nejmenší plochou.

Definice 3:
Konvexní obálka $\mathcal{H}$ množiny konečné množiny S představuje průsečnici všech polorovin obsahujících S.

Definice 4:
Konvexní obálka $\mathcal{H}$ množiny konečné množiny S představuje sjednocení všech trojúhelníků, jejichž vrcholy tvoří body v S.

Množinu S označíme jako konvexní, pokud spojnice libovolných dvou prvků leží zcela uvnitř této množiny.

3. Ukázka konvexní obálky $\mathcal{H}$ množiny S



4. Využití konvexní obálky

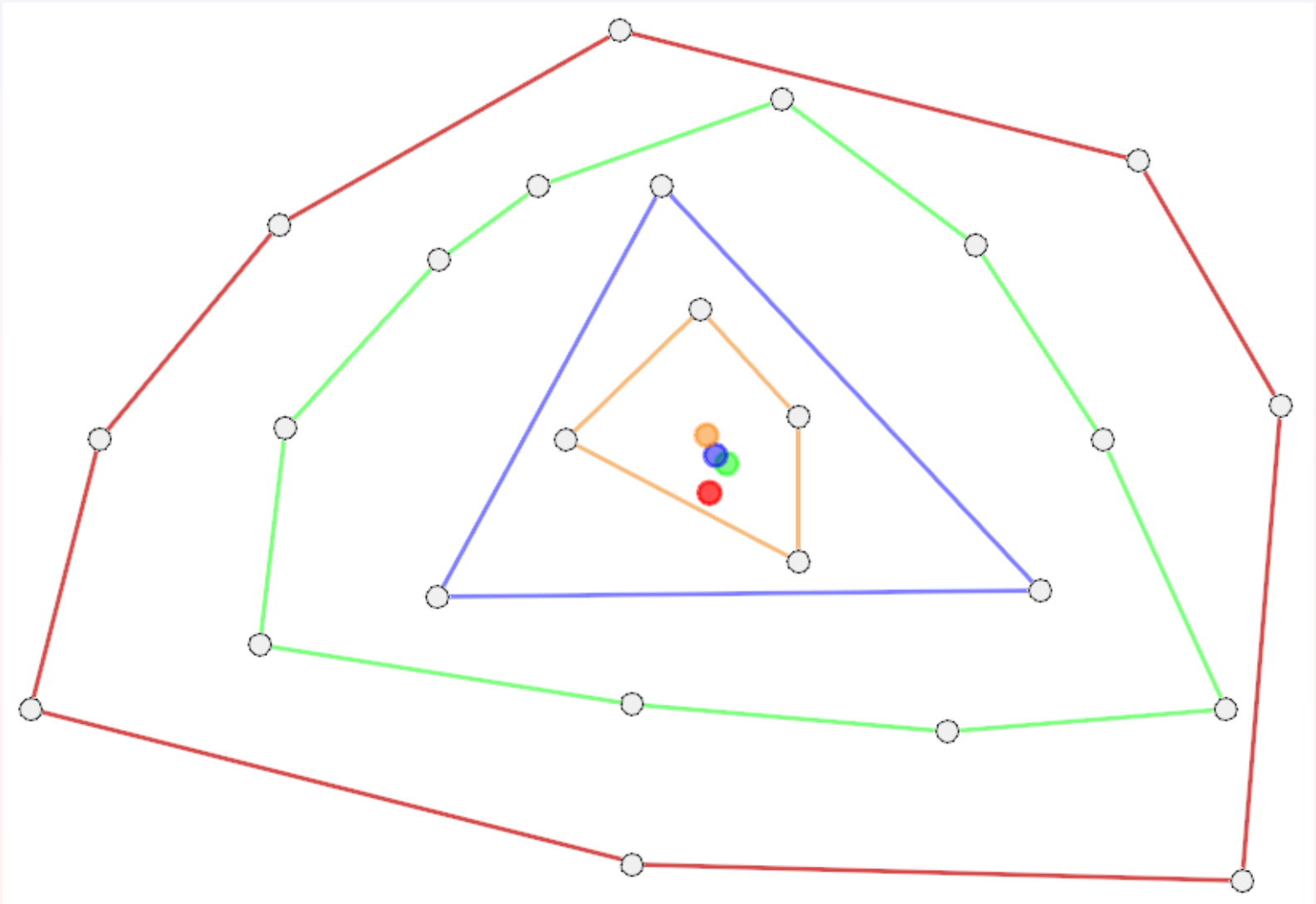
Jedna z nejpoužívanějších geometrických struktur, pomocná struktura pro řadu algoritmů.
Často používána jako první odhad tvaru prostorového jevu

Příklady použití:

- Detekce kolizí: plánování pohybu robotů.
- Konstrukce MBR: využití v kartografii pro detekci tvaru a natočení budov.
- Analýza tvarů (Shape Analysis): analýza tvarů objektů.
- Statistická analýza: analýza rozptylů, odhady, např. metoda Onion Peeling.
- Analýza shluků (Cluster Analysis): vlastnosti clusterovaných dat.

Poznámka:
Konvexní obálka $\mathcal{H}$ existuje v $\mathbb{R}^d$., budeme se zabývat pouze $\mathbb{R}^2$ variantou.

5. Odhad centroidu metodou Onion Peeling



6. Metody konstrukce konvexní obálky

Přehled nejčastěji používaných metod pro konstrukci konvexní obálky:

- Jarvis Scan (Gift Wrapping).
- Graham Scan.
- Quick Hull.
- Inkrementální konstrukce.
- Zametací přímka.
- Divide and Conquer.

Některé z metod použitelné pouze pro $\mathbb{R}^2$, jiné převeditelné do vyšší dimenze.

7. Jarvis Scan (Jarvis, 1973)

Připomíná postup balení dárku do papíru (Gift Wrapping Algorithm).
Jednoduchá implementace, lze rozšířit i do $\mathbb{R}^3$.

Předpoklad: v S nejsou tři kolineární body.

Nutnost předzpracování $O(n)$: nalezení pivotu q

$$q = \min_{\forall p_i \in S}(y_i).$$

Princip algoritmu:
Dva poslední body $\mathcal{H}$ označenyp_{j-1}, p_j.
Aktuálně přidávaný bod p_{j+1} do $\mathcal{H}$ maximalizuje úhel

$$p_{j+1} = \arg\max_{\forall p_i \in S} \angle(p_{j-1}, p_j, p_i).$$

Bod p_{j+1} hledáme ze všech $p_i \in S$, které dosud nejsou součástí $\mathcal{H}$.
Lze je značkovat.

8. Algoritmus Jarvis Scan

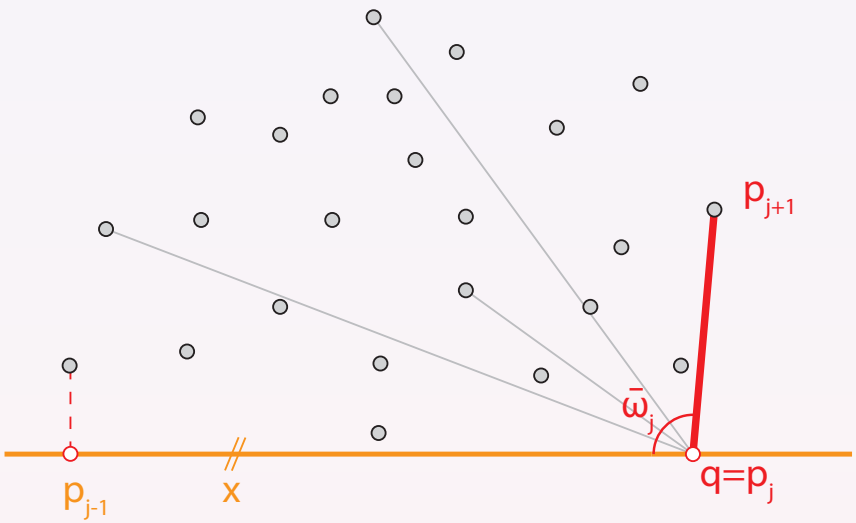
Nevýhodou složitost $O(n^2)$, nehodí se pro velké S.
Snadná implementace.

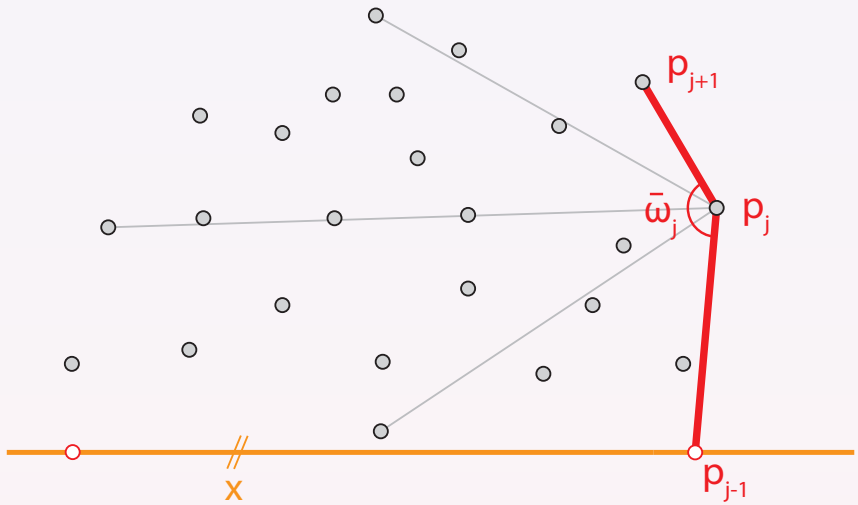
Algoritmus 1: Gift Wrapping $(P, \mathcal{H})$

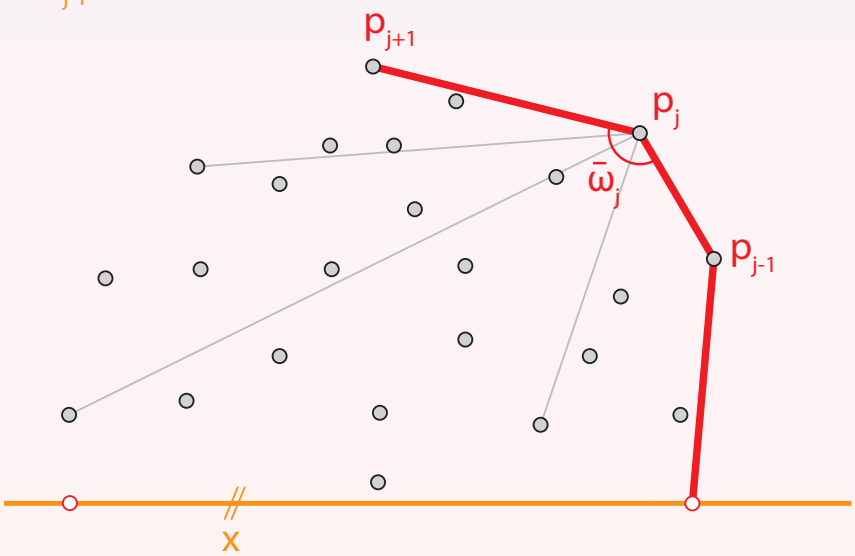
1: Nalezení pivota q, $q = \min(y_i)$,
2: Přidej $q \to \mathcal{H}$.
3: Inicializuj: $p_{j-1} \in X$, $p_j = q$, $p_{j+1} = p_{j-1}$.
4: Opakuj, dokud $p_{j+1} \neq q$:
5: Nalezni $p_{j+1} = \arg\max_{\forall p_i \in P} \angle(p_{j-1}, p_j, p_i)$
6: Přidej $p_{j+1} \to \mathcal{H}$.
7: $p_{j-1} = p_j$; $p_j = p_{j+1}$.

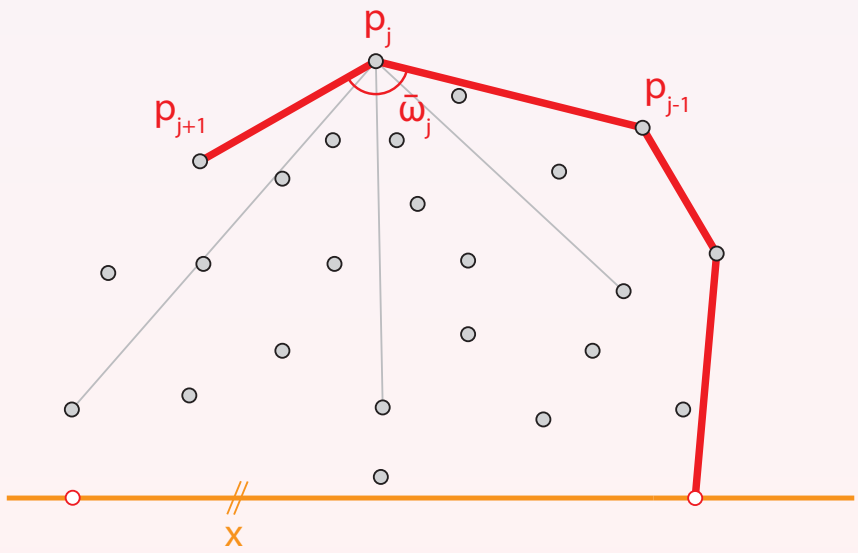
---

9. Ukázka Jarvis Scan









10. Graham Scan

Algoritmus není možné rozšířit do $\mathbb{R}^3$ (Graham 1972).
Časová složitost $O(n \cdot \log(n))$.
Lze použít i na rozsáhlé datasety.
Převod star-shaped polygonu na $\mathcal{H}$.

Princip:
Konstrukce star-shaped polygonu z pivotu q.
Tvořen vrcholy setříděnými dle úhlu $\omega = \angle(x, qp_i)$.
Body ve star-shaped polygonu: index j.

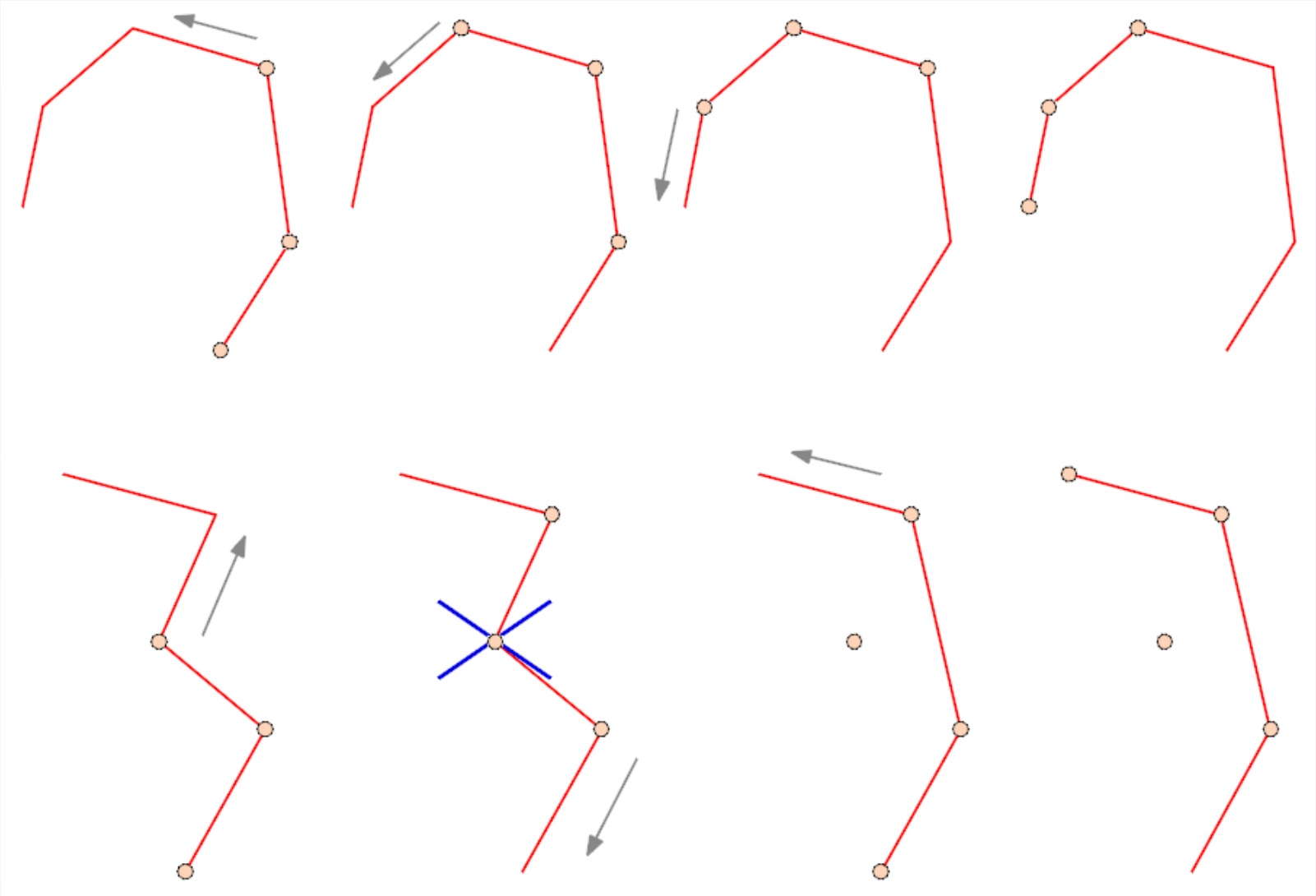
Kritérium levotočivosti

$$p_j \begin{cases} \notin \mathcal{H}, & p_{j+1} \in \sigma_r(p_j, p_{j+1}), \\ ? \in \mathcal{H}, & p_{j+1} \in \sigma_l(p_j, p_{j+1}). \end{cases}$$

Implementace:
Prvky v zásobníku S: t (top), $t-1$ (second).
Reflexní vrchol p_t odebrán z S, "konvexní" p_j přidán

$$S = \begin{cases} S.pop(), & p_j \in \sigma_r(p_{t-1}, p_t), \\ S.push(p_j), & p_j \in \sigma_l(p_{t-1}, p_t). \end{cases}$$

11. Ilustrace kritéria levotočivosti



12. Algoritmus Grahamova skenování

Předzpracování $O(n)$: pivot q

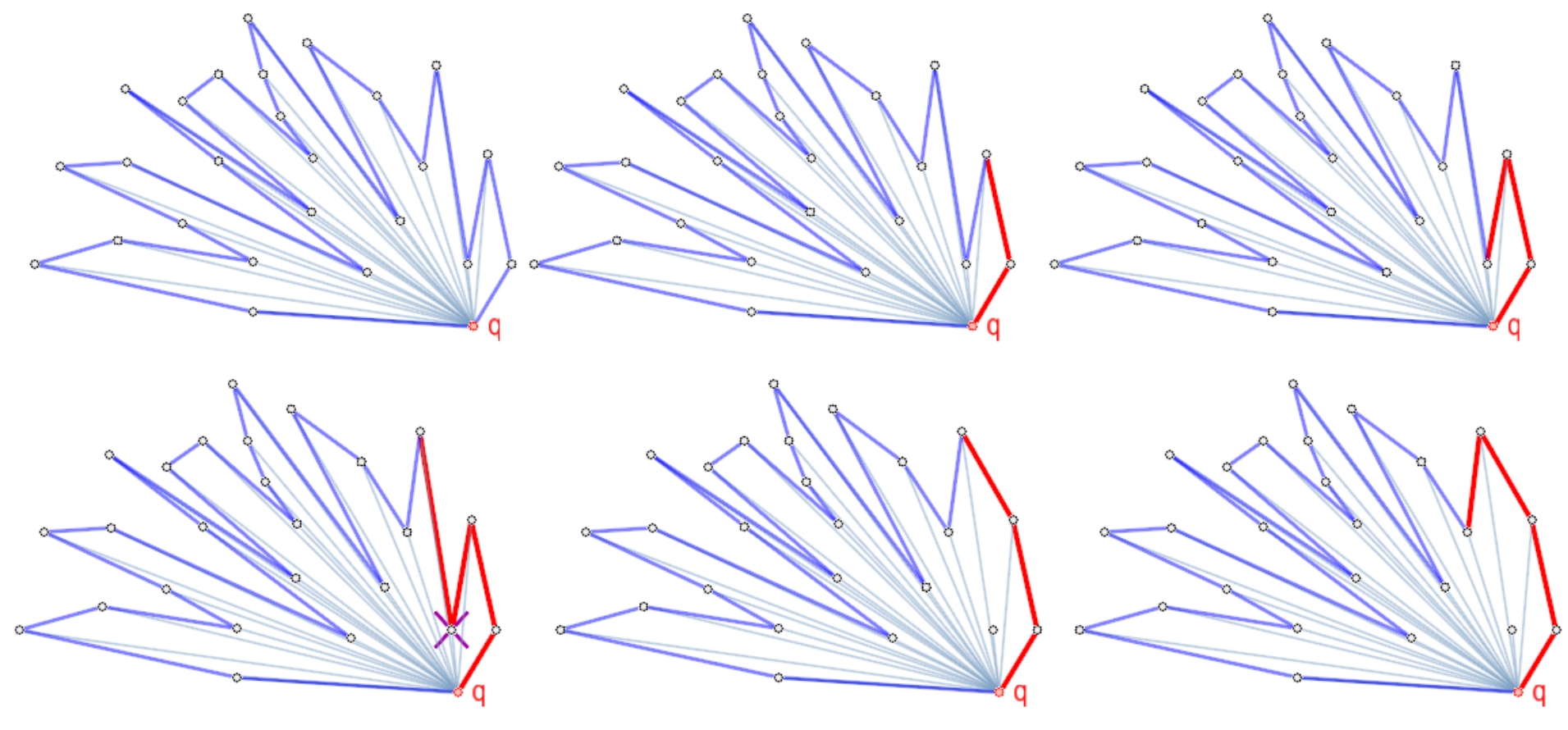
$$q = \min_{\forall p_i \in S}(y_i).$$

Předzpracování $O(n \cdot \log(n))$: setřídění datasetu S dle úhlu $\angle(x, qp_i)$.

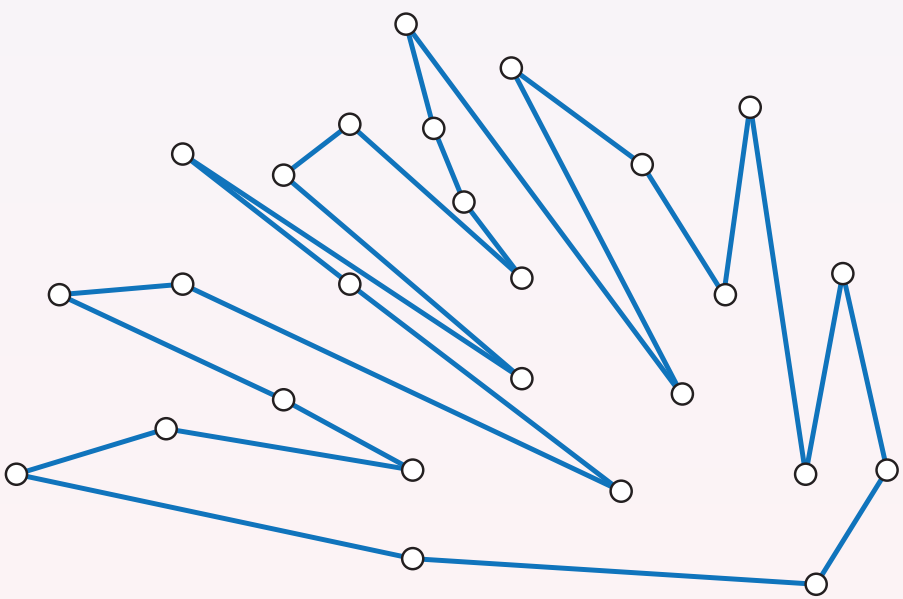
Algoritmus 2: Graham Scan (P, S)

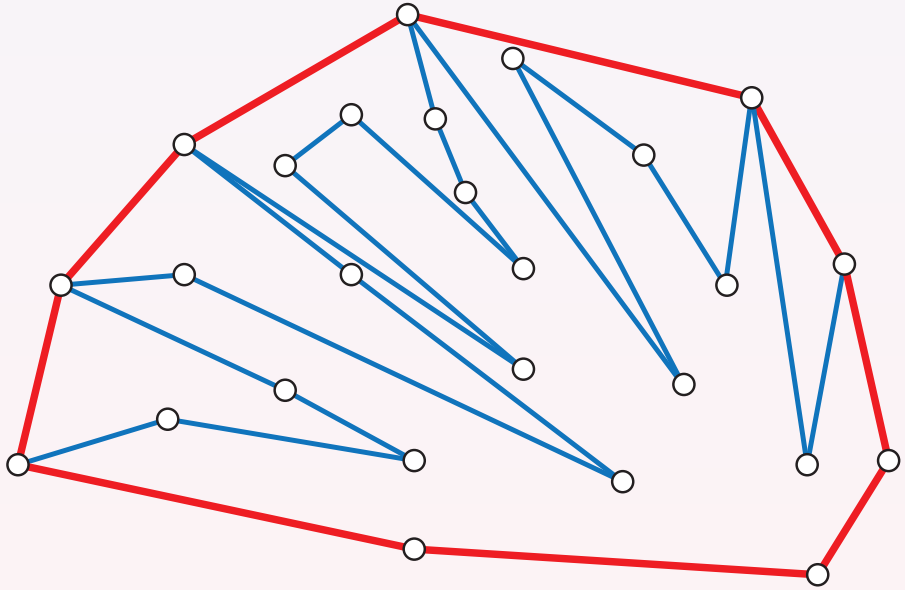
1: Nalezení pivota $q = \min_{\forall p_i \in S}(y_i)$, $q \in \mathcal{H}$.
2: Setřídění $\forall p_i \in S$ dle $\omega_i = \angle(p_i, q, x)$, index j odpovídá setříděnému pořadí.
3: Pokud $\omega_k = \omega_l$, vymaž bod p_k, p_l bližší ke q.
4: Inicializuj $j = 2$; $S = \emptyset$
5: $S \leftarrow q$, $S \leftarrow p_1$ (indexy posledních dvou prvků p_t, p_{t-1})
5: Opakuj pro $j < n$:
6: if p_j vlevo od p_{t-1}, p_t:
7: $S \leftarrow p_j$
8: $j = j + 1$
9: else pop S.

13. Ukázka konstrukce Graham Scan



14. Ukázka konstrukce Graham Scan





15. Algoritmus Quick Hull

Strategie Divide and Conquer.
Ve většině případů rychlý $\Theta(n \lg n)$, avšak $O(n^2)$.
Obálka postupně expanduje všemi směry.

Princip algoritmu:
Předzpracování: setřídění bodů dle x, $O(n \lg n)$.
Rozdělení S na S_U(upper) a S_L (lower) vzhledem k (q_1, q_3)

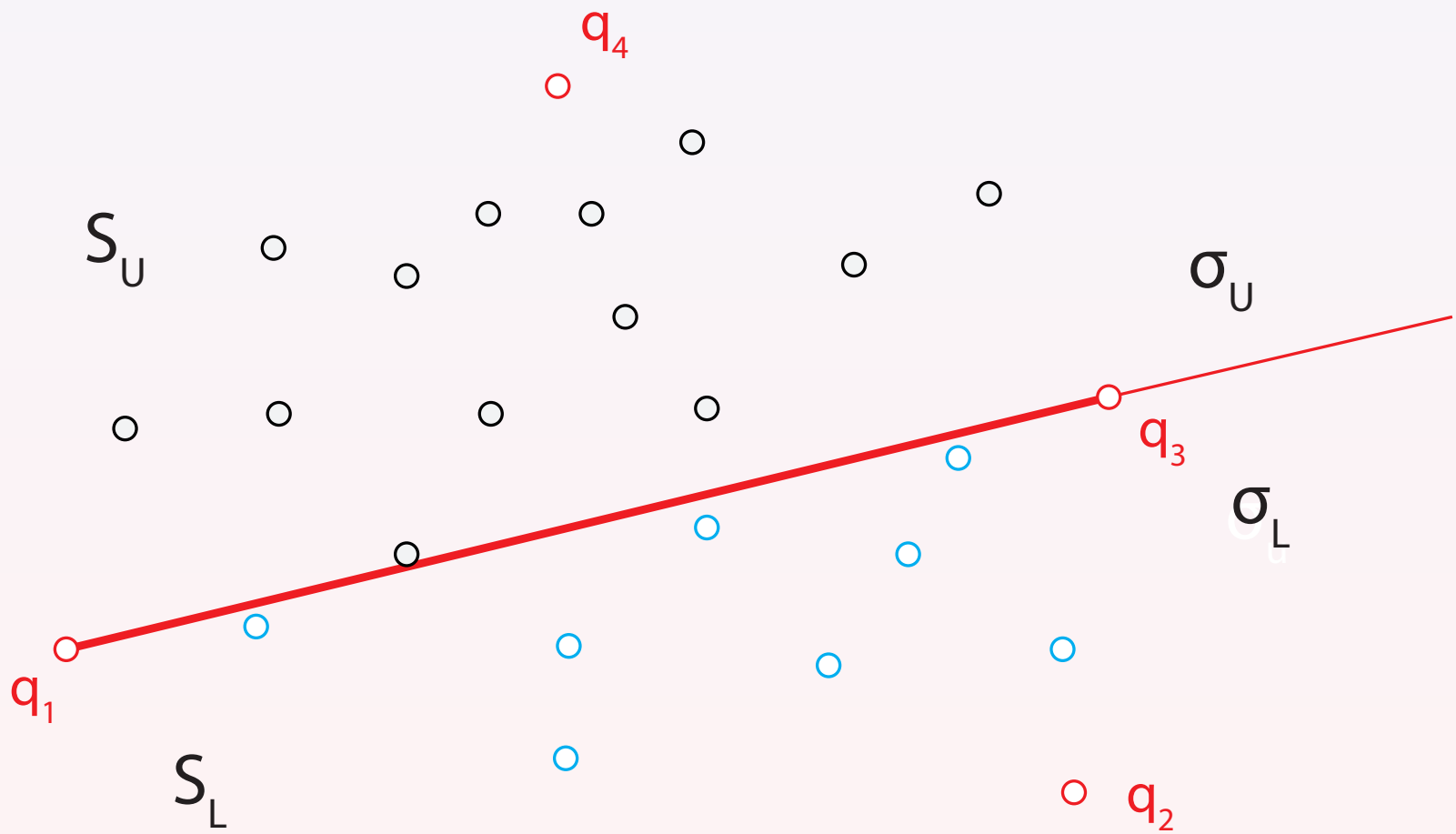
$$q_1 = \min_{\forall p_i \in S}(x_i), \qquad q_3 = \max_{\forall p_i \in S}(x_i).$$

$\mathcal{H}$ konstruována ze 2 částí: *Upper Hull* ($\mathcal{H}_U$), *Lower Hull* ($\mathcal{H}_L$).
$\mathcal{H}_U \subset S_U$, $\mathcal{H}_L \subset S_L$.
$\mathcal{H}_U$, $\mathcal{H}_L$ zpracovány samostatně

$$\mathcal{H} = \mathcal{H}_u \cup \mathcal{H}_l.$$

Důvod rychlosti:
Většinou málo rekurzivních kroků, málo bodů vně aproximace $\mathcal{H}_U$, $\mathcal{H}_L$.
Pokud $S \equiv \mathcal{H} \Rightarrow$ worst case.

16. Rozdělení na S_U, S_L



17. Algoritmus Quick Hull, globální procedura

Nalezení q_1, q_2,nutné předzpracování.

- Rozdělení S na: $S_U \subset S$(upper) a $S_L \subset S$ (lower) vzhledem k (q_1, q_2).
- Přidání $\mathcal{H}_U \leftarrow q_1, q_3$, rekurzivní procedura nad S_U.
- Přidání $\mathcal{H}_L \leftarrow q_3, q_q$, rekurzivní procedura nad S_L.

Při CCW orientaci vede ke konstrukci $\mathcal{H}$ bez nutnosti spojení.

Algoritmus 3: Quick Hull (S, $\mathcal{H}$), globální procedura

---

1: $\mathcal{H} = \emptyset$, $S_U = \emptyset$, $S_L = \emptyset$

2: $q_1 = \min_{\forall p_i \in S}(x_i)$, $q_3 = \max_{\forall p_i \in S}(x_i)$.

3: $S_U \leftarrow q_1$, $S_U \leftarrow q_3$.

4: $S_L \leftarrow q_1$, $S_L \leftarrow q_3$.

5: for $\forall p_i \in S$

6: if $(p_i \in \sigma_l(q_1, q_3))$ $S_U \leftarrow p_i$

7: else $S_L \leftarrow p_i$

8: $\mathcal{H} \leftarrow q_3$. //Přidej bod c do $\mathcal{H}$.

9: Quick Hull$(1, 0, S_U, \mathcal{H})$ //Upper Hull

10: $\mathcal{H} \leftarrow q_1$. //Přidej bod c do $\mathcal{H}$.

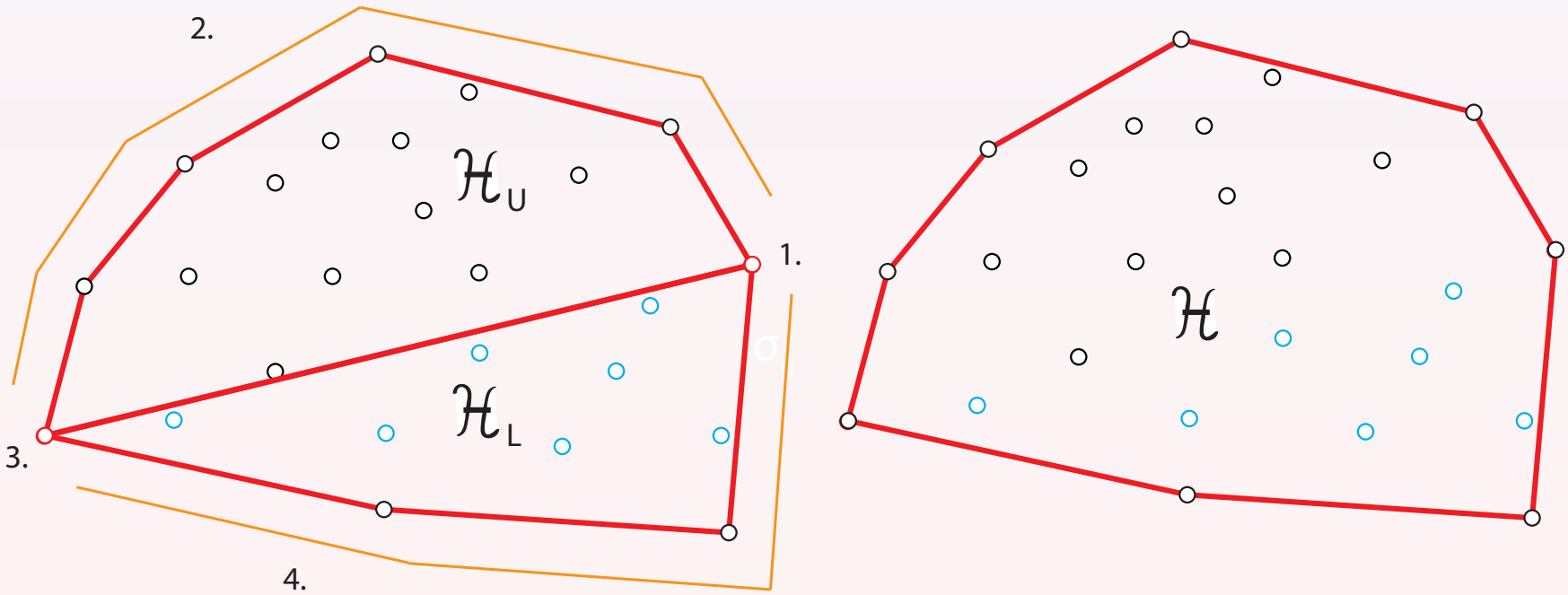
11: Quick Hull$(0, 1, S_L, \mathcal{H})$ //Lower Hull

---

18. Spojení do $\mathcal{H}$

Vhodné načasováním operací zajistí spojení $\mathcal{H}_U$ a $\mathcal{H}_L$ v globální proceduře:

1. Přidání q_3: $\mathcal{H} \leftarrow q_3$.
2. Rekurze $\mathcal{H}_U$.
3. Přidání q_1: $\mathcal{H} \leftarrow q_1$.
4. Rekurze $\mathcal{H}_L$.



19. Algoritmus Quick Hull, lokální procedura

Důležité pořadí operací (body $\mathcal{H}$ CCW orientovány).
Vstupní hrana (p_s, p_e) aproximované $\mathcal{H}$.

1) Nalezení bodu $\overline{p}$ s indexem $\overline{i}$

$$\overline{p} = \arg \max_{\forall p_i \in S} \|p_i - (p_s, p_e)\|_2 , \qquad \overline{p} \in \sigma_r(p_s, p_e).$$

2) Vytvoření nových segmentů $(p_s, \overline{p})$ a $(\overline{p}, p_e)$.
3) Rekurzivní volání nad prvním segmentem $(p_s, \overline{p})$.
4) Přidání $\mathcal{H} \leftarrow \overline{p}$.
5) Rekurzivní volání nad druhým segmentem $(\overline{p}, p_e)$.

V lokální proceduře předávány pouze indexy s, e koncových bodů segmentu.
Index $\overline{i}$ odpovídá bodu $\overline{p}$.

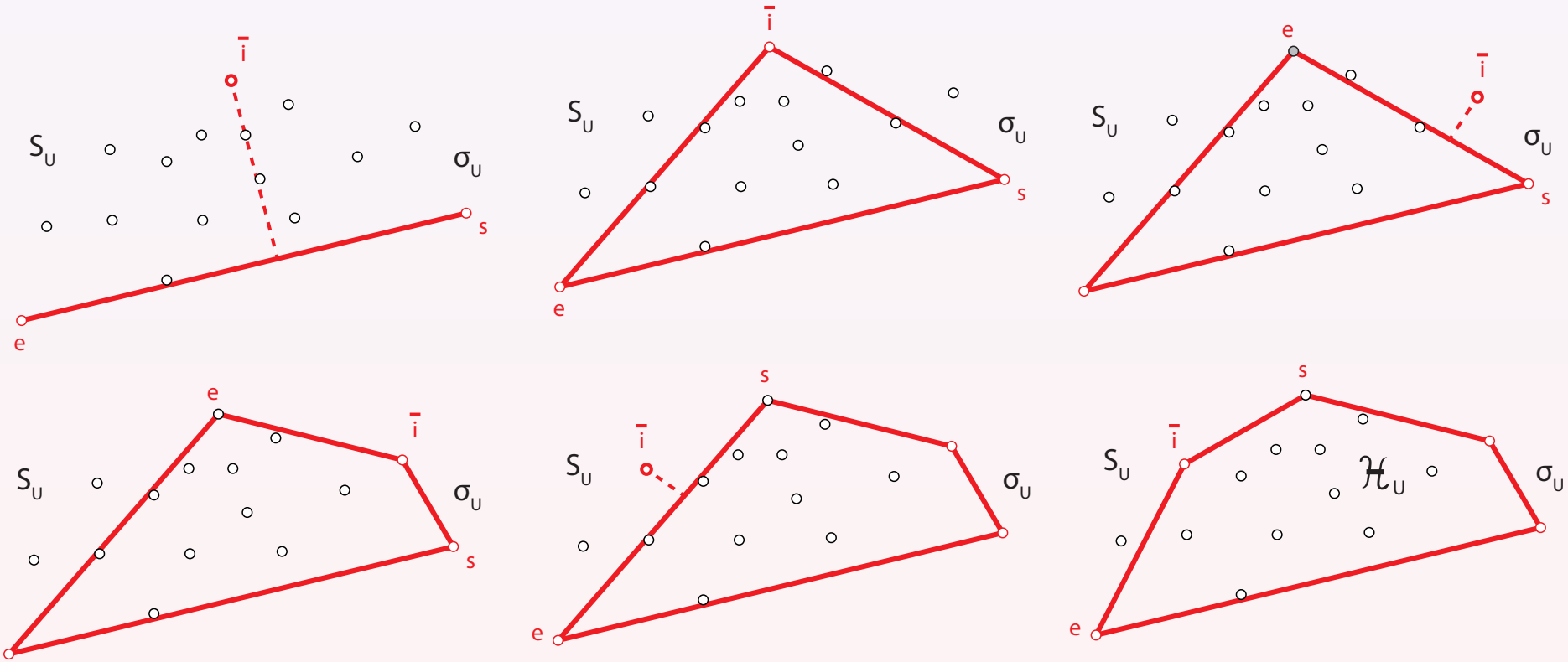
Algoritmus 3: Quick Hull (s, e, S, $\mathcal{H}$), lokální procedura

---

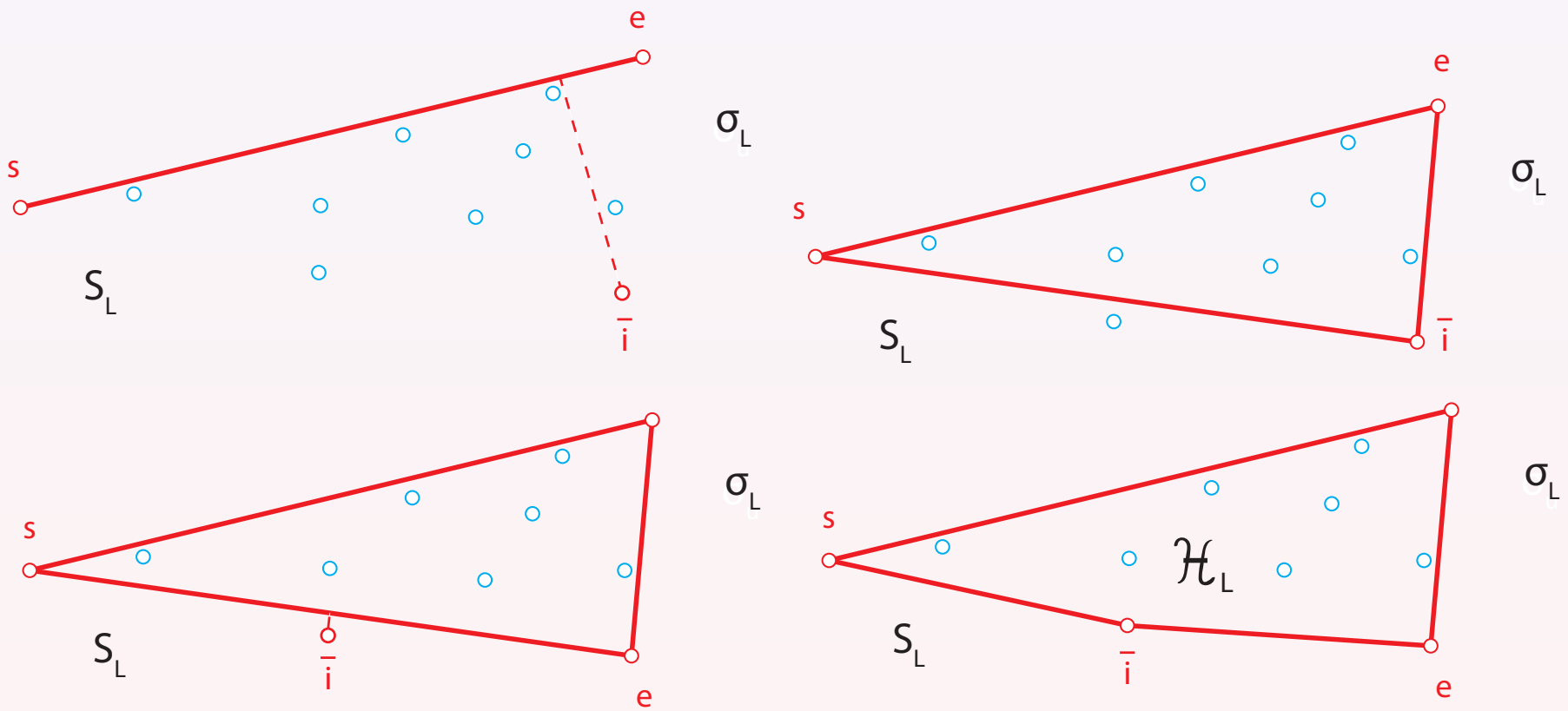
1: Najdi bod $\overline{p} = \arg\max_{\forall p_i \in S} \|p_i - (p_s, p_e)\| , \overline{p} \in \sigma_r(p_s, p_e)$.
2: If $\overline{p} \neq \emptyset$; //Existuje bod vpravo od hrany.
3: Quick Hull$(s, \overline{i}, S, \mathcal{H})$ //Upper Hull, $\overline{i}$ index $\overline{p}$
4: $\mathcal{H} \leftarrow \overline{p}$. //Přidej bod c do $\mathcal{H}$.
5: Quick Hull$(\overline{i}, e, S, \mathcal{H})$ //Lower Hull, $\overline{i}$ index $\overline{p}$

---

20. Lokální procedura S_U



21. Lokální procedura S_L



22. Doba běhu QHull

Časová funkce $T(n)$, očekávaná složitost (nejvzdálenější bod v mediánu):

$$\begin{aligned} T(n) &= 2T(n/2) + n, \\ &= 2[2T(n/4) + n/2] + n = 4T(n/4) + 2n, \\ &= 4[2T(n/8) + n/4] + 2n = 8T(n/8) + 3n, \\ &= 8[2T(n/16) + n/8] + 3n = 16T(n/16) + 4n, \\ &= iT(n/i) + n\log_2 i, \\ &= n(T(1)) + n\log_2 n, \\ &\leq n\log(n). \end{aligned}$$

Časová funkce $T(n)$, horní odhad složitosti (nejvzdálenější bod 2./předposlední):

$$\begin{aligned} T(n) &= T(n-1) + n, \\ &= [T(n-2) + n - 1] + n = T(n-2) + n - 1 + n, \\ &= T(n-3) + n - 2 + n - 1 + n, \\ &= T(n-i) + n - i + 1 + .. + n - 2 + n - 1 + n, \\ &= T(0) + \frac{n}{2}(n+1), \\ &\leq n^2. \end{aligned}$$

23. Aproximací body MBR

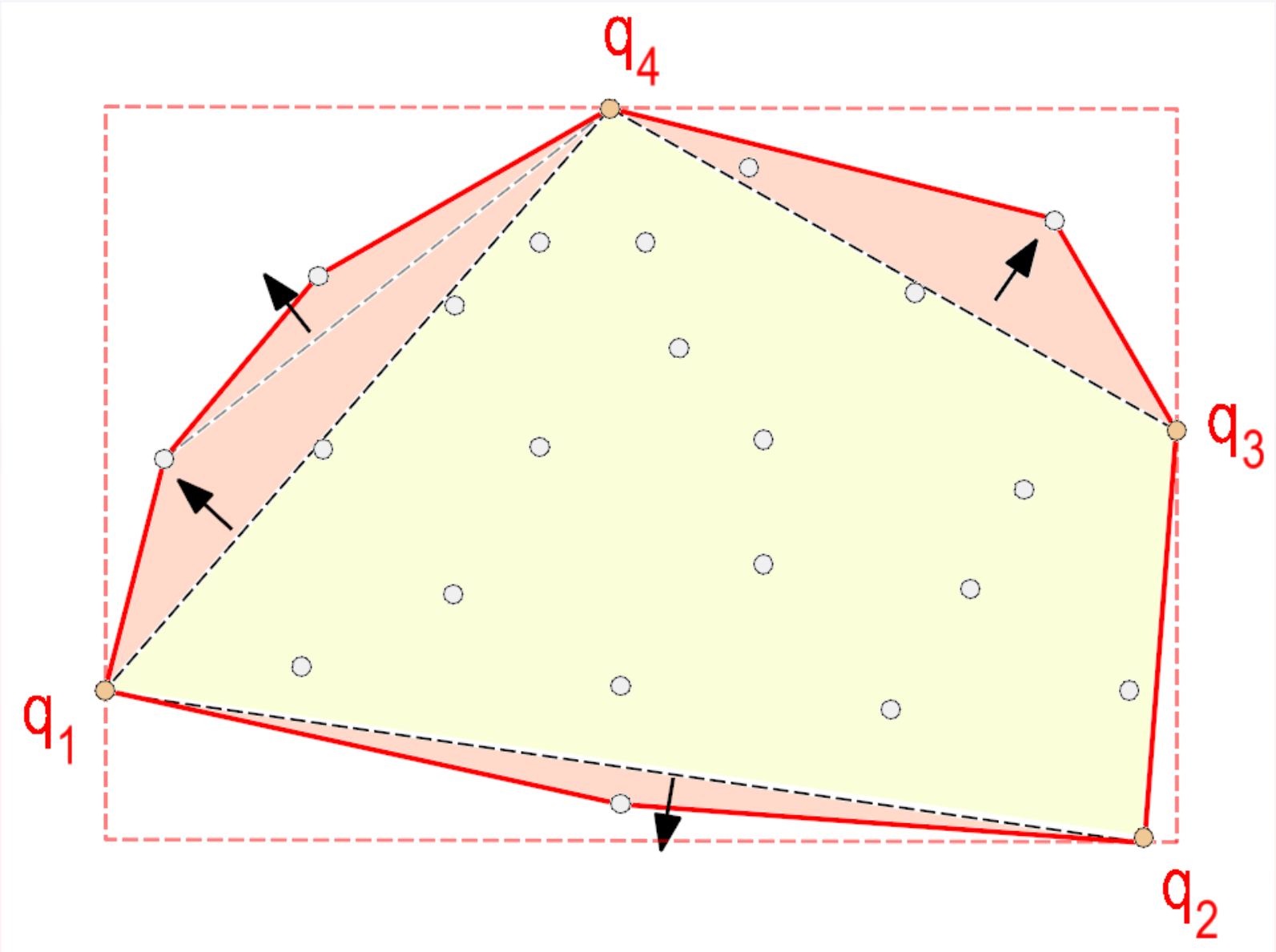
První aproximace $\mathcal{H} \equiv (q_1, q_2, q_3, q_4)$, kde q_i body MBR S.
Ošetření situací, kdy q_i mohou být identické (singularity).

4 rekurzivní volání QuickHull nad každou stranou $q_i q_{i+1}$.
$\mathcal{H}$ zpracováván po jednotlivých segmentech, expanduje do stran.

$q_1 = [\min(x_i), y]; q_2 = [x, \min(y_i)]; q_3 = [\max(x_i), y]; q_4 = [x, \max(y_i)];$
`Split` S `to` S_1, S_2, S_3, S_4;
$\mathcal{H} \leftarrow q_1$
`QuickHull`$(0, 1, S_1, \mathcal{H})$;
$\mathcal{H} \leftarrow q_2$
`QuickHull`$(0, 1, S_2, \mathcal{H})$;
$\mathcal{H} \leftarrow q_3$
`QuickHull`$(0, 1, S_3, \mathcal{H})$;
$\mathcal{H} \leftarrow q_4$
`QuickHull`$(0, 1, S_4, \mathcal{H})$;

Urychlení konstrukce pro běžná data.
Pokud většina $p \in \mathcal{H}$, stále pomalé.

24. Konstrukce Quick Hull s aproximací body MBR



25. Inkrementální vkládání

Rychlá konstrukce, složitost $O(n \cdot \log(n))$.
Lze převést do vyšší dimenze.

Princip algoritmu:
Body z S přidávány po jednom do vytvořené konvexní obálky $\mathcal{H}$

$$\mathcal{H}(S(m+1)) = \mathcal{H}(S(m)) + \Delta\mathcal{H}(p),$$

její tvar je modifikován.

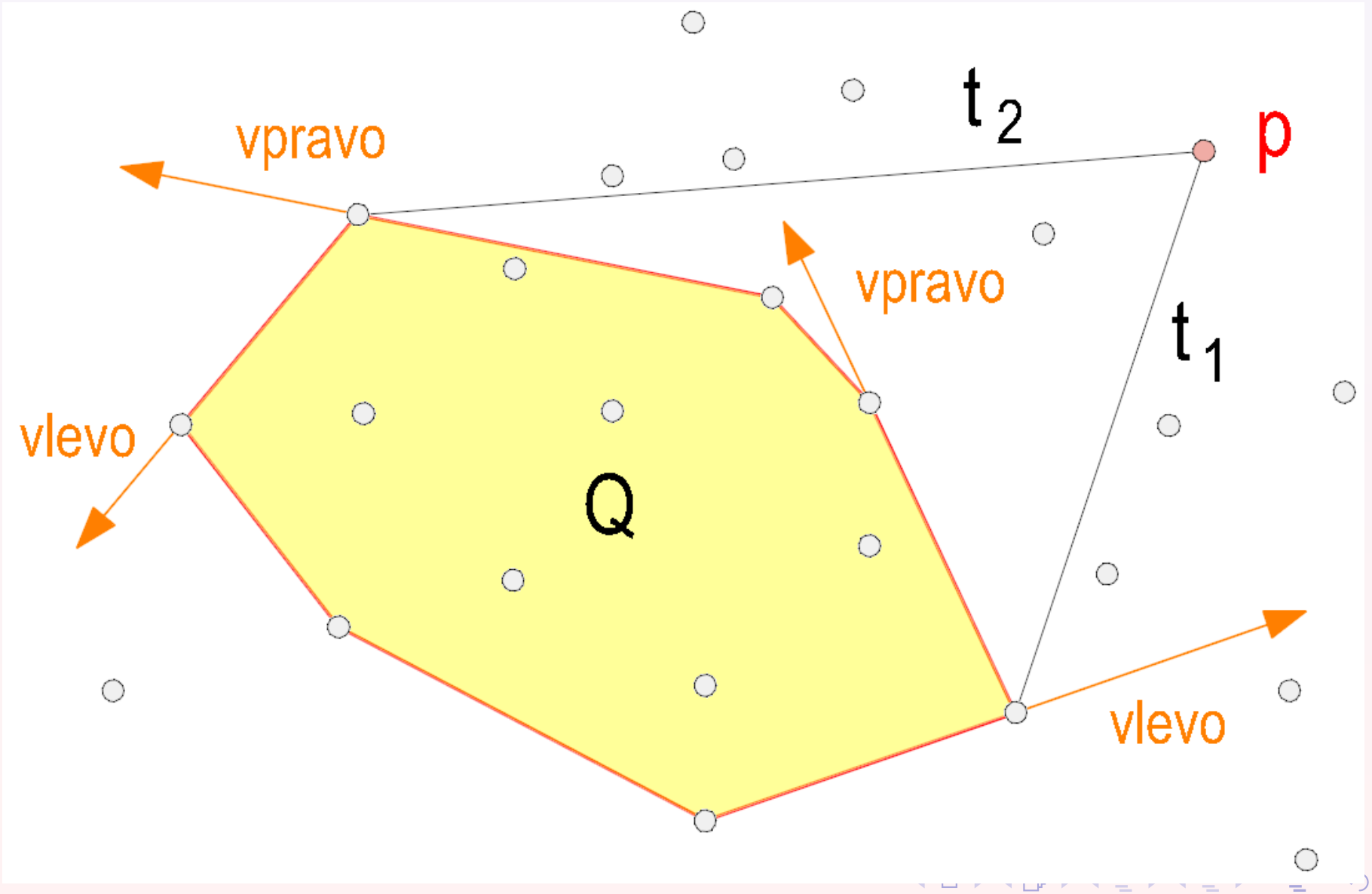
Při přidání bodu p do $\mathcal{H}(S(m))$ 2 situace:

- $p \in \mathcal{H}(S(m))$
  Bod p může být zanedbán, neovlivní tvar $\mathcal{H}(S(m))$

$$\mathcal{H}(S(m+1)) = \mathcal{H}(S(m)).$$

- $p \notin \mathcal{H}(S(m))$
  Bod p ovlivní tvar $\mathcal{H}(S(m))$.
  Nutno najít horní a dolní tečny t_1, t_2 procházející p kolmé k $\mathcal{H}(S(m))$.

26. Ukázka nalezení horní a dolní tečny



27. Princip nalezení horní a dolní tečny t_1, t_2

V každém bodě $\mathcal{H}(S(m))$ test polohy stran $p_{j-1}p_j$, p_jp_{j+1} vzhledem k p.
V bodech dotyku tečen platí:

Dolní tečna $t_1(p, p_j)$**:**
Bod p vlevo od $p_{j-1}p_j$ a vpravo od p_jp_{j+1}.

Horní tečna $t_2(p, p_j)$**:**
Bod p vpravo od $p_{j-1}p_j$ a vlevo od p_jp_{j+1}.

Ponechané vrcholy $p_i \in \mathcal{H}(S(m))$: p na stejné straně vzhledem k $p_{j-1}p_j$, p_jp_{j+1}.

Konvexní obálka mezi dotykovými body tečen odstraněna a nahrazena dvojicí tečen.

Algoritmus 4: Upper and Lower Tangent

1: for $j = 0$ to $n - 1$
2: if XOR(p vlevo od/na $p_{j-1}p_j$, p vlevo od/na p_jp_{j+1})
3: p_j je dotykový bod.

Poznámka: $XOR(A, B) = (A \cap \overline{B}) \cup (\overline{B} \cap A)$.

28. Metoda zametací přímky

Realizace strategie inkrementální konstrukce.
Nadrovina rozděluje S na zpracovanou/nezpracovanou část.
Složitost $O(n \cdot \log(n))$, lze převést do vyšší dimenze.
Citlivost vůči singularitám: duplicitní body v P.

Princip algoritmu:
Předzpracování: setřídění S dle souřadnice x, složitost $O(n \cdot \log(n))$.
Nad P_S se pohybuje nadrovina σ (L->P) přes $\forall p_i \in P_s$.
Iniciální řešení tvořeno dvojúhelníkem/trojúhelníkem

$$\mathcal{H}(P_s(3)) \leftarrow \triangle(p_1, p_2, p_3).$$

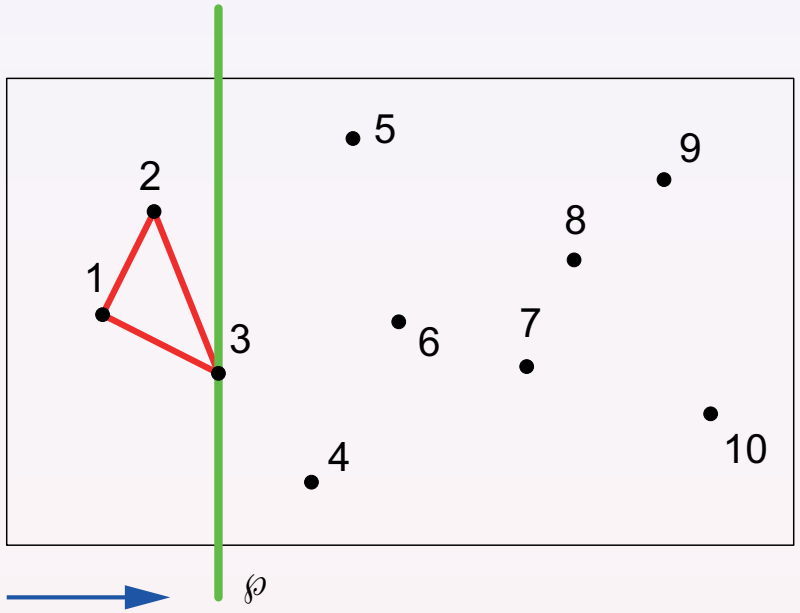
Pro přidávaný bod p updatována $\mathcal{H}(P_s(m))$, expanduje v kladném směru x

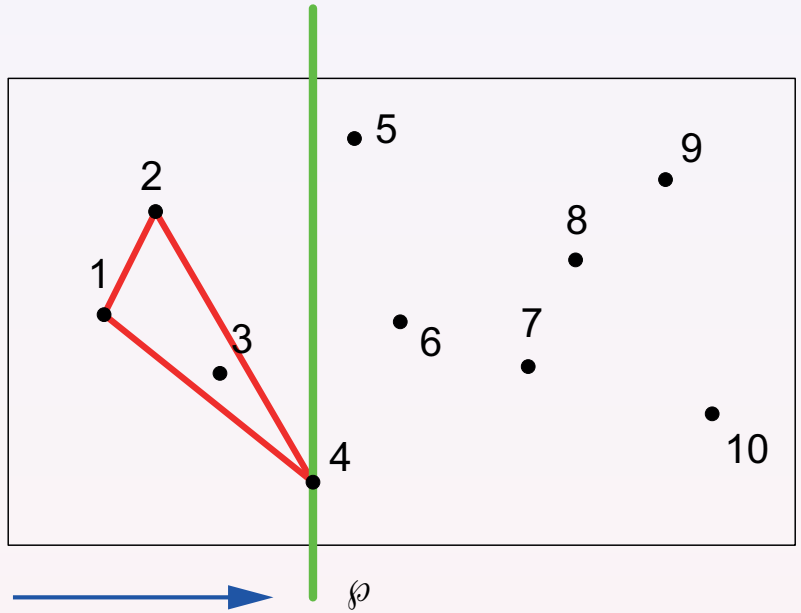
$$\mathcal{H}(P_s(m)) = \mathcal{H}(P_s(m-1) + \Delta\mathcal{H}(p).$$

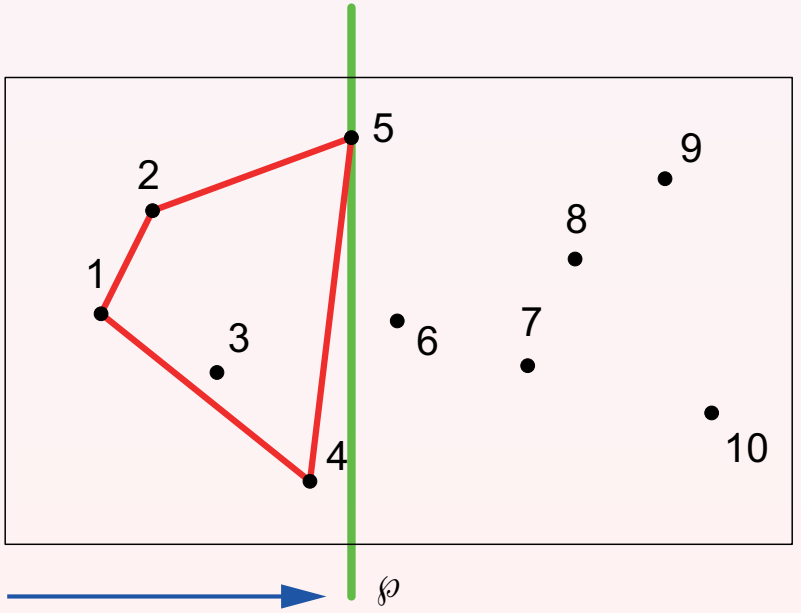
Algoritmus 1: Sweep Line (P, $\mathcal{H}$)

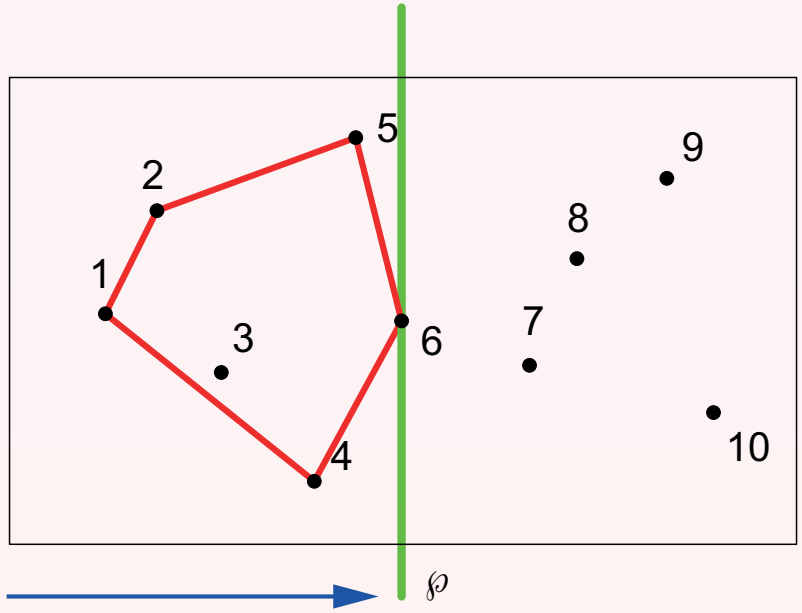
1: unique(P) //Odstraň duplicitní body
2: Sort $P_s = sort(P)$ by x
3: $\mathcal{H}(P_s(3)) \leftarrow \triangle(p_1, p_2, p_3)$ //Iniciální řešení pro $i \leq 3$
4: $i \leftarrow 4$.
5: Opakuj pro $\forall p_i \in P_s$
6: $\mathcal{H}(P_s(m)) = \mathcal{H}(P_s(m-1) + \Delta\mathcal{H}(p_i)$ //Aktualizace řešení

29. Sweep-line, konvexní obálka









30. Datový model

Použity celkem 3 seznamy, popis $\mathcal{H}$ formou *circular listu*.
První seznam obsahuje všechny vrcholy P_s
Dva pomocné seznamy: předchůdci `p[i]` a následníci `n[i]`.

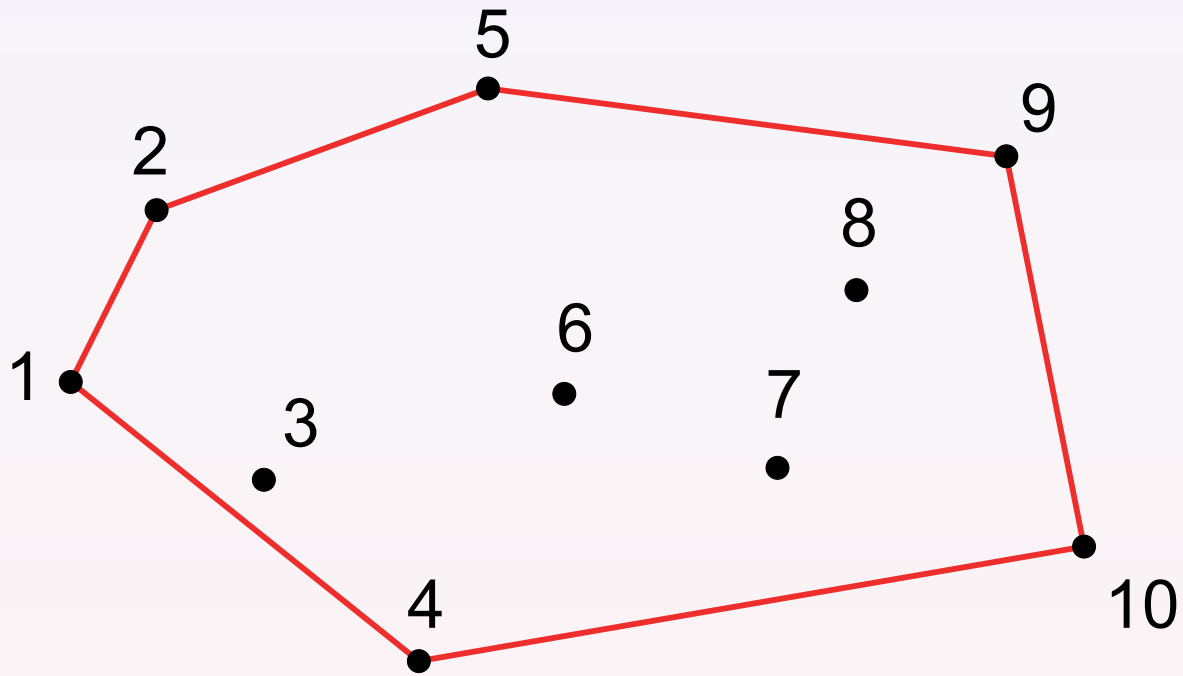
Vrchol bez předchůdce: p[i] = -1.
Vrchol bez následníka: n[i] = -1.
Pokud $p_i \in \mathcal{H}$, pak p[i] = -1, n[i] = -1

Příklady:

```
p[i] ..... předchůdce i-tého vrcholu, (i-1)-ty vrchol.
n[i] ..... následník i-tého vrcholu, (i+1)-ty vrchol.

p[p[i]]... předchůdce předchůdce, (i-2)-ty vrchol.
n[n[i]]... následník následníka, (i+2)-ty vrchol.
n[p[i]]... následník předchůdce, i-ty vrchol.
p[n[i]]... předchůdce následníka, i-ty vrchol.
```

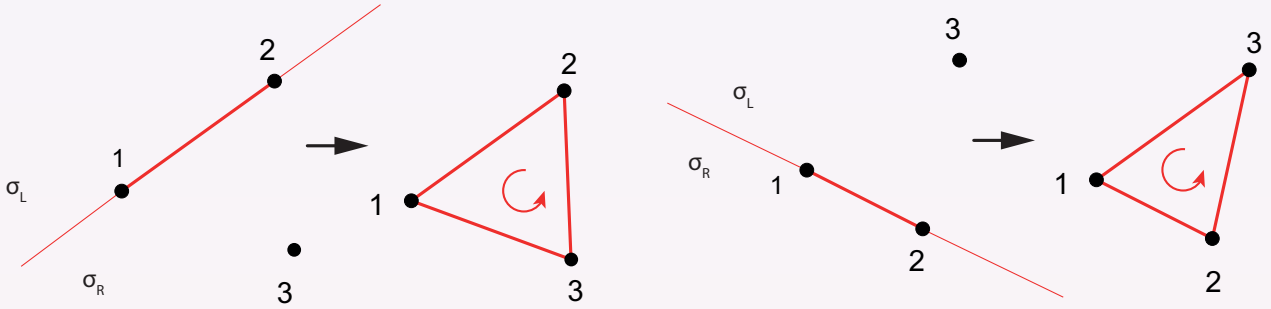
31. Ukázka reprezentace $\mathcal{H}$



	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
p:	-1	2	5	-1	1	9	-1	-1	-1	10	4
S:	x= ... y= ...	x= ... y= ...	x= ... y= ...	x= ... y= ...	x= ... y= ...	x= ... y= ...	x= ... y= ...	x= ... y= ...	x= ... y= ...	x= ... y= ...	x= ... y= ...
n:	-1	4	1	-1	10	2	-1	-1	-1	5	9

32. Iniciální fáze

První hrana tvořená body p_1, p_2, kde $x_0 = \min_{\forall p_i \in P_s}(x_i)$.
Třetí bod p_3 v levé nebo pravé polorovině, nutno zachovat CCW orientaci.



Algoritmus 1: Initial phase(P, $\mathcal{H}$)

1: if $(p_3 \in \sigma_L(p_1, p_2))$ //$\triangle(p_1, p_2, p_3)$
2: n[1] = 2; n[2] = 3; n[3] = 1
3: p[1] = 3; p[2] = 1; p[3] = 2
4: else //$\triangle(p_1, p_3, p_2)$
5: n[1] = 3; n[3] = 2; n[2] = 1
6: p[1] = 2; p[3] = 1; p[2] = 3

Alternativně lze vytvořit “dvojúhelník”:
1: n[1] = 2; n[2] = 1
2: p[1] = 2; p[2] = 1

33. Iterativní fáze I.

Vytvoření aproximace $\overline{\mathcal{H}}$ se zachovanou CCW orientací.
$\overline{\mathcal{H}}$ nemusí být konvexní, převod na $\mathcal{H}$.

Rozpojení $\mathcal{H}$ mezi body $\{i-1, n[i-1]\}$ resp. $\{p[i-1], i-1\}$.
Náhrada sekvence vrcholů:

$$\{p[i-1], i-1, n[i-1]\} \Rightarrow \begin{cases} \{p[i-1], i-1, i, n[i-1]\}, & y > y_{i-1}, \\ \{p[i-1], i, i-1, n[i-1]\}, & y \leq y_{i-1}. \end{cases}$$

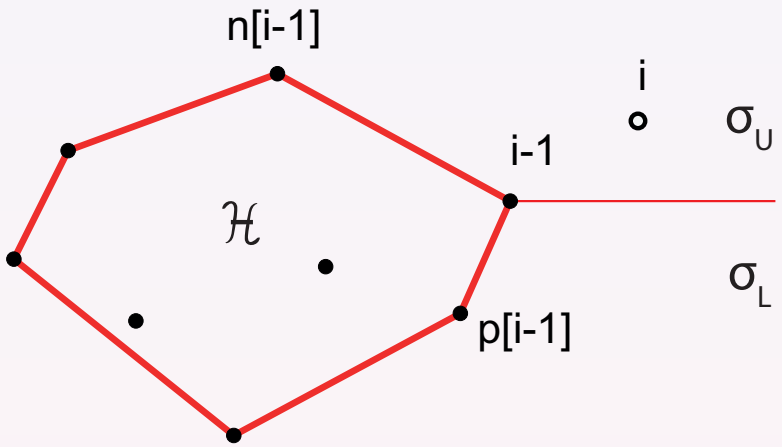
Algoritmus 1: Iterativní fáze I., konstrukce $\overline{\mathcal{H}}$

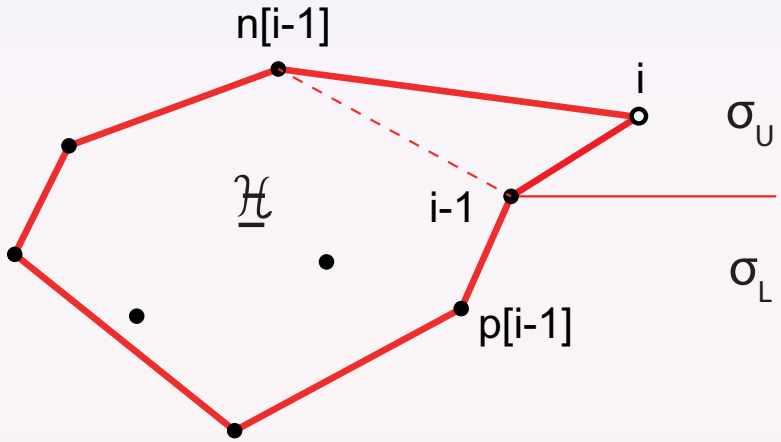
1: if $(y_i > y_{i-1})$ //$p_i \in \sigma_U$
2: p[i] = i-1; n[i] = n[i-1]; // Spojeni p[i] s predchudcem/naslednikem
3: else //$p_i \in \sigma_L$
4: n[i] = i-1; p[i] = p[i-1]; // Spojeni p[i] s predchudcem/naslednikem
4: n[p[i]] = i; //Propojeni predchudce s p[i]
4: p[n[i]] = i; //Propojeni naslednika s p[i]

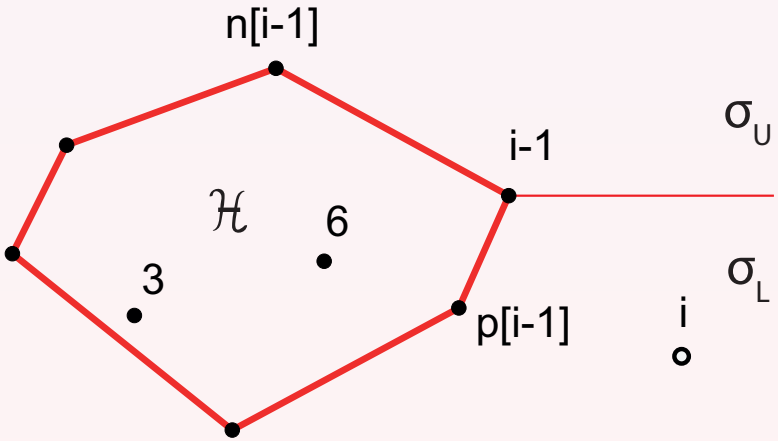
Oboustranné korektní napojení segmentů v $\overline{\mathcal{H}}$:

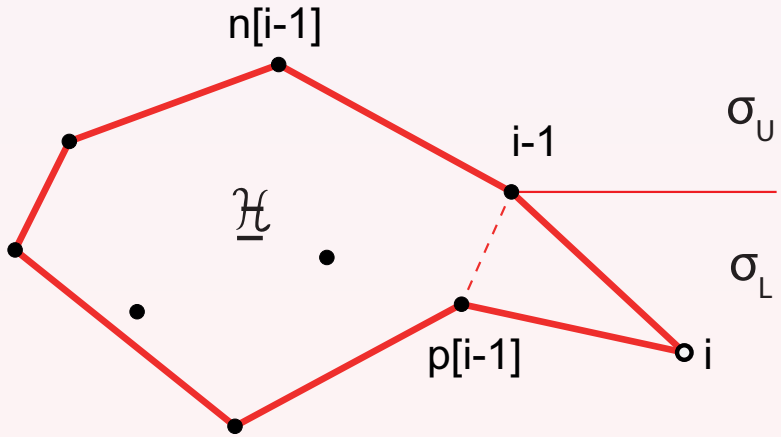
$$\{p[p[i]], p[i], i, n[i], n[n[i]]\}$$

34. Iterativní fáze I, ukázka.

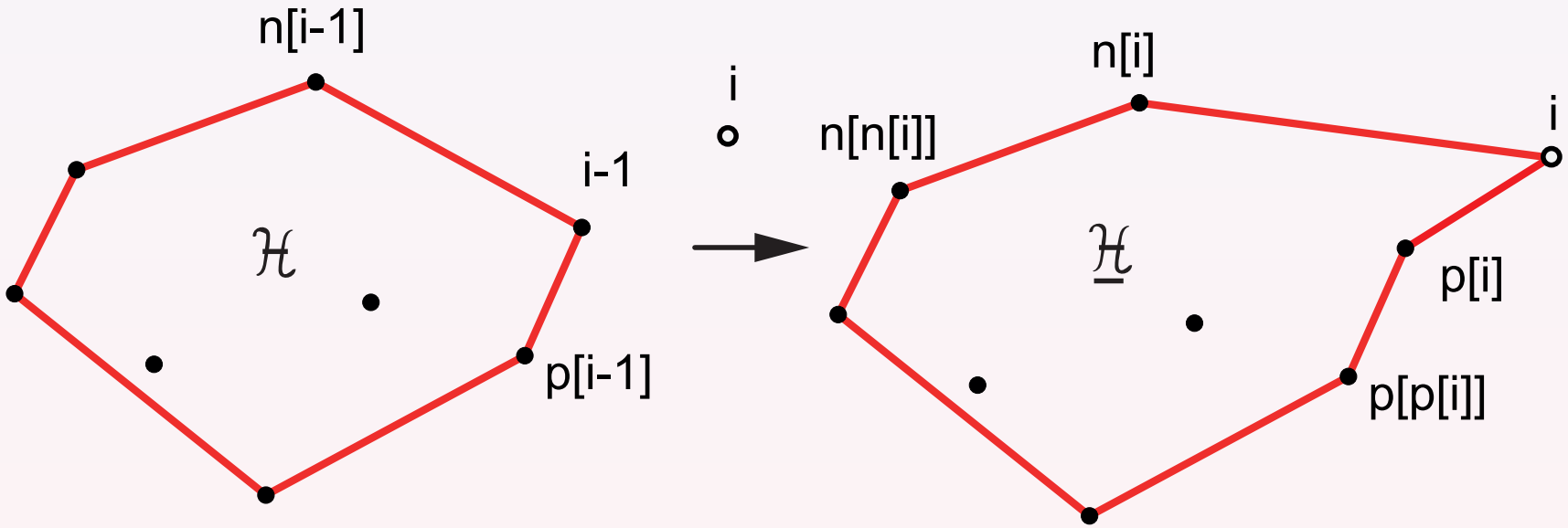








35. Iterativní fáze I, indexy.



36. Iterativní fáze II.

Převod nekonvexní $\overline{\mathcal{H}}$ na $\mathcal{H}$.
Náhrada sekvence nekonvexních vrcholů horní a dolní tečnou.
Nekonvexní vrcholy vynechány.

Horní tečna:
Nekonvexita $n[i]$ v trojici $\{i, n[i], n[n[i]]\}$ řešena vynecháním:

$$\{i, n[i], n[n[i]]\} \Rightarrow \{i, n[n[i]]\}.$$

Přemostění vrcholu možnou horní tečnou.

Dolní tečna:
Nekonvexita $p[i]$ v trojici $\{p[p[i]], p[i], i\}$ řešena vynecháním:

$$\{p[p[i]], p[i], i\} \Rightarrow \{p[p[i]], i\}.$$

Přemostění vrcholu možnou dolní tečnou.

Algoritmus 1: Iterativní fáze II., převod $\overline{\mathcal{H}}$ na $\mathcal{H}$

1: while $(n[n[i]]) \in \sigma_P(i, n[i])$ //Neni horni tecna: $n[n[i]]$) vpravo od $(i, n[i])$

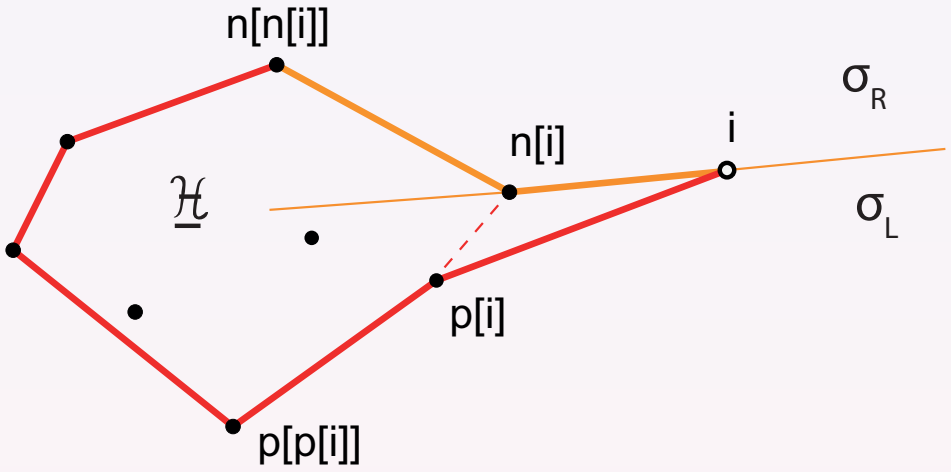
2: p[n[n[i]]] = i; n[i] = n[n[i]]; //Vynech n[i], aproximuj horni tecnou, poradi!

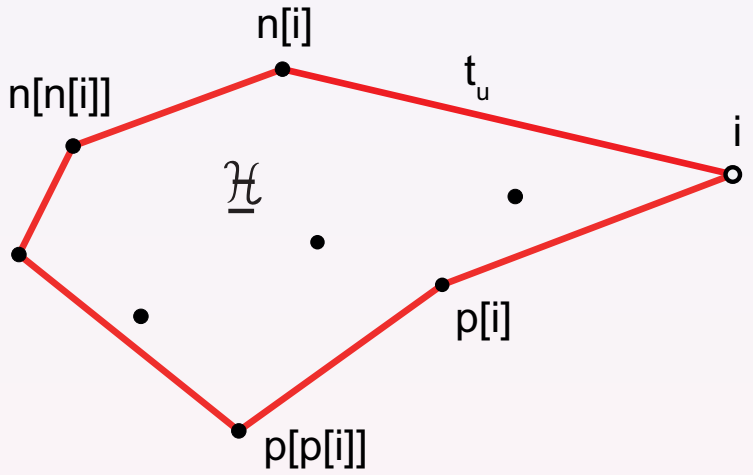
3: while $(p[p[i]]) \in \sigma_L(i, p[i])$ //Neni dolni tecna: $p[p[i]]$) vlevo od $(i, p[i])$

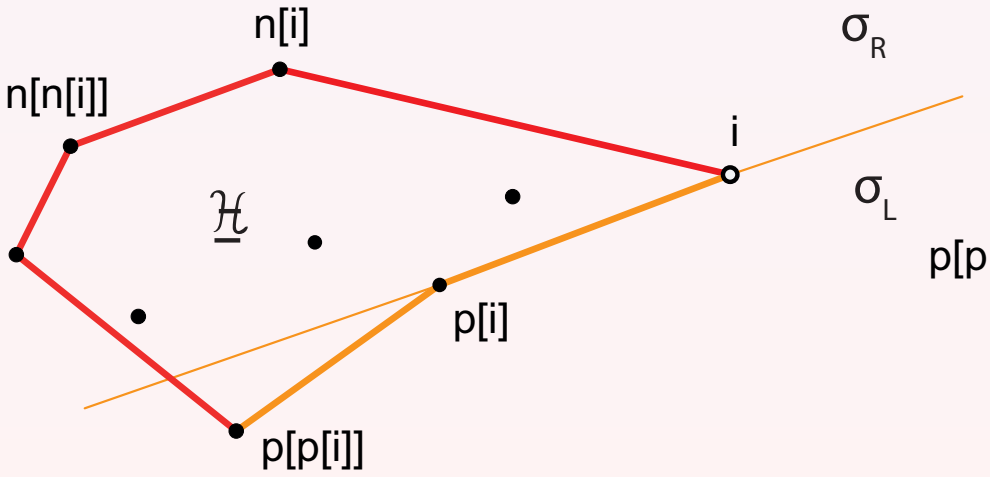
4: p[p[p[i]]] = i; p[i] = p[p[i]]; //Vynech p[i], aproximuj dolni tecnou, poradi!

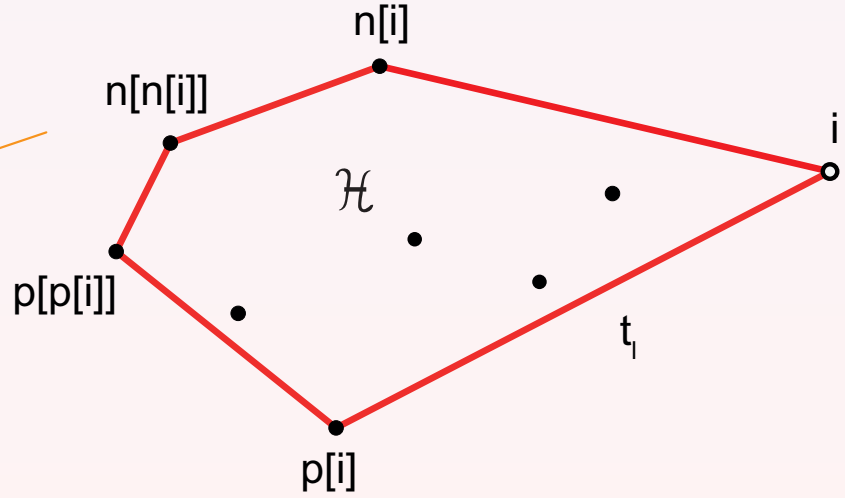
---

37. Iterativní fáze II., ukázka









38. Algoritmus Sweep Line, kompletní

Algoritmus 1: Sweep Line

1: Sort $P_s = sort(P)$ by x
2: if $(p_3 \in \sigma_L(p_1, p_2))$
3: n[1] = 2; n[2] = 3; n[3] = 1
4: p[1] = 3; p[2] = 1; p[3] = 2
5: else
6: n[1] = 3; n[3] = 2; n[2] = 1
7: p[1] = 2; p[3] = 1; p[2] = 3
8: for $p_i \in P_s$, $i > 3$
9: if $(y_i > y_{i-1})$
10: p[i] = i-1; n[i] = n[i-1];
11: else
12: n[i] = i-1; p[i] = p[i-1];
13: n[p[i]] = i; p[n[i]] = i;
14: while $(n[n[i]]) \in \sigma_R(i, n[i])$
15: p[n[n[i]]] = i; n[i] = n[n[i]]; //Poradi!
16: while $(p[p[i]]) \in \sigma_L(i, p[i])$
17: n[p[p[i]]] = i; p[i] = p[p[i]]; //Poradi!

39. Konverze reprezentace

Přechod na reprezentaci $\mathcal{H}$ tvořenou posloupností vrcholů.
Uchování CCW orientace, procházíme následníky.
Zapamatován index prvního vrcholu polygonu.
Složitost $O(n)$.

Algoritmus 1: Sweep Line: konverze reprezentace

1: $\mathcal{H} \leftarrow p_1$
2: $i =$n[1])
3: do
4: $\quad\quad\quad\quad \mathcal{H} \leftarrow p_i$
5: $\quad\quad\quad\quad i =$n[i]
6: while $(i \neq 1)$

---

40. Algoritmus Divide and Conquer

Předpoklad: žádná trojice bodů není kolineární a žádné dva body v datasetu nemají stejné souřadnice x.

Dělení S na dvě podmnožiny A, B a jejich následné spojení s nalezením horní a dolní tečny t_1, t_2.

Algoritmus 5: Divide and Conquer.

1: Setřídění S podle x.
2: Rozdělení S na dvě podmnožiny A, B obsahující $n/2$ bodů.
3: Konstrukce konvexních obálek $\mathcal{H}(A)$ a $\mathcal{H}(B)$.
4: Spojení konvexních obálek $\mathcal{H} = \mathcal{H}(A) \cup \mathcal{H}(B)$.
5: Nalezení dolní tečny t_1.
6: Nalezení horní tečny t_2.
7: Nahrazení úseků konvexních obálek A, B mezi oběma tečnami.

---

41. Algoritmus nalezení dolní tečny t_1

Existence řady algoritmů pro spojení dvojice konvexních obálek.
Ukázka spojení nalezení dolní tečny t_1 “procházkou”, Walking Algorithm (Preprata & Hong, 1977).

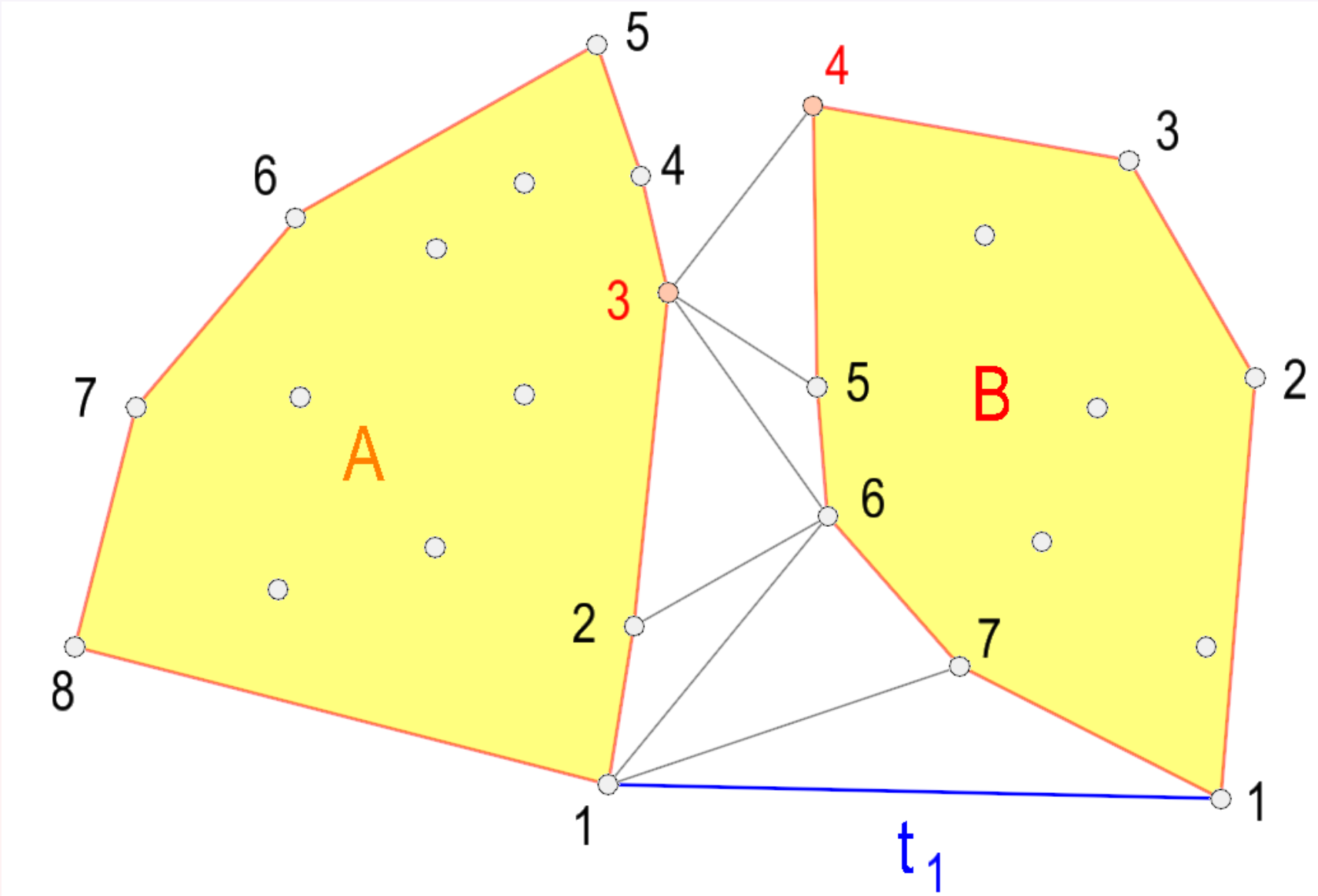
Z extrémních bodů a, b množin A, B nalezneme k bodu a bod b takový, že jejich spojnice je dolní tečnou B.
Z bodu b hledáme bod a takový, aby jejich spojnice byla dolní tečnou A.
Postup opakujeme, dokud a, b není dolní tečnou A i B.

Algoritmus 6: Lower Tangent t_1, Walk

1: Najdi bod a: nejpravější bod A.
2: Najdi bod b: nejpravější bod B.
3: Opakuj, dokud $t_1 = ab$ není dolní tečnou A a B.
4: Opakuj, dokud t_1 není dolní tečnou A:
5: $a = a - 1$.
6: Opakuj, dokud t_1 není dolní tečnou B:
7: $b = b + 1$

---

42. Princip nalezení dolní tečny



43. Využití konvexní obálky pro konstrukci obdélníka s minimální plochou

Dáno:	Množina n bodů S v $\mathbb{R}^2$.
Hledáme:	Obdélník $\mathcal{R}$ s nejmenší plochou opsaný S. (Minimum Area Enclosing Rectangle)

Naivní algoritmy problém řeší v kvadratickém čase $O(n^2)$.

Definice:
Nejméně jedna strana obdélníka $\mathcal{R}$ s minimální plochou A opsaného množině S je kolineární se stranou $\mathcal{H}$.

Algoritmy pro konstrukci obdélníka $\mathcal{R}$ s minimální plochou opsaného S pracují s konvexní obálkou $\mathcal{H} \Rightarrow$ předzpracování.

Složitost konstrukce konvexní obálky $\mathcal{H}$: $O(n \cdot \log(n))$.
Složitost konstrukce obdélníku $\mathcal{R}$ nad $\mathcal{H}$: pouze $O(n)$!

Algoritmus řeší problém v čase $O(n \cdot \log(n))$.
Alternativa: metoda Rotating Calipers (Toussaint, 1983), složitost $O(n)$.

Využití v kartografii pro detekci tvaru a natočení budov.

44. Princip konstrukce, opakovaná rotace

Body S s extrémními souřadnicemi: $\underline{x}$, $\overline{x}$, $\underline{y}$, $\overline{y}$.
Snadno umíme sestrojit MMB množiny S, který je $\|$ s osami x, y

$$V_1 = [\underline{x}, \underline{y}], \qquad V_2 = [\overline{x}, \underline{y}], \qquad V_3 = [\overline{x}, \overline{y}] \qquad V_4 = [\underline{x}, \overline{y}].$$

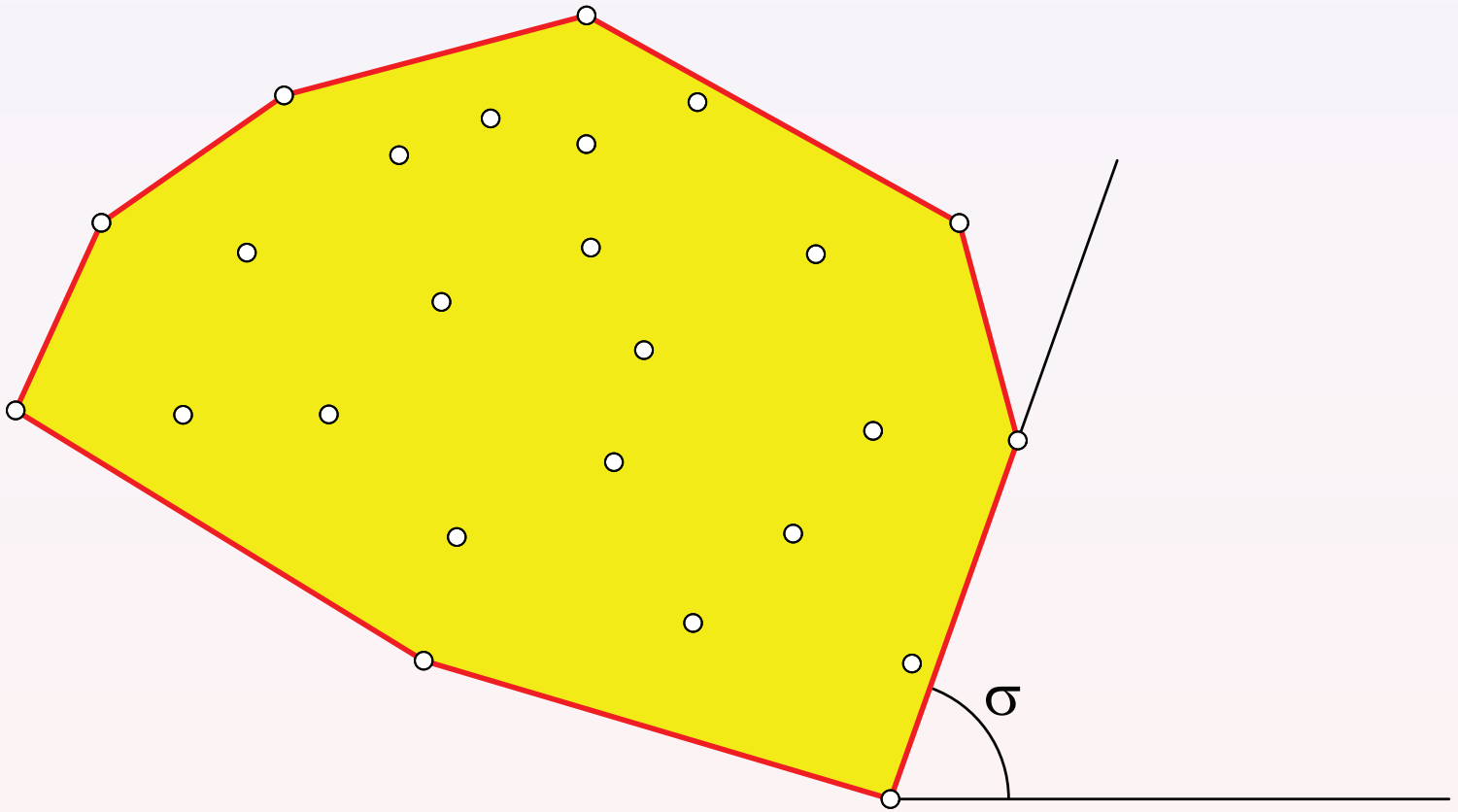
Myšlenka:

- Opakované otáčení S o úhel $-\sigma$ (směrnice segmentu $\mathcal{H}$).
- V této poloze nalezneme MMB množiny S a jeho plochu A.
- Otočením o σ min-max transformován na obecný obdélník (A se nezmění).
- Obdélník s minimální plochou bude $\mathcal{R}$.

Algoritmus 6: Minimum Area Enclosing Rectangle

1: Najdi $\mathcal{H} = CH(S)$
2: Inicializuj $\mathcal{R} = MMB(S)$, $\underline{A} = A(MMB(S))$
3: Opakuj pro každou hranu e obálky $\mathcal{H}$:
4: Spočti směrnici σ hrany e
5: Otoč S o $-\sigma$: $S_r = R(-\sigma)S$.
6: Najdi $MMB(S_r)$ a urči $A(MMB(S_r))$.
7: Pokud $A < \underline{A}$
8: $\underline{A} = A$, $\underline{MMB} = MMB$, $\underline{\sigma} = \sigma$.
9: $\mathcal{R} = R(\underline{\sigma})\underline{MMB}$

45. Směrnice σ hrany

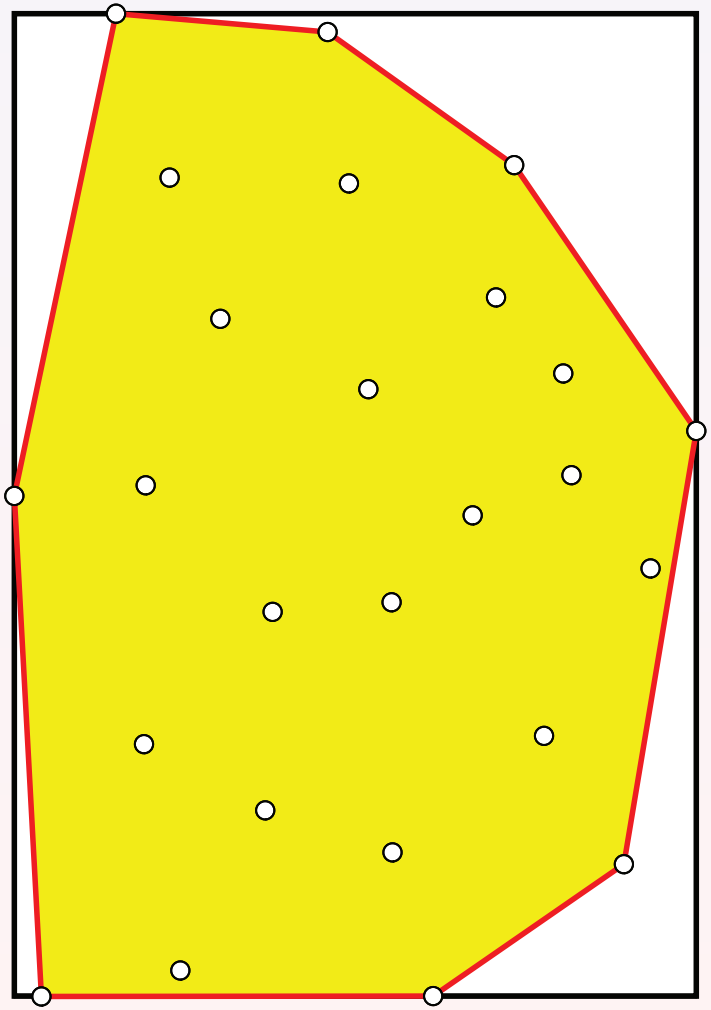


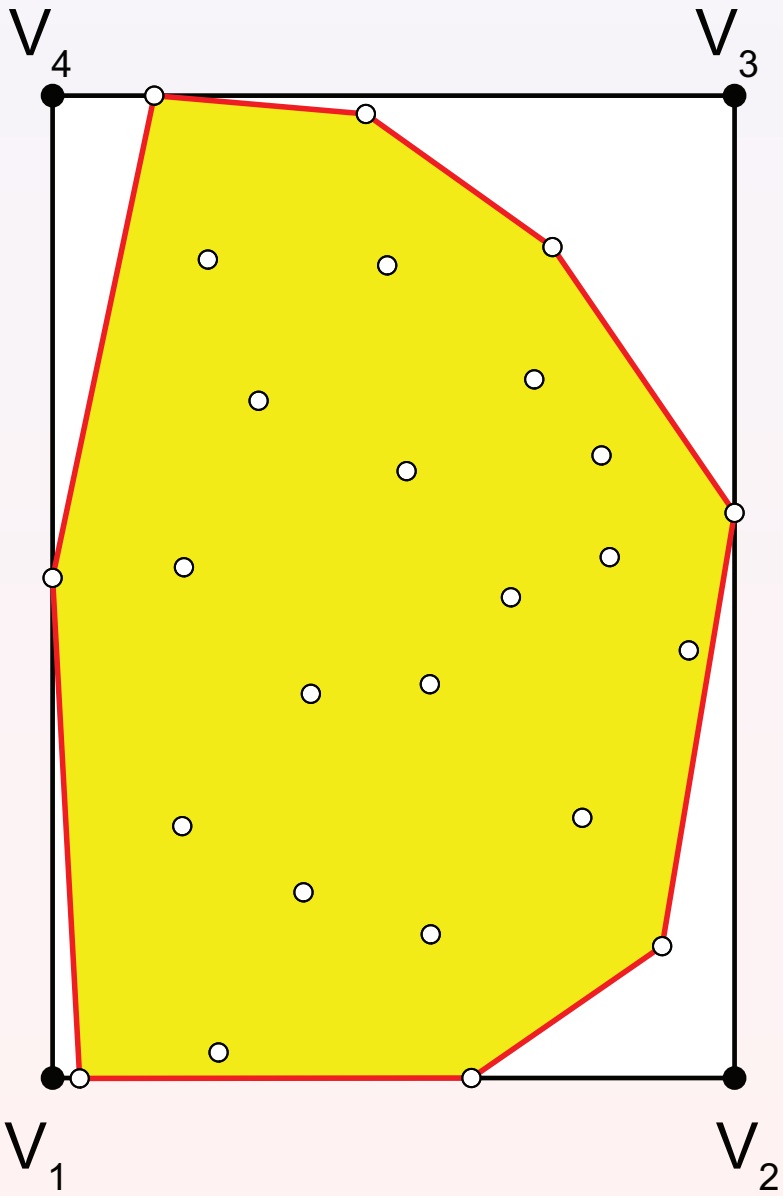
Rotace S o úhel $-\sigma$

$$S_r = R(-\sigma)S \Rightarrow \begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} \cos(-\sigma) & -\sin(-\sigma) \\ \sin(-\sigma) & \cos(-\sigma) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

46. Rotace S o úhel $-\sigma$

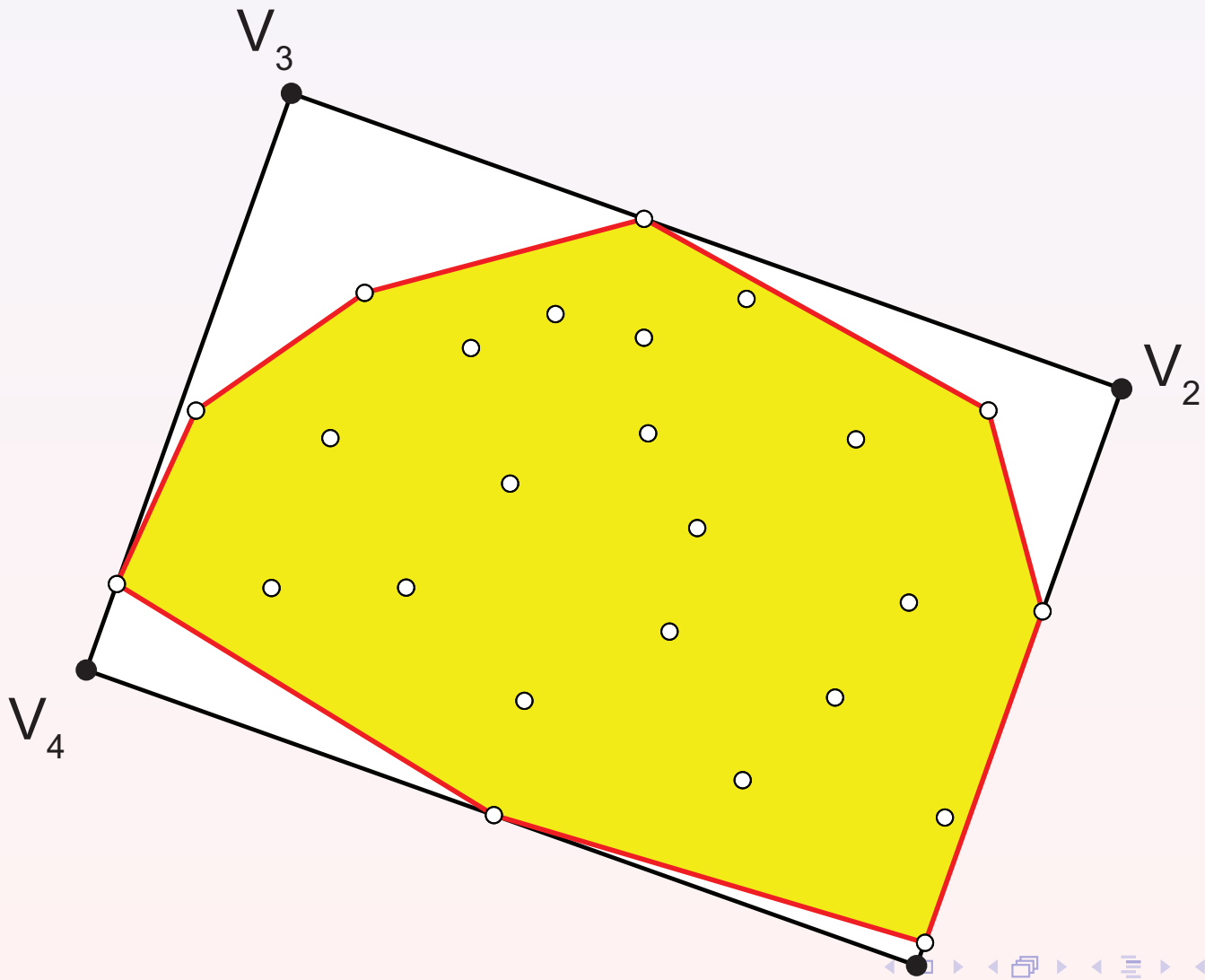
Rotace vrcholů S o $-\sigma$, nalezení extrémních bodů, konstrukce MMB.





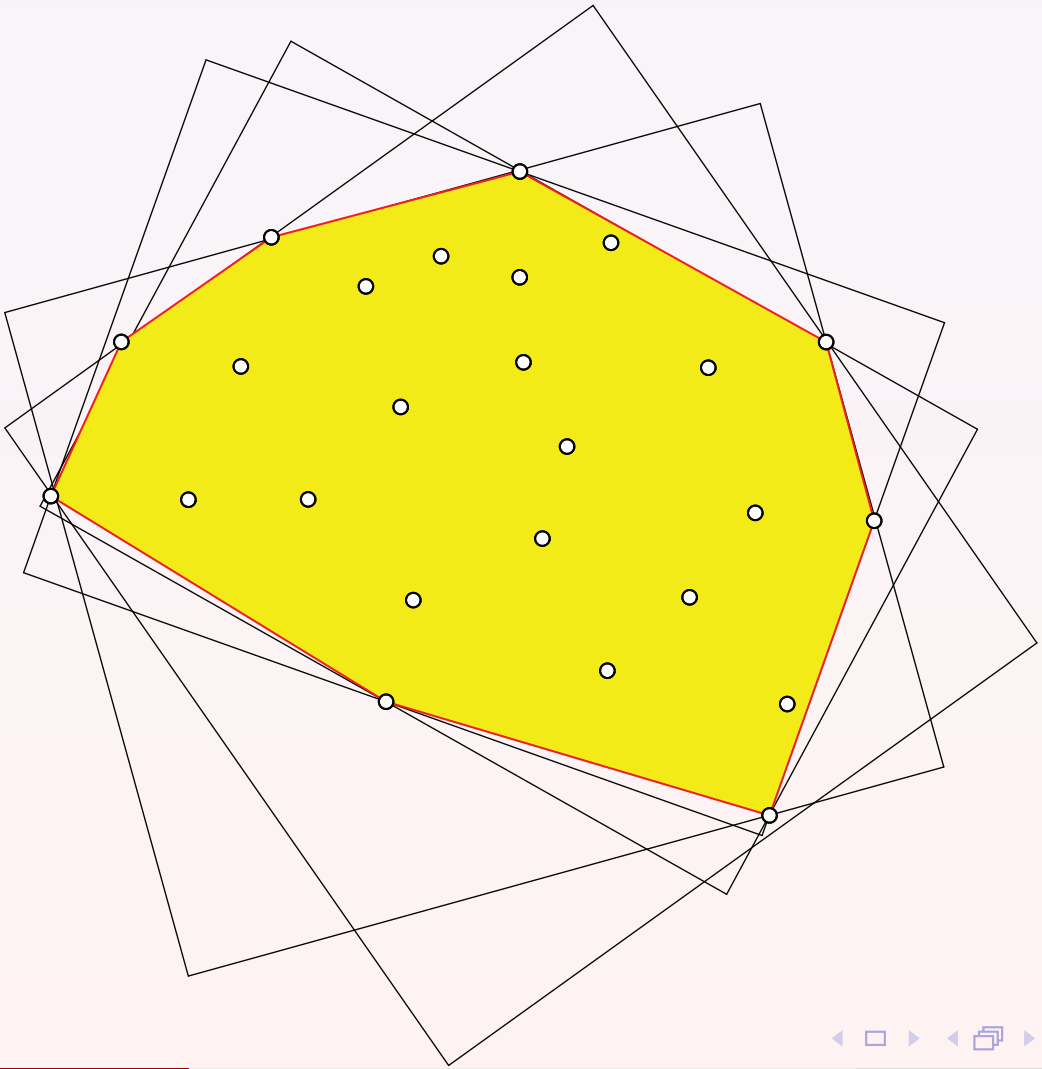
47. Zpětná rotace S o úhel σ

Plocha MMB se nezmění, výsledkem obecný obdélník.

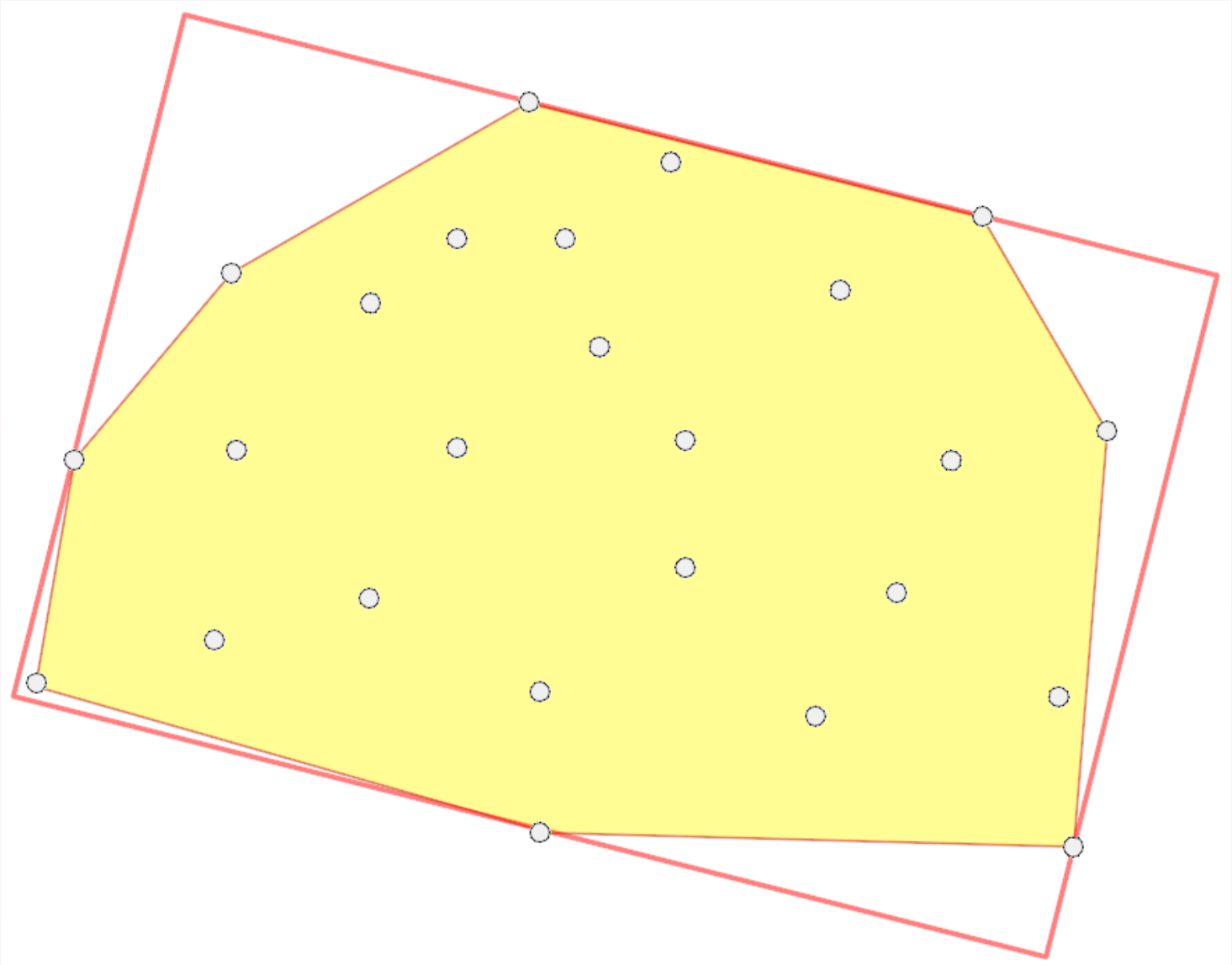


48. Postupné rotace

Opakované rotace $-\sigma$, konstrukce MMB, zpětné rotace o σ.
Chyby ze zaokrouhlení.



49. Výsledek, obdélník s nejmenší plochou



50. Detekce úhlu natočení budovy

Častý problém při kartografické generalizaci budov či jejich automatickém rozpoznávání.

Budova před generalizací a po generalizaci musí mít uchovánu orientaci vzhledem k ostatním obsahovým prvkům mapy (např. zachování uliční čáry).

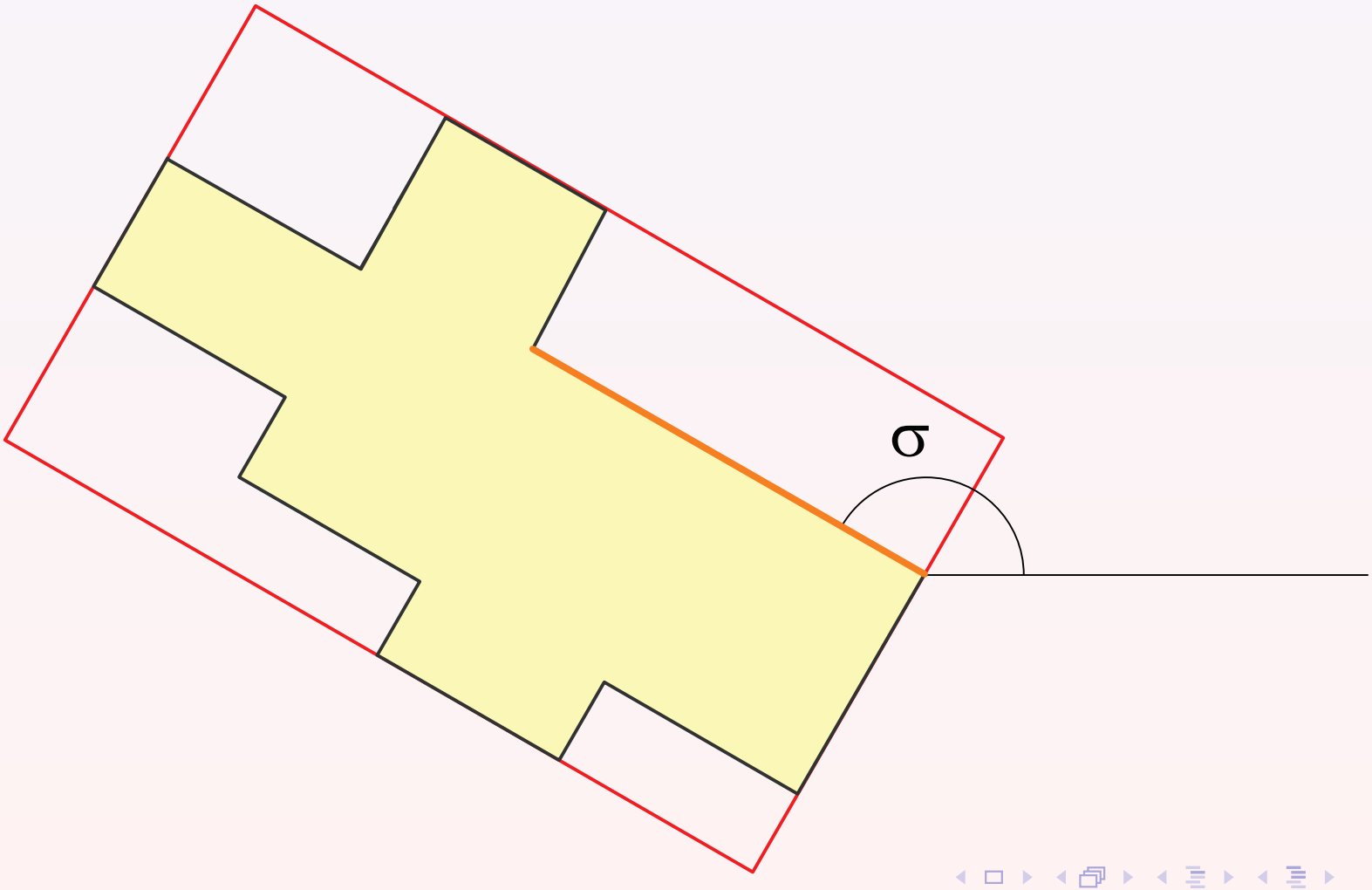
Nutnost detekovat tzv. hlavní **směry budovy** (jsou na sebe zpravidla kolmé). Popisují orientaci (tj. natočení) budovy vzhledem k ostatním prvkům mapy.

Metody detekce natočení budov:

- Longest Edge.
- Weighted Bisektor
- Minimum Area Enclosing Rectangle.
- Wall Average.

51. Longest Edge

První hlavní směr budovy představován nejdelší stranou v budově, druhý hlavní směr na ní kolmý.
Nedosahuje příliš dobrých výsledků.
Nejdelší strana nemusí reprezentovat hlavní směr.

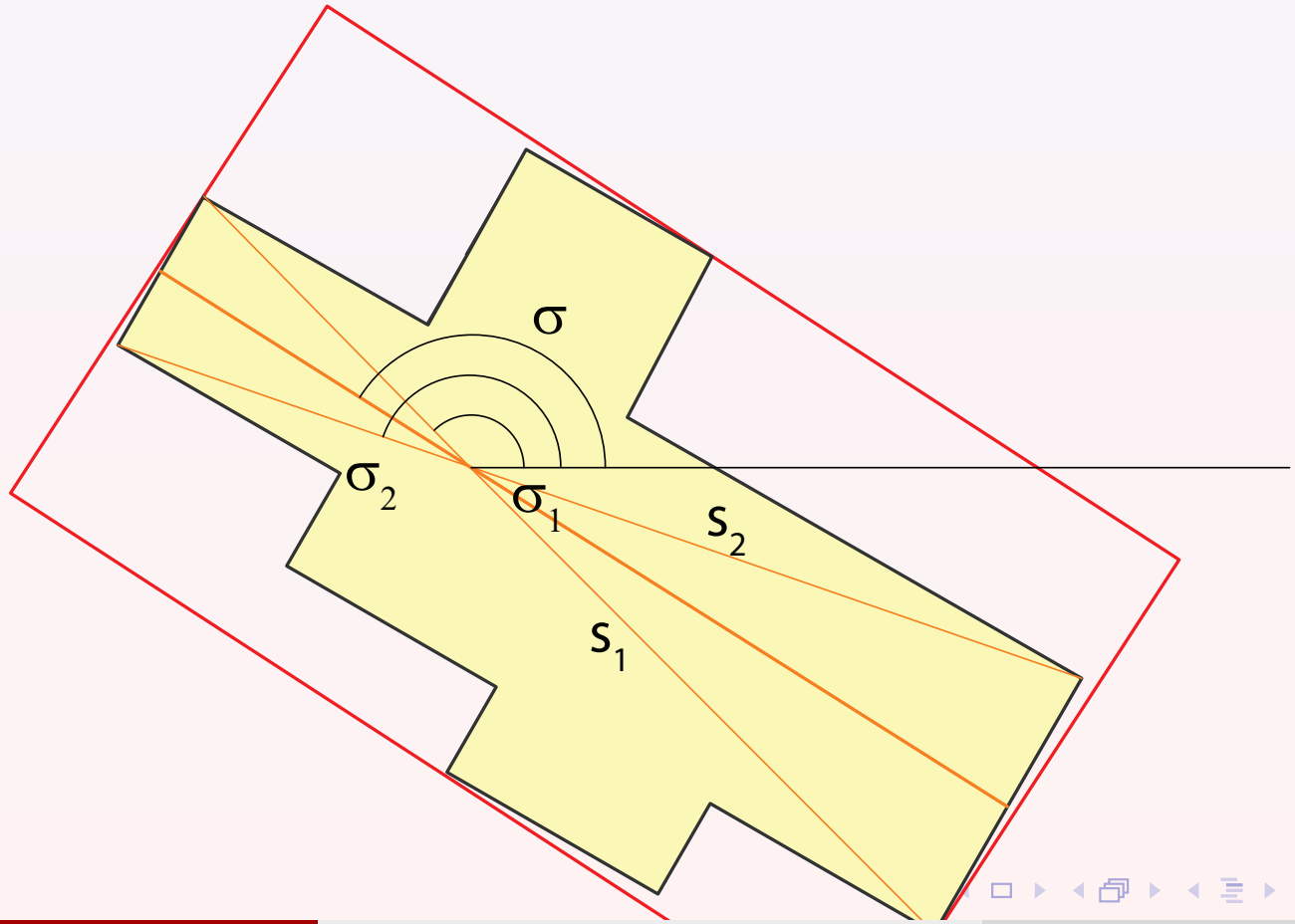


52. Weighted Bisector

Hledány dvě nejdelší úhlopříčky, směrnice σ_1, σ_2, délky s_1, s_2.
Hlavní směr dán váženým průměrem

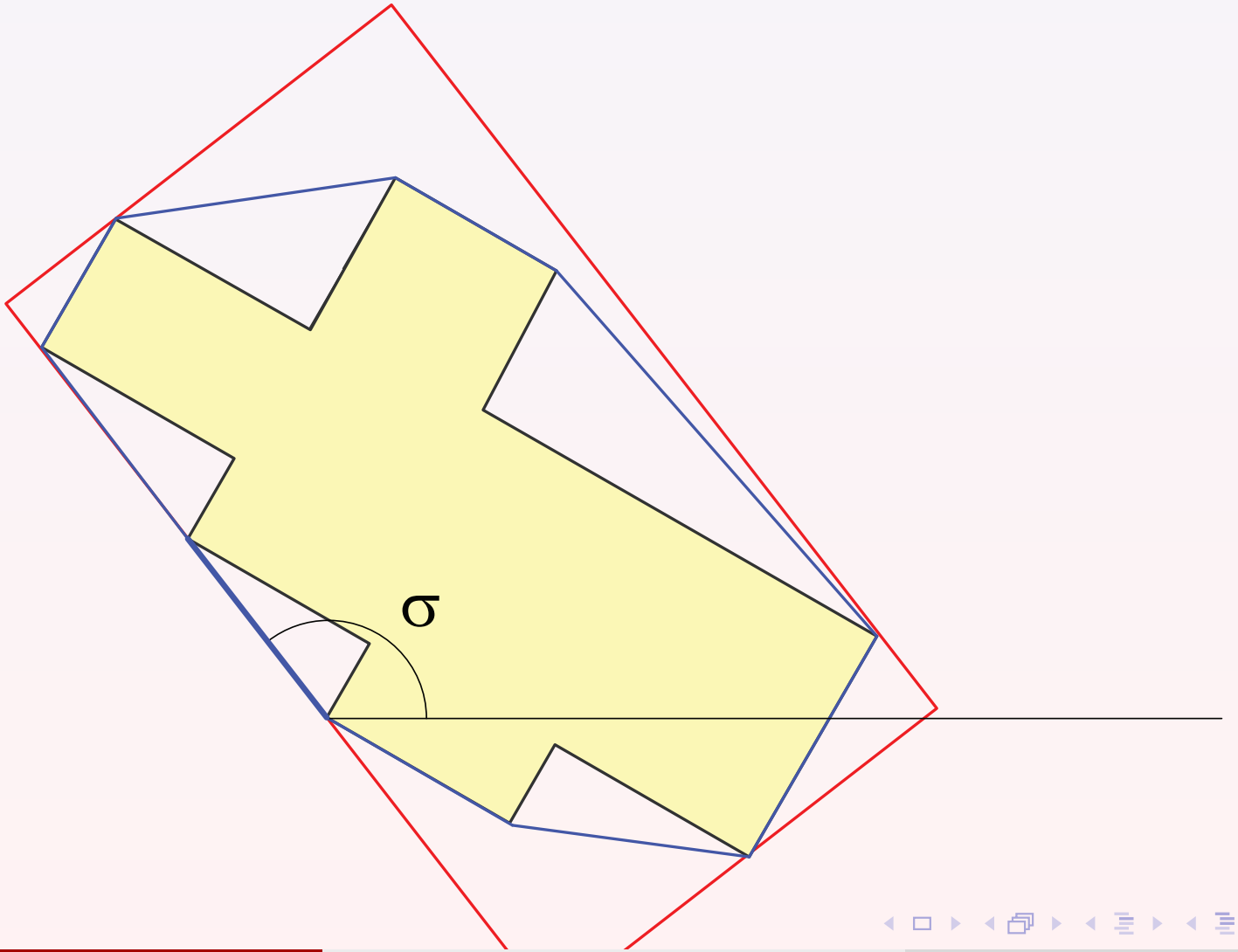
$$\sigma = \frac{s_1\sigma_1 + s_2\sigma_2}{s_1 + s_2}.$$

Dává velmi dobré výsledky.



53. Minimum Area Enclosing Rectangle

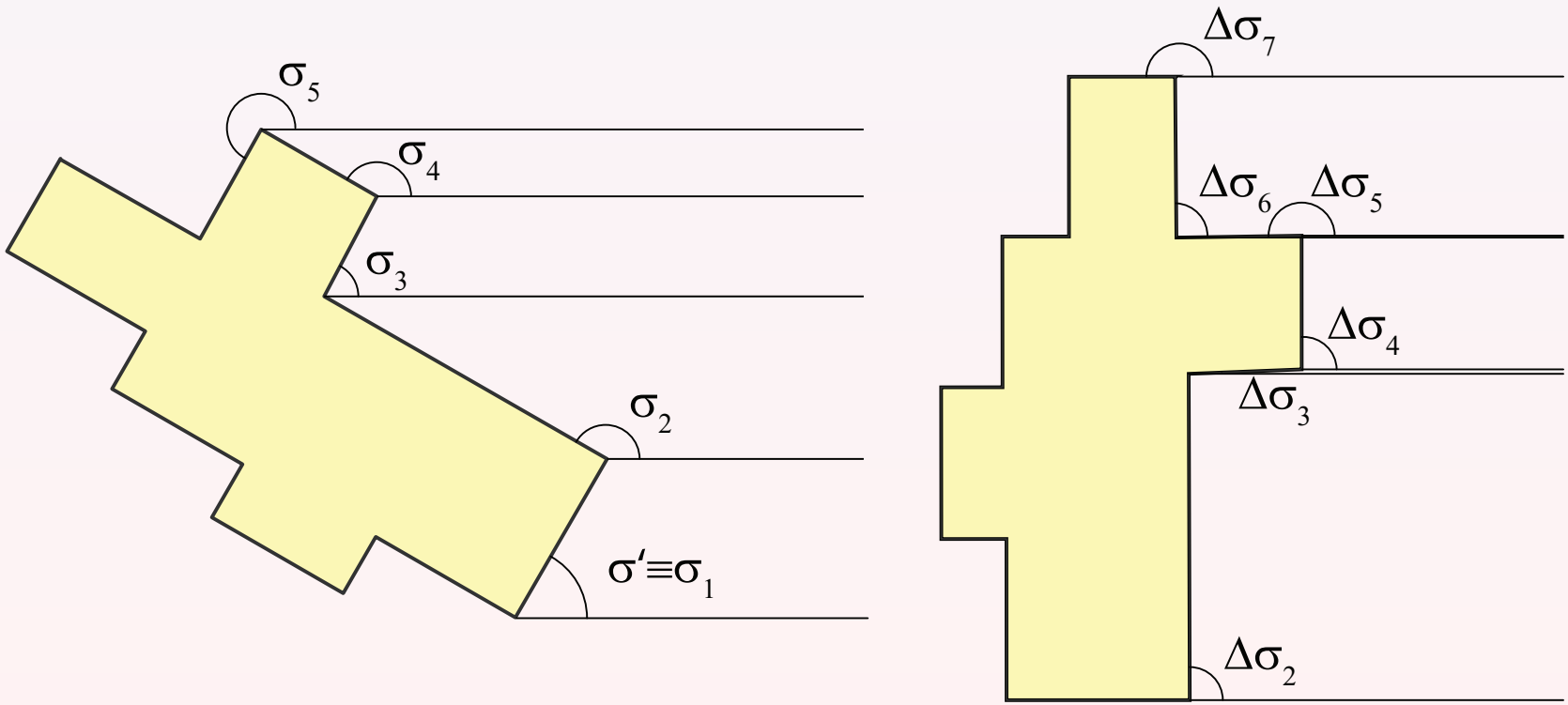
První hlavní směr představuje delší ze stran $\mathcal{R}$.
Dává dobré výsledky, problémy s budovami tvaru L a Z.



54. Wall Average

Na každou stranu budovy aplikována operace $mod(\frac{\pi}{2})$.
Ze “zbytků” hodnot spočten vážený průměr, váhou je délka strany.
Nejkomplexnější metoda, citlivá na “nepravé” úhly.

Nejprve určeny směrnice σ_i všech hran.
Poté redukce σ_i o úhel σ' (natočení budovy || s osami x, y).



55. Wall Average: výpočet zbytku

Pro každou hranu budovy redukujeme směrnice

$$\Delta\sigma_i = \sigma_i - \sigma'.$$

Výpočet zaokrouhleného podílu

$$k_i = \left\lfloor \frac{2\Delta\sigma_i}{\pi} \right\rceil.$$

Výpočet zbytku, odchylky od $0 \pm k\pi$ resp. $\pi/2 \pm k\pi$

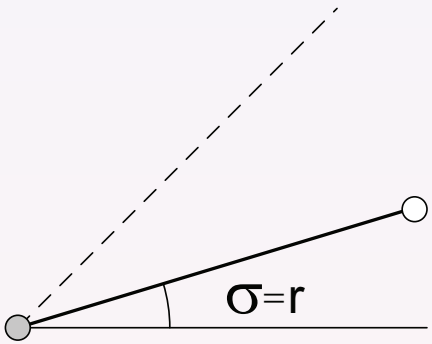
$$r_i = \Delta\sigma_i - k_i \frac{\pi}{2}.$$

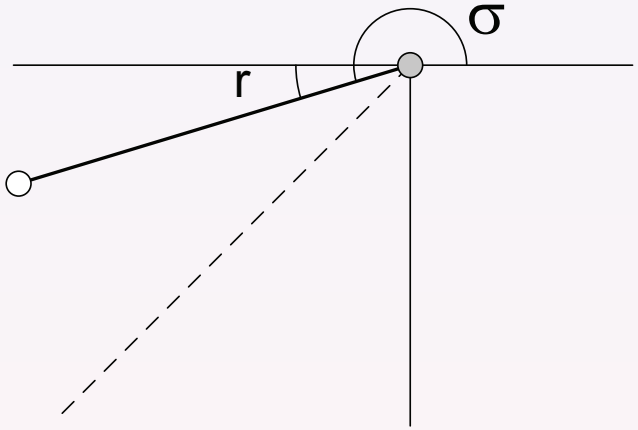
Segmenty se zbytkem po dělení $< \frac{\pi}{4}$: odchylka od vodorovného směru $(0 \pm k\pi)$.
Segmenty se zbytkem po dělení $> \frac{\pi}{4}$: odchylka od svislého směru $(\frac{\pi}{2} \pm k\pi)$.

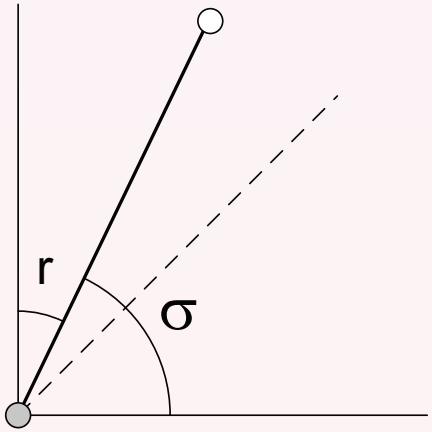
Směr natočení budovy dán váženým průměrem

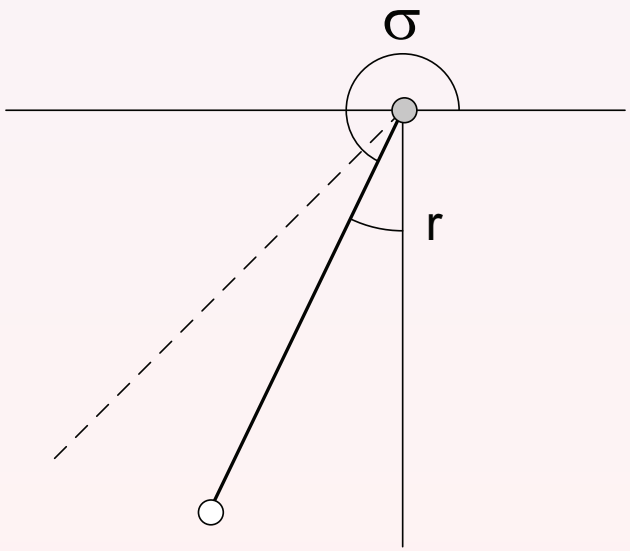
$$\sigma = \sigma' + \sum_{i=1}^{n} \frac{r_i s_i}{s_i}.$$

56. Wall Average: ukázka výpočtu zbytku



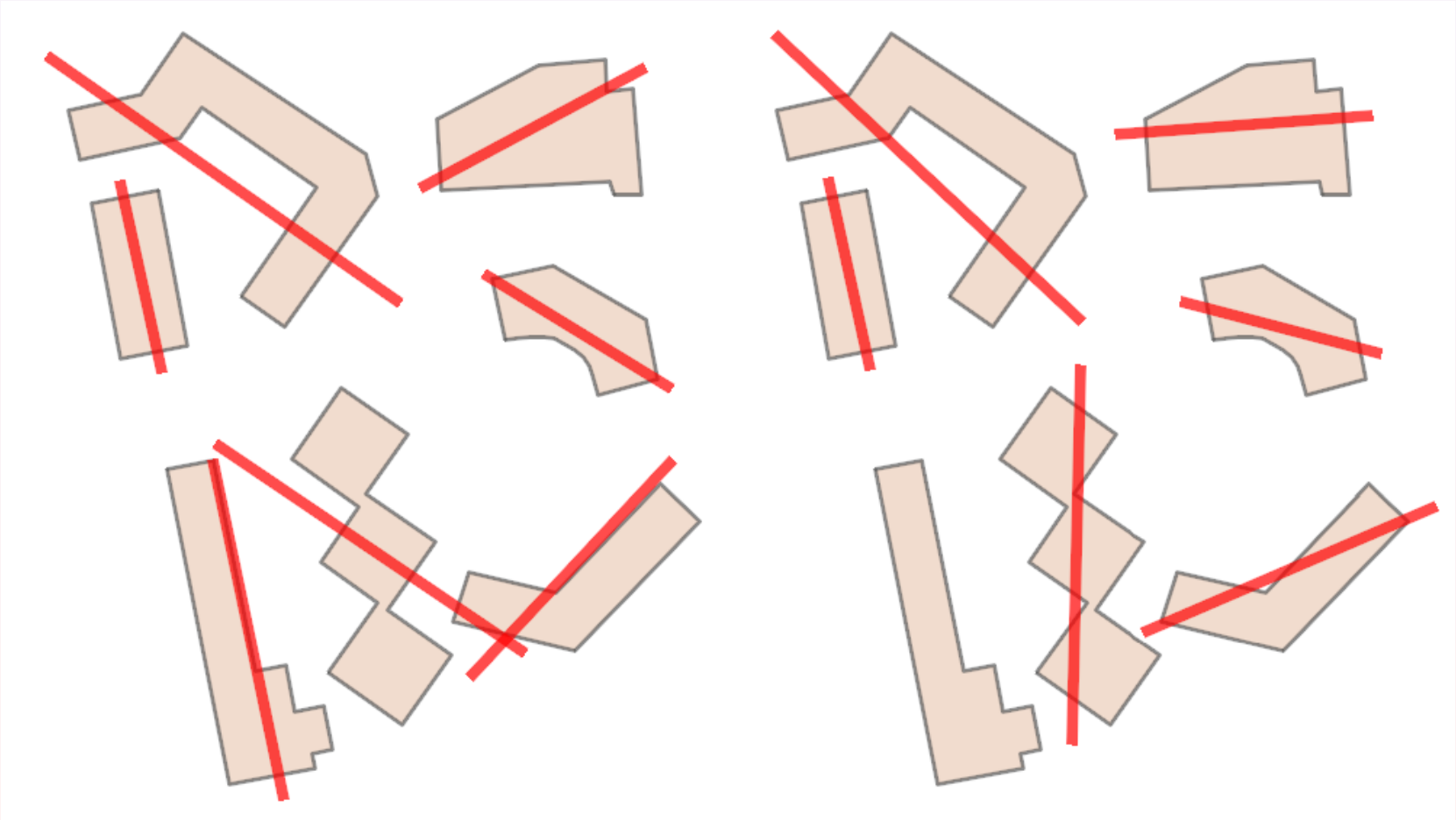






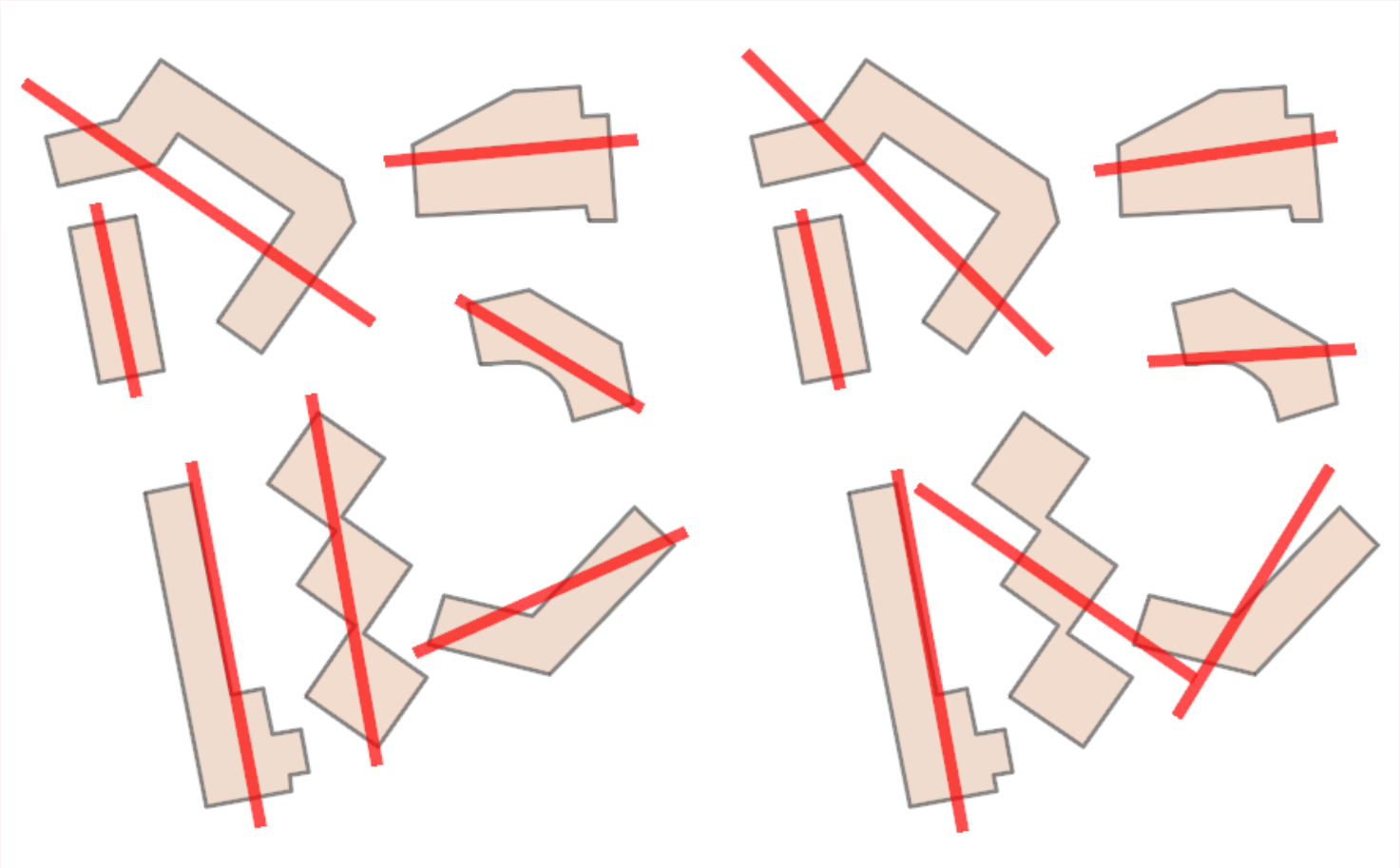
57. Detekce hlavních směrů budovy

Metody Longest Edge a Wall Average (Duchene et al, 2003).



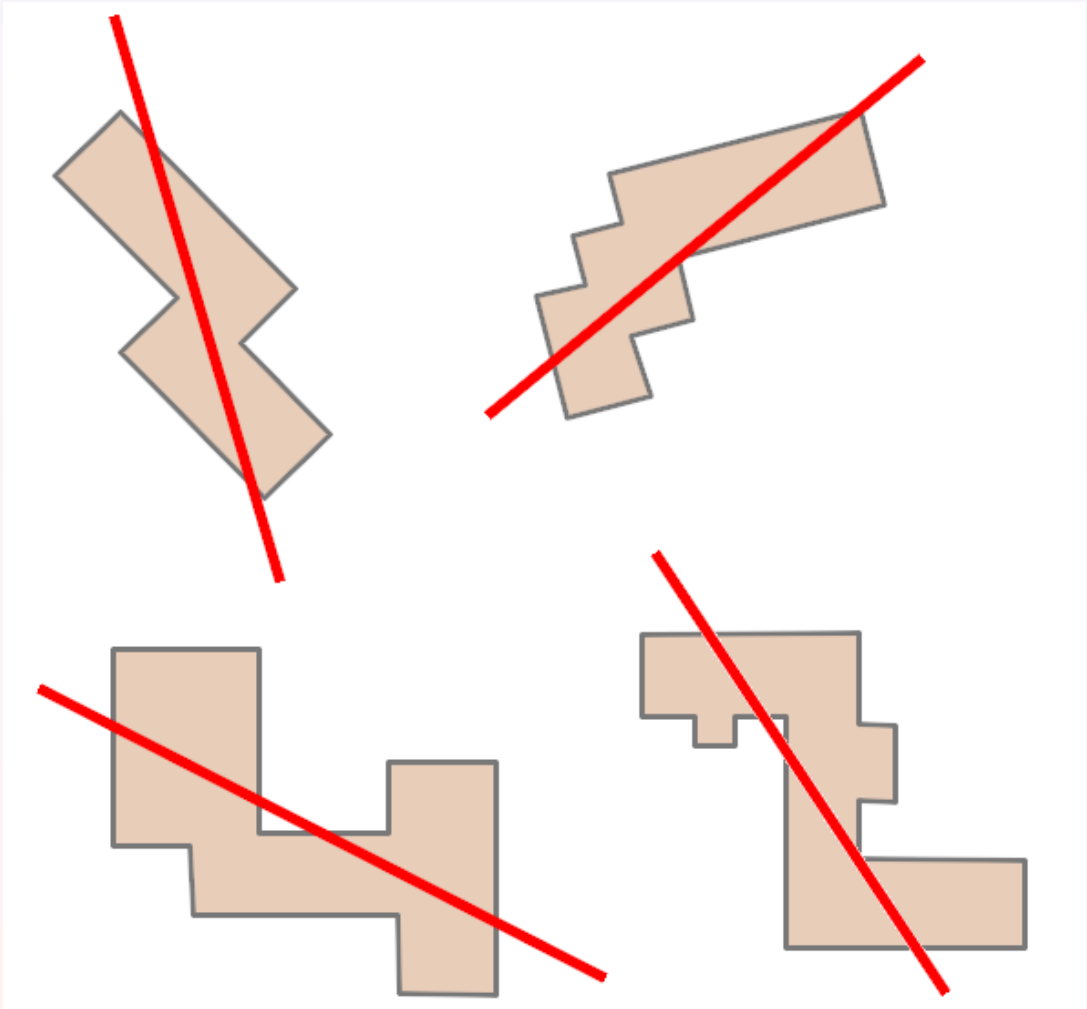
58. Detekce hlavních směrů budovy

Metody Smallest Area Enclosing Rectangle a Weigted Bisector (Duchene et al, 2003).



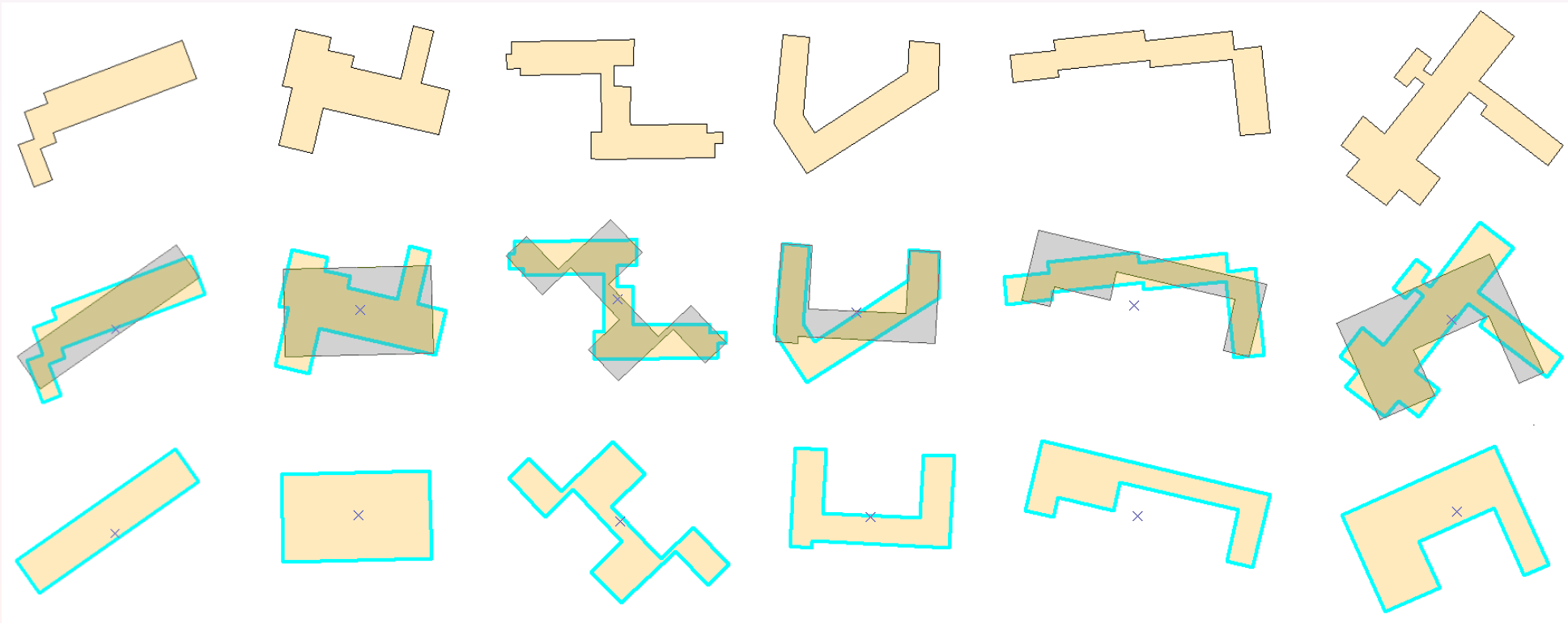
59. Problémy při detekci hlavních směrů

Problémy s detekcí hlavních směrů u budov tvarů L, Z.



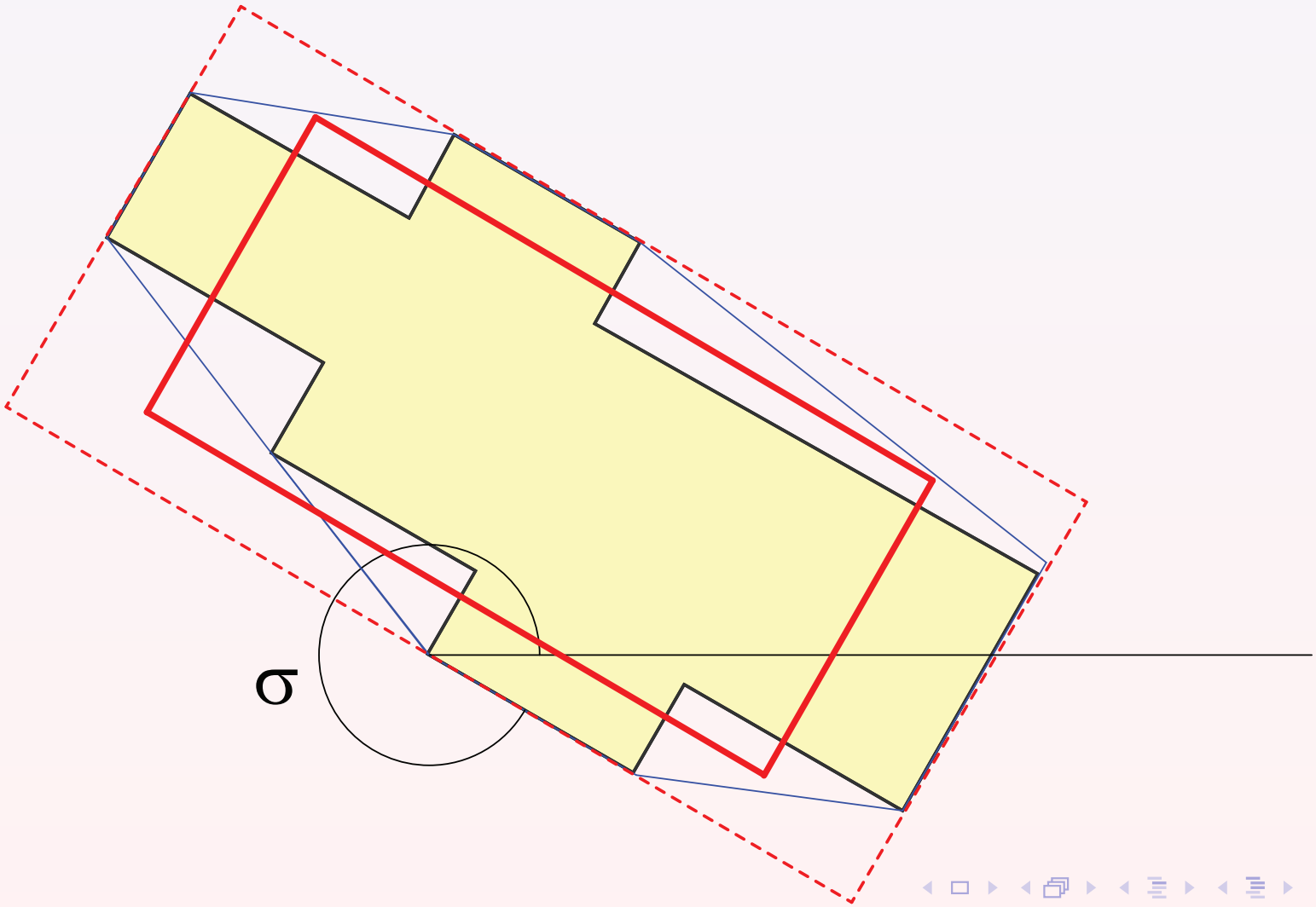
60. Vliv chybného určení hlavního směru na kartografickou generalizaci

Algoritmus Divide and Conquer (Bayer, 2009).



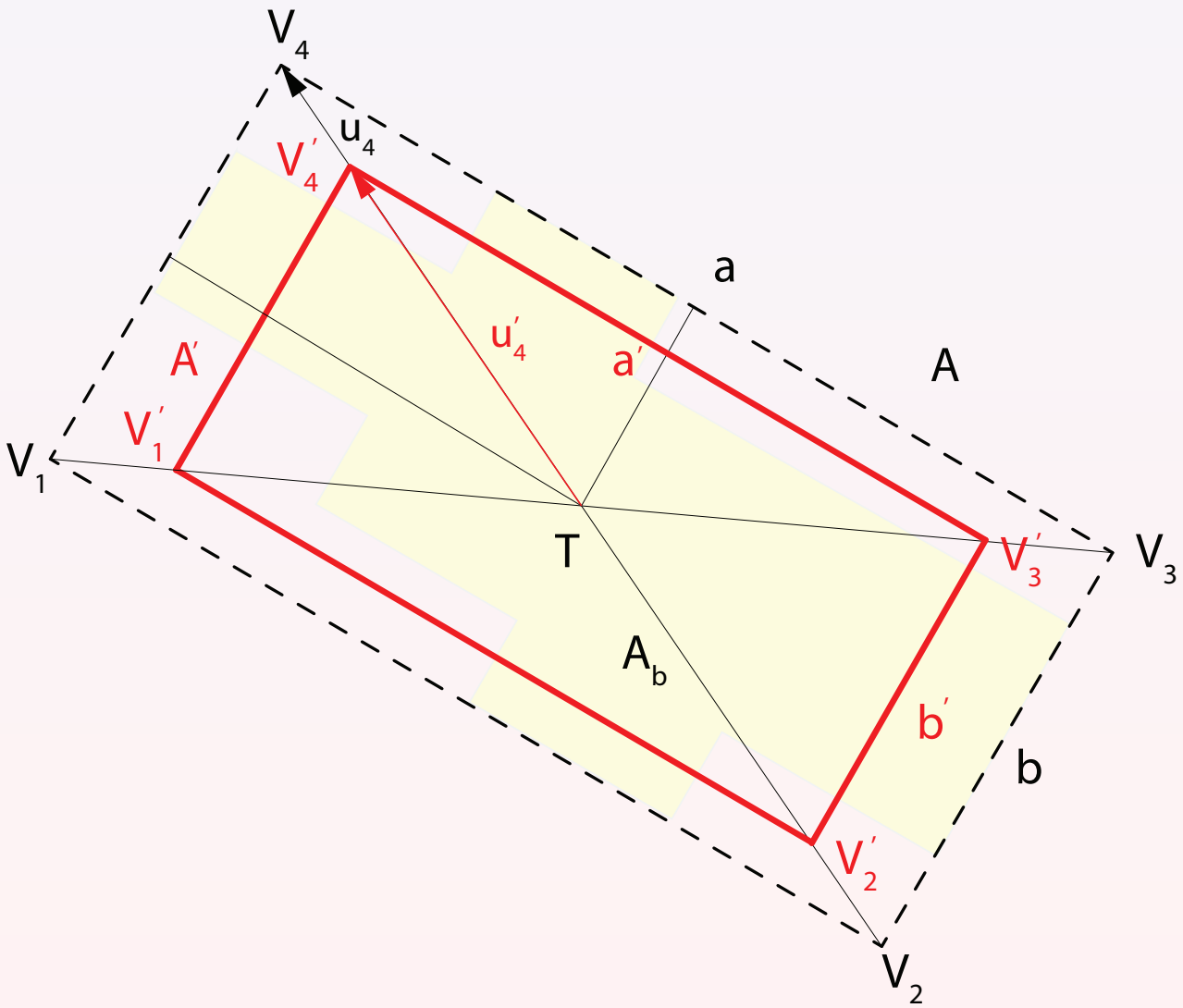
61. Náhrada obdélníkem se stejnou plochou

Často požadujeme, aby obdélník měl stejnou plochu jako generalizovaná budova.
Proporcionální zmenšení, společný střed T.



62. Ukázka

Obdélník $R = (V_1, V_1, V_3, V_4)$, strany a, b, plocha A.
Obdélník $R' = (V_1', V_1', V_3', V_4')$, strany a', b', plocha A'.



63. Výpočet nových vrcholů obdélníku

Plochy obdélníků R, R'

$$A = a \cdot b, \qquad A' = kA = a' \cdot b' = \sqrt{k}a \cdot \sqrt{k}b,$$

kde

$$k = \frac{A_b}{A}.$$

Úhlopříčky obdélníků R, R' určíme z Pyhagorovy věty

$$\left(\frac{\|u_i\|}{2}\right)^2 = \left(\frac{a}{2}\right)^2 + \left(\frac{b}{2}\right)^2, \qquad \left(\frac{\|u_i'\|}{2}\right)^2 = k\left(\frac{a}{2}\right)^2 + k\left(\frac{b}{2}\right)^2 = k\left(\frac{\|u_i\|}{2}\right)^2.$$

Pak

$$\left\|u_i'\right\| = \sqrt{k}\left\|u_i\right\|,$$

kde směrový vektor

$$u_i = V_i - T, \qquad i = 1, \ldots, 4.$$

Nový vrchol V_i' obdélníku

$$V_i' = T + u_i' = T + \sqrt{k}u_i.$$