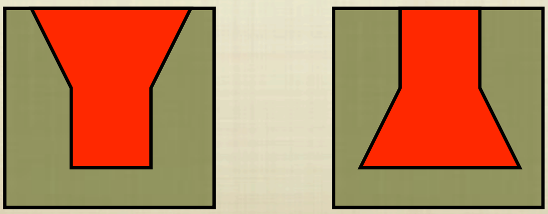
Programação Linear

João Comba / Rodrigo Barni

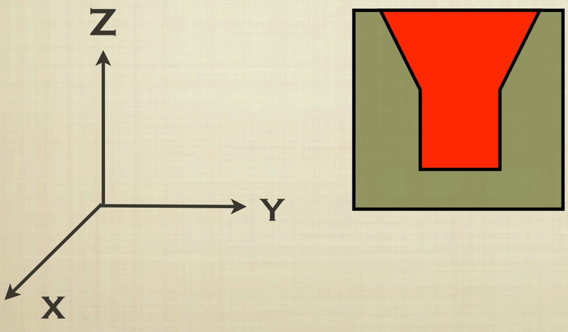
Problema: Moldes

- Como remover uma peça de um molde?
- Restrições:
    - Molde formado por uma única peça
    - Apenas objetos compostos por poliedros
    - Remover realizando apenas uma translação



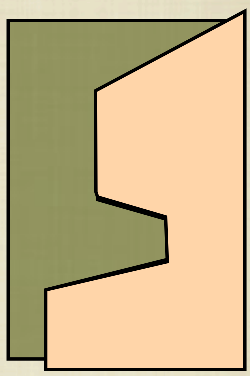
Problema: Moldes

- Molde com uma única peça:
  - Sempre há uma face no topo que não está em contato com o molde
  - Sempre há um componente em z na translação



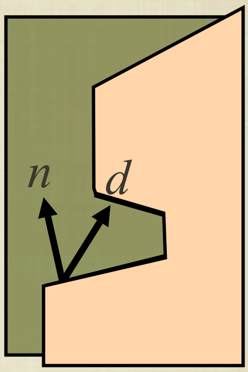
Problema: Moldes

- Verificar se é possível remover o objeto
- Encontrar uma direção viável
- O que é uma direção viável para remoção?



Problema: Moldes

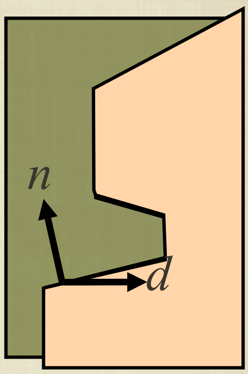
- Verificar se é possível remover o objeto
- Encontrar uma direção viável
- O que é uma direção viável para remoção?
  - Ângulo com a normal de cada face interna do molde deve ser maior que 90 graus



Ângulo entre ***d*** e ***n*** *< 90*
= Direção ***inviável***

Problema: Moldes

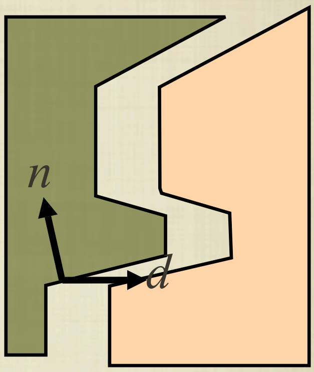
- Verificar se é possível remover o objeto
- Encontrar uma direção viável
- O que é uma direção viável para remoção?
  - Ângulo com a normal de cada face interna do molde deve ser maior que 90 graus



Ângulo entre ***d*** e ***n*** *> 90*
= Direção ***viável***

Problema: Moldes

- Verificar se é possível remover o objeto
- Encontrar uma direção viável
- O que é uma direção viável para remoção?
  - Ângulo com a normal de cada face interna do molde deve ser maior que 90 graus



Ângulo entre ***d*** e ***n*** *> 90* *= Direção* ***viável***

Problema: Moldes

- Translação sempre tem componente positivo em z

$$\boldsymbol{d} = (d_x, d_y, 1)$$

- Ponto no plano $z = 1$

Direção é viável se $\boldsymbol{d} \cdot \boldsymbol{n} \leq 0$

Problema: Moldes

- Translação sempre tem componente positivo em z

$$d = (d_x, d_y, 1)$$

- Ponto no plano $z = 1$

direção é viável se $d \cdot n \leq 0$

$$\Downarrow$$

$$n_x d_x + n_y d_y + n_z \leq 0$$

Problema: Moldes

- Translação sempre tem componente positivo em z

$$d = (d_x, d_y, 1)$$

- Ponto no plano $z = 1$

direção é viável se $d \cdot n \leq 0$

$$\Downarrow$$

$$n_x d_x + n_y d_y + n_z \leq 0$$

semiplano em z=1 !

Intersecção de semiplanos

- Direções viáveis:
  - Intersecção dos semiplanos referentes a todas as faces do objeto
- Conjunto de restrições
  - Uma para cada face, exceto a do topo

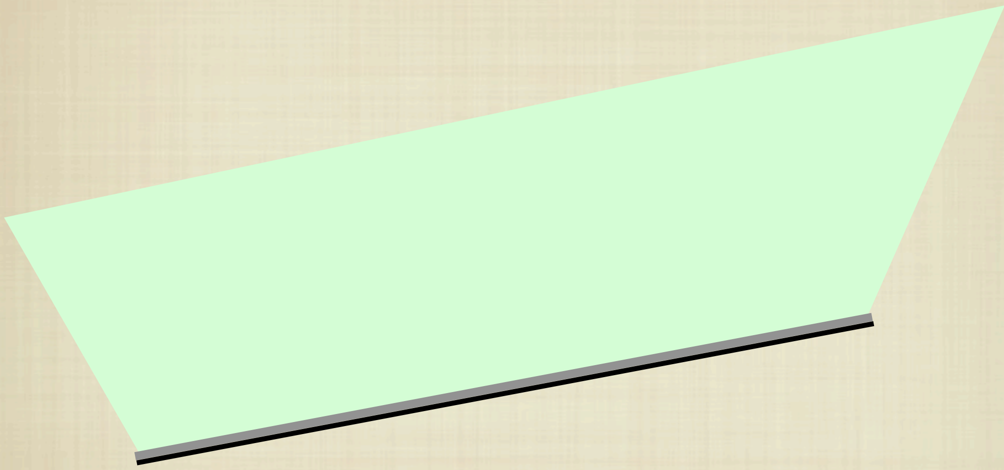
$$a_i x + b_i y \leq c_i \quad \text{p/ i = 1 até i = n-1}$$

Intersecção de semiplanos

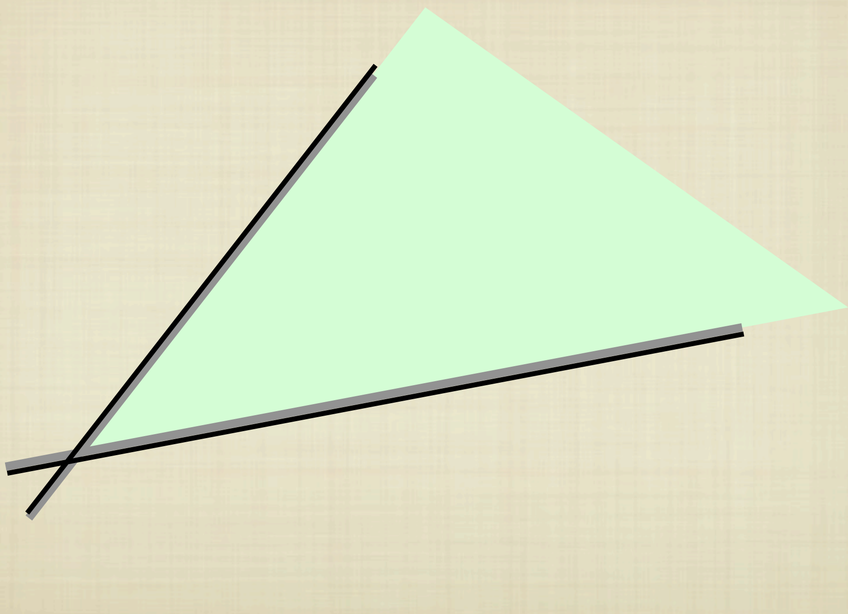
- Direções viáveis:
  - Intersecção dos semiplanos referentes a todas as faces do objeto
- Conjunto de restrições
  - Uma para cada face, exceto a do topo

$$\underbrace{a_i x + b_i y \leq c_i \quad \text{p/ i = 1 até i = n-1}}_{\textbf{Programação Linear}}$$

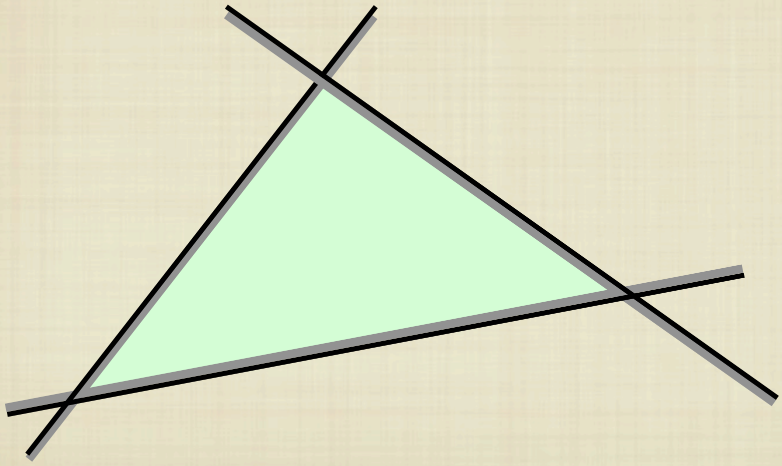
Intersecção de semiplanos



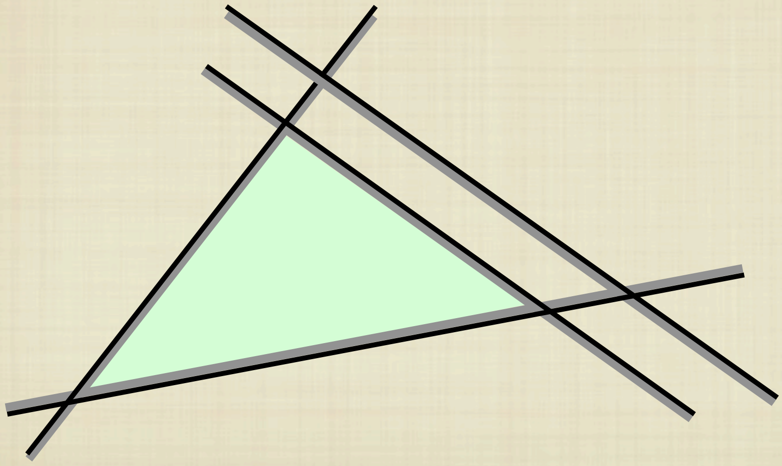
Intersecção de semiplanos



Intersecção de semiplanos



Intersecção de semiplanos



Intersecção de semiplanos

IntersecçãoSemiplanos(H)

Entrada: Conjunto H de semiplanos em um plano

Saída: Região poligonal convexa $C := \bigcap_{h \in H} h$

1. Se card(H) = 1
2. Então $C :=$ semiplano único h em H
3. Senão divide H em H_1 e H_2 de tamanhos [n/2] e [n/2]
4. C_1 := IntersecçãoSemiplanos(H_1)
5. C_2 := IntersecçãoSemiplanos(H_2)
6. C := IntersecçãoRegiõesConvexas(C_1,C_2)

Intersecção de semiplanos

IntersecçãoSemiplanos(H)

Entrada: Conjunto H de semiplanos em um plano

Saída: Região poligonal convexa $C := \bigcap_{h \in H} h$

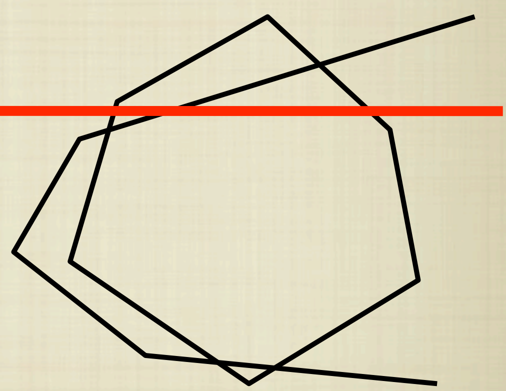
1. Se card(H) = 1
2. Então $C :=$ semiplano único h em H
3. Senão divide H em H_1 e H_2 de tamanhos [n/2] e [n/2]
4. C_1 := IntersecçãoSemiplanos(H_1)
5. C_2 := IntersecçãoSemiplanos(H_2)
6. **C := IntersecçãoRegiõesConvexas(C_1,C_2)**

Intersecção de semiplanos

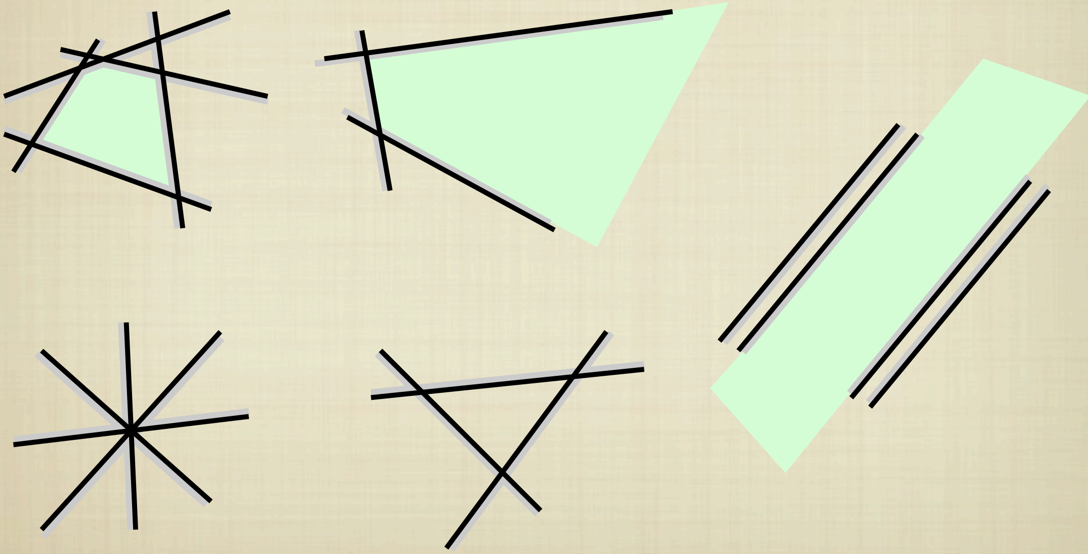
- Semiplanos: sempre convexos
  - Intersecção de regiões convexas: **convexa**
  - Cada semiplano contribui no máximo uma aresta
  - Pode ser uma região aberta
- Como calcular a intersecção?

Intersecção de regiões convexas

- *Sweep* vertical

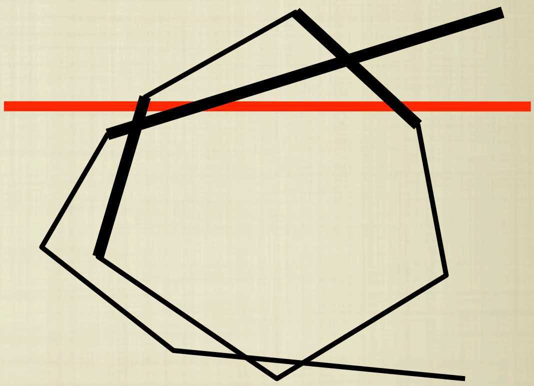


Intersecção de semiplanos



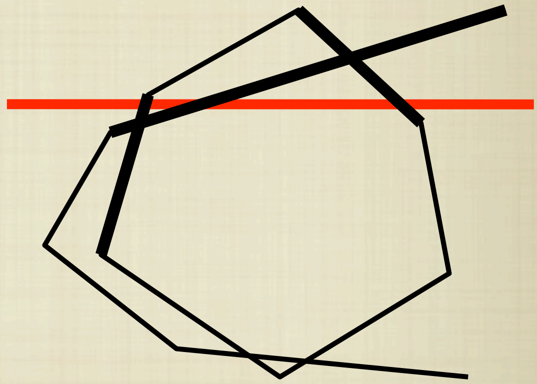
Intersecção de regiões convexas

- *Sweep* vertical
  - Até 4 arestas por vez



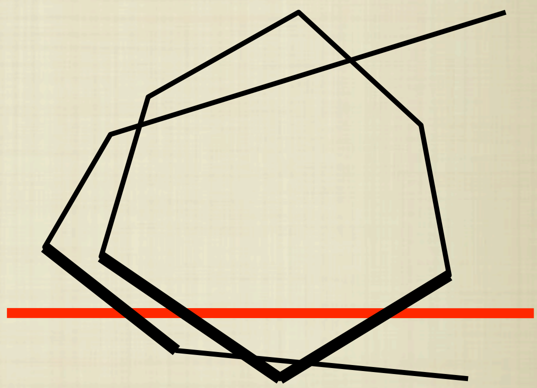
Intersecção de regiões convexas

- *Sweep* vertical
  - Até 4 arestas por vez
    - esquerda_C1
    - direita_C1
    - esquerda_C2
    - direita_C2
  - Se região é aberta, aresta pode ser nula



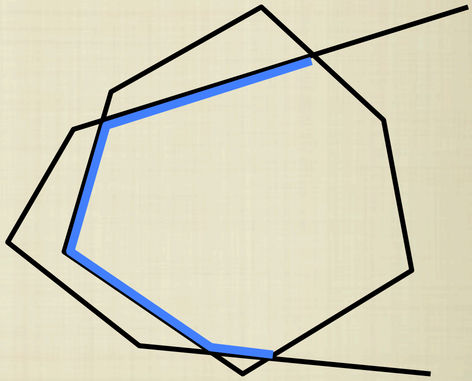
Intersecção de regiões convexas

- *Sweep* vertical
- Até 4 arestas por vez
  - esquerda_C1
  - direita_C1
  - esquerda_C2
  - direita_C2
- Se região é aberta, aresta pode ser nula



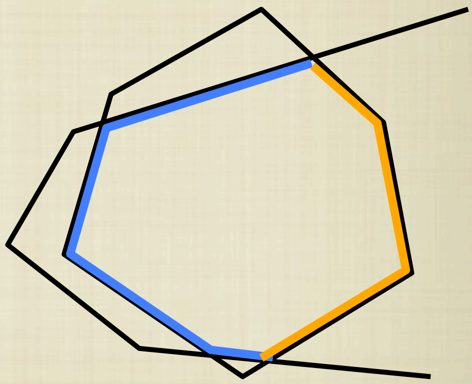
Intersecção de regiões convexas

- *Sweep* vertical
  - Até 4 arestas por vez
- Listas de arestas
  - Borda esquerda



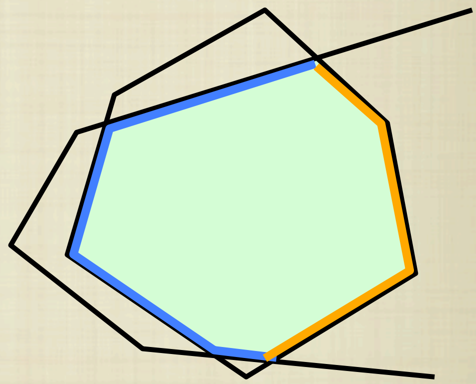
Intersecção de regiões convexas

- *Sweep* vertical
- Até 4 arestas por vez
- Listas de arestas
- Borda esquerda
- Borda Direita



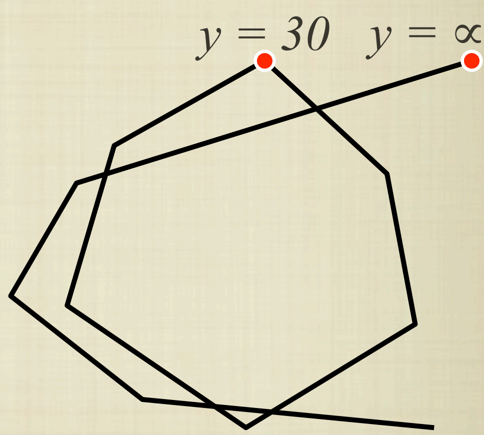
Intersecção de regiões convexas

- *Sweep* vertical
- Até 4 arestas por vez
- Listas de arestas
- Borda esquerda
- Borda Direita



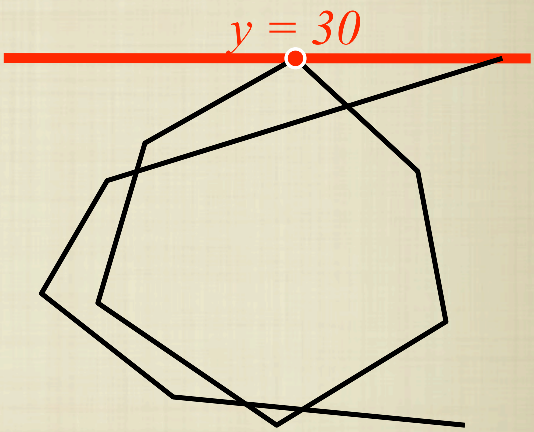
Intersecção de regiões convexas

- Linha de *sweep* começa no mais baixo dos pontos mais altos das regiões convexas



Intersecção de regiões convexas

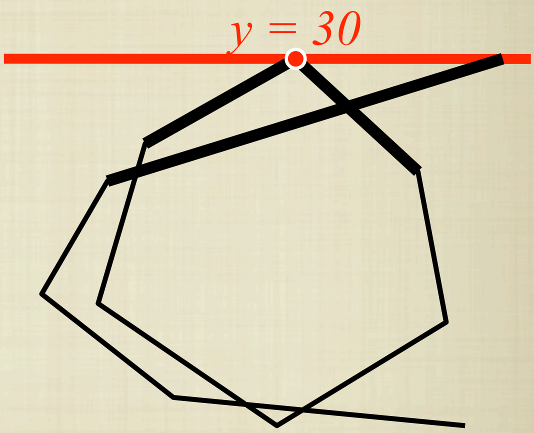
- Linha de *sweep* começa no mais baixo dos pontos mais altos das regiões convexas



Intersecção de regiões convexas

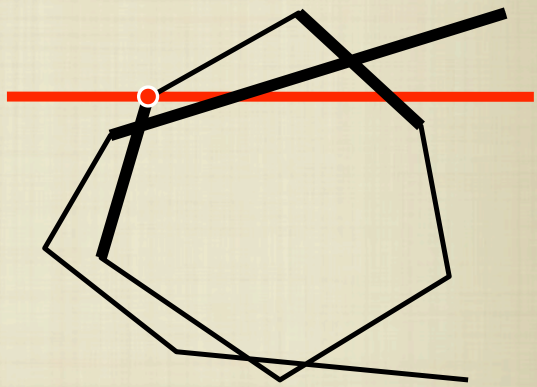
- Linha de *sweep* começa no mais baixo dos pontos mais altos das regiões convexas

- Começa processando arestas que interseccionam a linha na posição inicial



Intersecção de regiões convexas

- Próxima linha: mais alto dos vértices inferiores das arestas atuais
  - Não é necessária lista de eventos

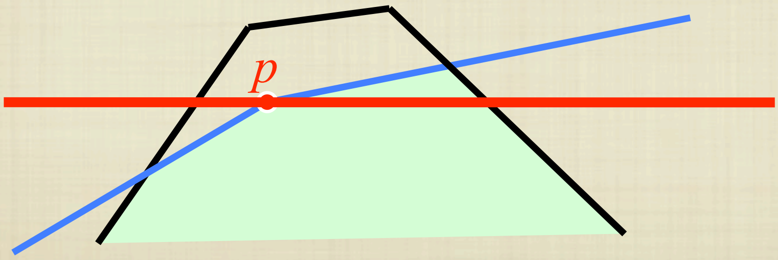


Intersecção de regiões convexas

- Testa o ponto atual (p) da linha de *sweep*
  - e = aresta que ***inicia*** em p
  - C = intersecção entre as duas regiões
- Três casos: (exemplo: p em *esquerda_C1*)
  - p entre *esquerda_C2* e *direita_C2*
  - e intersecciona *direita_C2*
  - e intersecciona *esquerda_C2*
  - Testados em seqüência

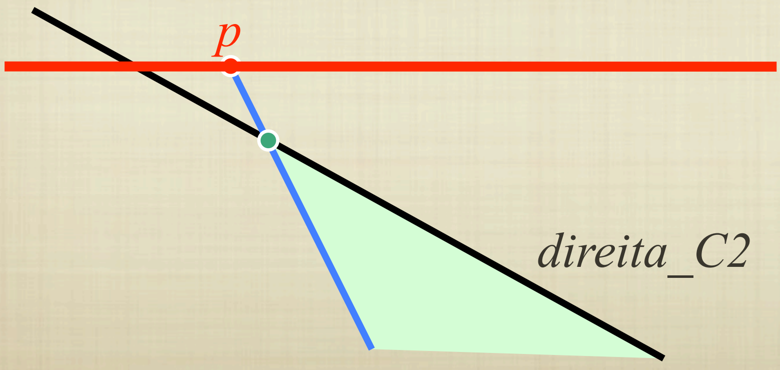
Intersecção de regiões convexas

- Caso 1: p entre $esquerda\_C2$ e $direita\_C2$
  - Aresta e pertence a C, iniciando em p



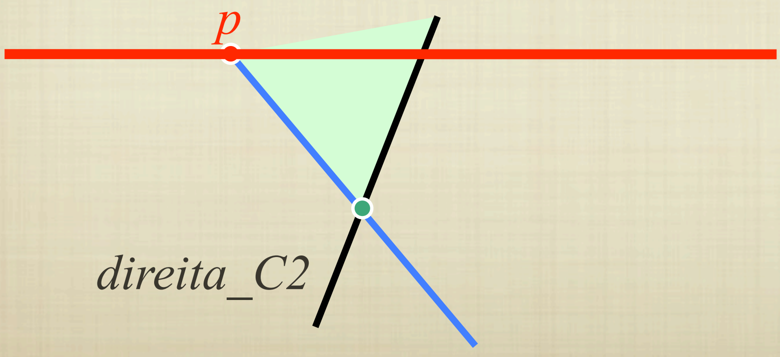
Intersecção de regiões convexas

- Caso 2: e intersecciona $direita\_C2$
  - p à direita de $direita\_C2$
  - e e $direita\_C2$ pertencem a C, iniciando no ponto de intersecção



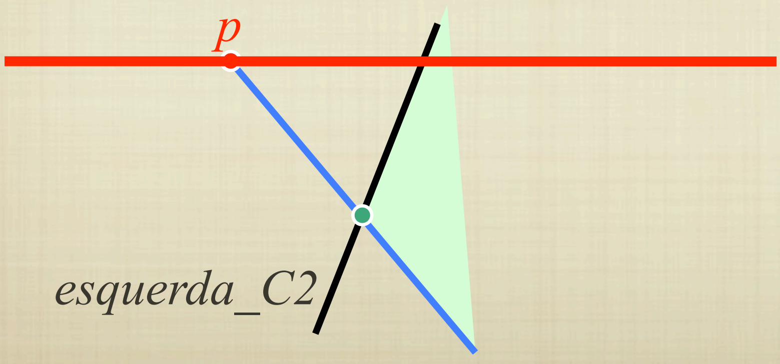
Intersecção de regiões convexas

- Caso 2: *e* intersecciona *direita_C2*
  - *p* à esquerda de *direita_C2*
  - *e* e *direita_C2* pertencem a *C*, terminando no ponto de intersecção



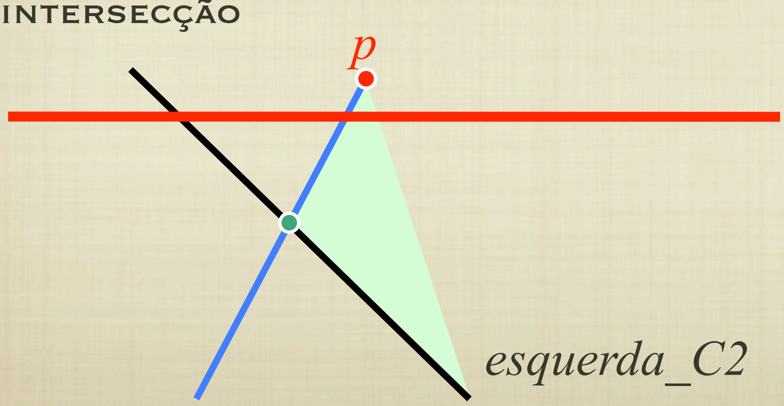
Intersecção de regiões convexas

- Caso 3: *e* intersecciona *esquerda_C2*
  - *p* à esquerda de *esquerda_C2*
  - *e* contribui aresta, começando na intersecção



Intersecção de regiões convexas

- Caso 3: *e* intersecciona *esquerda_C2*
  - p à direita de *esquerda_C2*
  - *esquerda_C2* contribui aresta, começando na intersecção



Intersecção de semiplanos

- Intersecção de regiões convexas: $O(n)$

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(n) + 2T(n/2) & \text{se } n > 1 \end{cases}$$

Intersecção de semiplanos

- Intersecção de regiões convexas: $O(n)$

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(n) + 2T(n/2) & \text{se } n > 1 \end{cases}$$

$$\downarrow$$

$$O(n \log n)$$

Programação linear incremental

- Não é necessária toda a intersecção
  - Uma solução viável basta!

Programação linear incremental

- Não é necessária toda a intersecção
  - Uma solução viável basta!

$$
\begin{array}{ll}
\text{Maximizar} & c_1x_1 + c_2x_2 + \dots + c_dx_d \\
\text{Respeitando} & a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,d}x_d \leq b_1 \\
& a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,d}x_d \leq b_2 \\
& \vdots \\
& a_{n,1}x_1 + a_{n,2}x_2 + \dots + a_{n,d}x_d \leq b_n
\end{array}
$$

Programação linear incremental

- Não é necessária toda a intersecção
  - Uma solução viável basta!

Função objetiva

$$
\begin{array}{ll}
\text{Maximizar} & \boxed{c_1x_1 + c_2x_2 + \dots + c_dx_d} \\
\text{Respeitando} & \boxed{\begin{array}{l} a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,d}x_d \leq b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \dots + a_{2,d}x_d \leq b_2 \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \dots + a_{n,d}x_d \leq b_n \end{array}}
\end{array}
$$

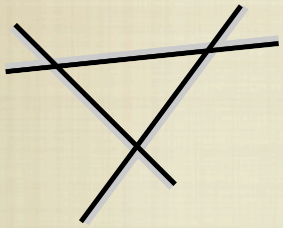
Restrições

d = dimensão (número de variáveis)

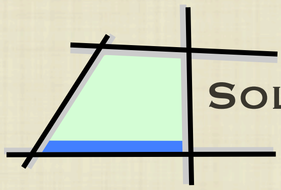
Programação linear incremental

- Tipos de solução:

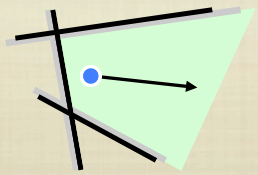
Sem solução



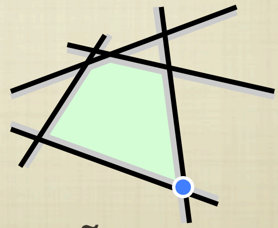
$\vec{c}$



Solução: aresta



Solução: raio

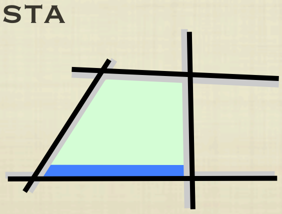


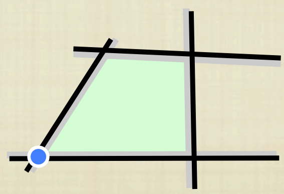
Solução: ponto

Programação linear incremental

- Padronização de solução:
  - Arestas: solução lexicográfica
  - Considerar ponto mais à esquerda da aresta

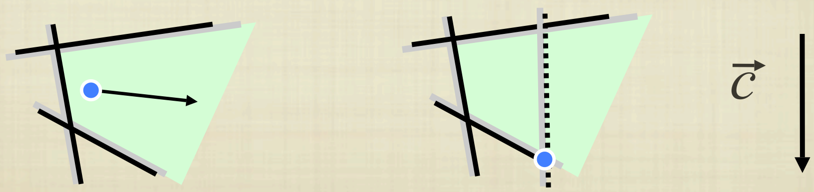
$\vec{c}$





Programação linear incremental

- Padronização de solução:
  - Soluções abertas: limitar a solução em um valor suficientemente grande

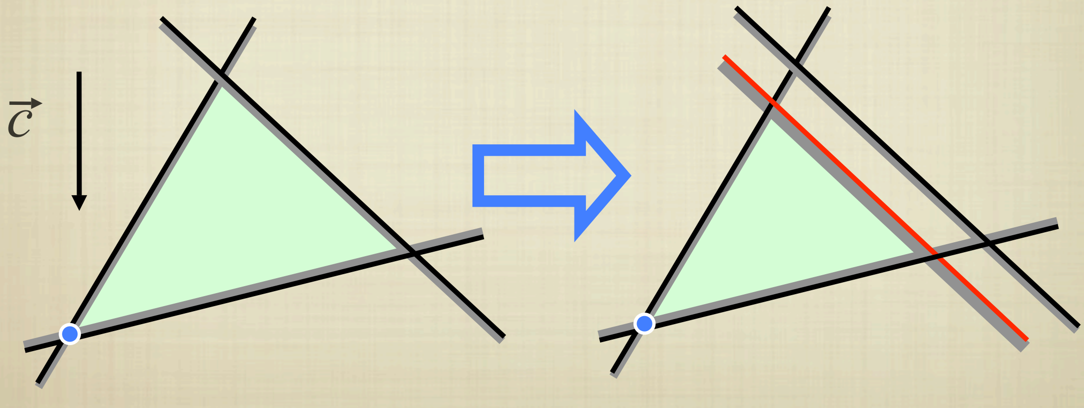


Programação linear incremental

- Considerar primeira restrição
  - Encontrar solução inicial
- Adicionar restrições uma a uma
  - Verificar se solução se mantém
  - Caso negativo, encontrar nova solução

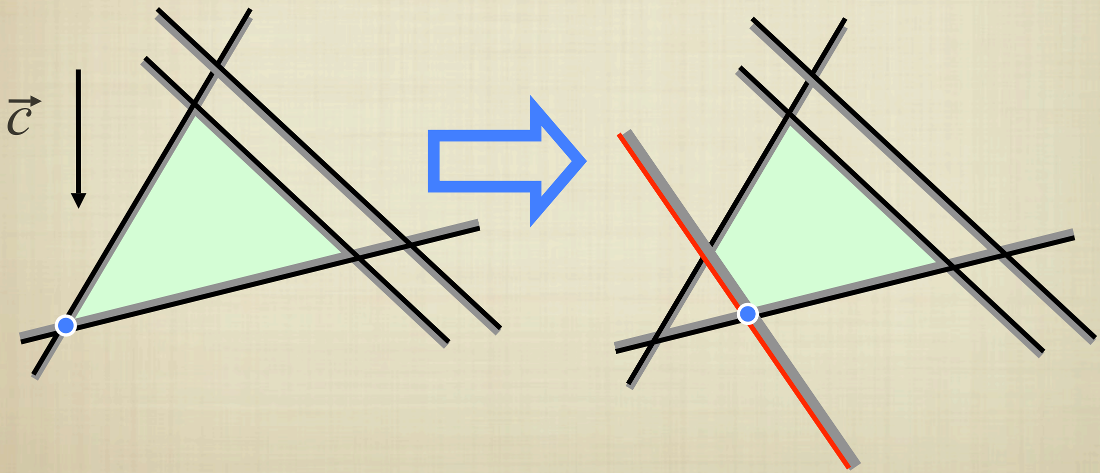
Programação linear incremental

Se a solução atual pertence ao semiplano definido pela
restrição sendo adicionada, ela é mantida



Programação linear incremental

Se a solução atual não pertence ao semiplano definido pela restrição sendo adicionada, uma nova solução deve ser encontrada



Programação linear incremental

- Intersecções de cada etapa devem estar contidas na da etapa anterior
- Se for detectada uma intersecção vazia, não há solução
  - Etapas posteriores não podem modificar isto!

Programação linear incremental

- Solução atual não satisfaz nova restrição
  - Nova solução está na linha que continha a solução da etapa anterior (l_i)
  - Programa linear com dimensão 1
    - Assumindo-se que l_i não é vertical
  - Pode ser encontrada em tempo linear
    - O(i) para uma dada etapa i

Programação linear incremental

ProgramaLinearLimitado(H, c, m_1, m_2)

Entrada: Conjunto H de semiplanos, função objetiva c, restrições limitantes m_1 e m_2

Saída: Menor ponto p que maximiza $f_c(p)$, ou indica que é inviável

```
1. Seja v_0 um canto de C_0
2. Sejam h_1 a hn semiplanos de H
3. Para i de 1 a n
4.  Se v_{i-1} ∈ h_i
5.     Então v_i := v_{i-1}
6.     Senão v_i recebe o ponto em l_i que maximiza f_c(p)
          respeitando as restrições de H
7.     Se p não existe
8.     Então informa que o programa é inviável
9. Retorna v_n
```

Programação linear incremental

- ProgramaLinearLimitado(H, c, m_1, m_2): $O(n^2)$

$$\sum_{i=1}^{N} O(i) = O(n^2)$$

Programação linear randomizada

- Realiza permutação aleatória da seqüência de entrada
  - Modifica ordem de H: $h_1, h_2, \dots h_n$
- Tempo esperado: O(n) – como?
  - O(1) quando é adicionado um semiplano
  - Constante quando solução não é alterada

Programação linear randomizada

- Realiza permutação aleatória da seqüência de entrada
  - Modifica ordem de H: $h_1, h_2, \dots h_n$
- Tempo esperado: O(n) – como?
  - O(1) quando é adicionado um semiplano
  - Constante quando solução não é alterada

    – **Ordem dos semiplanos importa!**

Programação linear randomizada

$$\sum_{i=1}^{n} O(i) \cdot E[X_i]$$

- $E[Xi]$ é a probabilidade de h_i modificar a solução atual
  - Valor esperado da soma de variáveis aleatórias é igual à soma dos valores esperados das variáveis

Programação linear randomizada

$$\sum_{i=1}^{n} O(i) \cdot E[X_i]$$

- $E[Xi]$ é a probabilidade de h_i modificar a solução atual
  - Valor esperado da soma de variáveis aleatórias é igual à soma dos valores esperados das variáveis

- $E[Xi] = 2/i$
  - Análise reversa: solução só muda quando o semiplano adicionado é um dos dois semiplanos que definem a solução atual!

Programação linear randomizada

$$\sum_{i=1}^{n} O(i) \,.\, E[X_i]$$

- $E[Xi]$ é a probabilidade de h_i modificar a solução atual
  - Valor esperado da soma de variáveis aleatórias é igual à soma dos valores esperados das variáveis
- $E[Xi] = 2/i$
  - Análise reversa: solução só muda quando o semiplano adicionado é um dos dois semiplanos que definem a solução atual!

$$\sum_{i=1}^{n} O(i) \,.\, 2/i \quad = \; O(n)$$