Enumeração de *traces*
e
Identificação de *breakpoints*

Estudo de aspectos da evolução

Defesa de Tese de Doutorado

Candidato: Christian Baudet
Orientador: Prof. Dr. Zanoni Dias

Instituto de Computação – Universidade Estadual de Campinas

13 de Dezembro de 2010

Agenda

1. Parte I – Enumeração de traces
   - Conceitos básicos
   - Enumeração de todos os traces de soluções
   - Enumeração parcial de traces de soluções
2. Parte II – Identificação de breakpoints
   - Conceitos básicos
   - Identificação e refinamento de breakpoints
   - Análise de sequências intergênicas

Parte I

Enumeração de *traces*

Permutações

- Cromossomos
  - Sequências de marcadores genômicos
    - ★ Genes
    - ★ Blocos conservados
- Identificadores numéricos
  - Permutação não orientada
  - Permutação orientada
    - ★ Orientação dos marcadores
    - ★ Sinais positivos $(+)$ e negativos $(-)$
- Dados de ortologia
  - Marcadores com ancestral comum
  - Comparação de diferentes genomas
  - Estudo de eventos de rearranjos

Permutações

- Cromossomos
  - Sequências de marcadores genômicos
    - ★ Genes
    - ★ Blocos conservados
- Identificadores numéricos
  - Permutação não orientada
  - Permutação orientada
    - ★ Orientação dos marcadores
    - ★ Sinais positivos $(+)$ e negativos $(-)$
- Dados de ortologia
  - Marcadores com ancestral comum
  - Comparação de diferentes genomas
  - Estudo de eventos de rearranjos

Permutações

- Cromossomos
  - Sequências de marcadores genômicos
    - ★ Genes
    - ★ Blocos conservados
- Identificadores numéricos
  - Permutação não orientada
  - Permutação orientada
    - ★ Orientação dos marcadores
    - ★ Sinais positivos $(+)$ e negativos $(-)$
- Dados de ortologia
  - Marcadores com ancestral comum
  - Comparação de diferentes genomas
  - Estudo de eventos de rearranjos

Ordenação por reversões

- Encontrar sequência ótima de reversões
- Permutações não orientadas
  - Problema NP-Completo
- Permutações orientadas
  - Hannenhalli e Pevzner, 1995 – $O(n^4)$
  - Bergeron, 2001 – $O(n^2)$
  - Tannier, Bergeron e Sagot, 2007 – $O(n^{3/2}\sqrt{\log n})$
  - Swenson *et al.*, 2010 – $O(n \log n + kn)$

Ordenação por reversões

- Encontrar sequência ótima de reversões
- Permutações não orientadas
  - Problema NP-Completo
- Permutações orientadas
  - Hannenhalli e Pevzner, 1995 – $O(n^4)$
  - Bergeron, 2001 – $O(n^2)$
  - Tannier, Bergeron e Sagot, 2007 – $O(n^{3/2}\sqrt{\log n})$
  - Swenson *et al.*, 2010 – $O(n \log n + kn)$

Ordenação por reversões

- Encontrar sequência ótima de reversões
- Permutações não orientadas
  - Problema NP-Completo
- Permutações orientadas
  - Hannenhalli e Pevzner, 1995 – $O(n^4)$
  - Bergeron, 2001 – $O(n^2)$
  - Tannier, Bergeron e Sagot, 2007 – $O(n^{3/2}\sqrt{\log n})$
  - Swenson *et al.*, 2010 – $O(n \log n + kn)$

Ordenação por reversões

- Exemplo:

$$\pi_0 = (-3, +2, +1, -4)$$
$$\pi_1 = (-3, +4, -1, -2)$$
$$\pi_2 = (+3, +4, -1, -2)$$
$$\pi_3 = (+1, -4, -3, -2)$$
$$\pi_4 = (+1, +2, +3, +4)$$

- Algoritmos tradicionais
  - Uma solução ótima
  - $\pi = (-3, +2, +1, -4)$
    - $n = 4$, $d(\pi) = 4$ e 28 soluções
  - $\pi = (-4, -11, +6, -9, -2, +1, -8, +3, -10, +7, -5)$
    - $n = 11$, $d(\pi) = 10$ e 6.345.019 soluções

Ordenação por reversões

- Exemplo:

$$\pi_0 = (-3, +2, +1, -4)$$
$$\pi_1 = (-3, +4, -1, -2)$$
$$\pi_2 = (+3, +4, -1, -2)$$
$$\pi_3 = (+1, -4, -3, -2)$$
$$\pi_4 = (+1, +2, +3, +4)$$

- Algoritmos tradicionais
  - Uma solução ótima
  - $\pi = (-3, +2, +1, -4)$
    - $n = 4$, $d(\pi) = 4$ e 28 soluções
  - $\pi = (-4, -11, +6, -9, -2, +1, -8, +3, -10, +7, -5)$
    - $n = 11$, $d(\pi) = 10$ e 6.345.019 soluções

Ordenação por reversões

- Exemplo:

$$\pi_0 = (-3, +2, +1, -4)$$
$$\pi_1 = (-3, +4, -1, -2)$$
$$\pi_2 = (+3, +4, -1, -2)$$
$$\pi_3 = (+1, -4, -3, -2)$$
$$\pi_4 = (+1, +2, +3, +4)$$

- Algoritmos tradicionais
  - Uma solução ótima
  - $\pi = (-3, +2, +1, -4)$
    - $n = 4$, $d(\pi) = 4$ e 28 soluções
  - $\pi = (-4, -11, +6, -9, -2, +1, -8, +3, -10, +7, -5)$
    - $n = 11$, $d(\pi) = 10$ e 6.345.019 soluções

Ordenação por reversões

- Exemplo:

$$\pi_0 = (-3, +2, +1, -4)$$
$$\pi_1 = (-3, +4, -1, -2)$$
$$\pi_2 = (+3, +4, -1, -2)$$
$$\pi_3 = (+1, -4, -3, -2)$$
$$\pi_4 = (+1, +2, +3, +4)$$

- Algoritmos tradicionais
  - Uma solução ótima
  - $\pi = (-3, +2, +1, -4)$
    - $n = 4$, $d(\pi) = 4$ e 28 soluções
  - $\pi = (-4, -11, +6, -9, -2, +1, -8, +3, -10, +7, -5)$
    - $n = 11$, $d(\pi) = 10$ e 6.345.019 soluções

Ordenação por reversões

- Exemplo:

$$\pi_0 = (-3, +2, +1, -4)$$
$$\pi_1 = (-3, +4, -1, -2)$$
$$\pi_2 = (+3, +4, -1, -2)$$
$$\pi_3 = (+1, -4, -3, -2)$$
$$\pi_4 = (+1, +2, +3, +4)$$

- Algoritmos tradicionais
  - Uma solução ótima
  - $\pi = (-3, +2, +1, -4)$
    - $n = 4$, $d(\pi) = 4$ e 28 soluções
  - $\pi = (-4, -11, +6, -9, -2, +1, -8, +3, -10, +7, -5)$
    - $n = 11$, $d(\pi) = 10$ e 6.345.019 soluções

Ordenação por reversões

- Exemplo:

$$\pi_0 = (-3, +2, +1, -4)$$
$$\pi_1 = (-3, +4, -1, -2)$$
$$\pi_2 = (+3, +4, -1, -2)$$
$$\pi_3 = (+1, -4, -3, -2)$$
$$\pi_4 = (+1, +2, +3, +4)$$

- Algoritmos tradicionais
  - Uma solução ótima
  - $\pi = (-3, +2, +1, -4)$
    - $n = 4$, $d(\pi) = 4$ e 28 soluções
  - $\pi = (-4, -11, +6, -9, -2, +1, -8, +3, -10, +7, -5)$
    - $n = 11$, $d(\pi) = 10$ e 6.345.019 soluções

Ordenação por reversões

- Exemplo:

$$\pi_0 = (-3, +2, +1, -4)$$
$$\pi_1 = (-3, +4, -1, -2)$$
$$\pi_2 = (+3, +4, -1, -2)$$
$$\pi_3 = (+1, -4, -3, -2)$$
$$\pi_4 = (+1, +2, +3, +4)$$

- Algoritmos tradicionais
  - Uma solução ótima
  - $\pi = (-3, +2, +1, -4)$
    - $n = 4$, $d(\pi) = 4$ e 28 soluções
  - $\pi = (-4, -11, +6, -9, -2, +1, -8, +3, -10, +7, -5)$
    - $n = 11$, $d(\pi) = 10$ e 6.345.019 soluções

Enumeração de soluções

- Estudo de cenários evolutivos alternativos
- *Optimal* 1-*sequences*
  - Conjunto de reversões
  - Diminuem $d(\pi)$ em 1 unidade
- Siepel, 2003
  - Conjunto de todas as *optimal* 1-*sequences*
  - Complexidade: $O(n^3)$
- Iteração do algoritmo de Siepel
  - Enumeração de todas as soluções
  - Complexidade: $O(n^{2n+3})$

Enumeração de soluções

- Estudo de cenários evolutivos alternativos
- *Optimal* 1-*sequences*
  - Conjunto de reversões
  - Diminuem $d(\pi)$ em 1 unidade
- Siepel, 2003
  - Conjunto de todas as *optimal* 1-*sequences*
  - Complexidade: $O(n^3)$
- Iteração do algoritmo de Siepel
  - Enumeração de todas as soluções
  - Complexidade: $O(n^{2n+3})$

Enumeração de soluções

- Estudo de cenários evolutivos alternativos
- *Optimal* 1-*sequences*
  - Conjunto de reversões
  - Diminuem $d(\pi)$ em 1 unidade
- Siepel, 2003
  - Conjunto de todas as *optimal* 1-*sequences*
  - Complexidade: $O(n^3)$
- Iteração do algoritmo de Siepel
  - Enumeração de todas as soluções
  - Complexidade: $O(n^{2n+3})$

Enumeração de soluções

- Estudo de cenários evolutivos alternativos
- *Optimal* 1-*sequences*
  - Conjunto de reversões
  - Diminuem $d(\pi)$ em 1 unidade
- Siepel, 2003
  - Conjunto de todas as *optimal* 1-*sequences*
  - Complexidade: $O(n^3)$
- Iteração do algoritmo de Siepel
  - Enumeração de todas as soluções
  - Complexidade: $O(n^{2n+3})$

Traces

- Bergeron *et al.*, 2002
  - Introdução do conceito de *traces*
- Se ρ e θ são reversões que não se sobrepõem
  - $\rho\theta$ e $\theta\rho$ são equivalentes
  - ρ e θ comutam
- Representação mais compacta
  - Um único *trace* pode representar diversas soluções
  - Todo *trace* tem uma única forma normal

Traces

- Bergeron *et al.*, 2002
  - Introdução do conceito de *traces*
- Se ρ e θ são reversões que não se sobrepõem
  - $\rho\theta$ e $\theta\rho$ são equivalentes
  - ρ e θ comutam
- Representação mais compacta
  - Um único *trace* pode representar diversas soluções
  - Todo *trace* tem uma única forma normal

Traces

- Bergeron *et al.*, 2002
  - Introdução do conceito de *traces*
- Se ρ e θ são reversões que não se sobrepõem
  - $\rho\theta$ e $\theta\rho$ são equivalentes
  - ρ e θ comutam
- Representação mais compacta
  - Um único *trace* pode representar diversas soluções
  - Todo *trace* tem uma única forma normal

Forma normal

- Um *trace* T está em sua forma normal se pode ser decomposto em sub-palavras $s = u_1|\ldots|u_m$:
  - todo par de elementos da sub-palavra u_i comuta;
  - para todo elemento ρ de uma sub-palavra u_i $(i > 1)$, existe ao menos um elemento θ da sub-palavra u_{i-1} tal que ρ e θ não comutam;
  - toda sub-palavra u_i é não vazia e seus elementos estão em ordem lexicográfica crescente.

- Exemplo: $\pi = (-3, +2, +1, -4)$
  - $\{1,2,4\}\{3\}|\{1,3,4\}|\{2,3,4\}$
    - ⋆ Tamanho $= 4$, Altura $= 3$ e Soluções $= 4$
  - $\{1\}\{1,2,3\}\{2\}\{4\}$
    - ⋆ Tamanho $= 4$, Altura $= 1$ e Soluções $= 24$

Forma normal

- Um *trace* T está em sua forma normal se pode ser decomposto em sub-palavras $s = u_1|\ldots|u_m$:
  - todo par de elementos da sub-palavra u_i comuta;
  - para todo elemento ρ de uma sub-palavra u_i $(i > 1)$, existe ao menos um elemento θ da sub-palavra u_{i-1} tal que ρ e θ não comutam;
  - toda sub-palavra u_i é não vazia e seus elementos estão em ordem lexicográfica crescente.

- Exemplo: $\pi = (-3, +2, +1, -4)$
  - $\{1,2,4\}\{3\}|\{1,3,4\}|\{2,3,4\}$
    - ★ Tamanho $= 4$, Altura $= 3$ e Soluções $= 4$
  - $\{1\}\{1,2,3\}\{2\}\{4\}$
    - ★ Tamanho $= 4$, Altura $= 1$ e Soluções $= 24$

Enumeração de *traces*

- Braga *et al.*, 2008
  - Combina idéias de Siepel, 2003 e Bergeron *et al.*, 2002.
  - Enumeração de todos *traces* de soluções ótimas
  - Complexidade: $O(Nn^{k_{max}+4})$
  - Contagem do número de soluções

- Restrições biológicas
  - Detecção de intervalos comuns
  - Detecção progressiva de intervalos comuns
  - Simetria ao término de replicação
  - Estratos evolucionários

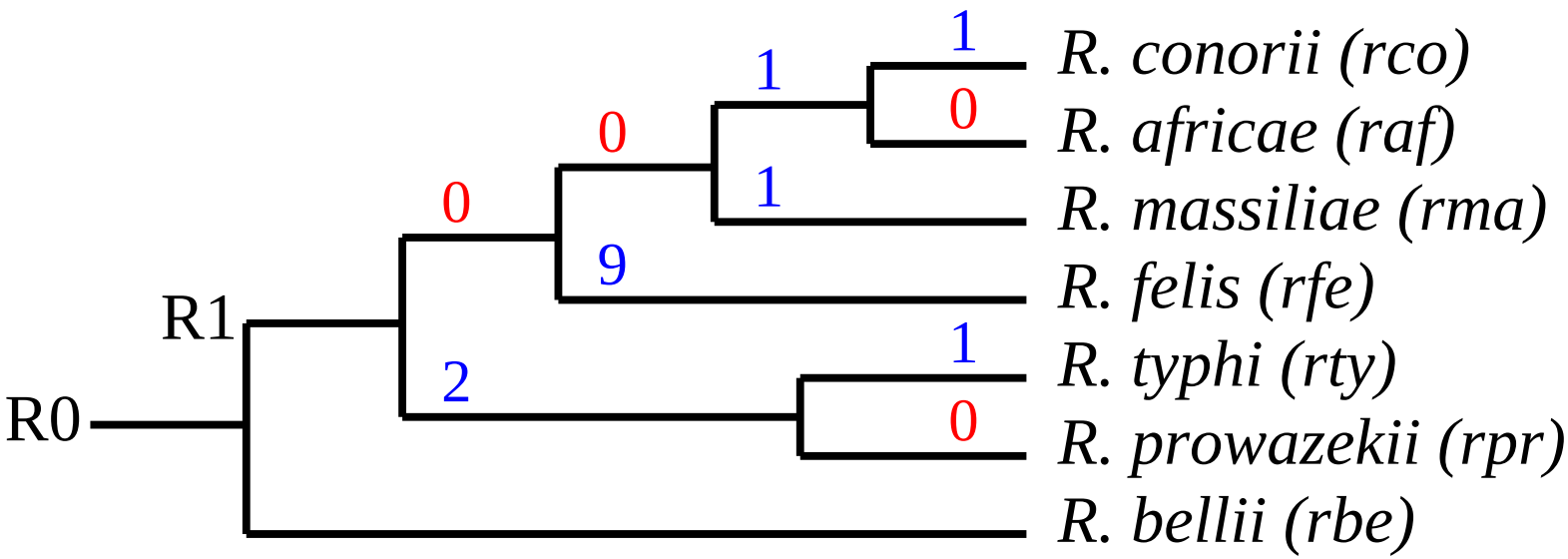
Enumeração de *traces*

- Braga *et al.*, 2008
  - Combina idéias de Siepel, 2003 e Bergeron *et al.*, 2002.
  - Enumeração de todos *traces* de soluções ótimas
  - Complexidade: $O(Nn^{k_{max}+4})$
  - Contagem do número de soluções

- Restrições biológicas
  - Detecção de intervalos comuns
  - Detecção progressiva de intervalos comuns
  - Simetria ao término de replicação
  - Estratos evolucionários

Comparação de grupos de espécies

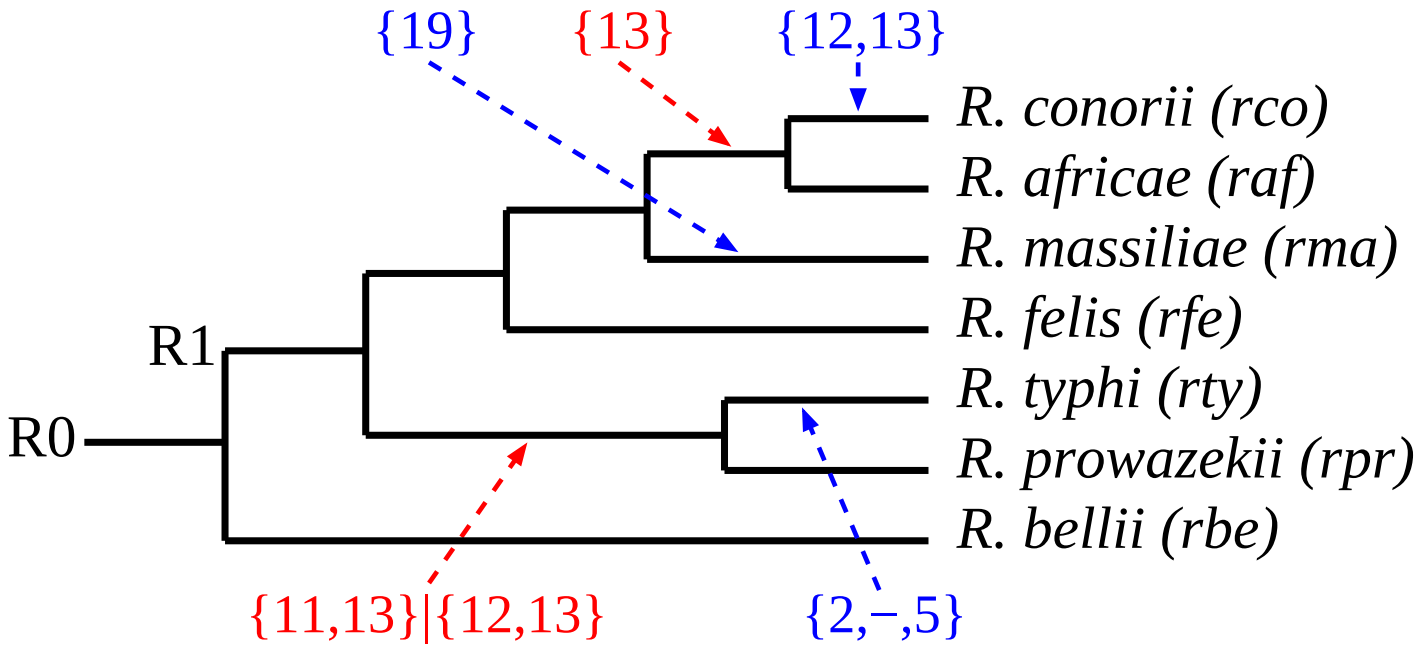
- Gênero *Rickettsia*
  - Grupo de α-proteobactérias intracelulares obrigatórias
  - Blanc *et al.*, 2007 analisaram 7 membros deste gênero



- Permutações construídas com base no ancestral R1
- Enumeração de *traces*

Comparação de grupos de espécies

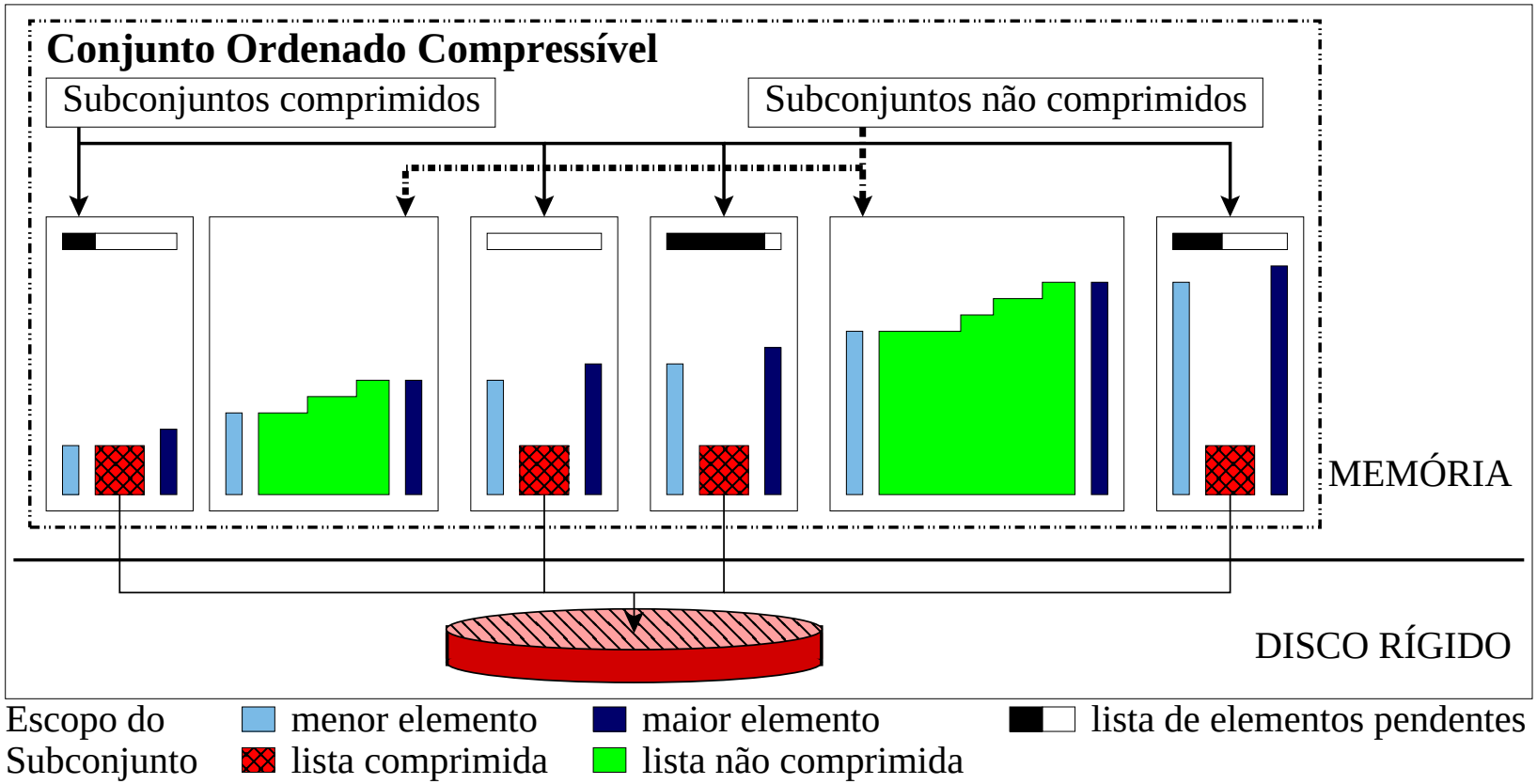
- Análise do conjunto de *traces* e árvore filogenética
  - Inferência da ordem cronológica de algumas reversões



- Trabalho apresentado no ISB 2010 (Calicute, Índia)

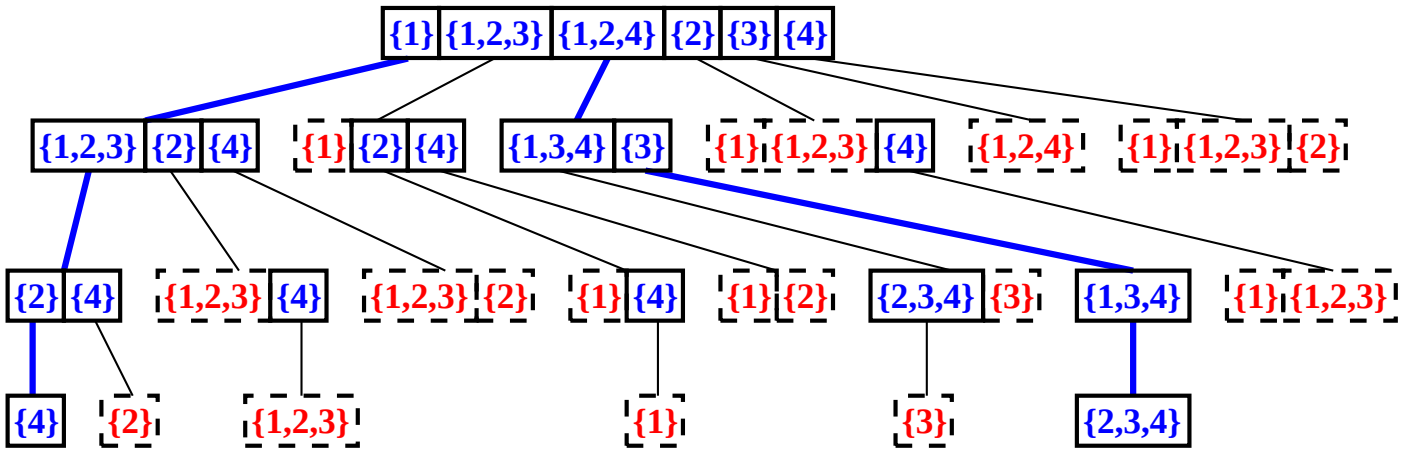
Otimização de desempenho

- Braga *et al.*, 2008
  - Conjunto ordenado compressível



Otimização de desempenho

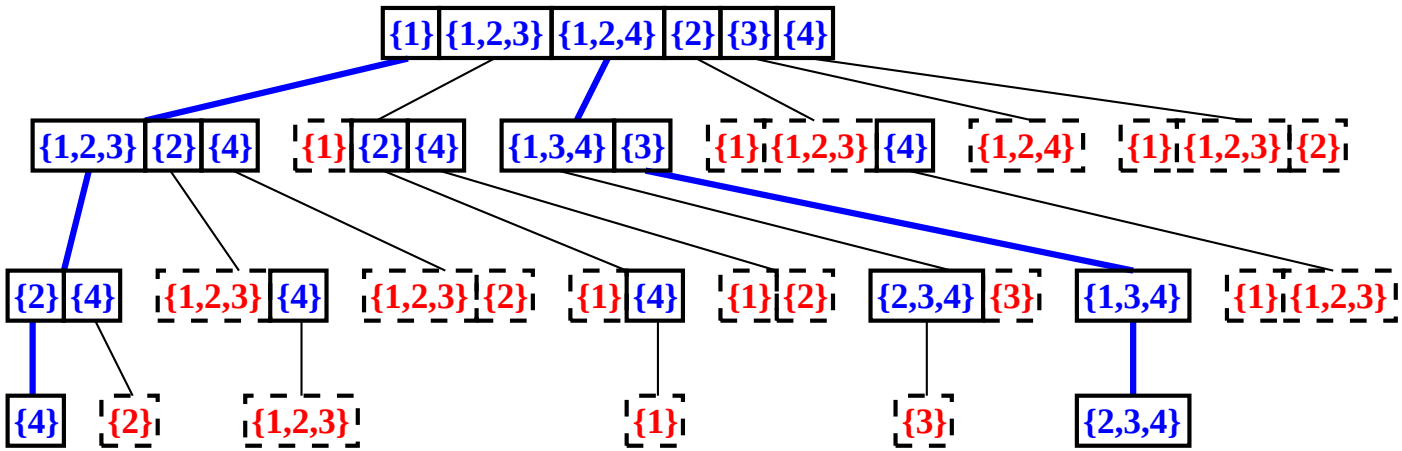
- Conjunto de *traces*
  - Representação em forma de árvore



- Braga *et al.*, 2008
  - Explora a árvore em largura
- Estratégia alternativa
  - Explorar a árvore em profundidade

Otimização de desempenho

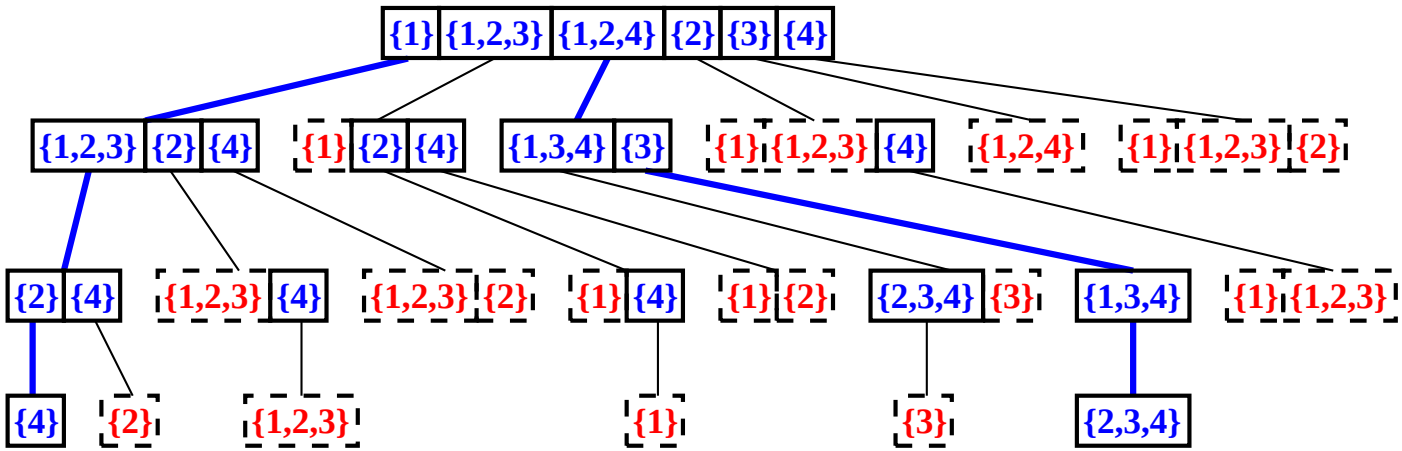
- Conjunto de *traces*
  - Representação em forma de árvore



- Braga *et al.*, 2008
  - Explora a árvore em largura
- Estratégia alternativa
  - Explorar a árvore em profundidade

Otimização de desempenho

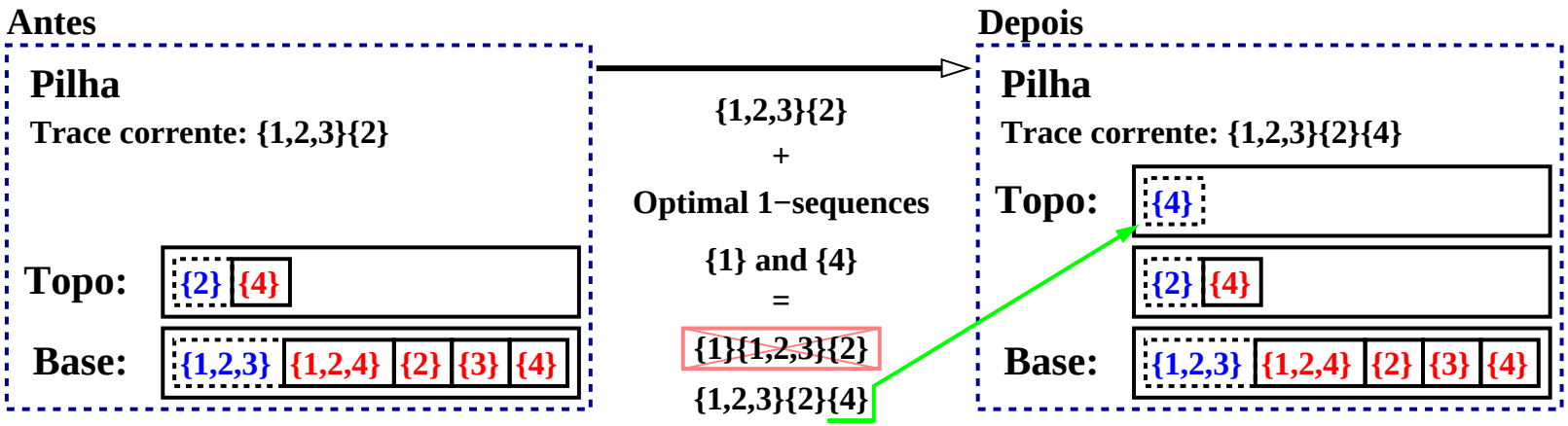
- Conjunto de *traces*
  - Representação em forma de árvore



- Braga *et al.*, 2008
  - Explora a árvore em largura
- Estratégia alternativa
  - Explorar a árvore em profundidade

Otimização de desempenho

- Estrutura de pilha
  - Critério de construção do topo da pilha
    - ★ Evita que todo o espaço de soluções seja explorado
    - ★ Garante enumeração de todos os *traces*



- Um único ramo da árvore é explorado

Otimização de desempenho

- Vantagens
  - Menor consumo de memória
  - Eliminação de acesso a discos
  - Redução do tempo de execução
  - Algoritmo paralelizável
  - Recuperação em caso de interrupção
- Desvantagens
  - Não permite contagem de soluções
  - Limitado a restrições que produzam *traces* perfeitos
- Trabalho apresentado no SAC 2010 (Sierre, Suíça)

Otimização de desempenho

- Vantagens
  - Menor consumo de memória
  - Eliminação de acesso a discos
  - Redução do tempo de execução
  - Algoritmo paralelizável
  - Recuperação em caso de interrupção
- Desvantagens
  - Não permite contagem de soluções
  - Limitado a restrições que produzam *traces* perfeitos
- Trabalho apresentado no SAC 2010 (Sierre, Suíça)

Otimização de desempenho

- Vantagens
  - Menor consumo de memória
  - Eliminação de acesso a discos
  - Redução do tempo de execução
  - Algoritmo paralelizável
  - Recuperação em caso de interrupção
- Desvantagens
  - Não permite contagem de soluções
  - Limitado a restrições que produzam *traces* perfeitos
- Trabalho apresentado no SAC 2010 (Sierre, Suíça)

Trabalhos relacionados

- Swenson, Badr e Sankoff, 2010
  - Algoritmo $O(n^2)$ para listagem de *optimal* 1-*sequences*
- Badr, Swenson e Sankoff, 2010
  - Análise de complexidade do algoritmo de pilha
    - $O(Nn^4 2^n)$ contra $O(Nn^{k_{max}+4})$ do algoritmo original
  - Nova estratégia
    - Agrupamento de *i*-*traces* de acordo com as permutações que eles geram
    - Redução do tempo de execução

Trabalhos relacionados

- Swenson, Badr e Sankoff, 2010
  - Algoritmo $O(n^2)$ para listagem de *optimal* 1-*sequences*
- Badr, Swenson e Sankoff, 2010
  - Análise de complexidade do algoritmo de pilha
    - $O(Nn^4 2^n)$ contra $O(Nn^{k_{max}+4})$ do algoritmo original
  - Nova estratégia
    - Agrupamento de *i*-*traces* de acordo com as permutações que eles geram
    - Redução do tempo de execução

Enumeração parcial de *traces*

- Enumeração total de *traces*
  - Número exponencial de soluções ótimas
  - Viável apenas para pequenas permutações
- Enumeração parcial de *traces*
  - Enumeração de um sub-conjunto de *traces*
  - Processamento de grandes permutações
  - Produção de cenários alternativos

Enumeração parcial de *traces*

- Enumeração total de *traces*
  - Número exponencial de soluções ótimas
  - Viável apenas para pequenas permutações
- Enumeração parcial de *traces*
  - Enumeração de um sub-conjunto de *traces*
  - Processamento de grandes permutações
  - Produção de cenários alternativos

Enumeração parcial de *traces*

- Algoritmos possuem critério de parada
  - Tempo limite de execução
- Proposição de três algoritmos
  1. Algoritmo de pilha limitado por tempo (`Stack`)
  2. Algoritmo aleatório para geração de *traces* (`Random`)
  3. Algoritmo de janela deslizante (`Window`)

Enumeração parcial de *traces*

- Algoritmos possuem critério de parada
  - Tempo limite de execução
- Proposição de três algoritmos
  1. Algoritmo de pilha limitado por tempo (`Stack`)
  2. Algoritmo aleatório para geração de *traces* (`Random`)
  3. Algoritmo de janela deslizante (`Window`)

Enumeração parcial de *traces*

- Algoritmos possuem critério de parada
  - Tempo limite de execução
- Proposição de três algoritmos
  1. Algoritmo de pilha limitado por tempo (`Stack`)
  2. Algoritmo aleatório para geração de *traces* (`Random`)
  3. Algoritmo de janela deslizante (`Window`)

Enumeração parcial de *traces*

- Algoritmos possuem critério de parada
  - Tempo limite de execução
- Proposição de três algoritmos
  1. Algoritmo de pilha limitado por tempo (`Stack`)
  2. Algoritmo aleatório para geração de *traces* (`Random`)
  3. Algoritmo de janela deslizante (`Window`)

Enumeração parcial de *traces*

- Algoritmos possuem critério de parada
  - Tempo limite de execução
- Proposição de três algoritmos
  1. Algoritmo de pilha limitado por tempo (`Stack`)
  2. Algoritmo aleatório para geração de *traces* (`Random`)
  3. Algoritmo de janela deslizante (`Window`)

Algoritmo Random

- Geração aleatória de *traces* entre π_0 e π_d
- A partir de π_i e do *i-trace* t associado
  1. Produção da lista L de *optimal* 1-*sequences* de π_i
  2. Seleção aleatória de uma reversão ρ de L
  3. Criação do $(i+1)$-*trace* $t' \leftarrow t + \rho$
  4. Geração de $\pi_{i+1} \leftarrow \pi_i \circ \rho$

Algoritmo Random

- Geração aleatória de *traces* entre π_0 e π_d
- A partir de π_i e do *i-trace t* associado
  1. Produção da lista L de *optimal* 1-*sequences* de π_i
  2. Seleção aleatória de uma reversão ρ de L
  3. Criação do $(i+1)$-*trace* $t' \leftarrow t + \rho$
  4. Geração de $\pi_{i+1} \leftarrow \pi_i \circ \rho$

Algoritmo Random

- Geração aleatória de *traces* entre π_0 e π_d
- A partir de π_i e do *i-trace* t associado
  1. Produção da lista L de *optimal* 1-*sequences* de π_i
  2. Seleção aleatória de uma reversão ρ de L
  3. Criação do $(i+1)$-*trace* $t' \leftarrow t + \rho$
  4. Geração de $\pi_{i+1} \leftarrow \pi_i \circ \rho$

Algoritmo Random

- Geração aleatória de *traces* entre π_0 e π_d
- A partir de π_i e do *i-trace t* associado
  1. Produção da lista L de *optimal* 1-*sequences* de π_i
  2. Seleção aleatória de uma reversão ρ de L
  3. Criação do $(i+1)$-*trace* $t' \leftarrow t + \rho$
  4. Geração de $\pi_{i+1} \leftarrow \pi_i \circ \rho$

Algoritmo Random

- Geração aleatória de *traces* entre π_0 e π_d
- A partir de π_i e do *i-trace* t associado
  1. Produção da lista L de *optimal* 1-*sequences* de π_i
  2. Seleção aleatória de uma reversão ρ de L
  3. Criação do $(i+1)$-*trace* $t' \leftarrow t + \rho$
  4. Geração de $\pi_{i+1} \leftarrow \pi_i \circ \rho$

Algoritmo Random

- Geração aleatória de *traces* entre π_0 e π_d
- A partir de π_i e do *i-trace* t associado
  1. Produção da lista L de *optimal* 1-*sequences* de π_i
  2. Seleção aleatória de uma reversão ρ de L
  3. Criação do $(i+1)$-*trace* $t' \leftarrow t + \rho$
  4. Geração de $\pi_{i+1} \leftarrow \pi_i \circ \rho$

Algoritmo `Window`

- Processamento de um "caminho" aleatório
  - Permutações $\pi_0, \pi_1, \pi_2, \ldots, \pi_{d-1}, \pi_d$
  - Algoritmo `Random` modificado
- Algoritmo de enumeração total
  - *w-traces* que transformam π_i em π_{i+w}
- Combinação de *traces*
  - *i-traces* que transformam π_0 em π_i
  - *w-traces* que transformam π_i em π_{i+w}
  - $(i + w)$-*traces* que transformam π_0 em π_{i+w}

Algoritmo `Window`

- Processamento de um "caminho" aleatório
  - Permutações $\pi_0, \pi_1, \pi_2, \ldots, \pi_{d-1}, \pi_d$
  - Algoritmo `Random` modificado
- Algoritmo de enumeração total
  - *w-traces* que transformam π_i em π_{i+w}
- Combinação de *traces*
  - *i-traces* que transformam π_0 em π_i
  - *w-traces* que transformam π_i em π_{i+w}
  - $(i + w)$-*traces* que transformam π_0 em π_{i+w}

Algoritmo `Window`

- Processamento de um "caminho" aleatório
  - Permutações $\pi_0, \pi_1, \pi_2, \ldots, \pi_{d-1}, \pi_d$
  - Algoritmo `Random` modificado
- Algoritmo de enumeração total
  - *w-traces* que transformam π_i em π_{i+w}
- Combinação de *traces*
  - *i-traces* que transformam π_0 em π_i
  - *w-traces* que transformam π_i em π_{i+w}
  - $(i + w)$-*traces* que transformam π_0 em π_{i+w}

Algoritmo `Window`

- Processamento de um "caminho" aleatório
  - Permutações $\pi_0, \pi_1, \pi_2, \ldots, \pi_{d-1}, \pi_d$
  - Algoritmo `Random` modificado
- Algoritmo de enumeração total
  - *w-traces* que transformam π_i em π_{i+w}
- Combinação de *traces*
  - *i-traces* que transformam π_0 em π_i
  - *w-traces* que transformam π_i em π_{i+w}
  - $(i + w)$-*traces* que transformam π_0 em π_{i+w}

Algoritmo `Window`

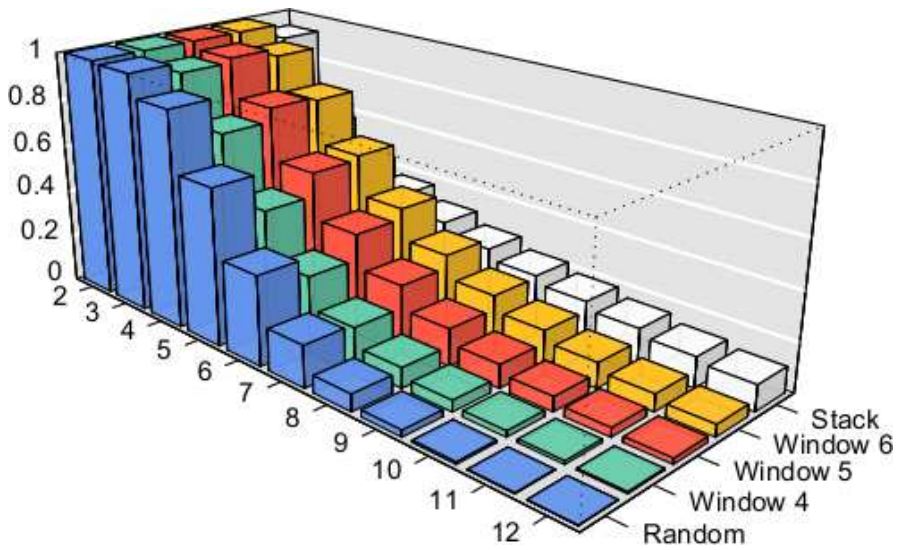
- Processamento de um "caminho" aleatório
  - Permutações $\pi_0, \pi_1, \pi_2, \ldots, \pi_{d-1}, \pi_d$
  - Algoritmo `Random` modificado
- Algoritmo de enumeração total
  - *w-traces* que transformam π_i em π_{i+w}
- Combinação de *traces*
  - *i-traces* que transformam π_0 em π_i
  - *w-traces* que transformam π_i em π_{i+w}
  - $(i + w)$-*traces* que transformam π_0 em π_{i+w}

Algoritmo `Window`

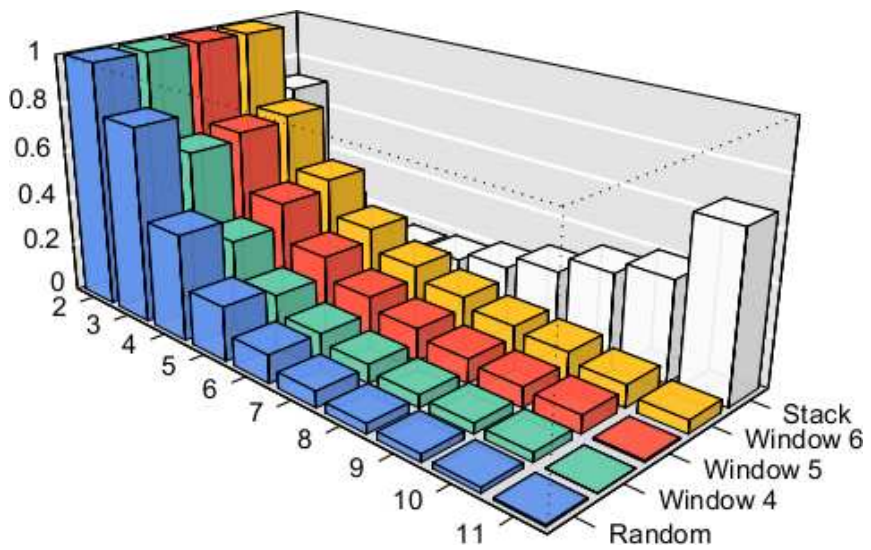
- Processamento de um "caminho" aleatório
  - Permutações $\pi_0, \pi_1, \pi_2, \ldots, \pi_{d-1}, \pi_d$
  - Algoritmo `Random` modificado
- Algoritmo de enumeração total
  - *w-traces* que transformam π_i em π_{i+w}
- Combinação de *traces*
  - *i-traces* que transformam π_0 em π_i
  - *w-traces* que transformam π_i em π_{i+w}
  - $(i + w)$-*traces* que transformam π_0 em π_{i+w}

Comparação dos algoritmos

- Primeiro teste – 6 segundos



(a) Altura média do trace

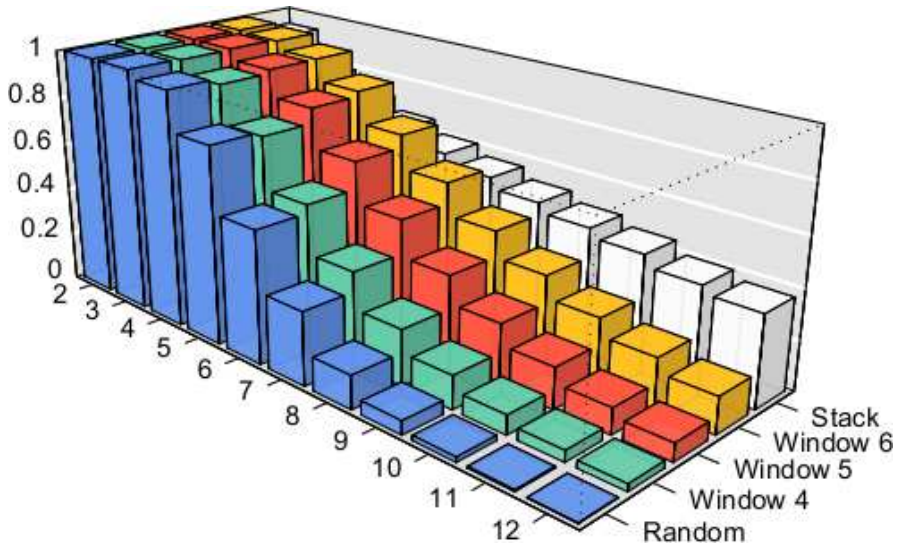


(b) Tamanho médio da reversão

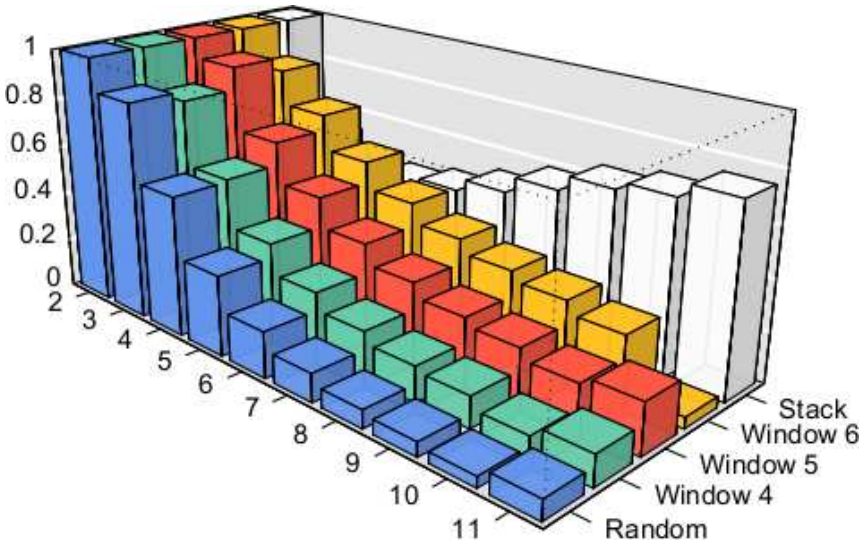
500 permutações aleatórias $n = 15$ e $d(\pi) = 12$

Comparação dos algoritmos

- Primeiro teste – 12 segundos



(a) Altura média do trace

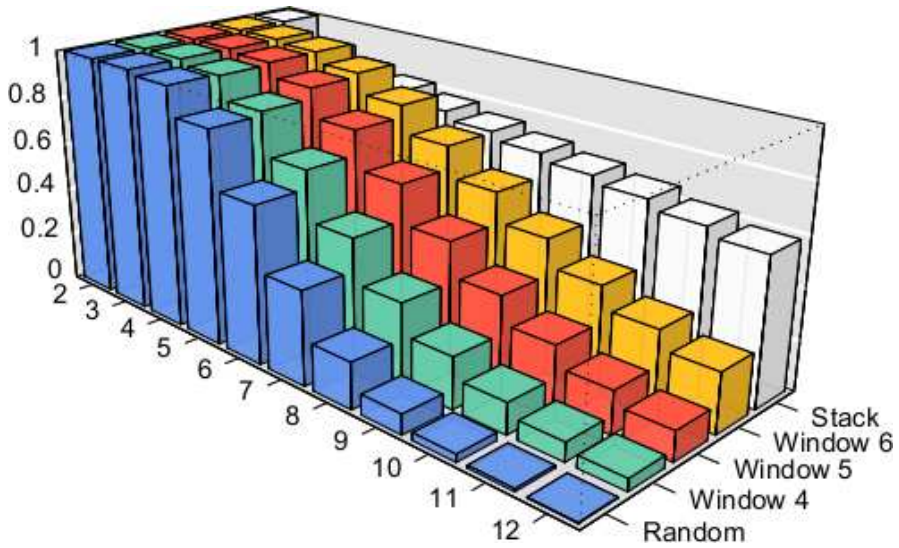


(b) Tamanho médio da reversão

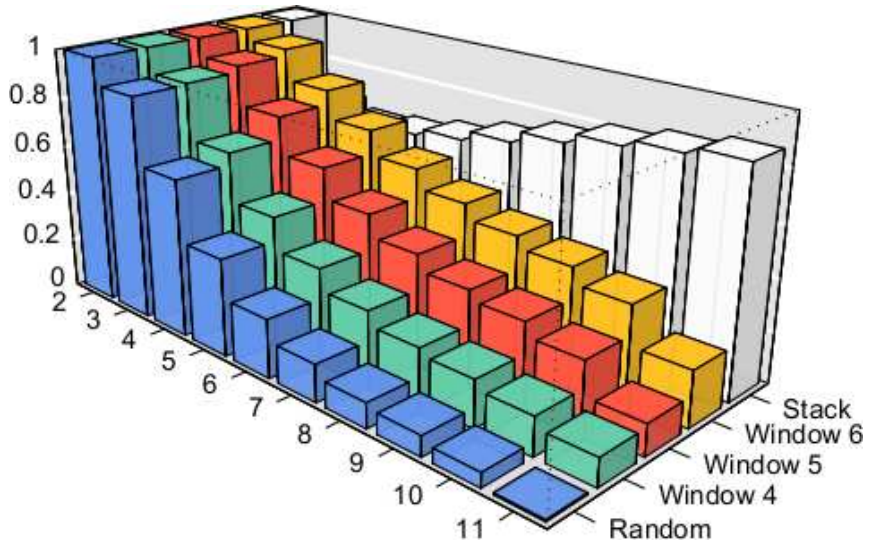
500 permutações aleatórias $n = 15$ e $d(\pi) = 12$

Comparação dos algoritmos

- Primeiro teste – 18 segundos



(a) Altura média do trace

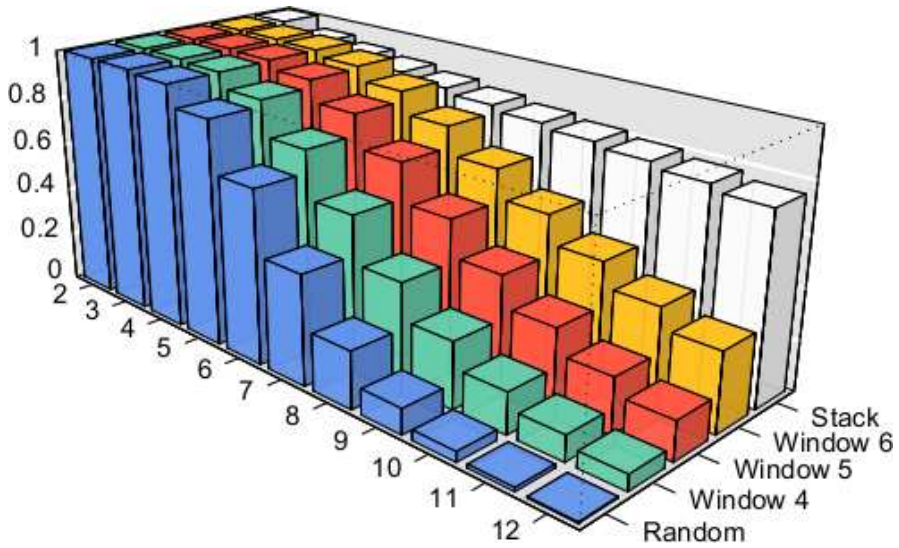


(b) Tamanho médio da reversão

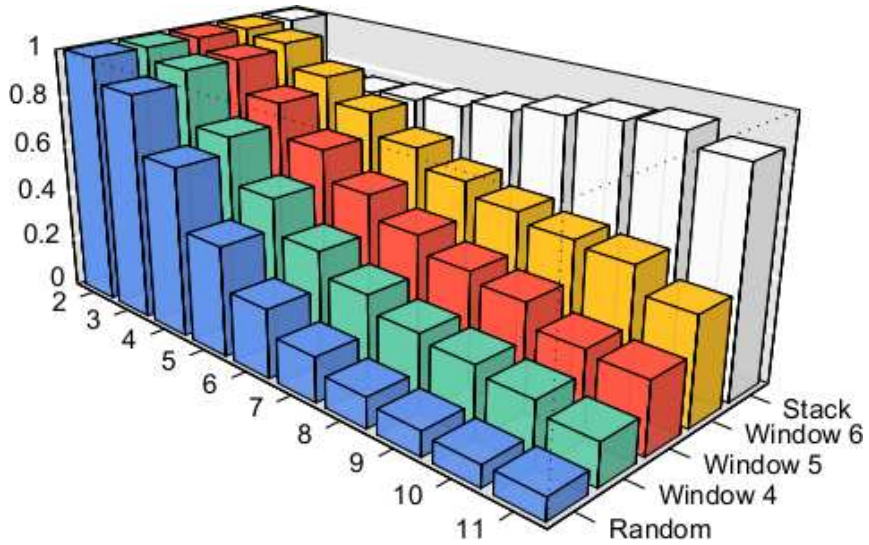
500 permutações aleatórias $n = 15$ e $d(\pi) = 12$

Comparação dos algoritmos

- Primeiro teste – 24 segundos



(a) Altura média do trace

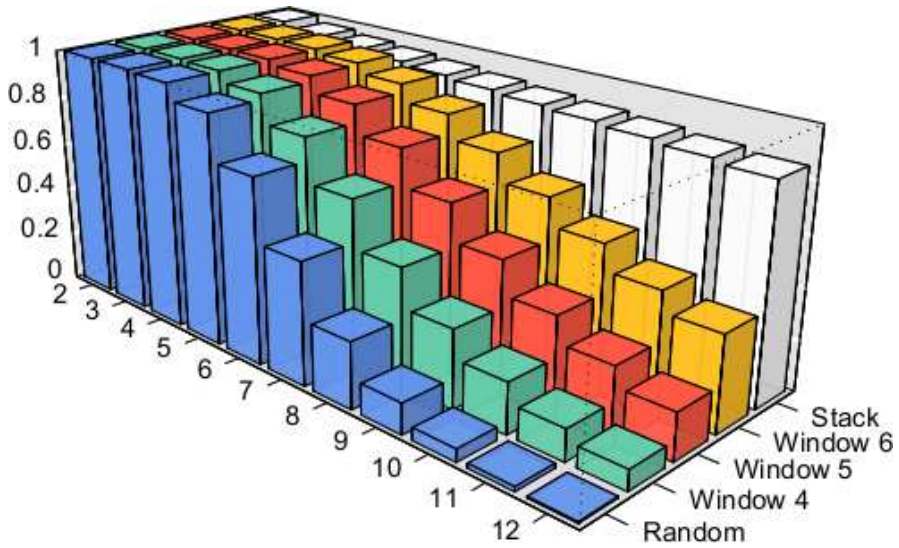


(b) Tamanho médio da reversão

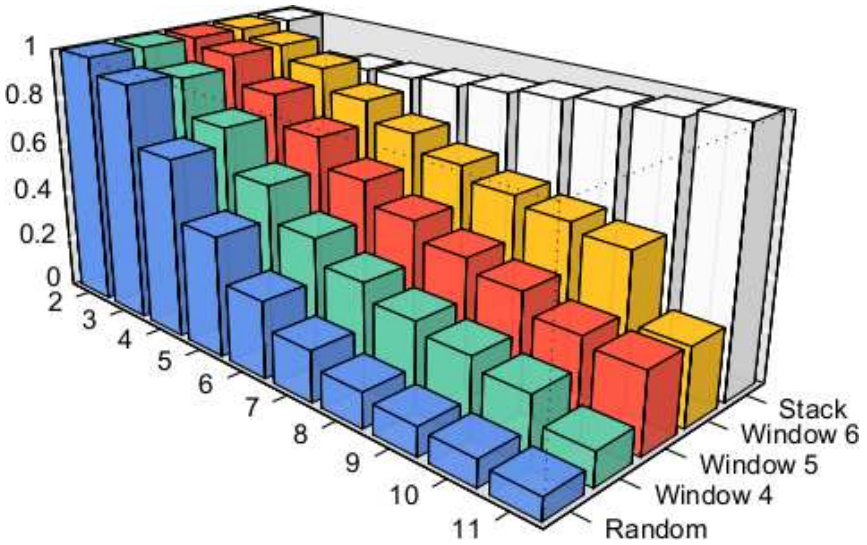
500 permutações aleatórias $n = 15$ e $d(\pi) = 12$

Comparação dos algoritmos

- Primeiro teste – 30 segundos



(a) Altura média do trace

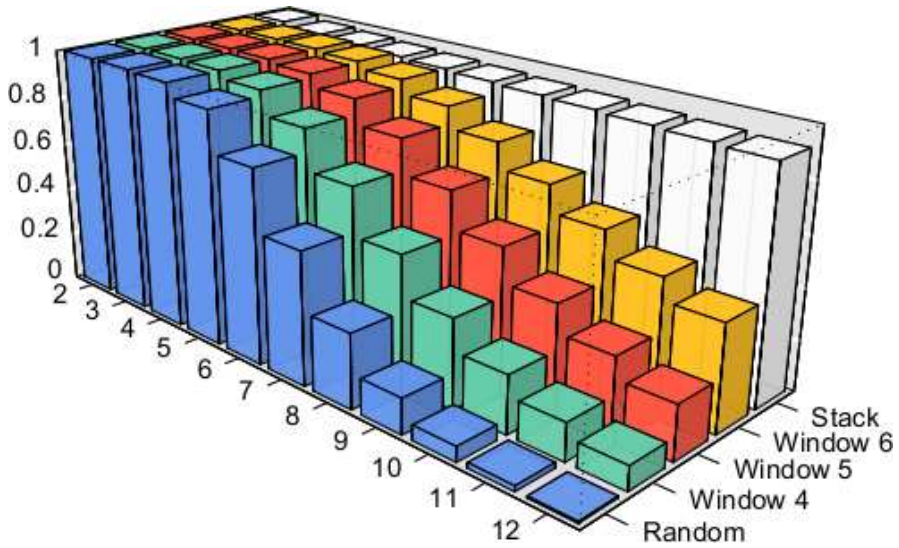


(b) Tamanho médio da reversão

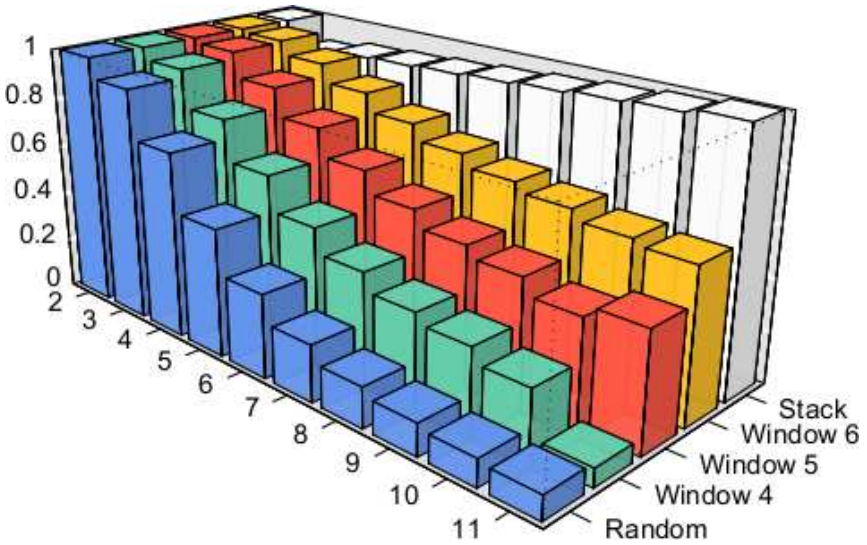
500 permutações aleatórias $n = 15$ e $d(\pi) = 12$

Comparação dos algoritmos

- Primeiro teste – 36 segundos



(a) Altura média do trace

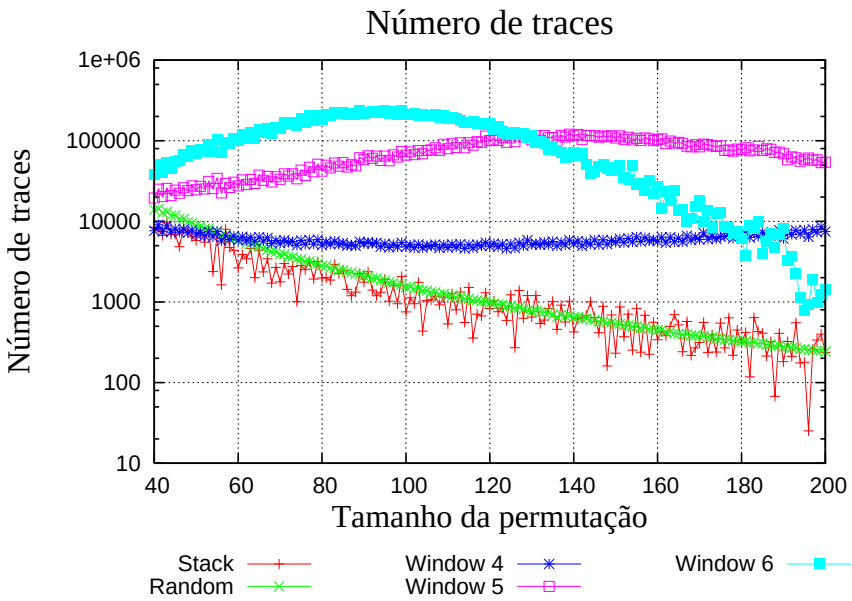


(b) Tamanho médio da reversão

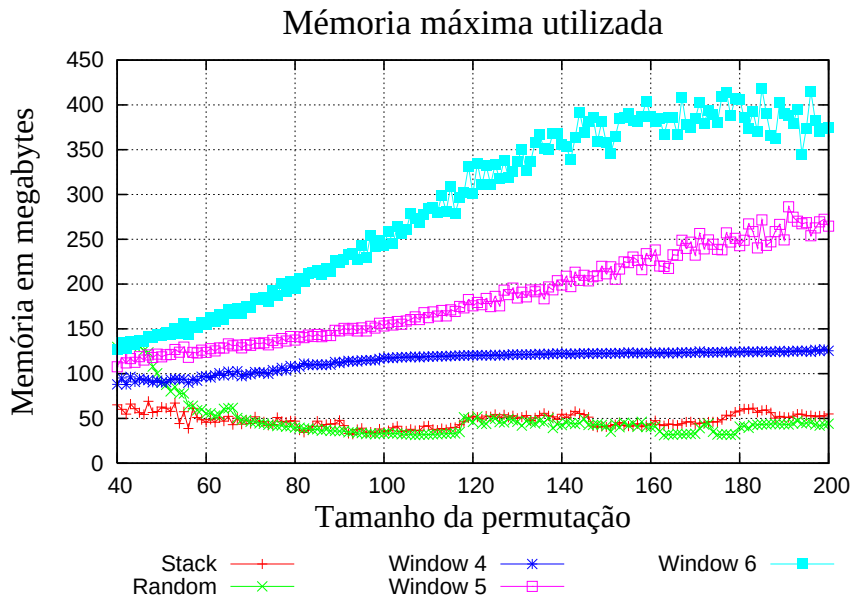
500 permutações aleatórias $n = 15$ e $d(\pi) = 12$

Comparação dos algoritmos

- Segundo teste – 1 minuto (valores médios)



(a) Número de traces



(b) Memória máxima utilizada

100 permutações aleatórias $40 \leq n \leq 200$ e $d(\pi) = \lceil (n+1)/2 \rceil$

Comparação dos algoritmos

- Algoritmo `Stack`
  - "Ramos mortos": Inadequado para grandes permutações
- Algoritmo `Random`
  - Performance similar ao do `Stack`
  - Aceita restrições
- Algoritmo `Window`
  - Capaz de produzir maior enumeração
  - Processo de combinação de *traces* reduz desempenho
  - Necessita de maior quantidade de memória

Comparação dos algoritmos

- Algoritmo `Stack`
  - "Ramos mortos": Inadequado para grandes permutações
- Algoritmo `Random`
  - Performance similar ao do `Stack`
  - Aceita restrições
- Algoritmo `Window`
  - Capaz de produzir maior enumeração
  - Processo de combinação de *traces* reduz desempenho
  - Necessita de maior quantidade de memória

Comparação dos algoritmos

- Algoritmo `Stack`
  - "Ramos mortos": Inadequado para grandes permutações
- Algoritmo `Random`
  - Performance similar ao do `Stack`
  - Aceita restrições
- Algoritmo `Window`
  - Capaz de produzir maior enumeração
  - Processo de combinação de *traces* reduz desempenho
  - Necessita de maior quantidade de memória

Trabalhos futuros

- Enumeração (total ou parcial) de *traces*
  - Passo inicial de um objetivo maior
    - ⋆ Auxiliar biólogos na análise de cenários evolutivos
- Incorporção de novas restrições biológicas
- Critérios de classificação de *traces*
  - Criação de um esquema de pontuação
  - Ordenação de *traces* de acordo com a pontuação
  - Identificação de *traces* relevantes
    - ⋆ Ponto de vista biológico

Trabalhos futuros

- Enumeração (total ou parcial) de *traces*
  - Passo inicial de um objetivo maior
    - ⋆ Auxiliar biólogos na análise de cenários evolutivos
- Incorporção de novas restrições biológicas
- Critérios de classificação de *traces*
  - Criação de um esquema de pontuação
  - Ordenação de *traces* de acordo com a pontuação
  - Identificação de *traces* relevantes
    - ⋆ Ponto de vista biológico

Trabalhos futuros

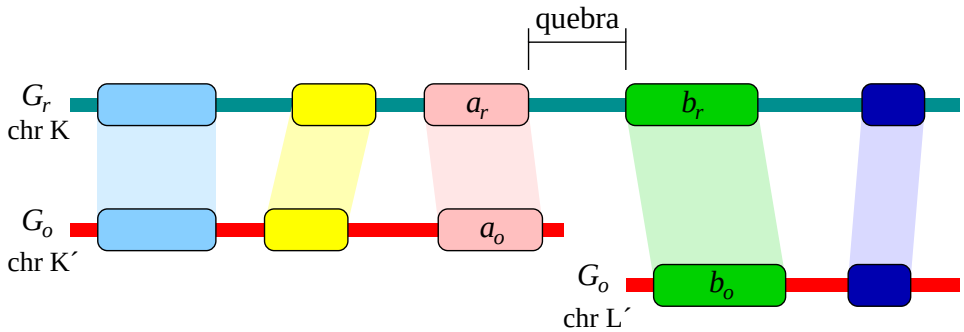
- Enumeração (total ou parcial) de *traces*
  - Passo inicial de um objetivo maior
    - ⋆ Auxiliar biólogos na análise de cenários evolutivos
- Incorporção de novas restrições biológicas
- Critérios de classificação de *traces*
  - Criação de um esquema de pontuação
  - Ordenação de *traces* de acordo com a pontuação
  - Identificação de *traces* relevantes
    - ⋆ Ponto de vista biológico

Parte II

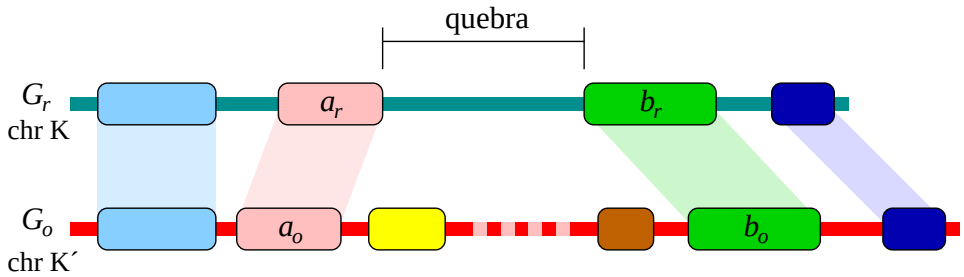
Identificação de *breakpoints*

Pontos de quebra

- *Breakpoints*
  - Regiões onde molécula de DNA foi "quebrada"
  - Eventos de rearranjo
    - ★ Evolução: rearranjos evolucionários



(a) Inter-cromossômico



(b) Intra-cromossômico

Identificação e refinamento de *breakpoints*

- Métodos existentes identificam blocos de sintenia
  - *Breakpoints* são apenas uma consequência
- Lemaitre *et al.*, 2008
  - Método baseado em duas etapas
    - Identificação de *breakpoints*
    - Refinamento de *breakpoints*

Identificação e refinamento de *breakpoints*

- Métodos existentes identificam blocos de sintenia
  - *Breakpoints* são apenas uma consequência
- Lemaitre *et al.*, 2008
  - Método baseado em duas etapas
    - Identificação de *breakpoints*
    - Refinamento de *breakpoints*

Identificação de *breakpoints*

- Lista de pares de genes ortólogos 1-1
- Blocos de sintenia ordenados
- Blocos sem sobreposição

Identificação de *breakpoints*

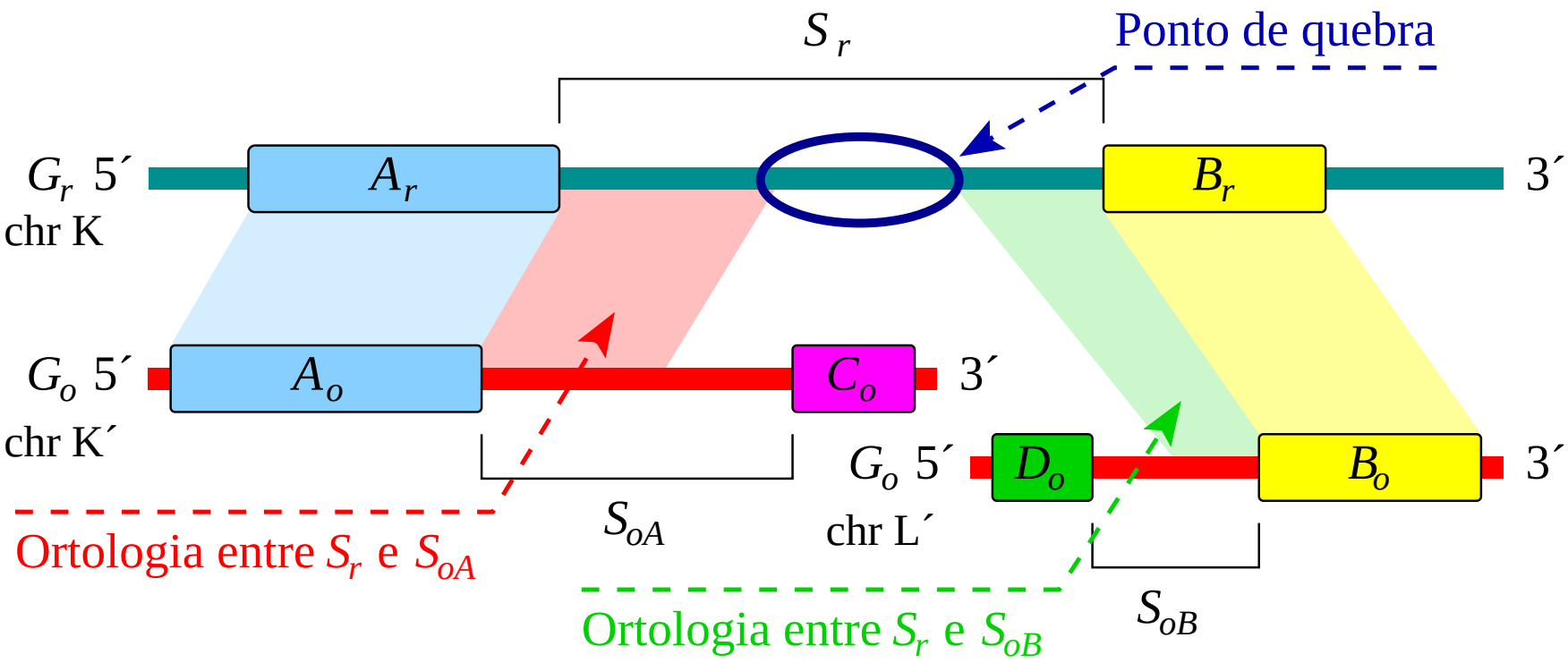
- Lista de pares de genes ortólogos 1-1
- Blocos de sintenia ordenados
- Blocos sem sobreposição

Identificação de *breakpoints*

- Lista de pares de genes ortólogos 1-1
- Blocos de sintenia ordenados
- Blocos sem sobreposição

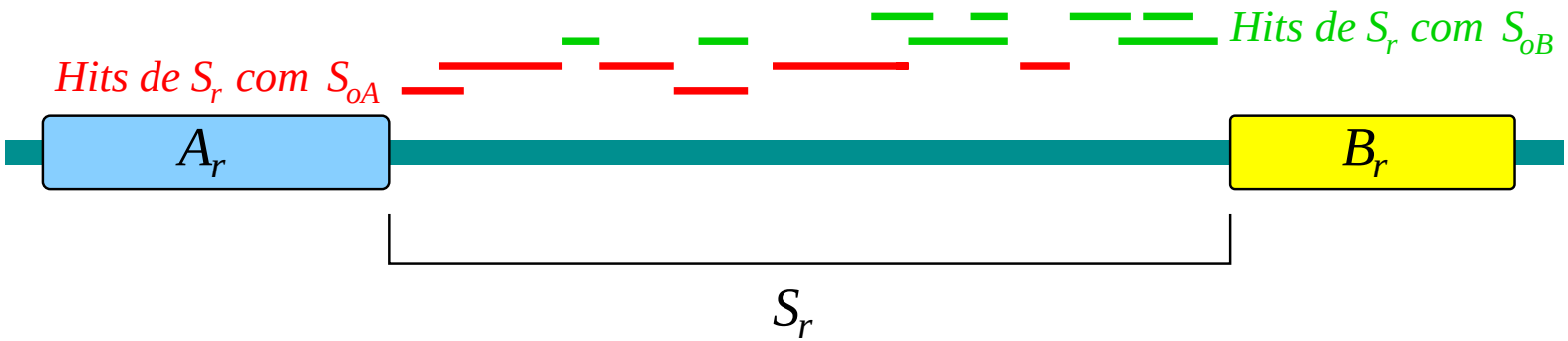
Refinamento de *breakpoints*

- Definição de sequências de interesse
  - S_r, S_{oA} e S_{oB}



Refinamento de *breakpoints*

- Alinhamento de sequências
  - S_r contra S_{oA}
  - S_r contra S_{oB}
- Algoritmo `BLASTZ`
  - Adequado para alinhamento de sequências intergênicas



Refinamento de *breakpoints*

- Conversão de S_r em uma sequência numérica I
  - $I_k = +1$: *hit* com S_{oA} e nenhum *hit* com S_{oB}
  - $I_k = -1$: *hit* com S_{oB} e nenhum *hit* com S_{oA}
  - $I_k = 0$: nenhum *hit* ou *hits* com S_{oA} e S_{oB}
- Métodos de detecção de ruptura
  - Segmentar I (S_r) em três regiões
  - Minimizar função de contraste (erro quadrático)
- Validação estatística
  - Verificar se S_r é estruturada em três segmentos

Refinamento de *breakpoints*

- Conversão de S_r em uma sequência numérica I
  - $I_k = +1$: *hit* com S_{oA} e nenhum *hit* com S_{oB}
  - $I_k = -1$: *hit* com S_{oB} e nenhum *hit* com S_{oA}
  - $I_k = 0$: nenhum *hit* ou *hits* com S_{oA} e S_{oB}
- Métodos de detecção de ruptura
  - Segmentar I (S_r) em três regiões
  - Minimizar função de contraste (erro quadrático)
- Validação estatística
  - Verificar se S_r é estruturada em três segmentos

Refinamento de *breakpoints*

- Conversão de S_r em uma sequência numérica I
  - $I_k = +1$: *hit* com S_{oA} e nenhum *hit* com S_{oB}
  - $I_k = -1$: *hit* com S_{oB} e nenhum *hit* com S_{oA}
  - $I_k = 0$: nenhum *hit* ou *hits* com S_{oA} e S_{oB}
- Métodos de detecção de ruptura
  - Segmentar I (S_r) em três regiões
  - Minimizar função de contraste (erro quadrático)
- Validação estatística
  - Verificar se S_r é estruturada em três segmentos

Implementação do pacote `Cassis`

- Implementação em `Perl` e `R`
  - *Script* principal facilita utilização
  - Modularização do código permite extensões
- Blocos de sintenia produzidos por outros métodos
  - Testes comparativos com `Compara` e `Mauve`
  - Capacidade de refinar *breakpoints* produzidos por outros métodos
- Trabalho publicado na *Bioinformatics*
- Disponível sob licença GNU GPL
  - `http://pbil.univ-lyon1.fr/software/Cassis/`

Implementação do pacote `Cassis`

- Implementação em `Perl` e `R`
  - *Script* principal facilita utilização
  - Modularização do código permite extensões
- Blocos de sintenia produzidos por outros métodos
  - Testes comparativos com `Compara` e `Mauve`
  - Capacidade de refinar *breakpoints* produzidos por outros métodos
- Trabalho publicado na *Bioinformatics*
- Disponível sob licença GNU GPL
  - `http://pbil.univ-lyon1.fr/software/Cassis/`

Implementação do pacote `Cassis`

- Implementação em `Perl` e `R`
  - *Script* principal facilita utilização
  - Modularização do código permite extensões
- Blocos de sintenia produzidos por outros métodos
  - Testes comparativos com `Compara` e `Mauve`
  - Capacidade de refinar *breakpoints* produzidos por outros métodos
- Trabalho publicado na *Bioinformatics*
- Disponível sob licença GNU GPL
  - `http://pbil.univ-lyon1.fr/software/Cassis/`

Implementação do pacote `Cassis`

- Implementação em `Perl` e `R`
  - *Script* principal facilita utilização
  - Modularização do código permite extensões
- Blocos de sintenia produzidos por outros métodos
  - Testes comparativos com `Compara` e `Mauve`
  - Capacidade de refinar *breakpoints* produzidos por outros métodos
- Trabalho publicado na *Bioinformatics*
- Disponível sob licença GNU GPL
  - `http://pbil.univ-lyon1.fr/software/Cassis/`

Trabalhos futuros

- Estender metodologia para múltiplos genomas
  - Definição de âncoras
    - ⋆ Exigências sobre relações de ortologia
  - Alinhamentos de sequências de interesse
    - ⋆ Alinhamento dois a dois
    - ⋆ Alinhamento múltiplo
  - Segmentação
    - ⋆ Adaptação para dados de múltiplos alinhamentos

Trabalhos futuros

- Estender metodologia para múltiplos genomas
  - Definição de âncoras
    - ⋆ Exigências sobre relações de ortologia
  - Alinhamentos de sequências de interesse
    - ⋆ Alinhamento dois a dois
    - ⋆ Alinhamento múltiplo
  - Segmentação
    - ⋆ Adaptação para dados de múltiplos alinhamentos

Trabalhos futuros

- Estender metodologia para múltiplos genomas
  - Definição de âncoras
    - ★ Exigências sobre relações de ortologia
  - Alinhamentos de sequências de interesse
    - ★ Alinhamento dois a dois
    - ★ Alinhamento múltiplo
  - Segmentação
    - ★ Adaptação para dados de múltiplos alinhamentos

Trabalhos futuros

- Estender metodologia para múltiplos genomas
  - Definição de âncoras
    - ★ Exigências sobre relações de ortologia
  - Alinhamentos de sequências de interesse
    - ★ Alinhamento dois a dois
    - ★ Alinhamento múltiplo
  - Segmentação
    - ★ Adaptação para dados de múltiplos alinhamentos

Análise de sequências intergênicas

- Sequências em regiões de *breakpoints*
  - Menores níveis de similaridade com seus ortólogos
  - Presença de duplicações e elementos transponíveis
- Nova metodologia de detecção de *breakpoints*
  - Independente de informações prévias de ortologia
  - Baseada nas características das regiões intergênicas

Análise de sequências intergênicas

- Sequências em regiões de *breakpoints*
  - Menores níveis de similaridade com seus ortólogos
  - Presença de duplicações e elementos transponíveis
- Nova metodologia de detecção de *breakpoints*
  - Independente de informações prévias de ortologia
  - Baseada nas características das regiões intergênicas

Alinhamento de sequências

- Dados obtidos da plataforma `Ensembl` (versão: 56)
  - *Homo sapiens*: 46.946 genes
  - *Mus musculus*: 31.458 genes
- Definição de regiões intergênicas ($>$ 100 bp)
  - *Homo sapiens*: 29.542 regiões
  - *Mus musculus*: 26.296 regiões
- Alinhamento com `BLASTN`
  - Pares de regiões com algum nível de similaridade
  - 5.022.606 pares de regiões
- Alinhamento com `BLASTZ`
  - Algoritmo mais adequado às regiões intergênicas
  - 135.967 pares de regiões

Alinhamento de sequências

- Dados obtidos da plataforma `Ensembl` (versão: 56)
  - *Homo sapiens*: 46.946 genes
  - *Mus musculus*: 31.458 genes
- Definição de regiões intergênicas ($>$ 100 bp)
  - *Homo sapiens*: 29.542 regiões
  - *Mus musculus*: 26.296 regiões
- Alinhamento com `BLASTN`
  - Pares de regiões com algum nível de similaridade
  - 5.022.606 pares de regiões
- Alinhamento com `BLASTZ`
  - Algoritmo mais adequado às regiões intergênicas
  - 135.967 pares de regiões

Alinhamento de sequências

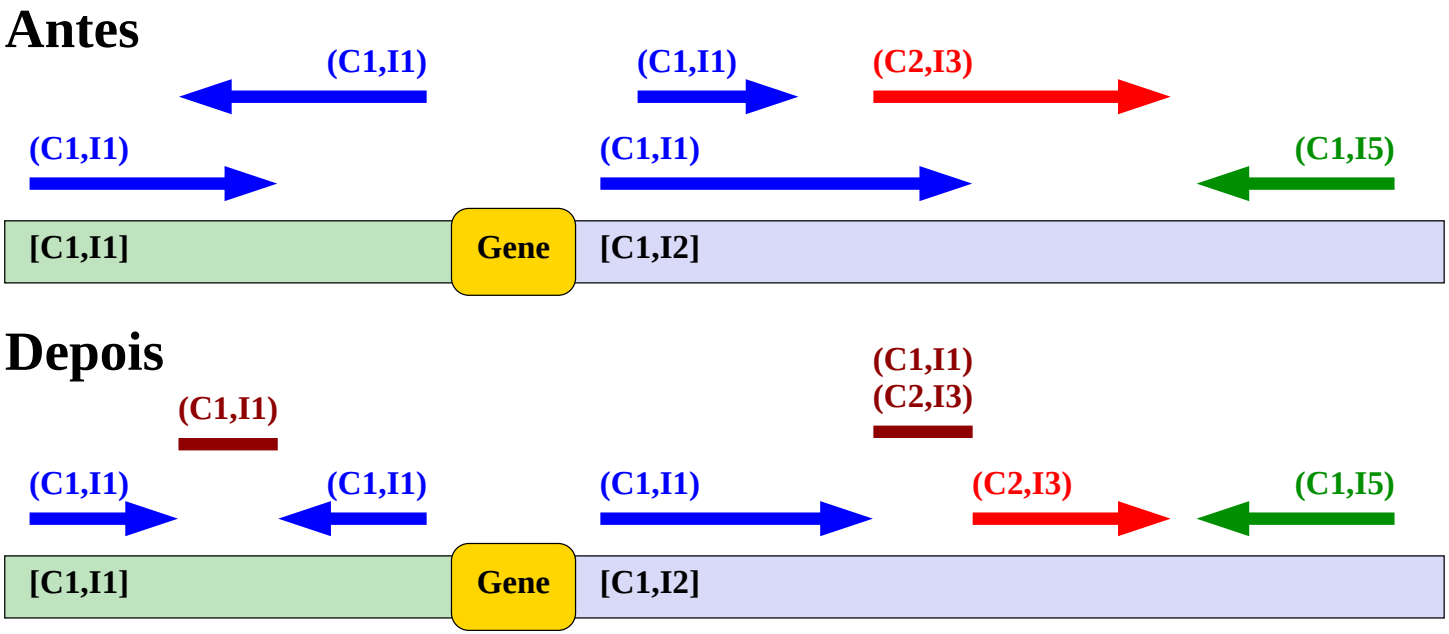
- Dados obtidos da plataforma `Ensembl` (versão: 56)
  - *Homo sapiens*: 46.946 genes
  - *Mus musculus*: 31.458 genes
- Definição de regiões intergênicas ($>$ 100 bp)
  - *Homo sapiens*: 29.542 regiões
  - *Mus musculus*: 26.296 regiões
- Alinhamento com `BLASTN`
  - Pares de regiões com algum nível de similaridade
  - 5.022.606 pares de regiões
- Alinhamento com `BLASTZ`
  - Algoritmo mais adequado às regiões intergênicas
  - 135.967 pares de regiões

Alinhamento de sequências

- Dados obtidos da plataforma `Ensembl` (versão: 56)
  - *Homo sapiens*: 46.946 genes
  - *Mus musculus*: 31.458 genes
- Definição de regiões intergênicas ($>$ 100 bp)
  - *Homo sapiens*: 29.542 regiões
  - *Mus musculus*: 26.296 regiões
- Alinhamento com `BLASTN`
  - Pares de regiões com algum nível de similaridade
  - 5.022.606 pares de regiões
- Alinhamento com `BLASTZ`
  - Algoritmo mais adequado às regiões intergênicas
  - 135.967 pares de regiões

Padronização dos *hits*

- Padronização dos *hits* ao longo de um cromossomo
  - Eliminação de sobreposições
  - Distinção de posições com *hits*
    - Com única região e orientação
    - Com mais de uma região e/ou orientação



Comparação de regiões intergênicas

- Dificuldades
  - Regiões intergênicas possuem diferentes comprimentos
  - Cobertura de *hits* ao longo do cromossomo é ruidosa
- Fragmentação de regiões intergênicas
  - Fragmentos de 100 bp sem sobreposição
  - Cobertura por *hits* calculada para cada fragmento
- Estratégia de janela deslizante
  - Cálculo de cobertura dentro da janela
  - Suavização do ruído

Comparação de regiões intergênicas

- Dificuldades
  - Regiões intergênicas possuem diferentes comprimentos
  - Cobertura de *hits* ao longo do cromossomo é ruidosa
- Fragmentação de regiões intergênicas
  - Fragmentos de 100 bp sem sobreposição
  - Cobertura por *hits* calculada para cada fragmento
- Estratégia de janela deslizante
  - Cálculo de cobertura dentro da janela
  - Suavização do ruído

Comparação de regiões intergênicas

- Dificuldades
  - Regiões intergênicas possuem diferentes comprimentos
  - Cobertura de *hits* ao longo do cromossomo é ruidosa
- Fragmentação de regiões intergênicas
  - Fragmentos de 100 bp sem sobreposição
  - Cobertura por *hits* calculada para cada fragmento
- Estratégia de janela deslizante
  - Cálculo de cobertura dentro da janela
  - Suavização do ruído

Janela deslizante

- Janela deslizante de tamanho $2w + 1$
  - Medida em número de fragmentos
  - Valor calculado é associado ao fragmento central $(w + 1)$
  - Janela deslocada de um em um fragmento
- Identificar regiões
  - Cobertura menor do que as regiões no entorno
- Um ou dois níveis de processamento
  - Nível 1 – Cromossomo
    - ⋆ Distinção de regiões que possuem cobertura menor e maior do que a média observada no cromossomo inteiro
  - Nível 2 – Regiões produzidas no nível 1
    - ⋆ Distinção de regiões que possuem cobertura menor e maior do que a média observada na região processada

Janela deslizante

- Janela deslizante de tamanho $2w + 1$
  - Medida em número de fragmentos
  - Valor calculado é associado ao fragmento central $(w + 1)$
  - Janela deslocada de um em um fragmento
- Identificar regiões
  - Cobertura menor do que as regiões no entorno
- Um ou dois níveis de processamento
  - Nível 1 – Cromossomo
    - ⋆ Distinção de regiões que possuem cobertura menor e maior do que a média observada no cromossomo inteiro
  - Nível 2 – Regiões produzidas no nível 1
    - ⋆ Distinção de regiões que possuem cobertura menor e maior do que a média observada na região processada

Janela deslizante

- Janela deslizante de tamanho $2w + 1$
  - Medida em número de fragmentos
  - Valor calculado é associado ao fragmento central $(w + 1)$
  - Janela deslocada de um em um fragmento
- Identificar regiões
  - Cobertura menor do que as regiões no entorno
- Um ou dois níveis de processamento
  - Nível 1 – Cromossomo
    - ⋆ Distinção de regiões que possuem cobertura menor e maior do que a média observada no cromossomo inteiro
  - Nível 2 – Regiões produzidas no nível 1
    - ⋆ Distinção de regiões que possuem cobertura menor e maior do que a média observada na região processada

Encadeamento de *hits*

- Regiões identificadas pela janela deslizante
  - Cobertura abaixo da média da região base
- Seleção de *hits* de C_r com $\sigma_o \in \{-1, +1\}$
  - Par ou dois pares (C_o, σ_o) mais representados
  - Segundo par deve ter pelo menos $p2_{min}\%$ dos *hits*
  - Par mais representado deve somar pelo menos s_{min} bp
  - *Hits* são posicionados ao longo de C_r
- Cálculo de Δ entre *hits* consecutivos
  - Se cromossomos e orientações são iguais
    - $\Delta_i = l_{o(i)} - l_{o(i-1)}$
  - Caso contrário
    - $\Delta_i = \infty$

Encadeamento de *hits*

- Regiões identificadas pela janela deslizante
  - Cobertura abaixo da média da região base
- Seleção de *hits* de C_r com $\sigma_o \in \{-1, +1\}$
  - Par ou dois pares (C_o, σ_o) mais representados
  - Segundo par deve ter pelo menos $p2_{min}\%$ dos *hits*
  - Par mais representado deve somar pelo menos s_{min} bp
  - *Hits* são posicionados ao longo de C_r
- Cálculo de Δ entre *hits* consecutivos
  - Se cromossomos e orientações são iguais
    - $\Delta_i = l_{o(i)} - l_{o(i-1)}$
  - Caso contrário
    - $\Delta_i = \infty$

Encadeamento de *hits*

- Regiões identificadas pela janela deslizante
  - Cobertura abaixo da média da região base
- Seleção de *hits* de C_r com $\sigma_o \in \{-1, +1\}$
  - Par ou dois pares (C_o, σ_o) mais representados
  - Segundo par deve ter pelo menos $p2_{min}\%$ dos *hits*
  - Par mais representado deve somar pelo menos s_{min} bp
  - *Hits* são posicionados ao longo de C_r
- Cálculo de Δ entre *hits* consecutivos
  - Se cromossomos e orientações são iguais
    - $\Delta_i = l_{o(i)} - l_{o(i-1)}$
  - Caso contrário
    - $\Delta_i = \infty$

Encadeamento de *hits*

- Sequência de *hits* de C_r é percorrida
- *Hits* com $|\Delta_i| \leq \Delta_{max}$ são encadeados
- *Hits* com $|\Delta_i| > \Delta_{max}$ são analisados
  - Se *hits* $h_{(i-1)}$ e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o)
    - ⋆ *Hit* h_i é "intruso"
    - ⋆ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado
  - Senão, se os *hits* h_i e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o) e $|\Delta_{(i+1)}| \leq \Delta_{max}$
    - ⋆ *Breakpoint* entre $h_{(i-1)}$ e $h_{(i)}$
  - Senão
    - ⋆ *Hit* h_i é "intruso"
    - ⋆ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado

Encadeamento de *hits*

- Sequência de *hits* de C_r é percorrida
- *Hits* com $|\Delta_i| \leq \Delta_{max}$ são encadeados
- *Hits* com $|\Delta_i| > \Delta_{max}$ são analisados
  - Se *hits* $h_{(i-1)}$ e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o)
    - ⋆ *Hit* h_i é "intruso"
    - ⋆ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado
  - Senão, se os *hits* h_i e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o) e $|\Delta_{(i+1)}| \leq \Delta_{max}$
    - ⋆ *Breakpoint* entre $h_{(i-1)}$ e $h_{(i)}$
  - Senão
    - ⋆ *Hit* h_i é "intruso"
    - ⋆ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado

Encadeamento de *hits*

- Sequência de *hits* de C_r é percorrida
- *Hits* com $|\Delta_i| \leq \Delta_{max}$ são encadeados
- *Hits* com $|\Delta_i| > \Delta_{max}$ são analisados
  - Se *hits* $h_{(i-1)}$ e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o)
    - ⋆ *Hit* h_i é "intruso"
    - ⋆ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado
  - Senão, se os *hits* h_i e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o) e $|\Delta_{(i+1)}| \leq \Delta_{max}$
    - ⋆ *Breakpoint* entre $h_{(i-1)}$ e $h_{(i)}$
  - Senão
    - ⋆ *Hit* h_i é "intruso"
    - ⋆ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado

Encadeamento de *hits*

- Sequência de *hits* de C_r é percorrida
- *Hits* com $|\Delta_i| \leq \Delta_{max}$ são encadeados
- *Hits* com $|\Delta_i| > \Delta_{max}$ são analisados
  - Se *hits* $h_{(i-1)}$ e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o)
    - ⋆ *Hit* h_i é "intruso"
    - ⋆ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado
  - Senão, se os *hits* h_i e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o) e $|\Delta_{(i+1)}| \leq \Delta_{max}$
    - ⋆ *Breakpoint* entre $h_{(i-1)}$ e $h_{(i)}$
  - Senão
    - ⋆ *Hit* h_i é "intruso"
    - ⋆ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado

Encadeamento de *hits*

- Sequência de *hits* de C_r é percorrida
- *Hits* com $|\Delta_i| \leq \Delta_{max}$ são encadeados
- *Hits* com $|\Delta_i| > \Delta_{max}$ são analisados
  - Se *hits* $h_{(i-1)}$ e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o)
    - ★ *Hit* h_i é "intruso"
    - ★ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado
  - Senão, se os *hits* h_i e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o) e $|\Delta_{(i+1)}| \leq \Delta_{max}$
    - ★ *Breakpoint* entre $h_{(i-1)}$ e $h_{(i)}$
  - Senão
    - ★ *Hit* h_i é "intruso"
    - ★ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado

Encadeamento de *hits*

- Sequência de *hits* de C_r é percorrida
- *Hits* com $|\Delta_i| \leq \Delta_{max}$ são encadeados
- *Hits* com $|\Delta_i| > \Delta_{max}$ são analisados
  - Se *hits* $h_{(i-1)}$ e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o)
    - ★ *Hit* h_i é "intruso"
    - ★ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado
  - Senão, se os *hits* h_i e $h_{(i+1)}$ possuem mesmo par (C_o, σ_o) e $|\Delta_{(i+1)}| \leq \Delta_{max}$
    - ★ *Breakpoint* entre $h_{(i-1)}$ e $h_{(i)}$
  - Senão
    - ★ *Hit* h_i é "intruso"
    - ★ *Hit* h_i é eliminado e $\Delta_{(i+1)}$ é atualizado

Testes

- Janela deslizante
  - Um nível
    - $w_1 \in \{50, 125, 250, 500, 1.000, 1.500, 2.000, 2.500\}$
  - Dois níveis
    - $w_1 = 500$
    - $w_2 \in \{62, 125, 250, 375, 500, 625, 750\}$

- Encadeamento de *hits*
  - $s_{min} = 100$ bp
  - $p2_{min} = 10\%$
  - $\Delta_{max} = 10$

Testes

- Janela deslizante
  - Um nível
    - $w_1 \in \{50, 125, 250, 500, 1.000, 1.500, 2.000, 2.500\}$
  - Dois níveis
    - $w_1 = 500$
    - $w_2 \in \{62, 125, 250, 375, 500, 625, 750\}$

- Encadeamento de *hits*
  - $s_{min} = 100$ bp
  - $p2_{min} = 10\%$
  - $\Delta_{max} = 10$

Comparação com Cassis

- Utilizando dados de ortologia
  - Cassis identificou 369 *breakpoints*
- Nova metodologia
  - Conforme a configuração das janelas deslizantes
    - Entre 47,7% e 62,3% dos *breakpoints* refinados do Cassis tiveram um único *breakpoint* correspondente
  - Pequena vantagem para janela deslizante com dois níveis
  - Problema
    - Grande número de *breakpoints* sem correspondência
    - Prováveis falsos positivos

Comparação com Cassis

- Utilizando dados de ortologia
  - Cassis identificou 369 *breakpoints*
- Nova metodologia
  - Conforme a configuração das janelas deslizantes
    - Entre 47,7% e 62,3% dos *breakpoints* refinados do Cassis tiveram um único *breakpoint* correspondente
  - Pequena vantagem para janela deslizante com dois níveis
  - Problema
    - Grande número de *breakpoints* sem correspondência
    - Prováveis falsos positivos

Comparação com Cassis

- Utilizando dados de ortologia
  - Cassis identificou 369 *breakpoints*
- Nova metodologia
  - Conforme a configuração das janelas deslizantes
    - ★ Entre 47,7% e 62,3% dos *breakpoints* refinados do Cassis tiveram um único *breakpoint* correspondente
  - Pequena vantagem para janela deslizante com dois níveis
  - Problema
    - ★ Grande número de *breakpoints* sem correspondência
    - ★ Prováveis falsos positivos

Comparação com Cassis

- Utilizando dados de ortologia
  - Cassis identificou 369 *breakpoints*
- Nova metodologia
  - Conforme a configuração das janelas deslizantes
    - Entre 47,7% e 62,3% dos *breakpoints* refinados do Cassis tiveram um único *breakpoint* correspondente
  - Pequena vantagem para janela deslizante com dois níveis
  - Problema
    - Grande número de *breakpoints* sem correspondência
    - Prováveis falsos positivos

Trabalhos futuros

- Embrião de nova metodologia
  - Mesmo sem utilizar informação de ortologia é capaz de encontrar *breakpoints*
  - Amplo potencial de aprimoramento
- Avaliação cuidadosa dos *breakpoints*
  - Falsos positivos
  - Falsos negativos
  - Verdadeiros positivos

Trabalhos futuros

- Embrião de nova metodologia
  - Mesmo sem utilizar informação de ortologia é capaz de encontrar *breakpoints*
  - Amplo potencial de aprimoramento
- Avaliação cuidadosa dos *breakpoints*
  - Falsos positivos
  - Falsos negativos
  - Verdadeiros positivos

Trabalhos futuros

- Considerar regiões com alta cobertura
- Aperfeiçoar critérios
  - Encadeamento de *hits*
  - Definição de ocorrência de *breakpoint*
- Ajuste de parâmetros

Trabalhos futuros

- Considerar regiões com alta cobertura
- Aperfeiçoar critérios
  - Encadeamento de *hits*
  - Definição de ocorrência de *breakpoint*
- Ajuste de parâmetros

Trabalhos futuros

- Considerar regiões com alta cobertura
- Aperfeiçoar critérios
  - Encadeamento de *hits*
  - Definição de ocorrência de *breakpoint*
- Ajuste de parâmetros

Contribuições do trabalho

- Novo algoritmo de enumeração de *traces*
  - Estrutura de pilha
  - Melhora no consumo de tempo e memória
- Nova abordagem para enumeração de *traces*
  - Enumeração parcial de *traces*
  - Três algoritmos propostos
- Identificação e refinamento de *breakpoints*
  - Metodologia proposta por Lemaitre *et al.*, 2008
  - Implementação do pacote `Cassis`
- Nova metodologia para identificação de *breakpoints*
  - Análise de regiões intergênicas
  - Não necessita de informações de ortologia

Contribuições do trabalho

- Novo algoritmo de enumeração de *traces*
  - Estrutura de pilha
  - Melhora no consumo de tempo e memória
- Nova abordagem para enumeração de *traces*
  - Enumeração parcial de *traces*
  - Três algoritmos propostos
- Identificação e refinamento de *breakpoints*
  - Metodologia proposta por Lemaitre *et al.*, 2008
  - Implementação do pacote `Cassis`
- Nova metodologia para identificação de *breakpoints*
  - Análise de regiões intergênicas
  - Não necessita de informações de ortologia

Contribuições do trabalho

- Novo algoritmo de enumeração de *traces*
  - Estrutura de pilha
  - Melhora no consumo de tempo e memória
- Nova abordagem para enumeração de *traces*
  - Enumeração parcial de *traces*
  - Três algoritmos propostos
- Identificação e refinamento de *breakpoints*
  - Metodologia proposta por Lemaitre *et al.*, 2008
  - Implementação do pacote `Cassis`
- Nova metodologia para identificação de *breakpoints*
  - Análise de regiões intergênicas
  - Não necessita de informações de ortologia

Contribuições do trabalho

- Novo algoritmo de enumeração de *traces*
  - Estrutura de pilha
  - Melhora no consumo de tempo e memória
- Nova abordagem para enumeração de *traces*
  - Enumeração parcial de *traces*
  - Três algoritmos propostos
- Identificação e refinamento de *breakpoints*
  - Metodologia proposta por Lemaitre *et al.*, 2008
  - Implementação do pacote `Cassis`
- Nova metodologia para identificação de *breakpoints*
  - Análise de regiões intergênicas
  - Não necessita de informações de ortologia