Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ordenação por Reversões de Permutações sem Sinal Usando uma Abordagem de Algoritmos Genéticos

José Luis Soncco Álvarez

Dissertação apresentado como requisito parcial
para a conclusão do Mestrado em Informática

Orientador
Prof. Dr. Mauricio Ayala Rincón

Brasília
2013

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenador: Prof. Dr. Ricardo Pezzuol Jacobi

Banca examinadora composta por:

Prof. Dr. Mauricio Ayala Rincón (Orientador) — CIC/UnB
Prof. Dr. Edward Hermann Haeusler — PUC-Rio
Prof. Dr. André Drummond — CIC-UnB
Prof. Dr. Carlos Humberto Llanos Quintero — EMC-UnB

CIP — Catalogação Internacional na Publicação

Soncco Álvarez, José Luis.

Ordenação por Reversões de Permutações sem Sinal Usando uma Abordagem de Algoritmos Genéticos / José Luis Soncco Álvarez. Brasília : UnB, 2013.
127 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2013.

1. Ordenação por reversões, 2. rearranjo de genomas, 3. ordenação de permutações sem sinal, 4. algoritmo de aproximação, 5. algoritmos genéticos

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ordenação por Reversões de Permutações sem Sinal Usando uma Abordagem de Algoritmos Genéticos

José Luis Soncco Álvarez

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof. Dr. Mauricio Ayala Rincón (Orientador)
CIC/UnB

Prof. Dr. Edward Hermann Haeusler
PUC-Rio

Prof. Dr. André Drummond
CIC-UnB

Prof. Dr. Carlos Humberto Llanos Quintero
EMC-UnB

Prof. Dr. Ricardo Pezzuol Jacobi
Coordenador do Mestrado em Informática

Brasília, 5 de março de 2013

Dedicatória

Dedico este trabalho a toda minha família, meus irmãos, primos, tios, avós, e especialmente aos meus pais que sempre me apoiaram para continuar meus estudos fora do meu pais.

Agradecimentos

Agradeço a todas a pessoas que contribuíram direta é indiretamente com a elaboração deste trabalho. Aos meus amigos que me acompanharam e presenciaram o esforço que eu fiz dia a dia durante o mestrado. Aos professores do mestrado, por compartilhar seus conhecimentos e sua experiência, sem os quais não poderia ter concluído com sucesso este trabalho. Ao meu orientador Mauricio Ayala Rincón, de quem eu sempre recebi bons conselhos para melhorar meu trabalho e de quem aprendi bastante sobre como fazer uma boa pesquisa; também devo agradecer a Thaynara que foi de muita ajuda nas correções do trabalho, e que soube ser paciente quando eu quis tirar duvidas sobre a minha pesquisa.

Abstract

Sorting permutations by reversals is one of the most challenging problems related to the analysis of the evolutionary distance between organisms, whose results can be used in the construction of phylogenetic trees bases on this distance.

In the case of signed permutations, the problem can be solved in linear time, however in the case of unsigned permutations the problem is more complex, since it was shown to be $\mathcal{NP}$-hard and it is unknown whether it is $\mathcal{NP}$-complete or not; this fact motivated the proposal of several approximation, and evolutionary computing algorithms.

In this work, we propose genetic algorithms (GA) to solve the problem of sorting unsigned permutations. Initially, we propose a standard GA approach based on the method proposed by Auyeung and Abraham, which uses exact polynomial solutions for the case of signed permutations, for solving the problem with unsigned permutations. Further, we propose an improved genetic algorithm, which uses the heuristic of elimination of break points in early generations and then the standard approach.

Several experiments were made using as inputs permutations generated randomly by choosing a random element over a set of numbers, and by applying random reversals over an sorted permutation. Also, was used Gollan permutations that it's well-known that can be sorted by $n - 1$ reversals, where n is the length of the permutation.

Since previous GA approaches have used imprecise control mechanisms for checking the accuracy of their answers, a great deal of effort was necessary in order to develop a reliable approximate algorithm. This gave rise to a theoretical development based on the well-known Christie's 1.5 ratio approximation algorithm and its further implementation. Experiments showed that both AG approaches compute answers that are better than the ones computed by previous approaches as well as than the ones computed with the adjusted correct 1.5 approximation algorithm.

Keywords: Sorting by reversals, genome rearrangement, sorting unsigned permutations, approximation algorithm ,genetic algorithms

Resumo

Ordenação de permutações por reversões é um dos problemas mais desafiantes relacionados com a análise da distância evolutiva entre organismos, cujos resultados podem ser usados na construção de árvores filogenéticas baseadas nesta distância.

No caso de permutações com sinal, o problema pode ser resolvido em tempo linear, porém, no caso de permutações sem sinal o problema é mais complexo, já que foi demonstrado ser $\mathcal{NP}$-difícil e com uma questão ainda em aberto: se é ou não $\mathcal{NP}$-completo; este foi o motivo pelo qual foram propostos diversos algoritmos de aproximação e de computação evolucionária.

Neste trabalho, é proposto um algoritmo genético(AG) padrão para resolver o problema de ordenação de permutações sem sinal. Este enfoque está baseado no método proposto por Auyeung e Abraham, que usa soluções exatas para o caso de permutações com sinal, para resolver a versão do problema com permutações sem sinal. Adicionalmente, foi proposto um algoritmo genético melhorado (hibrido), que usa uma heurística de eliminação de pontos de quebra em gerações iniciais.

Diversos experimentos foram feitos tomando como entrada permutações geradas aleatoriamente, escolhendo um elemento aleatório sobre um conjunto de números, ou aplicando reversões aleatórias sobre uma permutação ordenada. Ademais, foram usadas permutações de Gollan as quais sabemos que podem ser ordenadas usando $n - 1$ reversões, onde n é o comprimento da permutação.

Desde que muitos enfoques de AG's usaram mecanismos de controle imprecisos para validar a precisão das suas respostas, foi necessário um grande esforço para desenvolver uma algoritmo de aproximação confiável. Dando origem a um desenvolvimento teórico baseado no algoritmo de raio de aproximação 1.5 proposto por Christie, e sua posterior implementação. Os experimentos mostraram que ambos AG fornecem respostas que são melhores do que aquelas fornecidas por métodos relacionados prévios, tanto como os que são fornecidos pelo algoritmo de raio de aproximação 1.5 corrigido.

Palavras-chave: Ordenação por reversões, rearranjo de genomas, ordenação de permutações sem sinal, algoritmo de aproximação, algoritmos genéticos

Sumário

Capítulo 1

Introdução

No final dos anos 80 Jeffrey Palmer e seus colegas descobriram um novo padrão de mudança evolutiva em organelas de plantas. Eles compararam os genomas mitocondriais de *Brassica oleracea*(repolho) e *Brassica campestris*(nabo), os quais estão estreitamente relacionados (muitos genes são 99% - 99,9% idênticos). Para sua surpresa, enquanto os genes em si são quase idênticos, existem diferenças dramáticas em sua ordem (Figura 1.1). Estes estudos e muitos outros na mesma década provaram de forma convincente que os rearranjos de genomas é um modo comum de evolução molecular no DNA de cloroplastos, mitocondrial, viral, e bacteriano (1).

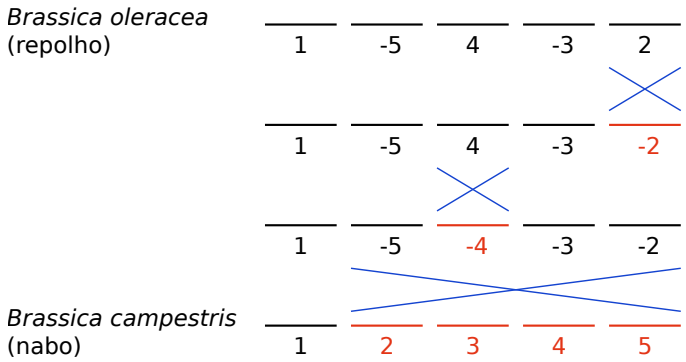

Figura 1.1: Transformação de *Brassica oleracea*(repolho) em *Brassica campestris*(nabo). (Dados tomados de (2))

Todo estudo de rearranjo de genomas envolve a solução de um problema combinatório que consiste em encontrar a menor sequência de rearranjos de genomas para transformar um genoma em outro (Na Figura 1.1 mostra-se as "reversões"necessárias para transformar o repolho em nabo). No caso de genomas que consistem de um número pequeno de blocos de genes, Palmer e co-autores conseguiram encontrar os cenários mais parcimoniosos para os rearranjos. No entanto, para genomas que consistem de um número maior a 10, uma busca exaustiva de todas as possíveis soluções está muito além das possibilidades de métodos que utilizem só "caneta e lápis".

A análise de rearranjos genômicos fornece uma série de desafios para os cientistas da computação (3). Uma abordagem computacional baseada na comparação das ordens dos genes versus uma comparação tradicional de sequências de DNA foi iniciada por Sankoff et al. (4, 5, 6). Kececioglu e Sankoff (7) introduziram o problema da distância de reversão e

estabeleceram os limites inferior e superior para a distância de reversão. Esta abordagem deu origem ao primeiro algoritmo de aproximação para o problema de ordenação por reversão gerando soluções quase ótimas, quer dizer perto da menor sequencia de reversões, para permutações aleatórias e permutações baseadas em dados biológicos.

1.0.1 Motivação

A comparação de sequências biológicas é um problema relevante em bioinformática para determinar as relações evolutivas entre organismos. Este problema pode ser resolvido usando algoritmos que levam em consideração mutações locais (deleções, inserções e substituições), tais como o algoritmo clássico de programação dinâmica para alinhar duas sequências de DNA. Mas quando se tenta entender como as sequências genéticas sofrem mutações ao nível cromossômico, é necessário considerar operações globais como reversões, troca de blocos, transposições, etc. (E entre essas operações globais a que mais ocorre no rearranjo de genomas é a *reversão* de uma subcadeia.)

A ordem dos genes de um genoma pode ser representada em notação de cadeia como uma permutação $\pi = (\pi_1, \pi_2, ..., \pi_n)$, que é uma função bijetiva do conjunto $\{1, .., n\}$ em si mesmo, onde n é o número de genes. Dois tipos diferentes de permutações têm recebido atenção do ponto de vista biológico: permutações com sinal e sem sinal. Em permutações sem sinal , os genes são abstraídos sem qualquer orientação, enquanto em permutações com sinal, cada π_i tem um sinal positivo ou negativo refletindo sua orientação dentro do genoma; ou da esquerda para direita ou da direita para esquerda, cuja notação respectiva é $\overrightarrow{\pi_i}$ e $\overleftarrow{\pi_i}$.

Dadas duas permutações, o número mínimo de reversões para transformar uma permutação em outra é conhecido como a *distância de reversão* entre duas permutações. Por propriedades algébricas de permutações sabemos que este problema é equivalente ao problema de determinar o mínimo número de reversões para transformar uma permutação na permutação identidade $\imath$. Este problema é conhecido como *ordenação por reversão*.

Os dois tipos de permutações dão origem a dois subproblemas de complexidade diferente para o problema de ordenação por reversão. No caso de permutações sem sinal, o problema foi demostrando ser $\mathcal{NP}$-difícil por Caprara (8). No caso de permutações com sinal, existe um algoritmo de tempo linear que dá soluções exatas (9).

No estudo do problema de ordenação de permutações por reversões uma estrutura muito usada é o grafo de reversões, que será explicado no Capítulo 2. Este grafo é construído a partir das relações de adjacência e pontos de quebra de uma permutação dada. A distância de reversão está relacionada com o número de ciclos deste grafo. Para permutações com sinal a decomposição em ciclos deste grafo é única. Isto não ocorre para permutações sem sinal. Esta observação é a base para compreender por que o problema de ordenação por reversões de permutações sem sinal é $\mathcal{NP}$-difícil, e usando permutações com sinal é polinomial. Uma questão, ainda em aberto, é se ordenar uma permutação sem sinal por reversões é ou não $\mathcal{NP}$-completo.

Devido a complexidade do problema para permutações sem sinal, foram propostos muitos algoritmos de aproximação, como o algoritmo de raio de aproximação 1.5 (10). Mas também foram propostos algoritmos evolutivos, como o algoritmo genético proposto por (11), que no seu trabalho relatou ter obtido melhores resultados do que o algoritmo de raio de aproximação 1.5. Muitos dos algoritmos genéticos propostos para este problema

((11), (12), (13)) não relatam como geram as permutações de entrada nos experimentos e desde que a maioria dos resultados para esta versão do problema só são aproximados, surge a necessidade de propor novos algoritmos ou de melhorar os já propostos, e também refazer alguns experimentos gerando de maneira correta as permutações de entrada.

1.0.2 Objetivos

O objetivo geral deste trabalho foi propor um algoritmo genético (AG) baseado em outras propostas de algoritmos de computação evolucionária, tanto como algoritmos de aproximação, para resolver o problema de ordenação por reversão de permutações sem sinal.

Como objetivos específicos temos:

- Realizar um estudo teórico e implementação do algoritmo de raio de aproximação 1.5;
- Realizar um estudo de outros AGs, tanto como algoritmos de aproximação, relacionados com o problema de ordenação por reversões, para propor e implementar um novo AG;
- Realizar experimentos utilizando como dados de entrada permutações geradas em diversas formas, de modo que possamos ter uma visão clara de quanto é a melhora sobre outros algoritmos;
- Comparar os resultados da nossa implementação do algoritmo de raio de aproximação 1.5, com os resultados de outros trabalhos que também implementaram o mesmo algoritmo;
- Comparar os resultados da nossa proposta de AGs com o algoritmo de raio de aproximação 1.5, tanto como com os resultados de outros trabalhos relacionados.

1.0.3 Organização do Trabalho

Este trabalho está organizado em capítulos. No Capítulo 2, são apresentadas as definições e terminologia usadas sobre permutações e o problema de ordenação por reversão, assim como noções sobre algoritmos genéticos. No Capítulo 3, é apresentada a revisão da literatura para o problema de ordenação por reversão com sinal, e a versão sem sinal. É apresentada, ainda, a correção feita sobre o algoritmo de raio de aproximação 1.5, uma análise do raio de aproximação, e detalhes da implementação. No Capítulo 4, é apresentada a nossa proposta por algoritmos genéticos, e também são mostrados alguns detalhes da implementação. No Capítulo 5, mostram-se os experimentos feitos tomando como entrada permutações geradas de diferentes formas; também é apresentada uma discussão sobre os resultados obtidos. Finalmente, no Capítulo 6, são apresentadas as conclusões sobre nosso trabalho e algumas sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Definições e Terminologia sobre Permutações sem Sinal

A maioria das definições e terminologia que serão apresentadas tem similitudes com aquelas que foram apresentadas tanto por Bafna e Pevzner (14), como por Kececioglu e Sankoff (7).

2.1.1 Permutações e Ordenação por Reversões

Definição 1. *Uma* ***permutação*** *no grupo simétrico S_n é uma bijeção π do conjunto $\{1, ..., n\}$ sobre ele mesmo.*

Uma permutação π, denotada em notação de cadeia como $\pi = (\pi_1, \pi_2, ..., \pi_n)$ é estendida adicionando um pivô inicial e final, $\pi_0 = 0$ e $\pi_{n+1} = n + 1$.

Definição 2. *Uma* ***reversão*** $\rho^{i..j}$ *de um intervalo $[i, j]$, para $1 \leq i \leq j \leq n$, transforma a permutação estendida π em*

$$\pi' = (\pi_0, ..., \pi_{i-1}, \pi_j, ..., \pi_i, \pi_{j+1}, ..., \pi_{n+1})$$

Por exemplo, considere a seguinte permutação

$$\pi = (0, 3, \mathbf{1, \ 5, \ 2, \ 4}, 6)$$

A reversão $\rho^{2..5}$ transforma π em

$$\pi = (0, 3, \mathbf{4, \ 2, \ 5, \ 1}, 6)$$

Considerando que o primeiro elemento da permutação estendia tem posição 0, pode-se notar que a reversão $\rho^{2..5}$ foi feita sobre o intervalo de posições $[2, 5]$ de π.

Observe que uma reversão também é uma permutação em S_n:

$$(\rho^{i..j})_k = \begin{cases} k, \text{ if } k < i \text{ or } k > j; \\ i + (j - k), \text{ if } i \leq k \leq j. \end{cases}$$

Assim, em notação posfixa $\pi\rho^{i..j}$ representa a aplicação da reversão $\rho^{i..j}$ à permutação π.

Definição 3. *Dadas duas permutações π e ρ, o* ***problema da distância de reversão*** *consiste em encontrar a menor sequência de reversões necessárias para transformar π em ρ. A* ***distância de reversão*** *entre π e ρ é o mínimo número de reversões necessárias para transformar π em ρ.*

Por propriedades algébricas simples de grupos de simetria, a distância de reversão entre π e ρ é igual à distância de reversão entre $\rho^{-1}\pi$ e a permutação identidade $\imath$, que é uma permutação ordenada. De fato, observe que, se $\rho_1 \dots \rho_k$, é uma sequência de reversões que transforma π em ρ, então se pode afirmar que $\pi\rho_1 \dots \rho_k = \sigma$, se e somente se $(\sigma^{-1}\pi)\rho_1 \dots \rho_k = \sigma^{-1}\sigma = \imath$.

Definição 4. *O problema de* ***ordenação por reversão*** *consiste em encontrar a distância de reversão $d(\pi)$ entre uma permutação π e a permutação identidade $\imath$.*

2.1.2 Grafo de Ciclos e Decomposição de Ciclos

Definição 5. *Denote por $i \sim j$ a propriedade $|i - j| = 1$. Dados dois elementos consecutivos π_i e π_j de π, para $0 < i < n + 1$ e, tanto $j = i - 1$ ou $j = i + 1$,*

- *disse-se que são* ***adjacentes*** *se $\pi_i \sim \pi_j$ e*
- *disse-se que formam um* ***ponto de quebra*** *se $\pi_i \nsim \pi_j$.*

Observe-se que a permutação identidade é única permutação sem pontos de quebra. O número de pontos de quebra em π é denotado por $b(\pi)$.

Definição 6. *Seja ρ uma reversão que transforma π em π'; é fácil conferir que $b(\pi) - b(\pi') \in \{-2, -1, 0, 1, 2\}$. As reversões que reduzem o número de pontos de quebra por i, são chamados de i-**reversões**.*

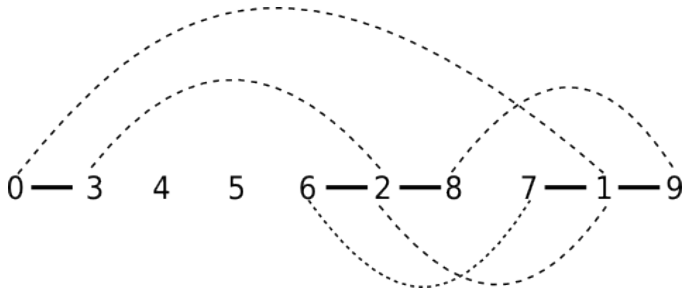

Figura 2.1: Grafo de ciclos $G(\pi)$ da permutação $\pi = (0, 3, 4, 5, 6, 2, 8, 7, 1, 9)$. Nesta figura as arestas pretas são representadas por linhas grossas pretas, e as arestas cinzas são representadas por linhas finas pontilhadas.

Definição 7. *Dada uma permutação π, define-se o* ***Grafo de Ciclos*** *(também conhecido como* ***Grafo de Pontos de Quebra****), $G(\pi)$ como um grafo não dirigido de arestas coloridas derivado das relações de adjacência e pontos de quebra de π,*

- *dois elementos i e j são unidos por uma* ***aresta preta*** *se (i, j) é um ponto de quebra em π, e*

- *dois elementos i e j são unidos por uma* ***aresta cinza*** *se $i \sim j$ e i, j não são consecutivos em π.*

Note-se que para qualquer permutação π, $G(\pi)$ pode ser completamente decomposta em ciclos disjuntos de arestas coloridas alternadas, já que cada nó tem um número igual de arestas incidentes pretas e cinzas. No entanto, existem provavelmente muitas *decomposições de ciclos* de arestas coloridas alternadas em $G(\pi)$. Por simplicidade, ciclos de arestas coloridas alternadas (pretas e cinzas) serão chamados tanto como ciclos alternados ou simplesmente ciclos. O grafo da Figura 2.1 pode ser decomposto em um o dois ciclos, na Figura 2.2 pode-se ver a decomposição em diferentes cores.

Definição 8. *Dada uma permutação π, e uma decomposição de ciclos $\mathcal{C}$. Define-se um* ***i-ciclo*** *como um ciclo de i arestas pretas ou de i arestas cinzas.*

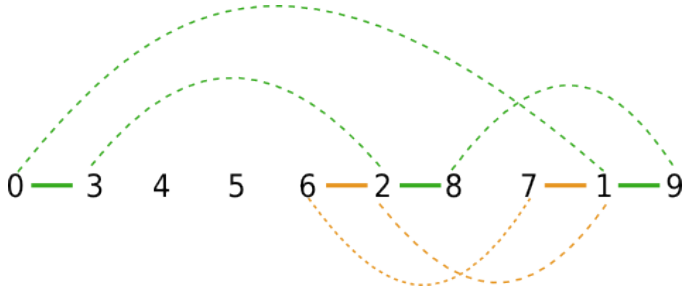

Figura 2.2: Decomposição de ciclos do grafo de ciclos $G(\pi)$ da permutação $\pi = (0, 3, 4, 5, 6, 2, 8, 7, 1, 9)$. Em verde o ciclo maior e em cor laranja um 2-ciclo.

O número máximo de ciclos numa decomposição de ciclos de $G(\pi)$, denotado como $c(\pi)$, proporciona um limite de grande utilidade para a distância de reversão (14): $d(\pi) \geq b(\pi) - c(\pi)$. A observação de que este limite é próximo da distância de reversão motivou o desenvolvimento de algoritmos de aproximação baseados na eliminação de pontos de quebra.

2.1.3 Grafo de Reversões

O algoritmo de raio de aproximação 1.5 utiliza o *grafo de reversões* para poder encontrar a sequência de reversões para ordenar uma permutação sem sinal.

Definição 9. *Dada uma permutação π, e uma decomposição de ciclos $\mathcal{C}$ em particular de um grafo de ciclos $G(\pi)$, define-se o* ***grafo de reversões*** *$R(\mathcal{C})$ correspondente à decomposição de ciclos $\mathcal{C}$ como uma grafo não dirigido, baseado nas seguintes relações:*

- *Por cada adjacência em π é introduzido um vértice azul isolado. Por cada aresta cinza que pertence a um ciclo da decomposição de ciclos $\mathcal{C}$ é introduzido um vértice, cuja cor será azul se a reversão que representa a aresta cinza não elimina nenhum ponto de quebra, caso contrario será de cor vermelho.*
- *Dois vértices do grafo de reversões são unidos por uma aresta, se as arestas cinzas que estes vértices representam no grafo de ciclos intercalam uma com a outra.*

Definição 10. *Dada uma permutação π, e uma decomposição de ciclos $\mathcal{C}$. Sejam (a, b) e (c, d) duas setas da esquerda para direita, que representam duas arestas pretas, respectivamente, relacionadas com uma aresta cinza u, então dizemos que:*

- *A aresta cinza u representa uma* ***reversão que não elimina nenhum ponto de quebra****, se conecta a cabeça com a cauda, ou a cauda com a cabeça das setas (a, b) e (c, d), ou seja, a aresta cinza u estaria formada pelos elementos (a, d) ou (b, c), como mostrado na Figura 2.3. A reversão deve ser feita sobre o intervalo das posições dos elementos b e c.*

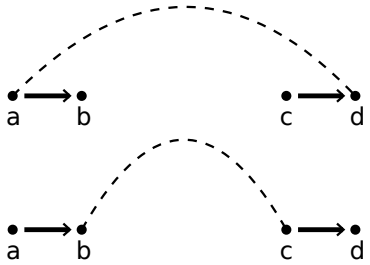

Figura 2.3: Arestas cinzas que representam reversões que não eliminam nenhum ponto de quebra

- *A aresta cinza u representa uma* ***reversão que elimina pelos menos um ponto de quebra****, se conecta a cauda com a cauda, ou a cabeça com a cabeça das setas (a, b) e (c, d), ou seja, a aresta cinza u estaria formada pelos elementos (a, c) ou (b, d), como mostrado na Figura 2.4. A reversão deve ser feita sobre o intervalo das posições dos elementos b e c.*

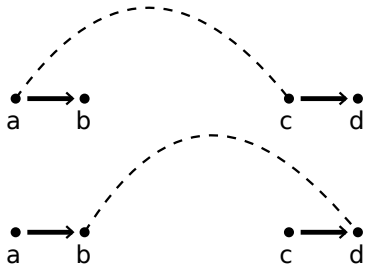

Figura 2.4: Arestas cinzas que representam reversões que eliminam pelo menos um ponto de quebra

Definição 11. *Dada uma permutação π, uma decomposição de ciclos $\mathcal{C}$ e considere u e v duas arestas cinzas em $G(\pi)$. Diz-se que u e v se* ***intercalam****,*

- *se as posições dos elementos das arestas cinzas também intercalam; quer dizer, sejam (a,b) e (c,d) as posições das arestas cinzas u e v, estas arestas intercalam se $a < c < b < d$;*
- *ou, se as posições mais a esquerda das arestas pretas na decomposição de ciclos $\mathcal{C}$ incidentes às arestas cinzas u e v intercalam; quer dizer, sejam (a', b') e (c', d') as posições mais à esquerda das arestas pretas relacionadas a u e v, estas arestas intercalam se $a' < c' < b' < d'$.*

Exemplo. Na Figura 2.5 pode-se ver que a aresta cinza (0,1), que conecta as arestas pretas (0,3) e (1,9), representa o vértice vermelho 0 no grafo de reversões, já que a aresta cinza (0,1) conecta a cauda da seta (0,3) com a cauda da seta (1,9) e portanto representa uma reversão que elimina pelo menos um ponto de quebra. Também, pode-se ver que a aresta cinza (3,2), que conecta as arestas pretas (0,3) e (2,8), representa o vértice azul 2 no grafo de reversões, já que a aresta cinza (3,2) conecta a cabeça da seta (0,3) com a cauda da seta (2,8) e portanto representa uma reversão que não elimina nenhum ponto de quebra.

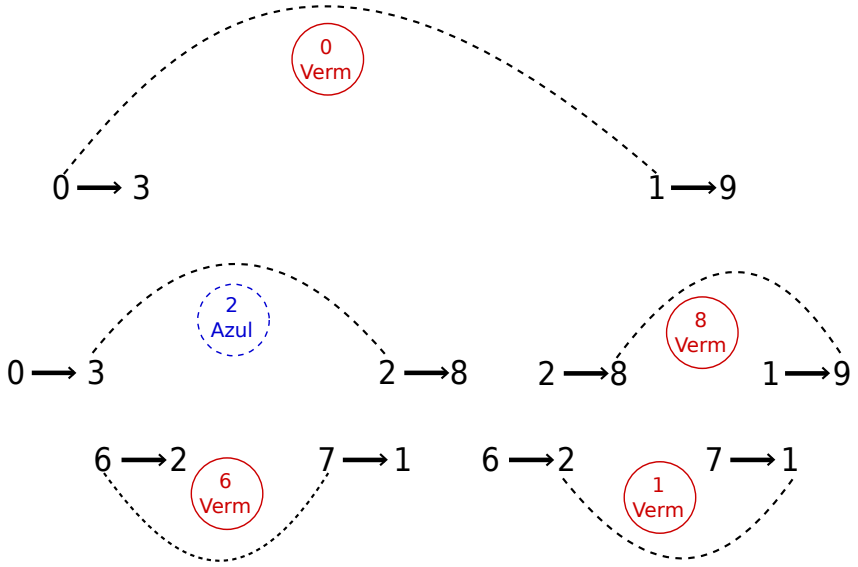

Figura 2.5: Exemplo de arestas cinzas do grafo de ciclos, e o respectivo vértice que representam no grafo de reversões.

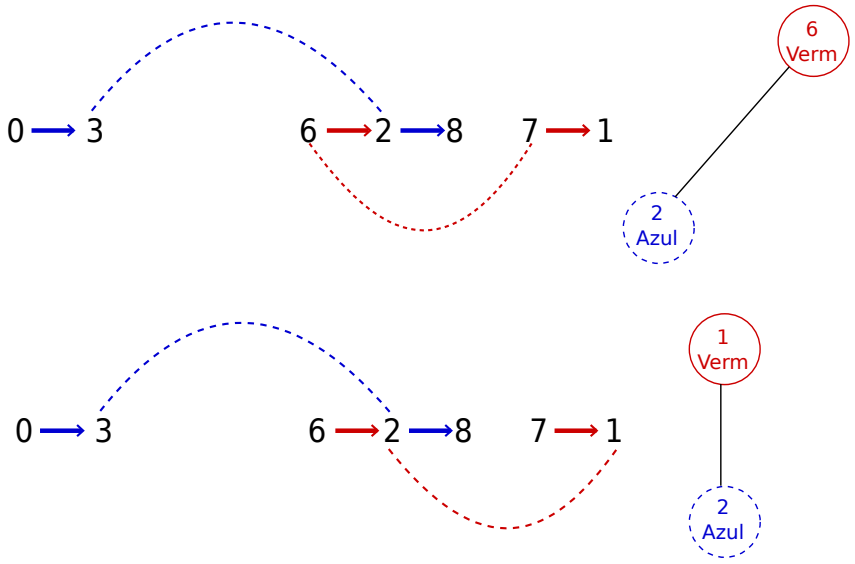

Figura 2.6: Exemplo de arestas cinzas intercaladas

Na Figura 2.6 pode-se ver que o vértice azul 2 e o vértice vermelho 6 estão unidos por uma aresta, já que as arestas cinzas que eles representam no grafo de ciclos intercalam

uma com outra, neste caso intercalam pelas posições dos elementos das arestas cinzas: (3,2) e (6,7) tal que $3 < 6 < 2 < 7$. Também, pode-se ver que o vértice azul 2 e o vértice vermelho 1 estão unidos por uma aresta, já que as arestas cinzas que eles representam no grafo de ciclos intercalam uma com outra; neste caso intercalam pelas posições mas à esquerda das arestas pretas relacionadas com as arestas cinzas: (0,2) e (6,7) tal que $0 < 6 < 2 < 7$.

A Figura 2.7 mostra o grafo de reversões $R(\mathcal{C})$ da decomposição de ciclos $\mathcal{C}$ do grafo de ciclos $G(\pi)$ da permutação $\pi = (0, 3, 4, 5, 6, 2, 8, 7, 1, 9)$. Note que, nesta figura para cada vértice representando a aresta cinza $\{i, i+1\}$ é dado o rótulo i.

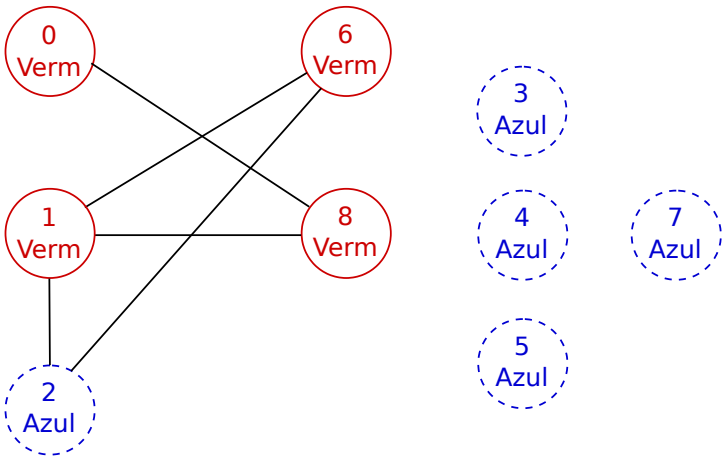

Figura 2.7: Exemplo de grafo de reversões

Definição 12. *Seja u um vértice do grafo de reversões $R(\mathcal{C})$ que surge de um ciclo da decomposição de ciclos $\mathcal{C}$. Denote por $\rho(u)$ a reversão representada por u. Então $\mathcal{C}_{\rho(u)}$ é definida como a decomposição de ciclos do grafo de ciclos da permutação obtida depois de aplicar $\rho(u)$. $\mathcal{C}_{\rho(u)}$ contém todos os mesmos ciclos de $\mathcal{C}$, exceto que um dos ciclos tem uma aresta preta a menos, ou que um 2-ciclo foi eliminado. O grafo de reversões obtido a partir de $\mathcal{C}_{\rho(u)}$ será denotado como $R_u(\mathcal{C})$.*

Definição 13. *Uma **componente** conexa de um grafo de reversões $R(\mathcal{C})$ é **orientada** se contém pelo menos um vértice vermelho, ou se consiste de um único vértice azul isolado. Caso contrário a componente é **não orientada**.*

Na Figura 2.7 pode-se ver que a componente conexa formada pelos vértices 0, 1, 2, 6, e 8 é orientada, já que contém vários vértices vermelhos. Também, pode-se ver que cada um dos vértices azuis 3, 4, 5, e 7 é uma componente conexa, pois são vértices azuis isolados. Uma componente não orientada seria uma componente conexa contendo só vértices azuis (pelo menos mais de um).

Definição 14. *Um **ciclo** C do grafo de ciclos $G(\pi)$ é **orientado** se contém uma aresta cinza que represente uma reversão que elimine pelo menos um ponto de quebra. Caso contrário, o ciclo é **não orientado**.*

Pela definição de reversões que eliminam pontos de quebra, e pela definição de ciclos orientados, o 2-ciclo (em cor laranja) da Figura 2.2, que contém a aresta (6,7) (que aparece na Figura 2.5), é um ciclo orientado.

2.2 Computação Evolucionária

O campo da computação evolucionária é uma comunidade de pessoas, ideias, e aplicações em constante desenvolvimento. Embora se possa traçar suas raízes genealógicas, já em 1930, foi o surgimento da tecnologia de computação digital barata na década de 1960 que tornou possível a utilização do computador como uma ferramenta para a análise de sistemas muito mais complexos do que aqueles matematicamente analisáveis (15).

Foi assim que vários grupos se tornaram cativados pela ideia de que até mesmo modelos evolutivos simples podem ser expressos em formas computacionais que poderiam ser usados para resolver problemas complexos baseados em computador. Destes, havia três grupos em particular, cujas atividades durante a década de 1960 serviram para definir e dar forma a este campo emergente.

Nos Estados Unidos, Fogel introduziu a Programação Evolucionária, enquanto que Holland chamou seu método de Algoritmo Genético. Na Alemanha Rechenberg e Schwefel formularam ideias que deram nascimento as Estratégias Evolucionárias. Até o inicio dos anos 90, essas áreas foram desenvolvidas separadamente e foi também que nessa década uma quarta corrente surgiu seguindo as mesmas ideias gerais, a Programação Genética. A terminologia contemporânea denota todo o campo como computação evolucionária e considera a programação evolucionária, estratégias evolucionárias, algoritmos genéticos, e programação genética como suas sub-áreas (15).

Em cada caso, estes algoritmos representam modelos ideais de processos evolutivos incorporados no contexto mais amplo de um paradigma de solução de problemas. A questão mais importante nesses algoritmos foi identificar e capturar os aspectos computacionais úteis de processos evolutivos.

Muitas das técnicas de computação evolucionária também são utilizadas por outros algoritmos de otimização, como os algoritmos baseados em inteligência coletiva (Swarm intelligence): otimização por colônia de formigas(ACO), otimização por enxame de partículas (PSO), colônia artificial de abelhas (ABC), etc.

2.3 Algoritmos Genéticos

Muitos dos conceitos e terminologias que serão apresentados nesta seção foram tomados do livro "*Introduction to Genetic Algorithms*"(16).

2.3.1 O que são os Algoritmos Genéticos?

Charles Darwin estabeleceu a teoria da evolução natural na origem das espécies, na qual ao longo de várias gerações os organismos biológicos evoluíram baseados no princípio de seleção natural, "a sobrevivência do mais apto". Portanto, se a evolução funciona tão bem na natureza, então deve ser interessante simular a evolução natural e desenvolver um método que resolva problemas de busca e otimização (16).

Em 1975, Holland desenvolveu essa ideia em seu livro "*Adaptation in natural and artificial systems*". Ele descreveu como aplicar os princípios da evolução natural para problemas de otimização e construiu os primeiros Algoritmos Genéticos. A teoria de Holland tem sido amplamente desenvolvida e agora os algoritmos genéticos (AG) sur-

gem como uma ferramenta poderosa para resolver problemas de busca e otimização. Os Algoritmos genéticos estão baseados nos princípios da genética e evolução (16).

A ideia básica é a seguinte: a informação genética de uma determinada população contém potencialmente a solução, ou a solução ótima, para um determinado problema. Esta solução não fica em um determinado indivíduo, já que a combinação genética em que se baseia está espalhada entre vários indivíduos. Somente a associação de diferentes genomas podem levar à solução, ou à solução ótima.

2.3.2 Terminologia Básica

É apresentada a terminologia básica relacionada com os algoritmos genéticos.

- **Indivíduos**

  Um indivíduo é uma única solução. Cada indivíduo está representado como um cromossomo, que está subdividido em genes. Um cromossomo deve, de alguma forma conter informações sobre a solução que ele representa.

- **Genes**

  Os genes são os componentes básicos para construir os algoritmos genéticos. Um cromossomo é uma sequência de genes. Os genes podem descrever uma possível solução para um problema sem ser realmente a solução. Um gene é uma cadeia de comprimento arbitrário.

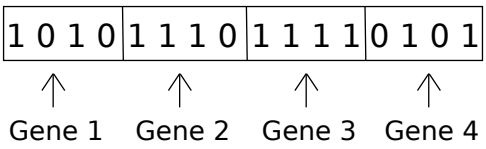

Figura 2.8: Representação de um cromossomo e seus genes.

- **Aptidão**

  A aptidão (*fitness*) de um indivíduo ou função de aptidão é usada para avaliar o cromossomo para posterior uso pelos operadores de reprodução. A função de aptidão possui como entrada uma cadeia e como saída um valor real. Deve-se notar que é necessário dispor de algum modo para avaliar quão distante um cromossomo está da solução ideal, o que nem sempre é uma tarefa trivial.

- **População**

  A população é um conjunto de indivíduos que estão sendo testados pelo algoritmo genético. Temos dois aspectos importantes da população em algoritmos genéticos: a geração da população inicial, e o tamanho da população. O tamanho da população dependerá da complexidade do problema. A geração da população inicial muitas vezes é feita de forma aleatória, mas pode haver casos em que a inicialização seja realizada com algumas soluções de boa qualidade.

População	Cromossomo 1	1 1 1 0 0 0 1 0
	Cromossomo 2	1 0 1 0 1 0 1 1
	Cromossomo 3	0 1 0 0 1 1 1 0
	Cromossomo 4	0 0 1 0 0 1 0 0

Figura 2.9: Representação de uma população.

2.3.3 Reprodução

O processo de reprodução é a parte central dos algoritmos genéticos, uma vez que é durante este processo que se busca criar novos indivíduos mais aptos. O ciclo de reprodução consiste em três etapas: seleção de indivíduos, recombinação (*crossover*) dos indivíduos selecionados para criar novos indivíduos, e substituição de indivíduos velhos na população por novos indivíduos.

- **Seleção**

  Seleção é o processo de escolha de dois indivíduos (pais) da população para fazer a recombinação. O objetivo da seleção é ressaltar indivíduos mais aptos da população na esperança de que seus descendentes tenham maior aptidão. Os cromossomos são selecionados a partir da população inicial para serem os pais na reprodução. Por analogia com a teoria da evolução de Darwin o melhor sobrevive para criar uma nova descendência.

- **Recombinação** (*Crossover*)

  A recombinação é o processo de tomar dois indivíduos (pais), que por sua vez representam duas soluções, para produzir a partir deles uma descendência. O operador de recombinação é aplicado sobre um conjunto de indivíduos (soluções) com a esperança de que criará uma melhor descendência. A recombinação é feita da seguinte forma: uma vez que se tem as soluções pais, escolhe-se aleatoriamente as posições onde haverá troca de genes, depois é feita a troca dos genes entre as duas soluções.

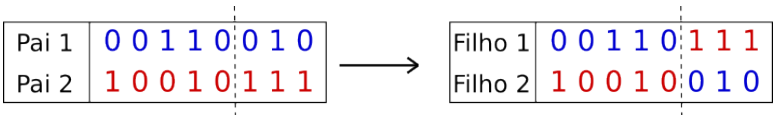

Figura 2.10: Exemplo de recombinação de ponto único.

- **Mutação**

  Depois da recombinação, alguns dos indivíduos são submetidos a mutação. A mutação desempenha o papel de recuperar o material genético perdido, e também de perturbar aleatoriamente a informação genética. Se a recombinação serve para explorar as soluções atuais a fim de encontrar outras melhores soluções, a mutação serve para ajudar na exploração do espaço de busca. Então a mutação é vista como um operador para manter a diversidade genética na população.

- **Substituição**

Pai	1 0 1 1 0 0 1 0
Filho	1 1 1 0 0 1 1 0

Figura 2.11: Exemplo de mutação sobre os genes de um cromossomo.

A substituição é o último passo do ciclo de reprodução. Depois que dois indivíduos (pais) são escolhidos de uma população de tamanho fixo e que eles produzem dois novos indivíduos (descendência), eles devem voltar a população. Porém, nem todos podem voltar, mas apenas dois; então deve-se escolher que membros da população atual devem ser substituídos pelas novas soluções.

2.3.4 Algoritmo Genético Simples

Em (16) apresenta-se uma classificação dos algoritmos genéticos como AG Paralelo, AG Messy, AG Distribuído, etc. Para o problema de ordenação por reversão foi tomado como base o Algoritmo Genético Simples que será descrito a seguir.

O procedimento padrão para executar o algoritmo é o seguinte:

1. Gerar aleatoriamente a população inicial;
2. Selecionar os indivíduos(pais), usando a função de aptidão;
3. Aplicar a recombinação sobre esses indivíduos, gerando uma descendência;
4. Aplicar a mutação sobre a descendência;
5. Colocar a descendência de volta na população;
6. Se uma solução melhor for encontrada, ou o número de gerações se esgotou, retornar o melhor indivíduo. Caso contrário voltar ao item (2).

Se tudo ocorrer bem, a simulação do processo evolutivo produzirá, a medida que as gerações forem se sucedendo, indivíduos(cromossomos) cada vez mais bem adaptados. Isto é, com melhor valor da função de aptidão, de modo que, no final obtém-se uma solução (cromossomo) com alto grau de adequação ao problema proposto.

Capítulo 3

Contextualização

3.1 Ordenação por Reversões de Permutações com Sinal

O problema de ordenação por reversões tem sido extensivamente estudado, tanto no campo da combinatória de permutações como na bioinformática por décadas. No caso de ordenação por reversão de permutações com sinal, inicialmente, Kececioglu e Sankoff (7) conjecturaram que o problema era $\mathcal{NP}$-difícil e propuseram um algoritmo de raio de aproximação 2, aproveitando a ligação entre a distância de reversão e o número de pontos de quebra. Depois, Bafna e Pevzner (14) melhoraram o raio de aproximação para 1.5, utilizando a estrutura de dados de grafo de pontos de quebra. Logo, Hannenhalli e Pevzner (2) deram um algoritmo de tempo polinomial exato $O(n^2)$ para o cálculo da distância de reversão ($O(n^4)$ quando a sequência mínima de reversões é fornecida) utilizando a estrutura de dados de grafo de sobreposição. Muito depois, Bergeron (17) apresentou um tratamento elementar do algoritmo. Finalmente, algoritmos mais eficientes baseados neste algoritmo polinomial exato (2) foram introduzidos; entre eles, o algoritmo de tempo $O(n\alpha(n))$ proposto por Berman e Hannenhalli (18), onde $\alpha(n)$ é a função inversa de Ackermann e, o algoritmo de tempo linear proposto por Bader, Moret e Yang (9), que utiliza uma nova estrutura de dados chamada de floresta de sobreposição. Incrementando a complexidade destes algoritmos exatos, se pode calcular não só a distância de reversão, mas também construir a sequência mínima de reversões.

Análise da relação que originou o algoritmo polinomial

O fato de saber que uma reversão elimina no máximo 2 pontos de quebra, implica a relação $d(\pi) \geq b(\pi)/2$; mas fazer o cálculo da distância de reversão usando esta relação da resultados muito imprecisos. Bafna e Pevzner (14) introduziram a noção de grafo de ciclos (ou grafo de pontos de quebra) de uma permutação, desse modo, proporcionando a estrutura básica para o cálculo da distância de reversão. Baseado neste grafo estabeleceram o vínculo entre os pontos de quebra e o ciclo de decomposição maximal deste grafo, que é a relação $d(\pi) \geq b(\pi) - c(\pi)$, a qual dá resultados mais precisos, especialmente quando se tratam permutações com sinal. Hannenhalli e Pevzner (2) desenvolveram uma nova relação para permutações com sinal, $b(\pi) - c(\pi) + h(\pi) \leq d(\pi) \leq b(\pi) - c(\pi) + h(\pi) + 1$, onde $h(\pi)$ é o número de *hurdles* que indica se uma permutação é difícil de ordenar. Baseada

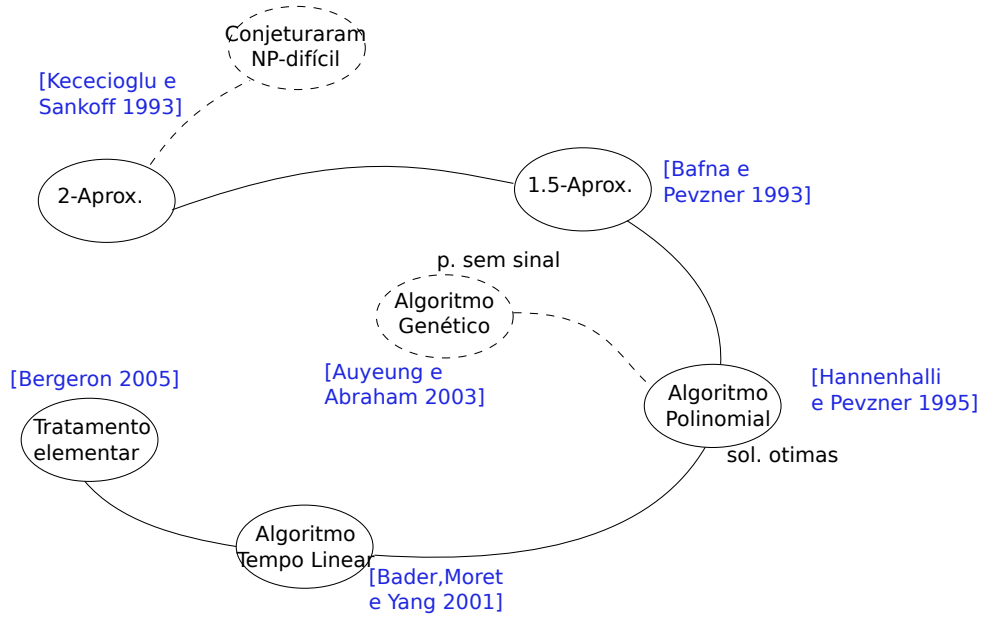

Figura 3.1: Revisão da literatura para o caso de permutações com sinal.

nesta última relação foi proposto o primeiro algoritmo de tempo polinomial para resolver o problema de ordenação por reversões de permutações com sinal.

3.2 Ordenação por Reversões de Permutações sem Sinal

No caso de permutações sem sinal, as que serão tratadas neste trabalho, o problema foi demonstrado ser $\mathcal{NP}$-difícil por Caprara (8). Antes que a complexidade fosse conhecida, Kececioglu e Sankoff (7) deram um algoritmo de raio de aproximação 2, e Bafna e Pevner (14) apresentaram um algoritmo de raio de aproximação 1.75. Mais tarde, o raio de aproximação foi melhorado para 1.5 por Christie (10), e depois para 1.375 por Berman, Hannenhalli e Karpinski (19). Este último algoritmo de aproximação é só de interesse teórico sendo a sua implementação prática de grande dificuldade.

Técnicas de computação evolucionária como os algoritmos genéticos foram propostas para lidar com o caso de permutações sem sinal, devido à complexidade do problema. Auyeung e Abraham (11) sugeriram uma abordagem de algoritmos genéticos (AG) para resolver o problema de ordenação por reversão de permutações sem sinal baseado no mapeamento de uma permutação sem sinal de tamanho n, em um subconjunto de 2^n possíveis permutações com sinal. Para uma permutação sem sinal dada, o mapeamento é feito gerando um conjunto de permutações com sinal atribuindo aleatoriamente um sinal positivo ou negativo a cada elemento da permutação sem sinal. É assim que a solução exata de uma destas permutações com sinal corresponde-se com uma solução viável da permutação original sem sinal. A função de aptidão de cada permutação com sinal é dada por sua distância de reversão que é calculada pelo algoritmo de tempo polinomial de Hannenhalli et al.

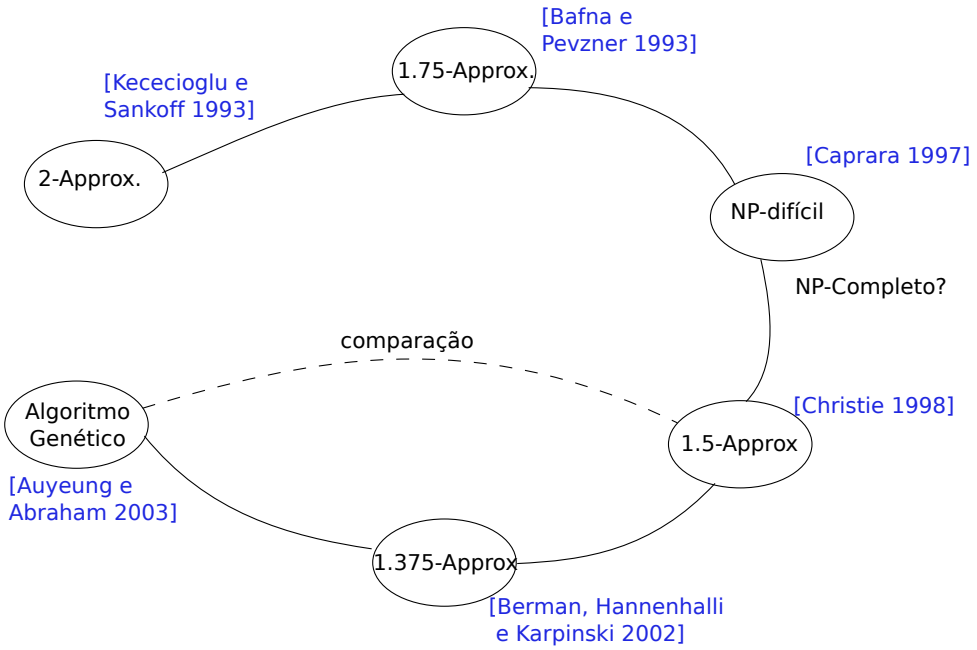

Figura 3.2: Revisão da literatura para o caso de permutações sem sinal.

Posteriormente, modificações ao método de Auyeung e Abraham foram relatadas em (13) sem alterar as premissas centrais desta abordagem. Mais recentemente, Ghaffarizadeh, Ahmadi e Flann (12) propuseram uma versão modificada do AG padrão utilizando indivíduos de tamanhos diferentes para reduzir o tempo de execução do algoritmo. Todas estas abordagens relataram melhorar os resultados obtidos pelo algoritmo de raio de aproximação 1.5 de Christie, o qual foi aplicado, a fim de controlar a precisão das soluções. Mas dois problemas surgem nestes artigos: em primeiro lugar, o algoritmo de aproximação de Christie apresenta algumas imprecisões na atualização do grafo de reversões, cuja solução não foi relatada em (11), (13) ou em (12), em segundo lugar, em muitos desses artigos não fica claro como é que as permutações foram geradas aleatoriamente.

3.3 Algoritmo de Raio de Aproximação 1.5 Corrigido

O algoritmo de raio de aproximação 1.5, que foi proposto por Christie (10), usa o fato de que qualquer decomposição de ciclos do grafo de ciclos (grafo de pontos de quebra) que maximiza o número de 2-ciclos, pode atingir uma sequência de ordenação de comprimento no máximo 1.5 da sequência de ordenação ideal.

3.3.1 Problemas encontrados e Correção

Sabemos que o grafo de reversões é importante na determinação da sequência de reversões para ordenar uma permutação. O Lema 1, proposto por Christie(10), está relacionado com a atualização do grafo de reversões depois de aplicar a reversão associada com um vértice deste grafo. Durante a implementação do algoritmo de raio de aproximação 1.5 foi encontrado um contraexemplo para o bom funcionamento da atualização do grafo de

reversões proposto pelo Lema 1. Então, foi proposto o Lema 2 corrigindo as imprecisões do Lema 1.

Lema 1 (Christie(10))**.** *Dada uma permutação π, e uma decomposição de ciclos $\mathcal{C}$ do grafo de ciclos $G(\pi)$. Seja u um vértice do grafo de reversões $R(\mathcal{C})$ que surge de um ciclo de $\mathcal{C}$. Então $R_u(\mathcal{C})$, que é o grafo de reversões resultante depois de aplicar a reversão correspondente ao vértice u, pode ser obtido a partir de $R(\mathcal{C})$ considerando as seguintes mudanças:*

(i) A cor de um vértice v é trocada se, e somente se, $\{u, v\}$ é uma aresta em $R(\mathcal{C})$.

(ii) Para todos os pares de vértices v e w em $R(\mathcal{C})$, tal que (u,v) e (u,w) são arestas em $R(\mathcal{C})$, então (v,w) é uma aresta em $R_u(\mathcal{C})$ se, e somente se, (v,w) não é uma aresta em $R(\mathcal{C})$.

(iii) Se u é um vértice vermelho então este se torna um vértice isolado azul, caso contrário não ocorrem mudanças neste vértice.

Lema 2 (Lema 1 modificado)**.** *Dada uma permutação π, e uma decomposição de ciclos $\mathcal{C}$ do grafo de ciclos $G(\pi)$. Seja u um vértice do grafo de reversões $R(\mathcal{C})$ que surge de um ciclo de $\mathcal{C}$. Então $R_u(\mathcal{C})$, que é o grafo de reversões resultante depois de aplicar a reversão correspondente ao vértice u, pode ser obtido a partir de $R(\mathcal{C})$ considerando as seguintes mudanças:*

(i) A cor de um vértice v, $v \neq u$, é trocada se, e somente se $\{u, v\}$ é uma aresta em R(C).

(ii) Para todos os pares de vértices v e w em R(C), tal que (u,v) e (u,w) são arestas em R(C), então (v,w) é uma aresta em $R_u(C)$ se, e somente se, (v,w) não é uma aresta em R(C) e ***as arestas que representam os vértices v e w, não compartilham exatamente um vértice e uma aresta preta no grafo de ciclos****.*

(iii) Se u é um vértice vermelho então este se torna um vértice isolado azul, caso contrário não ocorrem mudanças neste vértice.

Demonstração. A prova é por análise de casos.

(i) Suponha que o vértice v está separado do vértice u. A cor de v se preserva sempre que ao aplicar a reversão correspondente ao vértice u no grafo de ciclos da permutação obtida a orientação das arestas pretas incidentes a aresta cinza associada ao vértice v se preserva:

- caso v esteja totalmente isolado de u a reversão associada a u não afeta a aresta v;
- caso v esteja no interior de u ao aplicar a reversão, em cada extremo da aresta v caudas e cabeças das arestas pretas intercambiam-se.

Caso exista uma aresta entre u e v as arestas cinzas associadas estão intercaladas, logo, ao aplicar a reversão associada a u, a orientação de uma das arestas pretas nos extremos da aresta cinza associada com v muda, e dessa forma a cor mudará em $R_u(C)$.

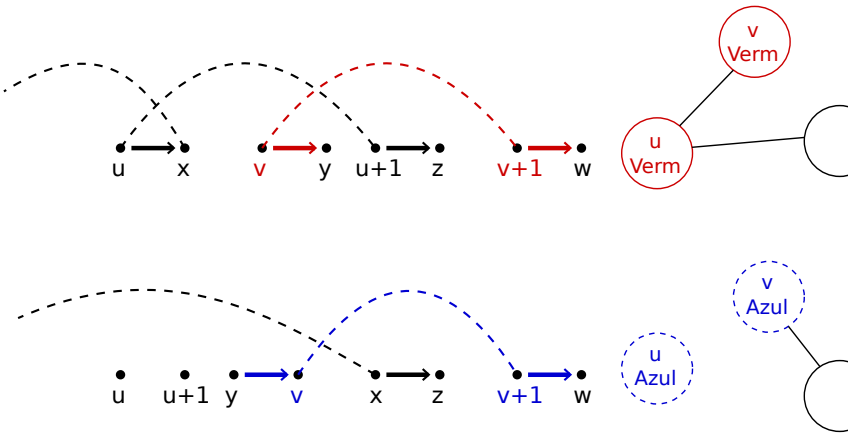

Figura 3.3: Parte (i) da demostração do Lema 2. No caso em que as arestas cinzas associadas aos vértices u e v estão intercaladas.

(ii)

- Suponha que u é um vértice vermelho que forma arestas com os vértices v e w, e que v e w não formam nenhuma aresta. Então, ao aplicar a reversão associada ao vértice u no grafo de ciclos, as arestas associadas aos vértices v e w estarão intercaladas, e dessa forma o vértices v e w formarão uma aresta em $R_u(C)$. O mesmo acontece para o caso em que u é um vértice azul.

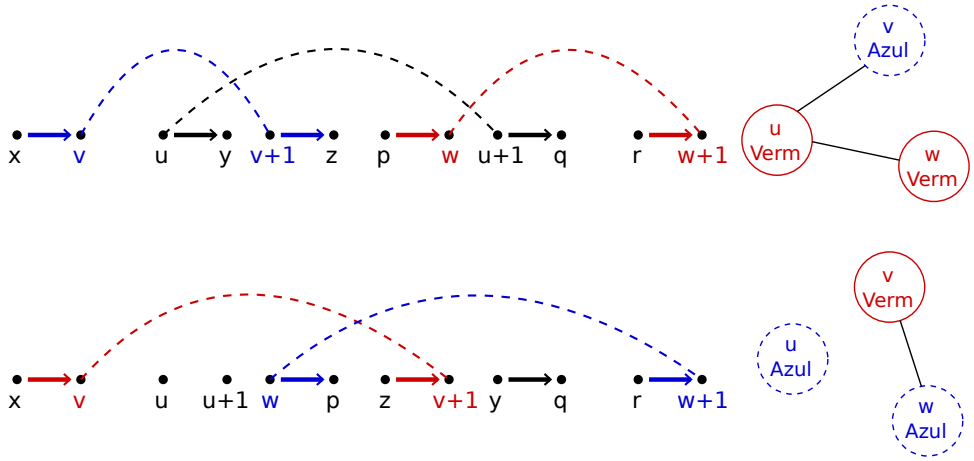

Figura 3.4: Parte (ii) da demonstração do Lema 2. No caso em que u é um vértice vermelho, e os vértices v e w não formam nenhuma aresta.

- Suponha que u é um vértice vermelho que forma arestas com os vértices v e w, e que v e w formam uma aresta. Então, ao aplicar a reversão associada ao vértice u no grafo de ciclos, as arestas associadas aos vértices v e w deixam de estar intercalas, e dessa forma os vértices v e w não formarão nenhuma aresta em $R_u(C)$. O mesmo acontece para o caso em que u é um vértice azul.
- Suponha que u é um vértice vermelho que forma arestas com os vértices v e w, e que as arestas associadas a v e w compartilham uma aresta preta e um ponto em comum. Então, ao aplicar a reversão associada ao vértice u no grafo de ciclos, as arestas associadas aos vértices v e w ficam compartilhando a mesma

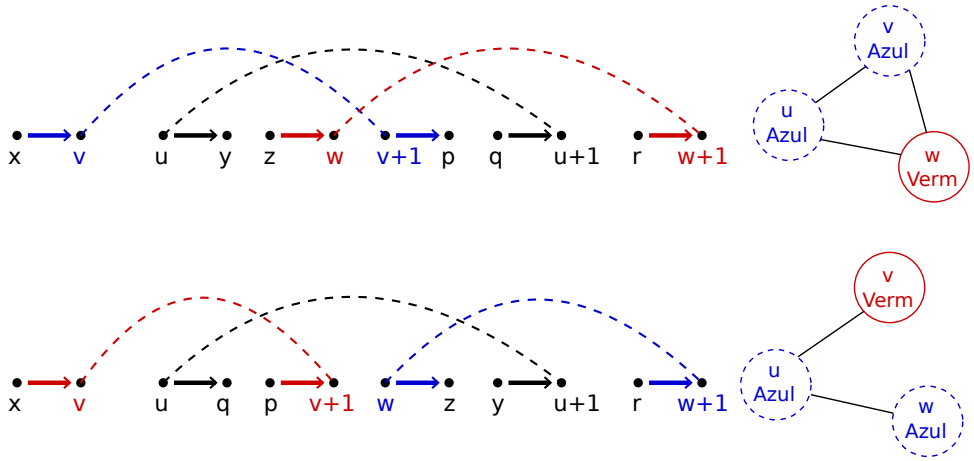

Figura 3.5: Parte (*ii*) da demonstração do Lema 2. No caso em que u é um vértice azul, e os vértices v e w formam uma aresta.

aresta preta e um ponto em comum, mudando só a orientação das arestas pretas adjacentes ao ponto em comum, desta forma a relação entre v e w não muda. O mesmo acontece para o caso em que u é um vértice azul.

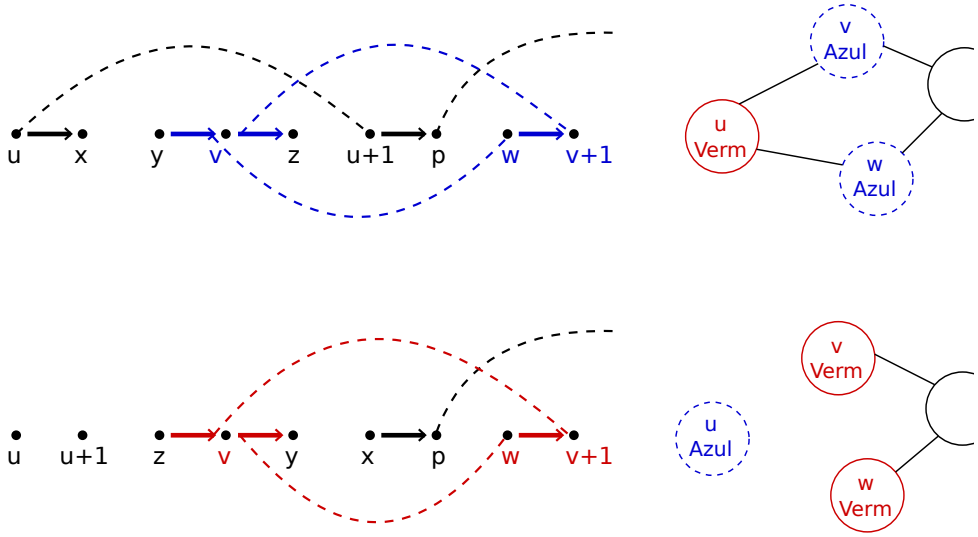

Figura 3.6: Parte (*ii*) da demonstração do Lema 2. No caso em que u é um vértice vermelho, e que as arestas associadas a v e w compartilham no grafo de ciclos uma aresta preta e um ponto em comum.

- Suponha que u é um vértice vermelho que não forma nenhuma aresta com v e w; ao aplicar a reversão associada a u no grafo de ciclos, nenhuma das arestas cinzas associadas a v e w são afetadas, desta forma a relação entre v e w não muda. O mesmo acontece para o caso em que u é um vértice azul.

(iii) Suponha que u é vermelho. Então, ao aplicar a reversão associada a u elimina-se no mínimo um ponto de quebra e a aresta cinza desaparecerá sempre que os seus extremos passam a formar uma adjacência, que é representada por um vértice azul

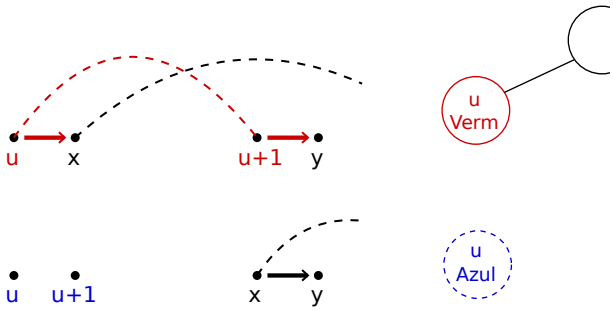

Figura 3.7: Parte (*iii*) da demonstração do lema 2. No caso em que u é um vértice vermelho.

isolado. Caso u é azul, ao aplicar a reversão associada a u não se elimina nenhum ponto de quebra, e a aresta cinza se preserva, desse modo a cor do vértice u se preserva.

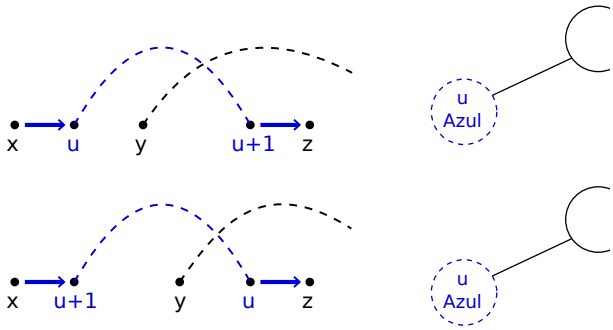

Figura 3.8: Parte (*iii*) da demonstração do Lema 2. No caso em que u é um vértice azul, e a aresta aresta cinza associada a u tem suas arestas pretas adjacentes por fora da aresta.

□

Exemplo. Considere a permutação $\pi = (0, 7, 2, 1, 3, 4, 5, 6, 8)$, cujo grafo de ciclos e grafo de reversões é mostrado na Figura 3.10. Suponhamos que a reversão representada pelo vértice vermelho 0 é aplicada sobre a permutação π, modificando também seu grafo de ciclos. Logo, atualizamos o grafo de reversões como proposto pelo Lema 1. O grafo de ciclos e o grafo de reversões resultante é apresentado na Figura 3.11. A informação contida nos dois grafos deveria ser a mesma, já que sempre é possível criar um grafo de reversões a partir do grafo de ciclos.

Pode-se observar que no grafo de reversões da Figura 3.11 existe uma ligação entre os vértices 6 e 7, quer dizer, que as arestas associadas a estes vértices deveriam intercalar. Mas, pode-se ver que estas duas arestas $(6, 7)$ e $(7, 8)$, respectivamente, pela definição de aresta cinzas que intercalam, não estão intercaladas.

Então, o Lema 1 falha ao determinar como o grafo de reversões deveria ser atualizado. Em geral, este tipo de contraexemplo só é possível se somente se duas arestas cinzas compartilham o mesmo vértice e a mesma aresta preta, como é o caso das arestas cinzas $(6, 7)$ e $(7, 8)$ mostradas no grafo de ciclos da Figura 3.11.

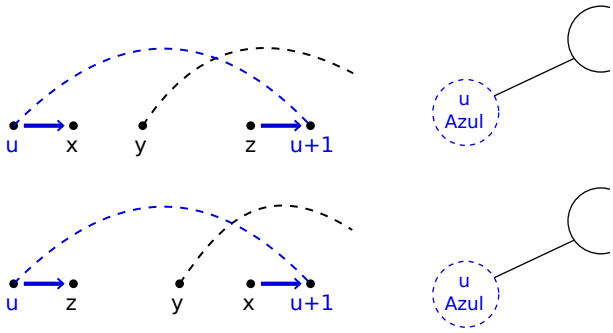

Figura 3.9: Parte (*iii*) da demonstração do Lema 2. No caso em que u é um vértice azul, e a aresta aresta cinza associada a u tem suas arestas pretas adjacentes por dentro da aresta.

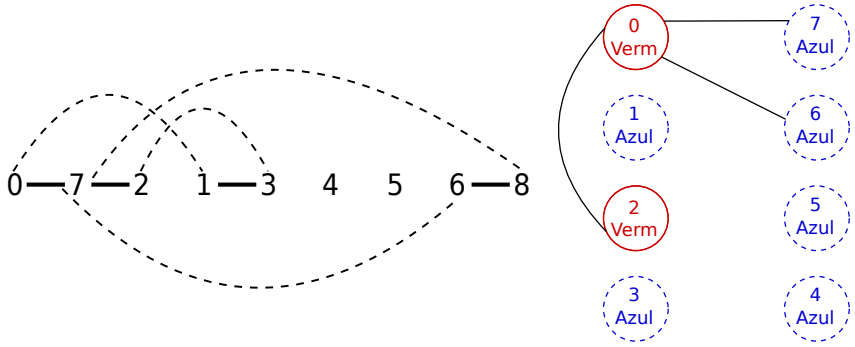

Figura 3.10: Grafo de ciclos, e grafo de reversões do contraexemplo para o Lema 1.

Agora, se tivéssemos atualizado o grafo de reversões da Figura 3.10 usando o Lema 2, teríamos gerado o grafo de reversões correto, como mostrado na Figura 3.12, onde os vértices 6 e 7 não tem mais nenhuma ligação, conforme a definição de grafo de reversões.

3.4 Demonstração do Raio de Aproximação 1.5

O raio de aproximação pode ser atingido comparando o número de reversões obtidos pelo algoritmo de aproximação, com um limite inferior teórico.

Considere que o número de reversões obtidos pelo algoritmo de aproximação é no máximo $b(\pi) - \frac{1}{2}c_2(\pi)$, e que o limite inferior teórico é no mínimo $\frac{2}{3}b(\pi) - \frac{1}{3}c_2(\pi)$. Então, se estas duas equações são divididas obteremos o raio de aproximação 1.5.

$$\frac{b(\pi) - \frac{1}{2}c_2(\pi)}{\frac{2}{3}b(\pi) - \frac{1}{3}c_2(\pi)} = \frac{3}{2} = 1.5$$

3.4.1 O Limite Inferior Teórico

Bafna e Pevzner (14) mostraram o limite inferior fundamental, como mostrado a seguir.

Seja $G(\pi)$ um grafo de ciclos de uma permutação π, e $b = b(\pi)$ o número de pontos de quebra em $G(\pi)$.

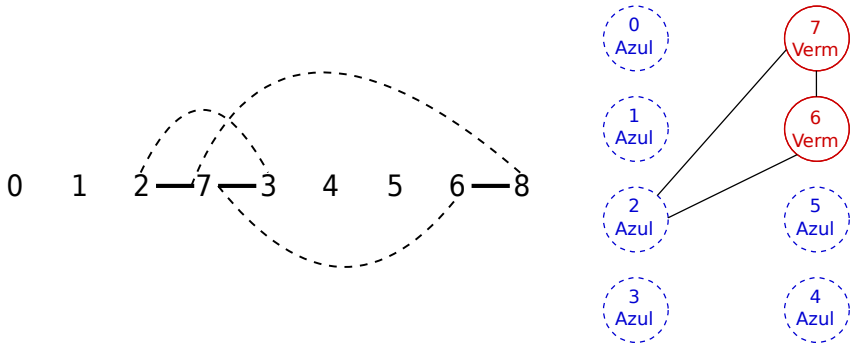

Figura 3.11: Grafo de ciclos, e grafo de reversões resultante depois de usar o Lema 1.

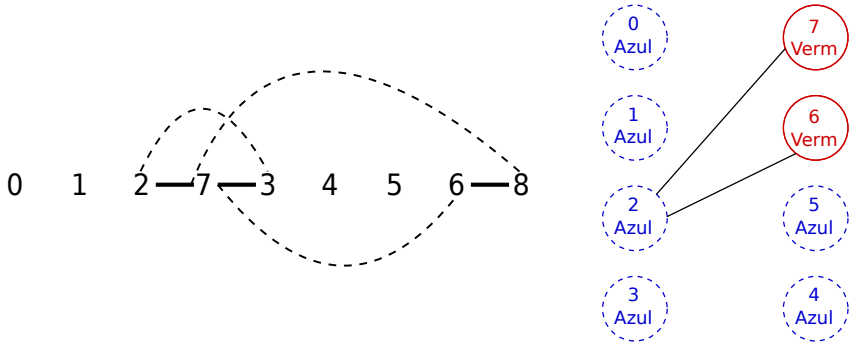

Figura 3.12: Grafo de ciclos, e grafo de reversões resultante depois de usar o Lema 2.

Seja ρ uma reversão qualquer, onde $\pi\rho$ representa a aplicação da reversão ρ sobre a permutação π. Denote como $G' = G(\pi\rho)$, $b' = b(\pi\rho)$, o número de pontos de quebra em G', e $c' = c(\pi\rho)$, o número de ciclos numa decomposição de ciclos máxima de G'.

Denote por $\Delta b = \Delta b(\pi, \rho) = b - b'$ à diminuição de pontos de quebra, e por $\Delta c = \Delta c(\pi, \rho) = c' - c$ ao aumento do número de ciclos numa decomposição maximal.

Então temos que para os 5 valores potenciais de Δb (-2,-1,0,1,2) é verdade que:

$$\Delta b(\pi, \rho) + \Delta c(\pi, \rho) \leq 1 \tag{3.1}$$

Seja $\rho_t, ..., \rho_1$ a menor sequência de reversões para transformar $\pi = \pi_t$ na permutação identidade π_0. Denote $\pi_{i-1} = \pi_i\rho_i$ para $i = 1, ..., t$, e aplique a equação (3.1), para π_i e ρ_i, sobre a seguinte equação, onde $d(\pi_i)$ é distância de reversão de π_i .

$$\begin{aligned} d(\pi_i) &= d(\pi_{i-1}) + 1 \\ &\geq d(\pi_{i-1}) + \Delta b(\pi_i, \rho_i) + \Delta c(\pi_i, \rho_i) \\ &= d(\pi_{i-1}) + (b(\pi_i) - b(\pi_i\rho_i)) + (c(\pi_i\rho_i) - c(\pi_i)) \end{aligned}$$

Note que $d(\pi_0) = b(\pi_0) = c(\pi_0) = 0$, e que

$$\begin{aligned} d(\pi_i) &\geq d(\pi_{i-1}) + (b(\pi_i) - b(\pi_i\rho_i)) + (c(\pi_i\rho_i) - c(\pi_i)) \\ d(\pi_i) - (b(\pi_i) - c(\pi_i)) &\geq d(\pi_{i-1}) - (b(\pi_{i-1}) - c(\pi_{i-1})) \\ &\geq d(\pi_{i-2}) - (b(\pi_{i-2}) - c(\pi_{i-2})) \\ &\vdots \\ &\geq d(\pi_0) - (b(\pi_0) - c(\pi_0)) \\ &= 0 \end{aligned}$$

Então:

$$\begin{aligned} d(\pi_i) - (b(\pi_i) - c(\pi_i)) &\geq 0 \\ d(\pi_i) &\geq b(\pi_i) - c(\pi_i) \end{aligned}$$

Substituindo i por t e tendo que $\pi = \pi_t$, temos o limite inferior fundamental:

$$d(\pi) \geq b(\pi) - c(\pi) \tag{3.2}$$

Para encontrar um novo limite inferior teórico, se fez a seguinte análise como feito por Christie (10).

Considere uma decomposição cíclica máxima $\mathcal{C}$, com um número mínimo de 2-ciclos denotado por $c_2(\pi)$. Suponha também que $c_{>2}(\pi)$ é o número de ciclos de comprimento maior que dois em $\mathcal{C}$. Então, substituindo $c(\pi)$ por $c_2(\pi) + c_{>2}(\pi)$ em (3.2) temos que

$$d(\pi) \geq b(\pi) - c_2(\pi) - c_{>2}(\pi) \tag{3.3}$$

Cada ciclo em $\mathcal{C}$, que não seja um 2-ciclo, possui pelo menos três arestas pretas. O número de arestas pretas restantes sem contar aquelas que formam parte de 2-ciclos é $b(\pi) - 2c_2(\pi)$. Então, o número de ciclos que podemos construir com três arestas pretas é $\frac{1}{3}(b(\pi) - 2c_2(\pi))$. Logo,

$$c_{>2}(\pi) \leq \frac{1}{3}(b(\pi) - 2c_2(\pi)) \tag{3.4}$$

Das equações (3.3) e (3.4) temos um novo limite inferior teórico

$$d(\pi) \geq \frac{2}{3}b(\pi) - \frac{1}{3}c_2(\pi)$$

3.4.2 O Número de Reversões Obtido pelo Algoritmo de Aproximação

O número de reversões obtidos pelo algoritmo de aproximação, está diretamente relacionado com a decomposição de ciclos do grafo de ciclos $G(\pi)$, e com o processo de encontrar uma sequência de eliminação.

Antes de encontrar uma decomposição de ciclos, vamos enunciar alguns lemas, proposições e teoremas úteis que são necessários para encontrar o número de reversões. A maioria das provas podem ser encontradas em (10).

O grafo de reversões $R(\mathcal{C})$ é usado para encontrar uma sequência de eliminação. O Lema 2, garante que $R(\mathcal{C})$ será atualizado corretamente considerando que depois de aplicar uma reversão sobre a permutação π, o grafo de ciclos $G(\pi)$ também deve ser atualizado. Os dados contidos nestes dois grafos $R(\mathcal{C})$ e $G(\pi)$ deve ser consistente, quer dizer, devem manter a mesma informação.

Lema 3. *Dada uma permutação π, uma decomposição de ciclos $\mathcal{C}$ do grafo de ciclos $G(\pi)$ e seja u um vértice do grafo de reversões $R(\mathcal{C})$. O vértice u é azul e isolado se, e somente se, representa uma adjacência em π.*

Demonstração.

- ($\leftarrow$) Pela definição de grafo de reversões, uma adjacência da permutação π corresponde-se com um vértice azul isolado em $R(\mathcal{C})$.
- ($\rightarrow$) Prova por contradição.
  Seja u um vértice azul isolado em $R(\mathcal{C})$ que não representa uma adjacência em π, quer dizer, u representa uma aresta cinza $(u, u+1)$ que forma parte de um ciclo da decomposição de ciclos $\mathcal{C}$. A aresta cinza $(u, u+1)$ só pode ter duas formas mostradas na Figura 3.13.

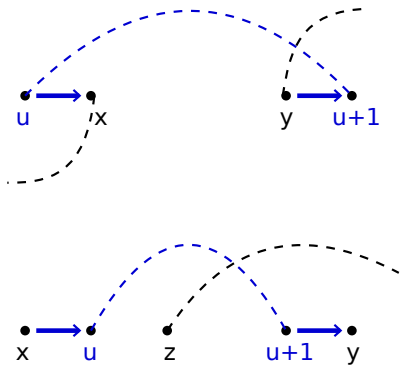

Figura 3.13: Duas únicas formas de uma aresta cinza, cuja reversão associada não elimina nenhum ponto de quebra.

  - No primeiro caso, como as arestas pretas (u, x) e $(y, u+1)$ são parte de um ciclo de $\mathcal{C}$ existirá uma aresta cinza saindo de x ou y, já que um ciclo de $\mathcal{C}$ está formado por arestas cinzas e pretas alternadas.

– No segundo caso, entre os pontos u e $u+1$ existirá um elemento z, tal que, formará um ponto de quebra com u ou com $u+1$, e já que esse ponto de quebra forma parte de um ciclo de $\mathcal{C}$ que está formado por arestas cinzas e pretas alternadas, existirá uma aresta cinza saindo de z.

Em ambos casos, a aresta cinza $(u, u+1)$ sempre *intercala* com outra aresta, e pela definição de grafo de reversões, o vértice u correspondente a $(u, u+1)$ no grafo $R(\mathcal{C})$ estará ligado a outro vértice. Claramente isto é uma contradição, desde que o vértice azul u é isolado.

□

Lema 4. *Dada uma permutação π e uma decomposição de ciclos $\mathcal{C}$ do grafo de ciclos $G(\pi)$. Um 2-ciclo não orientado de $\mathcal{C}$ deve estar intercalado com outro ciclo de $\mathcal{C}$.*

Demonstração. Sejam g_1 e g_2 duas arestas cinzas que formam um 2-ciclo não orientado; pela definição de ciclos não orientados sabemos que as arestas g_1 e g_2 representam reversões que não eliminam pontos de quebra. O 2-ciclo só pode ter a forma da Figura 3.14. Pela

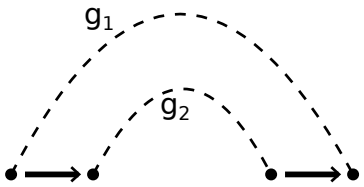

Figura 3.14: Única forma de um 2-ciclo não orientado

definição de grafos de ciclos as arestas g_1 e g_2 não são adjacências, pelo Lema 3, as arestas g_1 e g_2 representam vértices não isolados no grafo de reversões, quer dizer, que esses vértices estão ligados a um outro vértice; desta forma, as arestas g_1 e g_2 intercalam com uma outra aresta que é parte de um ciclo da decomposição de ciclos $\mathcal{C}$. Portanto, podemos que dizer que um 2-ciclo sempre intercala com outro ciclo. □

Lema 5. *Se uma componente A de $R(\mathcal{C})$ é orientada então contém um vértice vermelho u tal que cada componente de A_u é orientado.*

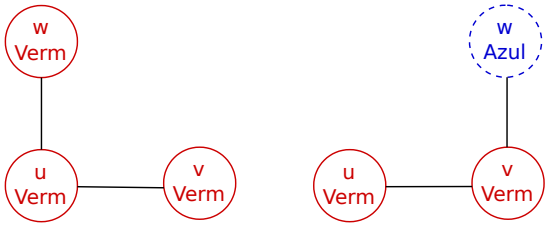

Figura 3.15: Vértice u do tipo 1 à esquerda, e do tipo 2 à direita

Demonstração. Prova por contradição. Suponha que A_u contém uma componente não orientada por cada vértice vermelho u em A. Pela definição de componente não orientada, as componentes não orientadas de A_u não podem ser um vértice azul isolado. Então, é fácil verificar que para cada vértice vermelho u em A, deve existir um par de vértices diferentes v e w tal que:

1) os vértices v e w ambos são vermelhos, $\{u, v\}$, $\{u, w\}$ são arestas em A e $\{v, w\}$ não é uma aresta em A, como mostrado no lado esquerdo da Figura 3.15. Ou,

2) o vértice v é vermelho e o vértice w é azul, $\{u, v\}$ e $\{v, w\}$ são arestas em A e $\{u, w\}$ não é uma aresta em A, como mostrado no lado direito da Figura 3.15.

Logo, depois de aplicar a reversão associado ao vértice vermelho u, atualizamos a componente A segundo o Lema 2 onde v e w serão parte de uma componente não orientada de A_u.

Um vértice u é do tipo 1, se cumpre só com o caso 1, caso contrário é do tipo 2. Seja V_r uma sequência de todos os vértices vermelhos do tipo 1 ou do tipo 2 em A, tal que V_r é circular, quer dizer, que o último e primeiro elemento são os mesmos. Para cada u de A, com $u = u_s$, se pode gerar a seguinte sequência $V_r = \{u_s, u_{s+1}, u_{s+2}, \ldots, u_{s+k}\}$, tal que $u_s = u_{s+k}$.

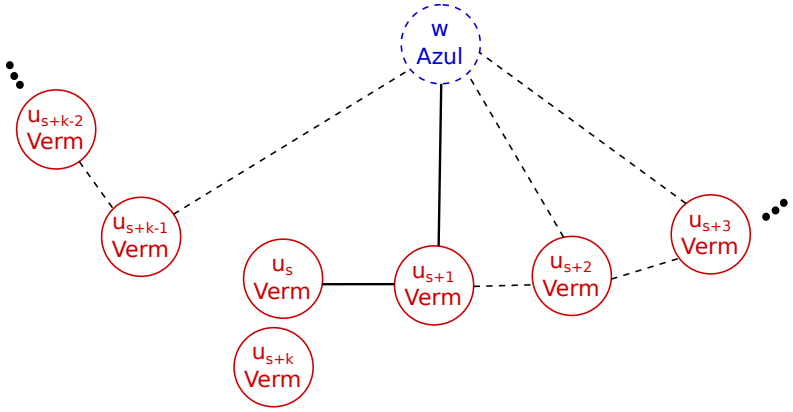

Figura 3.16: Sequência de vertices do tipo 2

Suponha que a sequência V_r contem um vértice u_s do tipo 2, como mostrado na Figura 3.16. Então existe um vértice w tal que $\{u_{s+1}, w\}$ é uma aresta, e $\{u_s, w\}$ não é uma aresta de A. Como o vértice vermelho u_{s+2} será parte de uma componente não orientada de $A_{u_{s+1}}$, então é necessário que exista a aresta $\{u_{s+2}, w\}$. Similarmente, devem existir as arestas $\{u_{s+3}, w\}, \{u_{s+4}, w\}, \ldots, \{u_{s+k}, w\}$; mas temos que $u_s = u_{s+k}$ com $\{u_s, w\}$ sendo uma aresta, e já que assumimos o contrário, temos uma contradição.

Agora só ficam vértices do tipo 1. Suponha que u_s é um vértice da sequência V_r, como mostrado na Figura 3.17. Então existe um vértice vermelho w tal que $\{u_s, u_{s+1}\}$ e $\{u_s, w\}$ são arestas, e $\{u_{s+1}, w\}$ não é uma aresta em A. Mas $u_s = u_{s+k}$, segue que u_s é uma componente não orientada de $A_{u_{s+k-1}}$. Então é necessário que exista a aresta $\{u_{s+k-1}, w\}$. Similarmente, $\{u_{s+k-1}, w\}, \{u_{s+k-1}, w\}, \ldots, \{u_{s+k-1}, w\}$, mas isto gera uma contradição desde que $\{u_{s+1}, w\}$ não é uma aresta.

□

O Lema 5, pode ser usado para encontrar uma sequência de eliminação para uma componente orientada A. É só aplicar o lema uma e outra vez até que a componente A seja eliminada totalmente, ou seja, não exista mais reversões por ser aplicadas.

Proposição 1. *Dada uma permutação π, uma decomposição de ciclos $\mathcal{C}$ do grafo de ciclos $G(\pi)$ e seja C um ciclo de $\mathcal{C}$. Existe uma sequência de reversões que elimina o ciclo C*

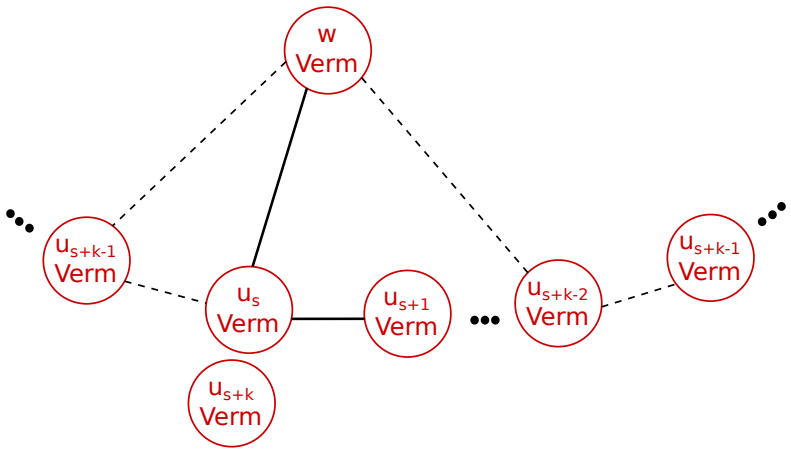

Figura 3.17: Sequência de vértices do tipo 1

tal que só uma dessas reversões, que é a última em ser aplicada, elimina dois pontos de quebra.

Lema 6. *Dada uma permutação π, uma decomposição de ciclos $\mathcal{C}$ do grafo de ciclos $G(\pi)$ e um grafo de reversões $R(\mathcal{C})$. Todos os vértices de $R(\mathcal{C})$ cujas arestas associadas formam parte de um mesmo ciclo C da decomposição de ciclos $\mathcal{C}$ são parte da mesma componente conexa em $R(\mathcal{C})$.*

Demonstração. Prova por contradição. Suponha que um ciclo C da decomposição de ciclos $\mathcal{C}$ dá origem a vértices em $R(\mathcal{C})$ que estão em diferentes componentes conexas. Cada uma destas componentes conexas tem associada uma sequência de reversões que elimina a componente. Pela Proposição 1, existe uma única reversão, associada a um vértice v de $R(\mathcal{C})$, da sequência de reversões para o ciclo C que elimina dois pontos de quebra. Suponha que uma das componentes conexas contem o vértice que representa a reversão v. Se aplicássemos essa sequência contendo a reversão v o ciclo seria eliminado. O que claramente é uma contradição já que ainda faltam aplicar as outras sequências de reversões. □

É possível usar os lemas 5 e 6 para calcular quantas reversões serão necessárias para eliminar uma componente orientada de $R(\mathcal{C})$, como é mostrado na Proposição 2.

Proposição 2. *Seja A uma componente orientada de $R(\mathcal{C})$ que contém vértices surgindo de k diferentes ciclos de $G(\pi)$. Então existe uma sequência de eliminação para A que contém k 2-reversões com as outras reversões sendo 1-reversão.*

O Lema 7 é uma consequência do Lema 2.

Lema 7. *Seja B uma componente não orientada de $R(\mathcal{C})$. Então B_v é uma componente orientada para cada vértice v de B.*

Para eliminar uma componente não orientada de $R(\mathcal{C})$, precisamos fazê-la orientada, isto é possível usando o Lema 7. A componente orientada resultante pode ser resolvida usando a proposição 2. Assim, o número de reversões para eliminar a componente não orientada é mostrado na seguinte proposição.

Proposição 3. *Seja A uma componente não orientada de $R(\mathcal{C})$ que contém vértices surgindo de k diferentes ciclos de $G(\pi)$. Então, existe uma sequência de eliminação para A que contém uma 0-reversão, e k 2-reversões, com o resto de reversões sendo 1-reversão.*

Aplicando repetidamente a Proposições 2 e 3 teremos um limite superior para a distância de reversão como é mostrado no seguinte teorema.

Teorema 1. *Seja $\mathcal{C}$ uma decomposição de ciclos de $G(\pi)$. Assim, usando só reversões de $R(\mathcal{C})$, π pode ser ordenada por $b(\pi) - |\mathcal{C}| + u(\mathcal{C})$, onde $u(\mathcal{C})$ é o número de componentes não orientadas de $R(\mathcal{C})$.*

Decomposição de Ciclos

Os seguintes passos são necessários para encontrar uma decomposição de ciclos que tenha um número máximo de 2-ciclos.

- Construir o **grafo de emparelhamento** $F(\pi)$.
  - Este grafo de emparelhamento $F(\pi)$ contém um vértice para cada aresta preta do grafo de ciclos $G(\pi)$.
  - Dois vértices u e v do grafo $F(\pi)$ estão ligados por uma aresta se existe um 2-ciclo no grafo $G(\pi)$ que contém ambas arestas pretas representas pelos vértices u e v.
- Encontrar no grafo $F(\pi)$ um **emparelhamento M de cardinalidade máxima**.
  - Inclusive depois de encontrar o emparelhamento M, ainda podem existir alguns 2-ciclos que compartilhem arestas cinzas.
  - Já que um 2-ciclo contém exatamente 2 arestas cinzas, é possível encontrar uma decomposição de ciclos que contém pelo menos $\lceil \frac{|M|}{2} \rceil$ 2-ciclos. Onde, $|M|$ é o número de 2-ciclos em M.

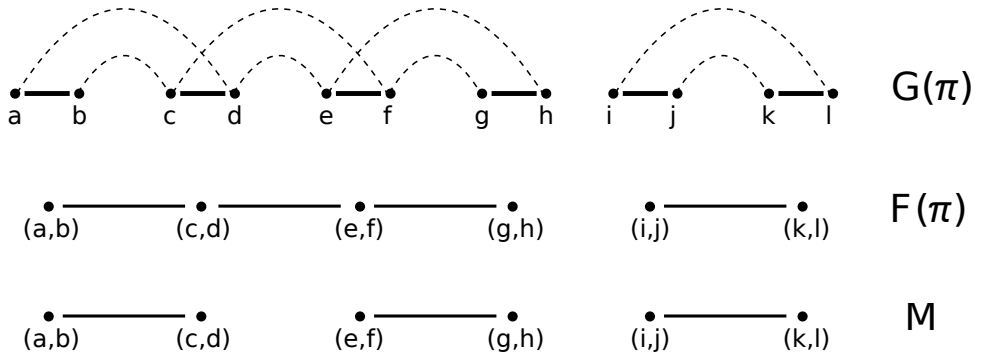

Figura 3.18: Exemplo de grafo de ciclos, grafo de emparelhamento, e um emparelhamento de cardinalidade máxima

- Construir o **grafo escada** $L(M)$
  - Este grafo escada $L(M)$ contém um vértice para cada 2-ciclo representado no grafo de emparelhamento M.

– Dois vértices do grafo $L(M)$ estão ligados por uma aresta se os 2-ciclos que eles representam compartilham uma aresta cinza.

Os 2-ciclos representados em M, que são representados com **vértices isolados** no grafo $L(M)$, são chamados de **2-ciclos independentes**. Por outro lado, os outros 2-ciclos representados no grafo M são chamados de **ciclos escada**, e os vértices em $L(M)$ representando estes ciclos, são chamados de **vértices escada**.

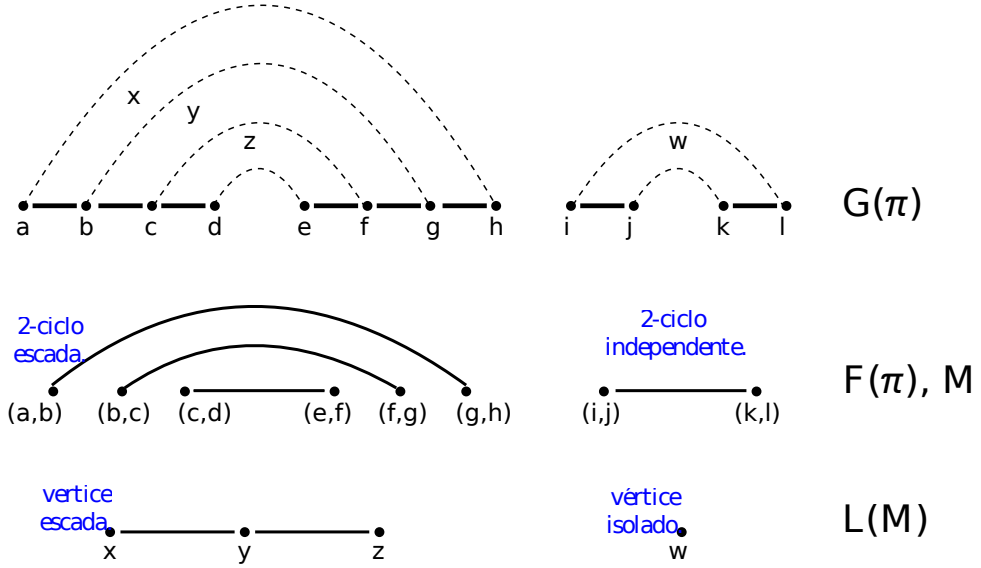

Figura 3.19: Exemplo de 2-ciclo escada, 2-ciclo independente, vértice escada, e vértice isolado. Neste caso M resulta igual do que $F(\pi)$.

Suponhamos que $L(M)$ contém z vértices isolados e y vértices escada. Note que $|M| = y + z$.

Lema 8. *Dado um emparelhamento M de cardinalidade máxima do grafo $F(\pi)$, é possível encontrar um ciclo de decomposição $\mathcal{C}$ do grafo $G(\pi)$ que contém no mínimo $\lceil \frac{y}{2} \rceil$ 2-ciclos escada e z 2-ciclos independentes.*

Demonstração. Construir o grafo $L(M)$ para o emparelhamento M. Este grafo consiste de caminho simples e vértices isolados. Assuma que $\mathcal{C}$ contém todos os z 2-ciclos independentes do grafo $L(M)$, e cada ciclo alternado de cada escada, ou seja, $\lceil \frac{y}{2} \rceil$ 2-ciclos escada. O resto de $\mathcal{C}$ pode ser obtido adicionando qualquer decomposição de ciclos das arestas restantes do grafo $G(\pi)$ que não foram usadas. □

Determinação do Número de Reversões

Para encontrar uma sequência de reversões que ordene uma permutação π usando no máximo $b(\pi) - \frac{1}{2}c_2(\pi)$ reversões, assumiremos, pelo Lema 8, que temos uma decomposição de ciclos $\mathcal{C}$ do grafo $G(\pi)$ que contém no mínimo $\lceil \frac{y}{2} \rceil$ 2-ciclos escada, e z 2-ciclos independentes.

Teorema 2. *É possível ordenar uma permutação π usando no máximo $b(\pi) - \frac{1}{2}c_2(\pi)$ reversões.*

Demonstração. Suponha que k dos 2-ciclos selecionados em $\mathcal{C}$ são parte de componentes orientadas do grafo de reversões $R(\mathcal{C})$. Então, pela Proposição 2, existe uma sequência de reversões que elimina todas as componentes orientadas e inclui no mínimo k 2-reversões e nenhuma 0-reversão.

Suponha que existem u componentes não orientadas em $R(C)$ que contém vértices representando ciclos que são 2-ciclos não selecionados, e que estas componentes incluem vértices representando l dos 2-ciclos selecionados em $\mathcal{C}$. Note que cada uma das u componentes não orientadas tem no mínimo um ciclo que é um 2-ciclo não selecionado. Então, pela Proposição 3, existe uma sequência de reversões que elimina estas componentes e inclui no mínimo $l + u$ 2-reversões e só u 0-reversões.

Suponha que as v componentes não orientadas restantes de $R(\mathcal{C})$, que consistem só de vértices representando 2-ciclos independentes, contêm vértices representando m 2-ciclos. Então, pela Proposição 3, existe uma sequência de reversões que eliminam estas componentes, e incluem no mínimo m 2-reversões e só v 0-reversões. Temos assumido que existem z 2-ciclos independentes, que são não orientados por formar parte das v componentes desorientas; pelo Lema 4 sabemos que um 2-ciclo não orientado intercala com outro ciclo, e como temos v componentes não orientadas que consistem só de vértices representando 2-ciclos independentes, os z 2-ciclos só intercalam com 2-ciclos não orientados. Então, cada uma das v componentes não orientadas contém vértices representando no mínimo dois 2-ciclos independentes. Portanto, se cumpre que $v \leq \lfloor \frac{z}{2} \rfloor$.

A sequência de reversões encontrada pelos passos antes descritos ordenará a permutação π usando no mínimo $k+l+m+u$ 2-reversões e $u+v$ 0-reversões. Então, pelo Teorema 1, são requeridas no máximo $b(\pi) - k - l - m + v$ reversões para ordenar a permutação π, assim

$$d(\pi) \leq b(\pi) - k - l - m + v$$

Porém $k + l + m \geq \lceil \frac{y}{2} \rceil + z$, então

$$k + l + m - v \geq \lceil \frac{y}{2} \rceil + z - v \tag{3.5}$$

Depois, $v \leq \lfloor \frac{z}{2} \rfloor$, então

$$\begin{aligned} -v &\geq -\lfloor \frac{z}{2} \rfloor \\ \lceil \frac{y}{2} \rceil + z - v &\geq \lceil \frac{y}{2} \rceil + z - \lfloor \frac{z}{2} \rfloor \\ \lceil \frac{y}{2} \rceil + z - v &\geq \lceil \frac{y}{2} \rceil + \lceil \frac{z}{2} \rceil \end{aligned} \tag{3.6}$$

Temos que $|M| \geq c_2(\pi)$ e $y + z = |M|$, então

$$\begin{aligned} y + z &\geq c_2(\pi) \\ \lceil \frac{y}{2} \rceil + \lceil \frac{z}{2} \rceil &\geq \frac{c_2(\pi)}{2} \end{aligned} \tag{3.7}$$

Das Equações 3.5, 3.6 e 3.7, podemos ver que

$$\begin{aligned} k+l+m-v &\geq \frac{c_2(\pi)}{2} \\ -k-l-m+v &\leq -\frac{c_2(\pi)}{2} \\ b(\pi)-k-l-m+v &\leq b(\pi)-\frac{c_2(\pi)}{2} \end{aligned}$$

Portanto

$$d(\pi) \leq b(\pi) - \frac{1}{2}c_2(\pi)$$

□

Em resumo, para atingir o raio de aproximação 1.5 é necessário encontrar uma decomposição de ciclos que maximize o número de 2-ciclos, ou seja, uma decomposição de ciclos que tenha no mínimo $\lceil \frac{y}{2} \rceil$ 2-ciclos (ciclos escada) e z 2-ciclos independentes, como foi assumido no Lema 8. Também é necessário aplicar as reversões usando o grafo de reversões $R(\mathcal{C})$, como foi feito usando as Proposições 2 e 3.

3.5 Implementação do Algoritmo de Aproximação

O algoritmo de raio de aproximação 1.5 foi usado como mecanismo de controle da qualidade das soluções produzidas por outros algoritmos, por exemplo algoritmo genéticos (11, 13, 12). Foi por esse motivo que houve a necessidade de fazer uma implementação deste algoritmo de aproximação. Em Algoritmo 1 se apresenta o pseudocódigo; também é apresentado a análise de complexidade tanto como alguns detalhes do algoritmo.

A ideia do algoritmo é operar sobre o grafo de reversões $R(\mathcal{C})$ eliminando vértices vermelhos, os quais representam reversões que eliminam pontos de quebra. Depois de escolher um vértice vermelho para ser eliminado é aplicado sobre π a reversão que este vértice representa. Também tem que ser atualizada a decomposição de ciclos $\mathcal{C}$ e o grafo de reversões $R(\mathcal{C})$ de acordo com o estabelecido no Lema 4.1 modificado. O algoritmo termina quando todos os vértices vermelhos tenham sido eliminados, ficando só vértices azuis isolados, o que só acontece quando a permutação está ordenada.

A construção do grafo de reversões $R(\mathcal{C})$ é feita sobre a decomposição de ciclos $\mathcal{C}$, cuja construção é um ponto crítico na demostração do raio de aproximação 1.5, já que é preciso encontrar o número máximo de 2-ciclos que não compartilhem arestas pretas nem cinzas. É baseada nesta decomposição que foi demonstrado em (10) que o algoritmo pode ordenar uma permutação usando no máximo $b(\pi) - \frac{1}{2}c_2(\pi)$ reversões.

3.5.1 Análise da Complexidade de Tempo do Algoritmo de Aproximação

O grafo de pontos de quebra $G(\pi)$ é construído em tempo linear.

Para construir uma decomposição de ciclos $\mathcal{C}$, temos que maximizar o número de 2-ciclos que não compartilhem arestas pretas nem cinzas; para este propósito primeiro é construído um grafo chamado de *grafo de emparelhamento* $F(\pi)$, a partir de $G(\pi)$. Este

Algoritmo 1: Algoritmo para ordenação por reversão de raio de aprox. 1.5

Entrada: Uma permutação π
Saída: Sequência de reversões S para ordenar a permutação de entrada

```
1  Construir o grafo de pontos de quebra G(π);
2  Construir o grafo de emparelhamento F(π);
3  Encontrar o emparelhamento máximo M;
4  Gerar a decomposição de ciclos C usando M;
5  Construir o grafo de reversões R(C);
6  para cada componente não orientada de R(C) faça
7      Aplicar reversão sobre permutação π representada por um vértice azul v;
8      Atualizar C e R(C);
9      Adicionar a reversão representada pelo vértice v a S;
10 enquanto permutação π não ordenada faça
11     Escolher uma componente d qualquer;
12     para cada vértice v da componente d faça
13         se vértice v é vermelho então
14             Aplicar sobre π a reversão representada pelo vértice v;
15             Atualizar C e R(C);
16             se reversão v gerou componentes não orientadas em R(C) então
17                 Voltar para o estado anterior de C e R(C);
18             senão
19                 Adicionar a reversão representada pelo vértice v a S;
20                 break;
```

grafo contém todos os 2-ciclos que compartilham arestas pretas. Este grafo é construído em tempo linear.

Logo para encontrar um emparelhamento máximo M, foi usado o algoritmo de emparelhamento de Edmonds, que tem complexidade de tempo de $O(n^{\frac{3}{2}})$. O mesmo procedimento é feito para obter 2-ciclos que não compartilhem arestas cinzas.

O emparelhamento máximo M nos dá todos os 2-ciclos que não compartilham arestas pretas nem cinzas, as arestas restantes do grafo $G(\pi)$ que não foram escolhidas para serem 2-ciclos, podem ser usadas para gerar uma decomposição de ciclos qualquer. Então a geração da decomposição de ciclos $\mathcal{C}$ é feita em tempo linear.

A partir da decomposição de ciclos $\mathcal{C}$ é construído o grafo de reversões $R(\mathcal{C})$ em tempo $O(n^2)$. O grafo de reversões estará formado por várias componentes conectadas, as quais são consideradas orientadas se têm pelo menos um vértice vermelho, ou consiste apenas de um vértice azul isolado. Caso contrário são componentes não orientadas.

Quando são encontradas componentes não orientadas, é aplicada a reversão sobre a permutação π representada por qualquer vértice azul desta componente, gerando vértices vermelhos e desta forma convertendo a componente em orientada.

Quando são encontradas componentes não orientadas, para cada componente não orientada é escolhido um vértice azul e aplicado a reversão correspondente sobre a permutação π e também atualizada a decomposição de ciclos $\mathcal{C}$. A atualização sobre o grafo

de reversões vai gerar vértices vermelhos convertendo a componente não orientada em orientada.

O algoritmo termina quando a permutação π está ordenada, e todos os vértices de $R(\mathcal{C})$ ficaram isolados e de cor azul. Por cada vértice vermelho é aplicada uma reversão sobre a permutação π em tempo linear e atualizada a decomposição de ciclos $\mathcal{C}$ em tempo constante, e dado que temos $O(n)$ vértices vermelhos, encontrar a sequência de reversões toma um tempo de $O(n^2)$, e portanto a complexidade de tempo total é de $O(n^2)$.

3.5.2 Representação das Estrutura de Dados Usadas

Na fase de implementação um dos grandes problemas foi encontrar a representação adequada das estruturas de dados do grafo de ciclos, e sua decomposição em ciclos. Desde que a modificação de um deles implica a modificação do outro, e ambos tem que conter a mesma informação, quer dizer que a informação de um deles seja coerente com a do outro.

Para representar a permutação π, foi usado um arranjo de inteiros de tamanho n. O grafo de ciclos $G(\pi)$ foi representado usando duas matrizes de inteiros, as dimensões de cada uma das matrizes é 2 x n; uma das matrizes é para salvar as arestas pretas e outro para salvar as arestas cinzas. Por cada elemento de π como máximo se pode ter duas arestas pretas (e como máximo duas cinzas).

Figura 3.20: Representação de um grafo de ciclos

A Figura 3.20 mostra a representação do grafo de ciclos mostrado na Figura 2.1. Note-se que no grafo de ciclos, o elemento 1 da permutação π forma uma aresta cinza com o elemento 0, e outra aresta cinza com o elemento 2; é por isso que na matriz de arestas cinzas na coluna correspondente ao elemento 1, estão guardados os valores 0 e 2. Analogamente temos que o elemento 1 de π forma uma aresta preta com o elemento 7 e outra com o elemento 9; é por isso que na matriz de arestas pretas na coluna correspondente ao elemento 1, estão guardados os valores 7 e 9.

Na representação da decomposição de ciclos $\mathcal{C}$, por cada ciclo de $\mathcal{C}$ se tem uma matriz de inteiros, de dimensões 3 x n. Um ciclo está formado por alguns elementos da permutação π e as arestas pretas e cinzas relacionadas com esses elementos. Por cada elemento v de um ciclo são guardados 3 elementos: um elemento $v+1$ que junto com v formam uma aresta cinza, um elemento p que junto com v formam uma aresta preta, e um elemento p' que junto com $v+1$ formam uma aresta preta. A representação do grafo de reversões $R(\mathcal{C})$ é usando uma matriz de n x n.

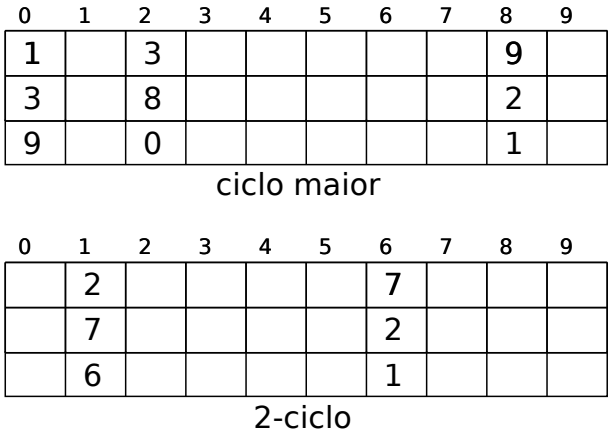

Figura 3.21: Representação de uma decomposição de ciclos

A figura 3.21 mostra a representação da decomposição de ciclos mostrada na Figura 2.2. Note-se que na matriz que representa o "ciclo maior", cada uma das colunas preenchidas representa uma aresta cinza com suas respectivas arestas pretas adjacentes. Por exemplo, o elemento 2 forma uma aresta cinza com o elemento 3, e por sua vez também forma uma aresta preta com o elemento 8; o elemento 3 forma uma aresta preta com o elemento 0. É por isso que na matriz do "ciclo maior"na coluna correspondente ao elemento 2, são guardados os valores 3, 8 e 0.

Capítulo 4

O Método Proposto

4.1 Mapeamento de permutações sem sinal em 2^n permutações com sinal

A ideia proposta por Auyeung e Abraham em (11) é a seguinte: Dada uma permutação sem sinal, explorar todas as possíveis decomposições em ciclos, isto pode ser feito atribuindo um sinal aleatoriamente positivo ou negativo a cada elemento da permutação sem sinal, gerando desta maneira um conjunto de permutações com sinal. A solução de uma destas permutações com sinal representará uma solução ótima para a distância de reversão da permutação sem sinal inicial.

Podemos tomar como exemplo a permutação $\pi = \{2\ , 1\}$. O conjunto de permutações com sinal correspondente seria $\{(\overrightarrow{2}\ ,\ \overrightarrow{1}), (\overrightarrow{2}\ ,\ \overleftarrow{1}), (\overleftarrow{2}\ ,\ \overrightarrow{1}), (\overleftarrow{2}\ ,\ \overleftarrow{1})\}$. Pode-se observar que serão geradas 2^n permutações com sinal. É a partir desse fato que surge a ideia de usar algoritmos genéticos para explorar o espaço de soluções.

É claro que nem sempre o algoritmo genético vai encontrar uma solução ótima, mas a exploração do espaço de busca de 2^n e a aplicação dos operadores genéticos darão como resultado uma solução próxima do ótimo.

4.2 Algoritmo Genético Híbrido

O algoritmo genético (AG) proposto no presente trabalho está baseado no algoritmo genético padrão proposto por Auyeung e Abraham em (11) e uma das características importantes a salientar em nossa proposta é a hibridização com métodos exatos como é o caso do algoritmo de tempo linear para o problema de ordenação de permutações com sinal, e o uso de uma heurística de eliminação de pontos de quebra. Todas essas ideias serão desenvolvidas e explicadas a seguir.

Um resumo do algoritmo é o seguinte: Inicialmente é gerada uma população de indivíduos(permutações com sinal) de tamanho $n\ log\ n$ a partir da permutação sem sinal inicial a ser ordenada. Depois em cada geração a reprodução de indivíduos segue o seguinte processo padrão: dois indivíduos da população são tomados para os quais os operadores de cruzamento e mutação são aplicados produzindo uma descendência. Os dois novos descendentes produzidos são colocados novamente na população. O AG termina depois de completar o número de gerações estabelecido inicialmente.

É importante salientar que em cada geração é calculado a função de aptidão dos indivíduos. A função de aptidão de cada indivíduo (permutação com sinal) é a distância de reversão, que foi calculada usando um algoritmo de tempo linear, cuja implementação foi fornecida em (9).

Foi feita uma melhora sobre o AG padrão proposto por Auyeung e Abraham que consiste na aplicação de uma heurística de eliminação simultânea de dois pontos de quebra sobre a permutação sem sinal inicial, quer dizer a aplicação de 2-reversões. Este processo é feito escolhendo e aplicando aleatoriamente 2-reversões e acaba quando não é possível aplicar mais reversões deste tipo. É justo depois desse passo que é gerada a população inicial a partir da permutação resultante de aplicar a heurística.

No entanto esta heurística é um mecanismo guloso que não produzirá necessariamente uma sequência de ordenação ótima; o fato de usar reversões que eliminam simultaneamente dois pontos de quebra foi usado como método de otimização local em muitos algoritmos de aproximação (14, 10). Também foi observado que as 2-reversões aparecem mais frequentemente nas soluções ótimas, por sobre as 1-reversões e 0-reversões, já que este tipo de reversões são as que eliminam mais pontos de quebra e portanto ajudam a ordenar a permutação usando menos reversões.

4.3 Implementação do Algoritmo Genético

O pseudocódigo do AG proposto é mostrado no Algoritmo 2. Quando a condição de *"melhorar"* é verdadeira, é aplicada a heurística e neste caso chamamos o AG de **hibrido**; Caso contrário quando a heurística não é aplicada chamamos o AG de **padrão**.

Algoritmo 2: AG para ordenar permutações por reversões com um parâmetro para aplicar a versão melhorada do AG.

Entrada: Uma permutação sem sinal π, um booleano *"melhorar"* que indica se deve-se aplicar a heurística.

Saída: Um número de reversões para ordenar a permutação π.

```
1  se "melhorar" então
2      Aplicar 2-reversões sobre π, que eliminam simultaneamente dois pontos de
       quebra , até que não possam ser aplicadas reversões deste tipo.
3  Gerar a população inicial de permutações com sinal a partir de π;
4  Calcular a aptidão de cada indivíduo da população inicial;
5  para i = 2 até (número de gerações) faça
6      Seleção;
7      Recombinação;
8      Mutação;
9      Calcular aptidão da descendência;
10     Substituição;
```

Análise da complexidade de tempo do AG

Seja n o tamanho da permutação de entrada. O tamanho da população inicial foi fixado para ser $n\ log\ n$. Cada indivíduo da população inicial foi gerado a partir da permutação inicial em tempo linear atribuindo aleatoriamente um sinal (positivo ou negativo) a cada um dos seus elementos. Já que temos $n\ log\ n$ indivíduos a geração da população inicial toma um tempo de $O(n^2\ log\ n)$.

O algoritmo usado para ordenar os indivíduos da população por seu valor de aptidão, na etapa da seleção, é a *ordenação por contagem* que é conhecido por tomar um tempo linear e já que temos que ordenar $n\ log\ n$ toma um tempo de $O(n\ log\ n)$.

Na etapa da recombinação, os melhores indivíduos da população são escolhidos para serem pais. Para cada par de pais o operador de recombinação é aplicado em tempo linear, gerando dois descendentes. Desde que temos $n\ log\ n$ indivíduos o tempo total para executar a etapa da recombinação é de $O(n^2\ log\ n)$.

Na etapa da mutação, para cada indivíduo da descendência é aplicado o operador de mutação. Para cada elemento de um indivíduo da descendência é verificado se pode ser aplicado a mutação, que consiste em mudar o sinal de positivo a negativo ou de negativos a positivo. Desde que temos $n\ log\ n$ indivíduos o tempo total para executar a etapa da mutação é de $O(n^2\ log\ n)$.

Sabe-se que calcular a aptidão de cada indivíduo pode ser feito usando o algoritmo fornecido em (9), que é de tempo linear. Desde que temos $n\ log\ n$ indivíduos o tempo total para executar a aptidão é de $O(n^2\ log\ n)$.

A etapa da substituição toma um tempo de $O(n^2\ log\ n)$, já que podem ser substituídos $n\ log\ n$ indivíduos.

O algoritmo termina depois de n gerações, então temos que a complexidade de tempo total do algoritmo é de $O(n^3\ log\ n)$.

4.4 Algoritmo de Tempo Polinomial para o Cálculo da Distância de Reversão de Permutações com Sinal

Vamos apresentar alguns detalhes para construir o algoritmo de tempo polinomial para o cálculo da distância de reversão de permutações com sinal, proposto inicialmente por Hannenhalli e Pevzner em (2), e posteriormente aprimorado para um algoritmo de tempo linear por Bader et al. em (9).

4.4.1 Grafo de Ciclos e Grafo de Sobreposição de Permutações sem sinal

Dada uma permutação com sinal $\overrightarrow{\pi} = \{\overrightarrow{\pi_1}, \overrightarrow{\pi_2}, \ldots, \overrightarrow{\pi_n}\}$ esta é transformada em uma permutação $\pi = \{\pi_1, \pi_2 \ldots, \pi_{2n}\}$ substituindo os elementos positivos x pelo par ordenado $(2x-1, 2x)$, e os negativos $-x$ pelo par ordenado $(2x, 2x-1)$. Então, a permutação π é estendida adicionando um pivô inicial e final, $\pi_0 = 0$ e $\pi_{2n+1} = 2n+1$. Por exemplo, considere a seguinte permutação com sinal

$$\overrightarrow{\pi} = (-2, +5, -3, +1, -4)$$

É transformada em:

$$\pi = (0,\ 4,\ 3,\ 9,\ 10,\ 6,\ 5,\ 1,\ 2,\ 8,\ 7,\ 11)$$

Os pontos de quebra e o grafo de ciclos $G(\overrightarrow{\pi})$ de uma permutação com sinal $\overrightarrow{\pi}$ correspondem aos da permutação sem sinal π obtida a partir de $\overrightarrow{\pi}$. Quando tratamos outras propriedades como número de pontos de quebra $b(\overrightarrow{\pi})$, e número de ciclos $c(\overrightarrow{\pi})$ de $G(\overrightarrow{\pi})$, na verdade estamos falando das propriedade da permutação π obtida a partir de $\overrightarrow{\pi}$, ou seja, $b(\pi)$, e $c(\pi)$ de $G(\pi)$ respectivamente. Por exemplo, na Figura 4.1 o grafo de ciclos $G(\overrightarrow{\pi})$ da permutação $\overrightarrow{\pi} = (-2, +5, -3, +1, -4)$ é o grafo de ciclos da permutação π obtida a partir de $\overrightarrow{\pi}$.

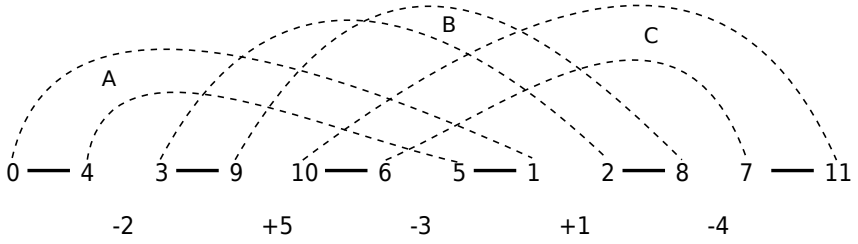

Figura 4.1: Grafo de ciclos de uma permutação com sinal

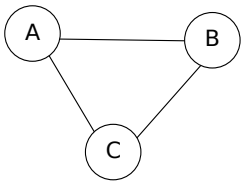

Figura 4.2: Grafo de sobreposição da permutação com sinal da Figura 4.1

Definição 15. *Seja uma permutação com sinal $\overrightarrow{\pi}$, e um grafo de ciclos $G(\overrightarrow{\pi})$*

- *Duas arestas cinzas (π_i, π_j) e (π_k, π_l) de $G(\overrightarrow{\pi})$ intercalam se as posições (i, j) e (k, l) intercalam($i < k < j < l$), mas nenhum deles contem ao outro.*
- *Similarmente, podemos dizer que dois ciclos C_1 e C_2 de $G(\overrightarrow{\pi})$ intercalam se existem duas arestas cinzas $g_1 \in C_1$ e $g_2 \in C_2$, tal que g_1 e g_2 intercalam.*

Definição 16. *Um **grafo de sobreposição** $GS(\overrightarrow{\pi})$ tem um vértice por cada ciclo no grafo de ciclos $G(\overrightarrow{\pi})$, e uma aresta entre dois vértices que se correspondem com ciclos que intercalam em $G(\overrightarrow{\pi})$.*

Na figura Figura 4.2 temos o grafo de sobreposição da permutação com sinal $\overrightarrow{\pi} = (-2, +5, -3, +1, -4)$.

4.4.2 Obstáculos, Componentes Conexas e Floresta de Sobreposição

As definições de ciclos e componentes conexas orientadas são as mesmas das permutações sem sinal. Sempre que os ciclos e componentes conexas estejam relacionadas com

arestas cinzas, do grafo de ciclos, que representem reversões que eliminem no mínimo um ponto de quebra, serão orientados, caso contrário serão não orientados.

Um **obstáculo** (do inglês *hurdle*) da permutação $\overrightarrow{\pi}$ é definido como uma componente conexa não orientada do grafo de sobreposição que satisfaz algumas condições adicionais, como definido em (2). Seja $h(\overrightarrow{\pi})$ o número de obstáculos de $\overrightarrow{\pi}$. Um obstáculo é simples quando depois de ser eliminado do grafo de sobreposição, nenhuma outra componente não orientada torna-se um obstáculo. Caso contrário, é um **super-obstáculo**. Uma **fortaleza** (do inglês *fortress*) é uma permutação que tem um número ímpar de obstáculos, todos os quais são super-obstáculos.

Defina: $f(\overrightarrow{\pi}) = \begin{cases} 1, & \text{se } \pi \text{ é uma fortaleza} \\ 0, & \text{caso contrário} \end{cases}$

Seja π a permutação obtida a partir da permutação com sinal $\overrightarrow{\pi}$, então definimos $b(\overrightarrow{\pi}) = b(\pi)$, $c(\overrightarrow{\pi}) = c(\pi)$, $h(\overrightarrow{\pi}) = h(\pi)$, e $f(\overrightarrow{\pi}) = f(\pi)$. O principal resultado obtido por Hannenhalli e Pevzner em (2) é:

Teorema 3 ((2))**.** *Seja $\overrightarrow{\pi}$ uma permutação com sinal, $d(\overrightarrow{\pi}) = b(\overrightarrow{\pi}) - c(\overrightarrow{\pi}) + h(\overrightarrow{\pi}) + f(\overrightarrow{\pi})$.*

Note que este teorema leva a um algoritmo de tempo polinomial, já que a distância de reversão está expressada em termos facilmente computáveis. Onde, $h(\overrightarrow{\pi})$ e $f(\overrightarrow{\pi})$ são facilmente detectáveis a partir de uma análise das componentes conexas. O algoritmo de tempo polinomial de Hannenalli e Pevzner (2) tem complexidade de tempo quadrática para calcular a distância de reversão, e requer o cálculo das componentes conexas do grafo de sobreposição.

Não existe algoritmo para construir o grafo de sobreposição que seja de tempo linear, já que o grafo de sobreposição pode ser de tamanho quadrático. Bader et al.(9) propuseram um algoritmo de tempo linear para calcular as componentes conexas construindo uma **floresta de sobreposição**, onde dois vértices f e g pertencem à mesma árvore exatamente quando eles pertencem à mesma componente conexa no grafo de sobreposição.

Uma floresta de sobreposição tem exatamente uma árvore por componente conexa do grafo de sobreposição e portanto é de tamanho linear. Este fato faz com que o algoritmo para o cálculo da distância de reversão também seja linear.

4.4.3 Apresentação Elementar da Teoria de Hannenhalli e Pevzner

Na subseção anterior foram apresentados alguns detalhes do algoritmo de tempo polinomial para o cálculo da distância de reversão, e que também podem ser utilizados para calcular a sequência de reversões para ordenar uma permutação com sinal incrementando a complexidade, mas mantendo-a de tempo polinomial. Bergeron apresentou em (17) um tratamento simplificado deste problema, operando diretamente sobre as permutações com sinal. Estes resultados serão apresentados a seguir.

Ordenação Básica sobre Permutações com Sinal

Seja $\overrightarrow{\pi} = (\overrightarrow{\pi}_1, \ldots, \overrightarrow{\pi}_i, \overrightarrow{\pi}_{i+1}, \ldots, \overrightarrow{\pi}_j, \ldots, \overrightarrow{\pi}_n)$ uma permutação com sinal, é estendida adicionando os pivôs 0 e $n+1$, os quais são sempre positivos.

$$\overrightarrow{\pi} = (0, \overrightarrow{\pi}_1, \overrightarrow{\pi}_2, \ldots, \overrightarrow{\pi}_n, n+1)$$

Uma reversão $\rho^{i..j}$, sobre o intervalo $[i, j]$, transforma $\overrightarrow{\pi}$ em:

$$\overrightarrow{\pi}' = (0, ..., \overrightarrow{\pi}_{i-1}, -\overrightarrow{\pi}_j, ..., -\overrightarrow{\pi}_i, \overrightarrow{\pi}_{j+1}, ..., n+1)$$

Um **par orientado** (π_i, π_j) é um par de inteiros consecutivos de sinais opostos, tal que $|\pi_i| - |\pi_j| = \pm 1$.

Por exemplo, seja $\overrightarrow{\pi} = (0, 5, 3, 1, -4, 2, 6)$, os seus pares orientados são $(5, -4)$ e $(3, -4)$.

Um par orientado indica uma reversão que crias dois inteiros consecutivos são adjacentes. Por exemplo, o par orientado $(5, -4)$ indica a reversão $\rho^{1..3}$, cuja permutação resultante seria:

$$\overrightarrow{\pi}' = (0, -1, -3, -5, -4, 2, 6)$$

A reversão $\rho^{1..3}$ cria os inteiros consecutivos -5 e -4, os quais são adjacentes. Generalizando, um par orientado (π_i, π_j) indica a reversão:

$$\begin{array}{ll} \rho^{i..j-1}, & se\ \pi_i + \pi_j = +1, ou \\ \rho^{i+1..j}, & se\ \pi_i + \pi_j = -1 \end{array}$$

Note que uma reversão que cria inteiros consecutivos adjacentes sempre está induzida por um par orientado, essas reversões são chamadas de **reversões orientadas**.

Definimos a **pontuação** de uma reversão orientada como o número de pares orientados na permutação resultante. Por exemplo, a pontuação da reversão $\rho^{1..3}$ é 4, já que a permutação resultante $\overrightarrow{\pi}'$ tem os seguintes pares orientados : $(0, -1)$, $(-1, 2)$, $(-3, 2)$, $(-5, 6)$.

O fato de que as reversões orientadas tem um efeito benéfico na ordenação de uma permutação, sugere a seguinte estratégia de ordenação.

Algoritmo 3: Algoritmo básico para ordenar permutações com sinal.

Entrada: Uma permutação com sinal $\overrightarrow{\pi}$.
Saída: Uma permutação $\overrightarrow{\pi}$ com apenas elementos positivos.

```
1 enquanto π⃗ tenha um par orientado faça
2 |   escolher a reversão orientada que tenha pontuação máxima;
3 |   aplicar a reversão orientada sobre π⃗;
```

Seguindo essa estratégia, ordenações ótimas são construídas (17):

$$\text{se } \overrightarrow{\pi}' \text{ é obtida de } \overrightarrow{\pi} \text{ apos } k \text{ aplicações do Algoritmo 3, então } d(\overrightarrow{\pi}) = k + d(\overrightarrow{\pi}')$$

Este fato é demostrado por Bergeron em (17). Especificamente se prova que uma reversão orientada que tenha pontuação máxima é segura. Uma reversão é **segura** desde que não crie novas componentes não orientadas exceto para vértices isolados(componentes que já foram ordenadas). Hannenhalli e Pezner estabeleceram em (2) que qualquer sequência de reversões seguras são ótimas.

Pode-se conferir, rolando este algoritmo, que esta estratégia simples ordena uma grande quantidade de permutações aleatórias, e quase todas as permutações que surgem de dados biológicos, já que só precisa encontrar pares orientados de pontuação máxima. No caso em que a saída do algoritmo é uma permutação com apenas elementos positivos e que ainda não esteja ordenada, será necessário um tratamento diferente explicado a seguir.

Ordenação de Permutações com Sinal com apenas Elementos Positivos

Seja $\overrightarrow{\pi}$ uma permutação com sinal com apenas elementos positivos, assumir que $\overrightarrow{\pi}$ é **reduzida**, quer dizer, que $\overrightarrow{\pi}$ não contém elementos consecutivos. Suponha também que $\overrightarrow{\pi}$ é estendida adicionando os pivôs 0 e $n+1$, e considere que a permutação tenha uma ordem circular tal que 0 é o sucessor de $n+1$.

Por exemplo, a permutação $\overrightarrow{\pi} = (0, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, 1, 7)$ é reduzida para a permutação $\overrightarrow{\pi} = (0, 2, 1, 7)$.

Defina um **intervalo emoldurado** em $\overrightarrow{\pi}$ como o intervalo da forma:

$$\mathbf{i}, \underbrace{\overrightarrow{\pi}_{j+1}, \overrightarrow{\pi}_{j+2}, \ldots, \overrightarrow{\pi}_{j+k-1}}_{k-1 \text{ elementos}}, \mathbf{i+k}$$

tal que todos os inteiros entre i e $i+k$ pertencem ao intervalo $[i..i+k]$. Por exemplo, considere a permutação

$$\overrightarrow{\pi} = (\mathbf{0}, 2, 5, 4, 3, 6, 1, \mathbf{7})$$

A permutação inteira é um intervalo emoldurado. Mas também temos o intervalo $\mathbf{2}, 5, 4, 3, \mathbf{6}$, o qual pode ser ordenado como $2, 3, 4, 5, 6$, e, por circularidade, o intervalo $\mathbf{6}, 1, 7, 0, \mathbf{2}$, o qual pode ser ordenado como $6, 7, 0, 1, 2$, desde que 0 é o sucessor de 7.

Definição 17. *Se $\overrightarrow{\pi}$ é reduzido, um **obstáculo** em $\overrightarrow{\pi}$ é um intervalo emoldurado que não contém intervalos emoldurados menores.*

Duas operações são introduzidas em (2), a primeira é **corte de obstáculo** que consiste em reverter o segmento entre os elementos i e $i+1$ de um obstáculo.

$$\mathbf{i}, \underline{\overrightarrow{\pi}_{j+1}, \overrightarrow{\pi}_{j+2}, \ldots}, i+1, \ldots, \overrightarrow{\pi}_{j+k-1}, \mathbf{i+k}$$

A reversão aplicada é suficiente para depois ordenar a permutação usando o Algoritmo 3. Por exemplo, a seguinte permutação contém somente um obstáculo.

$$\overrightarrow{\pi} = (\mathbf{0}, \underline{2, 4, 3}, 1, \mathbf{5})$$

A reversão dos elementos $2, 3$, e 4 corta o obstáculo, cuja permutação resultante seria:

$$\overrightarrow{\pi}' = (0, -3, -4, -2, 1, 5)$$

que pode ser ordenada usando o Algoritmo 3.

A segunda operação é a **mescla de obstáculos**, que age nos pontos extremos de dois obstáculos,

$$i, \ldots, \underline{i+k, \ldots i'}, \ldots, i'+k'$$

aplicando a reversão $\rho^{i+k..i'}$. Se uma permutação tem só dois obstáculos, sua mescla produzirá uma permutação que pode ser completamente ordenada pelo Algoritmo 3. Por exemplo, a mescla dos obstáculos $\mathbf{2}, 5, 4, 3, \mathbf{6}$ e $\mathbf{6}, 1, 7, 0, \mathbf{2}$ da permutação:

$$\overrightarrow{\pi} = (0, 2, 5, 4, 3, \underline{6}, 1, 7)$$

produz a permutação:

$$\overrightarrow{\pi}' = (0, 2, 5, 4, 3, -6, 1, 7)$$

que pode ser ordenada usando o Algoritmo 3.

O corte e mescla de obstáculos em uma permutação que contém mais de dois obstáculos dever ser feito cuidadosamente, já que se pode criar novos obstáculos.

Definição 18. *Um **obstáculo simples** é um obstáculo cujo corte decrementa o número de obstáculos. Obstáculos que não são simples são chamados de **super obstáculos**.*

Por exemplo, a permutação $\overrightarrow{\pi} = (0, 2, 5, 4, 3, 6, 1, 7)$ tem dois obstáculos. O corte e ordenação do obstáculo $\mathbf{2}, \underline{5, 4}, 3, \mathbf{6}$ produz a permutação:

$$\overrightarrow{\pi}' = (0, 2, 3, 4, 5, 6, 1, 7)$$

contraindo a sequência $2, 3, 4, 5, 6$, reduz a permutação $\overrightarrow{\pi}'$ em:

$$\overrightarrow{\pi}'' = (0, 2, 1, 3)$$

a qual tem só um obstáculo. Então, a sequência $\mathbf{2}, \underline{5, 4}, 3, \mathbf{6}$ é um obstáculo simples.

No entanto, a permutação $\overrightarrow{\pi} = (0, 2, 4, 3, 5, 1, 6, 8, 7, 9)$ também contém dois obstáculos, e quando se corta o obstáculo $\mathbf{2}, 4, 3, \mathbf{5}$, a permutação resultante depois de aplicar o corte, ordenar e reduzir é:

$$\overrightarrow{\pi}' = (0, 2, 1, 3, 5, 4, 6)$$

a qual ainda tem dois obstáculos: $\mathbf{0}, 2, 1, \mathbf{3}$ e $\mathbf{3}, 5, 6, \mathbf{6}$. Então, a sequência $\mathbf{2}, 4, 3, \mathbf{5}$ é um super-obstáculo.

A ideia do algoritmo é não aplicar um corte sobre super obstáculos desde que criariam novos obstáculos. A definição de obstáculos é a mesma utilizada em (2), onde os obstáculos são um subconjunto das componentes não orientadas. Foi demostrado em (2), que a aplicação de reversões sobre os obstáculos como indicado pelo Algoritmo 4, são seguras. Então, usando o Algoritmo 3, e Algoritmo 4 podemos ordenar otimamente qualquer permutação com sinal.

Algoritmo 4: Algoritmo para ordenar permutações com sinal com apenas elementos positivos.

Entrada: Uma permutação com sinal $\overrightarrow{\pi}$ com apenas elementos positivos.
Saída: Uma permutação $\overrightarrow{\pi}$ com sinal.

```
se uma permutação π tem 2k obstáculos (k ≥ 2) então
    mesclar dois obstáculos não consecutivos;
se uma permutação tem 2k + 1 obstáculos (k ≥ 1) então
    se tem um obstáculo simples então
        cortar o obstáculo;
    senão
        se k = 1 então
            mesclar dois obstáculos consecutivos;
        senão
            mesclar dois obstáculos não consecutivos;
```

Capítulo 5

Resultados Experimentais

5.1 Experimentos

A fim de validar os AGs propostos vários testes foram realizados. Desde testes usando permutações para as quais a distância de reversão é conhecida, até permutações geradas aleatoriamente utilizando diferentes enfoques.

Para cada teste feito, foi rodado o algoritmo de raio de aproximação 1.5, o AG padrão, e o AG hibrido. Todos estes algoritmos foram implementados na linguagem `C` e executados em plataformas `OS X` com processadores Intel core I5, I7, e outras plataformas similares.

Alguns parâmetros fixados para os AGs, que foram usados, são os seguintes: recombinação de ponto único (*single point crossover*), taxa de recombinação (*crossover rate*) de 0.9, e taxa de mutação (*mutation rate*) de 0.02, estes parâmetros foram fixados baseados nos resultados das Tabelas 5.5 e 5.6. A seleção dos pais para a recombinação foi feita sobre 60% da população que representam os melhores indivíduos. A substituição da descendência foi feita sobre o 60% da população que representam os piores indivíduos.

Para cada um dos testes foram gerados cem permutações de tamanho $i \in \{10, 20, 30, ..., 150\}$. E por cada permutação de tamanho i foi calculada a media, sobre as cem permutações geradas, para o algoritmo de raio de aproximação 1.5. Também foram calculadas as medias para o AG padrão e o AG hibrido.

É importante mencionar que para uma determinada permutação o AG padrão e o AG hibrido foram executados dez vezes e depois foi calculado a média sobre os dez resultados; esta média representa o resultado de cada um dos AGs para uma determinada permutação.

Adicionalmente, para um conjunto de cem permutações de tamanho i o desvio padrão sobre o algoritmo de aproximação, e sobre os AGs foi calculado, este valor representa quão longe estão os resultados da média.

5.1.1 Experimentos com Permutações de Gollan

Este experimento foi feito como mecanismo de controle, já que, as permutações de Gollan e suas inversas são as mais difíceis de ordenar usando reversões, e as únicas para as quais o número de reversões para ordená-las é conhecido.

Em (14) foi provado que no grupo simétrico S_n, só as permutações de Gollan, denotadas como γ_n, e sua inversa, γ_n^{-1}, precisam de exatamente $n-1$ reversões para ser ordenadas. Todas as permutações em S_n podem ser ordenadas com menos do que $n-1$ reversões.

Tabela 5.1: Resultados para o algoritmo de raio de aproximação 1.5, AG padrão e AG hibrido usando permutações de Gollan

n	**Tam. da Pop.**	**Média A. Aprox.**	**Média AG Padrão**	**Média AG Hibrido**
		γ_n/γ_n^{-1}	γ_n/γ_n^{-1}	γ_n/γ_n^{-1}
10	33	**9/11**	9/9	9/9
20	86	19/19	19/19	19/19
30	147	**29/31**	29/29	29/29
40	212	39/39	39/39	39/39
50	282	**49/51**	49/49	49/49
60	354	59/59	59/59	59/59
70	429	**69/71**	69/69	69/69
80	505	79/79	79/79	79/79
90	584	**89/91**	89/89	89/89
100	664	99/99	99/99	99/99
110	745	**109/111**	109/109	109/109
120	828	119/119	119/119	119/119
130	912	**129/131**	129/129	129/129
140	998	139/139	139/139	139/139
150	1084	**149/151**	149/149	149/149

As permutações de Gollan, em notação de ciclos (*cycle notation*), são definidas como segue:

$$\gamma_n = \begin{cases} (1,\ 3,\ 5,\ 7...n-1,\ n...8,\ 6,\ 4,\ 2), \text{ para } n \text{ par}, \\ (1,\ 3,\ 5,\ 7...n,\ n-1...8,\ 6,\ 4,\ 2), \text{ para } n \text{ impar}. \end{cases}$$

Por exemplo, para $n = 10$, $\gamma_{10} = (1, 3, 5, 7, 9, 10, 8, 6, 4, 2)$ and $\gamma_{10}^{-1} = (2, 4, 6, 8, 10, 9, 7, 5, 3, 1)$.

Sejam i e j dois elementos consecutivos da permutação de Gollan γ_n ou de sua inversa γ_n^{-1} em notação de ciclos, a permutação correspondente em notação de cadeia (*string notation*) é gerado colocando o elemento i na posição j, e o último elemento é colocado na posição dada pelo primeiro elemento. Por exemplo, γ_{10} em notação de cadeia é dado por $(2, 4, 1, 6, 3, 8, 5, 10, 7, 9)$ e sua inversa γ_{10}^{-1} por $(3, 1, 5, 2, 7, 4, 9, 6, 10, 8)$. Deve-se salientar que nos experimentos foram usadas permutações de Gollan em notação de cadeia.

Estas permutações foram usadas para validar os resultados fornecidos por nossas implementações do algoritmo de raio de aproximação 1.5 e os AGs, desde que soluções exatas para este tipo de permutações são conhecidas.

Os resultados deste experimento são mostrados na Tabela 5.1. Os valores que aparecem antes da barra (/) são os resultados para γ_n, e os que aparecem depois são os resultados para γ_n^{-1}. Note que todas as respostas dadas por ambos AGs, são exatos, enquanto o algoritmo de raio de aproximação 1.5 falha em computar soluções exatas para γ_n^{-1}, para $n = 10, 30, 50, 70, 90, 110, 130$ e 150.

Tabela 5.2: Resultados para o algoritmo de raio de aproximação 1.5, AG padrão e AG hibrido usando permutações geradas aleatoriamente

n	Tam. da População	Média Alg. Aproximação 1.5	Média AG Padrão	Média AG Hibrido	Desvio Padrão
10	33	5.84	5.83	5.98	0.0685
20	86	13.69	13.28	13.36	0.1775
30	147	21.88	21.08	21.07	0.3795
40	212	30.27	29.05	29.08	0.5682
50	282	39.64	37.53	37.46	1.0116
60	354	48.45	45.75	45.67	1.2921
70	429	57.56	54.28	54.17	1.5728
80	505	66.66	62.72	62.59	1.8887
90	584	75.86	71.65	71.54	2.0110
100	664	85.93	80.82	80.78	2.4184
110	745	94.03	88.91	88.87	2.4231
120	828	104.37	98.59	98.52	2.7414
130	912	113.38	107.46	107.37	2.8122
140	998	123.15	117	116.92	2.9182
150	1084	132.76	126.53	126.51	2.9416

5.1.2 Experimentos com Permutações Geradas Aleatoriamente

Neste experimento foram geradas permutações de tamanho n aplicando repetidamente a função `rand` (disponível na biblioteca padrão da linguagem `C`) sobre um conjunto de elementos $A = \{1, 2, 3, ..., n\}$ da seguinte forma: O primeiro elemento π_1 é gerado escolhendo uma posição aleatória dentro do conjunto A, o elemento que cai na posição aleatória é tirado do conjunto A e passa a ser o elemento π_1. Depois, para cada elemento $1 < i \leq n$, π_i é gerado analogamente como o primeiro elemento, eliminando-o do conjunto A depois de escolhido. Assumindo que, de fato, a função `rand` gera independentemente cada escolha aleatória, este mecanismo gera corretamente cada possível permutação com probabilidade de $1/n!$. Os resultados deste experimento são mostrados na Tabela 5.2.

Adicionalmente foi feito outro experimento com permutações de tamanho n, as quais foram geradas aplicando n reversões aleatórias sobre a permutação identidade. A motivação para realizar este experimento é por que outros trabalhos relacionados usaram este tipo de permutações como entrada, e forneceram resultados que sugeriram médias diferentes. Os resultados deste experimento são mostrados na Tabela 5.3.

Na primeira forma de gerar permutações o fato de escolher elementos aleatórios, faz com que a permutação fique mais desordenada do que só gerando-as da segunda forma, aplicando reversões sobre a identidade, já que é desconhecido o número de reversões implícitas que se está aplicando sobre a identidade, o que faz supor que é muito mais do que n reversões sobre a identidade. Está suposição pode ser conferida nos resultados da Tabela 5.2, e 5.3, onde as permutações são mais fáceis de ordenar gerando-as da segunda forma e portanto com resultados menores.

Tabela 5.3: Resultados para o algoritmo de raio de aproximação 1.5, AG padrão e AG hibrido usando permutações geradas aplicando reversões aleatórias sobre a identidade

n	Tam. da População	Média Alg. Aproximação 1.5	Média AG Padrão	Média AG Hibrido	Desvio Padrão
10	33	5.3	5.27	5.34	0.0287
20	86	12.01	11.72	11.77	0.1266
30	147	19.37	18.51	18.57	0.3920
40	212	27.2	25.82	25.85	0.6436
50	282	33.96	32.39	32.39	0.7401
60	354	42.12	39.72	39.69	1.1385
70	429	49.75	46.65	46.62	1.4685
80	505	57.3	53.95	53.88	1.5960
90	584	65.85	61.79	61.71	1.9330
100	664	73.64	69.26	69.11	2.1010
110	745	81.8	76.58	76.54	2.4702
120	828	89.34	84.23	84.15	2.4280
130	912	98.26	92.5	92.37	2.7464
140	998	105.49	99.8	99.72	2.7013
150	1084	114.04	108.04	107.98	2.8427

5.2 Testes Adicionais

5.2.1 Tempos de Execução dos Algoritmos

Para este experimento foram geradas permutações de tamanho $10, 50, 100, 150$ usando uma função aleatória sobre um conjunto de inteiros. Os tempos são dados em milissegundos.

Tabela 5.4: Tempos de execução do algoritmo de raio de aproximação 1.5, AG padrão e AG hibrido

n	Tamanho Pop. AG	Algoritmo Aproximação 1.5	AG Padrão	AG Hibrido
10	33	10.6530	0.8130	0.8580
50	282	50.5540	78.2309	79.6030
100	664	131.5091	665.1139	660.4431
150	1084	264.2081	2434.1100	2446.4791

5.2.2 Experimentos para Fixar os Parâmetros do AG Padrão

Para fixar o parâmetro de probabilidade de recombinação foram geradas 100 permutações de tamanho $10, 50, 100$, e 150; e para cada tamanho foi calculada a média usando o AG padrão. A probabilidade de recombinação varia nos valores $50\%, 60\%, 70\%, 80\%$, e 90%; enquanto o valor da probabilidade de mutação é fixada em 10% para este experi-

mento. Os resultados são mostrados na Tabela 5.5. A partir destes resultados podemos perceber que uma boa probabilidade de recombinação seria 90%.

Tabela 5.5: Resultados do AG padrão para diferentes probabilidades de recombinação, com probabilidade de mutação fixa 10%

prob. / **n**	**50%**	**60%**	**70%**	**80%**	**90%**
10	6.15	6.10	6.06	6.03	**6.00**
50	40.65	40.43	40.20	40.00	**39.82**
100	88.28	88.06	87.78	87.62	**87.42**
150	136.70	136.45	136.21	135.99	**135.83**

Para fixar o parâmetro de probabilidade de mutação foram geradas 100 permutações de tamanho $10, 50, 100$, e 150; e para cada tamanho foi calculada a média usando o AG padrão. A probabilidade de mutação varia nos valores $10\%, 8\%, 6\%, 4\%$, e 2%; enquanto o valor da probabilidade de recombinação é fixada em 70% para este experimento. Os resultados são mostrados na Tabela 5.6. A partir destes resultados podemos perceber que uma boa probabilidade de mutação seria 2%.

Tabela 5.6: Resultados do AG padrão para diferentes probabilidades de mutação, com probabilidade de recombinação fixa 70%

prob. / **n**	**10%**	**8%**	**6%**	**4%**	**2%**
10	5.885	5.893	**5.881**	5.885	5.912
50	39.908	39.495	38.889	38.188	**37.528**
100	87.634	86.931	85.955	84.450	**81.714**
150	136.339	135.492	134.357	132.410	**128.319**

5.3 Discussão

É importante salientar que nossos resultados mostrados nas Tabelas 5.1, 5.2, 5.3 também são apresentados em (20). As correções feitas sobre as imprecisões do Lema 1 do algoritmo de raio aproximação 1.5, foram apresentadas em (21).

Como foi previamente mencionado, da Tabela 5.1 pode-se ver que os AGs (padrão e hibrido) tem resultados exatos para as permutações de Gollan, em contraste com alguns resultados inexatos obtido pelo algoritmo de raio de aproximação 1.5. O único trabalho prévio que relata saídas para permutações específicas foi (13), nesse trabalho a permutação usada em S_{36} foi:

$$(12, 31, 34, 28, 26, 17, 29, 4, 9, 36, 18, 35, 19, 1, 16, 14, 32, 33,\\ 22, 15, 11, 27, 5, 20, 13, 30, 23, 10, 6, 3, 24, 21, 8, 25, 2, 7)$$

a qual foi ordenada usando 26 reversões, nossas implementações de AGs também obtiveram o mesmo resultado.

Algumas considerações são necessárias antes de discutir os nossos resultados para permutações geradas aleatoriamente. Aqui, é relevante destacar que não se tem muita informação sobre a forma das soluções para este problema. De fato, várias propriedades das saídas que são conhecidas para operações de ordenação diferentes das reversões, não são conhecidas para o caso das reversões; por exemplo, o número médio de reversões para ordenar permutações sem sinal em S_n é uma questão aberta interessante, enquanto que este número é conhecido para o problema de ordenação por intercâmbio de blocos (22).

Surpreendentemente, trabalhos anteriores relacionados relatam resultados contrastantes com permutações geradas aleatoriamente. A partir dos resultados obtidos em (11), (13) e (12), as seguintes porcentagens do número de reversões sobre o comprimento das permutações, necessários para ordenar permutações de tamanhos médios (entre 50 e 150), são sugeridos: $\sim 83\%$, $\sim 46\%$ and $\sim 80\%$ respectivamente. Especificamente, a maneira pela qual as permutações aleatórias de entrada foram geradas em (13) não foi reportada nesse artigo, mas em vista dos resultados de outros trabalhos, incluindo o nosso, podemos dizer que o número médio de reversões necessárias para ordenar permutações de tamanho n, é muito maior do que $n/2$, o que nos sugere que os autores desse trabalho usaram um mecanismo muito peculiar na geração das permutações de entrada.

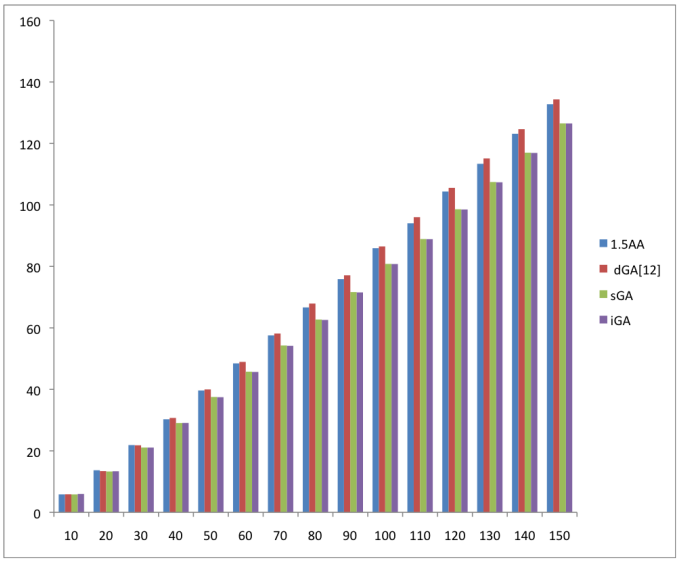

Figura 5.1: Comparação dos resultados para o Alg. de raio de aprox. 1.5 corrigido(1.5AA), o AG simples em (21) (dGA), e o AG padrão e hibrido(sGA and iGA) em (20). Onde o eixo horizontal são o comprimento das permutações, e o eixo vertical o número de reversões

Adicionalmente, trabalhos anteriores compararam os resultados de seus AGs com implementações do algoritmo de raio de aproximação 1.5, mas sem relatar as imprecisões na atualização do grafo de reversões no artigo original de Christie (10), estas imprecisões foram corrigidas e posteriormente publicadas em nosso trabalho (21), veja-se seção 3.3.1. Por exemplo, as saídas aproximadas em (11) para permutações de tamanho médio (entre 50 e 150) sugerem uma porcentagem de $\sim 94\%$ do número de reversões sobre o comprimento de permutações de tamanho n, enquanto que o algoritmo de raio de aproximação 1.5 corrigido, usado em nossos artigos (21) e (20) tem uma porcentagem de $\sim 87\%$, permi-

tindo deste modo uma análise mais razoável da verdadeira melhora nos resultados obtidos pelos nossos AGs sobre o algoritmo de aproximação.

Os resultados para permutações geradas aleatoriamente mostrados na Tabela 5.2 só podem ser comparados com aqueles obtidos pelo AG simples proposto em (21), os quais são mostrados na Figura 5.1. Os resultados relatados em (21) são próximos dos obtidos pelo algoritmo de raio de aproximação 1.5 corrigido, no entanto, os resultados dos AG padrão e hibrido fornecem melhores resultados do que o algoritmo de aproximação.

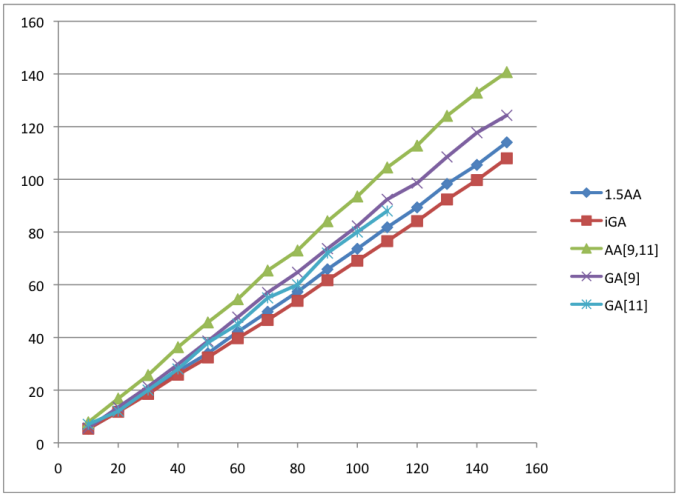

Figura 5.2: Comparação dos resultados para o Alg. de raio de aprox. 1.5 corrigido(1.5AA), o AG hibrido(iGA), o Alg. aprox. em (AA(11, 12)), o AG em (11), e o AG em (12). Onde o eixo horizontal são o comprimento das permutações, e o eixo vertical o número de reversões

Os resultados para permutações geradas aplicando reversões aleatórias sobre a permutação identidade, que são mostrados na Tabela 5.2, só podem ser comparados com precisão com aqueles apresentados em (11), já que é o único trabalho que fornece uma tabela com resultados numéricos. Comparações com os resultados em (12) são apresentadas, mas são imprecisas porque em vez de apresentar valores numéricos, só foi apresentado uma imagem gráfica (restrito a permutações de tamanho menor ou igual a 110) da qual foi extraída valores numéricos aproximados e incluídos na Figura 5.2. Nesta figura, foram construídos gráficos para os algoritmos aproximados usados em (11), (12) e em nosso trabalho em (20), também é mostrado um gráfico para o AG de Auyeung, cujos valores numéricos foram tomados diretamente de (11). Pode-se observar que todos os enfoques se desempenham melhor do que os algoritmos aproximados apresentados em (11) e (12), mas só os resultados dos AGs padrão e hibrido são melhores do que os resultados do algoritmo de raio de aproximação 1.5 corrigido. Neste ponto, observamos que a taxa de recombinação e mutação que nós usamos foi de 0.9 e 0.02, respectivamente, enquanto que em (11) estas taxas são de 0.3 e 0.8, respectivamente. Uma taxa de recombinação muito pequena e uma taxa de mutação muito alta afetam diretamente os resultados no experimento.

O desvio padrão dos resultados também foi incluído nas Tabelas 5.2 e 5.3. Pode-se observar que quando o comprimento das permutações cresce, o desvio padrão também cresce; isto indica que a medida que o comprimento das permutações aumenta, os resultados estão cada vez mais distantes da média. Isto também indica, que os resultados fornecidos

pelos AGs estão mais distantes dos resultados obtidos pelo algoritmo de aproximação, e desde que os AGs sempre tem melhores resultados, podemos dizer que os resultados dos AGs superam os resultados do algoritmo de raio de aproximação 1.5.

Em resumo, a partir dos experimentos, se pode concluir que os AGs padrão e hibrido obtiveram na média melhores resultados do que os obtidos pelo algoritmo de raio de aproximação 1.5. Estes resultados são melhores do que os relatados anteriormente pelo AG simples em (21), o qual está baseado puramente na eliminação de pontos de quebra em cada geração. Também, os AGs padrão e hibrido superam os resultados dos AGs apresentados em (11) e (12). Finalmente, pode-se observar que para permutações de comprimento maior ou igual a 50, o AG hibrido supera ao AG padrão.

Capítulo 6

Conclusões e Trabalhos Futuros

Caprara (8) demonstrou que o problema de ordenação de permutações sem sinal por reversões é $\mathcal{NP}$-difícil e a questão de se o problema é $\mathcal{NP}$-completo ainda está em aberto. Devido a sua complexidade foram propostos algoritmos de aproximação, tanto como algoritmos bio-inspirados, os quais dão resultados aproximados à solução ótima. Então, surge a necessidade de melhorar estes algoritmos aproximados, ou bio-inspirados, com objetivo de computar ordenações que possam ser ainda mais próximas à solução ótima.

Neste trabalho foi proposto um algoritmo genético baseado no método de Auyeung e Abraham (11), para resolver o problema de ordenação de permutações sem sinal por reversões. O método foi aprimorado incluindo uma heurística de eliminação de pontos de quebra, muito usada no desenvolvimento de algoritmos de aproximação ((14), (2)). A combinação desta heurística com a função de aptidão baseada no cálculo de soluções exatas para permutações com sinal, usando o algoritmo de tempo linear em (9), representam a característica distintiva do AG híbrido proposto.

Para o desenvolvimento do método e execução de testes experimentais que permitiram verificar a alta qualidade das respostas fornecidas foi necessário realizar o seguinte.

- Modificação/correção e implementação de um algoritmo de raio de aproximação 1.5 para o controle de qualidade das respostas fornecidas pelo método proposto. Verificaram-se erros na formulação do método em (10) não reportados por autores de AGs que o utilizaram como mecanismo de controle. Dessa forma foi realizado um estudo teórico do método e modificações foram propostas para atingir um algoritmo aproximado que computa corretamente soluções de raio 1.5. O mecanismo foi implementado e efetivamente apresenta soluções sempre melhores que àquelas dos mecanismos de controle utilizados na literatura relacionada. Resultados foram publicados em (21).

- Sobre a conjugação de módulos para obter uma implementação do método: primeiro, para o cálculo da função de aptidão do AG foi usado o código fonte do algoritmo de tempo linear para o problema de ordenação de permutações com sinal, disponibilizado pelos autores em (9). Para incluir esse algoritmo teve que ser isolada a parte referente a distância de reversão para permutações com sinal, já que estava dentro de um contexto de reconstrução de árvores filogenéticas. Depois, foram implementados os AGs baseados no método de Auyeung e Abraham, incluindo a heurística de eliminação de pontos de quebra, para os quais os parâmetros do AG

foram customizados de forma a obter melhores resultados. O método e resultados experimentais foram publicados em (20).

- Para garantir que os algoritmos dão bons resultados para qualquer tipo de permutação foram geradas aleatoriamente de duas formas: escolhendo aleatoriamente elementos distintos de um conjunto de números, ou aplicando reversões aleatórias sobre uma permutação ordenada. Adicionalmente, incluímos as permutações de Gollan, as quais foram utilizadas como mecanismo de controle dos nossos resultados desde que sabemos que tem as piores soluções exatas, como foi demonstrado por Bafna e Pevzner em (14). A geração de permutações em diferentes formas surgiu da necessidade de fazer um melhor estudo da qualidade dos resultados, já que não era claro como outros trabalhos prévios relacionados geravam suas permutações de entrada.
- Os parâmetros de taxa de recombinação, e mutação dos AGs propostos foram estabelecidos adequadamente (taxa de recombinação = 90%, taxa de mutação = 2%) de acordo com os experimentos feitos com este proposito (ver Tabelas 5.5 e 5.6), estes parâmetros estão dentro do sugerido na literatura, por exemplo em (16), onde uma taxa de recombinação de 100% faz com que todos os indivíduos de uma descendência sejam criados por recombinação, uma taxa de recombinação de 0% faz como que os indivíduos de uma nova geração sejam exatamente iguais a geração anterior. Uma taxa de mutação de 100% faz com que todos os elementos de um indivíduo (solução) mudem, uma taxa de mutação de 0% faz com que os elementos de um indivíduo não mudem. A mutação não deve acontecer muitas vezes por que senão o AG passaria a ser uma busca aleatória.

Os experimentos mostraram o seguinte.

- Os mecanismos de controle de qualidade utilizados na literatura relacionada não forneciam respostas aproximadas de raio 1.5.
- O AG simples (21), baseado em eliminação de pontos de quebra, apenas apresenta respostas aceitáveis, mas sempre piores que as fornecidas pelo algoritmo de raio de aproximação 1.5 corrigido, da mesma maneira que todos os AGs apresentados na literatura (11, 12)).
- Os resultados da nossa implementação (21, 20) do algoritmo de raio de aproximação 1.5 corrigido superam as outras implementações (11, 12) do algoritmo aproximado que não relataram os erros na formulação deste algoritmo.
- O AG padrão que aprimora os parâmetros propostos em (11) e o método (AG) híbrido fornecem respostas melhores do que o algoritmo 1.5-aproximado corrigido, tanto como outros trabalhos relacionados anteriores.

É necessário ressaltar, que o cálculo da função de aptidão está restrita ao cálculo da distância de reversão sem calcular a sequência de reversões para ordenar a permutação. Desde o ponto de vista biológico, isto não é uma desvantagem, já que não é necessário saber a sequência de ordenação para fazer uma reconstrução filogenética; além disso, esta sequência não aporta muito significado, já que, existem muitas sequências diferentes ótimas de ordenação. Desde o ponto de vista combinatório, a construção de sequências

ótimas é de grande interesse e sua implementação implica um aumento na complexidade do algoritmo. A fim de obter as sequências, se pode aplicar o algoritmo sub-quadrático apresentado por Swenson et al (23), que demora um tempo de $O(n\ log\ n)$ para a maioria das permutações com sinal.

Como trabalhos futuros temos o seguinte.

- A construção de sequências ótimas para o problema de ordenação por reversões de permutações sem sinal, desde que em nosso trabalho só calculamos a distância de reversão, sem considerar as possíveis sequências.
- Estudo e implementação do algoritmo de raio de aproximação 1.375, já que ainda não existe uma implementação para este algoritmo teórico.
- Hibridização dos nossos AGs com outros algoritmos de aproximação, com o objetivo de melhorar a eliminação de pontos de quebra em gerações iniciais.

Referências

[1] Vineet Bafna and Pave A. Pevzner*j. Sorting by reversals: Genome rearrangements in plant organelles and evolutionary history of x chromosome. *Mol. Biol. and Evol*, 12:239–246, 1995. 1

[2] Sridhar Hannenhalli and Pavel Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, STOC '95, pages 178–189, New York, NY, USA, 1995. ACM. 1, 14, 37, 39, 40, 41, 42, 52

[3] P.A. Pevzner and M.S. Waterman. Open combinatorial problems in computational molecular biology. In *Theory of Computing and Systems, 1995. Proceedings., Third Israel Symposium on the*, pages 158 –173, jan 1995. 1

[4] David Sankoff, Robert Cedergren, and Yvon Abel. [26] genomic divergence through gene rearrangement. volume 183 of *Methods in Enzymology*, pages 428 – 438. Academic Press, 1990. 1

[5] David Sankoff. Edit distance for genome comparison based on non-local operations. In Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Manber, editors, *Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 121–135. Springer Berlin Heidelberg, 1992. 1

[6] D Sankoff, G Leduc, N Antoine, B Paquin, B F Lang, and R Cedergren. Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. *Proceedings of the National Academy of Sciences*, 89(14):6575–6579, 1992. 1

[7] John Kececioglu and David Sankoff. Exact and approximation algorithms for the inversion distance between two chromosomes. In Alberto Apostolico, Maxime Crochemore, Zvi Galil, and Udi Manber, editors, *Combinatorial Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, pages 87–105. Springer Berlin / Heidelberg, 1993. 10.1007/BFb0029799. 1, 4, 14, 15

[8] Alberto Caprara. Sorting by reversals is difficult. In *Proceedings of the first annual international conference on Computational molecular biology*, RECOMB '97, pages 75–83, New York, NY, USA, 1997. ACM. 2, 15, 52

[9] David A. Bader, Bernard M. E. Moret, and Mi Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Roberto Tamassia, editors,

WADS, volume 2125 of *Lecture Notes in Computer Science*, pages 365–376. Springer, 2001. 2, 14, 36, 37, 39, 52

[10] David A. Christie. A 3/2-approximation algorithm for sorting by reversals. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, SODA '98, pages 244–252, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics. 2, 15, 16, 17, 23, 24, 31, 36, 49, 52

[11] A. Auyeung and Ajith Abraham. Estimating genome reversal distance by genetic algorithm. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 2, pages 1157 – 1161 Vol.2, dec. 2003. 2, 3, 15, 16, 31, 35, 49, 50, 51, 52, 53

[12] A. Ghaffarizadeh, K. Ahmadi, and N.S. Flann. Sorting unsigned permutations by reversals using multi-objective evolutionary algorithms with variable size individuals. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 292 –295, june 2011. 3, 16, 31, 49, 50, 51, 53

[13] Mo Zhongxi and Zeng Tao. An improved genetic algorithm for problem of genome rearrangement. *Wuhan University Journal of Natural Sciences*, 11:498–502, 2006. 10.1007/BF02836651. 3, 16, 31, 48, 49

[14] V. Bafna and P.A. Pevzner. Genome rearrangements and sorting by reversals. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 148 –157, nov 1993. 4, 6, 14, 15, 21, 36, 44, 52, 53

[15] K.A. De Jong. *Evolutionary computation: a unified approach.* Bradford Books. MIT Press, 2006. 10

[16] S.N. Sivanandam and S.N. Deepa. *Introduction to genetic algorithms.* Springer, 2007. 10, 11, 13, 53

[17] Anne Bergeron. A very elementary presentation of the hannenhalli–pevzner theory. *Discrete Applied Mathematics*, 146(2):134 – 145, 2005. <ce:title>12th Annual Symposium on Combinatorial Pattern Matching</ce:title>. 14, 39, 40

[18] Piotr Berman and Sridhar Hannenhalli. Fast sorting by reversal. In Daniel S. Hirschberg and Eugene W. Myers, editors, *CPM*, volume 1075 of *Lecture Notes in Computer Science*, pages 168–185. Springer, 1996. 14

[19] Piotr Berman, Sridhar Hannenhalli, and Marek Karpinski. 1.375-approximation algorithm for sorting by reversals. In *Proceedings of the 10th Annual European Symposium on Algorithms*, ESA '02, pages 200–210, London, UK, UK, 2002. Springer-Verlag. 15

[20] Jose Luis Soncco-Alvarez and Mauricio Ayala-Rincon. Sorting permutations by reversals through a hybrid genetic algorithm based on breakpoint elimination and exact solutions for signed permutations. *Electronic Notes in Theoretical Computer Science*, 2013. 48, 49, 50, 53

[21] J.L. Soncco-Alvarez and M. Ayala-Rincon. A genetic approach with a simple fitness function for sorting unsigned permutations by reversals. In *Computing Congress (CCC), 2012 7th Colombian*, pages 1 –6, oct. 2012. 48, 49, 50, 51, 52, 53

[22] Miklós Bóna and Ryan Flynn. The average number of block interchanges needed to sort a permutation and a recent result of stanley. *Inf. Process. Lett.*, 109(16):927–931, 2009. 49

[23] Krister Swenson, Vaibhav Rajan, Yu Lin, and Bernard Moret. Sorting signed permutations by inversions in o (nlogn) time. In Serafim Batzoglou, editor, *Research in Computational Molecular Biology*, volume 5541 of *Lecture Notes in Computer Science*, pages 386–399. Springer Berlin / Heidelberg, 2009. 54