UNIVERSIDADE D
COIMBRA

Diogo André Cardoso Conde Soares

COMPUTAÇÃO DO GRAFO DAS CLASSES DE COMUTAÇÃO PARA A MAIOR PERMUTAÇÃO COM SINAL

Dissertação no âmbito do Mestrado em Matemática, Ramo Análise Aplicada e Computação orientada pelo Professor Doutor Ricardo Mamede e pelo Professor Doutor José Luís Santos e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia.

Computação do grafo das classes de comutação para a maior permutação com sinal

Diogo André Cardoso Conde Soares

UNIVERSIDADE D
COIMBRA

Mestrado em Matemática

Master in Mathematics

Janeiro 2021 / January 2021

Agradecimentos

Em primeiro lugar quero agradecer ao Professor José Luís Santos e ao Professor Ricardo Mamede por todo o apoio, dedicação e disponibilidade que demonstraram ao longos destes últimos meses.

Quero também agradecer à Filipa que me apoiou sempre durente todo este processo, fosse nos momentos bons ou nos momentos menos bons e que nunca me deixou ir abaixo qualquer que fosse a circunstância.

Ao Gabriel, ao Fábio, ao Pedro, à Axelle e ao Bruno um grande obrigado por estarem sempre comigo já há tantos anos e por todos os bons momentos que passámos.

Também quero agradecer a todos os meus colegas da universidades. Sem vocês este percurso não teria sido o mesmo.

Gostaria também de agradecer aos meus companheiros na música, o João, o Eduardo, o Tomás e o Bruno por todos os concertos que demos e por todas as maluqueiras que fizémos.

Por último quero agradecer às minhas tias, à minha mãe e aos meus avós que também sem eles nada disso seria possível.

O trabalho aqui desenvolvido foi parcialmente realizado no âmbito do projeto MobiWise: From mobile sensing to mobility advising (P2020 SAICTPAC / 0011/2015), co-financiado pelo COMPETE 2020, Portugal 2020 - Programa Operacional de Competitividade e Internacionalização (POCI), do ERDF (Fundo Europeu de Desenvolvimento Regional) da União Europeia e da Fundação portuguesa para a Ciência e Tecnologia (FCT).

Resumo

Usando a apresentação de Coxeter padrão para o grupo das simetrias com sinal $\mathfrak{S}_{n+1}^B$ em $n+1$ letras, duas expressões reduzidas de uma permutação com sinal estão na mesma classe de comutação se uma se poder obter da outra aplicando uma sequência finita de comutações. As classes de comutação de uma dada permutação com sinal podem ser vistas como os vértices de um grafo, chamado grafo das classes de comutação, onde duas classes estão ligadas por uma aresta se existem elementos nessas classes que diferem por uma relação longa. O foco desta tese será o grafo das classes de comutação para a maior permutação com sinal onde vão ser exploradas diversas propriedades, mais especificamente, será calculado o seu diâmetro e raio através da construção de uma função que irá particionar este grafo em várias camadas. Vamos também mostrar que este grafo não é planar para $n > 2$, usando o famoso Teorema de Wagner, e vamos identificar todas as classes de comutação que possuem apenas um elemento. Estes grafos são estruturas que possuem grandes dimensões, o que torna a sua construção manual inexequível. Deste modo, foi desenvolvida uma aplicação que permitiu visualizar o grafo para alguns valores de n, bem como explorar algumas das suas propriedades. Para tal foi necessário construir um algoritmo que permitisse gerar todas as classes de comutação para alguns valor de n. Na parte final deste trabalho serão descritos todos os algoritmos usados para gerar todas as classes de comutação, algo que foi de extrema importância na obtenção dos resultados teóricos.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os *grupos de Coxeter* são uma família de grupos que são gerados por um conjunto de involuções S (elementos de ordem 2), também denominados por reflexões simples, que satisfazem certas relações entre si. Estes grupos são estudados em álgebra, geometria e combinatória, mas também são usados noutras áreas da Matemática. Qualquer elemento w de um grupo de Coxeter W pode ser escrito como um produto $\mathrm{w} = s_{i_1} s_{i_2} \cdots s_{i_l}$, com $s_{i_j} \in S$. Se l for minimal dizemos que w tem *comprimento* $l(\mathrm{w}) := l$ e que $s_{i_1} s_{i_2} \cdots s_{i_l}$ é uma *decomposição reduzida*. À palavra $a = i_1 i_2 \cdots i_l$ chamamos *palavra reduzida* de w. Todo o grupo de Coxeter possui um único elemento de comprimento maximal [5] que é denotado por w_0. Em geral, existem várias decomposições reduzidas de w, sendo que estas estão ligadas entre si através das relações que existem entre os geradores do grupo. Deste modo, podemos definir o grafo $G(\mathrm{w})$ das palavras reduzidas de w que possui como vértices o conjunto das palavras reduzidas de w, denotado por $R(\mathrm{w})$, e uma aresta entre duas palavras sempre que for possível obter uma através da outra usando alguma relação entre os seus geradores. Tits [25] mostrou que este grafo é conexo qualquer que seja $\mathrm{w} \in W$, sendo este resultado também uma consequência imediata do Teorema de Matsumoto [12].

Um exemplo de um grupo de Coxeter é o grupo simétrico $\mathfrak{S}_{n+1}$ formado por todas as permutações do conjunto $[n+1] := \{1, 2, \ldots, n+1\}$. Este grupo é gerado por todas as transposições de inteiros consecutivos e está categorizado como um grupo de Coxeter do tipo A [5]. No caso do grupo simétrico, sabe-se um pouco mais sobre o grafo das palavras reduzidas de algumas permutações, mais especificamente sobre $G(\mathrm{w}_0^{A_n})$, o grafo das palavras reduzidas para a maior permutação do grupo simétrico. Stanley [21] determinou o número de palavras reduzidas de $\mathrm{w}_0^{A_n}$ usando funções simétricas, que têm conexões com o cálculo de Schubert, caracteres de Demazure e funções de Schur [1, 4, 18]. É conhecido também o diâmetro de $G(\mathrm{w}_0^{A_n})$ [19], e sabe-se que é bipartido [2].

A partir do grafo $G(\mathrm{w})$ podemos obter o grafo das classes de comutação $C(\mathrm{w})$ através da contração das arestas que unem palavras reduzidas de w sempre que estas diferirem por uma comutação de geradores. Podemos também obter este grafo estabelendo uma relação $\sim$ no conjunto $R(\mathrm{w})$ das palavras reduzidas de w fazendo $a \sim b$ se b difere por uma sequência de comutações aplicadas a a. Esta relação é de equivalência e as classes que gera são denominadas *classes de comutação*. Para $\mathrm{w} \in \mathfrak{S}_{n+1}$, Elnitsky [11] estabeleceu uma bijeção entre $C(\mathrm{w})$ e pavimentações de certos polígonos e mostrou que este grafo é bipartido, e em [15] foi calculado o diâmetro de $C(\mathrm{w})$. No caso particular da

maior permutação do grupo simétrico $\mathrm{w} = \mathrm{w}_0^{A_n}$ foi ainda calculado o raio de $C(\mathrm{w})$, foi estudado a sua planaridade e determinado todas as classes com um elemento, os chamados átomos. [14].

Os grafos $G(\mathrm{w})$ e $C(\mathrm{w})$ são uma ferramenta importante em problemas enumerativos de permutações que contêm ou evitam determinados padrões. Por exemplo, as palavras reduzidas de permutações que evitam o padrão 2143 são enumeradas pelo número de tableaux de Young de um determinado formato, e as permutações que evitam o padrão 321 estão ligadas apenas por comutações de geradores [8, 23, 24]. O grafo $C(\mathrm{w})$ tem ainda ligações à teoria geométrica da representação [7].

Um outro exemplo de um grupo de Coxeter é o grupo hiperoctaedral $\mathfrak{S}_{n+1}^B$, que é formado por todas as permutações w do conjunto $[\pm n+1] := \{-(n+1), -n, \ldots, -1, 1, \ldots, n+1\}$ tais que $\mathrm{w}(-k) = -\mathrm{w}(k)$ para todo $k \in [n+1]$. É um grupo de Coxeter do tipo B e os seus elementos designam-se por *permutações com sinal*. Stanley conjeturou em [21] o número de palavras reduzidas para a maior permutação com sinal, denotada por $\mathrm{w}_0^{B_n}$, conjetura essa que foi provada em [16] através de uma bijeção entre as palavras reduzidas de $\mathrm{w}_0^{B_n}$ e tableaux standard de formato "shift". As informações sobre o grafo das palavras reduzidas de permutações com sinal, e mais concretamente do grafo das classes de comutação são muito escassas pelo que nesta tese vamos estudar algumas propriedades do grafo $C(\mathrm{w}_0^{B_n})$, mais concretamente o cálculo do seu raio e diâmetro, analisar a sua planaridade e determinar todos os seus átomos. Nos Capítulos 4 e 5 vamos mostrar que o diâmetro e o raio de $C(\mathrm{w}_0^{B_n})$ são dados pela expressão $\dfrac{n(n+1)(4n-1)}{6}$, generalizando alguns resultados de [14] usados no cálculo do diâmetro e raio do grafo $C(\mathrm{w}_0^{A_n})$. O Capítulo 6 destina-se ao estudo da planaridade de $C(\mathrm{w}_0^{B_n})$ e ao cálculo dos seus átomos, onde iremos mostrar que $C(\mathrm{w}_0^{B_n})$ não é planar para $n > 2$ e que possui apenas 2 átomos para $n \geq 1$. De salientar que com estes resultados teóricos foi escrito um artigo [17] que está submetido numa revista internacional com sistema de arbitragem, estando disponível um pre-print na plataforma do CMUC. O último capítulo é reservado para a parte computacional deste trabalho, mais concretamente analisar os algoritmos usados para gerar as classes de comutação e explorar a aplicação que foi desenvolvida para visualizar o grafo $C(\mathrm{w}_0^{B_3})$, algo que foi indispensável para obtenção dos resultados teóricos.

Capítulo 2

Preliminares

Um *grupo de Coxeter* é um grupo W com uma apresentação da forma

$$\left\langle S \mid s^2 = 1 \text{ para todo } s \in S \text{ e } (st)^{m(s,t)} = 1 \text{ para todo } s,t \in S \right\rangle,$$

onde S é um conjunto finito e $m(s,t) = m(t,s) \in \{2,3,\ldots\} \cup \{\infty\}$, com $m(s,t) = \infty$ se não houver nenhuma relação entre s e t. Os elementos de S designam-se por *reflexões simples*. Como s e t são elementos de ordem 2, a relação $(st)^{m(s,t)} = 1$ pode ser escrita da forma

$$\underbrace{ststs\cdots}_{m(s,t) \text{ letras}} = \underbrace{tstst\cdots}_{m(s,t) \text{ letras}} \tag{2.1}$$

para $m(s,t) \geq 2$. Se $m(s,t) = 2$, então a relação $st = ts$ é chamada de *comutação* ou *relação curta*. Caso contrário, se $m(s,t) \geq 3$, então a relação (2.1) é denominada *relação longa*.

Sendo S um conjunto gerador de W, qualquer elemento $\mathrm{w} \in W$ pode ser escrito como um produto $\mathrm{w} = s_{i_1} s_{i_2} \cdots s_{i_l}$ com $s_{i_j} \in S$. Quando l é minimal, dizemos que $l(\mathrm{w}) := l$ é o *comprimento* de w e que $s_{i_1} s_{i_2} \cdots s_{i_l}$ é uma *decomposição reduzida* de w. Para simplificar a escrita vamos representar o gerador s_{i_j} pelo seu índice e dizemos que $a = i_1 i_2 \cdots i_l$ é uma *palavra reduzida* de w, denotando por $R(\mathrm{w})$ o conjunto formado por todas as palavras reduzidas de w. Estas palavras relacionam-se entre si através das relações (2.1), pelo que podemos definir o grafo $G(\mathrm{w})$ das palavras reduzidas w como sendo o grafo que tem como vértices o conjunto $R(\mathrm{w})$ e uma aresta entre duas palavras sempre que estas diferirem por uma relação da forma (2.1).

O grafo $C(\mathrm{w})$ das classes de comutação de w obtém-se de $G(\mathrm{w})$ contraindo as arestas que ligam palavras que diferem por uma relação de comutação. Este processo particiona o conjunto $R(\mathrm{w})$ em classes de equivalência, que são chamadas *classes de comutação*, sendo que duas palavras pertencem à mesma classe se e só se diferem por uma sequência de comutações. Denotamos por $[a]$ a classe de comutação de $a \in R(\mathrm{w})$ e escrevemos $a \sim b$ sempre que b diferir por uma sequência de comutações aplicada a a. Na Figura 2.1 está representado o grafo $G(\mathrm{w}_0^{A_3})$, sendo que as arestas a negrito representam comutações, e as outras relações longas. Note-se que $C(w_0^{A_3})$ é obtido do grafo da Figura 2.1 contraindo as arestas que estão a negrito, ou seja juntar os pares de vértices que são unidos por essas arestas num só que vai ficar ligado aos vértices que estavam ligados aos pares de vértices originais.

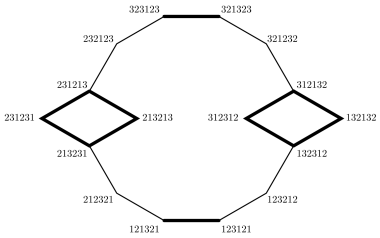

Figura 2.1 O grafo $G(\mathrm{w}_0^{A_3})$.

A *distância* $d(\mathrm{u},\mathrm{v})$ entre os vértices u e v de um grafo G define-se como o comprimento do caminho mais curto que une u com v. A *excentricidade* de u é a distância entre u e o vértice que está mais distante de u. O *raio* e o *diâmetro* de G são, respetivamente, os valores mínimo e máximo das excentricidades dos seus vértices. Nesta tese iremo-nos centrar no grafo das classes de comutação para a maior permutação com sinal, sendo necessário introduzir os grupos de Coxeter onde iremos trabalhar.

Dado $n \geq 2$, denotamos por $\mathfrak{S}_{n+1}$ o grupo simétrico formado por todas as permutações do conjunto $[n+1] := \{1, 2 \cdots n+1\}$, com a composição de funções (lida da direita para a esquerda) como operação de grupo. Usaremos a notação em linha $\mathrm{w} = (\mathrm{w}(1), \mathrm{w}(2), \ldots, \mathrm{w}(n+1))$ ou a notação cíclica clássica para representar um elemento $\mathrm{w} \in \mathfrak{S}_{n+1}$. O grupo $\mathfrak{S}_{n+1}$ é gerado pelas reflexões simples $\{s_1^A, s_2^A, \cdots, s_n^A\}$, onde s_i^A corresponde à transposição $(i\, i+1)$, que satisfazem as seguintes relações:

$$s_i^A s_j^A = s_j^A s_i^A, \ |i-j| > 1 \text{ e } i, j \in [n] \tag{2.2}$$

$$s_i^A s_{i+1}^A s_i^A = s_{i+1}^A s_i^A s_{i+1}^A, \ i \in [n-1]. \tag{2.3}$$

Note-se que (2.2) é uma comutação e (2.3) uma relação longa.

O grupo $\mathfrak{S}_{n+1}$ atua sobre si próprio por multiplicação à direita, pelo que o produto $\mathrm{w}s_i^A$ transpõe os valores nas posições i e $i+1$ de w.

Exemplo 2.1. Seja $\mathrm{v} = (3,2,4,1) \in \mathfrak{S}_4$. Então, $\mathrm{v}\, s_2 = (3, \mathbf{4}, \mathbf{2}, 1)$.

A permutação $\mathrm{w}_0^{A_n} := (n+1, n, \ldots, 1)$ é a maior permutação do grupo simétrico $\mathfrak{S}_{n+1}$, cujo comprimento é $\binom{n+1}{2}$. Esta permutação tem sido objeto de particular interesse, uma vez que as suas palavras reduzidas estão relacionadas com a ordem fraca de Bruhat e arranjos de hiperplanos. O grafo $G(\mathrm{w}_0^{A_n})$ foi amplamente estudado por diversos autores sendo que é conhecido o número de palavras reduzidas de $\mathrm{w}_0^{A_n}$ [21], o diâmetro de $G(\mathrm{w}_0^{A_n})$ [19], e que este grafo é conexo [25] e bipartido [2]. Relativamente ao grafo $C(\mathrm{w}_0^{A_n})$ das classes de comutação, sabe-se que este é conexo e bipartido [14, 25] e têm-se uma expressão para o seu raio e diâmetro [14]. Para além destas propriedades, pouco mais se sabe sobre este grafo. Por exemplo, não é conhecido o número de classes de comutação de $\mathrm{w}_0^{A_n}$ [9].

Dado $n \geq 1$ denotamos por $\mathfrak{S}_{n+1}^B$ o grupo das simetrias com sinal (ou grupo hiperoctaedral) de $[n+1]$ formado por todas as permutações w do conjunto $[\pm(n+1)] := \{\pm 1, \pm 2, \ldots, \pm(n+1)\}$ tais que $\mathrm{w}(-k) = -\mathrm{w}(k)$ para todo $k \in [\pm(n+1)]$, com a composição de funções (lida da direita para a

esquerda) como operação de grupo. Este grupo pode ser realizado como o grupo das simetrias de um hipercubo [13]. É um grupo de Coxeter do tipo B e os seus elementos designam-se por *permutações com sinal* ou *permutações do tipo B*. [5]

O grupo $\mathfrak{S}^B_{n+1}$ pode ser visto como um sub-grupo de $\mathfrak{S}_{2n+2}$, pelo que podemos usar a notação em linha $\mathrm{w} = (\overline{\mathrm{w}(n+1)}, \cdots, \overline{\mathrm{w}(1)}, \mathrm{w}(1), \cdots \mathrm{w}(n+1))$ e a notação cíclica para representar uma permutação com sinal w, onde para facilitar a notação escrevemos $\overline{\mathrm{w}(k)}$ para representar $-\mathrm{w}(k)$. Como w é completamente determinada pelos seus valores no conjunto $[n+1]$, podemos ainda usar a notação $\mathrm{w} = [\mathrm{w}(1), \mathrm{w}(2), \ldots, \mathrm{w}(n+1)]$, chamada *notação em janela* [5]. Iremo-nos referir a $\mathrm{w}(i)$, como sendo a i-ésima letra da notação em janela de w, com $i \in [n+1]$, e vamos considerar o conjunto $[0,n] := [n] \cup \{0\}$.

Exemplo 2.2. Seja $\mathrm{w} \in \mathfrak{S}^B_4$ com $\mathrm{w} = (\overline{3}, 4, 2, \overline{1}, 1, \overline{2}, \overline{4}, 3)$ a sua notação em linha. Então $\mathrm{w} = (2\ \overline{2})(3\ \overline{4}\ \overline{3}\ 4)$ é a sua notação cíclica e $\mathrm{w} = [1, \overline{2}, \overline{4}, 3]$ a sua notação em janela.

O grupo $\mathfrak{S}^B_{n+1}$ é gerado pelas involuções $\{s_0, s_1, \ldots, s_n\}$ com $s_i := [1, \ldots, i+1, i, \ldots, n+1]$ para $1 \leq i \leq n$ e $s_0 := [\overline{1}, 2, \ldots, n+1]$. Estes geradores satisfazem as seguintes relações:

$$s_i s_j = s_j s_i, \ |i-j| > 1 \text{ e } i, j \in [0,n] \tag{2.4}$$

$$s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1}, \ i \in [n-1], \tag{2.5}$$

$$s_0 s_1 s_0 s_1 = s_1 s_0 s_1 s_0. \tag{2.6}$$

Neste grupo há dois tipos de relações longas sendo que iremos chamar *relação longa do tipo 1* à relação (2.5) e *relação longa do tipo 2* à relação (2.6). De modo análogo a $\mathfrak{S}_{n+1}$, o grupo $\mathfrak{S}^B_{n+1}$ atua sobre si próprio por multiplicação à direita, o que significa que para $i > 0$, $\mathrm{w}s_i$ transpõe as letras nas posições i e $i+1$ da notação em janela de $\mathrm{w} \in \mathfrak{S}^B_{n+1}$. A reflexão s_0 muda o sinal da primeira letra da notação em janela de w, isto é $\mathrm{w}s_0 = [\overline{\mathrm{w}(1)}, \mathrm{w}(2), \ldots, \mathrm{w}(n+1)]$. Neste contexto, se $\mathrm{w} = s_{i_1} \cdots s_{i_l}$ dizemos que o gerador s_{i_j} atua sobre as letras nas posições i_j e i_j+1 (respetivamente, atua sobre a letra na primeria posição) da notação em janela de $s_{i_1} s_{i_2} \ldots s_{i_{j-1}}$, quando $i_j > 0$ (resp. $i_j = 0$). Para simplificação de escrita, vamos tratar a notação em janela de $s_{i_1} \ldots s_{i_{j-1}}$ como a notação em janela de $i_1 i_2 \ldots i_{j-1}$. Vamos também denotar por $k^{i_1 i_2 \ldots i_{j-1}}$ a k-ésima letra da notação em janela de $i_1 i_2 \ldots i_{j-1}$, ou seja $k^{i_1 i_2 \ldots i_{j-1}} = s_{i_1} \ldots s_{i_{j-1}}(k)$.

Exemplo 2.3. Se $\mathrm{w} = [2, \overline{4}, 3, 1]$, então $\mathrm{w}s_2 = [2, \mathbf{3}, \mathbf{\overline{4}}, 1]$ e $\mathrm{w}s_0 = [\mathbf{\overline{2}}, \overline{4}, 3, 1]$. Temos também que $1^{a\,2} = 2$, $2^{a\,2} = 3$, $3^{a\,2} = \overline{4}$ e $4^{a\,2} = 1$, onde a é uma palavra reduzida de w.

A maior permutação com sinal de $\mathfrak{S}^B_{n+1}$ corresponde à permutação $\mathrm{w}_0^{B_n} = [\overline{1}, \overline{2}, \ldots, \overline{n+1}]$, com comprimento $(n+1)^2$. É fácil de ver que a permutação associada à palavra

$$w_0 = 0 \cdot 10 \cdot 210 \cdot \ldots \cdot (n+1)n \cdots 210 \cdot (n+1)n \cdots 21 \cdot (n+1)n \cdots 2 \ldots \cdot (n+1)n \cdot n$$

é $\mathrm{w}_0^{B_n}$. Como w_0 tem $(n+1)^2$ geradores, esta é uma palavra reduzida de $\mathrm{w}_0^{B_n}$. Por exemplo, no caso $n = 3$ temos $w_0 = 0 \cdot 10 \cdot 210 \cdot 3210 \cdot 321 \cdot 32 \cdot 3$. A palavra w_0 terá um papel muito importante nos capítulos seguintes.

Apesar das palavras reduzidas das permutações de $\mathfrak{S}^B_{n+1}$, e mais especificamente das palavras reduzidas de $\mathrm{w}_0^{B_n}$, serem uma ferramenta importante em várias áreas da combinatória e da álgebra,

como por exemplo no estudo de funções simétricas, partições-P ou em problemas enumerativos que envolvem padrões proibidos [3, 22], sabe-se muito pouco sobre o grafo $C(\mathrm{w}_0^{B_n})$. S. Elnitsky, na sua tese de doutoramento e em [11], provou que o grafo $C(w)$ das classes de comutação de permutações do grupo simétrico é conexo e bipartido, estabelecendo uma bijeção entre palavras reduzidas de $\mathrm{w} \in \mathfrak{S}_{n+1}$ e pavimentações de certos polígonos, resultado este que se estende para as permutações com sinal. Ainda não é conhecida uma expressão para o número de classes de comutação de $\mathrm{w}_0^{B_n}$, conhecendo-se apenas esse número até $n = 8$, (sequência A180605 de [20]).

Na Figura 2.2 está representado o grafo $C(\mathrm{w}_0^{B_2})$ onde as arestas a sólido preto representam relações longas do tipo 1 e as a tracejado relações longas do tipo 2. Antes de avançarmos vamos introduzir algumas noções e notações que irão facilitar a escrita das palavras reduzidas de $\mathrm{w}_0^{B_n}$. Uma *sub-palavra* de a é uma palavra obtida de a através da eliminação de algumas das suas letras, consecutivas ou não, e um *fator* de a é uma sub-palavra formada apenas por letras consecutivas de a. Um *fatorial* é uma palavra da forma $k! := k(k-1)\cdots 10$, com $0! = 0$, e um *fatorial truncado* é uma palavra da forma $(k)_r := k(k-1)\cdots(k-r+1)$ para $1 \leq r \leq k+1$. Note-se que $(k)_{k+1} = k!$. Uma potência k de uma palavra p é a palavra composta pela concatenação de k palavras p, isto é $(p^k) := \underbrace{p\cdots p}_{k \text{ vezes}}$, com p^0 a palavra vazia. Com esta notação podemos escrever $w_0 := \prod_{i=0}^{n} i! \prod_{i=1}^{n} (n)_{n-i+1}$.

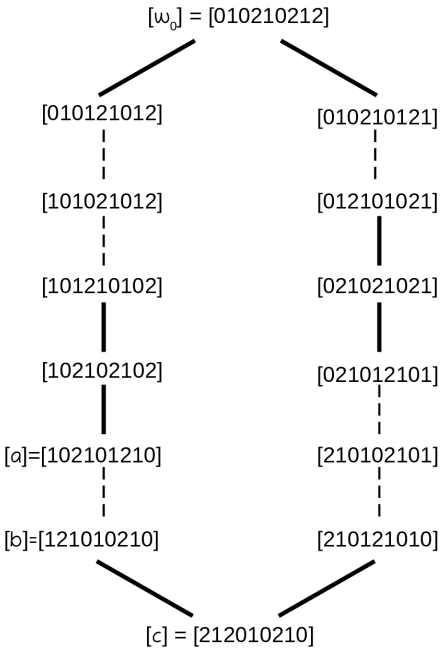

Figura 2.2 O grafo $C(\mathrm{w}_0^{B_2})$.

Capítulo 3

Palavras reduzidas e suas propriedades

Uma palavra $a = i_1 i_2 \cdots i_l$ é uma palavra reduzida de $\mathrm{w} \in \mathfrak{S}_{n+1}^B$ se $\mathrm{w} = s_{i_1} s_{i_2} \cdots s_{i_l}$ e $l = l(\mathrm{w})$. Este comprimento é dado pela fórmula [5]

$$l(\mathrm{w}) = inv(\mathrm{w}) - \sum_{\{j \in [n+1] : \mathrm{w}(j) < 0\}} \mathrm{w}(j), \tag{3.1}$$

com $inv(\mathrm{w}) = |\{(i,j) \in [n] \times [n] : i < j,\ \mathrm{w}(i) > \mathrm{w}(j)\}$. A partir de (3.1) podemos concluir que

$$l(\mathrm{w}s_i) = \begin{cases} l(\mathrm{w}) + 1, & se\ \mathrm{w}(i) < \mathrm{w}(i+1) \\ l(\mathrm{w}) - 1, & se\ \mathrm{w}(i) > \mathrm{w}(i+1) \end{cases}, \tag{3.2}$$

com $i \in [n]$ e

$$l(\mathrm{w}s_0) = \begin{cases} l(\mathrm{w}) + 1, & se\ \mathrm{w}(1) > 0 \\ l(\mathrm{w}) - 1, & se\ \mathrm{w}(1) < 0 \end{cases}. \tag{3.3}$$

Exemplo 3.1. Seja $\mathrm{w} = [2, \overline{1}, 3, \overline{4}]$, cujo comprimento é $l(\mathrm{w}) = |\{(1,2),(1,4),(2,4),(3,4)\}| - (-1 - 4) = 9$. Temos que $\mathrm{w}s_1 = [\overline{1}, 2, 3, \overline{4}]$, $\mathrm{w}s_2 = [2, 3, \overline{1}, \overline{4}]$ e $\mathrm{w}s_0 = [\overline{2}, \overline{1}, 3, \overline{4}]$, logo $l(\mathrm{w}s_1) = |\{(1,4),(2,4),(3,4)\}| - (-1-4) = 8$, $l(\mathrm{w}s_2) = |\{(1,3),(1,4),(2,3),(2,4),(3,4)\}| - (-1-4) = 10$ e $l(\mathrm{w}s_0) = |\{(1,4),(2,4),(3,4)\}| - (-1-2-4) = 10$.

De (3.2) e (3.3) temos o seguinte resultado.

Proposição 3.2. *Seja a uma palavra reduzida de $\mathrm{w} \in \mathfrak{S}_{n+1}^B$ e $i \in [n]$. Então:*

1. *A palavra $a \cdot i$ é uma palavra reduzida de $\mathrm{w}s_i$ se e só se $i^a < (i+1)^a$.*
2. *A palavra $a \cdot 0$ é uma palavra reduzida de $\mathrm{w}s_i$ se e só se $1^a > 0$.*

Demonstração. Iremos demonstrar apenas a primeira equivalência, sendo a outra obtida de forma análoga. Suponhamos que $a \cdot i$ é uma palavra reduzida de $\mathrm{w}s_i$ e suponhamos, por absurdo, que $i^a > (i+1)^a$. De (3.2), temos que $l(\mathrm{w}s_i) = l(\mathrm{w}) - 1$. Como $a \cdot i$ é uma palavra que possui $l(\mathrm{w}) + 1$ letras, $a \cdot i$ não pode ser uma palavra reduzida de $\mathrm{w}s_i$, contrariando assim a hipótese.

Na implicação contrária, se $i^a < (i+1)^a$, então $l(\mathrm{w}s_i) = l(\mathrm{w}) + 1$. Como a é uma palavra reduzida de w, a permutação associada a $a \cdot i$ é $\mathrm{w}s_i$. Temos ainda que $a \cdot i$ é uma palavra com $l(\mathrm{w}) + 1$ letras, logo $a \cdot i$ é uma palavra reduzida de $\mathrm{w}s_i$. □

Podemos encarar uma palavra reduzida de $\mathrm{w}_0^{B_n}$ como um conjunto de operações que transformam a permutação identidade na permutação $\mathrm{w}_0^{B_n}$, mais exatamente um conjunto de transposições de letras em posições adjacentes na notação em janela e mudanças de sinal na letra da primeira posição na notação em janela. Isto pode ser visto facilmente pelo diagrama em linha de uma palavra. O *diagrama em linha* de $a = s_{i_1} s_{i_2} \cdots s_{i_l}$ é o vetor $[\pm 2(n+1)] \times [l]$ em coordenadas cartesianas, que descreve as trajetórias de todas as letras em $[\pm(n+1)]$ à medida que são arranjadas na permutação $\mathrm{w}_0^{B_n}$ pelos geradores s_{i_j}. O diagrama em linha da palavra 101210102 está apresentado na Figura 3.1.

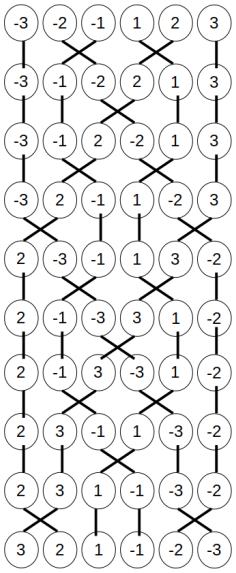

Figura 3.1 Diagrama em linha da palavra 101210102.

Consideremos agora um par $(x,y) \in [\pm(n+1)]^2$, com $x < y$. Como na notação em linha de $\mathrm{w}_0^{B_n}$, a letra y aparece antes da letra x, tem de existir um gerador i_j em a que irá tranpôr as letras x e y, e $\overline{y}$ e $\overline{x}$ na notação em linha de $i_1 i_2 \cdots i_{j-1}$. Neste caso, dizemos que i_j transpôs os pares (x,y) e $(\overline{y},\overline{x})$. Se $i_j = 0$, então este gerador transpõe apenas pares da forma $(\overline{k},k)$, com $k \in [n+1]$ tendo em conta que s_0 atua numa permutação $\mathrm{w} \in \mathfrak{S}_{n+1}^B$ transpondo as letras nas posições $\overline{1}$ e 1 da notação em linha de w. De facto, s_0 é a única reflexão que transpõe pares da forma $(\overline{k},k)$. Existem $(2n+1)(n+1)$ pares (x,y) em $[\pm(n+1)]^2$ com $x < y$, sendo que $n+1$ desses pares são da forma $(\overline{k},k)$, logo toda a palavra reduzida de $\mathrm{w}_0^{B_n}$ possui pelo menos $n+1$ geradores 0. Os restantes $(2n+1)(n+1) - (n+1) = 2n(n+1)$ pares terão de ser transpostos por geradores diferentes de 0. Como cada gerador $i_j \neq 0$ transpõe dois pares em simultâneo, cada palavra reduzida de $\mathrm{w}_0^{B_n}$ necessita de ter pelo menos $\frac{2n(n+1)}{2} = n(n+1)$ geradores diferentes de 0. Se somarmos estes dois minorantes, temos que toda a palavra reduzida de $\mathrm{w}_0^{B_n}$ possui pelo menos $(n+1)^2$ geradores, que é o comprimento de $\mathrm{w}_0^{B_n}$. Logo toda a palavra reduzida de $\mathrm{w}_0^{B_n}$ possui exatamente $n+1$ geradores 0 e exatamente $n(n+1)$ geradores diferentes de 0. Isto implica que cada par (x,y) em $[\pm(n+1)]^2$ com $x < y$ é transposto por um e um só gerador de a.

Dada uma palavra reduzida $a = i_1 i_2 \cdots i_l$ definimos $a^R := i_l i_{l-1} \cdots i_1$ como sendo a palavra *reversa* de a. Dizemos também que a classe gerada por a^R é a *classe reversa* de a. De imediato se conclui que $(a^R)^R = a$ e que a é uma palavra reduzida de $\mathrm{w}_0^{B_n}$ se e só se a^R for reduzida, usando o facto de $\mathrm{w}_0^{B_n}$ ser uma involução e de a e a^R possuirem o mesmo comprimento.

Proposição 3.3. *Seja a uma palavra no alfabeto $[0,n]$. A palavra a é reduzida se e só se a palavra a^R é reduzida.*

Demonstração. Suponhamos que $a = i_1 i_2 \cdots i_l$ é uma palavra reduzida de $\mathrm{w}_0^{B_n}$. Então

$$\begin{aligned} & s_{i_1} s_{i_2} \cdots s_{i_l} = \mathrm{w}_0^{B_n} \\ \Leftrightarrow & (s_{i_1} s_{i_2} \cdots s_{i_l})^{-1} = (\mathrm{w}_0^{B_n})^{-1} \\ \Leftrightarrow & s_{i_l} s_{i_{l-1}} \cdots s_{i_1} = \mathrm{w}_0^{B_n} \end{aligned}$$

Logo a palavra $a^R = i_l i_{l-1} \cdots i_1$ também é uma palavra reduzida. Recipocramente, e de maneira análoga tem-se que se a^R for uma palavra reduzida, então a é uma palavra reduzida. □

No Capítulo 4 iremos mostrar que a excentricidade da classe $[w_0]$ é a distância desta à classe $[w_0^R]$.

Exemplo 3.4. *Seja $a = 010210212$. Então $a^R = 212012010$ é a palavra reversa de a.*

Os geradores $i \in [0,n]$ são involuções, o que implica que nenhuma palavra reduzida poderá ter como fator a palavra $i^2 = ii$. No próximo lema identificamos uma família de fatores que não podem ocorrer numa palavra reduzida.

Proposição 3.5. *Se a é uma palavra que contém o fator $(i \ \cdots \ i+k)^2$ onde $i \in [n]$ e $0 \leq k \leq n-i$ então a não é reduzida.*

Demonstração. Façamos indução sobre k. Para $k = 0$ o resultado é trival visto que os geradores i são involuções. Suponhamos então que o resultado é válido para $k = j$, ou seja que nenhuma palavra reduzida pode conter o fator $(i \ \cdots \ i+j)^2$ para $i \in [n]$, $0 \leq j < n-i$ e mostremos que continua válido para $k = j+1$. Ora,

$$\begin{aligned} (i \ i+1 \cdots i+j+1)^2 &= (i \ i+1 \cdots i+j+1)(i \ i+1 \cdots i+j+1) \\ &\sim (i \ i+1 \cdots i+j)(i \ i+1 \cdots \boldsymbol{i+j+1 \ \ i+j \ \ i+j+1}) \\ &\overset{L_1}{\sim} (i \ i+1 \cdots i+j)(i \ i+1 \cdots \boldsymbol{i+j})(\boldsymbol{i+j+1 \ \ i+j}) \\ &= (i \ i+1 \cdots i+j)^2 (i+j+1 \ \ i+j), \end{aligned}$$

pelo que chegariamos a uma palavra reduzida com o fator $(i \ i+1 \ \cdots \ i+j)^2$, o que não pode acontecer pela hipótese de indução. □

Corolário 3.6. *Se a é uma palavra que contém o fator $(i \ \cdots \ i-k)^2$ onde $i \in [n]$ e $0 \leq k \leq i-1$ então a não é reduzida.*

Demonstração. Se existisse uma palavra reduzida a com tal fator, a^R teria de possuir o fator $(i-k \ i-k+1 \ \cdots \ i)^2$. Pela proposição anterior, a^R não poderia ser reduzida, contradizendo a Proposição 3.3. □

Podemos definir a relação de ordem lexicográfica no conjunto das palavras reduzidas de $\mathrm{w}_0^{B_n}$ do seguinte modo: dadas $a = a_1 ... a_l, b = b_1 ... b_l \in R(\mathrm{w}_0^{B_n})$, dizemos que a é *menor* do que b se existe um inteiro $k \leq l$ tal que $a_i = b_i$ para $1 \leq i \leq k-1$ e $a_k < b_k$.

Exemplo 3.7. Consideremos a classe $[w_0] \in C(\mathrm{w}_0^{B_2})$.
Temos que $w_0 = 010210212$ e que $[w_0] = \{010210212, 012010212, 010212012, 012012012\}$ sendo que a ordenação desta classe em relação à ordem lexicográgica é

$$010210212 < 010212012 < 012010212 < 012012012.$$

Definição 3.8. Dado $a \in R(\mathrm{w}_0^{B_n})$ denotamos por $lex(a)$ a menor palavra da classe de comutação $[a]$ relativa à ordem lexicográfica.

A palavra $lex(a)$ pode-se obter de a colocando todos os geradores $0, 1, \ldots, n-2$ de a o mais à esquerda possível efetuando apenas comutações da forma $i\ j \sim j\ i$ com $j < i$.

Exemplo 3.9. Seja $a = 3102 \cdot 1302 \cdot 1320 \cdot 3120 \in R(\mathrm{w}_0^{B_3})$. Vem que

$$\begin{aligned} 3102 \cdot 1302 \cdot 1320 \cdot 3120 &\sim 3102 \cdot 1\mathbf{03}2 \cdot 1\mathbf{032} \cdot 31\mathbf{02} \\ &\sim \mathbf{13}02 \cdot 1032 \cdot 1032 \cdot \mathbf{13}02 \\ &\sim 1\mathbf{03}2 \cdot 1032 \cdot 1032 \cdot 1\mathbf{03}2. \end{aligned}$$

Como não podemos mover mais para a esquerda os geradores 0 e 1 de a, temos que $lex(a) = 1032 \cdot 1032 \cdot 1032 \cdot 1032$.

Um método de verificar se duas palavras a e b pertencem à mesma classe de comutação é calculando os seus respetivos menores elementos relativamente à ordem lexicográfica.. Visto que esta ordem é uma relação de ordem total, cada classe possui um único elemento mínimo e temos $[a] = [b]$ se e só se $lex(a) = lex(b)$.

Lema 3.10. *Sejam $a, b \in R(\mathrm{w}_0^{B_n})$. Então, $[a] = [b]$ se e só se $lex(a) = lex(b)$.*

A menor palavra na ordem lexicográfica de uma classe teve um papel fundamental no desenvolvimento de um algoritmo que permitiu gerar todas as classes de comutação até $n = 6$, algoritmo esse que será descrito no último capítulo.

Capítulo 4

Função nível e diâmetro

Neste capítulo vamos determinar o diâmetro de $C(\mathrm{w}_0^{B_n})$ bem como recuperar algumas propriedades já conhecidas deste grafo através da construção de uma função que irá caracterizar o grafo por níveis.

Dadas duas palavras $a, b \in R(\mathrm{w}_0^{B_n})$ escrevemos $a \overset{\mathrm{L}_1}{\sim} b$ (respetivamente $a \overset{\mathrm{L}_2}{\sim} b$) se a e b diferem apenas por uma relação longa do tipo 1 (respetivamente tipo 2). Escrevemos também $[a] \overset{\mathrm{L}_1}{\sim} [b]$ (respetivamente $[a] \overset{\mathrm{L}_2}{\sim} [b]$) se existir $a' \in [a]$ e $b' \in [b]$ tais que $a' \overset{\mathrm{L}_1}{\sim} b'$ (respetivamente $a' \overset{\mathrm{L}_2}{\sim} b'$).

Exemplo 4.1. Sejam $a = 102101210$, $b = 121010210$ e $c = 212010210$ palavras reduzidas da permutação $\mathrm{w}_0^{B_2}$, como se pode verificar na Figura 2.2. Como $1\mathbf{02}101210 \sim 1\mathbf{20}101210$ e $12\mathbf{0101}210 \overset{\mathrm{L}_2}{\sim} 12\mathbf{1010}210$, vem que $[a] \overset{\mathrm{L}_2}{\sim} [b]$. Temos também que $\mathbf{121}010210 \overset{\mathrm{L}_1}{\sim} \mathbf{212}010210$, logo $[b] \overset{\mathrm{L}_1}{\sim} [c]$.

Na Figura 4.1 podemos ver um pequeno exemplo de como as classes são formadas e como se ligam umas às outras. A tracejado estão representadas relação curtas entre palavras da mesma classe sendo que a linha sólida preta representa uma relação longa do tipo 1 e a linha vermelha um relação longa do tipo 2. A sublinhado está a menor palavra na ordem lexicográfica de cada uma das classes representadas.

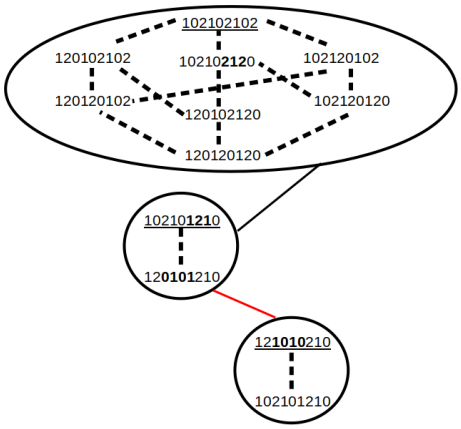

Figura 4.1 Algumas classes de comutação do grafo $C(\mathrm{w}_0^{B_2})$.

Note-se que se duas classes estão ligadas por uma relação longa não implica que todas as palavras de cada uma dessas classes estejam ligadas de alguma maneira. Pela Figura 4.1 notamos que $102102\mathbf{102} \sim 102102\mathbf{120}$, na classe mais acima, e que $10210\mathbf{212}0 \overset{L_1}{\sim} 10210\mathbf{121}0$, mas que

102102102 não está ligada de nenhuma forma à classe do meio. Também verificamos que a relação longa do tipo 2 que está na Figura 4.1 envolve as palavras 120101210 e 121010210 sendo que $12\mathbf{0101}210 \overset{L_2}{\sim} 12\mathbf{1010}210$.

Recordemos que uma palavra reduzida $a = i_1 \cdots i_l$ de $\mathrm{w}_0^{B_n}$ pode ser encarada como um conjunto de operações que transformam a permutação identidade na permutação $\mathrm{w}_0^{B_n}$. Este processo envolve mudar o sinal da letra que está na primeira posição (por ação de s_0) e transpôr letras em posições adjacentes (por ação de s_i com $i \in [n]$) na notação em janela, sendo que estas transposições podem envolver letras positivas ou negativas. Este facto motivou as definições seguintes.

Definição 4.2. Dada uma palavra reduzida $a = i_1 i_2 \cdots i_l$ com $i_1, \ldots, i_l \in [0,n]$, definimos a palavra $\widetilde{a} := \widetilde{i_1}\widetilde{i_2}\cdots\widetilde{i_l}$ onde $\widetilde{0} = 0$ e

$$\widetilde{i_j} = \begin{cases} i_j, & \text{se } (i_j)^{i_1 i_2 \cdots i_{j-1}} > 0 \wedge (i_j+1)^{i_1 i_2 \cdots i_{j-1}} > 0 \\ \overline{i_j}, & \text{caso contrário} \end{cases}$$

para $j \in [l]$ e $i_j > 0$. Definimos ainda a função soma $S(a) := \sum_{j=1}^{l} \widetilde{i_j}$ e a função $neg(a) := |\left\{ i_j | \widetilde{i_j} < 0 \right\}|$ designada por número negativo de a.

Em suma, dada uma palavra reduzida $i_1 i_2 \ldots i_{j-1} i_j i_{j+1} \ldots i_l$ com $i_j \neq 0$, nós temos $\widetilde{i_j} = i_j$ se s_{i_j} transpõe duas letras positivas na notação em janela de $i_1 i_2 \ldots i_{j-1}$. Neste caso dizemos que o gerador i_j (e a reflexão s_{i_j} correspondente) é *positivo*. Caso contrário, se s_{i_j} transpõe pelo menos uma letra negativa na notação em janela de $i_1 i_2 \ldots i_{j-1}$, dizemos que o gerador i_j (e a reflexão s_{i_j} correspondente) é *negativo*. A função S dá-nos a soma dos índices dos geradores positivos subtraido pela soma dos índices dos geradores negativos de uma palavra reduzida, e a função *neg* conta o número de geradores negativos.

Exemplo 4.3. Seja $a = 210102101$. A sucessão dos fatores esquerdos r_i, de comprimentos $1 \leq i \leq 9$, de a, a notação em janelas das respetivas permutações com sinal e as respetivas palavras $\widetilde{r_i}$ são apresentadas em baixo.

$$\begin{aligned}
r_1 = 2,\ [1,\mathbf{3},\mathbf{2}],\ \widetilde{r_1} &= 2 \\
r_2 = 21,\ [\mathbf{3},\mathbf{1},2],\ \widetilde{r_2} &= 21 \\
r_3 = 210,\ [\overline{\mathbf{3}},1,2],\ \widetilde{r_3} &= 210 \\
r_4 = 2101,\ [\mathbf{1},\overline{\mathbf{3}},2],\ \widetilde{r_4} &= 210\overline{1} \\
r_5 = 21010,\ [\overline{\mathbf{1}},\overline{3},2],\ \widetilde{r_5} &= 210\overline{1}0 \\
r_6 = 210102,\ [\overline{1},\mathbf{2},\overline{\mathbf{3}}],\ \widetilde{r_6} &= 210\overline{1}0\overline{2} \\
r_7 = 2101021,\ [\mathbf{2},\overline{\mathbf{1}},\overline{3}],\ \widetilde{r_7} &= 210\overline{1}0\overline{21} \\
r_8 = 21010210,\ [\overline{\mathbf{2}},\overline{1},\overline{3}],\ \widetilde{r_8} &= 210\overline{1}0\overline{2}\overline{1}0 \\
a = r_9 = 210102101,\ [\overline{\mathbf{1}},\overline{\mathbf{2}},\overline{3}],\ \widetilde{a} = \widetilde{r_9} &= 210\overline{1}0\overline{21}0\overline{1}.
\end{aligned}$$

Vem então que $S(a) = 2+1-1-2-1-1 = -2$ e $neg(a) = 4$.

Note-se que podemos escrever qualquer palavra $a \in R(\mathrm{w}_0^{B_n})$ na forma $a = p_1 \cdot p_2 \cdot p_3$ com p_1 e p_3 palavras no alfabeto $[n]$ e p_2 uma palavra no alfabeto $[0,n]$. Deste modo, todos os geradores 0 estão em p_2 fazendo com que os geradores de p_1 sejam todos positivos (pois só atuam em letras positivas) e os geradores de p_3 todos negativos (pois a notação em janela de $p_1 \cdot p_2$ não possui letras positivas), algo que pode ser visto facilmente no exemplo anterior se escrevermos $a = 21 \cdot 010210 \cdot 1$. De facto, podemos relaxar um pouco a restrição em p_3 e dizer que p_3 pode ser uma palavra no alfabeto $[0,n]$ com no máximo um gerador 0.

O resultado que se segue descreve o comportamento das funções S e neg quando aplicadas a palavras pertencentes à mesma classe.

Proposição 4.4. *Se $a \sim b$, então $S(a) = S(b)$ e $neg(a) = neg(b)$.*

Demonstração. Suponhamos sem perda de generalidade que a e b diferem por uma comutação. Deste modo, podemos escrever a e b da forma

$$a = p_1 \cdot i\ j \cdot p_2$$
$$b = p_1 \cdot j\ i \cdot p_2,$$

para certos $i, j \in [0,n]$ tais que $|i-j| > 1$, com p_1 e p_2 palavras no alfabeto $[0,n]$.

É óbvio que os sinais dos geradores de p_1 são os mesmos em a e em b. Como i e j atuam em letras distintas entre si na notação em janela de p_1, os sinais dos geradores i e j serão os mesmos em a e b. Além disso, como $p_1 \cdot i\ j$ e $p_1 \cdot j\ i$ representam a mesma permutação, os sinais dos geradores de p_2 também são os mesmos nas duas palavras, logo $S(a) = S(b)$ e $neg(a) = neg(b)$.

□

Nos próximos corolários, que são consequência da Proposição 3.2, vamos analisar os sinais das letras de palavra reduzidas de $\mathrm{w}_0^{B_n}$ antes de alguns fatores.

Corolário 4.5. *Seja $i \in [n]$, e $a = p_1 \cdot i \cdot p_2$, com p_1, p_2 palavras no alfabeto $[0,n]$, uma palavra reduzida de $\mathrm{w}_0^{B_n}$. Se $i^{p_1} > 0$, então $(i+1)^{p_1} > 0$.*

Demonstração. Como a é uma palavra reduzida, pela Proposição 3.2 têm-se $i^{p_1} < (i+1)^{p_1}$. Como $i^{p_1} > 0$, obrigatoriamente se tem $(i+1)^{p_1} > 0$. □

Corolário 4.6. *Seja $a = p_1 \cdot 1\,0\,1\,0 \cdot p_2$, com p_1, p_2 palavras no alfabeto $[0,n]$, uma palavra reduzida de $\mathrm{w}_0^{B_n}$. Então $1^{p_1} > 0$ e $2^{p_2} > 0$.*

Demonstração. Pela Proposição 3.2 temos que $1^{p_1} = 1^{p_1 \cdot 101} > 0$. Pelo Corolário 4.5, vem que $2^{p_1} > 0$. □

Proposição 4.7. *Sejam $a, b \in R(\mathrm{w}_0^{B_n})$.*

1. *Se $a \overset{L_1}{\sim} b$ então $|S(a) - S(b)| = 1$ e $neg(a) = neg(b)$.*
2. *Se $a \overset{L_2}{\sim} b$ então $|S(a) - S(b)| = 2$, $|neg(a) - neg(b)| = 1$ e $|S(a) + neg(a) - (S(b) + neg(b))| = 1$.*

Demonstração. Suponhamos que $a \overset{\mathrm{L}_1}{\sim} b$ e que podemos escrever a e b da forma

$$a = p_1 \cdot i\,(i+1)\,i \cdot p_2$$
$$b = p_1 \cdot (i+1)\,i\,(i+1) \cdot p_2,$$

para algum $i \in [n-1]$ e p_1, p_2 palavras no alfabeto $[0,n]$.

Como $p_1 \cdot i\,(i+1)\,i$ e $p_1 \cdot (i+1)\,i\,(i+1)$ representam a mesma permutação, os sinais dos geradores de p_2 são os mesmos em ambas as palavras, pelo que as funções S e neg irão depender exclusivamente do sinal dos geradores dos fatores $i\,(i+1)\,i$ e $(i+1)\,i\,(i+1)$ de a e b, respetivamente. Os geradores desses fatores atuam nas letras i^{p_1}, $(i+1)^{p_1}$ e $(i+2)^{p_1}$ da notação em janela de p_1, pelo que precisamos de analisar as diferentes possibilidades para os seus sinais. Se $i^{p_1} > 0$, então pelo Corolário 4.5 temos $(i+1)^{p_1} > 0$ e $(i+2)^{p_1} > 0$, logo

$$S(a) - S(b) = i + (i+1) + i - (i+1+i+i+1) = -1. \tag{4.1}$$

Suponhamos agora que $i^{p_1} < 0$. Se apenas essa letra for negativa, então exatamente dois geradores dos fatores $i(i+1)i$ e $(i+1)i(i+1)$ são negativos. Neste caso nós temos

$$S(a) - S(b) = -i - (i+1) + i - (i+1-i-(i+1)) = -1. \tag{4.2}$$

Finalmente, se duas ou mais letras forem negativas, então todos os geradores dos fatores $i(i+1)i$ e $(i+1)i(i+1)$ de a e b, respetivamente, são negativos e temos

$$S(a) - S(b) = -i - (i+1) - i - (-(i+1) - i - (i+1)) = 1. \tag{4.3}$$

Das equações (4.1), (4.2) e (4.3) vem que $|S(a) - S(b)| = 1$ e $neg(a) = neg(b)$.

Suponhamos agora que $a \overset{\mathrm{L}_2}{\sim} b$ e que podemos escrever a e b da forma

$$a = p_1 \cdot 1\,0\,1\,0 \cdot p_2$$
$$b = p_1 \cdot 0\,1\,0\,1 \cdot p_2,$$

com p_1, p_2 palavras no alfabeto $[0,n]$. De modo análogo ao caso anterior, apenas precisamos de analisar os sinais dos fatores $1\,0\,1\,0$ e $0\,1\,0\,1$ de a e b, respetivamente.

Do Corolário 4.6, o gerador 1 mais à esquerda de $(1\,0\,1\,0)$ é positivo. Em b, os dois geradores 1 de $(0\,1\,0\,1)$ são negativos, logo

$$S(a) - S(b) = 1 - 1 - (-1 - 1) = 2$$

e $neg(a) - neg(b) = -1$.

Segue também que $|S(a) + neg(a) - (S(b) + neg(b))| = 1$. □

As funções S e neg vão servir de base para uma caracterização por níveis do grafo $C(\mathrm{w}_0^{B_n})$, que será a chave para determinar o diâmetro deste grafo.

Definição 4.8. Dado $a \in R(\mathrm{w}_0^{B_n})$ definimos o *nível* de a por $N(a) := S(a) + neg(a)$.

Na próxima proposição vamos provar que a função $N\colon R(\mathrm{w}_0^{B_n}) \to \mathbb{N}$ é invariante dentro de cada classe, e que varia uma unidade entre palavras de classes que diferem por uma relação longa. Esta função generaliza a função usada em [14] que permitiu calcular o diâmetro e o raio do grafo das classes de comutação para a maior permutação do grupo simétrico.

Proposição 4.9. *Sejam $a, b \in R(\mathrm{w}_0^{B_n})$.*

1. *Se $a \sim b$, então $N(a) = N(b)$.*
2. *Se $a \overset{L_1}{\sim} b$ ou $a \overset{L_2}{\sim} b$, então $|N(a) - N(b)| = 1$.*

Demonstração. A condição (1) é consequência da invariância de S e *neg* dentro de cada classe, provada na Proposição 4.4. A condição (2) é uma consequência imediata da Proposição 4.7. □

Como a função N é invariante dentro de cada classe de comutação, vamos considerar o nível de uma classe $[a]$ como sendo o nível de a. Da proposição anterior podemos concluir que o grafo $C(\mathrm{w}_0^{B_n})$ se pode particionar em níveis, onde cada nível é definido pelos valores da função N sendo que duas classes de comutação estão ligadas por uma aresta se os seus níveis diferem por uma unidade. Na Figura 7.6 está representado o grafo $C(\mathrm{w}_0^{B_3})$ colorido de acordo com a função nível. Atendendo ao facto de que há sempre um número finito de classes comutação, para cada $n \geq 1$ a função nível terá sempre um valor mínimo e máximo. Se existir apenas uma classe $[a]$ com nível mínimo e uma classe $[b]$ com nível máximo, intuitivamente a excentricidade de $[a]$ será igual à distância entre esta e a classe $[b]$. De facto, como mostraremos no final deste capítulo, essa intuição é verdadeira.

Na tabela que se segue estão representados todos os casos do comportamento da função nível. Consideremos então duas palavras reduzidas a e b que diferem por uma relação longa (do tipo 1 ou 2). Então, podemos escrever estas palavras como $a = p_1 \cdot q_1 \cdot p_2$ e $b = p_1 \cdot q_2 \cdot p_2$, com p_1, p_2 palavras no alfabeto $[0, n]$ e onde q_1 e q_2 são dadas pelas expressões nas colunas 2 e 3 da Tabela 4.1 A coluna "Sinais" indica os sinais das letras i^{p_1}, $(i+1)^{p_1}$ e $(i+2)^{p_1}$, no caso de uma relação longa do tipo 1, e das letras 1^{p_1} e 2^{p_1} no caso de uma relação longa do tipo 2.

Relação longa	q_1	q_2	Sinais	$N(a) - N(b)$
1	$i\ (i+1)\ i$	$(i+1)\ i\ (i+1)$	+ + +, - + +	< 0
1	$i\ (i+1)\ i$	$(i+1)\ i\ (i+1)$	- - + ou - - -	> 0
2	(0 1 0 1)	(1 0 1 0)	+ +	< 0

Tabela 4.1 Comportamento da função nível de acordo com os sinais das letras nas posições onde os geradores de uma relação longa atuam.

Os valores da função nível nas palavras w_0 e w_0^R vão ser importantes para o cálculo do diâmetro, isto porque vamos mostrar mais à frente que w_0 e w_0^R são as palavras com menor e maior nível, respetivamente. Consideremos em primeiro lugar $w_0 = \prod_{i=0}^{n} i! \prod_{i=1}^{n} (n)_{n-i+1}$. Nesta palavra, todo o gerador diferente de 0 tranpõe pelo menos uma letra negativa na notação em janela, logo todos os geradores de w_0 diferentes de 0 são negativos. Como há $n(n+1)$ geradores diferentes de 0 em qualquer palavra reduzida de $\mathrm{w}_0^{B_n}$, têm-se

$$\begin{aligned} N(w_0) = S(p) + neg(p) &= -\frac{n(n+1)(n+2)}{6} - \frac{n(n+1)(2n+1)}{6} + (n+1)n \\ &= \frac{n(n+1)(1-n)}{2}. \end{aligned}$$

Quanto à palavra w_0^R, podemos escrevê-la na forma $w_0^R = (\prod_{i=1}^{n}(n)_{n-i+1})^R \cdot (\prod_{i=0}^{n} i!)^R$. Como todos os geradores 0 de w_0^R estão no fator $(\prod_{i=0}^{n} i!)^R$, todos os geradores do fator $(\prod_{i=1}^{n}(n)_{n-i+1})^R$ são positivos. Atendendo a que $(\prod_{i=0}^{n} i!)^R = 012\cdots n \cdot 012\cdots n-1 \cdot \ldots \cdot 012 \cdot 01 \cdot 0$, todos os $\frac{n(n+1)}{2}$ geradores diferentes de 0 deste fator são negativos. Temos então que

$$\begin{aligned} N(w_0^R) &= \frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)(n+2)}{6} + \frac{(n+1)n}{2} \\ &= \frac{n(n+1)(n+2)}{6}. \end{aligned}$$

De seguida vamos mostrar que a classe $[w_0]$ é a classe que possui menor nível. Para tal, vamos decompor a prova em vários lemas e introduzir uma nova definição.

Definição 4.10. Seja $a = p_1 \cdot i! \cdot p_2$ uma palavra reduzida de $\mathrm{w}_0^{B_n}$. Dizemos que o fatorial $i!$ é um *fatorial negativo* se todos os geradores diferentes 0 que o formam são negativos. Convencionamos que 0! é um fatorial negativo.

Exemplo 4.11. Seja p uma palavra com $[\overline{2},\overline{1},3]$ a sua permutação associada e consideremos a palavra $a = p \cdot 2! = p \cdot 210$. Vem que

$$\begin{aligned} p \cdot 2,\ [\overline{2},3,\overline{1}],\ \widetilde{p} \cdot \overline{2} \\ p \cdot 21,\ [3,\overline{2},\overline{1}],\ \widetilde{p} \cdot \overline{21} \\ p \cdot 210,\ [\overline{3},\overline{2},\overline{1}],\ \widetilde{w} = \widetilde{p} \cdot \overline{210}. \end{aligned}$$

Temos então que 2! é um fatorial negativo.

Lema 4.12. *Seja $a = p_1 \cdot i! \cdot p_2$ uma palavra reduzida com $i \in [n]$, e p_1 e p_2 palavras no alfabeto $[0,n]$. Então, o fator $i!$ é um fatorial negativo se e só se $k^{p_1} < 0$ para todo o $k \in [i]$.*

Demonstração. O fator $i!$ atua em p_1 "empurrando" a letra $(i+1)^{p_1}$ para a primeira posição da notação em janela, isto é $(i+1)^{p_1} = 1^{p_1 \cdot i(i-1)\cdots 21}$, mudando de seguida o seu sinal, que pela Proposição 3.2 é positivo. Como o fatorial $i!$ é negativo, todos os seus geradores são negativos e transpõem a letra $(i+1)^{p_1}$, logo $k^{p_1} < 0$ para todo $k \in [i]$. Reciprocamente, se $k^{p_1} < 0$ para todo $k \in [i]$, todo o gerador diferente de 0 de $i!$ vai atuar em letras negativas, pelo que o fatorial $i!$ será negativo. □

Exemplo 4.13. Seja p uma palavra tal que $[\overline{1},\overline{2},3,\overline{4},5]$ é a sua permutação associada e $a = p\cdot 4!$. Então,

$$\begin{aligned}
p,\ &[\overline{1},\overline{2},\mathbf{3},\overline{4},\mathbf{5}],\ \widetilde{p}\\
p\cdot 4,\ &[\overline{1},\overline{2},\mathbf{3},\mathbf{5},\overline{4}],\ \widetilde{p}\cdot\overline{4}\\
p\cdot 43,\ &[\overline{1},\overline{2},\mathbf{5},\mathbf{3},\overline{4}],\ \widetilde{p}\cdot\overline{4}3\\
p\cdot 432,\ &[\overline{1},\mathbf{5},\overline{2},\mathbf{3},\overline{4}],\ \widetilde{p}\cdot\overline{4}3\overline{2}\\
p\cdot 4321,\ &[\mathbf{5},\overline{1},\overline{2},\mathbf{3},\overline{4}],\ \widetilde{p}\cdot\overline{4}3\overline{21}\\
p\cdot 43210,\ &[\overline{\mathbf{5}},\overline{1},\overline{2},\mathbf{3},\overline{4}],\ \widetilde{w}=\widetilde{p}\cdot\overline{4}3\overline{21}0
\end{aligned}$$

Note-se que como $3^p > 0$, o gerador 3 de 4! ficou positivo, fazendo com que esse fatorial não fosse negativo.

Lema 4.14. *Seja $a = p_1\cdot\boldsymbol{j}\cdot i!\cdot p_2$ com $i\in[n]$, $\boldsymbol{j}\in[0,i-1]$, e p_1 e p_2 palavras no alfabeto $[0,n]$, tal que a é a menor palavra na ordem lexicográfica de $[a]$, e $N(a)$ o valor mínimo da função nível. Se $i!$ é um fatorial negativo, então $\boldsymbol{j}=0$.*

Demonstração. Para $i=1$ a afirmação do lema é trivial uma vez que a é a menor palavra de $[a]$ para a ordem lexicográfica. Se $i>1$, então $0\le j<i$ o que implica a que

$$a\sim a' = p_1\cdot i(i-1)\cdots\boldsymbol{j}(j+1)j\cdot(j-1)!\cdot p_2.$$

Note-se que $i!$ sendo um fatorial negativo força a que o gerador $\boldsymbol{j}$ seja também negativo, logo $a'\overset{\mathrm{L_1}}{\sim} b = p_1\cdot i(i-1)\cdots(j+1)\ j\ (j+1)\cdot(j-1)!\cdot p_2$, que pela Tabela 4.1 satisfaz $N(b)<N(a)$, contradizendo a minimalidade de $N(a)$. Logo temos ter de $\boldsymbol{j}=0$. □

Lema 4.15. *Seja $a = p_1\cdot j!\cdot i!\cdot p_2$ com $i\in[n]$, $j\in[0,n]$, $i>0$ e p_1,p_2 palavras no alfabeto $[0,n]$, tal que a é a menor palavra na ordem léxicográfica de $[a]$, e $N(a)$ o valor mínimo da função nível. Se $i!$ é um fatorial negativo, então $j!$ também é um fatorial negativo e $j<i$.*

Demonstração. Se $i=1$ e $j\ge 1$, então

$$a = p_1\cdot j(j-1)\cdots 10\cdot 10\cdot p_2\overset{\mathrm{L_2}}{\sim} p_1\cdot j(j-1)\cdots 01\cdot 01\cdot p_2 = b,$$

o que pela Tabela 4.1 implica que $N(b)<N(a)$, contradizendo a minimalidade de $N(a)$. Suponhamos agora que $j\ge i>1$. Então, podemos escrever a como

$$a = p_1\cdot j(j-1)\cdots(i+1)i!i!\cdot p_2.$$

Os dois fatoriais $i!i!$ atuam nas letras $i^{p_1\cdot j(j-1)\cdots(i+1)}$ e $(i+1)^{p_1\cdot j(j-1)\cdots(i+1)}$ mudando os seus sinais, pelo que o gerador i do fatorial $i!$ mais à esquerda de a é positivo. Temos então

$$\begin{aligned}
a\sim\ & p_1\cdot j(j-1)\cdots(i+1)\mathbf{i}(\mathbf{i}-\mathbf{1})\mathbf{i}\cdot(i-2)!(i-1)!\cdot p_2 = a'\\
\overset{\mathrm{L_1}}{\sim}\ & p_1\cdot j(j-1)\cdots(i+1)(\mathbf{i}-\mathbf{1})\mathbf{i}(\mathbf{i}-\mathbf{1})\cdot(i-2)!(i-1)!\cdot p_2 = b.
\end{aligned}$$

Já sabemos que o gerador i mais à esquerda do fator $i(i-1)i$ de a' é positivo, e o gerador i mais à direita desse mesmo fator é negativo pelo facto de pertencer a um fatorial negativo, o que implica pela Tabela 4.1 a que $N(b) < N(a)$, contradizendo a minimalidade de $N(a)$. Logo temos de ter $j < i$. Finalmente, note-se que sendo $i!$ um fatorial negativo, pelo Lema 4.12 temos $k^{p_1 \cdot j!} < 0$ para todo $k \in [i]$ fazendo com que a condição $j < i$ implique que $k^{p_1 \cdot j!} < 0$ para todo $k \in [j+1]$, provando assim que $j!$ é um fatorial negativo. □

Proposição 4.16. *Seja $a \in R(\mathrm{w}_0^{B_n})$. Se $[a] \neq [w_0]$ então existe $b \in R(\mathrm{w}_0^{B_n})$ tal que $[a] \overset{L_1}{\sim} [b]$ ou $[a] \overset{L_2}{\sim} [b]$ e $N(a) > N(b)$.*

Demonstração. Provemos a afirmação contra-recíproca, ou seja, vamos supôr que $[a]$ é uma classe tal que para toda a classe $[b]$ com $[a] \overset{L_1}{\sim} [b]$ ou $[a] \overset{L_2}{\sim} [b]$ se tem $N(a) < N(b)$ e iremos provar que nessas condições $[a] = [w_0]$. Iremos começar por provar que $\prod_{i=0}^{n} i!$ é um fator esquerdo da menor palavra da classe $[a]$, com respeito à ordem lexicográfica, o que, sem perda de generalidade, vamos assumir que é a. É fácil de ver que todo o gerador 0 de a pertence a um fatorial, pelo que a possui exatamente $n+1$ fatoriais. Temos também que o fatorial mais à direira de a tem de ser negativo visto que atua numa permutação com apenas uma letra positiva na sua notação em janela. Através de sucessivas aplicações do Lema 4.15 podemos escrever $a = i_0!i_1!\cdots i_n! \cdot p$, onde $i_j!$ é um fatorial negativo com $i_j < i_{j+1}$ e p uma palavra no alfabeto $[n]$. Temos então que

$$a = 0!1!\cdots n! \cdot p = \prod_{i=0}^{n} i! \cdot p.$$

É fácil de verificar que a permutação associada ao fator esquerdo $\prod_{i=0}^{n} i!$ de a é $[\overline{n+1}, \overline{n}, \ldots, \overline{1}]$, isto é, a restrição de p ao conjunto $[n]$ corresponde à maior permutação do grupo simétrico $\mathfrak{S}_{n+1}$. Temos ainda que, como p tem $\frac{(n+1)n}{2}$ letras, é de facto uma palavra reduzida de $\mathrm{w}_0^{A_n}$. Todos os geradores de p são negativos pelo que p não pode conter nenhum fator da forma $i(i+1)i$ de acordo com Tabela 4.1, para $i \in [n-1]$. Foi provado em [14] que de todas as palavras reduzidas de $\mathrm{w}_0^{A_n}$, as únicas que possuem o maior valor para a função soma são as que pertencem à classe $[\prod_{i=1}^{n} (n)_{n-i+1}]$, que são exatamente as palavras que não possuem nenhum fator da forma $i(i+1)i$. Segue então que $p = \prod_{i=1}^{n} (n)_{n-i+1}$ e $a = w_0$. □

Da Proposição 4.16 concluimos que toda a classe de comutação diferente de $[w_0]$ está ligada a outra classe com menor valor da função nível, o que significa que toda a classe de comutação está ligada a $[w_0]$, o que nos dá uma nova prova de que o grafo $C(\mathrm{w}_0^{B_n})$ é conexo [25]. Da Proposição 4.9 segue que $C(\mathrm{w}_0^{B_n})$ é um grafo bipartido [11], com a partição das classes comutação dada pela paridade dos valores da função nível nos seus vértices.

Lema 4.17. *Seja $a, b \in R(\mathrm{w}_0^{B_n})$ tal que $a \overset{L_1}{\sim} b$ ($a \overset{L_2}{\sim} b$, respetivamente) e $N(a) < N(b)$. Então $a^R \overset{L_1}{\sim} b^R$ ($a^R \overset{L_2}{\sim} b^R$, respetivamente) e $N(a^R) > N(b^R)$.*

Demonstração. É claro que se $a \overset{L_1}{\sim} b$ ($a \overset{L_2}{\sim} b$, respetivamente), então $a^R \overset{L_1}{\sim} b^R$ ($a^R \overset{L_2}{\sim} b^R$, respetivamente). Suponhamos que $a \overset{L_1}{\sim} b$. Então, podemos escrever a e b da forma

$$a = p_1 \cdot i\,(i+1)\,i \cdot p_2$$
$$b = p_1 \cdot (i+1)\,i\,(i+1) \cdot p_2,$$

para algum $i \in [n-1]$ e p_1, p_2 palavras no alfabeto $[0,n]$. Como estamos a assumir que $N(a) < N(b)$, pela Tabela 4.1 temos que $(i+1)^{p_1} > 0$ e $(i+2)^{p_1} > 0$. Isto implica que serão geradores do fator p_2 a mudar o sinal das letras $(i+1)^{p_1}$ e $(i+2)^{p_1}$, o que significa que $(i+1)^{p_2^R} < 0$ e $(i+2)^{p_2^R} < 0$, logo pela Tabela 4.1 temos $N(a^R) > N(b^R)$. No caso de $a \overset{L_2}{\sim} b$ é rápido verificar que se $N(a) < N(b)$, então $N(a^R) > N(b^R)$ através da Tabela 4.1.

□

Pela Proposição 4.16 e pelo lema anterior, podemos concluir que se $[a] \neq [w_0^R]$, então existe $b \in R(\mathrm{w}_0^{B_n})$ tal que $[a] \overset{L_1}{\sim} [b]$ ou $[a] \overset{L_2}{\sim} [b]$ e $N(a) < N(b)$, logo $N(w_0^R)$ é o valor máximo da função nível, e que $N(a) = N(w_0^R)$ se e só se $a \in [w_0^R]$.

Proposição 4.18. *Seja $[a]$ uma classe de comutação. Então $d([w_0],[a]) = N(a) - N(w_0)$ e $d([a],[w_0^R]) = N(w_0^R) - N(a)$. Em particular $d([w_0],[w_0^R]) = \dfrac{n(n+1)(4n-1)}{6}$.*

Demonstração. Pelas Proposições 4.9 e 4.16, existe $a \in R(\mathrm{w}_0^{B_n})$ tal que $[a] \overset{L_1}{\sim} [w_0^R]$ com $N(a) = N(w_0^R) - 1$. Repetindo este processo $N(w_0^R) - N(w_0)$ vezes chegamos à classe $[w_0]$ provando que

$$d([w_0],[w_0^R]) = N(w_0^R) - N(w_0) = \frac{n(n+1)(4n-1)}{6}.$$

Este processo pode ser aplicado também para qualquer classe pelo que $d([w_0],[a]) = N(a) - N(w_0)$ e $d([a],[w_0^R]) = N(w_0^R) - N(a)$. □

Teorema 4.19. *O diâmetro de $C(\mathrm{w}_0^{B_n})$ é $N(w_0^R) - N(w_0) = \dfrac{n(n+1)(4n-1)}{6}$.*

Demonstração. Para provar que o diâmetro do grafo $C(\mathrm{w}_0^{B_n})$ é $\dfrac{n(n+1)(4n-1)}{6}$, precisamos de mostrar que este número é o máximo das distâncias entre duas quaisquer classes no grafo. Consideremos duas classes $[a]$ e $[b]$. Como

$$d([w_0],[a]) + d([a],[w_0^R]) + d([w_0],[b]) + d([b],[w_0]) = 2(N(w_0^R) - N(w_0))$$

pelo Lema 4.18, usando a desigualdade triangular, concluimos que

$$d([a],[b]) \leq min\{d([w_0],[a]) + d([w_0],[b]), d([a],[w_0^R]) + d([b],[w_0^R])\} \leq N(w_0^R) - N(w_0).$$

Deste modo, provámos que a distância entre duas classe $[a]$ e $[b]$ é no máximo $N(w_0^R) - N(w_0)$. Logo o máximo das excentricidades de uma classe de comutação no grafo $G(w_0)$ é $\dfrac{n(n+1)(4n-1)}{6}$. □

Como consequência do teorema anterior, a excentricidade da classe $[w_0]$ é a sua distância à classe $[w_0^R]$.

Capítulo 5

Raio

Neste capítulo iremos determinar o raio de $C(\mathrm{w}_0^{B_n})$, fazendo corresponder uma palavra reduzida de $\mathrm{w}_0^{B_n}$ com uma palavra reduzida de $\mathrm{w}_0^{A_{2n+1}}$. Esta correspondência vai permitir usar alguns resultados de [14] sobre o grafo $C(\mathrm{w}_0^{A_n})$.

Consideremos o grupo simétrico $\mathfrak{S}_{2(n+1)}$ no alfabeto $[\pm(n+1)]$ gerado pelas reflexões $\{s_i^A : i \in [\pm n] \cup \{0\}\}$ onde $s_0^A = (-1\ 1)$ e, para $i \in [\pm n]$, $s_i^A = (i\ i+1)$ se $i > 0$ e $s_i^A = (i-1\ i)$ para $i < 0$. O grupo $\mathfrak{S}_{n+1}^B$ é um sub-grupo de $\mathfrak{S}_{2(n+1)}$ pelo que podemos considerar $\mathrm{w} \in \mathfrak{S}_{n+1}^B$ como uma permutação em $\mathfrak{S}_{2(n+1)}$ através da injeção

$$\begin{aligned} \phi \colon \mathfrak{S}_{n+1}^B &\to \mathfrak{S}_{2(n+1)} \\ \mathrm{w} &\mapsto \phi(\mathrm{w}) = \left(\overline{\mathrm{w}(n+1)}, \ldots, \overline{\mathrm{w}(1)}, \mathrm{w}(1), \ldots, \mathrm{w}(n+1)\right). \end{aligned}$$

Note-se que $\phi(\mathrm{w}_0^{B_n}) = \mathrm{w}_0^{A_{2n+1}}$, *i.e.* a maior permutação com sinal de $\mathfrak{S}_{n+1}^B$ corresponde à maior permutação de $\mathfrak{S}_{2(n+1)}$. Temos também que $\phi(s_i) = s_i^A s_{\bar{i}}^A$, para $i \in [n]$, e $\phi(s_0) = s_0^A$. Se $a = i_1 i_2 \cdots i_l$ é uma palavra reduzida de $\mathfrak{S}_{n+1}^B$, escrevemos $\phi(a)$ para denotar a palavra correspondente em $\mathfrak{S}_{2(n+1)}$, ou seja, $\phi(a) = \widetilde{i_1}\widetilde{i_2}\ldots\widetilde{i_l}$, com $\widetilde{i_j} = i_j\overline{i_j}$ se $i_j \neq 0$ e $\widetilde{i_j} = 0$ caso contrário.

Lema 5.1. *Se $a = i_1 i_2 \cdots i_l$ é uma palavra reduzida de $\mathrm{w}_0^{B_n}$, então $\phi(a)$ é uma palavra reduzida de $\mathrm{w}_0^{A_{2n+1}}$.*

Demonstração. Como todo o gerador diferente de 0 de $\mathfrak{S}_{n+1}^B$ corresponde a dois geradores em $\mathfrak{S}_{2(n+1)}$, e a possui exatamente $n+1$ geradores 0, temos que o comprimento de $\phi(a)$ é $2(n+1)n + (n+1) = (n+1)(2n+1)$. Concluimos assim que $\phi(a)$ é uma palavra reduzida de $\mathrm{w}_0^{A_{2n+1}}$, uma vez que a permutação associada a $\phi(a)$ é $\mathrm{w}_0^{A_{2n+1}}$.

□

Dadas duas palavras $a, b \in R(\mathrm{w}_0^{A_{2n+1}})$, escrevemos $a \overset{\mathrm{L}_A}{\sim} b$ se a e b diferem por uma relação longa. Relembramos que no caso do grupo simértico, só há um tipo de relação longa.

Lema 5.2. *Seja $a, b \in R(\mathrm{w}_0^{B_n})$.*

1. *Se $a \sim b$, então $\phi(a) \sim \phi(b)$.*
2. *Se $a \overset{L_1}{\sim} b$, então a palavra $\phi(a)$ difere de $\phi(b)$ por duas relações longas.*

3. *Se $a \overset{L_2}{\sim} b$, então $\phi(a)$ difere de $\phi(b)$ por quatro relações longas.*

Demonstração. Vamos provar apenas (1) e (3), sendo a prova de (2) análoga à prova de (3). Suponhamos sem perda de generalidade que a e b diferem por uma única comutação. Então, podemos escrever $a = p_1 \cdot i\ j \cdot p_2$ e $b = p_1 \cdot j\ i \cdot p_2$, para alguns $i, j \in [0,n]$ tais que $|i-j| > 1$, com p_1 e p_2 palavras no alfabeto $[0,n]$. Se $i \neq 0 \neq j$, temos

$$\begin{aligned}\phi(a) &= \phi(p_1) \cdot i\ \bar{i}\ j\ \bar{j} \cdot \phi(p_2) \\ &\sim \phi(p_1) \cdot j\ i\ \bar{i}\ \bar{j} \cdot \phi(p_2) \\ &\sim \phi(p_1) \cdot j\ \bar{j}\ i\ \bar{i} \cdot \phi(p_2) = \phi(b).\end{aligned}$$

Para o caso de $i = 0$ ou $j = 0$, o procedimento é análogo.

Suponhamos agora que $a \overset{L_2}{\sim} b$. Então, podemos escrever $a = p_1 \cdot 0\ 1\ 0\ 1 \cdot p_2$ e $b = p_1 \cdot 1\ 0\ 1\ 0 \cdot p_2$, com p_1, p_2 palavras no alfabeto $[0,n]$. Deste modo,

$$\begin{aligned}\phi(a) &= \phi(p_1) \cdot 0\ 1\ \overline{1}\ 0\ 1\ \overline{1} \cdot \phi(p_2) \\ &\sim \phi(p_1) \cdot 0\ 1\ \overline{1}\ 0\ \overline{1}\ 1 \cdot \phi(p_2) \\ &\overset{L_A}{\sim} \phi(p_1) \cdot 0\ 1\ 0\ \overline{1}\ 0\ 1 \cdot \phi(p_2) \\ &\overset{L_A}{\sim} \phi(p_1) \cdot 1\ 0\ 1\ \overline{1}\ 0\ 1 \cdot \phi(p_2) \\ &\sim \phi(p_1) \cdot 1\ 0\ \overline{1}\ 1\ 0\ 1 \cdot \phi(p_2) \\ &\overset{L_A}{\sim} \phi(p_1) \cdot 1\ 0\ \overline{1}\ 0\ 1\ 0 \cdot \phi(p_2) \\ &\overset{L_A}{\sim} \phi(p_1) \cdot 1\ \overline{1}\ 0\ \overline{1}\ 1\ 0 \cdot \phi(p_2) = \phi(b).\end{aligned}$$

□

Seja T_n o conjunto formado por todos os ternos $(x,y,z) \in [\pm(n+1)]^3$ tais que $x < y < z$. Recordemos que, dada uma palavra reduzida $a \in R(\mathrm{w}_0^{B_n})$, todos os pares $(x,y) \in [\pm(n+1)]^2$ com $x < y$ são transpostos uma e uma só vez durante o processo de transformação da permutação identidade na permutação $\mathrm{w}_0^{B_n}$.

Definição 5.3. Dada uma palavra $a \in R(\mathrm{w}_0^{B_n})$ e um terno $(x,y,z) \in T_n$ definimos $T(a,xyz) = 1$ se, pela ação dos geradores de a na permutação identidade, a transposição do par (x,y) ocorre antes da transposição do par (y,z), e definimos $T(a,xyz) = -1$ caso contrário.

O valor $T(a,xyz)$ pode ser obtido facilmente através do diagram em linha de a. Por exemplo, se considerarmos $a = 101210102$, cujo diagrama em linha está representado na Figura 3.1, temos $T(a,\overline{2}\ \overline{1}\ 1) = 1$ e $T(a,\overline{3}\ 1\ 2) = -1$. Foi provado em [14] que no grafo $C(\mathrm{w}_0^{A_{2n+1}})$, a função T é invariante em palavras pertencentes à mesma classe e que $[a] \overset{L_A}{\sim} [b]$ se e só se $T(a,xyz) = T(b,xyz)$ para todo o terno $(x,y,z) \in T_n$, excepto num único terno [15]. Do Lema 5.2 obtemos as seguintes propriedades.

Lema 5.4. *Sejam a,b duas palavras reduzidas de $\mathrm{w}_0^{B_n}$.*

- *$[a] = [b]$ se e só se $T(a,xyz) = T(b,xyz)$ para todo o terno $(x,y,z) \in T_n$.*

- *Se $a \overset{L_1}{\sim} b$, então $T(a,xyz) = T(b,xyz)$ para todo o terno $(x,y,z) \in T_n$, excepto em dois ternos.*
- *Se $a \overset{L_2}{\sim} b$, então $T(a,xyz) = T(b,xyz)$ para todo o terno $(x,y,z) \in T_n$, excepto em quatro ternos.*

Lema 5.5. *Seja a uma palavra reduzida de $\mathrm{w}_0^{B_n}$. Para todo o terno $(x,y,z) \in T_n$ têm-se $T(a,xyz) = -T(a,\overline{zyx})$.*

Demonstração. Seja $(x,y,z) \in T_n$ e suponhamos sem perda de generalidade que $T(a,xyz) = 1$. Note-se que $(\overline{z},\overline{y},\overline{x}) \in T_n$ porque $x < y < z$. Como $T(a,xyz) = 1$, o par (x,y) foi transposto antes do par (y,z). Os geradores que transposeram esses pares também foram responsáveis por inverter os pares $(\overline{x},\overline{y})$ e $(\overline{y},\overline{z})$, respetivamente, logo o par $(\overline{x},\overline{y})$ foi transposto antes do par $(\overline{y},\overline{z})$, o que implica a que $T(a,\overline{zyx}) = -1$. □

Proposição 5.6. *Dada uma palavra $a \in R(\mathrm{w}_0^{B_n})$ e um terno $(x,y,z) \in T_n$ temos que $T(a,xyz) = -T(a^R,xyz)$.*

Demonstração. O diagrama em linha de a^R corresponde a uma reflexão horizontal do simétrico de todas as componentes do diagrama em linha de a. Dado um terno (x,y,z), se o par (x,y) foi transposto antes do par (y,z) por geradores de a, então o par (y,z) foi transposto antes do par (x,y) por geradores de a^R, o que implica que $T(a,xyz) = -T(a^R,xyz)$. O mesmo aconteceria se o par (y,z) fosse transposto antes do par (x,y) por geradores de a. □

Dada uma palavra reduzida $a \in R(\mathrm{w}_0^{B_n})$ temos que, pela Proposição 5.6, o comprimento do caminho entre $[\phi(a)]$ e $[\phi(a^R)]$ é no máximo $|T_n| = \binom{2n+2}{3} = \frac{(2n+2)(2n+1)2n}{6}$. Foi provado em [14] que este comprimento é de facto a distância entre $[\phi(a)]$ e $[\phi(a^R)]$ no grafo $C(\mathrm{w}_0^{A_{2n+1}})$. Vamos usar este facto para calcular o raio do grafo $C(\mathrm{w}_0^{B_n})$. Consideremos o conjunto $T_n' = \{(x,y,z) \in T_n : x = \overline{y}, \text{ ou } x = \overline{z}, \text{ ou } y = \overline{z}\}$, o qual corresponde aos ternos de T_n que estão associados à transposição de um par $(\overline{k},k)$.

Lema 5.7. *Sejam $a,b \in R(\mathrm{w}_0^{B_n})$.*

1. *Se $a \overset{L_1}{\sim} b$ então $T(a,xyz) = T(b,xyz)$ para todo o terno $(x,y,z) \in T_n'$.*
2. *Se $a \overset{L_2}{\sim} b$, então $T(a,xyz) = T(b,xyz)$ para todo o terno $(x,y,z) \in T_n \setminus T_n'$*

Demonstração. Suponhamos que $a \overset{L_1}{\sim} b$. Podemos então escrever

$$a = p_1 \cdot i\ (i+1)\ i \cdot p_2$$
$$b = p_1 \cdot (i+1)\ i\ (i+1) \cdot p_2$$

para algum $i \in [n-1]$ e p_1, p_2 palavras no alfabeto $[0,n]$. Pelo Lema 5.4, $T(a,xyz) = T(b,xyz)$ para todo o terno $(x,y,z) \in T_n$, excepto para dois ternos. De facto, esses ternos são (x',y',z') e $(\overline{z'},\overline{y'},\overline{x'})$, com $x' = i^{p_1}$, $y' = (i+1)^{p_1}$ $z' = (i+2)^{p_1}$. Como $i \neq 0$, os geradores dos fatores $i\ (i+1)\ i$ e $(i+1)\ i\ (i+1)$ de a e b, respetivamente, não vão tranpôr pares da forma $(\overline{k},k)$, pelo que (x',y',z') e $(\overline{z'},\overline{y'},\overline{x'})$ não estão no conjunto T_n'.

Com argumentos análogos se prova que, se $a \overset{L_2}{\sim} b$, então $T(a,xyz) = T(b,xyz)$ para todo o terno $(x,y,z) \in T_n \setminus T_n'$. □

Lema 5.8. *Se $a \in R(\mathrm{w}_0^{B_n})$, então a e a^R diferem por pelo menos $\frac{n(n+1)}{2}$ relações longas do tipo 2, e pelo menos $\frac{n(n+1)(4n-4)}{6}$ relações longas do tipo 1.*

Demonstração. Pela Proposição 5.6, $T(a,xyz) = -T(a^R,xyz)$ para todo $(x,y,z) \in T_n$, e pelos Lemas 5.4 e 5.7, existem pelo menos $\frac{1}{4}|T'_n| + \frac{1}{2}|T_n \setminus T'_n|$ relações longas em qualquer caminho entre a e a^R. É fácil de verificar que $|T'_n| = \frac{4n(n+1)}{2}$, pelo que um caminho entre a e a^R tem pelo menos $\frac{n(n+1)}{2}$ relações longas do tipo 2. Como

$$|T_n \setminus T'_n| = \frac{(2n+2)(2n+1)2n}{6} - \frac{4n(n+1)}{2} = \frac{2n(n+1)(4n-4)}{6},$$

existem pelo menos $\frac{n(n+1)(4n-4)}{6}$ relações longas do tipo 1 entre qualquer caminho que une a e a^R. □

O próximo resultado é consequência do lema anterior e do Teorema 4.19.

Teorema 5.9. *O raio do grafo $C(\mathrm{w}_0^{B_n})$ é* $\dfrac{n(n+1)(4n-1)}{6}$.

Demonstração. Do lema anterior, para toda a palavra $a \in R(\mathrm{w}_0^{B_n})$ nós temos que

$$d([a],[a^R]) \geq \frac{n(n+1)}{2} + \frac{n(n+1)(4n-4)}{6} = \frac{n(n+1)(4n-1)}{6}. \tag{5.1}$$

Como o membro direito da equação (5.1) foi provado, no Teorema 4.19, ser o diâmetro de $C(\mathrm{w}_0^{B_n})$, temos que $d([a],[a^R]) = \frac{n(n+1)(4n-1)}{6}$, ou seja, a excentricidade de qualquer classe de comutação é igual ao diâmetro do grafo, o que prova que este número também é o raio do grafo. □

Capítulo 6

Planaridade e átomos

Na Figura 2.2 está representado o grafo $C(\mathrm{w}_0^{B_2})$, que é claramente planar. Neste capítulo iremos mostrar que para $n > 2$ o grafo $C(\mathrm{w}_0^{B_n})$ não é planar, usando o Teorema de Wagner [6]. Uma *contração* de uma aresta $e = \{u, v\}$ num grafo é o grafo que se obtém combinando os vértices u e v num único vértice que será adjacente a todos os vértices que eram adjacentes a u e v no grafo original. Um *menor* de um grafo é um novo grafo obtido através da eliminação de arestas, e/ou de contração de arestas do grafo original. O Teorema de Wagner afirma que um grafo é planar se e só se não contém K_5 ou $K_{3,3}$ como menor.

Lema 6.1. *O grafo $C(\mathrm{w}_0^{B_3})$ não é planar.*

Demonstração. O menor de $C(\mathrm{w}_0^{B_3})$ tendo como vértices os conjuntos:
$A = \{[1012101032101323]\}$,
$B = \{[1012101232101023]\}$,
$C = \{[1021021023210123]\}$,
$D = \{[1012101023210123]\}$,
$E = \{[1021021032101323], [1021021321010323], [1012101321010323]\}$,
$F = \{[1021021232101023], [1021012132101023], [1210102132101023], [2102102132101023],$
$[2101210132101023], [2101210103210123], [2101021013210123], [0210121013210123],$
$[0210210213210123], [0121010213210123], [0102101213210123], [0102102123210123],$
$[0101210123210123], [0101210132101323], [1010210132101323]\}$,
onde E e F são a contração dos seus vértices, é isomorfo a um $K_{3,3}$ dado que A, B, C estão conectados a todos os vértices D, E, F. Pelo Teorema de Wagner, concluímos que $C(\mathrm{w}_0^{B_3})$ não é planar. □

Na Figura 6.1 está representado o menor de $C(\mathrm{w}_0^{B_3})$ isomorfo a $K_{3,3}$, onde as arestas pretas representam relações longas do tipo 1, as arestas a tracejado representam relações longas do tipo 2, e os vértices dados pelo lema anterior, com E_i e F_j o i-ésimo e j-ésimo elementos de E e F, respetivamente, com $E_0 = E$ e $F_0 = F$.

Lema 6.2. *Se $n > 1$, então o grafo $C(\mathrm{w}_0^{B_{n-1}})$ é um sub-grafo de $C(\mathrm{w}_0^{B_n})$.*

Demonstração. Note-se que dada $a \in R(\mathrm{w}_0^{B_{n-1}})$, a palavra $a \cdot n! \cdot 123 \cdots n$ é uma palavra reduzida de $\mathrm{w}_0^{B_n}$ visto que possui $n^2 + 2n + 1 = (n+1)^2$ letras, e que corresponde à permutação $\mathrm{w}_0^{B_n}$. Logo, o sub-

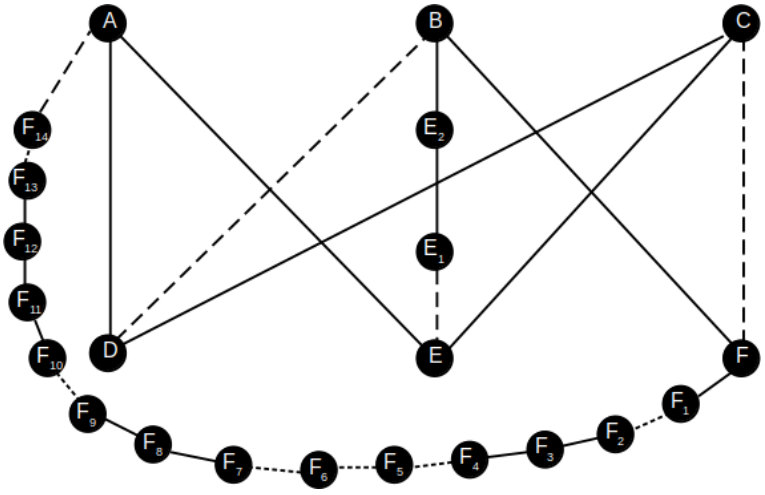

Figura 6.1 Menor de $C(\mathrm{w}_0^{B_3})$ isomorfo a $K_{3,3}$.

grafo de $C(\mathrm{w}_0^{B_n})$ formado pelas classes de comutação das palavras $a \cdot n! \cdot 123\cdots n$ com $a \in R(\mathrm{w}_0^{B_{n-1}})$ é isomorfo a $C(\mathrm{w}_0^{B_{n-1}})$. □

Dos Lemas 6.1 e 6.2 temos o seguinte resultado.

Teorema 6.3. *Para $n > 2$ o grafo $C(\mathrm{w}_0^{B_n})$ não é planar.*

A palavra $010121012 \in R(\mathrm{w}_0^{B_2})$ possui a propriedade de que $[010121012] = \{010121012\}$, isto é, a sua classe de comutação possui um único elemento. Uma classe de comutação que possui apenas uma palavra é chamada de *átomo*. É fácil de ver que uma palavra reduzida é um átomo se e só se todo o seu fator de comprimento 2 é formado por gerador com índices consecutivos. Em [14] provou-se que $C(\mathrm{w}_0^{A_n})$ possui apenas 4 átomos, para todo $n > 2$. Um dos passos dessa prova foi mostrar que os átomos de $C(\mathrm{w}_0^{A_n})$ possuiam certos fatores para poderem ser átomos. Para o caso do grafo $C(\mathrm{w}_0^{B_n})$ vamos usar um raciocínio análogo.

Lema 6.4. *A palavra $n!(12...n)$ é a única palavra reduzida de entre todas as palavras de comprimento $\geq 2n+1$ no alfabeto $[0,n]$ que possui o gerador n como gerador mais à direita e mais à esquerda, possui pelo menos um gerador 0 e todos os seus fatores de comprimento 2 são formados por geradores com índices consecutivos.*

Demonstração. Comecemos por considerar as palavras nas condições do enunciado que possuem exatemente um gerador 0. Se a for uma dessas palavras, então podemos escrever $a = p_1\ 0\ p_2$, com p_1 e p_2 palavras no alfabeto $[n]$, onde o gerador mais à esquerda de p_1 é n e o gerador mais à direita de p_1 é 1, e o gerador mais à esquerda de p_2 é 1 e o gerador mais à direita de p_2 é n. Foi provado em [14] que a palavra $12\cdots n$ (respetivamente, $n(n-1)\cdots 1$) é a única palavra reduzida com comprimento $\geq n$ no alfabeto $[n]$, tendo como gerador mais à esquerda 1 e o gerador mais à direita n (respetivamente, tendo como gerador mais à esquerda n e o gerador mais à direita 1), e onde todo o seu fator de comprimento 2 é formado por geradores com índices consecutivos. Como o comprimento de p_1 e p_2 é $\geq n$ e como são palavras no alfabeto $[n]$, vem que $p_1 = n(n-1)\cdots 1$ e $p_2 = 12\cdots n$, logo $a = n!(12...n)$.

Suponhamos agora que a é uma palavra nas condições do enunciado com exatamente dois geradores 0. Precisamos de mostrar que neste caso a não é reduzida. Podemos escrever a da forma

$$a = p_1\ 0\ p_2\ 0\ p_3,$$

com p_1, p_2 e p_3 palavras no alfabeto $[n]$. Usando argumentos anteriores, temos que $p_1 = n(n-1)\cdots 1$ e $p_3 = 12\cdots n$, pelo que

$$a = n!\ p_2\ 0\ (12\cdots n).$$

Como $1^{n!} = \overline{n+1}$, esta letra não pode estar na primeira posição da notação em janela de $n!\cdot p_2$, pelo que existe $k \in [n+1]\setminus\{1\}$ tal que $k^{n!\cdot p_2} = \overline{n+1}$ o que implica que $k^{n!\cdot p_2\cdot 01\cdots(k-1)} = \overline{n+1} < (k-1)^{n!\cdot p_2\cdot 01\cdots(k-1)}$. Pela Proposição 3.2, a palavra $n!\cdot p_2\cdot 01\cdots(k-1)$ não é reduzida, o que implica que a também não é reduzida, contradizendo a hipótese. Se a tiver mais do que dois geradores 0, usando um argumento indutivo prova-se que a continua a não ser reduzida. □

Lema 6.5. *Seja $a \in R(\mathrm{w}_0^{B_n})$. Se $a = p_1\cdot n!(12...n)\cdot p_2$, com p_1 e p_2 no alfabeto $[0,n]$, então a concatenação de p_1 e p_2 é uma palavra reduzida de $\mathrm{w}_0^{B_{n-1}}$, isto é, $p_1\cdot p_2 \in R(\mathrm{w}_0^{B_{n-1}})$.*

Demonstração. Vamos começar por provar que $(n+1)^{p_1} = n+1$. Se $(n+1)^{p_1} = k$ com $k < n+1$, então temos dois casos

1. $i^{p_1} = n+1$ para algum $i \in [n]$.
2. $i^{p_1} = \overline{n+1}$ para algum $i \in [n]$.

No primeiro caso temos que $i^{p_1\cdot n(n-1)\cdots(i+1)} = n+1 > (i+1)^{p_1\cdot n(n-1)\cdots(i+1)} = k$, uma vez que o fator $n(n-1)\cdots(i+1)$ "empurra" a letra $(n+1)^{p_1} = k$ para a posição $i+1$, e no segundo caso temos que $i^{p_1\cdot n!1\cdots(i-1)} = \overline{k} > (i+1)^{p_1\cdot n!1\cdots(i-1)} = \overline{n+1}$. A Proposição 3.2 implica que no primeiro caso a palavra $p_1\cdot n(n-1)\cdots(i+1)i$ não seja reduzida, e no segundo caso a palavra $p_1\cdot n!1\cdots(i-1)i$ não é reduzida, o que contradiz a hipótese de a ser reduzida. Logo, temos de ter $(n+1)^{p_1} = n+1$ sendo que a permutação associada a $n!(12...n)$ atua em p_1 mudando o sinal da letra $n+1$. Isto implica que $p_1\cdot p_2$ atua na permutação identidade mudando os sinais das letras 1, 2, ..., n. Como p_1p_2 tem $(n+1)^2-(2n+1) = n^2$ letras, concluimos que é uma palavra reduzida de $\mathrm{w}_0^{B_{n-1}}$. □

Lema 6.6. *Se a é um átomo de $C(\mathrm{w}_0^{B_n})$, então existem palavras p_1, p_2 no alfabeto $[0,n-1]$ tais que $a = p_1\cdot n!(12...n)\cdot p_2$ com $p_1\cdot p_2 \in R(\mathrm{w}_0^{B_{n-1}})$.*

Demonstração. Consideremos a primeira e última ocorrência do gerador n em a. Entre esses dois geradores tem de existir um gerador 0 que é responsável por mudar o sinal de $n+1$. Podemos assim escrever a da forma $a = p_1\cdot n\cdot p_3\cdot 0\cdot p_4\cdot n\cdot p_2$ com p_1, p_2 palavras no alfabeto $[0,n-1]$ e p_3, p_4 palavras no alfabeto $[0,n]$. Como a é um átomo, todo o seu fator de comprimento 2 é formado por geradores com índices consecutivos, o que, pelo Lema 6.4, se tem $n\ p_3\ 0\ p_4\ n = n!(12\cdots n)$. Finalmente, pelo lema anterior temos que $p_1\cdot p_2$ é uma palavra reduzida $\mathrm{w}_0^{B_{n-1}}$. □

Vamos agora introduzir um conjunto especial de palavras reduzidas, conjunto este que possui os átomos que procuramos.

Definição 6.7. Dada uma permutação $\mathrm{v} \in \mathfrak{S}_{n+1}$ seja $w_\mathrm{v} = \prod_{i=1}^{n+1} \mathfrak{f}_{\mathrm{v}(i)}$, onde $\mathfrak{f}_{j+1} = j!(12\cdots j)$. Dizemos que w_v é uma *palavra ordenada*, e definimos $O(n) = \{w_\mathrm{v} : \mathrm{v} \in \mathfrak{S}_{n+1}\}$, o conjunto de todas as palavras ordenadas.

A permutação associada a uma palavra da forma $i!(12\cdots i)$, com $i \in [0,n]$ atua numa permutação v trocando o sinal da i-ésima letra da notação em janela de v.

Exemplo 6.8. Se $\mathrm{v} = (2,4,1,3)$, então a palavra ordenada que se obtém de v é

$$w_\mathrm{v} = \mathfrak{f}_2\mathfrak{f}_4\mathfrak{f}_1\mathfrak{f}_3 = 101 \cdot 3210123 \cdot 0 \cdot 21012.$$

Proposição 6.9. *Toda a palavra ordenada $w \in O(n)$ é uma palavra reduzida de* $\mathrm{w}_0^{B_n}$.

Demonstração. Temos que $\mathfrak{f}_{i+1}$ possui comprimento $2i+1$, logo qualquer palavra ordenada possui comprimento

$$\begin{aligned}\sum_{i=0}^{n}(2i+1) &= 2\sum_{i=0}^{n} i + n + 1 \\ &= \frac{2n(n+1)}{2} + n + 1 = (n+1)^2.\end{aligned}$$

A permutação associada a $\mathfrak{f}_{i+1}$ atua numa permutação mudando o sinal da $(i+1)$-ésima letra da sua notação em janela. Como $\mathrm{v} \in \mathfrak{S}_{n+1}$, o fator $\mathfrak{f}_{i+1}$ aparece exatamente uma vez em w_v, para todo $i \in [0,n]$, logo w_v é uma palavra reduzida de $\mathrm{w}_0^{B_n}$. □

É fácil de verificar que a ordem pela qual as letras mudam de sinal durante o processo de transformação da permutação identidade na permutação $\mathrm{w}_0^{B_n}$ é diferente para cada palavra ordenada. Este facto vai implicar a que cada palavra ordenada pertença a uma classe diferente, como veremos de seguida.

Definição 6.10. Dada uma palavra $a \in R(\mathrm{w}_0^{B_n})$ definimos $ord(a) := (k_1, k_2, \ldots, k_{n+1})$ onde $k_i \in [n+1]$ é a i-ésima letra a mudar de sinal no processo de transformação da permutação identidade em $\mathrm{w}_0^{B_n}$ usando a.

Podemos obter $ord(a)$ através do diagrama em linha de a, observando a ordem pela qual os pares da forma $(\bar{k},k)$ são transpostos, para todo $k \in [n+1]$. No caso de $a = 101210102$, temos $ord(a) = (2,3,1)$ (ver Figura 3.1).

Lema 6.11. *Seja $a,b \in R(\mathrm{w}_0^{B_n})$ tal que $a \sim b$. Então $ord(a) = ord(b)$.*

Demonstração. Suponhamos, sem perda de generalidade, que $a = p_1 \cdot i\ j \cdot p_2$ e $b = p_1 \cdot j\ i \cdot p_2$, para alguns $i,j \in [0,n]$ tais que $|i-j| > 1$, com p_1 e p_2 palavras no alfabeto $[0,n]$. Se $i \neq 0 \neq j$, então todos os $n+1$ geradores 0 estão nos fatores p_1 e p_2. Neste caso, temos $ord(a) = ord(b)$, visto que $p_1 \cdot i\ j$ corresponde à mesma permutação associada a $p_1 \cdot j\ i$. Caso contrário, se i ou j for 0, então também teremos $ord(a) = ord(b)$ porque i e j atuam em letras distintas na notação em janela de p_1. □

Proposição 6.12. *O conjunto $O(n)$ tem $(n+1)!$ palavras reduzidas, cada uma pertencendo a uma classe de comutação diferente.*

Demonstração. É fácil de ver que pela construção das palavras ordenadas, $|O(n)| = |\mathfrak{S}_{n+1}| = (n+1)!$. Para mostrar que toda a palavra ordenada pertence a uma classe de comutação diferente basta notar que a ordem pela qual as letras mudam de sinal durante o processo de transformação da permutação identidade na permutação $\mathrm{w}_0^{B_n}$ é diferente para cada palavra ordenada, isto é, $ord(w_\mathrm{u}) = ord(w_\mathrm{v})$ se e só se $\mathrm{u} = v$. Logo, pelo Lema 6.11, temos o resultado. □

Teorema 6.13. *Para $n \geq 1$ existem exatamente dois átomos de $C(\mathrm{w}_0^{B_n})$, sendo eles w_{id} e $w_{\mathrm{w}_0^{A_n}}$.*

Demonstração. Vamos usar indução para mostrar o resultado. O grafo $C(\mathrm{w}_0^{B_1})$ tem apenas duas classes de comutação, sendo elas $[0101]$ e $[1010]$, e ambas são átomos, com $0101 = w_{id}$ e $1010 = w_{\mathrm{w}_0^{A_1}}$. Suponhamos agora que o resultado é válido até $n = k-1$, e consideramos o grafo $C(\mathrm{w}_0^{B_k})$. Pelo Lema 6.6, se a é um átomo de $C(\mathrm{w}_0^{B_k})$, então $a = p_1 \cdot k!(12\cdots k) \cdot p_2$ com $p_1 \cdot p_2$ uma palavra reduzida de $\mathrm{w}_0^{B_{k-1}}$. Como a é um átomo, se p_1 e p_2 forem palavras não vazias, o gerador mais à direita de p_1 e o gerador mais à esquerda p_2 têm de ser iguais a $k-1$, o que implicava que $p_1 \cdot p_2$ não fosse uma palavra reduzida. Logo, ou p_1 ou p_2 é a palavra vazia. Se p_1 for a palavra vazia, então $a = k!(12\cdots k)p_2$ com p_2 uma palavra reduzida de $\mathrm{w}_0^{B_{k-1}}$. Todo o fator de comprimento 2 de a é formado por geradores com índices consecutivos, o que implica que p_2 seja um átomo de $C(\mathrm{w}_0^{B_{k-1}})$. Por hipótese de indução, temos apenas duas possibilidades pelo que p_2 tem de ser a palavra $w_{\mathrm{w}_0^{A_{k-1}}}$, para que a continue a ser um átomo , logo $a = k!(12\cdots k)w_{\mathrm{w}_0^{A_{k-1}}} = w_{\mathrm{w}_0^{A_k}}$. Se p_2 fosse a palavra vazia, com argumentos análogos se prova que $a = w_{id}$. □

Capítulo 7

Componente Computacional

Neste capítulo vamos descrever a componente computacional deste trabalho, que está profundamente interligada com os capítulos mais teóricos anteriormente desenvolvidos. Este capítulo vai ser dividido em duas partes: a primeira parte é dedicada à descrição dos algoritmos usados na determinação das classes de comutação, com especial enfâse no último (o mais eficiente), e a segunda parte dedicada à aplicação que foi desenvolvida no âmbito desta tese e que permitiu visualizar o grafo $C(\mathrm{w}_0^{B_n})$ para $n = 2$ e $n = 3$. De facto, a visualização do grafo $C(\mathrm{w}_0^{B_n})$ permitiu intuir algumas das novas propriedades referidas nos capítulos anteriores, uma vez que estas estruturas possuem grandes dimensões o que torna inexequível a sua construção de forma manual (ver Tabelas 7.1 e 7.2). Contudo, o estudo teórico destas estruturas permitiu desenvolver versões cada vez mais eficientes do algoritmo usado na determinação do número de classes de comutação de $C(\mathrm{w}_0^{B_n})$, o que torna bem visível a interligação entre a componente prática e teórica deste trabalho. De salientar que estes algoritmos foram executados utilizando o computador *chita 3* do Laboratório de Cálculo do Departamento de Matemática da Universidade de Coimbra, o qual tem as seguintes características:

- Processador: Intel i7-5820K
- Velocidade do processador: 3.30GHz
- Memória Ram: 64GB.

n	Número de classes de comutação
1	2
2	14
3	330
4	25 396
5	6 272 842
6	4 925 166 862
7	12 221 171 869 734
8	95 482 373 388 042 562

Tabela 7.1 Número de classes de comutação (sequência A180605 de [20]).

n	Dimensão da classe $[w_0]$
1	1
2	4
3	169
4	141 696

Tabela 7.2 Algumas dimensões da classe gerada por w_0.

7.1 Algoritmos de geração das classes de comutação

7.1.1 Algoritmo de pesquisa exaustiva

Quando iniciámos este trabalho, a descrição existente na literatura para construir o grafo $C(\mathrm{w}_0^{B_n})$ consistia em ir agrupando palavras reduzidas que diferem por uma sequência de comutações, formando as classes de comutação, e depois atribuindo uma aresta entre duas classes $[a]$ e $[b]$ sempre que existem palavras $a' \in [a]$ e $b' \in [b]$ que diferem por uma relação longa. Deste modo, a primeira abordagem consistiu na construção de um algoritmo de pesquisa exaustiva que determina todas as palavras reduzidas de $\mathrm{w}_0^{B_n}$ e verifica quais as que correspondem a classes de comutação diferentes. Este procedimento está esquematizado no pseudocódigo do Algoritmo 1 e é iniciado com algPE(a_0), onde a_0 é uma palavra reduzida de $\mathrm{w}_0^{B_n}$, e coloca a classe $[a_0]$ no vetor C de classes exploradas pelo algoritmo (passo 1). Cada classe é guardada numa estrutura que possui um vetor com todos os seus elementos e uma variável onde é guardada a menor palavra na ordem lexicográfica dessa classe, à qual chamamos representante da classe, que é acedido com a função $lex(a)$. A seguir, são geradas todas as palavras da classe $[a_0]$ (passo 2) e para cada palavra a' de $[a_0]$ (passo 3) pesquisa-se todas as possíveis relações longas que permitem gerar palavras b em novas classes (passo 4). Usaremos a notação $a' \overset{L_1 \vee L_2}{\sim} b$ se $a' \overset{\mathrm{L}_1}{\sim} b$ ou $a' \overset{\mathrm{L}_2}{\sim} b$. Sempre que é efetuada uma relação longa em alguma palavra, o que permite sair da classe em análise, é necessário verificar se a classe desta nova palavra já foi gerada. Para isso é calculada a menor palavra na ordem lexicográfica de $[b]$, ou seja a representante da classe $[b]$ cujo o processo está descrito no Capítulo 3, e é verificado nas classes já geradas se existe alguma que tenha a mesma representante (passo 5). Se não existir, o Lema 3.10 garante que a classe $[b]$ ainda não foi gerada e o algoritmo deve explorá-la (passo 6).

Algoritmo 1: [Algoritmo de pesquisa exaustiva] algPE(a_0)

passo 1: Adicionar $[a_0]$ ao vetor C
passo 2: Gerar a classe $[a_0]$
passo 3: **PARA** cada a' na classe $[a_0]$
passo 4: **PARA** cada b tal que $b \overset{L_1 \vee L_2}{\sim} a'$
passo 5: **SE** $lex(b) \notin C$
passo 6: algPE(b)

Proposição 7.1. *O Algoritmo 1 gera todas as classes de comutação.*

Demonstração. Atendendo à descrição do grafo $C(\mathrm{w}_0^{B_n})$ e ao facto do algoritmo examinar exaustivamente todas as novas classes geradas, temos o resultado pretendido. □

Com esta versão do algoritmo foi possível calcular todas as classes de comutação até $n = 3$, tendo demorado 4 segundos no caso $n = 3$, não conseguindo finalizar para valores de n superiores pelo elevado tempo de execução que exigiria. No entanto, este método tem alguns aspetos negativos:

- necessita de muita memória uma vez que são guardados os representantes de cada classe, o que por exemplo para $n = 4$ implica guardar 25396 vetores com 25 inteiros cada um,
- gera todas as palavras reduzidas de $\mathrm{w}_0^{B_n}$, número este que é bastante elevado comparado com o número de classes de comutação (ver Tabela 7.3), exigindo grande esforço computacional,
- o processo de verificar repetições torna-se muito ineficiente à medida que o algoritmo vai gerando mais classes e piora à medida que o valor de n aumenta.

n	Número de palavras reduzidas
1	2
2	42
3	24 024
4	701 148 020
5	1 671 643 033 734 960

Tabela 7.3 Número de palavras reduzidas (sequência A039622 de [20]).

Este algoritmo, apesar das suas ineficiências, foi muito útil no ínicio deste trabalho porque forneceu-nos mais informações sobre o grafo $C(\mathrm{w}_0^{B_n})$. De facto, a análise de todas as palavras reduzidas em $R(\mathrm{w}_0^{B_n})$, permitiu desenvolver uma aplicação com interação gráfica com o utilizador que permite explorar algumas propriedades do grafo, no caso de $n = 2$ e $n = 3$. Outro aspeto positivo foi o facto deste algoritmo estar preparado para gerar o grafo a partir de qualquer palavra reduzida, o que permitiu observar algumas propriedades, nomeadamente que o grafo pode ser particionado por níveis, que o número de níveis é sempre o mesmo independemente da palavra inicial com que se começa a gerar o grafo, que o número de relações longas do tipo 1 e tipo 2 entre a classe do primeiro nível e último nível são constantes e que a classe do último nível corresponde à classe reversa da classe da palavra com que se inicia o algoritmo. Estas observações permitiram conjeturar uma fórmula para o diâmetro e o raio do grafo (Teoremas 4.19 e 5.9).

Como foi descrito no Capítulo 4, o grafo $C(\mathrm{w}_0^{B_n})$ é um grafo particionado por níveis, os quais podem ser caraterizados pela função nível N. Quando o grafo é gerado a partir da palavra w_0, o processo de verificar repetições torna-se mais eficiente como veremos de seguida na próxima versão do algoritmo.

7.1.2 Algoritmo com a função nível

A função nível foi de extrema relavância no Capítulo 4, pois permitiu determinar o diâmetro do grafo $C(\mathrm{w}_0^{B_n})$. Nesta secção vamos usar as propriedades da função nível para aprimorar o algoritmo descrito no ponto anterior.

O grafo $C(\mathrm{w}_0^{B_n})$ é um grafo particionado por níveis, isto é, podemos dividir o grafo em várias camadas sendo que cada uma delas possui apenas classes de comutação com a mesma imagem pela função N. Além disso, se duas classes estiverem ligadas por uma relação longa, então os seus níveis diferem de uma unidade. A classe $[w_0]$ é a classe que possui menor valor da função nível (Proposição 4.16), pelo que se geramos o grafo a partir desta classe apenas necessitamos de gerar classes que estejam no nível acima da classe que as gerou. Nesta nova versão do algoritmo, em vez de definirmos um vetor C que guarda todas as classes, definimos vários vetores C_k, cada um constituído apenas por classes de comutação com nível k. Deste modo, verificar se uma classe já foi gerada pelo algoritmo num passo anterior é mais eficiente. Nesta versão do algoritmo também foi desenvolvido um método de encontrar as classes adjacentes de uma classe de comutação sem que seja necessário gerar a classe por inteiro. Este método consiste em analisar todos os geradores da representante da classe, da esquerda para a direita, sendo que para cada gerador i_j, o algoritmo vai pesquisar nessa palavra se existem geradores à direita de i_j que permitam formar um fator da forma $i_j i_{j'} i_{j''}$, com $i_j = i_{j''}$ e $|i_j - i_{j'}| = 1$, ou um fator da forma $i_j i_{j'} i_{j''} i_{j'''} = 0101$, usando comutações de geradores válidas. No Algoritmo 2 está representado o pseudocódigo desta versão, sendo inicializada com a palavra $a = w_0$. Este algoritmo é muito semelhante ao anterior, diferindo apenas a verificação das classes repetidas, que é feita no vetor correspondente ao nivel da classe em analise.

Algoritmo 2: [Algoritmo com a funçaõ nível] algFN(a)

passo 1: Adicionar $[a]$ à lista $C_{N(a)}$
passo 2: **PARA** cada b tal que $b \overset{L_1 \vee L_2}{\sim} a'$ e $N(b) = N(a)+1$, com $a' \in [a]$
passo 3: **SE** $lex(b) \notin C_{N(b)}$
passo 4: algFN(b)

Proposição 7.2. *O Algoritmo 2 determina todas as classes de comutação.*

Demonstração. Existem duas diferenças entre este algoritmo e o anterior. A primeira consiste em não gerar explicitamente todos os elementos da classe, apesar de serem pesquisadas (de forma mais eficiente) todas as relações longas que permitem aumentar o valor da função N. A segunda diferença consiste no processo de verificação se uma nova palavra gerada a partir de a pertence a uma classe repetida uma vez que só são analisadas as classes que estão em $C_{N(a)+1}$. Porém, as Proposições 4.9 e 4.16 indicam que as restantes comparações são desnecessárias. □

Com esta implementação foi possível calcular todas as classes de comutação num período razoável de tempo até $n = 5$, sendo que para este último valor de n o algoritmo demorou cerca de 3 dias. Apesar desta versão ser mais eficiente que a pesquisa exaustiva, continua a ser necessário guardar todas as representantes de cada classe e o processo de verificar repetições também se torna menos eficiente à medida que o valor de n aumenta. No Gráfico 7.1 pode ser vista uma análise destas duas primeiras versões do algoritmo em termos do número de comparações que é feito para verificar repetições. De facto, um dos grandes problemas do algoritmo é exatamente a verificação de repetições. Mesmo

nesta versão, é necessário fazer muitas comparações pois o número de classes em cada nível aumenta significativamente com o aumento do n. Para resolver este problema foi necessário desenvolver um algoritmo com uma abordagem completamente diferente.

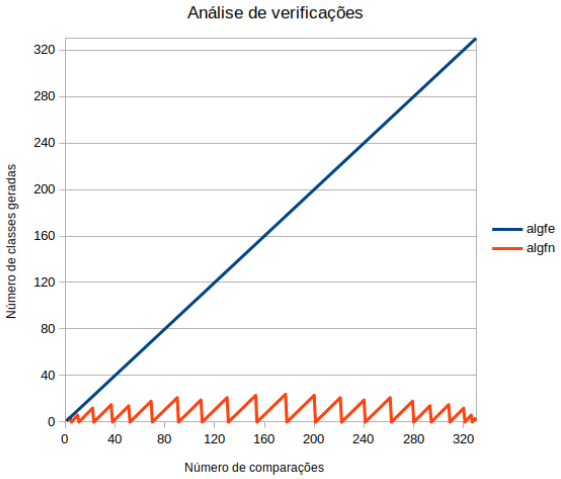

Figura 7.1 Número de comparações efetudas pelos algoritmos algFE e algFN.

7.1.3 Algoritmo de construção lexicográfica

As duas versões descritas até ao momento obrigam verificar a existência de repetições sempre que é gerada uma nova classe. A próxima abordagem que aqui vamos apresentar vai ser um método recursivo que não necessita dessa verificação e, por isso, evita o armazenamento de todos os representantes das classes. Contrariamente às versões anteriores, neste método vamos partir da palavra vazia e vamos construindo letra a letra todas as representantes de cada classe. Assim, existem $n+1$ possibilidades para o primeiro gerador de alguma representante. Seja $p = i_1$ com $i_1 \in [0,n]$. Uma vez que queremos gerar todas as menores palavras na ordem lexicográfica de cada classe, só podemos concatenar p com outro gerador i_2 se $i_2 \in \{i_1 - 1\} \cup [i_1 + 1, n]$. Obtemos assim $p' = i_1 i_2$. Do mesmo modo, p' pode ser concatenado com um gerador i_3 se $i_3 \in \{i_2 - 1\} \cup [i_2 + 1, n]$, formando assim $p'' = i_1 i_2 i_3$. Para concatenar geradores com p'', o processo difere do descrito nos passos anteriores, uma vez que a escolha do próximo gerador pode fazer com que a palavra deixe de ser reduzida. Por exemplo, se $p'' = 212$ e concatenássemos com o gerador 1 obtiamos $p''' = 2121$, o que não pode acontecer pois p''' não é uma palavra reduzida. Para evitar este caso é necessário guardar também a permutação associada à palavra que estamos a construir. Deste modo, usando a Proposição 3.2, podemos analisar a escolha dos geradores e impedir que a palavra deixe de ser reduzida. Logo percorremos todas as hipóteses possíveis para representantes de classes sem repetições.

O algoritmo começa por inicializar um vetor vazio p de dimensão $(n+1)^2$ (passo 1) e um vetor *perm* que no ínicio guarda a permutação identidade (passo 2). Depois o algoritmo vai construir todas as representantes de cada classe que têm como gerador mais à esquerda o gerador i, com $i \in [0,n]$ (passo 3), sendo usada uma função recursiva que tem como input um gerador e que verifica se é possível concatenar esse gerador com a palavra guardada em p. Cada chamada recursiva começa por

fazer uma análise de um gerador $i \in [0,n]$ a concatenar (passo A), usando as ideias da Proposição 3.2. Se a análise for positiva (passo B), então o gerador é inserido na primeira posição livre de p (passo C) e é atualizada a permutação associada à nova palavra (passo D). Se p ficar com $(n+1)^2$ geradores (passo E), então estamos perante uma representante de uma nova classe, pelo que o contador de classes é incrementado (passo F). Caso contrário, para cada gerador $j \in \{i-1\} \cup [i+1,n]$ (passo G), o algoritmo vai explorar se é possível inserir tal gerador na palavra p. No final, é retirado o gerador i da palavra p (passo H) e é atualizada a permutação (passo I). No Algoritmo 3 está descrito o pseudocódigo referente a esta versão.

Algoritmo 3: [Algoritmo de construção léxicográfica] algCL(n)

```
        contador ← 0
passo 1: p ← [ ]
passo 2: perm← [1,2,...n+1]
passo 3: PARA cada gerador 0 ≤ i ≤ n :
             construir_palavra(i)

      construir_palavra(i)
passo A: SE (perm[i] < perm[i+1] E i > 0) OU (perm[i] > 0 E i = 0)
passo B:      p ← p·i
passo C:      Atualizar permutação // se i>0 trocar perm[i] com perm[i+1],
    caso contrário perm[1]=-perm[1]
passo D:      SE p possuir (n+1)^2 geradores
passo E:           contador ← contador+1 // Foi gerada um nova representante
              SENÃO
passo F:           PARA cada gerador j em {i-1} ∪ [i+1,n]
passo G:                construir_palavra(j)
passo H:      Desfazer palavra // retirar o último gerador de p
passo I:      Desfazer permutação // se i>0 trocar perm[i] com perm[i+1],
    caso contrário perm[1]=-perm[1]
```

Proposição 7.3. *O Algoritmo 3 determina todas as classes de comutação.*

Demonstração. Cada classe de comutação pode ser representada pelo seu menor elemento na ordem lexicográfica. Uma vez que este algoritmo percorre todas as palavras lexicograficamente menores em relação às palavras que diferem apenas de relações de comutação e ignora as palavras lexicograficamente menores que não são reduzidas, todas as classes de comutação são geradas. □

Com esta versão foi possível calcular todas as classes de comutação até $n = 6$. De salientar que para $n = 5$, o algoritmo demorou 3 minutos a gerar todas as classes, uma melhoria substancial em relação à versão anterior que, para este valor de n demorava cerca de 3 dias. Com o intuito de melhorar os tempos de computação, foi também desenvolvida uma versão não recursiva do algoritmo. Contudo, a diferença de tempo não foi significativa, não permitindo obter o número de classes de comutação para valores de n superiores. Apesar deste método não precisar de guardar as representantes de cada classe e de não ser necessário verificar a existência de repetições, podemos melhorar ainda mais o

algoritmo. Note-se que nesta versão construimos as representantes das classes letra a letra. Se nos fixarmos no caso de $n = 4$, existem 25396 classes de comutação. Cada chamada recursiva acresenta uma letra à palavra, logo se houver uma chamada recursiva para cada letra de cada representante seria necessário haver $25396 \cdot 25 = 634900$ chamadas recursivas. Contudo, o algoritmo efetua 821 564 chamadas recursivas, isto porque alguns geradores que até poderiam deixar a palavra como menor palavra na ordem lexicografica fazem com que a palavra deixe de ser reduzida. Na próxima versão vamos descrever um algoritmo que não precisa de tantas chamadas recursivas e que usa também as propriedades da função nível.

7.1.4 Algoritmo de novas ligações

A ideia geral deste algoritmo baseia-se na propriedade de o grafo ser conexo e, por isso, existir sempre um caminho entre qualquer classe $[a]$ e $[w_0]$. Como esse caminho pode não ser único, vamos estabelecer a regra que cada vez que subimos de nível no grafo utilizamos a relação longa mais à direita possível. Deste modo, estabelecemos um único caminho (que verifica esta regra) de $[w_0]$ para $[a]$ o que evita gerar classes repetidas. Assim, o algoritmo irá partir de w_0 e gerar novas palavras reduzidas utilizando relações longas associadas a w_0. Seguidamente, irá explorar cada uma dessas palavras a expandido-as com uma relação longa para obter uma palavra reduzida b numa classe no nível acima e de forma a que essa relação longa seja a que se encontra mais à direita em b. Deste modo não é necessário reconstruir a palavra b partindo da palavra vazia (como no algoritmo anterior) e o conjunto de relações longas associadas a b é facilmente deduzido do conjunto de relações longas associado a a. Para descrever esta versão do algoritmo vamos introduzir algumas notações que vão facilitar a sua compreensão. Dada uma palavra reduzida $a = i_1 i_2 \cdots i_l$, dizemos que ela *suporta* uma relação longa na posição j se existir $a' \in [a]$ tal que $a' = p_1 \cdot i_{j'} i_j i_{j''} \cdot p_2$ com $i_{j'} = i_{j''}$ e $|i_j - i_{j'}| = 1$, ou se existir $a' = p_1 \cdot i_{j'} i_j i_{j''} i_{j'''} \cdot p_2$ com $i_{j'} i_j i_{j''} i_{j'''} = 0101$ ou $i_{j'} i_j i_{j''} i_{j'''} = 1010$. Temos então que $a' \overset{L_1 \vee L_2}{\sim} b$. Se $N(a') < N(b)$ dizemos que essa relação longa é *positiva* e escrevemos $a' \overset{L^+}{\sim} b$, caso contrário dizemos que essa relação longa é *negativa* e escrevemos $a' \overset{L^-}{\sim} b$. Vamos considerar o conjunto $L(a)$ constituído por todas as posições onde a suporta relações longas. Deste modo, uma relação longa pode ser identificada pela sua posição central, isto é, a posição do segundo gerador do fator onde podemos aplicar uma relação longa. Para simplificar a utilização do algoritmo, armazenaremos adicionalmente a posição de todos geradores, pelo que a relação longa vai ser representada pelo vetor $[j', j, j'']$ ou $[j', j, j'', j''']$ dependendo se se trata de uma relação longa do tipo 1 ou 2.

Exemplo 7.4. Seja $a = 010210121$. Temos que a suporta uma relação longa do tipo 2 na posição 5 que é positiva, uma vez que $a = 0102\mathbf{1}0121 \sim 0120\mathbf{1}0121$, e suporta uma relação longa do tipo 1 na posição 8 que é negativa, logo $L(a) = \{[3,5,6,7],[7,8,9]\}$

É fácil de verificar que se $a \sim b$, então $|L(a)| = |L(b)|$. Pela Proposição 4.16, se $a \notin [w_0]$, então a tem de suportar uma relação longa negativa. Consideremos então a relação longa negativa suportada por a na posição mais à direita possível. Contudo, pode acontecer o caso em que a suporta duas relações do tipo 1 negativas nessa posição, isto é, pode existir um gerador i que é o gerador central de duas relações longas do tipo 1. Nesse caso, vamos considerar a relação longa que admite um fator da forma $(i+1)i(i+1)$, para $i \in [n-1]$, uma vez que na menor palavra em relação à ordem lexicográfica,

os geradores desse fator estão mais à direita. Temos então que a suporta uma única relação longa negativa, nas condições descritas.

Exemplo 7.5. A relação longa negativa mais à direita suportada por $a = 0101321032102132$ acontece na posição 6. Visto que nesta posição a suporta duas relações longas negativas, vamos considerar a relação longa dada pelos geradores a negrito: $a = 0101\mathbf{32}10\mathbf{3}2102132$. Na palavra $b = 0102103210123212$, a relação longa negativa suportada por b mais à direita encontra-se na posição 15.

Com isto podemos concluir que para cada classe $[a]$ existe um único caminho $[a] \overset{L^-}{\sim} [a_1] \overset{L^-}{\sim} \cdots \overset{L^-}{\sim} [a_l] = [w_0]$, sendo que a relação longa que a_i suporta e que gera a classe $[a_{i+1}]$ é a relação longa negativa mais à direita. Deste modo, se cada classe for gerada por essa relação longa em específico são percorridas todas as classes e não vai haver repetições. Nesta versão vamos gerar o grafo a partir da palavra w_0 e vamos gerar todas as classes através dos caminhos anteriormente descritos.

Outro aspecto importante é a forma como o conjunto L se altera quando temos duas palavras que diferem por uma relação longa. Sejam $a, b \in R(\mathrm{w}_0^{B_n})$ tais que $a \overset{\mathrm{L}_1}{\sim} b$. Podemos então escrevrer

$$\begin{aligned} a &= p_1 \cdot i\,(i+1)\,i \cdot p_2 \\ b &= p_1 \cdot (i+1)\,i\,(i+1) \cdot p_2, \end{aligned}$$

para algum $i \in [n-1]$ e p_1, p_2 palavras no alfabeto $[0,n]$. Note-se que todas as relações longas suportadas por a que envolvem geradores do fator $i(i+1)i$ de a vão deixar de ser suportadas pela palavra b uma vez que esse fator já não existe em b. Todas as outras vão continuar a ser suportadas por b. Contudo, também pode acontecer que os geradores do fator $(i+1)i(i+1)$ passem a pertencer a relações longas suportadas por b. No caso de $a \overset{\mathrm{L}_2}{\sim} b$ o processo é análogo.

Exemplo 7.6. Seja $a = 0102103210321323$. Temos que $L(a) = \{[7,8,11],[11,12,14],[14,15,16]\}$. Se efetuarmos a relação longa suportada por a na posição 8 obtemos a palavra $b = 0102102\mathbf{3}21021323$. Note-se que b deixou de suportar uma relação longa na posição 12, uma vez que partilha o gerador na posição 11 com a relação longa suportada na posição 8. Além disso, b suporta duas novas relações longas, uma na posição 5, isto é $[4,5,7]$ e outra na posição 10, ou seja $[9,10,12]$.

Estruturas de dados

Como na versão anterior, este método também vai ser recursivo e vai usar variáveis globais que se vão alterando à medida que o algoritmo vai decorrendo. A utilização de variáveis globais não é aconselhável em programação mas neste caso justifica-se para evitar a cópia sucessiva de vetores que serão pontualmente modificados. As estruturas de dados usadas nesta versão são as seguintes:

- **word**: armazena a palavra que se encontra em análise, sendo inicializada com a palavra w_0. Nesta palavra vão ser efetuadas as relações longas e vão ser procuradas as relações que são suportadas pela palavra que está a ser analisada. Corresponde a um vetor com $(n+1)^2$ elementos.

- **signs**: guarda o sinal dos geradores da palavra que está na variável *word*. Serve para verificar se vamos subir ou descer de nível, antes de aplicarmos alguma relação longa na palavra guardada

na variável *word*. É também um vetor com $(n+1)^2$ elementos, sendo que o elemento na posição *i* de *signs* é 1 se o gerador na posição *i* de *word* for positivo, e é -1 se esse gerador for negativo. Aqui consideramos os geradores 0 como sendo negativos.

- **L**: Esta variável vai guardar as relações longas suportadas pela palavra que está na variável *word*. Mais uma vez, é um vetor com $(n+1)^2$ elementos sendo que, se a palavra que está na variável *word* suportar uma relação longa na posição *i*, então será guardado um vetor na posição *i* da lista *L* com as posições dos geradores da relação longa em questão e o tipo da relação longa. Caso contrário, atribuímos o valor de *None* a essa posição no vetor.

O programa começa por guardar a palavra w_0 na variável *word*, juntamente com os sinais dos geradores de w_0 e as relações suportadas por w_0. No caso de $n = 3$ temos os seguintes valores:

- **word**:[0,1,0,2,1,0,3,2,1,0,3,2,1,3,2,3]
- **signs**:[-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]. No caso de w_0, todos os seus geradores são negativos.
- **L**:[None, None, None, None, None, None, None, [7,8,11], None, None, None, [11,12,14], None, [14,15,16], None, None]

Será com estes dados iniciais que o algoritmo vai gerar todas as classes de comutação.

Arquitetura do programa

Já mostrámos que se cada classe for gerada através da relação longa negativa que é suportada na posição mais à direita nas condições anteriormente descritas, então não vão ser geradas classes repetidas. O algoritmo vai gerar as classes de comutação do grafo $C(\mathrm{w}_0^{B_n})$ a partir da palavra w_0 e começa por atribuir à variável *word* a palavra w_0 (passo 1), à variável *signs* os sinais dos geradores de w_0, que são todos negativos (passo 2) e à variável *L* todas as relações longas suportadas por w_0 (passo 3). Finalmente, para cada uma das relações longas em *L* (passo 4), o algoritmo var gerar as classes de comutação correspondentes com a função "gerar_classes" (passo 5). A função "gerar_classes(i, limite_1, limite_2)" vai gerar as classes efetuando em primeiro lugar a relação longa suportada na posição *i* da palavra word (passo A). De seguida, vai atualizar o vetor *L*, guardando numa variável auxiliar as relações que vão deixar de ser suportadas pela nova palavra (passo B) e adicionando as novas (passo C). Neste último passo é verificado se não existe alguma nova relação longa negativa que esteja à direita da relação longa que deu origem à palavra em *word* (passo D), caso em que, significa que esta nova classe foi gerada corretamente, e é incrementado o contador de classes (passo E). Ao aplicar a relação longa na posição *i*, o gerador que está na posição $i-1$ pode dar origem a uma nova relação longa na palavra. Desse modo, temos de explorar essa relação em particular e todas as que estão à direita da posição *i*, desde que elas sejam as relações longas negativas mais à direita das novas palavras que são geradas. Para isso, limitamos as posições onde essas ligações podem ser suportadas. O limite_2 permite restringir as posições onde pode ser suportada a relação longa que contêm o gerador da posição $i-1$ (passo F) enquanto que o limite_1 restringe as posições onde as restantes ligações podem ser suportadas (passo G). Em qualquer dos casos, é feita uma chamada recursiva desta

função para gerar as restantes classes com os respetivos limites (passo G e I). Por fim é necessário eliminar as relações longas que foram adicionadas ao vetor L (passo J), colocar de volta as relações longas que estão guardadas na variável auxiliar (passo K) e desfazer a relação longa que originou a palavra que se encontra em *word* (passo L).

Algoritmo 4: [Algoritmo de novas ligações] algNL(n)

contador $\leftarrow 0$
passo 1: $word \leftarrow w_0$
passo 2: $signs \leftarrow [-1,-1,\ldots,-1]$
passo 3: $L \leftarrow$ vetor das relações longas suportadas por w_0
passo 4: **PARA** cada i em L :
passo 5 gerar_classes(i,i,0)

gerar_classes(i,limite_1,limite_2)
passo A: Efetuar a relação longa suportada na posição i em *word*
passo B: Mover para uma variável auxiliar as relações longas suportadas por *word* que partilham geradores com a relação longa suportada na posição i
passo C: Adicionar a L as novas relações longas suportadas por *word*
passo D: **SE** não existir alguma relação longa negativa no vetor L em posições $> i$
passo E contador $\leftarrow$ contador+1
passo F: **SE** existir uma relação que envolva o gerador na posição $i-1$ e que seja suportada numa posição $j \geq$ limite_2
passo G gerar_classes(i,j,limite_2)
passo H **PARA** cada $j \geq i$ em L
passo I gerar_classes(i,j,limite_1)
passo J: Eliminar de L as relações longas que foram adicionadas no passo B
passo K: Mover da variável auxiliar para L as relações longas antigas
passo L: Desfazer a relação longa que originou a palavra em *word*

Proposição 7.7. *O Algoritmo 4 gera todas as classes de comutação.*

Demonstração. Uma vez que o algoritmo apenas gera novas palavras em que a relação que lhe deu origem é a que se encontra mais à direita e existe um único caminho de w_0 até qualquer classe de comutação que é constituído apenas por relações com estas características, então o Algoritmo 4 gera todas as classes de comutação. □

Com esta última versão do algoritmo, foram geradas todas as classes até $n = 6$, sendo que para este último n o algoritmo demorou 2 dias. Esta versão corrige todos os aspetos negativos das versões anteriores, uma vez que não necessita de guardar as representantes das classes, não necessita de gerar todas as palavras de uma classe e em cada chamada recursiva obtemos sempre uma nova palavra, sendo depois necessário verificar se foi gerada de acordo com as regras que impusemos. Na Tabela 7.4 apresenta-se o tempo médio que o algoritmo necessita para gerar cada uma das classes de comutação, podendo-se observar que esse valor médio aumenta ligeiramente com n.

n	Tempo médio de geração das classes
2	2.12e-05 segundos
3	2.62e-05 segundos
4	3.20e-05 segundos
5	4.05e-05 segundos

Tabela 7.4 Tempo médio de geração das classes de comutação para diferentes valores de n.

7.2 Exemplos de utilização

De forma a tentar ter uma maior compreensão da estrutura do grafo $C(\mathrm{w}_0^{B_n})$ e testar conjeturas, foi desenvolvido um programa na linguagem *Python* que permite visualizar o grafo para $n = 2$ e $n = 3$. Este programa possui várias funcionalidades que permitem ao utilizador explorar algumas informações sobre o grafo. Esta secção destina-se a apresentar alguns exemplos das funcionalidades da aplicação desenvolvida. Ao executar-se no terminal o comando *phyton3 permutacao_sinal.py*, é iniciado o algoritmo que gera todas as classes de comutação para $n = 3$ e é apresentado o grafo correspondente, representado na Figura 7.2. Nesta figura, o número dentro de cada classe representa o identificador (id) da classe correspondente, enquanto as arestas pretas e vermelhas representam relações longas do tipo 1 e do tipo 2, respetivamente. Note-se também que no canto superior esquerdo da Figura 7.2 estão os menus com todas as funcionalidades do programa.

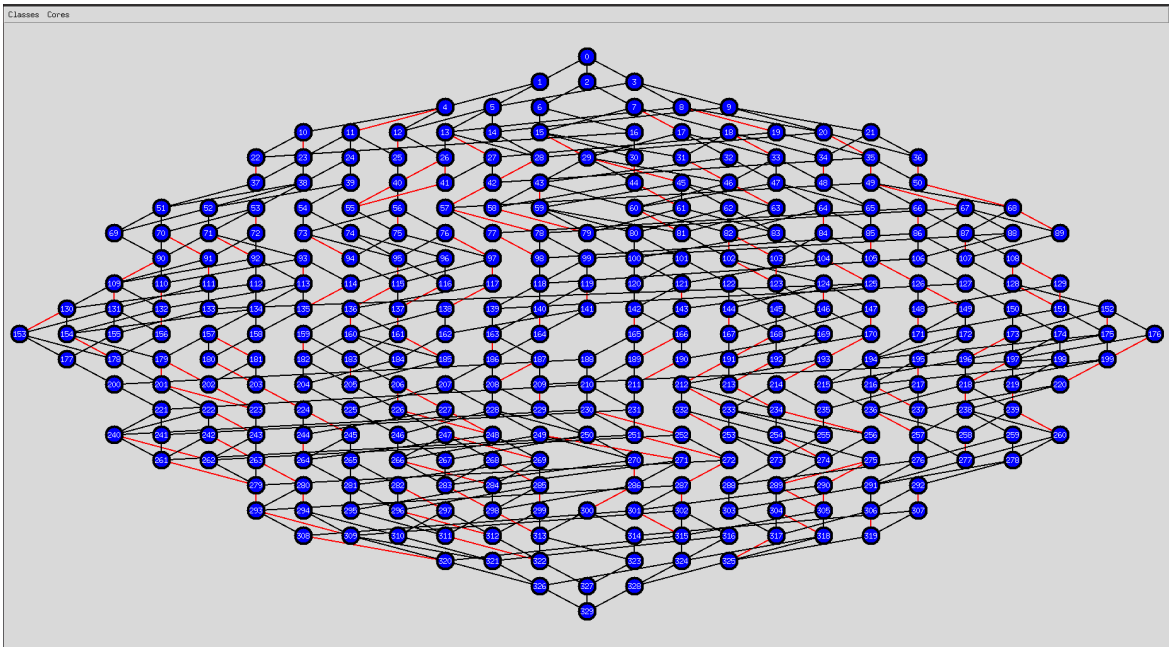

Figura 7.2 Aspeto inicial da aplicação.

7.2.1 Informação sobre uma classe

A função mais usufruída foi a que permitia obter algumas informações sobre uma classe, função esta que pertence ao menu "Classes". Esta funcionalidade permite uma interação muito simplificada com o utilizador uma vez que a seleção da classe é feita com o rato. Ao selecionar uma classe são dadas várias informações sobre essa classe, como se ilustra na Figura 7.3 com a seleção da classe 99.

A classe selecionada está destacada a vermelho e a sua classe reversa (a mais distante) está representada a verde (neste caso a classe 231). Também está representado no grafo as classes vizinhas da classe selecionada (a azul) as respetivas classes reversas (a amarelo). Juntamente com esta informação, no canto superior direito é apresentado um conjunto de dados sobre a classe selecionada, nomeadamente:

- Representante: 1012103210210232, que é a menor palavra na ordem lexicográfica dessa classe.
- Dimensão da classe: 40, indicando que existem 40 palavras diferentes nesta classe (isto é, obtidas apenas a partir de comutações aplicadas à palavra 1012103210210232).
- Nível: -4. Este número representa o nível da classe, e é caculado através da função nível. Os seus valores variam de -12 a 10 neste grafo.
- Conteúdo: 4552, significa que há quatro geradores 0, cinco geradores 1, cinco geradores 2 e dois geradores 3 em cada palavra desta classe.
- Grau: 4, indica que a classe selecionada possui quatro classes vizinhas. Neste caso, correspondem às classes com os id's 78, 88, 118 e 119.
- Dimensão do cone: 57, indica que há 57 classes no cone da classe selecionada. O cone de uma classe $[a]$ é formado por todos as classes $[b]$ tais que existe um caminho $[a] = [a_0] \overset{L_1 \vee L_2}{\sim} [a_1] \overset{L_1 \vee L_2}{\sim} \cdots \overset{L_1 \vee L_2}{\sim} [a_l] = b$ onde a sequência $N(a_0), N(a_1), \ldots, N(a_l)$, é monótona.
- Mudanças de sinal: 9, significa que há 9 geradores negativos na palavra representante. No caso desta classe, os geradores negativos estão representados a negrito na próxima palavra: 10**12**10**3**2**1**0**2**1**0232**
- Geradores positivos: [1,1,2] indica os índices dos geradores positivos, destacados a negrito na seguinte palavra: **1**012**1**03**2**10210232.

À direita destas informações surge ainda uma tabela que representa parte do diagrama em linha da representante da classe seleccionada.

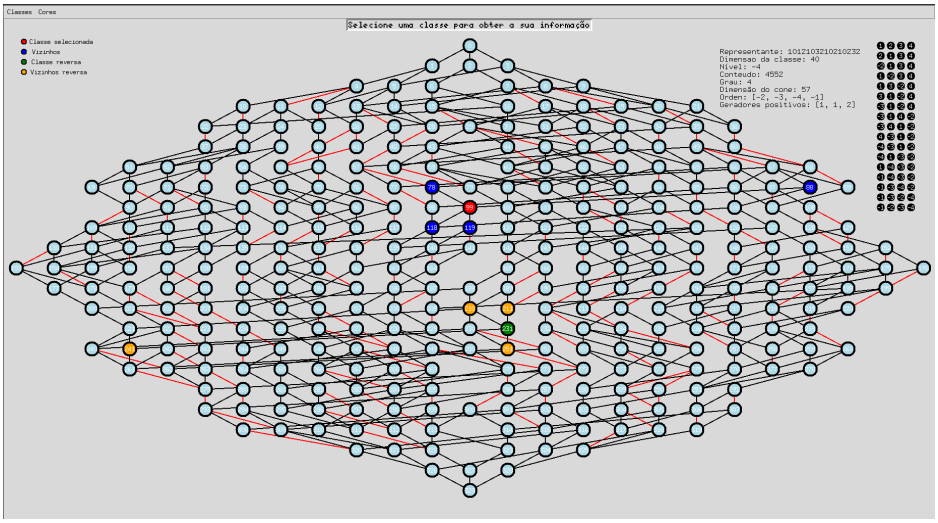

Figura 7.3 Informação sobre um nó.

7.2.2 Caminho mais curto

No grafo $C(\mathrm{w}_0^{B_n})$ o conceito de caminho mais curto entre dois nós está intrisecamente ligado às noções de excentricidade, diâmetro e raio definidas no primeiro capítulo, pelo que foi importante implementar um algortimo (neste caso foi implementado o algoritmo de Dijkstra [10]) que permitisse determinar um caminho de comprimento minimal entre dois nós. Deste modo, foi construída uma função que permite selecionar, com o rato, duas classes e de seguida apresentar um dos caminhos mais curtos que une as classes selecionadas. Na Figura 7.4 foram selecionadas as classes 74 e 253 (a azul) e foi apresentado um caminho entre essas classes sendo que cada classe pertencente ao caminho encontra-se destacada a cor vermelha.

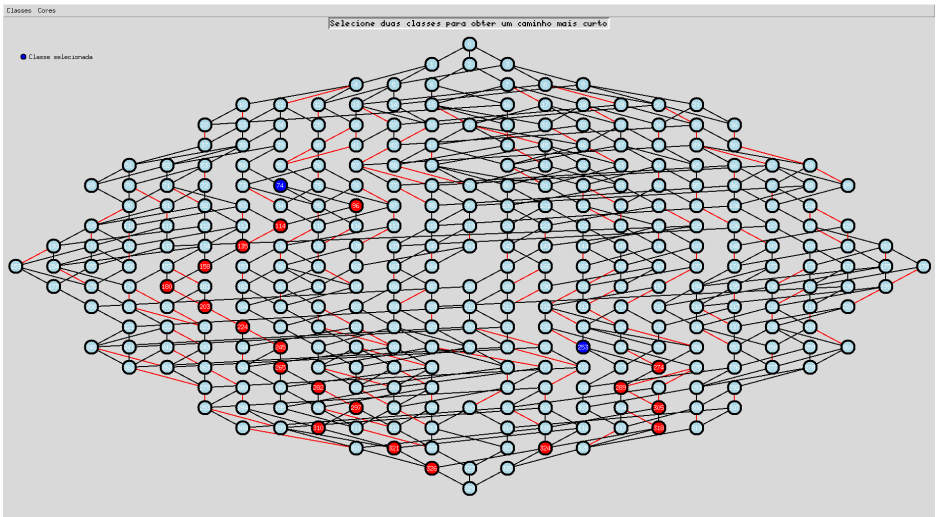

Figura 7.4 Caminho mais curto.

Esta função motivou o cálculo do diâmetro do grafo e permitiu conjeturar que a excentricidade de cada classe é a distância à sua classe reversa, algo que foi provado no Capítulo 5.

7.2.3 Procurar palavra

As classes de comutação do grafo $C(\mathrm{w}_0^{B_n})$ são classes de equivalência, pelo que em geral possuem mais do que uma palavra. Frequentemente, quando se analisa propriedades deste grafo, é necessário testar se certas palavras são reduzidas, ou seja, se existe alguma classe que as contêm. Este processo é trabalhoso e muito suscetível a erros se for feito manualmente, pelo que foi implementado uma função que o permite fazê-lo de maneira rápida. Esta função designa-se por "Procurar palavra" e permite que o utilizador insira uma palavra qualquer no alfabeto $[0,n]$ sendo depois destacada a classe da palavra introduzida, caso ela seja uma palavra reduzida. Como se pode ver na Figura 7.5, há um retângulo abaixo dos menus onde o utilizador introduz a palavra que quer procurar, selecionando de seguida o botão "Procurar". Se essa palavra pertencer a alguma classe é colorido a vermelho a tal classe, caso contrário nenhuma classe é salientada.

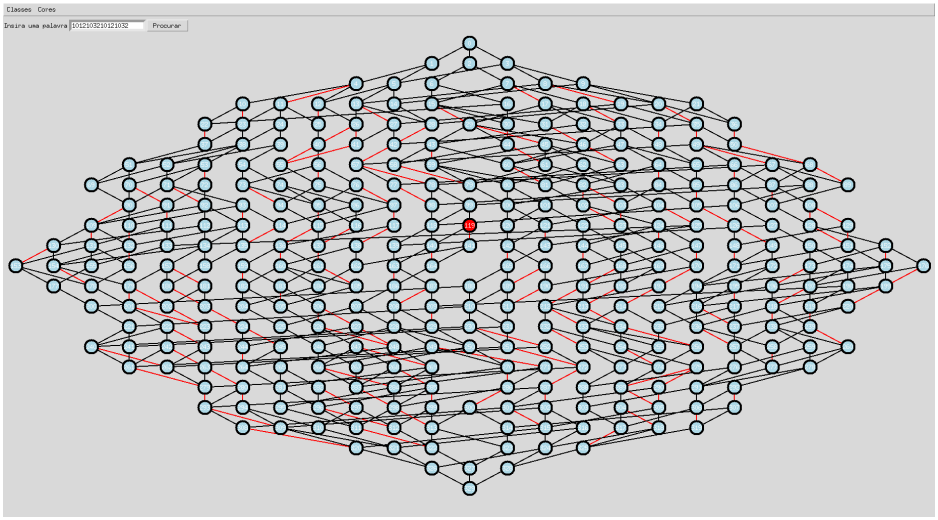

Figura 7.5 Procurar palavra.

7.2.4 Menú "Cores"

Este menú tem várias funções que permitem colorir as classes do grafo de acordo com uma determinada característica. A título de exemplo, salientamos:

- **Colorir por nível**: A função nível foi muito importante pois permitiu calcular o diâmetro do grafo $C(\mathrm{w}_0^{B_n})$ e recuperar algumas propriedades já conhecidas como a sua conexidade e o ser bipartido. Na Figura 7.6 está representado o grafo colorido pela função nível. Note-se que cada classe só está ligada a classes com uma variação unitária da função nível. A descoberta desta propriedade foi essencial para melhorar o desempenho dos algoritmos usados para calcular as classes de comutação.

- **Colorir por dimensão**: Como já foi mostrado na Tabela 7.2, o número de palavras na classe $[w_0]$ aumenta muito rápido com o valor de n. De facto, a classe $[w_0]$ nem sequer é a que possui mais palavras, como se pode observar na Figura 7.7. Esta é mais uma das funcionalidades do

menú “Cores” que permite colorir as classes de acordo com a sua dimensão. Na Figura 7.7 também é possível identificar os átomos (classes 36 e 294).

- **Colorir por número de relações longas**: Esta funcionalidade permite colorir o grafo de acordo com o número de classes vizinhas que cada uma classe possui. Na Figura 7.8 está o grafo colorido de acordo com este critério.

- **Colorir por número de geradores positivos**: Com esta funcionalidade é colorido o grafo de acordo com o número de geradores positivos da representante da classe. Na Figura 7.9 está o grafo colorido de acordo com este critério

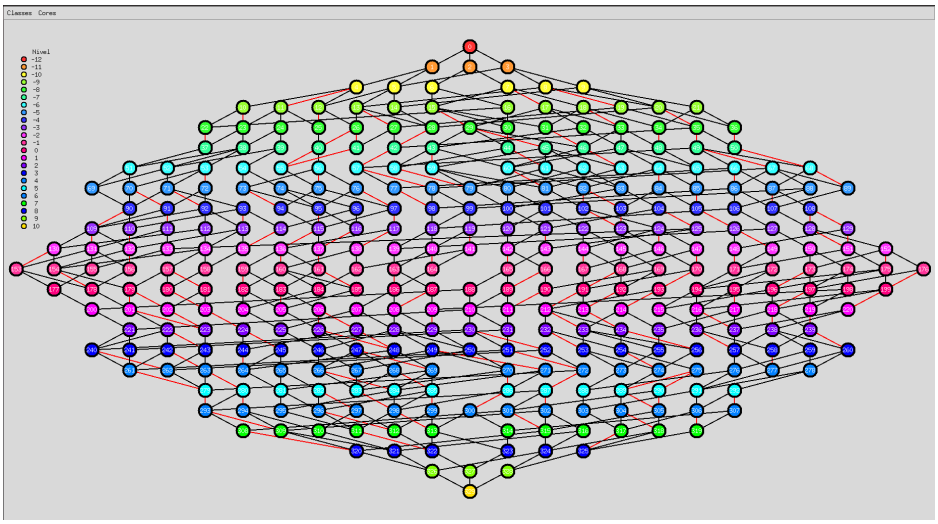

Figura 7.6 Colorir por nível.

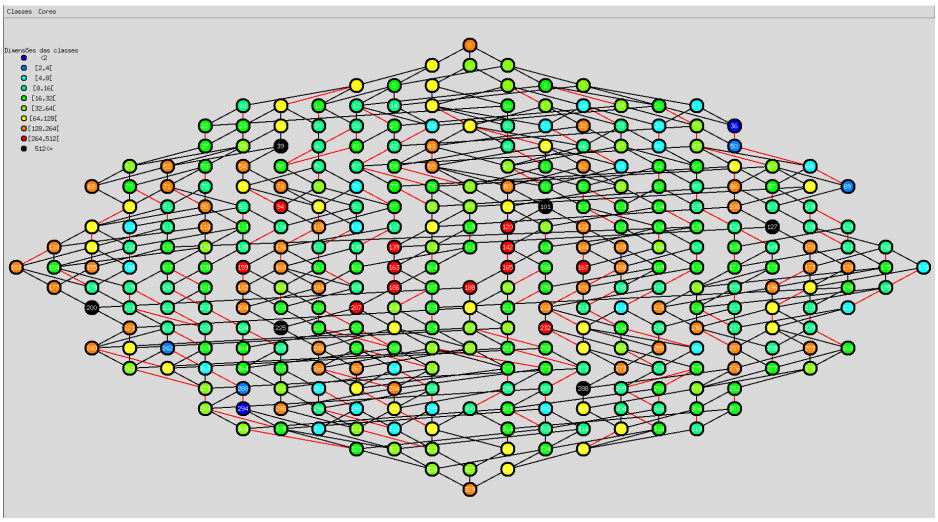

Figura 7.7 Colorir por dimensão.

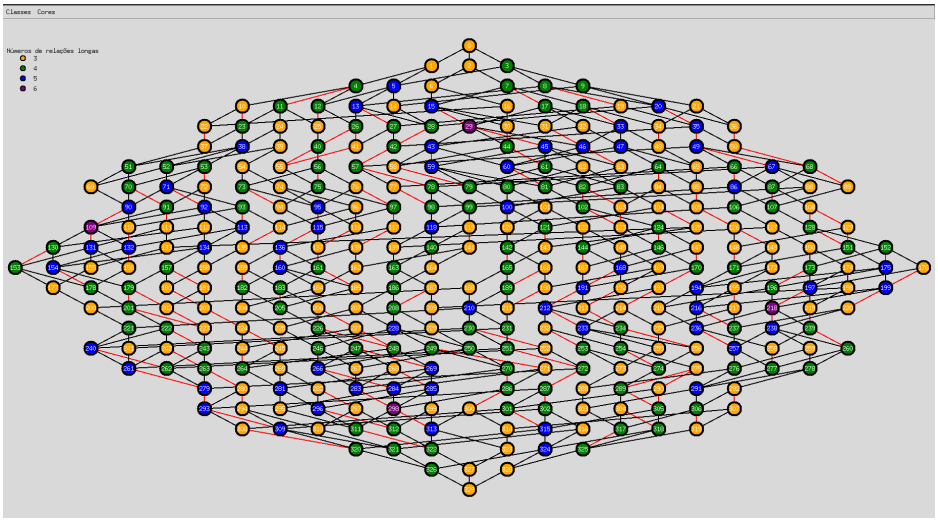

Figura 7.8 Colorir por número de relações longas.

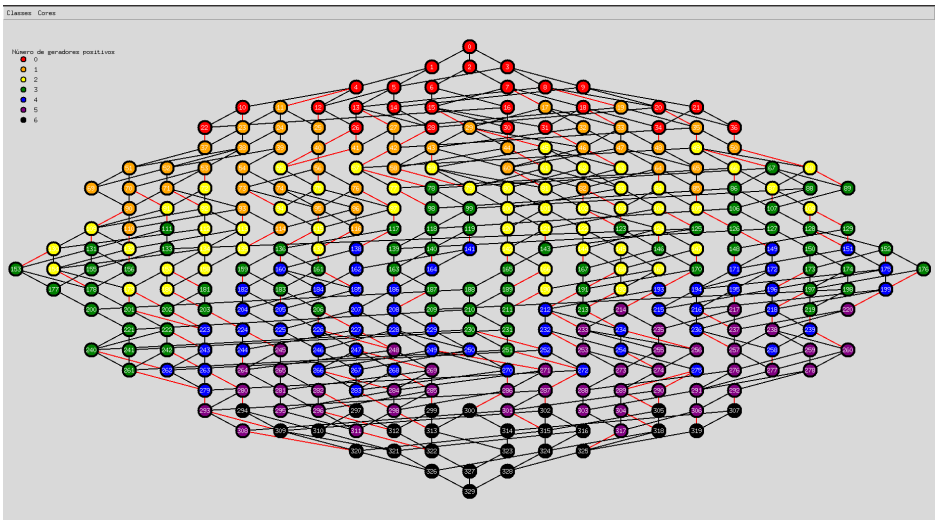

Figura 7.9 Colorir por número de geradores positivos.

Bibliografia

[1] Assaf, S. and Schilling, A. (2018). A demazure crystal construction for schubert polynomials. *Algebraic Combinatorics*, 1(2):225–247.

[2] Bergeron, N., Ceballos, C., and Labbé, J. (2015). Fan realizations of type *A* subword complexes and multi-associahedra of rank 3. *Discrete & Computational Geometry*, 54:195–231.

[3] Billey, S., Hamaker, Z., Roberts, A., and Young, B. (2014). Coxeter-knuth graphs and a signed little map for type b reduced words. *Electronic Journal of Combinatorics*, 21.

[4] Billey, S., Jockusch, W., and Stanley, R. (1993). Some combinatorial properties of schubert polynomials. *Journal of Algebraic Combinatorics*, 2:345–374.

[5] Björner, A. and Brenti, F. (2005). *Combinatorics of Coxeter Groups*. Springer.

[6] Chartrand, G., Lesniak, L., and Zhang, P. (2005). *Graphs & digraphs*. Chapman& Hall/CRC.

[7] Cho, Y., Kim, J., and Lee, E. (2020). Enumeration of gelfand-cetlin type reduced words. *arXiv:2009.06906*.

[8] Daly, D. (2013). Reduced decompositions with one repetition and permutation pattern avoidance. *Graphs and Combinatorics*, 29:173–185.

[9] Denoncourt, H., Ernst, D., and Story, D. (2016). On the number of commutation classes of the longest element in the symmetric group. *arXiv:1602.08328*.

[10] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

[11] Elnitsky, S. (1997). Rhombic tilings of polygons and classes of reduced words in coxeter groups. *Journal of Combinatorial Theory Series A*, 77(2):193–221.

[12] Geck, M. and Pfeiffer, G. (2000). Characters of finite coxeter groups and iwahori-hecke algebras. *Bulletin of the London Mathematical Society*, 33:758–768.

[13] Grove, L. and Benson, C. (1971). *Finite Refletion Groups*. Springer.

[14] Gutierres, G., Mamede, R., and Santos, J. (2020). Commutation classes of the reduced words for the longest element of $\mathfrak{S}_n$. *The Electronic Journal of Combinatorics*, 27(2).

[15] Gutierres, G., Mamede, R., and Santos, J. (2021). Diameter of the commutation graph of a permutation. *DMUC preprint*, 21-12.

[16] Kraskiewcz, W. (1989). Reduced decompositions in hyperoctahedral groups.

[17] Mamede, R., Santos, J., and Soares, D. (2021). The commutation graph for the longest signed permutation. *DMUC pre-print*, 21-22.

[18] Morse, J. and Schilling, A. (2016). Crystal approach to affine schubert calculus. *International Mathematics Research Notices*, 2016:2239–2294.

[19] Reiner, V. and Roichman, Y. (2012). Diameter of graphs of reduced words and galleries. *Transactions of the American Mathematical Society*, 365:2779–2802.

[20] Sloane, N. (1964). The on-line encyclopedia of integer sequences.

[21] Stanley, R. (1984). On the number of reduced decompositions of elements of coxeter groups. *European Journal of Combinatorics*, 5(4):359–372.

[22] Stembridge, J. (1997). Some combinatorial aspects of reduced words in finite coxeter groups. *Transactions of the American Mathematical Society*, 349:1285–1332.

[23] Tenner, B. (2005). Reduced decompositions and permutation patterns. *Journal of Algebraic Combinatorics*, 24:263–284.

[24] Tenner, B. (2015). Reduced word manipulation: patterns and enumeration. *Journal of Algebraic Combinatorics*, 46:189–217.

[25] Tits, J. (1969). Le problème des mots dans les groupes de coxeter. *In: Symposia Mathematica*, pages 175–185.