Autômatos sincronizados e a Conjectura de Černý

Letícia Gindri

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Mestrado em Ciência da Computação

Orientador: Prof. Dr. Arnaldo Mandel

Durante o desenvolvimento deste trabalho a autora recebeu auxílio financeiro da CAPES/CNPq

São Paulo, Junho de 2013

Autômatos sincronizados e a Conjectura de Černý

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 10/07/2013. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Arnaldo Mandel (orientador) - IME-USP
- Prof. Dr. José Augusto Ramos Soares - IME-USP
- Prof. Dr. Orlando Lee - IC-UNICAMP

Agradecimentos

Gostaria de expressar a minha gratidão ao meu orientador, o Prof. Dr. Arnaldo Mandel, pela sua orientação, paciência e estímulo, sem os quais este trabalho não teria sido concluído. E à CAPES e ao CNPQ pelo apoio financeiro.

Agradeço ao meu pai que sempre me foi um grande exemplo e que me deu total apoio desde o momento da minha inscrição no mestrado. Obrigada por ter estado ao meu lado sempre que precisei (mesmo que à distância) e por ter me dado forças para continuar a batalha nos momentos difíceis. Espero que eu possa te deixar orgulhoso de mim!

Agradeço ao meu irmão por ter sido tão grande amigo este tempo todo, sempre se preocupando e sempre ligando para saber como eu estava. A tua presença é muito importante na minha vida. Agradeço também a minha mãe e a minha família por todo apoio e carinho.

Agradeço a todos os amigos e colegas do IME que de uma forma ou de outra fizeram parte desta trajetória, me ajudaram e me apoiaram. Agradeço muito em especial aos dois grandes amigos que o IME trouxe pra minha vida: Guilherme e Santiago. Ao Rafael por ter sido a única pessoa capaz de conseguir me acalmar nos momentos próximos à defesa. Aos outros amigos queridos que São Paulo me trouxe: Cássio, Grazi, Júnior, Michel, Patrícia e Suelen.

Aos amigos de Aracaju que mesmo de longe sempre me apoiaram: Ana Paula, Ana Luiza, Betão, Biazinha, Catuxe, Daniela, Danielle, Danillo, Hirlaine, Jejé, Lipão, Loli, Marco Adolfo, Rodrigo, Tekinha e Tonhão.

Muito obrigada por tudo!

Resumo

GINDRI, L. **Autômatos Sincronizados e a Conjectura de Černý**. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013.

Černý, em 1964, conjecturou que um autômato sincronizado com n estados possui uma palavra sincronizadora mínima de tamanho no máximo $(n-1)^2$. Esta conjectura permanece em aberto. Neste trabalho são apresentados algoritmos para obter palavras sincronizadoras e é feito um experimento comparativo entre os resultados obtidos por estes algoritmos em relação a algumas séries infinitas de autômatos. Por fim, é feito um breve histórico sobre os resultados parciais obtidos até a presente data e alguns destes trabalhos são apresentados em mais detalhes.

Palavras-chave: autômato sincronizado, conjectura de Černý, palavra sincronizadora, palavra sincronizadora mínima, algoritmo, Coloração de Estradas, sincronização, grafo, Lema das Probabilidades.

Abstract

GINDRI, L. **Synchronizing Automata and the Černý Conjecture**. Dissertation (Master) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013.

Černý, on 1964, conjectured that a synchronizing automata with n states has a synchronizing word of size at most $(n-1)^2$. The conjecture remains open. We show some algorithms for obtaining synchronizing sequences and a comparative experiment between these algorithms with respect to some infinite series of automata. Furthermore, we briefly survey some of the partial results obtained until the present day.

Keywords: synchronizing automata, Černý conjecture, reset words, shortest reset word, algorithm, Road Coloring, graph, Averaging Lemma.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Teoria dos Autômatos estuda as definições, propriedades, classificações e os relacionamentos dos modelos matemáticos de computação. Esses modelos matemáticos são chamados de máquinas de estado finitas ou autômatos finitos.

Informalmente, um autômato finito é formado por um conjunto de estados e por regras que determinam transições por esses estados baseadas no símbolo de entrada. Formalmente, um autômato finito determinístico é uma tupla (Q, Σ, δ), onde Q é um conjunto finito de estados, Σ é um alfabeto e $\delta : Q \times \Sigma \to Q$ é uma função de transição que descreve a ação de uma letra sobre um estado[1] [Sip07, HU79].

1.1 Notação, conceitos e definições

O **grafo** G_A de um autômato $A = (Q, \Sigma, \delta)$ é o digrafo cujos vértices são os estados de A e para cada par $(q, \sigma) \in Q \times \Sigma$ existe uma aresta dirigida do vértice q ao vértice $\delta(q, \sigma)$ em G_A. A Figura 1.1 mostra um exemplo do grafo de um autômato.

Um autômato $A = (Q, \Sigma, \delta)$ é dito **fortemente conexo** se o grafo G_A deste autômato for fortemente conexo. Ou seja, se para cada par de estados $q_i, q_j \in Q$ existirem palavras v, w tais que $\delta(q_i, v) = q_j$ e $\delta(q_j, w) = q_i$ [BJG02].

A definição da função de transição será estendida para $w \in \Sigma^*$, ou seja, $\delta : Q \times \Sigma^* \to Q$ descreve a ação de uma palavra sobre um estado ou um conjunto de estados. Dessa forma, para $w \in \Sigma^*$ e $P \subseteq Q$ definimos:

1. $P \cdot w = \{\delta(p, w) \mid p \in P\}$.[2]
2. $P \cdot w^{-1} = \{q \mid \delta(q, w) \in P\}$.

A cardinalidade desses conjuntos é representada por $|P \cdot w|$ e $|P \cdot w^{-1}|$.

[1] Na Teoria de Autômatos essa definição é atribuída aos semi-autômatos, uma vez que não estão especificados estados iniciais e finais. Contudo, como não trabalharemos com outros tipos de autômatos, optamos por utilizar o termo mais simples.

[2] O símbolo "$\cdot$" pode ser omitido sem prejuízo para o significado da representação: $P\, w = \{\delta(p, w) \mid p \in P\}$.

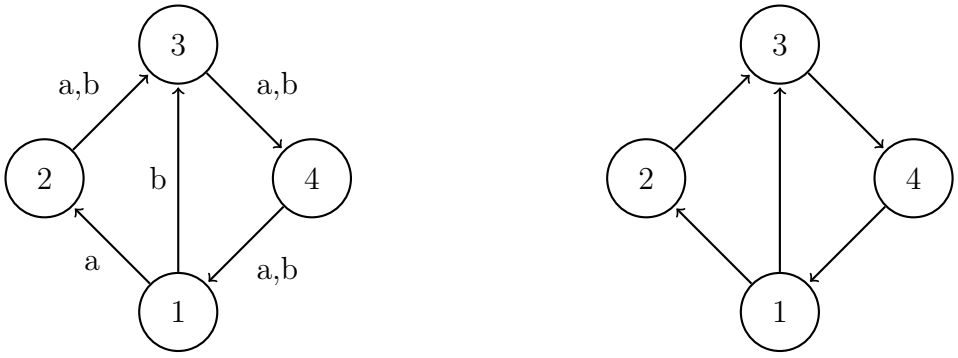

Figura 1.1: *Um autômato (esquerda) e seu respectivo grafo G_A (direita).*

Um autômato A é dito **sincronizado** se existe $w \in \Sigma^*$ tal que o seu mapeamento a partir de qualquer estado em Q leva a um único estado. Ou seja, $\delta(q_i, w) = \delta(q_j, w)$ para todo $q_i, q_j \in Q$. Dessa forma, $|Q \cdot w| = 1$. Tal palavra é chamada **sincronizadora**. A concatenação de duas palavras v e w pode ser representada por $v \cdot w$ ou apenas vw. Observe que se w é uma palavra sincronizadora, vw e wv também são sincronizadoras.

O autômato da Figura 1.1 é sincronizado e $w = baabaab$ é um exemplo de palavra que o sincroniza. Vamos representar por $|w|$ o tamanho de uma palavra w. Neste caso, $|w| = 7$.

No decorrer deste trabalho, quando nos referirmos a um autômato A assumiremos que ele é finito e determinístico, possui n estados, alfabeto Σ, seu conjunto de estados é Q e sua função de transição é δ. Vamos denotar a palavra vazia por λ.

Em relação a autômatos sincronizados, existem dois problemas que figuram entre os mais importantes na Teoria dos Autômatos: o Problema da Coloração de Estradas, apresentado com mais detalhes na Seção 1.3, e a Conjectura de Černý, apresentada no Capítulo 2.

1.2 O primeiro autômato sincronizado

Em seu livro *An introduction to cybernetics* [Ash56], Ashby apresenta o Enigma dos Ruídos Fantasmagóricos. Este enigma é considerado como o primeiro registro de um autômato sincronizado. O problema trata de dois fantasmas que assombram uma casa: um do canto irreverente e outro da risada sarcástica.

De acordo com o enigma, o morador da casa descobriu que o comportamento destes fantasmas segue algumas regras obscuras, porém infalíveis, e que é afetado quando se toca piano ou acende-se um incenso.

O fantasma que canta permanece no estado em que se encontrava anteriormente (canto ou silêncio), exceto se o piano for tocado na ausência da risada sarcástica. Neste caso, irá mudar para o estado oposto. O fantasma da risada sarcástica na ocorrência de queima de incenso irá soar, se o fantasma do canto irreverente estava soando no momento anterior, e permanecer em silêncio, se este estava em silêncio. Se não ocorreu queima de incenso, a risada sarcástica vai fazer o oposto do que o canto irreverente estava fazendo.

O rótulo a, na Figura 1.2, representa a ausência de ações, b representa tocar o piano, c

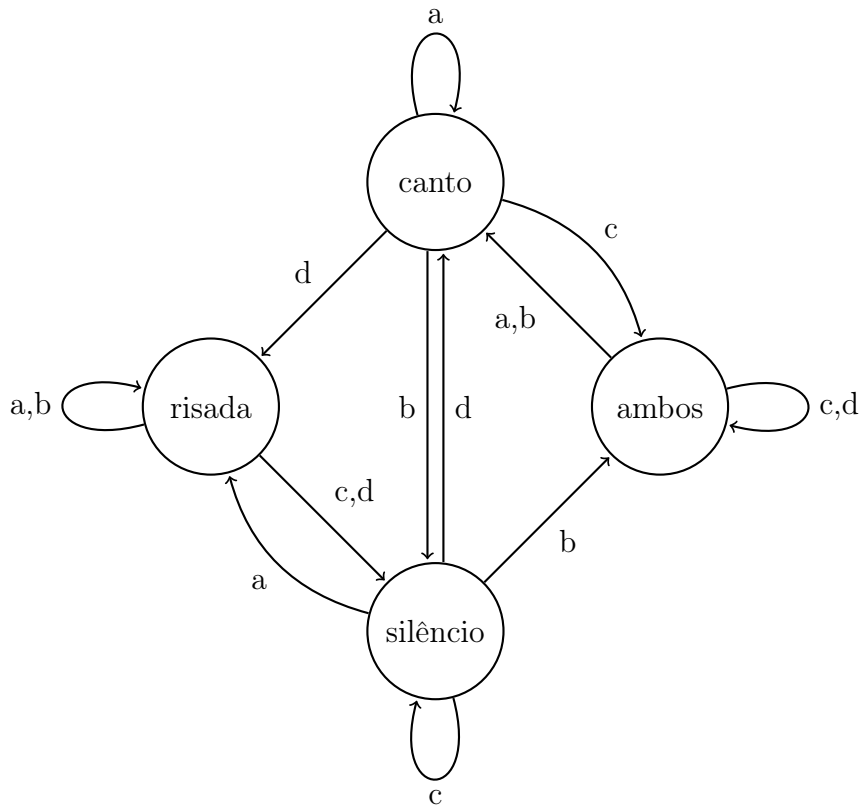

Figura 1.2: *Autômato que representa o enigma dos ruídos fantasmagóricos de Ashby.*

representa acender o incenso e *d* ambos (acender o incenso e tocar o piano). O problema é realizar uma sequência de ações que leve o autômato para o estado "silêncio", ou seja, garantir o silêncio na casa. Observe que as sequências de ações *abc* e *adc* garantem a sincronização do autômato no estado "silêncio".

1.3 O Problema da Coloração de Estradas

O **Problema da Coloração de Estradas** (PCE), proposto na década de 70 por Adler, Goodwyn e Weiss, tornou-se objeto de pesquisa de inúmeros matemáticos [AGW77]. Alguns resultados parciais foram alcançados durante este tempo, contudo, esta conjectura permaneceu em aberto até 2007 quando foi provada por Avraham Trahtman [Tra07].

O PCE consiste em encontrar uma coloração sincronizadora em um grafo dirigido com grau de saída igual em todos os vértices. Uma coloração sincronizadora para um grafo é uma coloração de arestas para a qual o autômato correspondente[3] é sincronizado.

Tal coloração é possível se e somente se:

- A condensação acíclica[4] do grafo possui um único sumidouro [5] e a componente forte-

[3]O autômato correspondente a um grafo com arestas coloridas é o autômato obtido se olharmos para cada cor como um símbolo do alfabeto.

[4]A condensação acíclica de um grafo é a condensação de todos os seus componentes fortemente conexos tornando o grafo acíclico.

[5]Vértice com grau de saída zero.

mente conexa correspondente a este sumidouro possui uma coloração sincronizadora. Dessa forma, é suficiente considerar o problema para grafos fortemente conexos.

- Os tamanhos dos ciclos do grafo são primos entre si (ou seja, o grafo é aperiódico).

Se existir $1 < k \in \mathbb{N}$ que divide o tamanho de todos os ciclos, então é possível particionar o conjunto de vértices em subconjuntos $V_0, V_1, \ldots, V_{k-1}$, tal que cada aresta que parte de V_i chega em $V_{(i+1) \ mod \ k}$ e este conjunto de vértices claramente não é sincronizável. Adler, Goodwyn e Weiss conjecturaram que essas condições necessárias eram também suficientes para a existência de uma coloração sincronizadora [AGW77]:

Problema da Coloração de Estradas. *Todo grafo finito dirigido, fortemente conexo, com grau de saída constante para todos os vértices e com o maior divisor comum do tamanho de todos os ciclos igual a um pode ser colorido tornando-se um autômato finito determinístico sincronizado.*

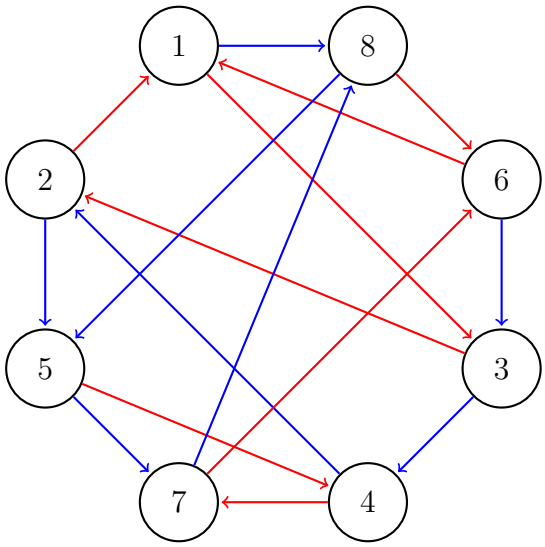

Figura 1.3: *Um grafo com coloração sincronizadora.*

Observe que o grafo da Figura 1.3 é fortemente conexo, possui grau de saída constante e todos os seus ciclos possuem tamanhos primos entre si. Dessa forma, sabemos que existe uma coloração sincronizadora para este grafo. A sequência $aavaavaav$ (a representa a cor azul e v a vermelha) sincroniza o autômato correspondente ao grafo e portanto a coloração ilustrada é sincronizadora.

1.4 Organização

Este trabalho está organizado da seguinte maneira: no Capítulo 2 é apresentada a Conjectura de Černý; no Capítulo 3 são apresentados algoritmos para encontrar palavras sincronizadoras, além de experimentos comparativos entre estes algoritmos; e por fim, no Capítulo 4 é feita uma exposição do que já foi provado em relação à conjectura e é apresentado um lema através do qual é possível provar resultados parciais em relação a Conjectura de Černý.

Capítulo 2

A Conjectura de Černý

A Conjectura de Černý configura hoje um dos problemas mais antigos em aberto na Teoria dos Autômatos. Černý primeiro observou que o tamanho da menor palavra sincronizadora poderia ser limitado inferiormente por $(n-1)^2$ e superiormente por $2^n - n - 1$ [Cer64]. Alguns anos mais tarde, durante a *Bratislava Cybernetics Conference*, a conjectura foi então formulada da maneira conhecida atualmente.

Conjectura de Černý ([Cer64])**.** *Um autômato sincronizado contendo n estados possui uma palavra sincronizadora de tamanho no máximo* $(n-1)^2$*.*

Teorema 2.1. *Se a Conjectura de Černý for válida para autômatos finitos determinísticos fortemente conexos, é também válida para todos os autômatos finitos determinísticos.*

Prova: ([Pin78a, Wal08]) Seja $A = (Q, \Sigma, \delta)$ um autômato finito determinístico sincronizável com $|Q| = n$. Agora suponha que todos os autômatos finitos determinísticos fortemente conexos podem ser sincronizados de acordo com a conjectura.
Vamos dividir o autômato A em dois componentes: o componente fortemente conexo maximal X cujo conjunto de vértices é um sorvedouro (ou seja, não existem arestas saindo de X) e que necessariamente contém um estado sincronizador; e o componente $Y = Q \setminus X$, com $|Y| = m$. Observe que $m \leq n-1$, visto que o componente X contém pelo menos um estado (o estado sincronizador) e $m \geq 1$, pois caso contrário o componente X seria equivalente a Q e o autômato seria fortemente conexo e não teríamos o que mostrar, pois para este caso estamos supondo que a conjectura é válida.
Como X é um sorvedouro, existem caminhos que mapeiam os elementos de Y em X, mas não existem caminhos mapeando qualquer estado de X em Y (caso contrário, X não seria um sorvedouro). Dessa forma, para cada estado $y \in Y$ teremos uma palavra de tamanho no máximo m que leva y para algum estado de X.
Se o tamanho da palavra for maior que m, algum estado de Y está sendo atravessado mais de uma vez e basta eliminar esta repetição. Observe que se $|Q| = 2$, com $|X| = 1$ e $|Y| = 1$, como X é um sorvedouro, então, com uma palavra de tamanho no máximo 1 é possível mapear Y em X. Além disso, como Y contém no máximo m estados, então com no máximo

m palavras é possível mapear todos os estados de Y em X da seguinte forma: escolha um estado i de Y e aplique a palavra que mapeia este estado em X a todos os estados de Y e faça Y_i conter os estados obtidos após esta ação. Repita o processo até que não restem estados fora de X. Ao concatenar estas palavras é possível obter uma palavra u de tamanho no máximo m^2 que mapeia Y em X. Este processo de concatenar palavras que reduzem o número de estados é chamado **Método de Fusão** e pode ser visto em mais detalhes na Seção 3.1.
O componente fortemente conexo X, que contém no máximo $n - m$ estados, tem, pela suposição inicial, uma palavra sincronizadora $|v| \leq ((n - m) - 1)^2$. Dessa forma, o autômato A pode ser sincronizado por uma palavra w resultante da concatenação das palavras u e v, de tamanho máximo:

$$\begin{aligned} |w| &= m^2 + ((n-m)-1)^2 \\ &= m^2 + n^2 - 2mn + m^2 - 2(n-m) + 1 \\ &\leq 2(n^2 - 2n + 1) + n^2 - 2(n^2 - n) - 2n + 2n - 2 + 1. \text{ (Pois } m \leq n-1.) \\ &\leq (n-1)^2. \end{aligned}$$

□

Dessa forma, é suficiente provar a conjectura para autômatos fortemente conexos.
Pin formulou uma generalização da conjectura de Černý:

Conjectura 2.1 ([Pin83])**.** *Seja A um autômato sincronizado com n estados e seja $0 \leq k \leq n-1$. Se existe uma palavra w com $|Q \cdot w| \leq n - k$ em A, então existe v, com $|v| \leq k^2$, tal que $|Q \cdot v| \leq n - k$.*

Observe que obtemos a conjectura original fazendo $k = n - 1$. Pin provou a conjectura para $k \in \{0, 1, 2, 3\}$. Contudo, o autômato da Figura 2.1, com 6 estados, é um contra-exemplo para esta generalização quando $k = 4$, uma vez que a menor palavra w com $|Q \cdot w| \leq n - k = 2$ é $w = baababababaabbbabaab$ com tamanho 17 e não 16 como propunha a conjectura [Kar01]. Isto pode ser verificado usando a construção apresentada na Seção 3.2.

Volkov propôs um novo problema chamado Problema Híbrido Černý-Coloração de Estradas que consiste em, dado um grafo dirigido que admite uma coloração sincronizadora, determinar uma coloração com palavra sincronizadora de tamanho mínimo [Vol08].

Durante quase meio século, diversos autores têm trabalhado na conjectura de Černý utilizando técnicas de álgebra linear, teoria dos grupos e combinatória. Atualmente o melhor limite superior alcançado para a conjectura de Černý é de ordem cúbica: $n(7n^2 + 6n - 16)/48$ [Tra11]. Palavras sincronizadoras com tamanho limitado por uma expressão cúbica em função de n podem ser encontradas como saída do algoritmo apresentado na Seção 3.3.

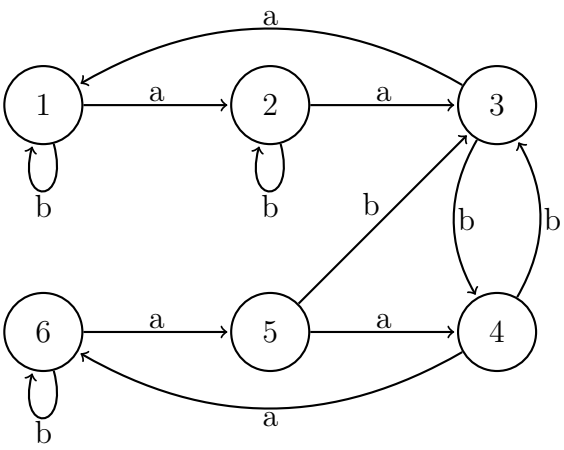

Figura 2.1: *O contra-exemplo para a generalização da conjectura apresentado por Kari.*

2.1 A Série de Černý

Černý apresentou uma série infinita de autômatos sincronizados: a Série de Černý [Cer64, Vol08]. Representada por C_n, onde n é o número de estados do autômato, esta série possui menor palavra sincronizadora com tamanho precisamente $(n-1)^2$. Os autômatos pertencentes à série têm n estados, sendo $Q = \{0, 1..., n-1\}$, alfabeto $\{a, b\}$ e função de transição definida por:

i. $\delta(0, b) = 1$ e $\delta(q, b) = q$ para $q \neq 0$;

ii. $\delta(q, a) = q + 1 \bmod n$, para todo $q \in Q$.

A Figura 2.2 apresenta a Série de Černý e a Figura 2.3 apresenta o autômato de 4 estados pertencente a esta série: o C_4. Uma palavra sincronizadora mínima para este autômato é *baaabaaab*.

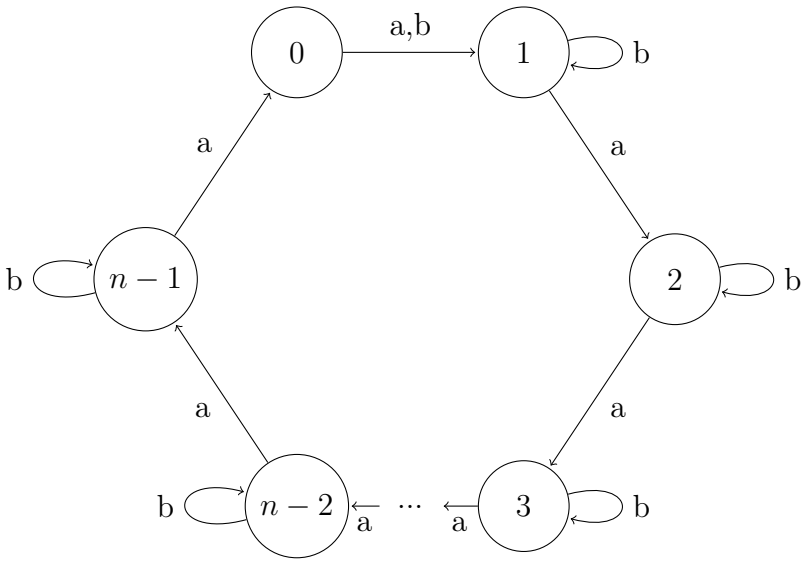

Figura 2.2: *A Série de Černý.*

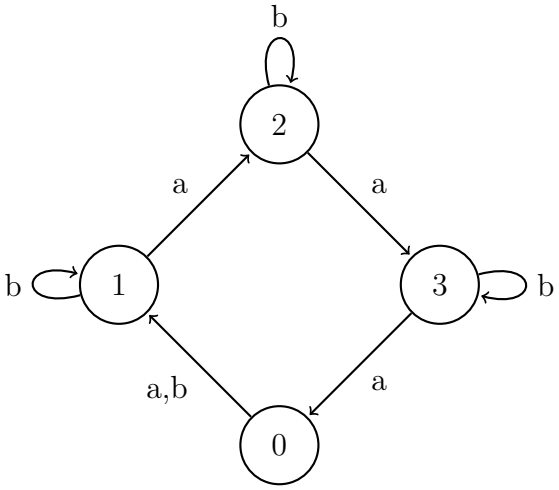

Figura 2.3: *O autômato C_4.*

Černý provou que a palavra $(ba^{n-1})^{n-2}b$ é uma palavra mínima para esta série de autômatos [Cer64], não necessariamente única. Esta é a única série infinita conhecida até o momento na qual o tamanho da menor palavra sincronizadora é precisamente o tamanho estabelecido pela conjectura. Autômatos com essa característica são chamados de autômatos extremos.

Teorema 2.2. *Todo autômato C_n tem palavra sincronizadora mínima de tamanho $(n-1)^2$.*

Prova: ([Cer64, AGV10]) A palavra $(ba^{n-1})^{n-2}b$ é sincronizadora para o autômato e o seu tamanho é $(1+(n-1))(n-2)+1=(n-1)^2$.

Primeiro observe dois fatos evidentes sobre a série C_n:

1. A letra a atua como uma permutação cíclica sobre o conjunto de estados;

2. A letra b atua como função identidade sobre todos os estados, exceto o 0.

Dessa forma, se w é uma palavra sincronizadora mínima, então w termina com b. Uma vez que a não reduz a cardinalidade do conjunto, se w terminasse por a, então, através da remoção de todos os a's após o último b ainda teríamos uma palavra sincronizadora e w não seria mínima. Sendo assim, podemos reescrever w como $w = u\ b$, com $u \in \{a,b\}^*$ e, portanto, $Q \cdot w = Q \cdot u\ b = \{0,1\} \cdot b = \{1\}$.

Cada ocorrência de b, exceto a última, deve ser seguida por uma ocorrência de a. Observe que nenhum estado atinge 0 através da leitura de b. Além disso, esta letra atua como função identidade em todos os estados exceto o 0 e este estado é necessário para a redução do conjunto.

Dessa forma, se fizermos $c = ba$ é possível criar um novo autômato T_n induzido pela ação do alfabeto $\{a,c\}$ em C_n e transformar a palavra $u \in \{a,b\}^*$ em uma palavra $v \in \{a,c\}^*$. Este novo autômato também é sincronizado, pois a ação de u em C_n é equivalente à ação de v em T_n e portanto vc o sincroniza.

Seja i o estado alcançado através da leitura de vc em T_n. Então existe um caminho de i para o próprio i rotulado por esta palavra. Uma vez que vc é sincronizadora, para todo

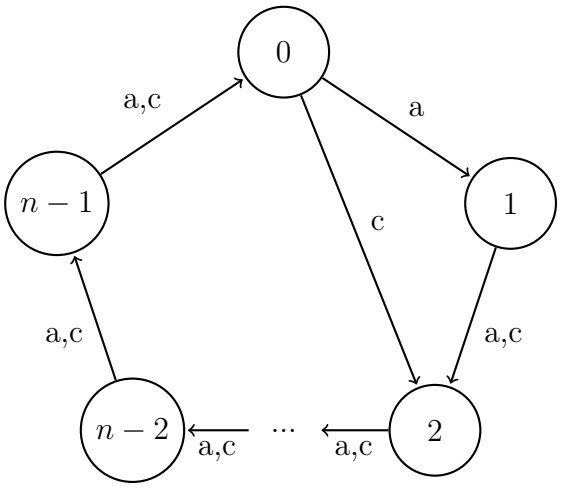

Figura 2.4: *O autômato T_n.*

$v' \in \{a, c\}^*$, $v'vc$ também é uma palavra sincronizadora. Dessa forma, para todo $t \geq |vc|$ existe um ciclo de tamanho t em T_n.

Como este autômato possui apenas dois ciclos com tamanhos n e $n-1$, o ciclo com tamanho t será uma combinação destes ciclos. Como n e $n-1$ são primos entre si, é possível limitar inferiormente o tamanho dessa combinação linear não-negativa.

Ferdinand Georg Frobenius propôs o seguinte problema chamado Problema Diofantino de Frobenius (**PDF**): dados n números inteiros não-negativos co-primos $x_1, x_2, \cdots, x_n$, encontrar o maior número inteiro $\mathrm{g}(x_1, x_2, \cdots, x_n)$ que não é representável como uma combinação inteira não-negativa de $x_1, x_2, \cdots, x_n$. A seguir é apresentado o caso particular do PDF para $n = 2$.

Sejam m, n dois números inteiros não-negativos e co-primos. Então $\mathrm{g}(m, n) = mn - m - n$. Como $\mathrm{mdc}(m, n) = 1$, então qualquer inteiro k é representável como $k = am + bn$ com $a, b \in \mathbb{Z}$. Note que k pode ser representado de muitas formas diferentes, mas a representação torna-se única quando $0 \leq a < n$. Neste caso, k é representável se $b \geq 0$ e não é representável se $b < 0$. Portanto, o maior valor inteiro não representável como uma combinação inteira não-negativa de m, n é obtido quando $a = n-1$ e $b = -1$. Então $\mathrm{g}(m, n) = (n-1)\, m + (-1)\, n = mn - m - n$ [Alf05].

Dessa forma, o limitante inferior no tamanho da combinação linear não-negativa é dado por $t > n\ (n-1) - n - (n-1) \Rightarrow t \geq n\ (n-1) - n - (n-1) + 1 = n^2 - 3n + 2$.

Agora suponha que $|vc| = n^2 - 3n + 2$. Então existe um caminho do estado imediatamente anterior a i até o próprio i com este tamanho. Claramente, a menor distância entre estes dois estados é 1 (uma aresta) e o caminho entre estes dois estados rotulado por vc será formado por esta aresta seguida pelo ciclo com inicio e fim em i. Logo, este ciclo terá tamanho $n^2 - 3n + 1$. Mas, como visto, não existe ciclo deste tamanho composto por ciclos de tamanho n e $n-1$. Portanto, $|vc| = n^2 - 3n + 3$. Como visto no Fato 1, a não altera a cardinalidade do conjunto de estados sobre o qual atua. Além disso, a ação de c reduz a cardinalidade do conjunto em no máximo uma unidade. Então, vc contém pelo menos $n-1$ ocorrências de c. Isto implica

que $|v| \geq n^2 - 3n + 2$ e que v possui pelo menos $n - 2$ ocorrências de c.

Cada ocorrência de c em v equivale a uma ocorrência de ab em u, logo $|u| \geq (n^2 - 3n + 2) + (n - 2) = n^2 - 2n$ e por fim $|w| = |ub| \geq (n^2 - 2n) + 1 = (n - 1)^2$.

□

Esta prova, apresentada por Ananichev, não é a única forma de demonstrar a afirmação. A seguir é apresentada uma prova alternativa (de autoria própria).

Prova: (alternativa) A palavra $b(a^{n-1}b)^{n-2}$ é sincronizadora para o autômato C_n e o seu tamanho é $1 + ((n-1)+1)(n-2) = (n-1)^2$. Agora vamos mostrar que esta palavra tem tamanho mínimo. Seja $Q = \{0, 1, \cdots, n-1\}$ o conjunto de estados do autômato C_n, e para $S \subseteq Q$, seja $\bar{S} = Q \setminus S$. Seja $I(j,k)$ a função Intervalo para $j, k \in \{0, 1, \cdots, n-1\}$, onde k representa o tamanho do intervalo:

$$\begin{cases} I(j,k) = \{j, j+1, j+2, \cdots, j+k-1\} \subseteq \mathbb{Z}_n & \text{se } k \neq 0; \\ I(j,k) = \emptyset & \text{se } k = 0. \end{cases}$$

E seja $I_{max}(S)$ a função Intervalo Máximo definida por:

$$I_{max}(S) = \max\{\ k \mid \exists\ j\ : I(j,k) \subseteq \bar{S}\}.$$

Note que estamos considerando os intervalos em $\bar{S}$. A função I_{max} tem valor mínimo quando $S = Q$, sendo $I_{max}(Q) = 0$, e máximo quando $S = \{q\}$, sendo $I_{max}(\{q\}) = n - 1$. Para $S = Q$, $\bar{S} = \emptyset$ e portanto não existe intervalo algum fora do conjunto S. Para $S = \{q\}$, $\bar{S} = Q \setminus \{q\}$ e a quantidade de estados em sequência fora de S tem tamanho $n - 1$. Dessa forma, fica evidente que, a partir do conjunto Q, até ser alcançado um estado único são necessárias $n - 1$ aumentações no valor da função I_{max}.

Agora observe dois fatos sobre o C_n:

i. Ação da letra a não altera o valor da função I_{max}, visto que atua como uma permutação cíclica sobre o conjunto de estados.

ii. A ação da letra b aumenta o tamanho do intervalo em no máximo 1. Para isto é necessário que $0 \in S$ e que o maior intervalo atual termine no estado $n - 1$.

Se w é uma palavra mínima para este autômato, então w começa por b. Caso contrário, a aplicação de um ou mais a's levaria a uma permutação sobre os estados, não alterando o valor do intervalo e w não seria mínima. Então w pode ser reescrita como $w = bv$.

Como nenhum estado atinge 0 pela leitura de b, e para uma aumentação deve-se ter $0 \in S$, precisamos de uma subpalavra mínima que garanta, após uma aumentação de intervalo, as condições necessárias para a próxima. Então agora precisamos rotacionar o maior intervalo atual de maneira que seu extremo direito seja $n - 1$ e que se obtenha $0 \in S$. A palavra a^{n-1} é a menor palavra que garante isso. Então, $u = a^{n-1}b$ é uma palavra mínima que se aplicada exatamente após uma aumentação de intervalo, garante outra aumentação.

Uma vez que b proporciona no máximo uma aumentação de intervalo, a partir de Q é necessário ler pelo menos $n-1$ b's para se obter as $n-1$ aumentações. Portanto, o total de b's necessários na palavra sincronizadora mínima é $n-1$. Como $w = b\ v$, então a palavra v, sendo mínima, deve conter $n-2$ b's. Logo, com $v = (a^{n-1}b)^{n-2}$ atingimos o máximo da função I_{max}. Dessa forma, $w = b\ (a^{n-1}b)^{n-2}$ é uma palavra mínima com a qual é possível alcançar o valor máximo de I_{max} e, portanto, sincronizar o autômato. E por fim, $|w| = 1 + ((n-1)+1)(n-2) = 1 + (n(n-2)) = 1 + n^2 - 2n = (n-1)^2$. □

Além desta série, poucos autômatos extremos são conhecidos [Vol08, Tra06]. Trahtman conjecturou que além da série de Černý existem apenas 8 autômatos extremos. As figuras 2.5, 2.6 e 2.7 apresentam estes autômatos.

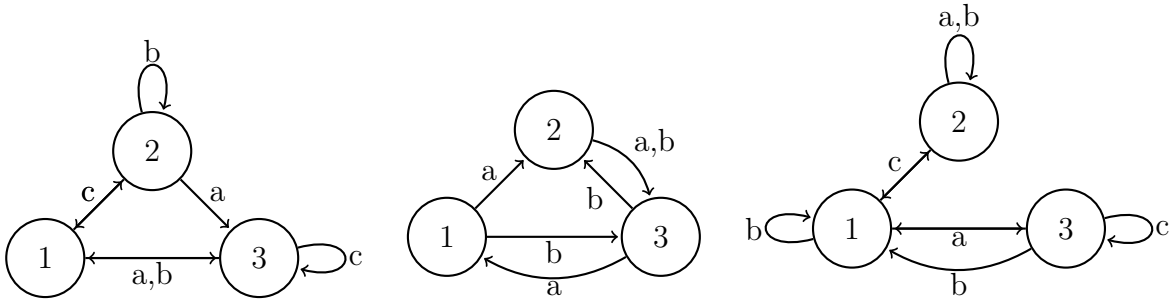

Figura 2.5: *Automatos extremos com 3 estados [Tra06].*

As palavras *acba*, *baab* e *bacb* são, respectivamente, sincronizadoras e mínimas para os autômatos da Figura 2.5.

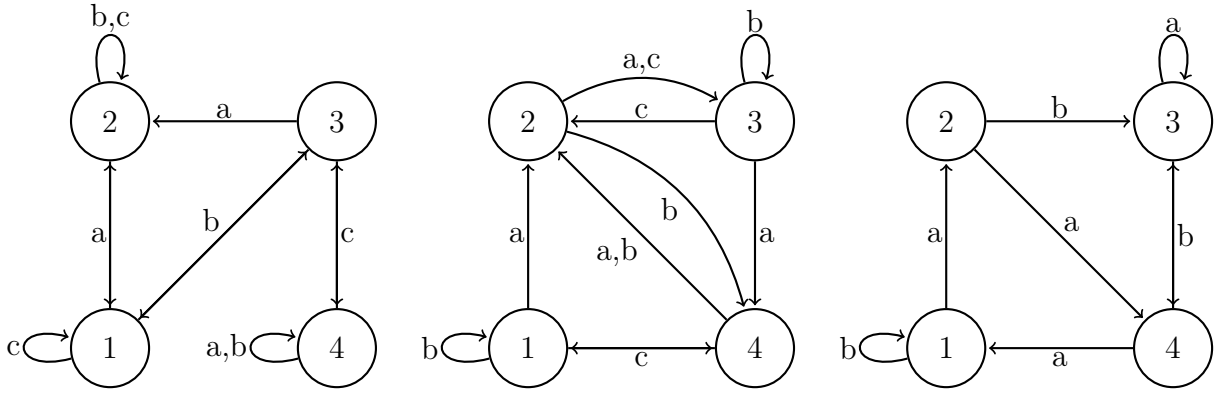

Figura 2.6: *Autômatos extremos com 4 estados [Tra06].*

Para os autômatos da Figura 2.6, as palavras *abcacabca*, *acbaaacba* e *baababaab* (de comprimento 9) são sincronizadoras e mínimas. Por fim, para os autômatos da Figura 2.7, as palavras *abcacacbcaacabca* (de tamanho 16) e *baababababaabbabaabaababaab* (de tamanho 25) são sincronizadoras mínimas.

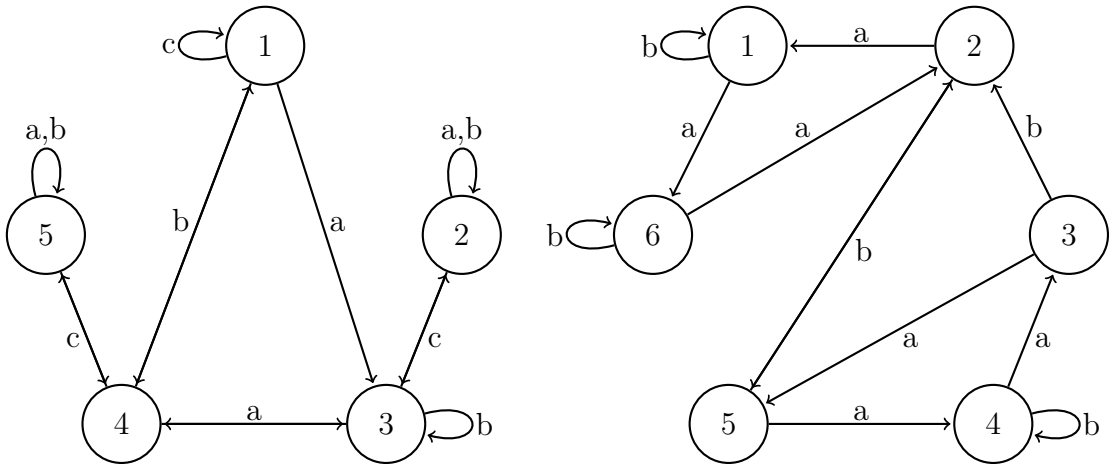

Figura 2.7: *Autômatos extremos com 5 e 6 estados [Tra06].*

2.2 A NP-completude do Problema da Sincronização

Dados um autômato com n estados e um inteiro t, SINC é o problema de decidir se este autômato possui uma palavra sincronizadora com tamanho no máximo t.

Variáveis booleanas podem tomar valores **verdadeiro** (que pode ser representado por 1) ou **falso** (que pode ser representado por 0). Uma fórmula proposicional (ou booleana) na Forma Normal Conjuntiva (FNC) é uma expressão contendo variáveis booleanas e formada por conjunções de disjunções. Além disso, uma fórmula booleana é dita satisfazível se existe uma atribuição de 0's e 1's às variáveis que torna o resultado da fórmula igual a 1. O Problema da Satisfabilidade (SAT) é o problema de testar se uma fórmula booleana é satisfazível [Sip07].

Teorema 2.3 ([Epp90])**.** *O Problema* SINC *é* NP*-completo.*

Prova: A NP-completude do Problema SINC é demonstrada através de uma redução do SAT para o Problema SINC. A entrada é um problema de satisfabilidade na FNC com t variáveis $x_1, x_2, \cdots, x_t$ e n cláusulas. A partir desta entrada é construído um autômato com $n\ (t+1)+1$ estados que podem ser de três tipos:

- $q_{i,j}$, para $1 \leq i \leq n$ e $1 \leq j \leq t$;
- c_i, para $1 \leq i \leq n$;
- Um estado especial s.

O alfabeto é $\{0,1\}$ e a associação entre os problemas é feita da seguinte forma: $x_j =$ **verdadeiro**, se a j-ésima letra da palavra é 1; ou $x_j =$ **falso** se a j-ésima letra da palavra é 0. A função de transição é definida como segue:

- Uma transição rotulada por 1 de $q_{i,j}$ para s e outra rotulada por 0 para $q_{i,j+1}$ se a i-ésima cláusula da fórmula somente contiver a variável x_j na sua forma literal não-negada.

- Uma transição rotulada por 0 de $q_{i,j}$ para s e outra rotulada por 1 para $q_{i,j+1}$ se a i-ésima cláusula da fórmula somente contiver a variável x_j na sua forma literal negada;
- Uma transição rotulada por $0, 1$ de $q_{i,j}$ para s, se a i-ésima cláusula da fórmula contiver a variável x_j nas formas negativa e positiva;
- Uma transição rotulada por $0, 1$ de $q_{i,j}$ para $q_{i,j+1}$, se $j < t$, e de $q_{i,j}$ para c_i, se $j = t$, caso a i-ésima cláusula da fórmula não contenha a variável x_j;
- Uma transição rotulada por $0, 1$ para o estado s a partir de todos os estados c_i;
- Uma transição rotulada por $0, 1$ de s para s.
- As transições para s, exceto se partirem de algum estado do tipo c_i, são chamadas de **atalho**.

A Figura 2.8 apresenta a construção deste autômato para a fórmula proposicional $(x_1 \vee \bar{x_2} \vee x_4) \wedge (x_2 \vee x_3) \wedge (x_3 \vee \bar{x_4})$. Observe que neste exemplo $t = 4$ e $n = 3$. Neste autômato, cada linha representa uma cláusula da fórmula proposicional.

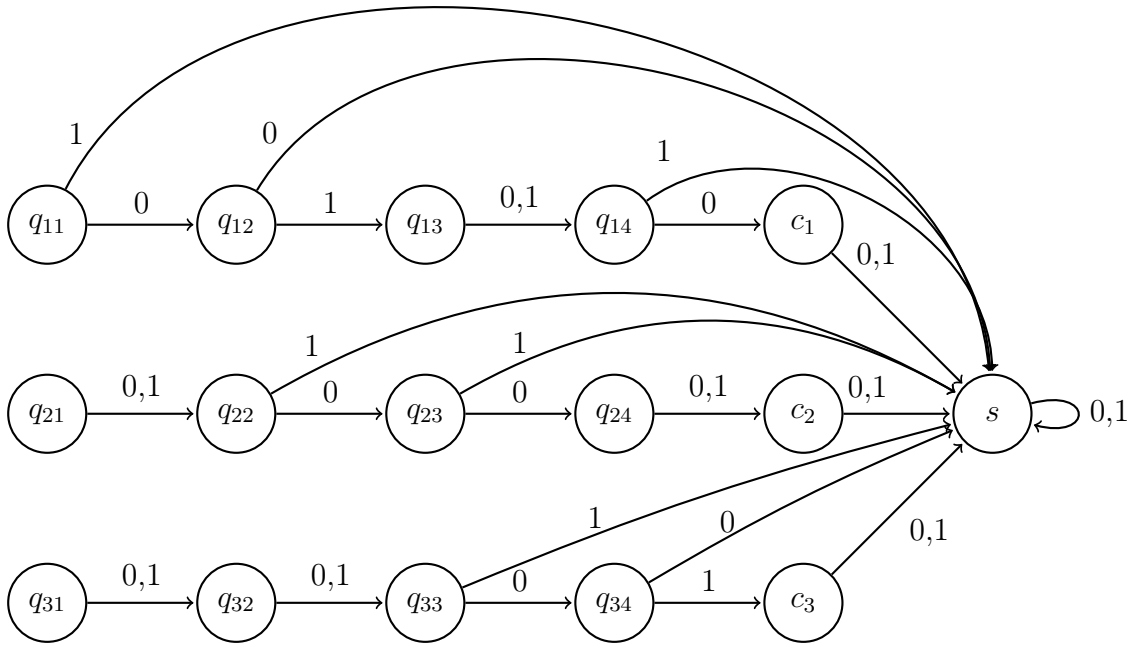

Figura 2.8: *O autômato construído por Eppstein para provar a* NP*-completude do problema* SINC.

Toda palavra de comprimento $t + 1$ sincroniza o autômato, ou seja, atinge s. Contudo, somente uma palavra com tamanho no máximo t poderá corresponder a uma associação satisfazível. Atingir algum estado c_i representa que a cláusula não foi satisfeita, uma vez que passou por todos os estados (que correspondem às variáveis) e não atingiu s através de um atalho (que corresponde à variável ter uma associação verdadeira na cláusula).

Observe que a partir dos estados $q_{i,2}$ toda palavra de tamanho t alcança s. Da mesma forma, a partir de $q_{i,3}$ toda palavra de tamanho $t-1$ leva estes estados a s e assim sucessivamente. Portanto, todos os estados, exceto os $q_{i,1}$, atingem s com qualquer palavra de tamanho pelo menos t. Dessa forma, é suficiente analisar apenas o que acontece em relação à leitura da palavra a partir de algum estado $q_{i,1}$.

Se existir uma palavra sincronizadora de tamanho no máximo t, então todos os estados $q_{i,1}$ irão atingir s através de um atalho. Caso contrário, pelo menos um estado passaria por um estado c_i antes de alcançar s e para isso seria necessário uma palavra de tamanho $t+1$.

Uma palavra sincronizadora de tamanho no máximo t equivale a um atalho por cláusula, ou seja, ao valor 1 (verdadeiro) associado a cada cláusula e portanto a fórmula proposicional tem como resultado o valor 1, ou seja, é satisfazível. Dessa forma, se existe uma palavra sincronizadora de tamanho no máximo t, então, uma associação de valores que torna a fórmula satisfazível pode ser obtida a partir desta palavra. Por outro lado, se temos uma associação satisfazível, então temos t valores para as t variáveis e podemos construir uma palavra com este tamanho. A leitura desta palavra a partir de qualquer estado $q_{i,1}$ atinge s antes de atingir qualquer estado c_i, pois sabemos que esta associação é satisfazível e que portanto não atinge nenhum c_i. Logo, esta é uma palavra sincronizadora para o autômato e tem tamanho t. Sendo assim, a fórmula é satisfazível se e somente se o autômato tiver uma palavra sincronizadora com tamanho t.

Em relação à complexidade computacional da redução, para a construção do autômato é necessária uma varredura inicial sobre a fórmula proposicional, a fim de determinar o número de cláusulas e variáveis. Em seguida se constrói o autômato que contém $n\ (t+1)+1$ estados e em tn passos é possível determinar as transições, com base nas variáveis presentes (e em seus sinais) em cada cláusula. Dessa forma, se pudermos supor que o tamanho t da palavra é polinomial em relação ao número de estados do autômato original (n), então o tempo da redução é polinomial.

Por fim, uma vez que podemos testar o autômato para uma palavra sincronizadora de tamanho t, podemos também testar a satisfabilidade de uma fórmula proposicional (na FNC) com t variáveis simplesmente convertendo a fórmula para o autômato. Portanto, o Problema `SINC` é NP-completo, visto que `SAT` é NP-completo. □

Capítulo 3

Algoritmos para obter palavras sincronizadoras

Como visto, o problema SINC é NP-completo e, portanto, não possui solução em tempo polinomial a menos que $P = NP$. Deste modo, na tentativa de obter soluções mais eficientes para encontrar palavras sincronizadoras surgem os métodos heurísticos.

No início deste capítulo são apresentados dois métodos aplicados tanto em demonstrações de resultados parciais da conjectura, quanto em algoritmos para encontrar palavras sincronizadoras. Em seguida são exibidos algoritmos com diferentes abordagens para encontrar palavras sincronizadoras. Por fim são apresentados os resultados dos experimentos comparativos realizados com estes algoritmos.

3.1 Métodos de Fusão e Extensão

Existem dois métodos gerais que são utilizados tanto em demonstrações que apresentam resultados parciais sobre a conjectura, quanto em algoritmos para encontrar palavras sincronizadoras. Em ambos os métodos é construída uma sequência finita de palavras $W = \{w_1, w_2, \ldots, w_k\}$ de forma que a sua concatenação resulta em uma palavra sincronizadora.

O primeiro método é chamado **Método de Fusão**. A ideia principal deste método é ir concatenando palavras que eliminam estados até que se obtenha um conjunto com somente um estado. Ou seja, as palavras em W vão unindo os estados de Q até que seja alcançado um estado unitário q: $|Q| > |Q \cdot w_1| > |Q \cdot w_1 w_2| > \cdots > |Q \cdot w_1 w_2 \cdots w_k| = |\{q\}|$. Este método é mais utilizado algoritmicamente. Nas seções seguintes são apresentados três algoritmos que usam diferentes heurísticas na aplicação deste método.

O segundo método é chamado **Método de Extensão**. A ideia deste método é inversa à do método anterior: a palavra é construída do fim para o início e as palavras concatenadas agregam estados até que se obtenha o conjunto total de estados do autômato. Isto significa que as palavras em W estendem um estado q até o conjunto Q. Uma vez que o autômato

é sincronizado, existe um estado sincronizador q que é alcançado por uma mesma letra $a \in \Sigma$ a partir de pelo menos dois estados. Então podemos supor que $w_k = a$: $|\{q\}| < |q \cdot a^{-1}| < |q \cdot a^{-1}w_{k-1}^{-1}| < \cdots < |q \cdot a^{-1}w_{k-1}^{-1} \cdots w_1^{-1}| = |Q|$. Este método é mais utilizado em demostrações de resultados parciais. Na seção 3.5 é apresentado um método algorítmico para a aplicação deste método.

Observe que o número de palavras em W não ultrapassa $n - 1$, pois esse é o valor máximo de aumentações (ou reduções) de estados possíveis. Logo, $k \leq n - 1$ e, portanto, se for possível apresentar um limitante linear no tamanho das palavras em W, então é possível obter um limitante superior quadrático no tamanho da menor palavra sincronizadora.

3.2 Algoritmo para obter uma palavra mínima

Determinar se um autômato é sincronizado e obter uma palavra sincronizadora mínima pode ser realizado de forma trivial através da construção de um autômato formado pelo conjunto das partes dos estados do autômato original.

Dado um autômato $A = (Q, \Sigma, \delta)$ (com $|Q| = n$), constrói-se $A' = (Q', \Sigma, \delta')$, um novo autômato chamado **autômato de subconjuntos**. Neste novo autômato, cada estado é um subconjunto não-vazio de Q, isto é, o conjunto de estados do autômato A' é formado pelo conjunto das partes do conjunto de estados autômato original (que chamaremos de $P(Q)$), sem o conjunto vazio: $Q' = P(Q)\backslash\emptyset$. As transições deste autômato são definidas por $\delta'(Q'_i, a) = \{\delta(q_i, a) | q_i \in Q'_i\}$, onde $Q'_i \in Q'$ e $a \in \Sigma$ [Vol08].

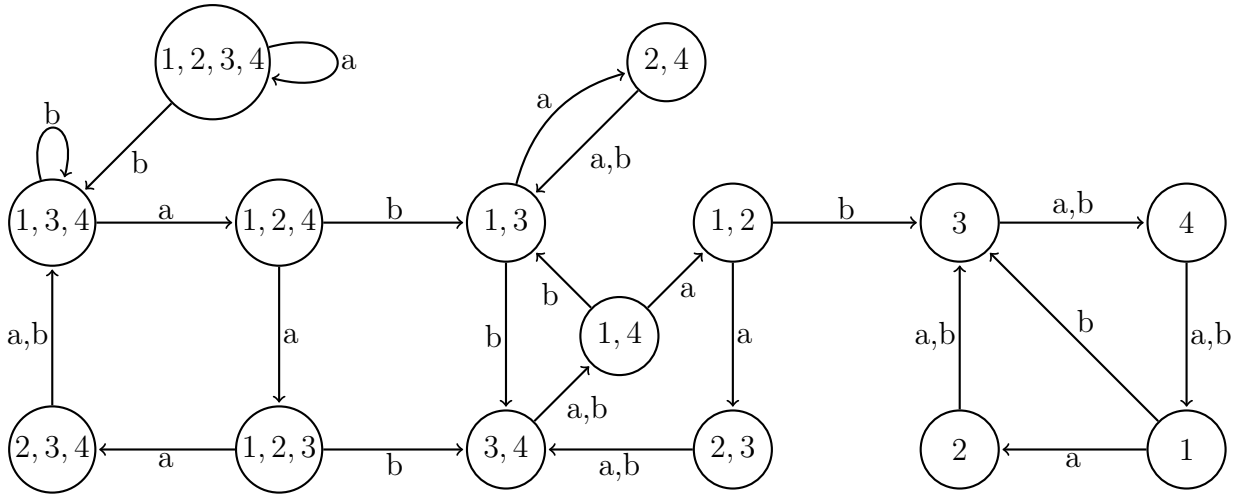

Figura 3.1: *O autômato de subconjuntos.*

O autômato de subconjuntos da Figura 3.1 foi construído a partir do autômato da Figura 1.1 e, como visto, a palavra *baabaab* é sincronizadora para este autômato. É fácil perceber que *baabbab* também sincroniza o autômato A' e que ambas são palavras sincronizadoras de tamanho mínimo. Observe na Figura 3.1 que estas palavras, lidas a partir do estado $\{1, 2, 3, 4\}$, levam ao estado $\{3\}$.

Note que este autômato é bastante útil para análise da ação de uma letra, ou de uma sequência de letras, sobre um conjunto de estados. Além disso, se durante a construção do autômato de subconjuntos for definido que o estado que contém Q é o estado inicial e os conjuntos unitários são os estados finais, então o autômato resultante aceita todas as palavras que sincronizam o autômato original.

Portanto, se for possível encontrar um ou mais caminhos do vértice que contém Q até um conjunto unitário, o autômato é sincronizado. Observe que através deste autômato identificar palavras sincronizadoras torna-se trivial: basta traçar um caminho inverso a partir de um conjunto unitário até o estado que contém Q. Dessa forma, através de uma busca em largura neste autômato podemos identificar uma palavra sincronizadora de tamanho mínimo.

Contudo, como o número de estados do autômato de subconjuntos é $2^n - 1$, esta construção leva tempo exponencial em relação ao número de estados do autômato original. E, desta forma, a busca em largura também gasta tempo exponencial em relação a n.

3.3 Algoritmo de Eppstein

Eppstein apresentou um algoritmo guloso para encontrar uma palavra sincronizadora [Epp90]. As palavras obtidas por este algoritmo possuem limitante cúbico e, portanto, não estão dentro do limite da conjectura. De fato, como visto no início do Capítulo 2, o melhor limitante para o caso geral da conjectura ainda é de ordem cúbica.

O algoritmo de Eppstein é, na verdade, uma modificação do algoritmo apresentado por Natarajan [Nat86]. O que difere estes dois algoritmos é um pré-processamento realizado sobre o autômato que permite um ganho relevante em eficiência na versão de Eppstein.

O pré-processamento é realizado em duas etapas. Na Etapa 1 é criado um novo autômato a partir do autômato original. Este novo autômato, o **autômato de pares**, é representado por A^2 e construído da seguinte forma:

i. O conjunto de estados Q_{A^2} consiste de todos os pares de estados não-ordenados, além de um estado estado especial *sinc*;

ii. As transições, para todo $a \in \Sigma$, são:

- $\delta_{A^2}(\{q, r\}, a) = \{s, t\}$, se $\delta(q, a) = s$ e $\delta(r, a) = t$;
- $\delta_{A^2}(\{q, r\}, a) = sinc$ se $\delta(q, a) = \delta(r, a)$;
- $\delta_{A^2}(sinc, a) = sinc$.

Ainda na Etapa 1 é realizada uma busca em largura no autômato de pares tendo como raiz o estado *sinc*. Durante a busca, para cada par q, r é construído o caminho mínimo até o estado *sinc*. O produto desta etapa é uma **Árvore de Caminhos Mínimos**.

Observe que o tempo desta etapa é resultado da construção do autômato de pares e da busca em largura neste autômato, sendo, portanto: $O(kn^2) + O(n^2) = O(kn^2)$, onde k é o tamanho do alfabeto.

Na Etapa 2 são armazenadas informações importantes para cada par de estados: a palavra mínima do par obtida através da Árvore de Caminhos Mínimos e os estados que permanecem ativos após a aplicação desta palavra sobre o conjunto Q de estados.

Para isto, a Árvore de Caminhos Mínimos criada na etapa anterior é explorada através de uma busca em profundidade e cada par de estados x, y é rotulado com a palavra mínima obtida percorrendo o caminho, nesta árvore, deste par até o estado *sinc*. Desta forma, a raiz é rotulada com a palavra vazia e um par q, r em um nível i da árvore, cuja aresta de saída tem rótulo $\alpha \in \Sigma$ e atinge outro par s, t no nível $i-1$ tem como rótulo a palavra $\alpha\bar{w}_{s,t}$, onde $\bar{w}_{s,t}$ é a palavra mínima obtida através da Árvore de Caminhos Mínimos para o par s, t.

Ainda nesta etapa do pré-processamento, ao explorar cada par de estados q, r é também computado $\delta(x, \bar{w}_{q,r})$, $\forall\ x \in Q$. Isto pode ser calculado, em tempo constante para cada estado, da seguinte forma: seja $\bar{w}_{q,r} = \alpha\bar{w}_{s,t}$, para $\alpha \in \Sigma$, com $\delta(q, \alpha) = s$ e $\delta(r, \alpha) = t$, se $s = t$ então $\bar{w}_{s,t} = \lambda$ e basta calcular $\delta(x, \alpha)$. Se $s \neq t$, então, $\delta(x, \bar{w}_{q,r}) = \delta(\delta(x, \alpha), \bar{w}_{s,t})$ pode ser computado apenas pelo cálculo de $\delta(x, \alpha)$ seguido pela consulta ao valor de $\delta(\delta(x, \alpha), \bar{w}_{s,t})$ previamente calculado, visto que a árvore é explorada através de uma busca em profundidade.

Dessa forma, serão realizadas $O(n^3)$ computações de tempo constante, $O(n^2)$ para cada vértice da árvore e para cada um deles são realizadas n computações de transições, uma para cada estado de Q. Portanto, $O(n^3)$ é o tempo total da Etapa 2 e o tempo total do pré-processamento é $O(n^3 + kn^2)$.

Esta segunda etapa do pré-processamento visa otimizar o cálculo da linha 7 do algoritmo.

Observe que um autômato é sincronizado se e somente se existem palavras sincronizadoras para cada par de estados. Uma vez que a cardinalidade do conjunto de estados é reduzida após a aplicação de uma palavra que sincroniza um par, e cada par possui uma palavra sincronizadora, então, após a leitura de cada uma destas palavras a cardinalidade do conjunto de estados será a mínima e o autômato sincroniza. Desta forma, após o pré-processamento é possível responder se o autômato é sincronizado.

O algoritmo de Eppstein é descrito a seguir:

Algoritmo 1: Algoritmo de Eppstein.

Entrada: $A(Q, \Sigma, \delta)$: um autômato fortemente conexo.

Saída: w: uma palavra sincronizadora.

```
1 início
2     Pré-processamento do autômato;
3     P ← Q;
4     w ← λ;
5     enquanto |P| > 1 faça
6         Encontre um par q, r com q ≠ r tal que: w̄_{q,r}=min{w̄_{p,s} | p, s ∈ P};
7         P ← δ(P, w̄_{q,r});
8         w ← w w̄_{q,r};
9 fim
```

Terminado o pré-processamento, o algoritmo entra no laço da linha 5. A cada iteração do laço, para encontrar a menor dentre as palavras que sincronizam os pares ativos, basta procurar pela palavra associada a cada par de estados na estrutura computada na Etapa 2 do pré-processamento. Dessa forma, uma vez que existem $O(n^2)$ pares de estados temos que:

i. A menor palavra escolhida tem tamanho $O(n^2)$;

ii. Percorrer cada par de estados para escolher o par com menor palavra gasta tempo $O(n^2)$.

Calcular $\delta(P, \overline{w}_{q,r})$ na verdade significa procurar pelos estados $s \in P$ calculados na Etapa 2 do pré-processamento. Desta forma, este passo gasta tempo $O(n)$. Observe que o tamanho do conjunto P diminui em pelo menos 1 a cada iteração, logo, o laço principal será executado no máximo $n - 1$ vezes. Portanto, cada iteração gasta $O(n^2 + n)$ e o laço principal tem, portanto, complexidade $(n-1) \times O(n^2 + n) = O(n^3 + n^2)$. Uma vez que o tempo do pré-processamento é $O(n^3 + kn^2)$, o tempo total gasto pelo algoritmo é limitado por $O(n^3 + kn^2)$.

Para ilustrar o funcionamento deste algoritmo vamos usar como exemplo o autômato C_4 (Figura 2.3). As Figuras 3.2 e 3.3 apresentam o autômato de pares e a árvore de caminhos mínimos construídos durante o pré-processamento sobre o autômato C_4.

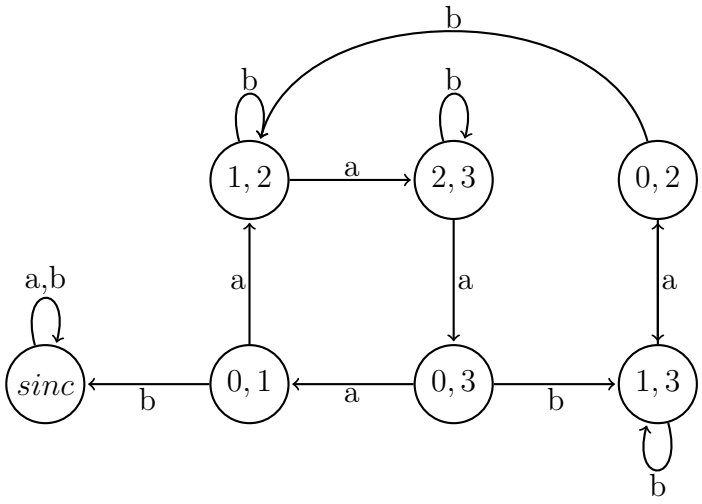

Figura 3.2: *O autômato A^2 gerado durante o pré-processamento.*

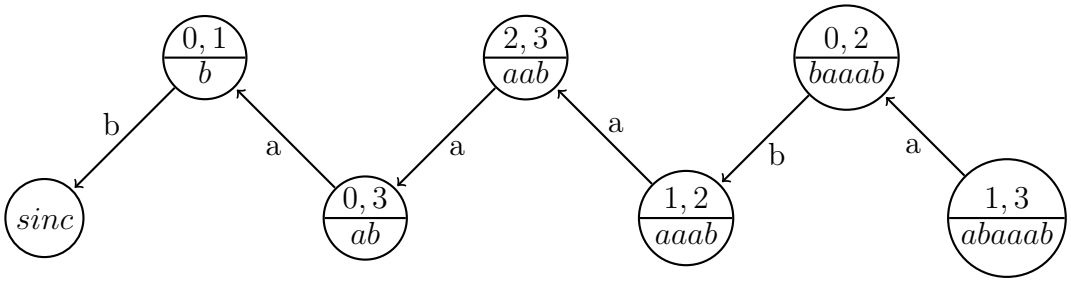

Figura 3.3: *Árvore de caminhos mínimos construída durante o pré-processamento.*

Durante a Etapa 2 a árvore de caminhos mínimos é percorrida para o cálculo das transições. A seguir apresentamos como é feito este cálculo para os três primeiros pares visitados em pré-ordem (a operação que é de fato realizada em cada passo é apresentada em negrito):

Par 0,1 : $\overline{w}_{0,1} = b$

$\boldsymbol{\delta(0, b)} = 1$

$\boldsymbol{\delta(1, b)} = 1$

$\boldsymbol{\delta(2, b)} = 2$

$\boldsymbol{\delta(3, b)} = 3$

Par 0,3 : $\overline{w}_{0,3} = ab$

$$\delta(0, ab) = \delta(0, a\overline{w}_{0,1}) = \delta(\boldsymbol{\delta(0, a)}, \overline{w}_{0,1}) = \delta(1, b) = 1$$
$$\delta(1, ab) = \delta(1, a\overline{w}_{0,1}) = \delta(\boldsymbol{\delta(1, a)}, \overline{w}_{0,1}) = \delta(2, b) = 2$$
$$\delta(2, ab) = \delta(2, a\overline{w}_{0,1}) = \delta(\boldsymbol{\delta(2, a)}, \overline{w}_{0,1}) = \delta(3, b) = 3$$
$$\delta(3, ab) = \delta(3, a\overline{w}_{0,1}) = \delta(\boldsymbol{\delta(3, a)}, \overline{w}_{0,1}) = \delta(0, b) = 1$$

Par 2,3 : $\overline{w}_{2,3} = aab$

$$\delta(0, aab) = \delta(0, a\overline{w}_{0,3}) = \delta(\boldsymbol{\delta(0, a)}, \overline{w}_{0,3}) = \delta(1, ab) = 2$$
$$\delta(1, aab) = \delta(1, a\overline{w}_{0,3}) = \delta(\boldsymbol{\delta(1, a)}, \overline{w}_{0,3}) = \delta(2, ab) = 3$$
$$\delta(2, aab) = \delta(2, a\overline{w}_{0,3}) = \delta(\boldsymbol{\delta(2, a)}, \overline{w}_{0,3}) = \delta(3, ab) = 1$$
$$\delta(3, aab) = \delta(3, a\overline{w}_{0,3}) = \delta(\boldsymbol{\delta(3, a)}, \overline{w}_{0,3}) = \delta(0, ab) = 1$$

Após o pré-processamento e antes de entrar no laço temos: $w = \lambda$ e $P = \{0, 1, 2, 3\}$. A cada passo o algoritmo procura pelo par de estados que possui menor palavra sincronizadora. Na primeira iteração o par escolhido será $0, 1$, com $\overline{w}_{0,1} = b$. Aplicando esta palavra ao conjunto atual de estados vamos obter: $\{0, 1, 2, 3\} \cdot b = \{1, 2, 3\}$ e agora $w = b$. Os pares de estados disponíveis agora são: $\{1, 2\}, \{1, 3\}, \{2, 3\}$. Dentre estes, o que possui a menor palavra sincronizadora é o par $\{2, 3\}$, com $\overline{w}_{2,3} = aab$. Então teremos: $\{1, 2, 3\} \cdot aab = \{1, 3\}$ e $w = b \cdot aab$. Agora o único par disponível é $\{1, 3\}$ e portanto a palavra escolhida neste passo é $\overline{w}_{1,3} = abaaab$. Sendo assim, aplicamos esta palavra aos estados atuais obtendo $\{1, 3\} \cdot abaaab = \{1\}$ e $w = b \cdot aab \cdot abaaab$. Finalmente temos $|P| = 1$, com $w = baababaaab$. Observe que a palavra obtida possui tamanho 10 e, como visto, a palavra mínima para este autômato possui tamanho 9.

3.4 Algoritmos de Roman

Roman apresentou dois algoritmos polinomiais para encontrar palavras sincronizadoras. A ideia destes algoritmos se assemelha à do algoritmo de Eppstein: é construído um autômato com pares de estados e são escolhidas palavras mais curtas que sincronizam estes pares [Rom05]. A principal diferença nos algoritmos de Roman é a análise de um passo a frente do atual, isto é, a verificação de como a escolha de uma palavra afeta os outros estados.

Esta análise do passo a frente acarreta em um custo computacional maior dos Algoritmos de Roman em relação ao Algoritmo de Eppstein. Contudo, a proposta principal destes algoritmos é tentar obter palavras sincronizadoras menores.

Seja $w_{min}(p) \in \Sigma^*$ uma palavra de tamanho mínimo que leva um par de estados p até o estado sincronizador $sinc$ e seja $m(p)$ o tamanho desta palavra.

Se Q é o conjunto de estados e $\delta(Q, w) = P$, então este conjunto P de estados será chamado de **estados ativos** após a leitura de w. Os estados $Q \setminus P$ são chamados **estados inativos**.

Assim como no algoritmo de Eppstein, um pré-processamento é realizado sobre o autômato. Durante este pré-processamento o autômato A^2 é construído (idêntico ao apresentado na Seção 3.3) e, da mesma forma que no Algoritmo de Eppstein, através de uma busca em largura com raiz em $sinc$ é construída uma **Árvore de Caminhos Mínimos**. Além disso, para cada par de estados p em Q_{A^2} é armazenada a palavra $w_{min}(p)$ e o seu tamanho $m(p)$. Isto pode ser feito em tempo $O(kn^2)$, onde $k = |\Sigma|$ e $n = |Q|$.

Computar a avaliação do passo à frente trata-se de analisar o impacto (sobre os demais estados ativos) da palavra a ser concatenada em uma certa iteração. Em vista disto, vamos definir a função Δ para $p, q \in Q_{A^2}$:

i. $\Delta_p(q, w_{min}(p)) = m(\delta(q, w_{min}(p))) - m(p)$, se $q \neq p$;

ii. $\Delta_p(q, w_{min}(p)) = 0$, se $q = p$.

O valor dessa função para dois pares p, q determina, para o estado q, se a aplicação da palavra $w_{min}(p)$ o deixa mais próximo ou mais distante da sincronização. Durante a etapa de pré-processamento é construída uma tabela com os valores da função Δ de um par de estados em relação a todos os pares.

Seja P o conjunto de estados ativos em uma determinada iteração. Desejamos calcular o impacto da palavra mínima para um par de estados p, sobre os estados em P. Este impacto é calculado pelas funções Φ. A função Φ_1 calcula o valor deste impacto para todos os estados ativos e Φ_2, além de calcular o impacto, leva em conta o tamanho da palavra. As funções Φ são definidas a seguir:

- $\Phi_1(w_{min}(p)) = \sum_{q \in P} (\Delta_p(q, w_{min}(p)))$;
- $\Phi_2(w_{min}(p)) = \sum_{q \in P} (\Delta_p(q, w_{min}(p))) + |w_{min}(p)|$.

Algoritmo 2: Algoritmo de Roman.

Entrada: $A(Q, \Sigma, \delta)$: um autômato fortemente conexo.

Saída: w: uma palavra sincronizadora.

```
1  início
2      Pré-processamento do autômato;
3      Constrói a tabela de funções delta;
4      P ← Q ;
5      w ← λ ;
6      enquanto |P| > 1 faça
7          Para cada par de estados ativo q ∈ Q_{A²}, calcule Φ1(w_min(p)), escolha o estado
           q' com menor valor da função Φ1 e armazene em w̄ a palavra w_min(q');
8          w ← w w_min(q');
9          P ← δ(P, w_min(q'));
10     Devolva w;
11 fim
```

Na linha 7, ao subtituir Φ_1 por Φ_2 obtemos o segundo algoritmo de Roman.

As duas primeiras operações realizam um pré-processamento sobre o autômato. A primeira consiste da construção do autômato A^2, da árvore de caminhos mínimos e do armazenamento das palavras de tamanho mínimo para cada par. Este pré-processamento pode ser realizado em tempo $O(kn^2)$, como visto anteriormente. No pré-processamento realizado na linha 2, para cada par de estados do A^2 é computada a função Δ em relação a todos os outros pares. É construída uma tabela Δ com estes valores. Calcular o valor da função Δ de cada par de estados $q \in Q_{A^2}$ em relação a todos os outros pares tem complexidade $O(n^4)$, visto que $|Q_{A^2}| = \frac{n(n-1)}{2} + 1 = O(n^2)$. Então o tempo do pré-processamento dos algoritmos de Roman é $O(n^4)$.

Em relação ao tempo do algoritmo principal, note que quando um par de estados é sincronizado o tamanho do conjunto P e o número de estados ativos diminuem em pelo menos 1. Sendo assim, uma vez que P inicialmente contém todo o conjunto de estados do autômato original, o laço da linha 6 é executado no máximo $n - 1$ vezes. Além disso, na primeira iteração os estados ativos são, na verdade, todos os estados do autômato de pares. Dessa forma, a linha 7 é executada no máximo $\frac{n(n-1)}{2}$ vezes.

Agora observe que o cálculo da função Φ para um par de estados q realiza a soma dos valores Δ para todos os outros estados ativos, ou seja, soma os valores dos estados ativos na linha referente a este par na tabela Δ. Esta operação percorre então, no máximo, $\frac{n(n-1)}{2}$ estados para realizar a soma. Logo, a linha 7 gasta tempo da ordem $O(n^4)$. E como este laço será executado no máximo $n - 1$ vezes, o tempo do algoritmo principal é $O(n^5)$.

Para ilustrar o funcionamento deste algoritmo vamos usar o autômato C_4 (Figura 2.2). O autômato A^2 e a árvore de caminhos mínimos construídos durante o pré-processamento

do algoritmo são apresentados nas Figuras 3.2 e 3.3. Na Figura 3.3 foi omitido o valor do tamanho da palavra mínima.

Ainda durante o pré-processamento é criada uma tabela com todos os valores da função Δ. A seguir mostramos como é calculada a primeira linha desta tabela:
$\Delta_{\{0,1\}}(\{0,1\},b) = 0$ (por definição).
$\Delta_{\{0,1\}}(\{0,2\},b) = m(\delta(\{0,2\},b)) - m(\{0,2\}) = m(\{1,2\}) - m(\{0,2\}) = 4-5 = -1.$
$\Delta_{\{0,1\}}(\{0,3\},b) = m(\delta(\{0,3\},b)) - m(\{0,3\}) = m(\{1,3\}) - m(\{0,2\}) = 6-2 = 4.$
$\Delta(_{\{0,1\}}\{1,2\},b) = m(\delta(\{1,2\},b)) - m(\{1,2\}) = m(\{1,2\}) - m(\{1,2\}) = 4-4 = 0.$
$\Delta_{\{0,1\}}(\{1,3\},b) = m(\delta(\{1,3\},b)) - m(\{1,3\}) = m(\{1,3\}) - m(\{1,3\}) = 6-6 = 0.$
$\Delta_{\{0,1\}}(\{2,3\},b) = m(\delta(\{2,3\},b)) - m(\{2,3\}) = m(\{2,3\}) - m(\{2,3\}) = 3-3 = 0.$

Estes valores representam o impacto, sobre os outros pares, da menor palavra $w_{min}(0,1) = b$ que leva o par $0,1$ até o estado *sinc*.

pares	0,1	0,2	0,3	1,2	1,3	2,3
0,1	0	-1	4	0	0	0
0,2	-1	0	2	-4	-2	1
0,3	3	1	0	-1	-2	3
1,2	5	1	1	0	-2	1
1,3	-1	-1	-2	0	0	1
2,3	2	-1	2	2	0	0

Tabela 3.1: *Tabela com os valores da função* Δ.

Na primeira iteração do laço da linha 8 temos como estados ativos todo o conjunto Q_{A^2} de estados. Portanto, a função Φ é calculada para todos os pares de estados. A tabela 3.2 apresenta os valores de Φ_1 e Φ_2 que são calculados na linha 9 do algoritmo. Na primeira iteração, para ambas as funções o menor valor (em negrito) é o valor referente à palavra $w_{min}(0,2) = baaab$. Então $w \leftarrow \lambda \cdot baaab$ e o novo conjunto de estados é $Q \cdot baaab = \{1,2\}$. Como o único par à disposição na próxima iteração é $1,2$, calcula-se apenas $\Phi_1(\{1,2\})$ e $\Phi_2(\{1,2\})$. Dessa forma, como $\Delta_p(p, w_{min}(p)) = 0$ para todo $p \in Q_{A^2}$, então $\Phi_1(\{1,2\}) = 0$ e $\Phi_2(\{1,2\}) = 0 + |w_{min}(\{1,2\})| = 4$. Agora $w \leftarrow baaab \cdot aaab$ e o conjunto de estados é $\{1,2\} \cdot aaab = \{2\}$. Portanto, o algoritmo obtém $w = baaabaaab$ como palavra sincronizadora. Neste caso, a palavra obtida coincide com a palavra mínima.

Iteração	Φ	0,1	0,2	0,3	1,2	1,3	2,3
1	Φ_1	3	**-4**	4	6	-3	5
	Φ_2	4	**1**	6	10	3	8
2	Φ_1				0		
	Φ_2				4		

Tabela 3.2: *Tabela com os valores da função* Φ.

3.5 Método Algorítmico de Extensão

O método algorítmico a seguir foi apresentado por Berlinkov e reúne a essência da ideia de sequências de extensão para encontrar uma palavra sincronizadora [Ber10]. As entradas para este algoritmo são:

i. Um autômato fortemente conexo $A(Q, \Sigma, \delta)$ com n estados;

ii. P: um subconjunto não-vazio de Q;

iii. $v_p \in \Sigma^*$, tal que $|P \cdot v_p| = 1$.

Algoritmo 3: Método Algorítmico de Extensão.

Entrada: A, P e v_p .

Saída: w: uma palavra sincronizadora.

```
1  início
2      P̄ ← P;
3      w ← v_p ;
4      enquanto |P̄| < |Q| faça
5          Procure uma palavra u ∈ Σ*, de tamanho mínimo, tal que |P̄ · u^{-1}| > |P̄|;
6          se não for possível encontrar tal palavra então
7              Devolva "Falha";
8          w ← uw;
9          P̄ ← P̄ · u^{-1};
10     Devolva w;
11 fim
```

Se o autômato é sincronizado, então, a cada iteração do laço (linha 4) existe u tal que $|\bar{P} \cdot u^{-1}| > |\bar{P}|$ [1]. Além disso, este laço é executado no máximo $|Q| - |P| = n - |P|$ vezes, pois a cada iteração o conjunto $\bar{P}$, que inicialmente é o próprio P, aumenta de tamanho em pelo menos uma unidade. Após a última iteração deste laço teremos $|\bar{P}| = |Q|$ e $\bar{P} = p \cdot w^{-1}$, para algum $p \in Q$. Logo, $p \cdot w^{-1} = \bar{P} = Q$.

Dessa forma, $|w| \leq |v_p| + (|Q| - |P|) \max u$, onde $\max u$ é o tamanho da maior palavra de extensão u. Observe que se o autômato é sincronizado a seguinte situação existe: $S \cdot a^{-1} = s$, para $a \in \Sigma$ e $S \subset Q$, com $|S| \geq 2$. Agora suponha $v_p = a$ e $P = S$, então $|w| \leq 1 + (n-2) \max u$.

Portanto, uma vez que seja possível estabelecer um limitante linear para a palavra de extensão u, o algoritmo terá como saída uma palavra sincronizadora de tamanho quadrático. Se isto for possível, torna-se viável alcançar um limitante quadrático para o caso geral da conjectura.

[1] definição do significado de $P \cdot w^{-1}$ na Seção 1.1

3.6 Comparação entre os algoritmos

Nesta seção são apresentados os experimentos realizados com algumas séries infinitas de autômatos. O objetivo destes experimentos foi comparar o comprimento das palavras obtidas pelos algoritmos de Eppstein e de Roman em relação ao comprimento da palavra sincronizadora mínima e ao limitante da conjectura $(n-1)^2$.

Como visto na Seção 3.3, durante o pré-processamento dos algoritmos de Eppstein e de Roman é criado um autômato de pares e é realizada uma busca em largura neste autômato a fim de criar uma árvore de caminhos mínimos. Contudo, durante a busca em largura pode ocorrer a seguinte situação: um par de estados atinge outro par através de mais de um símbolo do alfabeto. Diante disso, durante a implementação tornou-se necessário eleger alguns critérios de decisão sobre qual aresta explorar primeiro. Desta forma, com o intuito de tornar estas comparações mais imparciais, o algoritmo de Eppstein foi implementado utilizando cinco critérios distintos:

i. Eppstein 1 (Ep 1): A busca em largura é realizada respeitando a ordem lexicográfica do alfabeto;

ii. Eppstein 2 (Ep 2): A busca em largura é realizada com ordem lexicográfica reversa;

iii. Eppstein Aleatorizado 1 (Ep A 1): São realizadas duas buscas em larguras: uma com ordem lexicográfica e outra com a ordem reversa. No passo da linha 6 do algoritmo, caso exista mais de uma palavra com tamanho mínimo, estas palavras são armazenas e em seguida uma destas é sorteada de forma aleatória. Esta implementação aleatorizada do algoritmo de Eppstein é então executada 30 vezes e a menor palavra gerada durante estas execuções é retornada.

iv. Eppstein Aleatorizado 2 (Ep A 2): Para cada par de estados são computadas no pré-processamento palavras sincronizadoras de forma independente em relação a ordem lexicográfica. Durante o cálculo destas palavras, sempre que houver a situação de um par indo para outro com mais de um símbolo do alfabeto é realizado um sorteio aleatório de um símbolo. Além disso, assim como no critério anterior, caso no passo da linha 6 do algoritmo principal exista mais de uma palavra com tamanho mínimo, estas palavras são armazenas e em seguida uma destas é sorteada de forma aleatória. Esta implementação é também executada 30 vezes e a menor palavra gerada nestas execuções é retornada.

v. Eppstein Aleatorizado 3 (Ep A 3): O cálculo das palavras não é feito durante o pré-processamento, é calculado apenas o tamanho da palavra mínima para cada par. Neste caso, no passo da linha 6 do algoritmo principal, se existir mais de um par com palavra de tamanho mínimo, um destes pares é sorteado aleatoriamente e uma palavra mínima é então gerada. Novamente, sempre que houver a situação de um par indo para outro

com mais de um símbolo do alfabeto é realizada uma escolha aleatória. Esta implementação é também executada 30 vezes e a menor palavra gerada nestas execuções é retornada.

Mais informações sobre as implementações podem ser encontradas no Apêndice A. A seguir são apresentadas as séries testadas, as tabelas e os gráficos com os dados obtidos. Estas séries foram encontradas de maneira independente durante experimentos realizados em busca de séries infinitas de autômatos com palavras sincronizadoras quadráticas. Contudo, algumas destas séries já haviam sido apresentadas por Ananichev [AGV10].

Devido a restrições de memória, foram testados autômatos com até 28 estados.

Série 1

A Figura 3.4 exibe a Série 1 para n estados. Ananichev provou que esta série possui uma palavra sincronizadora mínima com tamanho $n^2 - 3n + 3$ [AGV10]. Observe que este autômato é isomorfo ao autômato da Figura 2.4 apresentado na Seção 2.1. Um exemplo de palavra sincronizadora mínima padrão para esta série é $(ba^{n-2})^{n-2}b$.

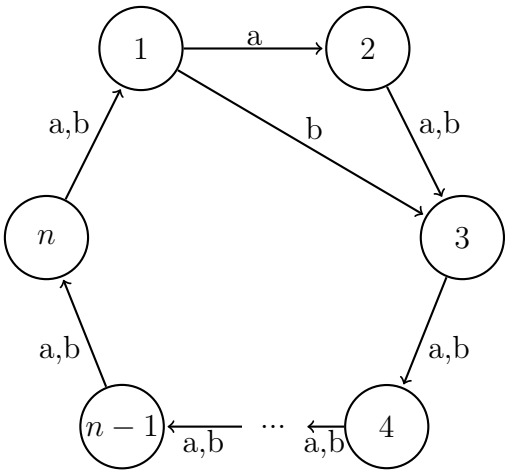

Figura 3.4: *A série infinita* 1.

A Tabela 3.3 apresenta os dados obtidos para autômatos pertencentes a esta série com $n \in \{6, 7, \ldots, 28\}$ estados.

Estados	Pal. Min.	Ep 1	Ep 2	Ep A 1	Ep A 2	Ep A 3	Roman Φ_1	Roman Φ_2	$(n-1)^2$
6	21	21	23	21	21	21	21	21	25
7	31	35	31	31	31	31	31	31	36
8	43	51	50	43	43	43	43	43	49
9	57	66	65	57	57	57	57	57	64
10	73	83	82	73	73	73	73	73	81
11	91	106	105	95	91	91	91	91	100
12	111	135	137	122	111	111	111	111	121
13	133	167	157	146	133	133	133	133	144
14	157	207	187	170	157	163	157	157	169
15	183	255	239	202	183	183	183	183	196
16	211	307	301	241	226	227	211	211	225
17	241	343	337	273	267	258	241	241	256
18	273	381	383	309	293	277	273	273	289
19	307	425	397	352	325	331	307	307	324
20	343	483	476	402	377	387	343	343	361
21	381	536	529	475	442	422	381	381	400
22	421	605	610	523	489	453	421	421	441
23	463	682	669	587	538	519	463	463	484
24	507	771	768	624	588	620	507	507	529
25	553	844	817	703	650	663	553	553	574
26	601	929	892	755	718	729	601	601	625
27	651	1022	919	828	743	763	651	651	676
28	703	1139	1035	932	848	830	703	703	729

Tabela 3.3: *Valores das palavras sincronizadoras obtidas pelos algoritmos para a Série* 1.

Esta série, assim como a Série de Černý, possui alfabeto de tamanho 2, uma das letras atua como uma permutação cíclica sobre o conjunto de estados e a outra letra é responsável por unir dois estados, neste caso: $\{0, 1\} \cdot b = \{3\}$. Dessa forma, é razoável pensar que nas situações onde a busca em largura precisa decidir qual aresta explorar (no caso de um par ir para outro com ambas as letras do alfabeto) seja mais interessante explorar primeiro a letra que une dois estados. Esta parece ser a explicação mais lógica para o algoritmo Eppstein 2 ter encontrado palavras menores que o Eppstein 1 para a maioria dos casos apresentados na Tabela 3.3.

Uma vez que os algoritmos aleatorizados não restringirem a escolha da próxima menor palavra a ser concatenada, torna-se possível, durante as 30 execuções do algoritmo, realizar uma escolha de palavra que sincronize mais pares de estados e que leve à sincronização do autômato de forma mais rápida. Desta forma, os algoritmos aleatorizados obtiveram palavras sincronizadoras menores em comparação com as versões não-aleatorizadas do Eppstein.

A Figura 3.5 ilustra graficamente a comparação dos algoritmos para esta série. Note que para todos os autômatos testados, os algoritmos Roman Φ_1 e Roman Φ_2 obtiveram palavra sincronizadora de mesmo comprimento da palavra mínima.

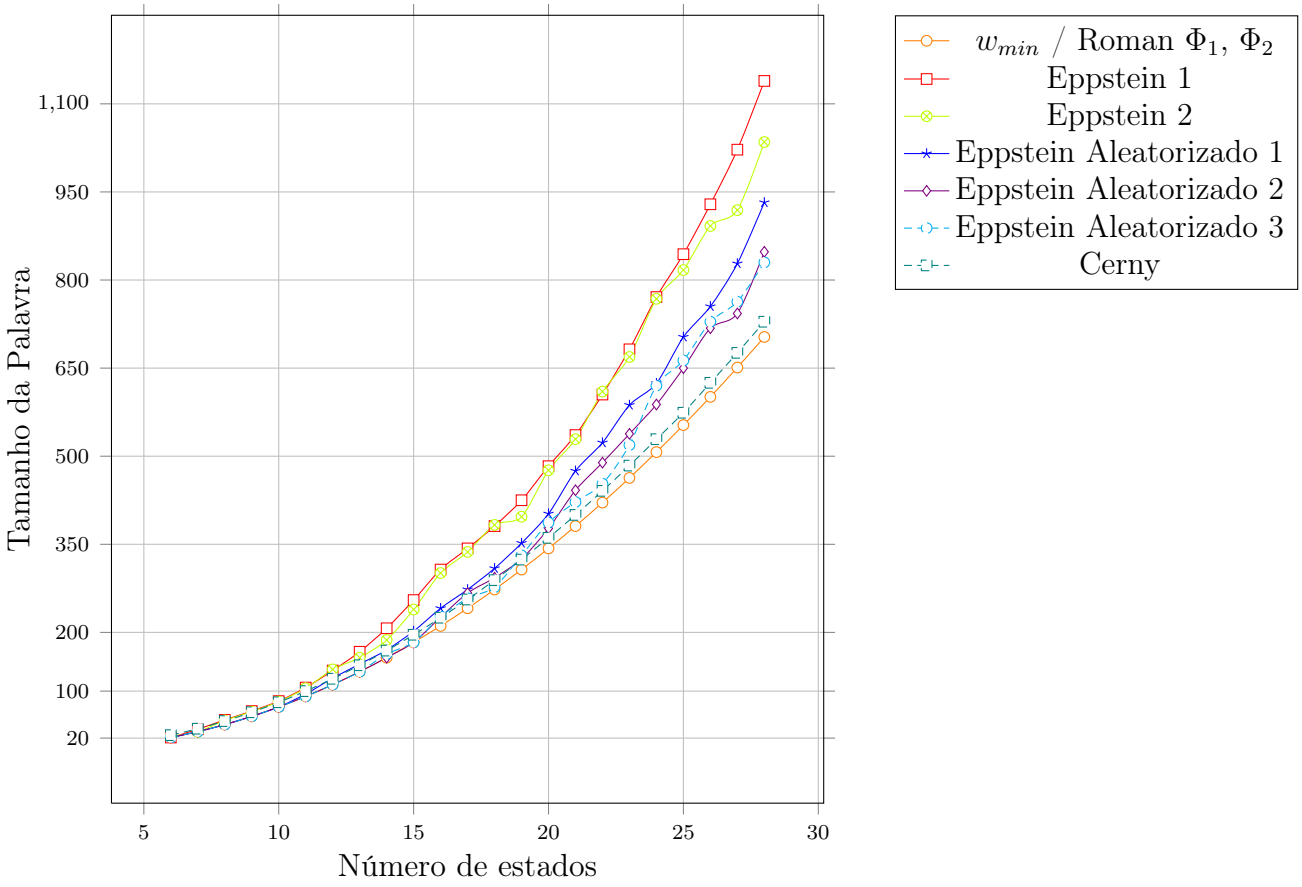

Figura 3.5: *Comparação gráfica dos algoritmos para a Série* 1.

Série 2

A Figura 3.6 exibe a Série 2 para n estados. Uma palavra sincronizadora mínima para esta série é $(baa)^{n-2}b$, com tamanho $3n-5$. Observe que esta série sincroniza com palavra de tamanho linear.

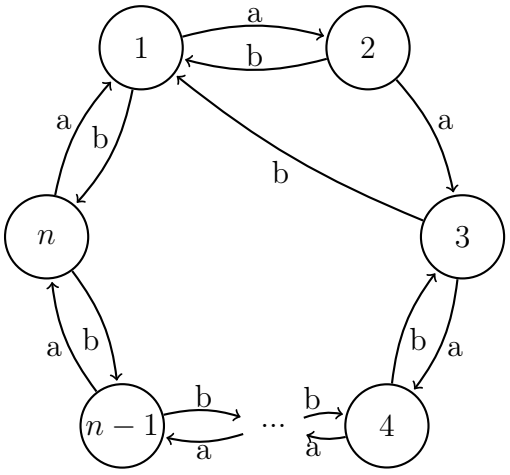

Figura 3.6: *A série infinita* 2.

Prova: A palavra $(baa)^{n-2}b$ é sincronizadora para os autômatos da Série 2 e esta palavra é mínima. No decorrer da prova são usadas as funções Intervalo e Intervalo Máximo definidas na Seção 2.1 durante a segunda prova para a Série de Černý, com apenas duas pequenas diferenças: $Q = \{1, \ldots, n\}$ e $j \in \{1, \ldots, n\}$.
Primeiro observe alguns fatos importantes sobre esta série:

i. A ação da letra a não altera o valor do intervalo máximo, visto que atua como uma permutação cíclica sobre o conjunto de estados.

ii. A ação da letra b aumenta o tamanho do intervalo em no máximo 1.

iii. Para aumentar o intervalo é necessário que $\{2, 3\} \in S \subseteq Q$.

De $(i.)$ concluímos que se w é uma palavra mínima para esta série, então w começa por b, pois, caso contrário, aplicando um ou mais a's estaríamos apenas permutando o conjunto de estados sem alterar o tamanho do intervalo máximo. Dessa forma, $w = b \cdot v$, com $v \in \Sigma^*$.

Inicialmente $I_{max}(Q) = 0$ e após a leitura do primeiro b: $S = Q \setminus \{2\}$, $\bar{S} = \{2\}$ e $I_{max}(S) = 1$.

Como nenhum estado atinge o estado 2 através de b e, por $(iii.)$, para haver uma nova aumentação de intervalo esta é uma condição necessária, então, uma palavra mínima que leva a uma nova aumentação deve composta por a's. A menor palavra que rotaciona os estados de forma que $\{2, 3\} \in S$ é aa, pois lendo somente um a o estado 3 não seria alcançado, visto que $\{2\} \notin S$ após a leitura de b.

Logo, $u = baa$ é uma palavra mínima que realiza uma aumentação de intervalo e a rotação necessária para uma nova aumentação. Como temos n estados, através da leitura da palavra

$(baa)^{n-2}$ são obtidas $n-2$ aumentações de intervalo e rotações. Portanto, $Q \cdot (baa)^{n-2} = \{2,3\}$.

Por fim, como b leva $\{2,3\}$ para o mesmo estado, $w = (baa)^{n-2}b$ é uma palavra sincronizadora para os autômatos da Série 2 e o seu tamanho é mínimo.

□

A Tabela 3.4 apresenta os dados comparativos obtidos para os autômatos pertencentes a esta série. Para os autômatos testados, todos os algoritmos obtiveram uma palavra sincronizadora mínima.

Estados	Pal. Min.	Ep 1	Ep 2	Ep A 1	Ep A 2	Ep A 3	Roman Φ_1	Roman Φ_2	$(n-1)^2$
6	13	13	13	13	13	13	13	13	25
7	16	16	16	16	16	16	16	16	36
8	19	19	19	19	19	19	19	19	49
9	22	22	22	22	22	22	22	22	64
10	25	25	25	25	25	25	25	25	81
11	28	28	28	28	28	28	28	28	100
12	31	31	31	31	31	31	31	31	121
13	34	34	34	34	34	34	34	34	144
14	37	37	37	37	37	37	37	37	169
15	40	40	40	40	40	40	40	40	196
16	43	43	43	43	43	43	43	43	225
17	46	46	46	46	46	46	46	46	256
18	49	49	49	49	49	49	49	49	289
19	52	52	52	52	52	52	52	52	324
20	55	55	55	55	55	55	55	55	361
21	58	58	58	58	58	58	58	58	400
22	61	61	61	61	61	61	61	61	441
23	64	64	64	64	64	64	64	64	484
24	67	67	67	67	67	67	67	67	529
25	70	70	70	70	70	70	70	70	574
26	73	73	73	73	73	73	73	73	625
27	76	76	76	76	76	76	76	76	676
28	79	79	79	79	79	79	79	79	729

Tabela 3.4: *Valores das palavras sincronizadoras obtidas pelos algoritmos para a Série* 2.

Série 3

A Figura 3.7 exibe a Série 3 para n estados. Esta série possui uma palavra sincronizadora mínima com tamanho $n^2 - 3n + 4$ e uma palavra mínima padrão para a série é $(ba^{n-2})^{n-2}ab$. Uma prova de que este tamanho é mínimo para a série foi apresentada por Ananichev [AGV10].

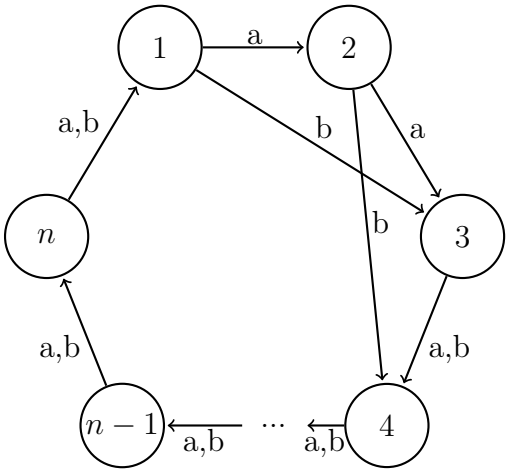

Figura 3.7: *A série infinita* 3.

A Tabela 3.5 apresenta os dados obtidos para autômatos pertencentes a esta série. Note que mais uma vez os valores obtidos pelos algoritmos de Roman coincidem com a palavra mínima.

Estados	Pal. Min.	Ep 1	Ep 2	Ep A 1	Ep A 2	Ep A 3	Roman Φ_1	Roman Φ_2	$(n-1)^2$
6	22	22	22	22	22	22	22	22	25
7	32	32	32	32	32	32	32	32	36
8	44	47	47	44	44	44	44	44	49
9	58	58	61	58	58	58	58	58	64
10	74	80	74	74	74	74	74	74	81
11	92	112	102	92	92	92	92	92	100
12	112	136	134	112	115	123	112	112	121
13	134	160	158	134	134	134	134	134	144
14	158	189	171	158	158	158	158	158	169
15	184	214	212	184	190	193	184	184	196
16	212	250	248	218	212	212	212	212	225
17	242	302	296	265	258	258	242	242	256
18	274	346	348	310	274	280	274	274	289
19	308	396	380	344	333	326	308	308	324
20	344	459	407	385	376	378	344	344	361
21	382	520	468	424	431	437	382	382	400
22	422	594	569	476	444	444	422	422	441
23	464	692	662	538	489	517	464	464	484
24	508	772	761	602	587	579	508	508	529
25	554	829	818	654	651	642	554	554	574
26	602	891	864	707	678	689	602	602	625
27	652	949	950	764	774	766	652	652	676
28	704	1018	947	822	819	805	704	704	729

Tabela 3.5: *Valores das palavras sincronizadoras obtidas pelos algoritmos para a Série* 3.

A Figura 3.8 apresenta graficamente os valores obtidos por estes algoritmos.

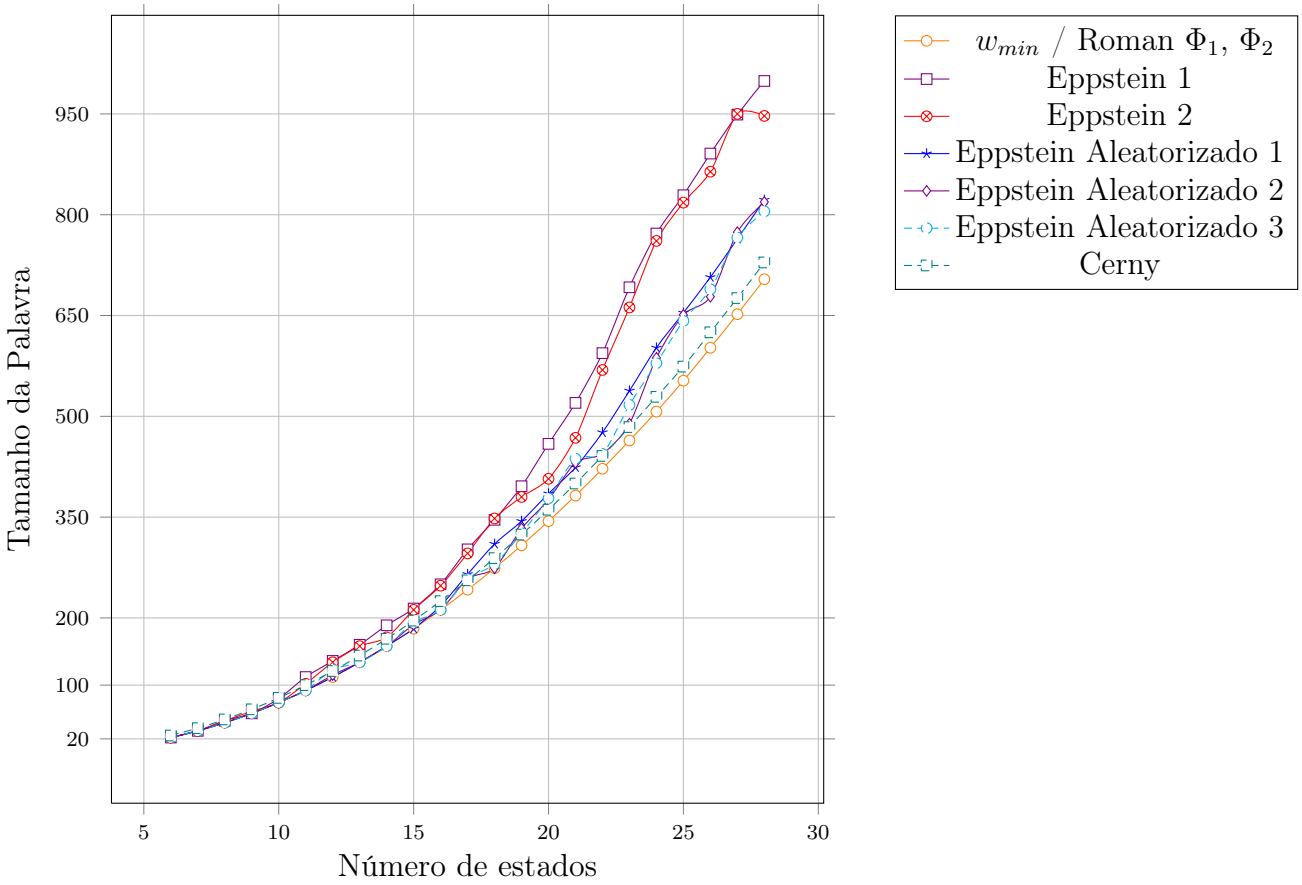

Figura 3.8: *Comparação gráfica dos algoritmos para a Série* 3.

Série 4

Assim como nas séries anteriores, os autômatos desta série possuem n estados, alfabeto $\{a, b\}$ e a letra a atua como permutação cíclica sobre o conjunto de estados. Seja $Q = \{0, \ldots, n-1\}$, as transições com a letra b são:

i. $\delta(0, b) = 2$ e $\delta(1, b) = \left\lfloor \frac{n}{2} \right\rfloor - 1$;

ii. $\delta(q, b) = q + 1 \mod n$, para $q \neq \{0, 1\}$.

A Figura 3.9 exibe o autômato pertencente à Série 4 com 10 estados. A palavra $(ba)^5(ba^7)^3b$ é uma palavra sincronizadora mínima para este autômato e foi encontrada através do algoritmo apresentado na Seção 3.2.

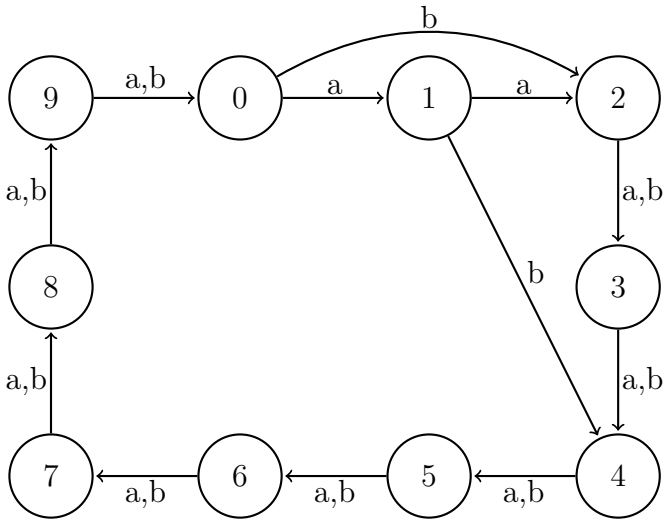

Figura 3.9: *A série infinita* 4.

Os autômatos pertencentes à Série 4 são sincronizados, entretanto, para eles não foi possível obter um formato padrão de palavras sincronizadoras.

Prova: Como visto na Seção 3.3, um autômato é sincronizado se e somente se existem palavras sincronizadoras para cada par de estados. A prova da sincronização dos autômatos da Série 4 é baseada nesta ideia e, portanto, vamos demonstrar que cada par $\{x, y\} \in Q \times Q$ é sincronizado.

Para $x \in \mathbb{Z}$, seja $\bar{x} \equiv x \mod n$. Para $x, y \in \mathbb{Z}_n$, com $x \neq y$, sejam $d\{x, y\} = \{\overline{x-y}, \overline{y-x}\}$ e $C_k = \{\{x, y\} \mid k \in d\{x, y\}\}$. Dessa forma, $\bar{x} \in \{0, 1, \ldots, n-1\}$. E por fim, seja $K = \lfloor \frac{n}{2} \rfloor - 3$.

Primeiro observe que se $\{x, y\} \in Q \times Q$ é um par de estados, então $\{x, y\} = \{y, x\}$.

Segue da definição desta série que:

$$\left\{1, \left\lfloor \frac{n}{2} \right\rfloor - 2\right\} \cdot b = \left\{\left\lfloor \frac{n}{2} \right\rfloor - 1\right\} \tag{3.1}$$

Além disso, para $1 < k \in \mathbb{Z}_n$, por $d\{x, y\}$ e pela definição da série:

$$\begin{aligned} \{0, k\} \cdot b &= \{2, k+1\} \\ d\{0, k\} &= \{-\overline{k}, \overline{k}\} = \{n-k, k\} \\ d\{2, k+1\} &= \{\overline{2-k-1}, \overline{k+1-2}\} = \{\overline{1-k}, \overline{k-1}\} = \{n-k+1, k-1\} \end{aligned} \tag{3.2}$$

Para dois pares de estados $\{x, y\}$ e $\{z, w\}$:

$$d\{x, y\} = d\{z, w\} \;\Leftrightarrow\; \exists\, k : \{x, y\} \cdot a^k = \{z, w\} \tag{3.3}$$

3.3($\Rightarrow$) $d\{x, y\} = d\{z, w\} \;\Rightarrow\; \exists\, k : \{x, y\} \cdot a^k = \{z, w\}$

Sem perda de generalidade, faça $\overline{x-y} = \overline{z-w}$ e $x < z$. Dessa forma, seja $k = z - x$, então $\{z, w\} = \{x, y\} \cdot a^k$ e $\{x, y\} = \{z, w\} \cdot a^{n-k}$.

3.3($\Leftarrow$) $\exists\, k : \{x, y\} \cdot a^k = \{z, w\} \;\Rightarrow\; d\{x, y\} = d\{z, w\}$

Claramente, computando em Z_n, $d(\{x,y\}\cdot a^k) = d(\{x\cdot a^k, y\cdot a^k\}) = \{\overline{x\cdot a^k - y\cdot a^k}, \overline{y\cdot a^k - x\cdot a^k}\} = \{\overline{x-y}, \overline{y-x}\} = d(\{x,y\})$. Uma vez que $\{x,y\}\cdot a^k = \{z,w\}$, então $d\{x,y\} = d\{z,w\}$. Portanto, $d\{x,y\} = d\{z,w\}$.

Agora observe que $\bigcup_{j\geq K} C_j = \{\ \{x,y\} \mid \{x,y\} \in Q\times Q\}$. Seja $\{x,y\} \in Q\times Q$ um par de estados. Vamos mostrar que $\{x,y\} \in \bigcup_{j\geq K} C_j$.

Se $0 < y-x < K$, então $y > x$. Logo, $x-y < 0$ e $\overline{x-y} = x-y+n = n-(y-x)$. Como $y-x < K$, então $\overline{x-y} > n-K$. Por fim, como $K < \frac{n}{2}$, então $\overline{x-y} > K$ e portanto $\{x,y\} \in \bigcup_{j\geq K} C_j$.

Se $j \geq K$ e $\{x,y\} \in C_j$, então $j \in d\{x,y\} = \{n-j, j\}$. Uma vez que $d\{0,j\} = \{n-j,j\}$, $\exists\ k : \{x,y\}\cdot a^k = \{0,j\}$ (por 3.3).

Agora observe que $\{0,j\}\cdot b = \{2, j+1\}$ e $d\{2,j+1\} = \{j-1, n-j+1\}$ (por 3.2). Dessa forma, $\{0,j\}\cdot b \in C_{j-1}$.

Então para $j > K$ é possível ler sequências de a's seguidos por um b e alcançar C_K. Se $\{x,y\} \in C_K$, então $K \in d\{x,y\}$ e portanto $d\{x,y\} = \{n-K, K\}$.

Observe que $d\{1, \lfloor\frac{n}{2}\rfloor - 2\} = \{\overline{-\lfloor\frac{n}{2}\rfloor + 3}, \overline{\lfloor\frac{n}{2}\rfloor - 3}\} = \{\overline{n-(\lfloor\frac{n}{2}\rfloor - 3)}, \overline{\lfloor\frac{n}{2}\rfloor - 3}\} = \{n-K, K\}$. Dessa forma, para $\{x,y\} \in C_K$, $\exists\ k : \{x,y\}\cdot a^k = \{1, \lfloor\frac{n}{2}\rfloor - 2\}$. Por fim, C_K é sincronizado (por 3.1).

□

A Tabela 3.6 apresenta os valores obtidos para autômatos pertencentes a esta série.

Estados	Pal. Min.	Ep 1	Ep 2	Ep A 1	Ep A 2	Ep A 3	Roman Φ_1	Roman Φ_2	$(n-1)^2$
6	21	21	23	21	21	21	21	21	25
7	31	35	31	31	31	31	31	31	36
8	44	47	47	44	44	44	44	44	49
9	58	58	61	58	58	58	58	58	64
10	35	35	35	35	35	35	43	43	81
11	48	49	53	49	49	49	68	57	100
12	42	58	42	42	42	42	62	59	121
13	54	82	58	54	56	54	83	74	144
14	51	63	75	57	51	51	78	78	169
15	63	69	70	69	69	70	103	92	196
16	72	82	100	80	72	78	131	139	225
17	78	99	115	80	82	82	158	131	256
18	99	99	135	99	99	99	145	145	289
19	98	130	148	111	112	109	192	212	324
20	123	147	162	136	134	142	248	248	361
21	136	151	180	136	140	138	280	280	400
22	149	177	209	151	149	149	265	265	441
23	164	262	219	187	187	189	391	407	484
24	138	215	245	175	142	151	264	264	529
25	194	252	273	196	211	218	351	351	576
26	139	185	252	179	145	167	315	315	625

Estados	Pal. Min.	Ep 1	Ep 2	Ep A 1	Ep A 2	Ep A 3	Roman Φ_1	Roman Φ_2	$(n-1)^2$
27	181	307	338	208	220	218	360	360	676
28	161	275	272	209	182	178	439	439	729

Tabela 3.6: *Valores das palavras sincronizadoras obtidas pelos algoritmos para a Série 4.*

Observe que para esta série de autômatos os algoritmos de Roman obtiveram palavras sincronizadoras maiores que as versões do algoritmo de Eppstein para a maior parte dos autômatos testados.

A Figura 3.10 mostra graficamente os valores obtidos pelos algoritmos.

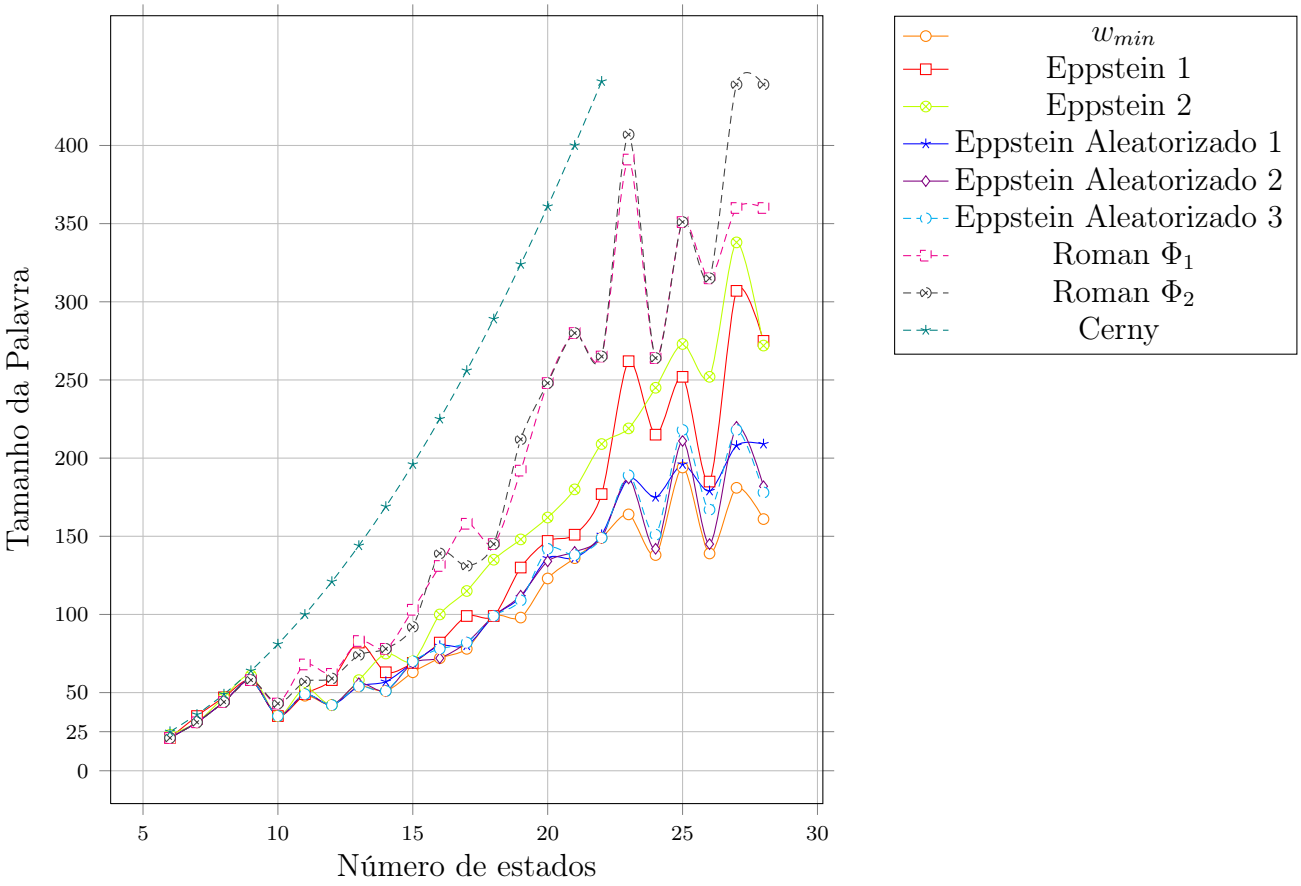

Figura 3.10: *Comparação gráfica dos algoritmos para a Série 4.*

Capítulo 4

Resultados Parciais

Na primeira parte deste capítulo são apresentados resultados parciais obtidos até a presente data em relação à Conjectura de Černý. Em seguida, um destes trabalhos será abordado com mais detalhes.

4.1 Histórico

O caso geral da Conjectura de Černý permanece em aberto, no entanto, diversos resultados parciais foram obtidos. Um dos primeiros trabalhos neste sentido foi apresentado por Pin, que mostrou que a conjectura é válida para autômatos que possuem uma letra que atua como permutação circular sobre o conjunto de estados, com a restrição do número de estados ser primo [Pin78b]. Duas décadas após, Dubuc provou a conjectura para autômatos com esta forma, porém sem a restrição sobre o número de estados [Dub98].

Rystsov utilizou técnicas de álgebra linear para provar um limitante superior de $2(n-1)^2$ no tamanho da palavra sincronizadora de autômatos que possuem uma coleção de palavras tal que para cada par s, t de estados existem exatamente k palavras, nesta coleção, que levam do estado s para o estado t [Rys95].

Béal e Perrin provaram um limitante superior quadrático no tamanho da palavra sincronizadora para autômatos nos quais existe uma letra $a \in \Sigma$ tal que, quando se observa somente as transições com essa letra, existe apenas um ciclo em Q [BP09, BBP11]. Steinberg provou que a conjectura é verdadeira para autômatos deste tipo com ciclo de tamanho primo [Ste11].

Kari provou a conjectura de Černý para autômatos cujo grafo é Euleriano [Kar03]. Kari, Dubuc, e Steinberg fazem extenso uso de técnicas elementares de álgebra linear em seus trabalhos.

Ananichev e Volkov mostraram que todo autômato cujo conjunto de estados admite uma ordenação parcial $\leq$ tal que para cada $\sigma \in \Sigma$ e $q_i \in Q$ a transição $\delta(q_i, \sigma)$ em Q preserva a ordenação parcial (ou seja, $\delta(q_i, \sigma) \leq \delta(q_j, \sigma)$ sempre que $q_i \leq q_j$) possui uma palavra sincronizadora de tamanho no máximo $n - 1$ [AV03, AV05]. Observe que neste caso temos

um limitante linear no tamanho da menor palavra sincronizadora.

Steinberg observou a existência de um padrão nas ideias baseadas no Método de Extensão usando álgebra linear utilizadas em diferentes trabalhos para provar resultados parciais da conjectura e demostrou um lema por meio do qual é possível chegar a resultados semelhantes ou melhores que estes outros autores [Ste10]. Estas ideias são apresentadas em mais detalhes na Seção 4.2.

A maioria dos trabalhos citados nesta seção utiliza em suas demonstrações um dos métodos vistos na Seção 3.1: Fusão ou Extensão.

O Método de Fusão foi usado para provar a conjectura para algumas classes particulares de autômatos. No entanto, Berlinkov apresentou alguns exemplos mostrando que nem sempre é possível limitar por uma função linear o tamanho das palavras concatenadas e, portanto, não existem bons indícios de que o Método de Fusão possa levar a um limitante quadrático no tamanho da menor palavra para o caso geral [Ber10].

O Método de Extensão tem sido, na verdade, o mais adotado nos últimos anos. Rystsov, Dubuc, Kari e Steinberg utilizam em seus trabalhos esta ideia.

4.2 O Método das Probabilidades

Nesta seção será apresentado o método demonstrado pelo Steinberg, através do qual é reunido um conjunto de ideias semelhantes que foram utilizadas por outros autores para provar resultados parciais da conjectura. Através deste método é possível chegar a resultados próximos, melhorar resultados ou mesmo generalizar o trabalho de alguns destes outros autores [Ste10].

Dado um alfabeto Σ, seja $\Sigma^{\leq d} = \bigcup_{m=0}^{d} \Sigma^m$ e $\mathbb{R}\Sigma$ o anel dos polinômios com coeficientes reais nas variáveis não-comutativas de Σ. Uma **Probabilidade** P sobre Σ^* com suporte finito, tal que $P \in \mathbb{R}\Sigma$, é definida por:

$$P = \sum_{w \in \Sigma^*} P(w) \cdot w,$$

onde:

- $P(w) \geq 0$, $\forall\ w \in \Sigma^*$ e
- $\sum_{w \in \Sigma^*} P(w) = 1$.

Além disso, o **suporte** de uma probabilidade é definido por: $\sigma(P) = \{w \in \Sigma^* \mid P(w) > 0\}$. Agora observe que:

i. Se P_1 e P_2 são probabilidades, então P_1P_2 também é uma probabilidade;

ii. $\sigma(P_1P_2) = \sigma(P_1)\sigma(P_2)$[1].

[1] $\sigma(P_1)\sigma(P_2)$ é a concatenação dos suportes das probabilidades P_1 e P_2.

Prova: (i.) Sejam P_1 e P_2 duas probabilidades. Então $P_1 = \sum_{w_1 \in \Sigma^*} P_1(w_1)w_1$ e $P_2 = \sum_{w_2 \in \Sigma^*} P_2(w_2)w_2$. Dessa forma:

$$\begin{aligned} P_1P_2 &= \left(\sum_{w_1 \in \Sigma^*} P_1(w_1)w_1\right)\left(\sum_{w_2 \in \Sigma^*} P_2(w_2)w_2\right) \\ &= \sum_{w_1 \in \Sigma^*}\sum_{w_2 \in \Sigma^*} (P_1(w_1)w_1)(P_2(w_2)w_2) \\ &= \sum_{w_1 \in \Sigma^*}\sum_{w_2 \in \Sigma^*} P_1(w_1)P_2(w_2)w_1w_2 \\ &= \sum_{w_1w_2 \in \Sigma^*} P_{12}(w_1w_2)w_1w_2. \end{aligned}$$

Além disso:

$$\sum_{w_1w_2 \in \Sigma^*} P_{12}(w_1w_2) = \sum_{w_1 \in \Sigma^*}\sum_{w_2 \in \Sigma^*} P_1(w_1)P_2(w_2) = \sum_{w_1 \in \Sigma^*} P_1(w_1) \sum_{w_2 \in \Sigma^*} P_2(w_2) = 1.$$

Portanto, P_1P_2 é uma probabilidade. □

Prova: (ii.) Sejam P_1 e P_2 duas probabilidades e sejam $\sigma(P_1)$ e $\sigma(P_2)$ os suportes destas probabilidades. Então $\forall\ w_1 \in \sigma(P_1)$ e $\forall\ w_2 \in \sigma(P_2) : P_1(w_1) > 0$ e $P_2(w_2) > 0$. Dessa forma, $P(w_1w_2) = P_1(w_1)P_2(w_2) > 0$ e $w_1w_2 \in \sigma(P_1P_2)$. Seja P uma probabilidade tal que $P = P_1P_2$. Temos que $P(w) = \sum P_1(w_1)P_2(w_2)$, onde $w = w_1w_2$. Se $P(w) > 0$, então existem w_1 e w_2 tais que $P_1(w_1) > 0$ e $P_2(w_2) > 0$ e, portanto, $w_1 \in \sigma(P_1)$ e $w_2 \in \sigma(P_2)$. □

Se $S \subseteq Q$, $[S]$ denota o vetor linha característico de S, ou seja, se $i \in S$ então a posição i de $[S]$ tem valor 1, caso contrário, 0. Dessa forma, $[Q]$ tem todas as entradas iguais a 1. De forma similar, os elementos de $\mathbb{R}[Q]$ são tratados como vetores linha. Além disso, seja $\pi(w) : \Sigma^* \to M_n(\mathbb{R})$ a matriz de representação de $w \in \Sigma^*$ com linhas e colunas indexadas por Q tal que: se $i \cdot w = j$ então a posição j da linha i desta matriz tem valor 1. Caso contrário, esta posição tem valor 0. Por simplicidade, iremos nos referir a esta matriz de representação apenas como w.

Seja o vetor $w[S]^t$, onde $[S]^t$ é o vetor transposto de $[S]$. Se $j \in S$, então a posição j de $[S]^t$ é 1. A linha i tem 1 na posição j se e somente se $i \cdot w = j$. Observe que sempre que $j \in S$ e $i \cdot w = j \ \Rightarrow\ j \cdot w^{-1} = i$, o vetor $w[S]^t$ tem 1 na posição i. Desta forma, um fato importante é: $w[S]^t = [Sw^{-1}]^t$.

Por fim, seja $\pi(P) : \mathbb{R}\Sigma \to M_n(\mathbb{R})$. Se $P = \sum_{w \in \Sigma^*} P(w)w$ e $i, j \in Q$, então:

$$\pi(P)_{i,j} = \sum_{\{w \in \Sigma^* | i \cdot w = j\}} P(w)\pi(w)$$

Por exemplo, para o autômato C_4 da Figura 2.3, seja $P = \frac{2}{3}\, a + \frac{1}{3}\, b$. Claramente $P \in \mathbb{R}\Sigma$ e $\sum_{w\in\Sigma^*} P(w) = 1$. Dessa forma, P é uma probabilidade com suporte em Σ e portanto:

$$\pi(P) = \begin{bmatrix} 0 & \frac{2}{3} & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & \frac{2}{3} \\ \frac{2}{3} & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & 0 & 0 & \frac{1}{3} \end{bmatrix}$$

Lema das Probabilidades. *Seja $A = (Q, \Sigma, \delta)$ um autômato sincronizado com n estados e sejam: P_1 uma probabilidade sobre Σ^*, $R \subseteq Q$ e L o tamanho da maior palavra em $\sigma(P_1)$. Agora seja $c \in \{1, 2\}$ tal que se para cada subconjunto próprio não-vazio $S \subset R$, existem $w_1, w_2 \in \sigma(P_1)$ com $Sw_1^{-1} \neq Sw_2^{-1}$, então $c = 2$; caso contrário, $c = 1$. Suponha que exista uma probabilidade P_2 com suporte em $\Sigma^{\leq n-c}$ tal que:*

1. *$[R]\ \pi(P_2)\ \pi(P_1) = [R]$;*
2. *$R \subseteq q \cdot \Sigma^*$, $\forall\ q \in R$. Ou seja, R é fortemente conexo;*
3. *$\exists\ w_0 \in \Sigma^*$: $Q \cdot w_0 \subseteq R$.*

Então a palavra mínima tem tamanho no máximo:

- *$(n-2)(n-c+L)+c$, se $R = Q$;*
- *$(|R|-1)(n-c+L)+|w_0|+c-1$, se $R \subset Q$.*

A seguir vamos mostrar como utilizar o lema para provar alguns resultados e na Subseção 4.2.1 apresentamos a prova do Lema das Probabilidades.

Autômatos Pseudo-Eulerianos

Através do Lema das Probabilidades Steinberg mostra como generalizar o trabalho de Kari sobre autômatos Eulerianos (autômatos cujo grafo é Euleriano) e prova que a conjectura é verdadeira para autômatos pseudo-eulerianos. O limitante para o tamanho da menor palavra apresentado por Kari para esta classe de autômatos tem valor: $(n-1)(n-2)+1$.

Um autômato é **pseudo-euleriano** se for possível encontrar uma probabilidade P com suporte em Σ tal que $\pi(P)$ é uma matriz duplamente estocástica. Uma matriz é duplamente estocástica se todas as linhas e todas as colunas somam 1.

Seja A um autômato Euleriano e M_A a matriz de adjacência de A. Seja

$$P = \sum_{\alpha\in\Sigma} \frac{1}{|\Sigma|}\alpha.$$

Claramente $\pi(P) = \frac{1}{|\Sigma|}M_A$ e $\pi(P)$ é uma matriz duplamente estocástica. Além disso, P tem suporte em Σ. Portanto, todo autômato Euleriano é pseudo-Euleriano.

Dado um autômato fortemente conexo, o seguinte sistema de equações lineares torna possível verificar se ele é pseudo-Euleriano:

$$\begin{aligned} \sum_{\alpha\in\Sigma} p_\alpha &= 1, \\ \sum_{\alpha\in\Sigma} p_\alpha \ |q\cdot\alpha^{-1}| &= 1, \ \forall\ q\in Q. \end{aligned}$$

Para o qual $p_\alpha \geq 0, \forall\ \alpha\in\Sigma$.

Então, se for possível encontrar uma solução não-nula para o sistema, o autômato é pseudo-Euleriano.

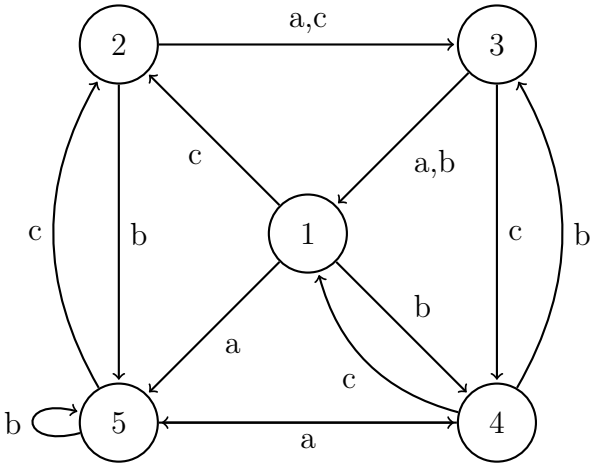

Figura 4.1: *Um autômato pseudo-Euleriano.*

O sistema de equações para o autômato da Figura 4.1 é:

$$\begin{aligned} p_a + p_b + p_c &= 1 \ (\text{ para } q\in\{1,3,4\}) \\ 2\ p_c &= 1 \ (\text{ para } q = 2) \\ 2\ p_a + 2\ p_b &= 1 \ (\text{ para } q = 5) \end{aligned}$$

Uma possível solução para este sistema é $p_a = \frac{1}{6}$, $p_b = \frac{1}{3}$ e $p_c = \frac{1}{2}$. Dessa forma, $P = \frac{1}{6}\ a + \frac{1}{3}\ b + \frac{1}{2}\ c$ é uma probabilidade sobre Σ. Por fim

$$\pi(P) = \begin{bmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{3} & \frac{1}{6} \\ 0 & 0 & \frac{2}{3} & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{3} & 0 & \frac{1}{6} \\ 0 & \frac{1}{2} & 0 & \frac{1}{6} & \frac{1}{3} \end{bmatrix}$$

Claramente $\pi(P)$ é uma matriz duplamente estocástica. Dessa forma, o autômato da Figura 4.1 é pseudo-Euleriano.

Então seja $A = (Q, \Sigma, \delta)$ um autômato sincronizado com n estados e pseudo-Euleriano. Como este autômato é pseudo-Euleriano, existe uma probabilidade P com suporte em Σ tal que $\pi(P)$ é duplamente estocástica. Seja P_1 uma probabilidade cuja única palavra no suporte é a palavra vazia. Dessa forma, $\nexists\, w_1, w_2 \in \sigma(P_1)$ tais que $S \cdot w_1^{-1} \neq S \cdot w_2^{-1}$ e portanto $c = 1$. Agora seja $R = Q$.

Uma vez que os autômatos são fortemente conexos, é fácil perceber que a condição 2 do lema é satisfeita. A palavra vazia satisfaz a condição 3 do lema. Como $c = 1$, a probabilidade P_2 deve ser uma probabilidade sobre $\Sigma^{\leq n-1}$. Então seja

$$P_2 = \frac{1}{n}(P^0 + P^1 + \cdots + P^{n-1}) = \frac{1}{n}\sum_{k=0}^{n-1} P^k,$$

onde, $P^k = (\sum_{\alpha\in\Sigma} P(\alpha)\ \alpha)^k$ é uma probabilidade com suporte em Σ^k. Vamos provar por indução em k que $P_k = \sum_{w\in\Sigma^k}$ é uma probabilidade sobre Σ^k. Se $k = 0$, então

$$P^0 = (\sum_{\alpha\in\Sigma} P(\alpha)\cdot\alpha)^0 = 1.$$

Logo, P^0 é uma probabilidade com suporte em Σ^0. Se $k > 0$, então $P^k = P^{k-1}P$. Como P^{k-1} e P são probabilidades, então P^k é uma probabilidade. Dessa forma

$$\begin{aligned} P^k &= (\sum_{w\in\Sigma^{k-1}} P(w)\cdot w)(\sum_{w\in\sigma(P)} P(w)\cdot w) \\ &= (\sum_{w\in\Sigma^{k-1}} P(w)\cdot w)(\sum_{\alpha\in\Sigma} P(\alpha)\cdot\alpha) \\ &= \sum_{w\in\Sigma^{k-1}}\sum_{\alpha\in\Sigma} P(w)P(\alpha)\cdot w\alpha \\ &= \sum_{w\in\Sigma^k} P(w)\cdot w. \end{aligned}$$

Logo, $P_2 = \frac{1}{n}\sum_{k=0}^{n-1} P^k$ é uma probabilidade com suporte em $\Sigma^{\leq n-1}$. Agora como $\pi(P)$ é uma matriz duplamente estocástica e o produto de matrizes duplamente estocásticas resulta em uma outra matriz duplamente estocástica, então $\sum_{m=0}^{n-1}\pi(P^m)$ resulta em uma matriz com linhas e colunas somando n. Por fim, multiplicando por $\frac{1}{n}$ é possível obter novamente uma matriz duplamente estocástica.

Dessa forma, P_2 é uma probabilidade com suporte em Σ^{n-1} e $\pi(P_2)$ é uma matriz duplamente estocástica. Logo, $[Q]\pi(P_2) = [Q] \Rightarrow [Q]\pi(P_2)\pi(P_1) = [Q]\pi(P_1) = [Q]$ e a condição 1 do lema é válida.

Portanto, um autômato sincronizado com n estados pseudo-Euleriano tem uma palavra sincronizadora de tamanho no máximo $(n-2)(n-1)+1$.

Autômatos Regulares

Rystsov chama um autômato de regular se existe uma coleção de palavras $W = \{w_1, \ldots, w_m\}$ com as seguintes propriedades:

- Para cada par de estados $q, s \in Q$, com $|Q| = n$, existem exatamente k palavras em W, para $k \in \mathbb{N}$, que levam o estado q ao estado s;
- $\lambda \in W$;
- O tamanho da maior palavra em W é $n - 1$.

Rystsov mostrou que autômatos regulares têm uma palavra sincronizadora de tamanho no máximo $2\ (n-1)^2$.

Seja $q, s \in Q$. Agora seja $W_{qs} = \{w \mid q \cdot w = s\}$. Claramente, $|W_{qs}| = k$ e como o autômato é determinístico para $s_1, s_2 \in Q$, $W_{qs_1} \cap W_{qs_2} = \emptyset$.

Seja $W_q = \bigcup_{s \in Q} W_{qs}$, logo $|W_q| = kn$. Agora suponha que $W_q \subset W$ Então existe $w \in W \backslash W_q$. Como $w \in W$ e $Q \cdot W = Q$, então, $q \cdot w = s \in Q$. Contradição, pois haveriam $k+1$ palavras levando q a s. Portanto, $W_q = W$ e $|W| = kn$.

Seja A um autômato regular. Então A possui um conjunto de palavras W, contendo kn palavras tais que $Q \cdot W = Q$. Além disso, $\lambda \in W$, e o tamanho da maior palavra em W é $n - 1$. Seja P_1 uma distribuição uniforme sobre as palavras em W, ou seja, para $w \in W$, $P_1(w) = \frac{1}{|W|} = \frac{1}{kn}$. Se $w \notin W$, então $P_1(w) = 0$. Logo, $\sigma(P_1) = W$.

Seja $S \subset Q$, tal que $S \neq \emptyset$. Sejam $s_1 \in S$ e $s_2 \in Q \backslash S$. Sejam $w_1, w_2 \in W$ tais que, para $q \in Q$, $q \cdot w_1 = s_1 \Rightarrow q = s_1 \cdot w_1^{-1}$ e $q \cdot w_2 = s_2 \Rightarrow q = s_2 \cdot w_2^{-1}$. Note que $q \in S \cdot w_1^{-1}$ e $q \notin S \cdot w_2^{-1}$, uma vez que $s_2 \notin S$. Dessa forma, $c = 2$.

Agora seja P_2 uma probabilidade com suporte em $\Sigma^{\leq n-2}$. Sabemos que $\sum_{w \in \Sigma^*} P_2(w) = 1$ (pela definição de uma probabilidade) e cada matriz de representação da ação das palavras em $\sigma(P_2)$ é estocástica (visto que o autômato é determinístico). Então $\pi(P_2)$ é uma matriz estocástica. Para $q \in Q$, a coluna q de $\pi(P_1)$ é $\frac{k}{|W|}[Q]^t = \frac{k}{kn}[Q]^t = \frac{1}{n}[Q]^t$, pois $\forall\ q, s \in Q$, $\exists\ W_{qs} \subset W$ tal que $q \cdot W_{qs} = s$ e $|W_{qs}| = k$. Então, como $\pi(P_2)$ é uma matriz estocástica: $\pi(P_2)\pi(P_1) = \pi(P_1)$. Portanto $[Q]\pi(P_2)\pi(P_1) = [Q]\pi(P_1)$.

Agora note que como cada coluna de $\pi(P_1)$ é $\frac{1}{n}[Q]^t$, cada entrada do vetor $[Q]\pi(P_1)$ é $[Q]\frac{1}{n}[Q]^t = \sum_{i=1}^{n} 1 \cdot \frac{1}{n} = n \cdot \frac{1}{n} = 1$. Logo $[Q]\pi(P_1) = [Q]$, então $[Q]\pi(P_2)\pi(P_1) = [Q]$ e a condição 1 do lema está satisfeita. Como o autômato é fortemente conexo, $Q \subseteq q \cdot \Sigma^*$, $\forall\ q \in Q$ (condição 2 do lema). Por fim, $w_0 = \lambda$ satisfaz a condição 3 do lema. Então, A tem uma palavra sincronizadora de tamanho no máximo $(n-2)(n-2+n-1)+2 = (n-2)(2n-3)+2 = 2n^2 - 7n + 8 < 2n^2 - 4n + 2$.

Portanto, o limitante obtido através do lema é inferior ao limitante do Rystsov, o que resulta em uma melhoria em relação a este resultado.

Autômatos 1-ciclo

Um autômato pertence à classe 1-ciclo se existe $\alpha \in \Sigma$ tal que para todo par de estados $q_1, q_2 \in Q$, $\exists\, i, j \in \mathbb{N}$, tal que $q_1 \cdot \alpha^i = q_2 \cdot \alpha^j$. Em outras palavras, para $\alpha \in \Sigma$, existe apenas um ciclo R em Q rotulado por α. A Figura 4.2 apresenta a estrutura de um autômato 1-ciclo.

Em 2011 foi provado o limitante superior $2n^2 - 7n + 7$ para esta classe de autômatos [BBP11], uma melhora em relação ao limitante apresentado em 2009 no valor $2n^2 - 6n + 5$ [BP09].

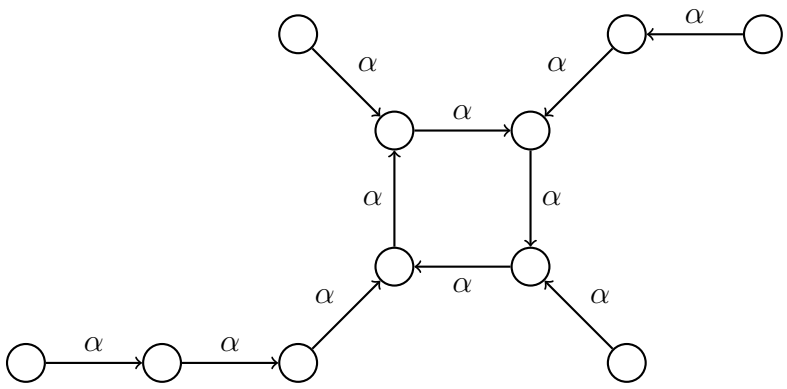

Figura 4.2: *Um autômato* 1*-ciclo no qual são exibidas somente as transições com o símbolo* α*.*

Seja A um autômato 1-ciclo e seja $R \subseteq Q$ o conjunto de estados que compõe o ciclo único rotulado por α. Seja $r = |R|$ e $W = \{\alpha^{n-r}, \ldots, \alpha^{n-1}\}$, claramente, $|W| = r$. Note que $w_0 = \alpha^{n-r}$ leva qualquer estado de Q a R e $|w_0| = n - r$. Dessa forma, a condição 3 do lema está satisfeita. Segue de forma direta da definição da classe 1-ciclo que a condição 2 do lema é válida.

Agora seja P_1 uma distribuição uniforme sobre W. Se $w \in W$, $P_1(w) = \frac{1}{|W|} = \frac{1}{r}$, caso contrário, $P_1(w) = 0$. Claramente $L = n - 1$. Seja $\emptyset \neq S \subset R$. Suponha que $s_1 \in S$ e $s_2 \in R \backslash S$. Sejam $w_1, w_2 \in W$ tais que $q \cdot w_1 = s_1 \Rightarrow q = s_1 \cdot w_1^{-1}$ e $q \cdot w_2 = s_2 \Rightarrow q = s_2 \cdot w_2^{-1}$. Dessa forma, $c = 2$. Agora seja P_2 uma probabilidade com suporte em $\Sigma^{\leq n-2}$. Pela mesma razão que no caso dos Autômatos Regulares, $\pi(P_2)$ é claramente uma matriz estocástica. Em $\pi(P_1)$ as colunas correspondentes aos estados de $Q \backslash R$ são formadas por 0, pois as palavras em W são palavras que levam de Q a R. Como para cada estado em Q existe uma palavra em W que o leva a cada estado de R, então, se $s \in R$, a coluna que representa s em $\pi(P_1)$ tem valor: $\frac{1}{|W|}[Q]^t = \frac{1}{r}[Q]^t$. Dessa forma, como $\pi(P_2)$ é estocástica, $\pi(P_2)\pi(P_1) = \pi(P_1)$.

Para $i \in \{1, \ldots, n\}$, seja a_i uma entrada do vetor $[R]$. Agora observe que cada entrada v_i do vetor $[R]\pi(P_1)$ tem valor:

- $v_i = 0$, se $i \notin R$;
- $v_i = \sum_{i=1}^{n} a_i \cdot \frac{1}{r}$. Como $a_i = 1$ somente quando $i \in R$, então $v_i = r \cdot \frac{1}{r} = 1$

Dessa forma, $[R]\pi(P_1) = [R]$ e portanto $[R]\pi(P_2)\pi(P_1) = [R]$. Então, pelo Lema das Probabilidades, esta classe de autômatos tem uma palavra sincronizadora de tamanho no máximo:

- Se $R \subset Q \Rightarrow r \leq n-1$: $(r-1)(n-2+n-1)+(n-r)+2-1 \leq (n-2)(2n-3)+(n-(n-1))+1 = 2n^2-7n+8$;
- Se $R = Q \Rightarrow r = n$: $(n-2)(n-2+n-1)+2 = 2n^2-7n+8$.

Limitante para a Série 4

Como visto na Seção 3.6, os autômatos da Série 4 são sincronizados. Além disso, são claramente fortemente conexos. Contudo, não foi possível apresentar uma palavra com formato padrão devido à falta de regularidade nestes autômatos. Note que uma palavra mínima para o autômato desta série com 10 estados tem tamanho inferior a uma palavra mínima para o autômato com 9 estados.

Através do Lema das Probabilidades é possível mostrar um limitante quadrático para o tamanho da menor palavra sincronizadora para estes autômatos, porém, como visto na Tabela 3.6, na prática para os autômatos testados estas palavras tem tamanho bastante inferior ao limitante aqui apresentado.

Seja A um autômato pertencente à Série 4 com $Q = \{0, \ldots, n-1\}$. Uma vez que a letra a atua como permutação cíclica sobre o conjunto de estados, as palavras em $W = \{\lambda, a, \ldots, a^{n-1}\}$ satisfazem: $Q \cdot W = Q$. Claramente, $|W| = n$.

Seja $R = Q$. Note que $w_0 = \lambda$ satisfaz a condição 3 do lema. Como estes autômatos são fortemente conexos, segue de forma direta que a condição 2 do lema é válida.

Agora seja P_1 uma distribuição uniforme sobre W. Se $w \in W$, $P_1(w) = \frac{1}{|W|} = \frac{1}{n}$, caso contrário, $P_1(w) = 0$. Claramente, $\sigma(P_1) = W$ e $L = n-1$.

Sejam $w_1, w_2 \in W$ tais que $w_1 = a^2$ e $w_2 = a$. Como a atua como uma permutação sobre o conjunto de estados, note que se $x \in S \subset Q$, então $x \cdot w_1^{-1} = (x-2) \bmod n$, $x \cdot w_2^{-1} = (x-1) \bmod n$ e esta propriedade é válida para todo subconjunto próprio não vazio de Q. Então $S \cdot w_1^{-1} \neq S \cdot w_2^{-1}$ e dessa forma, $c = 2$. Seja P_2 uma probabilidade com suporte em $\Sigma^{\leq n-2}$. Pela mesma razão que nos casos anteriores, $\pi(P_2)$ é uma matriz estocástica.

Para $q \in Q$, a coluna q de $\pi(P_1)$ é $\frac{1}{|W|}[Q]^t = \frac{1}{n}[Q]^t$. Então, como $\pi(P_2)$ é uma matriz estocástica: $\pi(P_2)\pi(P_1) = \pi(P_1)$. Portanto $[Q]\pi(P_2)\pi(P_1) = [Q]\pi(P_1)$.

Agora note que como cada coluna de $\pi(P_1)$ é $\frac{1}{n}[Q]^t$, então, cada entrada do vetor $[Q]\pi(P_1)$ é $[Q]\frac{1}{n}[Q]^t = \sum_{i=1}^{n} 1 \cdot \frac{1}{n} = n \cdot \frac{1}{n} = 1$. Logo $[Q]\pi(P_1) = [Q]$, então $[Q]\pi(P_2)\pi(P_1) = [Q]$ e a condição 1 do lema está satisfeita. Então, os autômatos da Série 4 têm uma palavra sincronizadora de tamanho no máximo $(n-2)(n-2+n-1)+2 = (n-2)(2n-3)+2 = 2n^2-7n+8$.

4.2.1 Prova do Lema das Probabilidades

Seja $Z : \Sigma^* \to \mathbb{R}$ uma variável aleatória. A Esperança de Z em relação a uma probabilidade P é:

$$E_p(Z) = \sum_{w \in \Sigma^*} P(w)Z(w) = \sum_{w \in \sigma(P)} P(w)Z(w).$$

Segue, a partir da Esperança e da definição de Probabilidade, que:

- Ou $Z(w) = E_p(Z)$, $\forall\ w \in \sigma(P)$;
- Ou $\exists\ w \in \sigma(P)$ com $Z(w) > E_p(Z)$.

Agora seja $X \subseteq \Sigma^*$ e W um subespaço vetorial de $\mathbb{R}^n$. Então XW é o conjunto gerador de todos os vetores xw com $x \in X$ (a matriz de representação de uma palavra) e $w \in W$.

Fato 4.1. *Seja $\pi(w) : \Sigma^* \to M_n(\mathbb{R})$[2] uma matriz de representação para $w \in \Sigma^*$. Sejam $W, V \subseteq \mathbb{R}^n$ vetores coluna com $W \subseteq V$ e $\Sigma^* W \nsubseteq V$. Seja S um conjunto gerador de W. Então, existe $s \in S$ e $w \in \Sigma^*$ com $|w| \leq dim\ V - dim\ W + 1$ e $ws \notin V$.*

Prova: Seja $W_m = \Sigma^{\leq m} W$. Vamos mostrar que a cadeia de subespaços:

$$W = I_n W = \lambda W = \Sigma^{\leq 0} W = W_0 \subseteq W_1 \subseteq W_2 \subseteq \cdots$$

onde In é a matriz identidade, se estabiliza para algum $e \geq 0$ tornando-se equivalente a $\Sigma^* W$.
Uma vez que, para algum $i \geq 0$, $W_i = W_{i+1}$ a cadeia se estabiliza e não se altera mais. Suponha que $W_i = W_{i+1} = \cdots = W_{i+k-1} \subsetneq W_{i+k}$, para $k \in \mathbb{N}$, com $i + k \geq e$. Então $\exists\ w \in \Sigma^{i+k}$ e $\exists\ t_1 \in W$: $w\ t_1 \notin W_{i+1}$.
Considere $a \in \Sigma$ o primeiro símbolo de w. Então $w = av$, $v \in \Sigma^{i+k-1}$ e $w\ t_1 = av\ t_1$. Faça $t_2 = v\ t_1$; evidentemente $t_2 \in W_{i+k-1} \Rightarrow t_2 \in W_i$. Então, $\exists\ u \in \Sigma^{\leq i}$ e $t_3 \in W_i$: $u\ t_3 = t_2 \in W_i$. Dessa forma $w\ t_1 = av\ t_1 = a\ t_2 = au\ t_3 \in W_{i+1}$. Mas $au \in \Sigma^{\leq i+1}$, logo $w\ t_1 = au\ t_3 \in W_{i+1}$. O que leva a uma contradição.
Perceba que $W_e = \Sigma^* W \nsubseteq V$. Considere m o menor inteiro tal que $W_m \nsubseteq V$. Claramente $m \leq e$ e $W_{m-1} \subseteq V$. Dessa forma:

$$W_0 \subset W_1 \subset \cdots \subset W_{m-1} \subseteq V.$$

Portanto, $dim\ W_0 + (m-1) \leq dim\ V \Rightarrow m \leq dim\ V - dim\ W + 1$. Então, como $W_m \nsubseteq V$, $\exists\ w \in \Sigma^{\leq dim\ V - dim\ W + 1}$ com $wW \nsubseteq V$. Claramente, $|w| \leq dim\ V - dim\ W + 1$. Como W é gerado por S, então $\exists\ s \in S$ com $ws \notin V$. □

[2]Novamente, por simplicidade, usaremos apenas w para representar $\pi(w)$.

Fato 4.2. *Sejam $X = \sigma(P_1)$, $R \subseteq Q$ e $c \in \{1, 2\}$. Para todo $\emptyset \neq S \subset R$, existe $w \in \Sigma^*$ de tamanho máximo $n - c + L$ com $|S \cdot w^{-1} \cap R| > |S|$; exceto quando $c = 2$ e $|S| = r/2$ (onde $r = |R|$), neste caso, $|w| \leq n - 1 + L$.*

Prova: Seja $P = P_2P_1$ uma probabilidade em Σ^* e seja $Z_S : \Sigma^* \to \mathbb{R}$ uma variável aleatória definida por: $Z_S(w) = |S \cdot w^{-1} \cap R|$. Observe que $|S \cdot w^{-1} \cap R| = [R][S \cdot w^{-1}]^t$ e $[S \cdot w^{-1}]^t = w[S]^t$.

Logo, $Z_S(w) = [R]w[S]^t$. Dessa forma, a Esperança de Z_S é:

$$E_p(Z_S) = \sum_{w \in \Sigma^*} P(w)Z_S(w) = \sum_{w \in \Sigma^*} P(w)[R]w[S]^t = [R] \left(\sum_{w \in \Sigma^*} P(w)w \right) [S]^t = [R]P[S]^t.$$

Dessa forma, se $[R]\pi(P_2)\pi(P_1) = [R]$, então $E_p(Z_S) = [R][S]^t$. Além disso, como $S \subset R$, temos $E_p(Z_S) = |S|$. Uma vez que $P = P_2P_1$, então $\sigma(P) = \sigma(P_2)\sigma(P_1) \subseteq \Sigma^{\leq n-c}X$.

Logo, se for possível encontrar $v \in \Sigma^{\leq n-c}X$ com $Z_S(v) = |S \cdot v^{-1} \cap R| \neq |S|$, então $\exists\, w \in \Sigma^{\leq n-c}X$ com $Z_S(w) = |S \cdot w^{-1} \cap R| > |S|$. Como L é o comprimento da maior palavra em $\sigma(P_1)$, então $|w| \leq n - c + L$.

Se existe $x \in X$ tal que $|S \cdot x^{-1} \cap R| \neq |S|$, então existe $x' \in \Sigma^{\leq L}$ tal que $|S \cdot x'^{-1} \cap R| > |S|$ e temos uma palavra $w = x'$ que satisfaz $|S \cdot w^{-1} \cap R| > |S|$, como queríamos demonstrar.

Caso contrário, $|S \cdot x^{-1} \cap R| = |S|$. Então seja $\gamma = [S]^t - \frac{|S|}{r}[Q]^t$ um vetor coluna. Se $w \in \Sigma^*$, então:

$$w\gamma = w[S]^t - \frac{|S|}{r}w[Q]^t = w[S]^t - \frac{|S|}{r}[Q \cdot w^{-1}]^t = w[S]^t - \frac{|S|}{r}[Q]^t.$$

Portanto, uma vez que $r = |R|$:

$$[R]w\gamma = [R]w[S]^t - \frac{|S|}{r}[R][Q]^t = |S \cdot w^{-1} \cap R| - \frac{|S|}{r}|R| = |S \cdot w^{-1} \cap R| - |S|.$$

Uma vez que $|S \cdot x^{-1} \cap R| = |S|$, $\forall\, x \in X$: $[R]x\gamma = |S \cdot x^{-1} \cap R| - |S| = |S| - |S| = 0$. Como $|S| < r$, $x\gamma = [Sx^{-1}]^t - \frac{|S|}{r}[Q]^t \neq 0$. Dessa forma, como $[R]x\gamma = 0$ ($[R]$ e $x\gamma$ são ortogonais), se W é o subespaço gerado pelos vetores coluna $x\gamma$ e $[R]^\perp$ é o subespaço gerado pelos vetores ortogonais a $[R]$, então $0 \neq W \subseteq [R]^\perp$.

Agora precisamos verificar que $dim\ W \geq c$, exceto quando $c = 2$ e $|S| = \frac{r}{2}$.

Quando $c = 1$ não existem $w_1, w_2 \in X$ tais que $S \cdot w_1^{-1} \neq S \cdot w_2^{-1}$. Ou seja, $\forall\, w_1, w_2 \in X$, $S \cdot w_1^{-1} = S \cdot w_2^{-1}$. Além disso, como visto, $w_1\gamma \neq 0$ e $w_2\gamma \neq 0$.

Então $[S \cdot w_1^{-1}]^t - \frac{|S|}{r}[Q]^t \neq 0$ e $[S \cdot w_2^{-1}]^t - \frac{|S|}{r}[Q]^t \neq 0$, mas $S \cdot w_1^{-1} = S \cdot w_2^{-1}$.

Dessa forma:

$$[S \cdot w_1^{-1}]^t - \frac{|S|}{r}[Q]^t = [S \cdot w_2^{-1}]^t - \frac{|S|}{r}[Q]^t,\ \forall\, w_1, w_2 \in X.$$

Como W é o espaço gerado pelos vetores coluna $x\gamma$, com $x \in X$, então $dim\ W \geq 1$. Logo, $dim\ W \geq c$.

Quando $c = 2$ e $|S| \neq \frac{r}{2}$ existem w_1, $w_2 \in X$ tais que $S \cdot w_1^{-1} \neq S \cdot w_2^{-1}$. Se $S \cdot w_1^{-1} \neq S \cdot w_2^{-1}$, então $w_1\gamma \neq w_2\gamma$. Agora suponha, por contradição, que $w_1\gamma$ e $w_2\gamma$ são linearmente dependentes.

Então $w_1\gamma = k\ (w_2\gamma)$, para $1 \neq k \in \mathbb{R}$ e:

$$[S \cdot w_1^{-1}]^t - \frac{|S|}{r}[Q]^t = k\ [S \cdot w_2^{-1}]^t - k\ \frac{|S|}{r}[Q]^t \Rightarrow [S \cdot w_1^{-1}]^t - k\ [S \cdot w_2^{-1}]^t = (1-k)\ \frac{|S|}{r}[Q]^t.$$

Como $[S \cdot w_1^{-1}]^t$ e $[S \cdot w_2^{-1}]^t$ são vetores coluna com entradas 0 ou 1 e $[Q]^t$ é um vetor coluna com todas as entradas iguais a 1, então, a seguir são apresentadas as possibilidades para esta igualdade:

1) $1 - k\ (1) = (1-k)\frac{|S|}{r} \Rightarrow \frac{1-k}{1-k} = \frac{|S|}{r} \Rightarrow \frac{|S|}{r} = 1 \Rightarrow |S| = r$. Mas $S \subset R$ e $r = |R|$. Contradição;

2) $0 - k\ (0) = (1-k)\frac{|S|}{r} \Rightarrow \frac{|S|}{r} = 0 \Rightarrow |S| = 0$. Mas $S \neq \emptyset$. Contradição;

3) $1 - k\ (0) = (1-k)\frac{|S|}{r} \Rightarrow 1 = (1-k)\ \frac{|S|}{r} \Rightarrow r = (1-k)\ |S| \Rightarrow |S| = \frac{r}{1-k} \Rightarrow k \neq 1$;

4) $0 - k\ (1) = (1-k)\frac{|S|}{r} \Rightarrow \frac{|S|}{r} = \frac{-k}{1-k}$. Mas, $|S| = \frac{r}{1-k}$, então $\frac{r}{r(1-k)} = \frac{-k}{1-k} \Rightarrow -k(1-k) = 1 - k \Rightarrow -k + k^2 = 1 - k \Rightarrow k^2 = 1 \Rightarrow k = -1$, pois $k \neq 1$.

Os casos possíveis, 3 e 4, mostram que $S \cdot w_1^{-1}$ e $S \cdot w_2^{-1}$ são complementares. Dessa forma: $[S \cdot w_1^{-1}]^t + [S \cdot w_2^{-1}]^t = [Q]^t$. Além disso, como $k = -1$ temos:

$$[S \cdot w_1^{-1}]^t - (-1)[S \cdot w_2^{-1}]^t = (1 - (-1))\frac{|S|}{r}[Q]^t$$

$$[S \cdot w_1^{-1}]^t + [S \cdot w_2^{-1}]^t = 2\frac{|S|}{r}[Q]^t$$

E uma vez que $[S \cdot w_1^{-1}]^t + [S \cdot w_2^{-1}]^t = [Q]^t$:

$$[Q]^t = 2\frac{|S|}{r}[Q]^t$$

$$2|S| = r$$

$$|S| = \frac{r}{2}$$

Contradição.

Então, $w_1\gamma$ e $w_2\gamma$ são linearmente independentes e $dim\ W \geq 2 = c$.

Quando $c = 2$ e $|S| = \frac{r}{2}$ existem w_1, $w_2 \in X$ tais que $S \cdot w_1^{-1} \neq S \cdot w_2^{-1}$, porém $w_1\gamma$ e $w_2\gamma$ são linearmente dependentes. Portanto, $dim\ W \geq 1$.

Agora precisamos mostrar apenas que $\Sigma^* W \nsubseteq [R]^{\perp}$.

Primeiro observe que se w é uma palavra sincronizadora para o autômato A, então ww_0 também é e o sincroniza em um estado $q \in R$. Como $q\Sigma^* \supseteq R$, é possível sincronizar A em qualquer estado de R. Em particular, é possível sincronizar o autômato através de uma

palavra y em algum estado dentro de $S \cdot x^{-1} \cap R$, para $x \in X$. Dessa forma, $S \cdot x^{-1}y^{-1} = Q$ e $[R]\ yx\gamma = |S \cdot x^{-1}y^{-1} \cap R| - |S| > 0$. Isto mostra que $[R]\ yx\gamma \neq 0$, logo, $yx\gamma \nsubseteq [R]^{\perp}$.

Uma vez que W é gerado pelos vetores coluna $x\gamma$, com $x \in X$, então, $\Sigma^* W \nsubseteq [R]^{\perp}$.

Além disso, como $dim\ W \geq c$ e $dim\ [R]^{\perp} = n - 1$, pelo Fato 4.1 existe $s \in X\gamma$ e $u \in \Sigma^*$ tal que $us \notin [R]^{\perp}$ e $|u| \leq dim\ [R]^{\perp} - dim\ W + 1 = n - 1 - c + 1 = n - c$.

Logo, $u \in \Sigma^{\leq n-c}$. Seja $s = x_1\gamma$, com $x_1 \in X$. Então, $ux_1\gamma \notin [R]^{\perp}$ e portanto $[R]\ ux_1\gamma \neq 0$.

Agora seja $v = ux_1 \in \Sigma^{\leq n-c}X$. Dessa forma, $[R]v\gamma \neq 0 \Rightarrow |S \cdot v^{-1} \cap R| - |S| > 0 \Rightarrow |S \cdot v^{-1} \cap R| > |S|$ e $|v| \leq n - c + L$, pois L é o tamanho da maior palavra em X.

Então para $R = Q$, como o autômato é sincronizado é possível achar um estado $q \in Q$ e uma letra $\alpha \in \Sigma$ tal que $|q \cdot \alpha^{-1}| > 1$.

Usando o Método de Extensão é possível expandir essas imagens inversas através de palavras com tamanho no máximo $n - c + L$; exceto quando $c = 2$ e $|S| = r/2$, neste caso, este tamanho é $n - 1 + L$. Em no máximo $(n - 2)$ expansões o conjunto de estados se iguala a Q.

Para $R \subset Q$, se $\exists\ w_0$ tal que $Q \cdot w_0 \subseteq R$ então $|Q \cdot w_0| \leq |R|$. Como o autômato é sincronizado é possível encontrar uma palavra $w \in \Sigma^*$ com tamanho no máximo $(r-1)(n - c + L) + c - 1$ tal que $|R \cdot w^{-1}| = 1$ usando a mesma ideia do caso anterior.

Então $|Q \cdot w\ w_0| \leq |R \cdot w| = 1$ e o tamanho da menor palavra sincronizadora neste caso é $(r - 1)(n - c + L) + |w_0| + c - 1$. □

Capítulo 5

Considerações Finais

Černý conjecturou que um autômato sincronizado com n estados possui uma palavra sincronizadora de tamanho no máximo $(n-1)^2$. O caso geral da conjectura permanece em aberto, contudo, resultados parciais importantes foram alcançados. Neste trabalho foi apresentado um breve histórico dos resultados obtidos até a presente data. Dois métodos gerais são bastante usados tanto nas provas dos resultados parciais, quanto em algoritmos para encontrar palavras sincronizadoras mínimas, sendo conhecidos como Método de Fusão e Método de Extensão. O primeiro é, na prática, mais aplicado em algoritmos, enquanto que o segundo é mais aplicado em provas de resultados parciais.

Dentre estes resultados, destaca-se o lema proposto por Steinberg que reúne as ideias do método de Extensão com álgebra linear e por meio do qual faz-se possível provar novos resultados parciais. Neste trabalho foi mostrado como aplicar o lema para encontrar um limitante para as seguintes classes de autômatos: Pseudo-Eulerianos, Regulares, 1-ciclo e para a Série 4.

O Problema da Sincronização é NP-completo, e portanto, os algoritmos polinomiais para encontrar palavras sincronizadoras são baseados em heurísticas. Neste trabalho apresentamos os seguintes algoritmos: o algoritmo trivial para obter uma palavra mínima, o algoritmo de Eppstein e os dois algoritmos de Roman. Os algoritmos de Eppstein e de Roman foram implementados e para o algoritmo de Eppstein foram utilizados 5 critérios diferentes de implementação. Além disso, foram realizados experimentos comparativos em relação ao tamanho da palavra obtida por cada algoritmo. Estes experimentos foram feitos sobre algumas séries infinitas de autômatos sincronizados e mostraram que os algoritmos de Roman nem sempre cumprem a proposta de obter palavras sincronizadoras menores através de um custo computacional maior. Como visto, para a Série 4, o algoritmo de Eppstein obteve melhores resultados.

Apêndice A

Implementação dos Algoritmos

Os algoritmos das Seções 3.3 e 3.4 foram implementados usando o sistema GAP [GAP13]. O GAP é um sistema que fornece uma linguagem de programação, além de diversas bibliotecas e pacotes, dentre eles um pacote específico para a manipulação de autômatos. O sistema é um software livre[1].

O pacote para manipulação de autômatos está disponível em http://www.gap-system.org/Packages/automata.html e oferece uma grande quantidade de funções[2]. Além disso, o pacote com as implementações dos algoritmos está disponível em http://www.ime.usp.br/~gindri/. Não é necessário instalar nenhum dos pacotes, basta colocá-los dentro da pasta `/gap4r5/pkg/` e ao iniciar o GAP carregar o pacote **Smallwords** através do comando `LoadPackage("smallwords");`.

Dessa forma torna-se possível realizar os primeiros testes. Para construir o autômato da Figura A.1 no GAP: `aut:= Automaton("det",5,"ab",[ [ 2, 3, 4, 5, 1 ], [ 3, 3, 4, 5, 1 ] ],[ ],[ ]);`. Todas as chamadas de função em GAP devem terminar em ";".

Note que `[ [ 2, 3, 4, 5, 1 ], [ 3, 3, 4, 5, 1 ] ]` é a matriz de transições deste autômato, onde `[ 2, 3, 4, 5, 1]` representa as transições com a letra a e `[ 3, 3, 4, 5, 1 ]` representa as transições com a letra b. Na posição 1 do primeiro vetor encontra-se o valor 2, isto significa que neste autômato o estado 1 possui uma transição com a para o estado 2, o que pode ser confirmado na Figura A.1.

O comando `Display(aut)` exibe as transições do autômato.

Para criar o autômato de pares a partir deste autômato: `autPares:= A2(aut)`. No autômato de pares, o estado 1 representa o estado *sinc*, o estado 2 representa o par $[1, 2]$, o estado 3 representa o par $[1, 3]$. A função `gerarParesDeEstados(n)` retorna uma lista com essa relação.

Ou seja, `paresDeEstados:= gerarParesDeEstados(5) = [[ 1 ], [ 1, 2`

[1] As informações sobre como instalar o GAP estão disponíveis em http://www.gap-system.org/Download/index.html

[2] Para consultar todas as funções disponíveis neste pacote acesse http://www.gap-system.org/Manuals/pkg/automata/doc/chap0.html

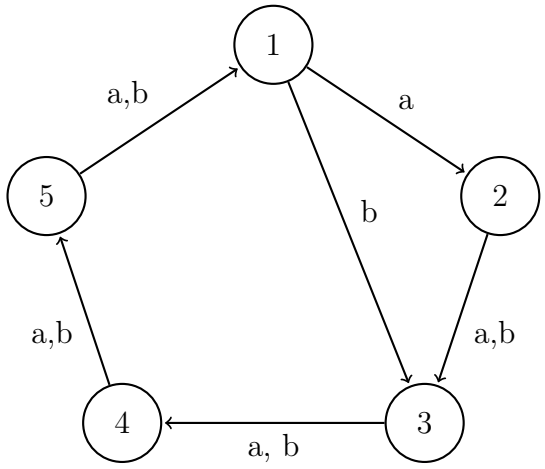

Figura A.1: *O autômato da Serie* 1 *com* 5 *estados.*

`], [ 1, 3 ], [ 1, 4 ], [ 1, 5 ], [ 2, 3 ], [ 2, 4 ], [ 2, 5 ], [ 3, 4 ], [ 3, 5 ], [ 4, 5 ] ]`.

Em seguida cria-se a árvore de caminhos mais curtos (invertida), passando o autômato de pares e uma lista com o alfabeto do autômato:

- `arvInv1 := arvoreBLInv(autPares,[1,2])`: gera a árvore de caminhos mais curtos respeitando a ordem lexicográfica;
- `arvInv2 := arvoreBLInv(autPares,[2,1])`: gera a árvore de caminhos mais curtos com ordem lexicográfica reversa.

Neste caso, `arvInv1 = [ [ [ 2, 2 ] ], [ [ 1, 5 ] ], [ [ 1, 8 ] ], [ [ 1, 10 ] ], [ [ 1, 11 ] ], [ [ 0, 0 ] ], [ [ 0, 0 ] ], [ [ 1, 4 ] ], [ [ 2, 3 ], [ 1, 6 ] ], [ [ 1, 7 ] ], [ [ 1, 9 ] ] ]`. Observe que `arvInv1[1] = [ [ 2, 2 ] ]` significa que o estado 1 nesta árvore atinge o estado 2 através de b. A primeira posição representa a letra da transição e a segunda o estado alcançado. Dessa forma, como `arvInv1[9] = [ [ 2, 3 ], [ 1, 6 ] ]`, então do estado 9 partem duas transições, uma através de b atingindo o estado 3 e outra atingindo 6 através de a. A Figura A.2 representa graficamente a estrutura `arvInv1`.

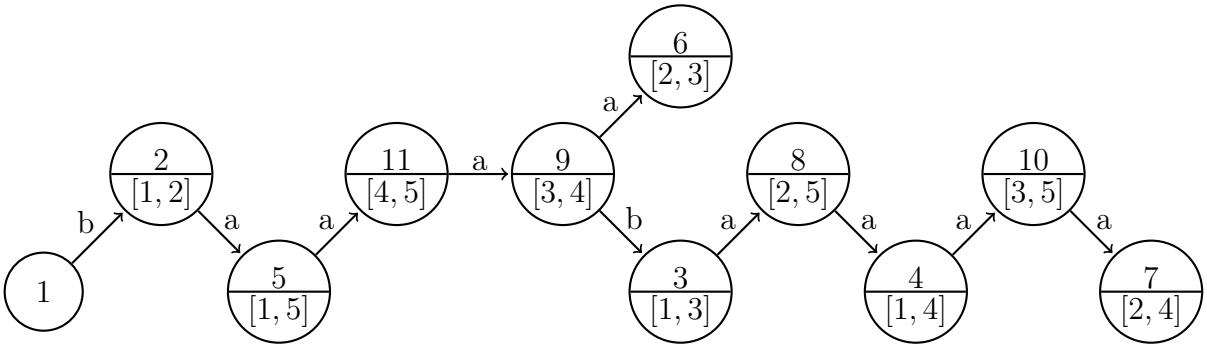

Figura A.2: *Árvore de caminhos mais curtos gerada pela função arvoreBLInv.*

Antes de poder gerar as palavras mínimas para cada estado é necessário iniciar as estruturas:

```
words1 := [[]];
words2 := [[]];
for i in [1.. NumberStatesOfAutomaton( autPares )] do


    words1[i] := [0," ",0];

    words2[i] := [0," ",0];

od;
```

Cada tripla `[0," ",0]` irá conter: par de estados, palavra mínima que leva este estado ao estado *sinc* e o tamanho desta palavra.

Para gerar as palavras mínimas para cada par de estado:

- `words1:= gerarPalavraMinimaParaCadaPar(arvInv1, paresDeEstados, words1, 1," ", 0)`: palavras geradas a partir da busca em largura em ordem lexicográfica;
- `words2:= gerarPalavraMinimaParaCadaPar(arvInv2, paresDeEstados, words2, 1, " ", 0)`: palavras geradas a partir da busca em largura em ordem lexicográfica reversa.

Então é possível chamar os algoritmos fazendo:

```
algoEppstein(aut,autPares,words1);

algoEppstein(aut,autPares,words2);

chamaAlgoEppsteinAleat(aut,autPares,words1,words2,30, 1);

chamaAlgoEppsteinAleat(aut,autPares,words1,words2,30, 2);

chamaAlgoEppsteinAleat(aut,autPares,words1,words2,30, 3);

algoRoman(aut,autPares,words1,1);

algoRoman(aut,autPares,words1,2).
```

Para testar todos os algoritmos com as séries apresentadas na Seção 3.6, basta chamar: `comparaAlgoritmosParaAsSeries(n)`, onde $5 < n \in \mathbb{N}$ é o número de estados.

Referências Bibliográficas

[AGV10] D. Ananichev, V. Gusev, and M. Volkov. Slowly synchronizing automata and digraphs. In *Proceedings of the 35th international conference on Mathematical foundations of computer science*, MFCS'10, pages 55–65, Berlin, Heidelberg, 2010. Springer-Verlag.

[AGW77] R. Adler, W. Goodwyn, and B. Weiss. *Equivalence of topological Markov shifts.* Israel Journal of Mathematics, 1977.

[Alf05] J. L. R. Alfonsín. *The diophantine Frobenius problem.* Oxford lectures series in mathematics and its applications. Oxford university press, Oxford, New York, 2005.

[Ash56] W. R. Ashby. *An introduction to cybernetics.* London : Chapman and Hall, 1956.

[AV03] D. Ananichev and M. Volkov. Synchronizing monotonic automata. In Zoltán Ésik and Zoltán Fülöp, editors, *Developments in Language Theory*, volume 2710 of *Lecture Notes in Computer Science*, pages 111–121. Springer, 2003.

[AV05] D. Ananichev and M. Volkov. Synchronizing generalized monotonic automata. *Theor. Comput. Sci.*, 330:3–13, January 2005.

[BBP11] M. Beal, M. Berlinkov, and D. Perrin. A quadratic upper bound on the size of a synchronizing word in one-cluster automata. *International Journal of Foundations of Computer Science*, 22(02):277–288, 2011.

[Ber10] M. Berlinkov. On a conjecture by Carpi and D'Alessandro. In *Proceedings of the 14th international conference on Developments in language theory*, DLT'10, pages 66–75, Berlin, Heidelberg, 2010. Springer-Verlag.

[BJG02] Jorgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications.* Springer, August 2002.

[BP09] M. Beal and D. Perrin. A quadratic upper bound on the size of a synchronizing word in one-cluster automata. In *Proceedings of the 13th International Conference on Developments in Language Theory*, DLT 2009, pages 81–90, Berlin, Heidelberg, 2009. Springer-Verlag.

[Cer64] J. Cerný. A remark on homogeneous experiments with finite automata. *Mat.-Fyz. Casopis Sloven..*, 1964.

[Dub98] L. Dubuc. Sur les automates circulaires et la conjecture de Černý. *Theoretical Informatics and Applications*, 32:21–34, 1998.

[Epp90] D. Eppstein. Reset sequences for monotonic automata. *SIAM J. Comput.*, 19(3):500–510, 1990.

[GAP13] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.6.4*, 2013.

[HU79] JE Hopcroft and JD Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts, 1979.

[Kar01] J. Kari. A counter example to a conjecture concerning synchronizing words in finite automata. *Bulletin of the EATCS*, 73, 2001.

[Kar03] J. Kari. Synchronizing finite automata on eulerian digraphs. *Theor. Comput. Sci.*, 295:223–232, February 2003.

[Nat86] B. K. Natarajan. An algorithmic approach to the automated design of parts orienters. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 132–142, Washington, DC, USA, 1986. IEEE Computer Society.

[Pin78a] J. E. Pin. Le problème de la synchronisation et la conjecture de Černý. In *Noncommutative structures in algebra and geometric combinatorics*, pages 37–48, 1978.

[Pin78b] J. E. Pin. Sur un cas particulier de la conjecture de Černý. In Giorgio Ausiello and Corrado Böhm, editors, *Automata, Languages and Programming*, volume 62 of *Lecture Notes in Computer Science*, pages 345–352. Springer Berlin / Heidelberg, 1978. 10.1007/3-540-08860-1_25.

[Pin83] J. E. Pin. On two combinatorial problems arising from automata theory. *Annals of Discrete Mathematics*, 17:535–548, 1983.

[Rom05] A. Roman. New algorithms for finding short reset sequences in synchronizing automata. In Cemal Ardil, editor, *IEC (Prague)*, pages 13–17. Enformatika, Çanakkale, Turkey, 2005.

[Rys95] I. Rystsov. Quasioptimal bound for the length of reset words for regular automata. *Acta Cybern.*, 12(2):145–152, 1995.

[Sip07] M. Sipser. *Introdução à Teoria da Computação: Tradução da 2ª edição norte-americana (trad. Ruy José Guerra Barreto de Queiroz)*. Thomson Learning, São Paulo, 2007.

[Ste10] B. Steinberg. The averaging trick and the Černý conjecture. In *Proceedings of the 14th international conference on Developments in language theory*, DLT'10, pages 423–431, Berlin, Heidelberg, 2010. Springer-Verlag.

[Ste11] B. Steinberg. The Černý conjecture for one-cluster automata with prime length cycle. *Theoretical Computer Science*, 412(39):5487 – 5491, 2011.

[Tra06] A. N. Trahtman. Notable trends concerning the synchronization of graphs and automata. *Electronic Notes in Discrete Mathematics*, 25:173–175, 2006.

[Tra07] A. N. Trahtman. The Road Coloring Problem. *Israel J. Math*, abs/0709.0099:51–60, 2007.

[Tra11] A.N. Trahtman. Modifying the upper bound on the length of minimal synchronizing word. In Olaf Owe, Martin Steffen, and JanArne Telle, editors, *Fundamentals of Computation Theory*, volume 6914 of *Lecture Notes in Computer Science*, pages 173–180. Springer Berlin Heidelberg, 2011.

[Vol08] M. Volkov. Language and automata theory and applications. chapter Synchronizing Automata and the Černý Conjecture, pages 11–27. Springer-Verlag, Berlin, Heidelberg, 2008.

[Wal08] P. Walker. Synchronizing automata and a conjecture of Černý. Master's thesis, University of Manchester - School of Mathematics, 2008.

Índice Remissivo