Mineração de padrões frequentes em séries temporais para apoio à tomada de decisão em agrometeorologia

Daniel Yoshinobu Takada Chino

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura:_____

Mineração de padrões frequentes em séries temporias para apoio à tomada de decisão em agrometeorologia[1]

Daniel Yoshinobu Takada Chino

Orientadora: **Profa. Dra. Agma Juci Machado Traina**

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

USP – São Carlos
Maio de 2014

[1] Trabalho realizado com apoio financeiro da FAPESP – Processo 2011/15017-0

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

C539m

Chino, Daniel Yoshinobu Takada
Mineração de padrões frequentes em séries temporais para apoio à tomada de decisão em agrometeorologia / Daniel Yoshinobu Takada Chino; orientadora Agma Juci Machado Traina. -- São Carlos, 2014.
78 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2014.

1. Séries Temporais. 2. Mineração de Dados. 3. Padrões Frequentes. I. Traina, Agma Juci Machado, orient. II. Título.

"Temos a sorte de viver numa era em que ainda estamos fazendo descobertas."

Richard P. Feynman

Agradecimentos

Aos meus pais, que sempre me incentivaram e apoiaram nos momentos de indecisões e escolhas de novos caminhos, permitindo que eu pudesse continuar meus estudos. Ao meu irmão que sempre me aconselhou e à minha irmã que sempre me mostrou o quanto o esforço é recompensador.

À minha orientadora Profa. Dra. Agma Juci Machado Traina por toda a paciência, apoio e todo o conhecimento transmitido.

À minha namorada Mayumi pelo carinho que sempre me confortou nos momentos difíceis.

À Dra. Luciana Alvim Romani pela paciência e auxílios durante o projeto. Às meninas do CEPAGRI-EMBRAPA em Campinas pela ajuda e sugestões.

Aos amigos do GBDI pelas discussões e conselhos: Willian Denner, Alceu Ferraz, Pedro Lopes, Sérgio, Mônica, Marcos Bêdo, Ives Pola, Arthur, Ândrea, Marcelo Ponciano, Letrícia, Bruno, Santiago, Luiz, Christian, Jéssica e em especial ao Felipe Louza pela parceria na realização de parte desse trabalho de Mestrado.

Aos amigos do Grupo de Estudos para Maratona de Programação (GEMA) pelos momentos de diversão e muito estudo.

Aos funcionários e professores do ICMC-USP.

À FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) pelo apoio financeiro à realização desse trabalho.

Resumo

O crescente aumento no volume de dados complexos tem se tornado um desafio para pesquisadores. Séries temporais são um tipo de dados complexos que tem tido um crescimento em sua relevância, devido a sua importância para o monitoramento e acompanhamento de safras agrícolas. Assim, a mineração de informação a partir de grandes volumes de séries temporais para o apoio a tomada de decisões tem se tornado uma atividade valiosa. Uma das atividades importantes na mineração em séries temporais é a descoberta de padrões frequentes. Entretanto, a complexidade dessa atividade requer métodos rápidos e eficientes. Nesse contexto, esta dissertação de mestrado apresenta propostas para novos algoritmos e métodos para minerar e indexar séries temporais. Uma das propostas dessa dissertação é o índice *Telesto*, que utiliza uma estrutura baseada em árvores de sufixo generalizada para recuperar séries temporais em uma base de dados de séries temporais de modo rápido e eficiente. Outra proposta dessa dissertação é o algoritmo *TrieMotif*, que se baseia em uma *trie* para eliminar comparações desnecessárias entre subsequências, agilizando o processo de mineração de padrões frequentes em séries temporais. Os algoritmos propostos foram utilizados para a análise de dados climáticos e agrometeorológicos. Os resultados apresentados nessa dissertação de mestrado mostram que os algoritmos são escaláveis, podendo ser utilizados para grandes volumes de dados.

Abstract

Dealing with large volumes of complex data is a challenging task that has motivated many researchers around the world. Time series is a type of complex data that is growing in importance due to the increasing demand of sensors for surveillance and monitoring. Thus, mining information from large volumes of time series to support decision making is a valuable activity nowadays. This Master dissertation goes in this direction, as it proposes new algorithms and methods to mine and index time series. The novelty of the TrieMotif, a new algorithm to mine frequent patterns (motifs) from time series employing a trie structure that allows clever comparison between the sequences, as well as the Telesto index structure based on suffix trees area presented and discussed in the context of agrometeorological and climatological data, being the two main contributions of this work. The dissertation shows that the proposed algorithms are scalable, being suitable to big data, and when compared to the competitors they always presented the best results.

Sumário

Lista de Figuras **vii**

Lista de Tabelas **xi**

Lista de Algoritmos **xiii**

Lista de Símbolos **xv**

1 Introdução **1**

1.1 Considerações Iniciais 1

1.2 Motivação 2

1.3 Objetivos do Trabalho 2

1.4 Organização do Trabalho 3

2 Séries Temporais **5**

2.1 Considerações Iniciais 5

2.2 Pré-processamento em Séries Temporais 6

2.2.1 Limpeza de Dados 6

2.2.1.1 Dados faltantes 6

2.2.1.2 Remoção de Ruídos 7

2.2.2 Normalização 8

2.3 Medidas de Similaridade em Séries Temporais 10

2.3.1 Distâncias Baseadas na Família Minkowski 11

2.3.2 *Dynamic Time Warping* 12

2.3.3 Maior Subsequência em Comum 14

2.4 Representação de Séries Temporais 14

2.4.1 Técnicas Não Adaptativas 14

2.4.2 Técnicas Adaptativas 17

2.4.3 Técnicas Baseadas em Modelos 20
2.4.4 Técnicas Ditadas pelos Dados 20
2.5 Técnicas de Sumarização de Séries Temporais 21
2.5.1 Sumarização Baseadas em Estatística 21
2.5.2 Sumarização Baseada em Codificação *Run-Length* 22
2.5.3 Sumarização Baseada em Histograma 23
2.6 Considerações Finais 24

3 Mineração de *Motifs* em Séries Temporais 27
3.1 Considerações Iniciais 27
3.2 Definições e Notações 27
3.3 Padrões Frequentes 31
3.4 Pares de Padrões Mais Próximos 35
3.5 Considerações Finais 37

4 O Índice *Telesto* 39
4.1 Considerações Iniciais 39
4.2 Árvore de Sufixo 40
4.3 O Índice *Telesto* 42
4.4 Teste de Desempenho 45
4.5 Considerações Finais 48

5 O Algoritmo *TrieMotif* 49
5.1 Considerações Iniciais 49
5.2 O Algoritmo *TrieMotif* 49
5.3 O Algoritmo *TrieMotif* 49
5.4 Teste de Desempenho 56
5.5 Aplicação em Imagens de Satélite 59
5.6 Considerações Finais 65

6 Conclusões 67
6.1 Considerações Finais 67
6.2 Principais Contribuições 67
6.3 Trabalhos Futuros 68

Referências Bibliográficas 71

Lista de Figuras

2.1 Suavização de ruídos de uma série temporal através de médias móveis. 8
2.2 Aplicação dos diferentes tipos de normalização. 9
2.3 Casamento de subsequência convertido em casamento completo. 10
2.4 Comparação entre séries temporais. 12
2.5 Apresentação hierárquica das representações de séries temporais. Adaptado de [Lin 07] 15
2.6 Representação PAA de uma série temporal de tamanho 128 para o tamanho 8. Adaptado de [Lin 03] 16
2.7 Série temporal discretizada utilizando a representação SAX com um alfabeto de três símbolos. Adaptado de [Lin 03]. 18
2.8 Representação de uma série temporal através de uma sequência de *bits*. Adaptado de [Ratanamahatana 05] 21
2.9 Uma série temporal apresentando *run-length* nos níveis 5 e 7. 22
2.10 Histograma da série temporal apresentada na Figura 2.9 24

3.1 Casamento entre as subsequências S_p e S_q. 28
3.2 Tanto as subsequências que antecedem quanto as que sucedem formam um casamento com S_q. 29
3.3 Representação no espaço bidimensional da unicidade dos *K-Motifs* Frequentes. . . 30
3.4 Primeira etapa do algoritmo de projeção aleatória. Através de uma janela deslizante a subsequência S_1 é extraída e convertida na cadeia de caracteres $\hat{S}_1$. A cadeia S_1 é então armazenada na matriz $\hat{S}$ de modo que cada linha corresponde a posição temporal da subsequência na série temporal. 32
3.5 A esquerda é possível observar a máscara $\{1,2\}$ escolhida aleatoriamente. A matriz de colisão armazenando as colisões está representada a direita. Adaptado de [Chiu 03]. 33

3.6 Na segunda iteração do algoritmo foi utilizada a máscara $\{2,4\}$ e a posição $(58,1)$ da matriz de colisão é incrementada. Adaptado de [Chiu 03]. 33
3.7 Discretização através dos ângulos 34
3.8 Proposta de ordenação de sequências no espaço 2D. Adaptado de [Mueen 09b]. . . 36

4.1 Exemplo de uma consulta da subcadeia *ana* em uma árvore de sufixo contendo a palavra *banana*. 41
4.2 Example of a generalized Suffix Tree 42
4.3 Etapa de construção do índice *Telesto*. 43
4.4 Etapa de consulta do índice *Telesto*. 43
4.5 Consulta por abrangência utilizada no índice *Telesto*. 44
4.6 Tempo gasto para a construção do índice *Telesto*. 46
4.7 Tempo de resposta para uma consulta por abrangência utilizando-se o índice *Telesto*. 47
4.8 Tempo de resposta para uma consulta por vizinho mais próximo utilizando-se o índice *Telesto*. 47

5.1 As subsequências presentes na região destacada apresentam um comportamento similar à subsequência de consulta S_q. 50
5.2 As etapas realizadas pelo *TrieMotif* para a descoberta de *motifs*. 51
5.3 Conversão de uma subsequência de tamanho $n = 128$ para a representação simbólica de $a = 3$ caixas de tamanho $w = 8$. 51
5.4 As subsequências S_1, S_2, S_3 e S_4 convertidas para as representações simbólica $\hat{S}_1 = AABD$, $\hat{S}_2 = BABD$, $\hat{S}_3 = ABDA$ e $\hat{S}_4 = DCCA$ respectivamente, utilizando como parâmetro $w = 4$ e $a = 4$. 52
5.5 As representações simbólicas são indexadas em ordem não sequencial para diminuir a quantidade de caminhos percorridos durante a busca por abrangência. 53
5.6 Através de uma busca na *trie*, é possível obter apenas candidatos a casamento, descartando as subsequências com comportamentos diferentes da subsequência de consulta. 54
5.7 Utilizando uma indexação em ordem sequencial, o número de caminhos percorridos durante a busca pode aumentar. Diferentemente da indexação não sequencial, o caminho para a subsequência $\hat{S}_3$ marcado em azul também é percorrido. 54
5.8 Dados sintéticos de passeio aleatório com *motifs* implantados nas posições pré-determinadas 32, 287, 568 e 875. 57
5.9 Comparações realizadas com os algoritmos *TrieMotif*, força bruta e projeção aleatória. 58
5.10 Número médio de nós visitados por consulta utilizando indexação sequencial e não sequencial. 59
5.11 Os 4-*Motif* mais frequentes utilizando toda a área do estado de São Paulo. 60

5.12 O estado de São Paulo foi dividido em 5 regiões de interesse. A região 1 (roxo) corresponde a água e áreas urbanas, as regiões 2 e 3 (azul e verde respectivamente) a pastagem e misto de culturas, a região 4 (amarelo) apresenta áreas de cultivo de cana-de-açúcar e a região 5 a florestas. 61
5.13 Os 4-*Motif* mais frequentes para a região de interesse 1 (Água e Região Urbana). . 62
5.14 Os 4-*Motif* mais frequentes para a região de interesse 2 (Pastagem e Mistura de Culturas). 62
5.15 Os 4-*Motif* mais frequentes para a região de interesse 3 (Pastagem e Mistura de Culturas). 63
5.16 Os 4-*Motif* mais frequentes para a região de interesse 4 (Cultura de Cana-de-açúcar). 63
5.17 Os 4-*Motif* mais frequentes para a região de interesse 5 (Regiões de Floresta). . . 64

Lista de Tabelas

5.1 Tamanho n dos *motifs* procurados na *trie* e os respectivos tamanhos w utilizados para as representações simbólicas. Foi utilizada uma razão de 6 entre n e w para definir o tamanho da representação simbólica. 58
5.2 Regiões de interesse segundo os especialistas em agrometeorologia e sensoriamento remoto. 61

Lista de Algoritmos

3.1 Algoritmo de Força Bruta para descobrir 1-*Motif* 31
3.2 Algoritmo de Força Bruta para descobrir o *Motif* de Vizinhos mais Próximos . 36
5.1 Algoritmo *K-TrieMotif* 56

Lista de Símbolos

α_i	Símbolo de um alfabeto.
β_i	Pontos de controle da técnica SAX.
δ	Valor limiar para a busca em abrangência da *trie*.
ε	Taxa de erro.
μ	Média dos valores de uma série temporal.
σ	Desvio padrão dos valores de uma série temporal.
$\psi(t)$	Função mãe da transformada de *wavelers*.
$\varphi(t)$	Função escala da transformada de *Haar*.
Σ	Alfabeto de símbolos de tamanho a.
Σ^*	Conjunto de todas as cadeias de caracteres a partir de um alfabeto Σ.
$\overline{AB}$	Segmento de reta entre os pontos A e B.
\$	Caracter que representa o fim de uma cadeia de caracteres.
a	Tamanho de uma alfabeto de símbolos.
M	Quantidade de valores de um histograma.
m	Tamanho de uma série temporal.
n	Tamanho de uma subsequência.
C	Série temporal na representação *Clipping*.
c_i	i-ésimo elemento de uma série temporal na representação *Clipping*.
$D(S_p, S_q)$	Função de distância entre duas subsequências.
F_k	k-ésimo coeficiente complexo de *Fourier*.
L_p	Distância de *Minkowski*.
L_∞	Distância de *Chebychev*.
L_1	Distância de *Manhattan*.
L_2	Distância Euclidiana.
$N(0,1)$	Distribuição normal.
R	Raio de abrangência.
RNN	Distância até o vizinho mais próximo.

$\mathcal{S}$	Conjunto de cadeias de caracteres.
S_p	Subsequência de uma série temporal iniciando no tempo p.
S_q	Subsequência de consulta.
$S^{\{k\}}$	k-ésimo *motif* mais frequente.
$\hat{S}$	Série temporal ou subsequência em uma representação simbólica.
$\hat{S}_q$	Subsequência de consulta na representação simbólica.
$\hat{S}[i]$	i-ésimo elemento de um subsequência na representação simbólica.
$\hat{S}[i:j]$	Uma subcadeia de caracteres entre o i-ésimo e o j-ésimo caracter.
$\hat{S}[i::]$	Sufixo de uma cadeia de caracteres começando na posição i.
s_i	i-ésimo elemento de uma representação simbólica.
$Simbolo_{No}$	Símbolo de um nó da *trie* que possui um nó filho ou elemento.
$\mathcal{T}$	Base de dados de séries temporais.
T	Série temporal.
$\|T\|$	Tamanho de uma série temporal/subsequência.
T_q	Série temporal de consulta.
$\bar{T}$	Série temporal na representação PAA.
$\check{T}$	Série temporal na representação APCA.
t_i	i-ésimo elemento de uma série temporal.
$\bar{t}_i$	i-ésimo elemento de uma série temporal na representação PAA.
w	Tamanho de uma representação PAA.

Capítulo 1

Introdução

1.1 Considerações Iniciais

Há uma grande demanda por algoritmos e métodos para apoiar processos de tomada de decisão em grandes bases de dados complexos, que são tipicamente imagens, vídeo, áudio, séries temporais, entre outros. Sensores de monitoramento, em geral produzem uma grande quantidade de dados visando trazer informações relevantes para os seus proprietários.

A comunidade científica no Século XXI encontra-se muito preocupada com o cenário das mudanças climáticas causadas pelo aquecimento global, e busca respaldar a escolha de ações administrativas e governamentais necessárias para mitigar as consequências negativas relativas ao meio ambiente, agricultura, e ciclo vegetativo/animal em geral, além de buscar mecanismos para revertê-las [IPCC 07]. Para isso, dados de monitoramento ambiental e de sensoreamento remoto, por meio de satélites tem sido bastante utilizados

O Brasil, pela sua dimensão continental e importância estratégica de sua agricultura, tem investido nesse tipo de monitoramento. Pelos estudos atuais e cenários projetados de aumento da temperatura o Brasil poderá sofrer alterações em seus recursos e divisas provenientes da agricultura, e conhecer os mecanismos que estão induzindo tais cenários pode auxiliar a mitigar os resultados de tais ocorrências, bem como fazer gestões de planejamento e decisões que visem manter a agricultura viável economicamente.

Neste projeto de mestrado foram estudados métodos de organização e extração de padrões frequentes de séries temporais que são obtidas de sequências de dados e imagens coletados por sensoreamento remoto. Como tais dados são muito volumosos, é muito importante sumarizá-lo e organizá-los em estruturas de dados que permitam sua indexação, o que agilizará o processamento das consultas que serão realizadas. Além disso, devido ao grande tamanho das bases de

dados geradas, a demanda por algoritmos escaláveis torna-se mandatória.

1.2 Motivação

Séries temporais são os dados usualmente coletados por sensores de monitoramento, e no escopo deste trabalho foram utilizados sensores climáticos e meteorológicos posicionados tanto em estações terrestres quanto em satélites. Sensores terrestres e atmosféricos fornecem medidas diárias, semanais, mensais ou em intervalos de tempo previamente definidos pela demanda da aplicação em estudo pelos especialistas. Sequências de imagens obtidas por satélite podem gerar séries temporais de medidas de interesse que serão utilizadas para apoiar processos de análise e tomada de decisão por especialistas na área. Um modo muito interessante de obter tais medidas ou índices agrometeorológicos é calculá-los sobre sequências de imagens coletadas por satélite. Assim, séries temporais de medidas obtidas das imagens, como por exemplo o NDVI (*Normalized Difference Vegetation Index*), que indica o vigor vegetativo do solo [Rouse 73], podem ser automática e rapidamente obtidas dessas sequências e séries temporais de tais medidas podem ser construídas, como foi realizado no trabalho desenvolvido em [Chino 11]. Desse modo, a série temporal sumariza um grande conjunto de informações que anteriormente demandavam um grande esforço de análise pelos especialistas da área.

Após ter em mãos as séries temporais, é importante extrair informações que permitam recuperar da base de dados outras ocorrências semelhantes ou padrões que se repetem na série (*motif*). Tais informações permitirão que o especialista na área possa efetuar suas análises, por exemplo, em busca de cenários similares com o que está no momento analisando. A descoberta de padrões frequentes é uma importante atividade, apresentando diversas aplicações como reconhecimento de gestos [Meng 08], validação de assinaturas manuscritas [Gruber 06] e análise do mercado de ações [Jiang 09]. Além de sua capacidade em sumarizar e comprimir os dados, a descoberta de *motifs* também pode ser utilizada como uma etapa de pré-processamento para a mineração de regras de associação [Li 10] e predição de séries temporais [Truong 12]. Entrentanto, a descoberta de *motifs* apresenta uma alta complexidade [Mueen 09b], acarretando em grandes desafios. Assim, devido ao fato de que atualmente as bases de dados coletadas são muito volumosas, o processamento de tais atividades deve ser ágil. Foi nessa direção que esta pesquisa de mestrado foi realizada.

1.3 Objetivos do Trabalho

O foco deste trabalho de Mestrado foi desenvolver algoritmos e técnicas de mineração de dados e séries temporais, que permitam reconhecer padrões frequentes que ocorrem nas séries, além de organizá-las em uma nova estrutura de dados que possibilita sua indexação para responder consultas por similaridade. Ou seja, consultas do tipo: “Dada a série de NDVI de uma determinada região do Brasil, quais são as 5 séries de outras regiões do Brasil mais semelhantes a ela?”,

ou "Dada esta série de perfil de temperatura da região central do estado de São Paulo, quais são as outras séries da base que diferem dela em até 10%?".

Além disso, o desenvolvimento desta pesquisa objetivou a criação de métodos que não sofram com o crescimento do tamanho da base de dados. Esse é um aspecto importante, porque tais bases de dados podem ser da ordem de terabytes, como por exemplo, as séries obtidas por execução de modelos climáticos. Então, este trabalho de mestrado teve por objetivo o desenvolvimento de métodos rápidos e eficientes para a extração de conhecimento das séries temporais de dados do domínio agrometeorológico e climático. Nesse contexto, foram propostos métodos para:

- Realizar consultas por similaridade em séries temporais;
- Realizar a descoberta dos k padrões mais frequentes em uma série temporal.

Comumente, as séries temporais apresentam valores no domínio dos números reais e também uma alta dimensionalidade, assim, os métodos propostos se utilizam de técnicas de representações simbólicas de séries temporais. Desse modo, é possível realizar consultas por similaridade utilizando estruturas hiearárquicas de busca como as árvores de sufixo e a *trie*. A descoberta de padrões frequentes é um dos grandes desafios na mineração de dados em séries temporais, devido a sua alta complexidade computacional. Assim, um dos focos desse projeto de mestrado foi encontrar maneiras de reduzir essa complexidade reduzindo o número de cálculos de distância entre os padrões da série temporal.

1.4 Organização do Trabalho

Essa dissertação de mestrado está organizada como descrito a seguir. No Capítulo 2 são discutidos os principais conceitos de séries temporais, como medidas de similaridade e representação de séries temporais. No Capítulo 3 são apresentados os principais conceitos e técnicas para a mineração de padrões em séries temporais. No Capítulo 4 é descrito o índice *Telesto* para consultas por similaridade proposto neste projeto de mestrado. No Capítulo 5 é apresentado o algoritmo *TrieMotif* para a descoberta de padrões frequentes em séries temporais. As conclusões, contribuições e possibilidades de pesquisas futuras decorrentes deste trabalho são apresentadas no Capítulo 6.

Capítulo 2

Séries Temporais

2.1 Considerações Iniciais

Uma série temporal é uma coleção de m observações realizadas sequencialmente ao longo do tempo (Definição 2.1).

Definição 2.1. *Série Temporal:* Uma série temporal é um conjunto $T = \{t_1, t_2, \ldots, t_m\}$ de m valores reais ordenado no tempo.

Usualmente os valores de uma série temporal pertencem ao domínio dos números reais. Podendo ser encontradas nas mais variadas áreas de aplicação, tais como finanças, *marketing*, agricultura, meteorologia e coleta de dados industriais e científicos. Uma série temporal é chamada de **contínua** quando seus valores são registrados continuamente. Enquanto que outras séries em que os valores são medidos em intervalos de tempo específicos são chamadas de **discretas**. Uma série temporal também pode ser classificada como estacionária ou não-estacionária. Séries temporais estacionárias apresentam média e variância constantes no decorrer do tempo, enquanto que séries temporais não-estacionárias sua média e variância não possuem padrões regularmente definidos.

Séries temporais são muito estudadas nas áreas de Estatística e Processamento de Sinais. Há vários objetivos referentes ao estudo de séries temporais, tais como:

- Investigar o mecanismo que gera a série temporal;
- Fazer previsões para valores futuros da série, a curto ou a longo prazo;
- Descrever o comportamento da série, construindo gráficos, verificando a existência de tendências, ciclos e variações sazonais, etc.;

- Procurar periodicidades importantes nos dados.

Na área de computação, os estudos de séries temporais podem ser divididos em duas vertentes: técnicas eficientes para indexação de séries temporais e mineração de séries temporais.

Neste capítulo serão discutidos brevemente alguns dos conceitos envolvidos no estudo de séries temporais. Inicialmente será discutido o pré-processamento de séries temporais, em seguida serão exploradas algumas medidas de similaridade. Serão mostrados os diferentes tipos de representação para séries temporais, como também algumas técnicas de sumarização.

2.2 Pré-processamento em Séries Temporais

Em grandes bancos de dados é comum a presença de dados incompletos (ausência de atributos), ruídos (presença de erros ou valores que desviam muito do esperado - *outliers*) e dados inconsistentes [Han 11]. O pré-processamento dos dados é uma das etapas mais importantes para a mineração de dados, podendo corresponder a 80% de todo o processo [Zhang 05]. O pré-processamento também pode aumentar a qualidade dos resultados das técnincas de mineração de dados.

A presença de dados incompletos pode ocorrer devido a diversos fatores. Os atributos de interesse podem não estar disponíveis, ou desinteresse do profissional responsável pela coleta dos dados, como também da inserção dos mesmos na base de dados. O mal funcionamento dos equipamentos de coleta de dados também pode acarretar em falhas na leitura, tornando assim a base de dados incompleta. Outro problema ocasionado pelo mal funcionamento dos equipamentos de coleta é a presença de ruídos nos dados. Além desse fator, os ruídos podem ser inseridos por fatores humanos no momento da inserção da informação, como também por falhas na transmissão dos dados. A inconsistência dos dados pode ser resultante de diferentes tipos de convenções ou códigos utilizados. Há também a necessidade de processar dados replicados. Esses problemas podem ser resolvidos através da limpeza dos dados, tratando dados faltantes e removendo ruídos, como também através da normalização dos dados.

2.2.1 Limpeza de Dados

A limpeza dos dados é o processo responsável em ajustar os dados para evitar que ocorram erros ao se aplicar técnicas de mineração de dados em séries temporais [Han 11]. A limpeza de dados pode ser realizada através de tratamentos para dados faltantes e a suavização de ruídos. A seguir serão discutidas técnicas para o tratamento de dados faltantes e remoção de ruídos.

2.2.1.1 Dados faltantes

Dados faltantes representam um dos maiores problemas em mineração de dados. Existem diversos fatores que podem causar a ausência de dados relevantes, como o mal funcionamento

de equipamentos, falha humana e condições do ambiente. O tratamento para dados faltantes depende do domínio de aplicação, como também da quantidade e tipos de dados faltantes. A seguir serão discutidas duas abordagens para lidar com dados faltantes:

- **Ignorar dados faltantes:** Uma abordagem que pode ser utilizada é ignorar os dados faltantes. No caso de comparações por similaridade, existem algoritmos que computam a similaridade entre duas séries temporais comparando intervalos locais. Caso esteja faltando um segmento de dados em uma das séries temporais, este segmento é simplesmente ignorado no cálculo de similaridade.

- **Preencher dados faltantes:** Outra abordagem é estimar um possível valor interpolando as medidas adjacentes. Pode ser utilizada em séries temporais ou caso esteja faltando um pequeno número de valores contíguos. Quanto maior a distância entre os valores adjacentes utilizados para estimar um valor intermediário, maior o erro da interpolação. Uma das abordagens utilizadas mais simples é a interpolação linear. Sejam os pontos x_1 e x_3 com os respectivos valores $f(x_1)$ e $f(x_3)$. Seja x_2 um ponto entre x_1 e x_3, que esteja com seu valor de $f(x_2)$ faltando. Pode-se utilizar a interpolação linear para estimar o valor de $f(x_2)$ através da Equação 2.1

$$f(x_2) = f(x_1) + \frac{(x_2 - x_1)(f(x_3) - f(x_1))}{x_3 - x_1} \tag{2.1}$$

Um exemplo do uso de interpolação linear em mineração de dados temporal é discutido em [Keogh 98], em que as séries temporais são aproximadas por segmentos de retas. Uma outra abordagem que apresenta resultados melhores do que a interpolação linear sem apresentar um alto custo computacional é a interpolação *spline*. Ela utiliza polinômios de ordem baixa para aproximar os dados faltantes, a interpolação cúbica é utilizada em [Perng 00].

2.2.1.2 Remoção de Ruídos

O ruído pode ser definido como erros aleatórios que ocorrem durante o processo de aquisição de dados. Pode ocorrer devido a diversos fatores como mal funcionamento dos equipamentos de coleta de dados, como também de fatores ambientais. A seguir serão discutidas duas técnicas para o tratamento de ruídos em mineração de dados: *binning* e suavização por média móvel.

Na técnica de *binning*, os dados são distribuídos em caixas ou baldes de mesmo tamanho. Então os dados são suavizados utilizando ou a média ou a mediana ou os valores extremos do balde. Por exemplo, para a série temporal descrita por *T={10, 11, 11, 15, 16, 19, 23, 24, 24, 26, 27, 28, 30, 32, 35,36, 36, 39, 41, 45}*, os dados são distribuídos em 4 caixas de tamanho 5:

$Caixa_1 = \{10, 11, 11, 15, 16\}$ $\quad$ $Caixa_2 = \{19, 23, 24, 24, 26\}$

$Caixa_3 = \{27, 28, 30, 32, 35\}$ $\quad$ $Caixa_4 = \{36, 36, 39, 41, 45\}$

Se for utilizada a suavização através da média, as caixas são mostradas abaixo:

$Caixa_1^{Media} = \{12.6, 12.6, 12.6, 12.6, 12.6\}$ $\quad$ $Caixa_2^{Media} = \{23.2, 23.2, 23.2, 23.2, 23.2\}$
$Caixa_3^{Media} = \{30.4, 30.4, 30.4, 30.4, 30.4\}$ $\quad$ $Caixa_4^{Media} = \{39.4, 39.4, 39.4, 39.4, 39.4\}$

Utilizando a suavização através da mediana:
$Caixa_1^{Mediana} = \{11, 11, 11, 11, 11\}$ $\quad$ $Caixa_2^{Mediana} = \{24, 24, 24, 24, 24\}$
$Caixa_3^{Mediana} = \{30, 30, 30, 30, 30\}$ $\quad$ $Caixa_4^{Mediana} = \{39, 39, 39, 39, 39\}$

A suavização por média móvel é muito utilizada em finanças para suavizar flutuações de curto prazo no preço de ações. Seja a série temporal com 5 pontos, representada na Figura 2.1 pelos quadrados. Deseja-se encontrar a curva de 3 pontos contento a média móvel central (*central moving average* - CMA), utilizando tanto o valor anterior quanto o valor seguinte do ponto de interesse. Por exemplo, a CMA do ponto no tempo 2, representada pelo triângulo na Figura 2.1, é a média dos valores no tempo 1 e no tempo 3. A CMA no tempo 4 é a média nos tempos 3 e 5 e assim por diante.

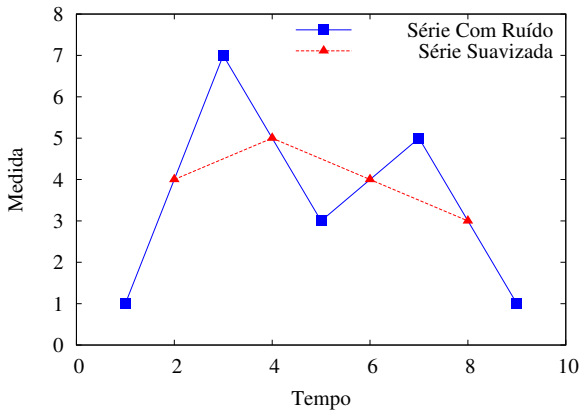


Figura 2.1: Suavização de ruídos de uma série temporal através de médias móveis.

2.2.2 Normalização

No processo de normalização, os dados são ajustados de modo que seus valores pertençam a um intervalo especificado. A normalização permite que os dados sejam ajustados a uma mesma escala e assim permitindo uma comparação direta entre seus valores. Por exemplo, no caso em que se queira comparar o comportamento da temperatura em duas cidades, porém em uma das cidades a temperatura é medida na escala Celsius e na outra cidade a temperatura é medida em Fahrenheit. Os valores medidos em ambas as cidades devem ser compatibilizados para que possam ser comparados. A normalização dos dados é uma etapa importante para a mineração de dados, sendo amplamente estudada na literatura [Keogh 03]. A seguir serão discutidas duas técnicas de normalização:

- **Normalização *min-max*:** Os valores das séries temporais são ajustados em um intervalo especificado, por exemplo $[0, 1]$, conservando o formato original da série temporal. Para

realizar esse tipo de normalização, é necessário o conhecimento prévio dos valores mínimo x_{min} e máximo x_{max} dos dados originais. A normalização *min-max* sofre com a presença de *outliers* com valores muito extremos de mínimo e máximo, o que acabam influenciando a normalização. O valor normalizado é obtido através da Equação 2.2.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.2}$$

- **Z-Normalização:** Os valores das séries temporais são ajustados de modo que a média de seus valores seja nula e que o desvio padrão seja unitário. Assim como na *min-max*, o formato original da série temporal também é conservado, porém de modo mais similar do que a curva obtida após a normalização *min-max*. Diferentemente da normalização *min-max*, na z-normalização não há a necessidade de se conhecer os valores extremos de mínimo x_{min} e máximo x_{max}. Esse método é útil quando há a presença de *outliers* no conjunto de dados que prejudicariam a normalização *min-max* [Han 11]. O valor normalizado pode ser obtido através da Equação 2.3, sendo necessário o conhecimento do valor médio μ e o desvio padrão σ dos dados originais.

$$x_{norm} = \frac{x - \mu}{\sigma} \tag{2.3}$$

Na Figura 2.2 pode ser visto o perfil de uma série temporal e seus valores normalizados. Pode-se observar que em ambas as técnicas o formato das séries temporais é conservado, sendo que na z-normalização há uma maior semelhança entre a série original e a normalizada. Pode-se observar também que pode haver modificações nos valores das séries temporais (deslocamentos para cima ou escala), sendo necessário guardar os parâmetros utilizados na normalização.

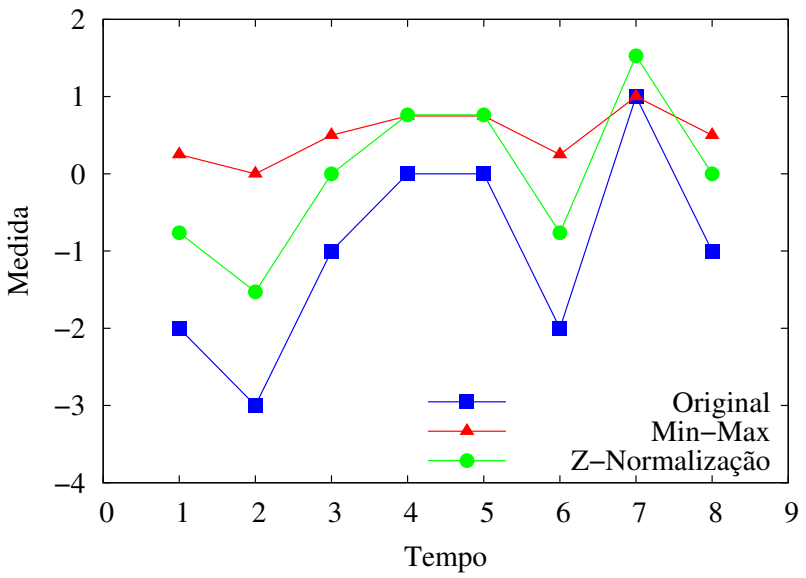


Figura 2.2: Aplicação dos diferentes tipos de normalização.

2.3 Medidas de Similaridade em Séries Temporais

A busca por similaridade (ou dissimilaridade) em séries temporais é um conceito essencial em mineração de dados, pois auxilia em tarefas de predição, teste de hipótese e detecção de regras [Faloutsos 94]. O problema de similaridade em série temporais pode ser expresso de dois modos diferentes [Mitsa 09]:

1. Encontrar todos os pares de séries temporais com distância menor que um dado R;

2. Consultas por similaridade, havendo duas abordagens para essa tarefa. A busca por abrangência, a qual dado um raio de abrangência R, deseja-se encontrar todas as séries temporais com distância menor que R de uma série temporal específica. Outra abordagem, é a dos k vizinhos mais próximos (*k nearest neighbors* kNN), a qual dada uma série temporal específica, deseja-se encontrar as k séries temporais mais próximas.

Em ambos os casos citados anteriormente, existem duas classes de problemas, o casamento (*matching*) completo e o casamento de subsequência [Keogh 00]. No casamento completo, as séries temporais comparadas possuem o mesmo tamanho, sendo utilizado todo o seu comprimento. Já no casamento de subsequências, as séries temporais comparadas possuem tamanhos diferentes. O casamento por subsequências pode ser transformado em um casamento completo utilizando uma janela deslizante, em que a série temporal de menor tamanho T_q é comparado com todas as subsequências da série de maior tamanho T, como ilustrado na Figura 2.3.

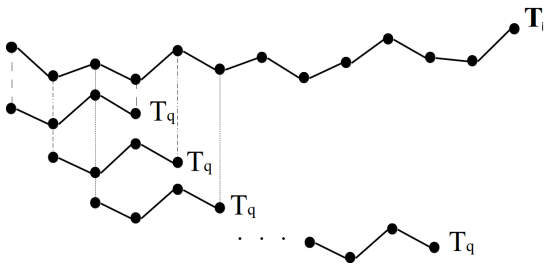


Figura 2.3: Casamento de subsequência convertido em casamento completo.

Visando agilizar os processos de casamento, podem ser utilizados esquemas de indexação. Existem algumas propriedades desejaveis para um esquema de indexação [Faloutsos 94]:

Propriedade 2.1. *Deve haver uma aumento significativo na velocidade das buscas se comparado à busca sequencial.*

Propriedade 2.2. *Não deve permitir a presença de falsos negativos. Não há problemas na ocorrência de falsos positivos, pois podem estes podem ser detectados por um pós processamento.*

Propriedade 2.3. *Deve permitir buscas de diversos tamanhos, permitindo também a inserção e remoção de dados sem a necessidade de reconstrução do índice.*

Propriedade 2.4. *O índice deve ser construído em um tempo admissível e ser capaz de tratar diferentes medidas de distância.*

É importante observar que ao se utilizar técnicas de normalização de séries temporais, a construção de um índice respeitando a Propriedade 2.3 se torna difícil. Muitas técnicas foram propostas para a indexação de séries temporais envolvendo estruturas de árvores. Uma das estruturas utilizadas é a *R-Tree* e suas variações, e a *kd-Tree* [Mitsa 09].

A seguir serão discutidas três das abordagens mais utilizadas em medidas de similaridade, baseadas na família *Minkowski*, maior subsequência em comum (*Longest Common Subsequence* - LCSS) e a *Dynamic Time Warping*.

2.3.1 Distâncias Baseadas na Família Minkowski

Algumas medidas de similaridade que podem ser utilizadas são pertencentes a família de distâncias *Minkowski* [Yi 00]. Sejam duas séries temporais $X = \{x_1, ..., x_m\}$ e $Y = \{y_1, ..., y_m\}$ de tamanho m, a família *Minkowski* (ou L_p) é definida pela Equação 2.4.

$$L_p(X,Y) = \left(\sum_{i=1}^{m} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{2.4}$$

onde $p \geq 0$.

Normalmente são utilizados $p = 1$ ou $p = 2$. A seguir serão apresentadas três distâncias pertencentes à família *Minkowski*, a distância máxima, a distância de *Manhattan* e a distância Euclidiana:

- **Distância de *Chebychev*:** A distância *Chebychev* é definida quando $p \to \infty$ e está descrita na Equação 2.5. A distância *Chebychev* enfatiza as características com maior dissimilaridade.

$$L_\infty = max(|x_i - y_i|) \tag{2.5}$$

- **Distância de *Manhattan*:** A distância de *Manhattan* é definida quando $p = 1$, possui esse nome devido à geografia das ruas de *Manhattan*. Pode ser obtida através da Equação 2.6 e não é tão enfática nas dissimilaridades.

$$L_1 = \sum_{i=1}^{m} |x_i - y_i| \tag{2.6}$$

- **Distância Euclidiana:** A distância Euclidiana é definida quando $p = 2$ e está indicada na Equação 2.7. Uma desvantagem do uso da distância Euclidiana para comparação de

séries temporais é a sensibilidade a ruídos ou variações no eixo x devido à comparação ponto a ponto e também a baixa tolerância a qualquer lacuna.

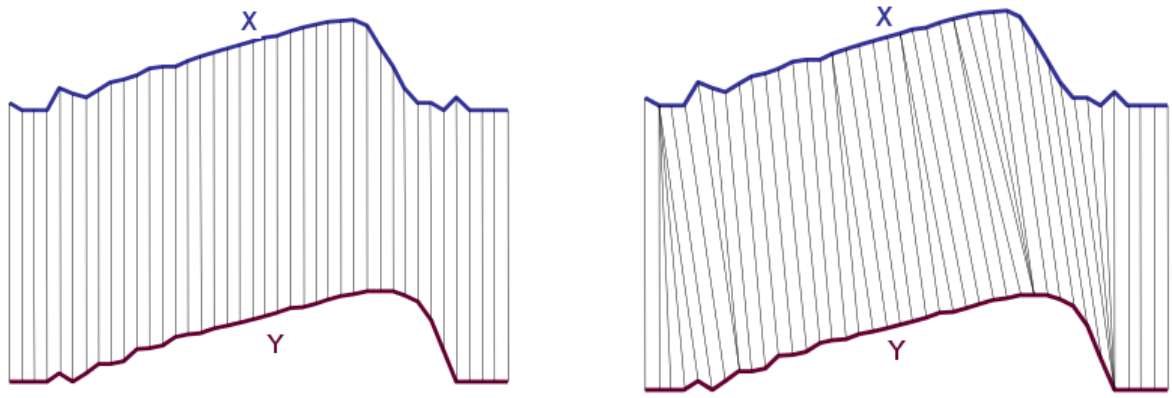
$$L_2 = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{2.7}$$

As distâncias L_p apresentam uma complexidade linear ($O(m)$) com o tamanho das séries temporais a serem comparadas, entretanto, demandam que ambas apresentem o mesmo tamanho. Em [Ding 08] é realizado um estudo comparativo entre medidas de similaridade em consultas do primeiro vizinho mais próximo (1NN). Apesar das desvantagens apresentadas pela distância Euclidiana, para bases de dados gigantes, a precisão alcançada pela distância Euclidiana se equipara à DTW, porém com um menor custo computacional.

2.3.2 *Dynamic Time Warping*

Dentre as técnicas mais difundidas usadas para se comparar duas séries temporais, destaca-se a *Dynamic Time Warping* (DTW) [Keogh 05], que consiste na aplicação de programação dinâmica para avaliar a proximidade de séries, minimizando a medida da distância entre séries que possuem desenvolvimentos semelhantes, mesmo que estejam defasadas ou "deformadas". A DTW vem sendo utilizada com sucesso em diversas aplicações tais como: mineração de dados [Ratanamahatana 04], reconhecimento de gestos [Alon 09], processamento da fala [Zhang 11], comparação entre séries de imagens médicas [Felipe 05].

A sensibilidade à deformação apresentada por esta técnica deve-se ao fato de que as comparações entre os pontos não são feitas rigidamente entre os pares localizados nas mesmas coordenadas, mas com a flexibilidade de que um ponto de uma série possa ser comparado com pontos adjacentes da outra série, conforme mostrado na Figura 2.4.



(a) Métodos tradicionais comparando ponto a ponto de mesma coordenada

(b) Utilizando a DTW que permite o deslocamento entre coordenadas

Figura 2.4: Comparação entre séries temporais.

Sejam duas séries $X = (x_1, ..., x_n)$ e $Y = (y_1, ..., y_m)$, de tamanhos n e m respectivamente. Para compará-las utilzando a DTW, uma matriz é construída, de modo que o elemento (i, j) contém a distância $d(x_i, y_j)$ (tipicamente é utilizada a distância Euclidiana). Cada elemento da matriz corresponde ao alinhamento entre os pontos nele representados.

Um percurso de ajuste $W = (w_1, ..., w_k)$, onde $\max(m,n) \leq k \leq m+n-1$, é um conjunto de elementos contíguos da matriz que define um mapeamento entre X e Y. Há muitos possíveis percursos de ajuste, porém o percurso a ser escolhido é aquele que minimiza o custo da deformação, ou seja, cuja distância acumulada ao longo do percurso é mínima. O caminho pode ser encontrado utilizando programação dinâmica calculando-se a distância acumulada na matriz construída através da Equação 2.8.

$$D_{acum}(i,j) = d(x_i, y_j) + \min(D_{acum}(i-1, j-1), D_{acum}(i-1, j), D_{acum}(i, j-1)) \tag{2.8}$$

O percurso de ajuste deverá obedecer às seguintes regras [Keogh 05]:

- Iniciar e terminar em células diagonalmente opostas da matriz, ou seja, $w_1 = (1,1)$ e $w_k = (n,m)$;
- A sequência entre os elementos do percurso deverá conter elementos adjacentes da matriz (incluindo elementos diagonalmente posicionados)
- A sequência não poderá "voltar" no caminho dos elementos da matriz.

O percurso ótimo, ou seja, aquele que minimiza o custo é calculado através da Equação 2.9.

$$DTW(X,Y) = \frac{\min\left(\sqrt{\sum_{i=1}^{K} w_i}\right)}{K} \tag{2.9}$$

onde w_k é o k-ésimo elemento do percurso de ajuste e K é o número de elementos do percurso de ajuste.

A DTW apresenta complexidade $O(nm)$, entretanto, é possível reduzir sua complexidade ao se utilizar uma janela de *warping* [Keogh 05]. Ao se restringir o tamanho da janela de *warping* pode ocorrer uma melhora na velocidade do algoritmo, pois apenas parte da matriz é calculada. Outra melhoria na velocidade do algoritmo é proposta em [Sakurai 05], a qual evita o cálculo de todos os possíveis percursos, excluíndo os percursos que não obterão sucesso na busca. Em [Chan 03] também é proposto um método para diminuir o custo computacional da DTW utilizando o conceito de *wavelets*. Em [Ratanamahatana 04] foi realizado um amplo estudo a respeito da DTW, onde é mostrado que o estado da arte atual, já atingiu seu limite assintótico em termos do melhoramento da velocidade da DTW exata. Porém, é possível realizar otimizações no cálculo da DTW através de soluções aproximadas, em [Salvador 07] foi proposta uma aproximação da DTW com complexidade linear em espaço e tempo.

2.3.3 Maior Subsequência em Comum

Em contraste com a distância Euclidiana, a LCSS (*Longest Common Subsequence*) é uma medida que é capaz de tolerar lacunas nas duas séries temporais comparadas. Assim como a DTW, a LCSS apresenta complexidade $O(nm)$. A LCSS apresenta as seguintes vantagens em relação à DTW [Vlachos 02, Vlachos 03]:

- LCSS manipula ruídos e *outliers* de modo mais otimizado, pois a LCSS permite que alguns pontos não sejam utilizados na comparação, enquanto que a DTW utiliza todos os pontos, incluíndo os *outliers*;
- A DTW pode distorcer a distância real entre pontos na série temporal devido ao *overfitting*;
- A DTW apresenta uma complexidade computacional significativa, pois em algumas situações, faz-se necessário o cálculo de L_p [Vlachos 02], não possuindo uma boa escalabilidade.

2.4 Representação de Séries Temporais

Usualmente, as séries temporais apresentam alta complexidade. Logo, utilizá-las em seu domínio original torna-se uma tarefa extremamente custosa [Han 11]. Assim, é desejável representar as séries temporais de modo a simplificá-las, reduzindo sua complexidade, mas ainda assim mantendo suas principais características. Além do aumento na velocidade computacional, a representação resumida de séries temporais permite compressão dos dados, como também a remoção de ruídos. Segundo [Ratanamahatana 05], a representação de séries temporais pode ser dividida em quatro categorias: adaptativas, não adaptativas, baseadas em modelo e ditadas pelos dados. Na Figura 2.5 é mostrado um diagrama da hierarquia de representações em séries temporais. Uma representação de uma série temporal deve respeitar duas propriedades [Faloutsos 94]:

Propriedade 2.5. *Não deve permitir a presença de falsos negativos, ou seja, é possível determinar uma função de distância no domínio transformado que seja menor ou igual à distância no domínio original (limitante inferior).*

Propriedade 2.6. *Deve reduzir a complexidade do problema de buscas por similaridade.*

2.4.1 Técnicas Não Adaptativas

As técnicas não adaptativas consideram apenas propriedades locais dos dados e constroem uma representação aproximada da série temporal [Wang 13]. As técnicas mais utilizadas nessa categoria são a Transformada Discreta de *Fourier* (*Discrete Fourier Trasnform* - DFT) [Agrawal 93],

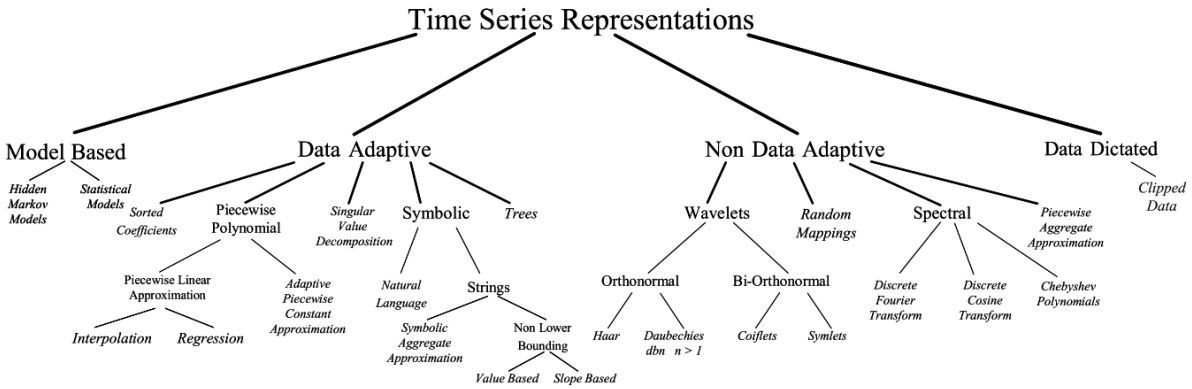


Figura 2.5: Apresentação hierárquica das representações de séries temporais. Adaptado de [Lin 07]

a Transformada Discreta de *Wavelet* (*Discrete Wavelet Transform* - DWT) [Chan 99] e a *Piecewise Aggregate Approximation* (PAA) [Keogh 01a]. Todas essas representações satisfazem a Propriedade 2.5.

A Transformada Discreta de *Fourier* (*Discrete Fourier Transform* - DFT) representa uma série temporal no domínio de frequências. A série temporal $T = \{t_1, t_2, ..., t_m\}$ passa a ser representada por uma sequência de coeficientes complexos F_k que são calculados através da Equação 2.10.

$$F_k = \sum_{j=1}^{m} t_j e^{\frac{-i2\pi(j-1)k}{m}} \tag{2.10}$$

onde $i = \sqrt{-1}$ e $k = 1, ..., m$.

Através da DFT é possível representar uma série temporal utilizando apenas os coeficientes iniciais, que representam frequências mais baixas, sem haver uma perda significativa. Por exemplo, se for aplicada a DFT em um caminho aleatório e for mantido apenas 10% dos coeficientes, ainda é mantida 90% da energia [Mitsa 09]. Outra vantagem de se utilizar a DFT é a existência de um algoritmo rápido para computá-la, o algoritmo de *Fast Fourier Transform* - FFT. O FFT possui a complexidade de $O(n \log n)$.

O cálculo de similaridade entre duas séries temporais representadas pela DFT pode ser feita utilizando as diferenças entre as partes reais e imaginárias dos coeficientes da transformada, sendo um limitante inferior para a distância Euclidiana. Uma outra medida que pode ser utilizada é computar a diferença entre as magnitudes da DFT ao quadrado, embora não seja um limitante inferior para a distância Euclidiana.

A Transformada Discreta de *Wavelet* (*Discrete Wavelet Transform* - DWT) utiliza como base uma função conhecida como *wavelet*. As *wavelets* são funções que permitem representar a série temporal no domínio espaço-frequência. Outra característica das *wavelets* é permitir a análise em diferentes escalas (resoluções). As funções de *wavelets* podem ser criadas a partir de uma função mãe ψ e as as *wavelets* filhas são criadas a partir de translações e contrações (ou

dilatações) da *wavelet* mãe através da Equação 2.11

$$\psi(t)^{s,\tau} = \frac{1}{\sqrt{s}}\psi(\frac{t-\tau}{s}) \tag{2.11}$$

onde s é a constante de escala utilizada para contrações ou dilatações e τ é a constante de translação utilizada para deslizar uma janela pela série temporal. O fator de escala permite analisar as séries temporais de modo multi-resolução. Os sinais de baixa frequência são analisados em baixas resoluções, que podem gerar a série temporal inteira, já os sinais de alta frequência são analisados em altas resoluções.

Uma das transformadas de *wavelet* mais simples é a transformada de *Haar*. Devido a sua simplicidade, a transformada de *Haar* é uma das mais utilizads em mineração de dados [Mitsa 09]. A função de escala $\varphi(t)$ da transformada de *Haar* é uma função degrau descrita pela Equação 2.12

$$\varphi(t) = \begin{cases} 1, & t \in (0,1) \\ 0, & \text{caso contrário} \end{cases} \tag{2.12}$$

A função mãe de *Haar* é descrita na Equação 2.13.

$$\psi(t) = \begin{cases} 1, & t \in [0,1/2) \\ -1, & t \in [1/2,1) \\ 0, & \text{caso contrário} \end{cases} \tag{2.13}$$

A complexidade computacional da transformada de *Haar* é de $O(n)$. A DWT não apresenta falsos negativos após passar um processo de normalização adequado [Chan 03].

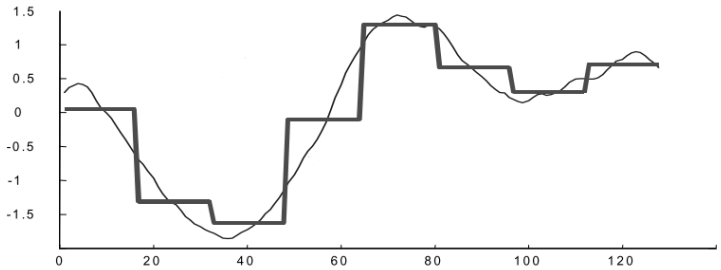


Figura 2.6: Representação PAA de uma série temporal de tamanho 128 para o tamanho 8. Adaptado de [Lin 03]

A *Piecewise Aggregate Approximation* (PAA) [Keogh 01a] é uma representação que apresenta uma redução significativa da dimensionalidade de uma série temporal. A série temporal é dividida em k segmentos de mesmo tamanho e cada segmento é representado pela média dos valores presentes no segmento. Seja a série temporal $T = \{t_1, ..., t_m\}$ uma série temporal

de tamanho m. A representação PAA para a série temporal T pode ser descrita pelo vetor $\bar{T} = \{\bar{t}_1, ..., \bar{t}_w\}$ de tamanho w, cujo i-ésimo elemento $\bar{t}_i$ de $\bar{T}$ é descrito pela Equação 2.14.

$$\bar{t}_i = \frac{w}{m} \sum_{j=\frac{m}{w}(i-1)+1}^{\frac{m}{w}i} t_j \tag{2.14}$$

A redução de dimensionalidade da série temporal T de tamanho m para tamanho w por meio da técnica PAA pode ser observada na Figura 2.6. A série temporal é dividida em janelas de tamanho igual a w e, para cada janela, calcula-se a média de seus valores. A série temporal passa então a ser representada pelo vetor de médias obtido pela Equação 2.14. Através da representação PAA é possível utilizar a função de distância descrita na Equação 2.15 para se comparar duas séries temporais X e Y de tamanho m, que é um limitante inferior da distância Euclidiana.

$$DR(\bar{X}, \bar{Y}) = \sqrt{\frac{m}{w}} \sqrt{\sum_{i=1}^{w} (\bar{x}_i - \bar{y}_i)^2} \tag{2.15}$$

2.4.2 Técnicas Adaptativas

As técnicas adaptativas utilizam uma representação em comum para todos os elementos da base de dados, de forma a minimizar os erros globais na reconstrução [Wang 13]. Nessa categoria se encontram as técnicas *Singular Value Decomposition* (SVD) [Korn 97], *Symbolic Aggregate Approximation* (SAX) [Lin 03], *Adaptive Piecewise Constant Approximation* (APCA) [Keogh 01b] e *Piecewise Linear Approximation* (PLA) [Keogh 01c].

Em [Korn 97] é proposta uma representação de séries temporais baseada no conceito de *Singular Value Decomposition* (SVD) de matrizes. A representação utilizando o SVD permite realizar, de modo eficiente, consultas em um grande conjunto de séries temporais. Utilizando apenas alguns coeficientes, o SVD atinge uma boa taxa de compressão dos dados, sendo capaz de reconstruir seus valores com uma baixa taxa de erro. O SVD aplica uma transformação em todo o conjunto de dados com o objetivo de encontrar os melhores eixos que descrevem a variação dos dados. Os dados são rotacionados de modo que o primeiro eixo tenha o máximo possível de variação, o segundo eixo tenha a máxima variação possível ortogonal ao primeiro eixo e assim por diante. Uma desvantagem dessa técnica é seu alto custo computacional e ao se adicionar uma nova série temporal, é necessário realizar os cálculos novamente [Mitsa 09]. Deve-se ressaltar que o SVD satisfaz a Propriedade 2.5.

A representação *Symbolic Aggregate Approximation* (SAX) aprimora a representação PAA. Nela as séries temporais discretizadas resultantes da representação PAA passam a ser representadas através de uma cadeia de símbolos. A técnica SAX consiste das seguintes etapas: Primeiro, a série temporal é z-normalizada de modo que a média de seus valores seja nula e que o desvio padrão seja unitário. Na sequência, a série temporal z-normalizada é convertida para

a representação PAA. Por fim, a série temporal na representação PAA é discretizada em uma cadeia de símbolos de alfabeto de tamanho $a > 2$. Na Figura 2.7 é mostrada uma série temporal na representação SAX.

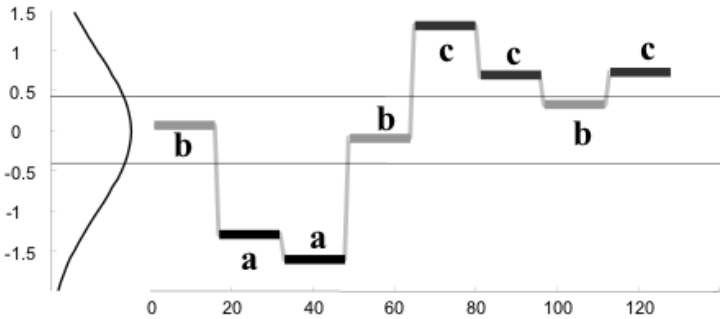


Figura 2.7: Série temporal discretizada utilizando a representação SAX com um alfabeto de três símbolos. Adaptado de [Lin 03].

A série temporal normalizada $T = \{t_1, ..., t_m\}$ é convertida para a representação PAA $\bar{T} = \{\bar{t}_1, ..., \bar{t}_w\}$ de tamanho w. Em seguida, a representação PAA $\bar{T}$ é convertida em uma sequência de símbolos discretos equiprováveis pertencentes a um alfabeto $\Sigma = \{\alpha_1, ..., \alpha_a\}$ com a caracteres. Séries temporais normalizadas apresentam uma distribuição Gaussiana, assim, definem-se os pontos crescentes $\beta_1, ..., \beta_{a-1}$, de modo que a área abaixo de uma curva Gaussiana $N(0,1)$ no intervalo entre β_i e β_{i+1} seja igual a $1/\sigma$ (β_0 e β_a são definidos respectivamente por $-\infty$ e ∞). Por fim, a série temporal $\bar{T}$ é discretizada na cadeia de símbolos $\hat{S} = s_1 ... s_w$, de modo que o i-ésimo termo s_i é obtido por meio da Equação 2.16.

$$s_i = \alpha_j, \qquad \beta_{j-1} < \bar{t}_i < \beta_j \tag{2.16}$$

Para comparar duas séries temporais na representação SAX $\hat{S}_a$ e $\hat{S}_q$ é necessário utilizar a função de distância *MINDIST*, proposta em [Lin 03] e descrita na Equação 2.17. Nessa equação, a função $dist(\alpha_i, \alpha_j)$ representa a distância entre dois símbolos e é calculada por meio da Equação 2.18.

$$MINDIST(\hat{S}_a, \hat{S}_q) = \sqrt{\frac{m}{w}} \sqrt{\sum_{i=1}^{w} (dist(s_{ai}, s_{qi}))^2} \tag{2.17}$$

$$dist(\alpha_i, \alpha_j) = \begin{cases} 0, & |i-j| \leq 1 \\ \beta_{max(i,j)-1} - \beta_{min(i,j)}, & \text{caso contrário} \end{cases} \tag{2.18}$$

Em [Lin 07] é mostrado que a função *MINDIST* se apresenta como um limitante inferior da distância Euclidiana, não implicando na ocorrência de falsos negativos.

A representação *Adaptive Piecewise Constant Approximation* (APCA) se assemelha à representação PAA, porém ao invés de utilizar segmentos de mesmo tamanho, a APCA permite que

a série temporal seja representada por segmentos de tamanhos distintos. Segundo os autores em [Keogh 01b], não é possível determinar um paralelo entre o desempenho da APCA com o da PAA, pois o resultado de ambas varia conforme a natureza dos dados utilizados. Para uma série temporal $T = \{t_1, ..., t_m\}$, a APCA utiliza dois valores para cada segmento, um representando a média e o outro que permite indicar o comprimento do segmento, como descrito na Equação 2.19:

$$\check{T} = \{\langle tv_1, tr_1 \rangle, ..., \langle tv_M, tr_M \rangle\} \tag{2.19}$$

onde tv_i é a média do i-ésimo segmento ($tv_i = media(t_{(tr_{i-1}+1)}, ..., t_{tr_i})$), tr_i o término direito do i-ésimo segmento e $tr_0 = 0$. Como são armazenados os locais de término de cada segmento, o comprimento do i-ésimo segmento pode ser obtido através da diferença $tr_i - tr_{i-1}$. A representação APCA não apresenta falsos negativos, pois existe uma função de distância que respeita a Propriedade 2.5.

Uma vantagem da APCA é poder adaptar o número de segmentos para um período da série temporal. Assim, regiões menos detalhadas podem ser representadas por um único segmento e regiões com maior detalhamento utilizam um número maior de segmentos [Keogh 01b].

A PLA [Keogh 01c] é uma técnica que aproxima as séries temporais através de segmentos de reta e uma função custo é aplicada na criação de cada segmento. A função custo representa o erro existente ao se aproximar um segmento da série temporal em uma linha reta. Uma função custo que pode ser utilizada é a distância Euclidiana entre o segmento da série temporal e a linha aproximada. Existem três abordagens para realizar essa tarefa: janela deslizante, *bottom-up* e *top-down* [Last 04]. Ao se utilizar a janela deslizante, o algoritmo é iniciado na parte mais à esquerda da série temporal, aumentando ponto a ponto o tamanho do segmento e a cada iteração a função custo é calculada. O algoritmo continua até atingir um erro estipulado e então passa para o próximo segmento de reta, o procedimento é repetido até o final da série temporal. Na abordagem *bottom-up*, a série temporal é dividida nos menores segmentos possíveis, ou seja, para uma série temporal de n valores, o algoritmo é iniciado com $n/2$ segmentos. Os segmentos menores são agrupados em segmentos maiores, sendo que a cada iteração a função custo é checada. E por último, na abordagem *top-down*, a série temporal é primeiramente aproximada pelo maior segmento de reta possível. Em seguida o segmento maior é dividido em segmentos menores de modo que a função custo seja a menor possível.

Em [Keogh 01c] é realizado um amplo estudo na representação de séries temporais através de segmentos de reta. Segundo o autor, a abordagem de janela deslizante não apresenta um resultado satisfatório, já a abordagem *top-down* apresenta resultado adequado, porém não se mostra muito escalável. Por último a abordagem *bottom-up* mostrou o melhor resultado, além de uma boa escalabilidade.

2.4.3 Técnicas Baseadas em Modelos

Essas técnicas são baseadas em modelos estatísticos ou em modelos estocásticos. É possível representar uma série temporal utilizando modelos de *Markov*. Modelos de *Markov* cujos parâmetros são desconhecidos são denominados de modelos escondidos de *Markov* (*Hidden Markov models* - HMM) [Rabiner 89]. Um HMM de primeira ordem, ou seja, depende apenas do estado atual para determinar o próximo estado, sendo descrito pelos seguintes parâmetros:

- Número de estados;
- A distribuição de probabilidade de transição de um estado ir a outro;
- A densidade de observações;
- A distribuição de probabilidade do estado inicial.

Para que um sistema seja completamente modelado probabilisticamente é preciso especificar o seu estado atual e os estados anteriores. Um caso especial é a cadeia de *Markov*, pois para ela é necessário apenas o conhecimento do estado atual e do estado antecessor. O problema de estimar os parâmetros de um HMM, dadas as observações, pode ser visto como um problema de máxima verossimilhança. Porém, este problema não possui uma solução global, algoritmos iterativos como o *Baum-Welch* [Baum 70] garantem a convergência apenas para máximos locais.

2.4.4 Técnicas Ditadas pelos Dados

Nas técnicas adaptativas, não adaptativas e baseadas em modelo, o usuário tem a opção (implícita ou explícita) de escolher a taxa de compressão dos algortimos. Já na categoria de técnicas ditadas pelos dados, não há a necessidade de o usuário optar pela taxa de compressão, os dados em si são responsáveis por indicar sua compressão [Ratanamahatana 05]. Para a técnica *Clipping* [Ratanamahatana 05], a compressão das séries temporais depende apenas da média.

A transformação da série temporal é realizada substituindo os valores reais da série temporal por um único *bit*. Dada uma série temporal $T = \{t_1, t_2, \ldots, t_m\}$ de m valores reais, a série transformada $C = c_1 c_2 \ldots c_m$ é obtida substituindo-se os valores reais de T por *bits* respeitando a Equação 2.20. Na Figura 2.8 é possível observar uma representação visual da transformação de uma série temporal para uma sequência de *bits*.

$$C_i = \begin{cases} 1, & \text{se } t_i > \mu \\ 0, & \text{caso contrário} \end{cases} \tag{2.20}$$

onde μ é a média de X.

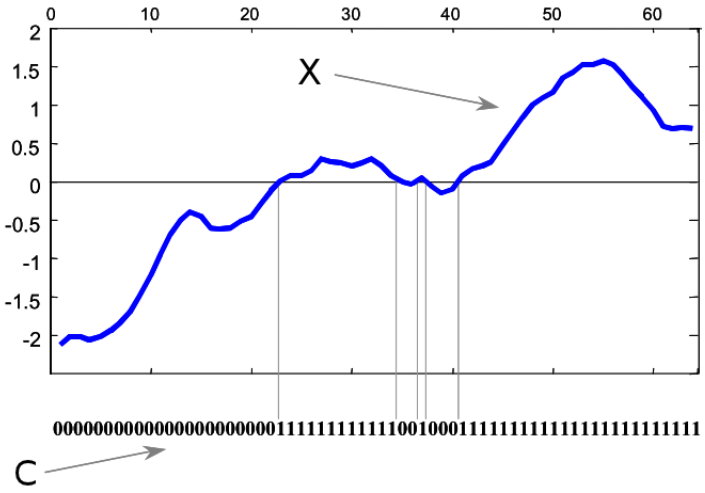


Figura 2.8: Representação de uma série temporal através de uma sequência de *bits*. Adaptado de [Ratanamahatana 05]

A sequência de *bits* obtidas pode então ser comprimida através da codificação *run length*. Segundo o autor essa representação apresenta um limitante inferior mais próximo do domínio original da série temporal, logo um maior poder de poda, agilizando as consultas.

2.5 Técnicas de Sumarização de Séries Temporais

A sumarização é uma técnica que utiliza as características globais de uma série temporal para representá-la. A sumarização é uma técnica muito vantajosa por apresentar uma redução significativa da dimensionalidade da série temporal. Geralmente as séries temporais são representadas por mais de uma técnica de sumarização, criando assim um vetor de características. O vetor de características consiste de várias características globais de uma série temporal.

2.5.1 Sumarização Baseadas em Estatística

Seja uma série temporal com m valores $T = \{t_1, t_2, ..., t_m\}$, é possível caracterizá-la através das seguintes medidas estatísticas:

- **Média:** Representa a média dos valores de uma série temporal, também chamada de primeiro momento e pode ser calculado pela Equação 2.21.

$$\mu = \sum_{i=1}^{m} \frac{x_i}{m} \tag{2.21}$$

- **Mediana:** Para a série temporal ordenada em ordem crescente de seus valores, a mediana é o valor intermediário que separa a metade com valores menores que a mediana da metade com valores maiores que a mediana. Se m for ímpar, a mediana é o valor central

da série temporal ordenada. Caso m seja par, a mediana é a média dos dois valores centrais da série temporal ordenada.

- **Moda:** É o valor que ocorre com maior frequência na série temporal.

- **Variância:** Mostra a variação dos valores da série temporal em relação a média, é também chamada de segundo momento e pode ser calculada pela Equação 2.22.

$$\sigma^2 = \frac{\sum_{i=1}^{m} (t_i - \mu)^2}{m-1} \tag{2.22}$$

Outra medida estatística que pode ser utilizada é o desvio padrão, que é definido pela raíz quadrada da variância.

2.5.2 Sumarização Baseada em Codificação *Run-Length*

A codificação *run-length* codifica uma sequência de valores que possuem o mesmo valor em uma série temporal. Por exemplo, a série temporal $T = \{1,7,7,1,3,5,5,7,1\}$ (Figura 2.9) possui duas *run-length*, uma para o valor 5 e outra para o valor 7. A série temporal pode ser codificada por $\{(7,2),(5,2)\}$, ou seja, a série temporal possui uma *run-length* de tamanho 2 no valor 5 e outra de tamanho 2 no valor 7.

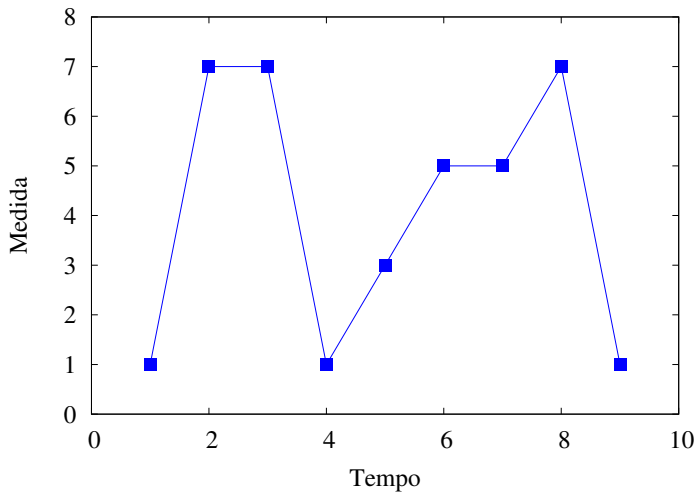


Figura 2.9: Uma série temporal apresentando *run-length* nos níveis 5 e 7.

Além da codificação *run-length*, é possível obter outras medidas através da matriz *run-length* descrita em [Uppaluri 02] para detectar a presença de patologias pulmonares em imagens de tomografia. Os elementos $r(i,j)$ dessa matriz indicam a presença de uma *run-length* de tamanho j para o nível de cinza i. Através dessa matriz é possível obter as seguintes medidas:

- ***Short Run-Length Emphasis*:** Essa medida indica a presença de *run-length* curtas e pode ser calculado através da Equação 2.23. Altos valores dessa medida indicam a presença de *run-length* curtas.

$$SRE = \frac{\sum_{i=1}^{NL}\sum_{j=1}^{NR}\frac{r(i,j)}{j^2}}{\sum_{i=1}^{NL}\sum_{j=1}^{NR} r(i,j)} \tag{2.23}$$

  onde NL é o número de valores (níveis) na série temporal, NR é o número de *run-length* e $r(i,j)$ é a *run-length* de tamanho j para o nível i.

- ***Long Run-Length Emphasis*:** Essa medida indica a presença de *run-length* longas e pode ser calculada através da Equação 2.24. Altos valores dessa medida indicam a presença de *run-length* longas.

$$SLE = \frac{\sum_{i=1}^{NL}\sum_{j=1}^{NR} j^2 r(i,j)}{\sum_{i=1}^{NL}\sum_{j=1}^{NR} r(i,j)} \tag{2.24}$$

  onde NL é o número de valores (níveis) na série temporal, NR é o número de *run-length* e $r(i,j)$ é a *run-length* de tamanho j para o nível i.

2.5.3 Sumarização Baseada em Histograma

O histograma é uma representação gráfica que indica a distribuição de frequências dos diferentes valores da série temporal. Em [Chen 04] é proposta uma técnica de sumarização de séries temporais baseada em histograma para realizar buscas exatas. O eixo das abscissas representa todos os possíveis valores da série temporal e no eixo das ordenadas está representada a frequência com que cada valor aparece. Normalmente o histograma é representado através de um gráfico de barras. Para a série temporal T mostrada na Figura 2.9, o histograma da série temporal é descrito por $(1,3),(2,0),(3,1),(4,0),(5,2),\ (6,0),(7,3)$ e pode ser visto na Figura 2.10.

Através do histograma é possível calcular medidas estatísticas que podem ser utilizadas para caracterizar uma série temporal. Para um histograma possuindo M valores, média μ, desvio padrão σ e $f(m)$ representando a frequência do valor m, tem-se:

- **Obliquidade:** Mede a assimetria de um histograma, é calculado pela Equação 2.25.

$$Obliquidade = \frac{\sum_{m=1}^{M}(f(m)-\mu)^3}{M\sigma^3} \tag{2.25}$$

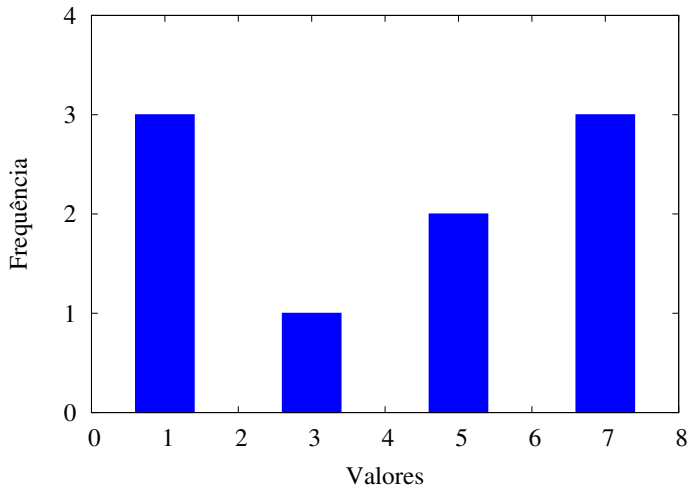


Figura 2.10: Histograma da série temporal apresentada na Figura 2.9

- **Curtose:** Mede a dispersão que caracteriza o pico ou "achatamento"do histograma, é calculado pela Equação 2.26.

$$Curtose = \frac{\sum_{m=1}^{M} (f(m) - \mu)^4}{M\sigma^4} \tag{2.26}$$

- **Entropia:** Mede a quantidade de informação ou incerteza de um histograma, pode ser calculado pela Equação 2.27.

$$Entropia = -\sum_{m=1}^{M} Pr(m) log(Pr(m)) \tag{2.27}$$

onde $Pr(m)$ é a probabilidade de ocorrência do valor m.

- 5° **e** 95° **Percentil:** Percentil é o valor o qual uma certa porcentagem dos valores se encontra abaixo dele. O 5° percentil apresenta 5% dos valores do histograma abaixo dele, já o 95° percentil apresenta 95% dos valores do histograma abaixo dele.

2.6 Considerações Finais

Neste capítulo foram apresentados resumidamente os principais conceitos relacionados a séries temporais. Foram apresentadas técnicas de pré-processamento de séries temporais, etapa fundamental ao se considerar dados agrometeorológicos. Também foi discutido o conceito de similaridade entre séries temporais, sendo apresentadas as principais medidas de similaridade da literatura reportadas para esse tipo de dados. Outro conceito importante apresentado nesse capítulo foi a representação de séries temporais de um modo mais simplificado. Visto que a

natureza contínua e sua alta complexidade dificultam a aplicação de técnicas de mineração de dados.

Capítulo 3

Mineração de *Motifs* em Séries Temporais

3.1 Considerações Iniciais

O crescente aumento no volume de séries temporais geradas a partir de sensores remotos, modelos climáticos e ações financeiras exige métodos automáticos para a extração de conhecimento a partir desses dados. Uma das principais tarefas na mineração de séries temporais é a descoberta de *motifs*, ou seja, padrões que ocorrem com frequência nas séries temporais. A mineração de *motifs* pode ser utilizada como entrada para a descoberta de regras de associação temporais, algoritmos de agrupamento e de classificação [Lin 02].

A mineração de *motifs* utiliza conceitos de similaridade entre subsequências de séries temporais (padrões), podendo ser dividida em dois problemas, a descoberta de padrões mais frequentes e a descoberta de pares de vizinhos mais próximos. Neste capítulo serão introduzidos os conceitos necessários para o entendimento do método proposto nesse projeto de mestrado, que será apresentado no Capítulo 5. Também serão apresentados os principais algoritmos para a mineração de *motifs*.

Este capítulo está organizado do seguinte modo. Na Seção 3.2 são apresentandas as principais definições e notações para o entendimento desse capítulo. As Seções 3.3 e 3.4 apresentam, respectivamente, algoritmos para a descoberta de *motifs* de padrões mais frequentes e de pares de vizinhos mais próximos. E por último são apresentadas as considerações finais na Seção 3.5.

3.2 Definições e Notações

Motifs em séries temporais são padrões previamente desconhecidos que ocorrem frequentemente ao longo de uma série temporal. A descoberta de *motifs* foi inicialmente definida por

Lin et al. em [Lin 02], sendo apresentada uma definição mais geral por Chiu et al. [Chiu 03]. Nesta seção, serão apresentados os conceitos e definições necessários para o entendimento da descoberta de *motifs*. Uma série temporal é um conjunto $T = \{t_1, t_2, \ldots, t_m\}$ de m valores reais ordenados no tempo (Definição 2.1).

Como os *motifs* são padrões menores que ocorrem ao longo de uma série temporal maior, não há o interesse em utilizar a série como um todo, sendo utilizado apenas pedaços da série. Um pedaço da série temporal é chamado de subsequência e é definido a seguir (Definição 3.1).

Definição 3.1. *Subsequência:* Dada uma série temporal T de tamanho m, uma subsequência S_p de T é uma amostra contígua de tamanho $n < m$ de T a partir da posição p, ou seja, $S_p = \{t_p, \ldots, t_{p+n-1}\}$ para $1 \leq p \leq m - n + 1$.

Uma etapa importante do processo de descoberta de *motifs* é a determinação de similaridade entre padrões distintos. Duas subsequências são consideradas como o mesmo padrão quando a similaridade entre as duas é menor do que um certo limiar, tal fato é também chamado de casamento de subsequências. O casamento entre duas subsequências é definido como mostra a Definição 3.2. Um exemplo de casamento entre subsequências pode ser visto na Figura 3.1, as subsequências S_p e S_q apresentam um padrão de comportamento similar, podendo ser classificadas como o mesmo padrão.

Definição 3.2. *Casamento:* Dada uma função de distância $D(S_p, S_q)$ entre duas subsequências, um número real positivo R (raio de abrangência) e uma série temporal T contendo as subsequências S_p e S_q, se $D(S_p, S_q) \leq R$, então diz-se que a subsequência S_q é um casamento de S_p.

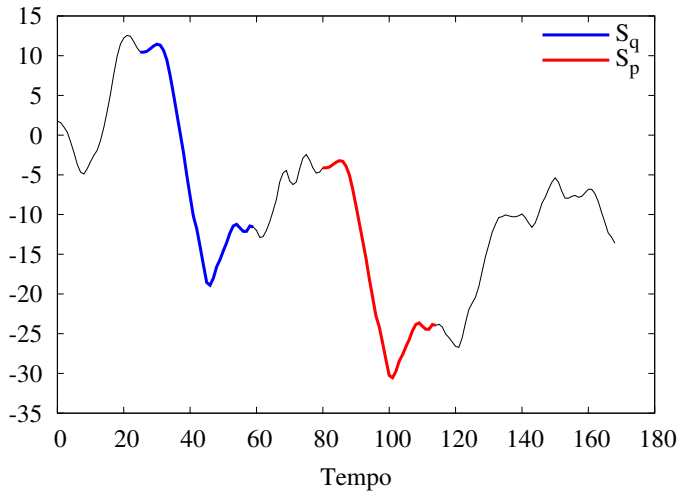


Figura 3.1: Casamento entre as subsequências S_p e S_q.

Lembrando que no problema da descoberta de *motifs* as subsequências são extraídas de uma mesma série temporal, os melhores casamentos podem ocorrer entre duas subsequências

que apresentam apenas um leve deslocamento temporal, ou seja, apresentam sobreposição. O casamento entre duas subsequências que se sobrepõem é chamado de casamento trivial. O casamento trivial é um problema, visto que poderiam existir *motifs* consistindo apenas de subsequências que ocorreram em instantes seguidos e não possuindo nenhuma relevância. Na Figura 3.2 é possível observar que para a subsequência S_q, tanto a subsequência que ocorre instantes antes, como a que ocorre instantes depois, formam um casamento. Pode-se notar que as três subsequências podem ser consideradas apenas como um único padrão. Para a descoberta de *motifs* é necessário então definir o conceito de casamento não-trivial.

Definição 3.3. *Casamento Não-Trivial:* Dada uma série temporal T contendo um casamento entre as subsequências S_p e S_q, diz-se que S_p e S_q formam um casamento não-trivial se houver pelo menos uma subsequência $S_{q'}$ começando na posição q', tal que $D(S_p, S_{q'}) > R$, sendo $q < q' < p$ ou $p < q' < q$. Ou seja, duas subsequências formam um casamento não-trivial, se houver pelo menos uma subsequência entre elas que não forme um casamento com uma delas.

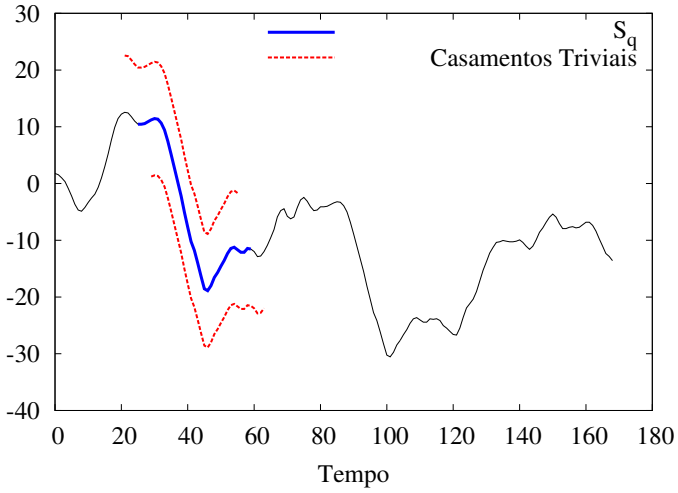
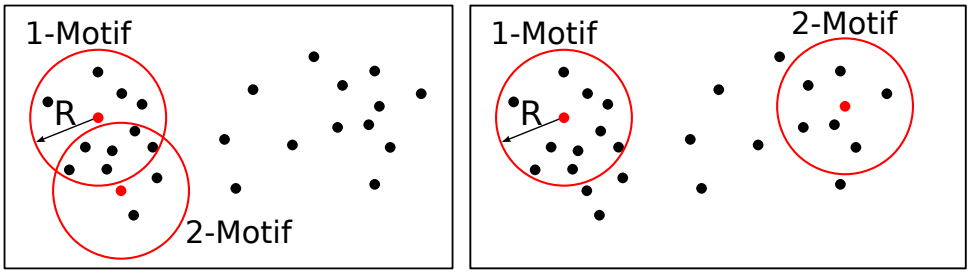


Figura 3.2: Tanto as subsequências que antecedem quanto as que sucedem formam um casamento com S_q.

A partir dessas definições, é possível formalizar o problema da descoberta dos *K-Motifs* Frequentes (Definição 3.4).

Definição 3.4. *K-Motifs Frequentes:* Dada uma série temporal T, uma subsequência de tamanho n e um raio de abrangência R, o *motif* frequente mais significativo em T (*1-Motif*) é a subsequencia $S^{\{1\}}$ que apresentar o maior número de casamentos não-triviais. O K-ésimo *motif* mais significativo em T (*K-Motif*) é a subsequência $S^{\{K\}}$ que apresentar o maior número de casamentos não-triviais e satisfizer $D(S^{\{K\}}, S^{\{i\}}) > 2R$, para todo i, tal que $1 \leq i < K$.

É importante notar que a Definição 3.4 exige que o conjunto de elementos (subsequências) dos *K-motifs* sejam mutualmente exclusivos. Assim, evita-se que dois *motifs* compartilhem



(a) $D(S^{\{K\}}, S^{\{i\}}) > R$ (b) $D(S^{\{K\}}, S^{\{i\}}) > 2R$

Figura 3.3: Representação no espaço bidimensional da unicidade dos *K-Motifs* Frequentes.

a maioria dos seus elementos, fazendo com que ambos pudessem ser considerados como o mesmo *motif*. A Figura 3.3 ilustra a diferença quando os centros dos *motifs* distam de R ou $2R$. Na Figura 3.3 as subsequências estão representadas em um espaço bidimensional. Note que se caso o centro dos *motifs* fossem distantes de apenas R (Figura 3.3a), grande parte dos elementos do 2-*motif* também estariam contabilizados no 1-*motif*. Já quando o centro dos *motifs* distam de $2R$ (Figura 3.3b), os elementos do 1-*motif* e do 2-*motif*, não são compartilhados. Assim, a Definição 3.4 impede que os k *motifs* apresentem sobreposição de elementos.

Uma outra definição de *motif* foi proposta em Yankov et al. [Yankov 07], o *Motif* de Vizinhos Próximos. Nessa proposta, o par de subsequências que apresentar a menor distância é considerado como o *motif* mais significativo, o par que apresentar a segunda menor distância é o segundo *motif* mais significativo e assim por diante. Desse modo, essa proposta alivia o problema de escolha de um raio de abrangência R, que é dependente dos dados que estão sendo utilizados. Porém, como já dito anteriormente, essa proposta tem por objetivo encontrar o par de subsequências mais similares e não as subsequências com o maior número de ocorrências. Assim como no *Motif* Frequente, a descoberta de vizinhos que ocorram entre intervalos de tempos pequenos não é relevante, chamados de vizinhos triviais.

Definição 3.5. *Vizinho Não-Trivial:* Dada uma série temporal T contendo as subsequências S_p e S_q, diz-se que S_p e S_q são vizinhos não-triviais se $p = q$ ou se houver pelo menos uma subsequência $S_{q'}$ começando na posição q', tal que $D(S_p, S_{q'}) > D(S_p, S_q)$, sendo $q < q' < p$ ou $p < q' < q$.

Pela Definição 3.5 para que duas subsequências S_p e S_q sejam vizinhos não-triviais, deve haver pelo menos uma subsequência entre elas que seja mais distante de S_p do que S_p é de S_q. Pode-se agora definir o *Motif* de Vizinhos Próximos.

Definição 3.6. *Motifs de Vizinhos Próximos:* Dada uma série temporal T, o motif de vizinhos próximos mais significantivo em T é o par de subsequencias S_p e S_q, de tamanho n, que sejam vizinhos não-triviais e $D(S_p, S_q)$ seja a menor possível.

3.3 Padrões Frequentes

Como descrito anteriormente, *motifs* são padrões previamente desconhecidos, sendo necessário verificar em todo o espaço de busca, ou seja, todas as subsequências de uma série temporal, para que sejam determinados os padrões mais frequentes. Uma possível abordagem para realizar essa tarefa faz uso de uma busca exaustiva por todas as subsequências, essa abordagem é chamada de algoritmo de força bruta. O Algoritmo 3.1 ilustra os passos necessários para achar o 1-*Motif* através da força bruta. Para se extrair todas as subsequências S_i ($1 \leq i \leq m-n+1$) de tamanho n de uma série temporal T de tamanho $m > n$ utiliza-se uma janela deslizante. Em seguida, para cada subsequência S_i, o algoritmo de força bruta passa por todas as subsequências S_j ($1 \leq j \leq m-n+1$) de T verificando quais formam um casamento não-trivial com S_i. Cada casamento não-trivial é então contabilizado e armazenado. A cada passagem, o algoritmo armazena apenas o *motif* mais significativo, ou seja, o que apresentar mais casamentos não-triviais. Ao final do processo, o algoritmo retorna o 1-*motif*. É possível notar que a abordagem de força bruta possui um custo computacional quadrático em relação ao tamanho da série temporal, visto que são necessárias $O(m^2)$ chamadas para a função de distância, tornando o seu uso impraticável.

Algoritmo 3.1 Algoritmo de Força Bruta para descobrir 1-*Motif*

Entrada: T, n, R
Saída: 1-*Motif*

```
melhorContagemMotif = 0;
melhorPosicaoMotif = NULL;
para i = 1 até tamanho(T) - n + 1 faça
  contagem = 0;
  vetorCasamentos = NULL;
  para j = 1 até tamanho(T) - n + 1 faça
    se casamentoNãoTrivial(S_i, S_j, R) então
      contagem = contagem + 1;
      vetorCasamentos = acrescentar(vetorCasamentos, j);
    fim se
  fim para
  se contagem > melhorContagemMotif então
    melhorContagemMotif = contagem;
    melhorPosicaoMotif = i;
    melhorVetorCasamentos = vetorCasamentos;
  fim se
fim para
```

Em Chiu et al. [Chiu 03] foi proposta uma abordagem eficiente para a descoberta de *motifs* baseada na projeção aleatória. O algoritmo de projeção foi proposto por Buhler e Tompa [Buhler 01] com o propósito de encontrar *motifs* implantados (cadeias de caracteres que diferem de apenas alguns símbolos) em sequências de DNA [Pevzner 00]. Como o algoritmo de projeção busca apenas por cadeias de caracteres, é necessário adaptar o algoritmo para que

seja possível utilizá-lo no domínio das séries temporais. Essa adaptação pode ser realizada através de representações simbólicas de séries temporais, ou seja, as séries temporais passam a ser representadas por cadeias de caracteres. Chiu et al. utiliza a técnica SAX para discretizar as séries temporais e também reduzir sua dimensionalidade.

Na primeira etapa, as subsequências são extraídas através de uma janela deslizante e então convertidas para uma notação simbólica através de técnica SAX. Cada subsequência já discretizada é então armazenada na matriz $\hat{S}$. Note que os índices das linhas da matriz $\hat{S}$ fazem uma correspondência direta com a posição temporal da subsequência discretizada na série temporal. O número de linhas da matriz deve ser igual ao número de subsequências extraídas da série temporal. Por exemplo, considere uma série temporal T de tamanho $m = 1.000$ que contenha apenas um *motif* de tamanho $n = 16$ nas posições S_1 e S_{58}. Na Figura 3.4 é possível observar o processo de extração e discretização das subsequências para essa série temporal. São extraídas 985 subsequências de tamanho $n = 16$ e após convertidas em cadeias de caracteres, são armazenadas na matriz $\hat{S}$. A cadeia de caracteres da linha 1 corresponde à subsequência S_1, a linha 2 corresponde à S_2 e assim por diante.

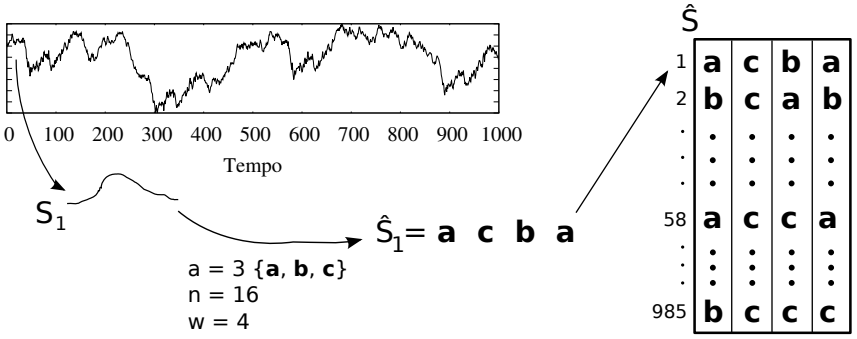


Figura 3.4: Primeira etapa do algoritmo de projeção aleatória. Através de uma janela deslizante a subsequência S_1 é extraída e convertida na cadeia de caracteres $\hat{S}_1$. A cadeia S_1 é então armazenada na matriz $\hat{S}$ de modo que cada linha corresponde a posição temporal da subsequência na série temporal.

Na segunda etapa do algoritmo, d colunas da matriz $\hat{S}$ são escolhidas aleatoriamente para serem utilizadas como máscara. Então, cada uma das subsequências de $\hat{S}$ são colocadas em baldes baseando-se nos valores das d colunas. Por exemplo, na Figura 3.5 foram escolhidas as colunas $\{1,2\}$ como máscara e cada subsequência foi armazenada no balde correspondente. Como as subsequências $\hat{S}_1$ e $\hat{S}_{58}$ possuem os mesmos símbolos, elas pertencem ao mesmo balde, assim como as subsequências $\hat{S}_2$ e $\hat{S}_{985}$ também pertencem ao mesmo balde.

A terceira etapa consiste em atualizar a *matriz de colisão*. Essa tarefa é realizada utilizando-se a distribuição das subsequências nos baldes. Se duas subsequências i e j pertencerem a um mesmo balde, a célula (i, j) da matriz de colisão deve ser incrementada, no exemplo (Figura 3.5), a posição $(58, 1)$ é incrementada. Note que embora apenas as subsequências S_1 e S_{58} sejam de fato um *motif*, a célula $(985, 2)$ também possui o mesmo valor na matriz de colisão,

indicando assim um falso positivo. Para evitar que esse problema ocorra, as etapas dois e três são repetidas por i iterações. A cada iteração uma nova máscara é escolhida aleatoriamente e os baldes são esvaziados. Na Figura 3.6 é possível observar uma segunda iteração utilizando a máscara $\{2,4\}$. O número de iterações pode ser determinado pelo usuário ou até que o critério previamente definido seja alcançado.

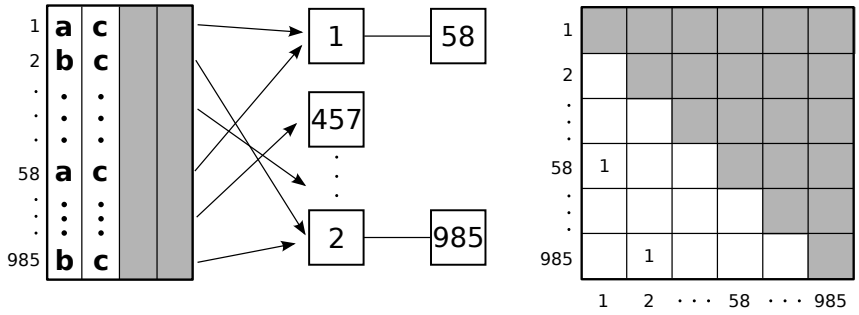


Figura 3.5: A esquerda é possível observar a máscara $\{1,2\}$ escolhida aleatoriamente. A matriz de colisão armazenando as colisões está representada a direita. Adaptado de [Chiu 03].

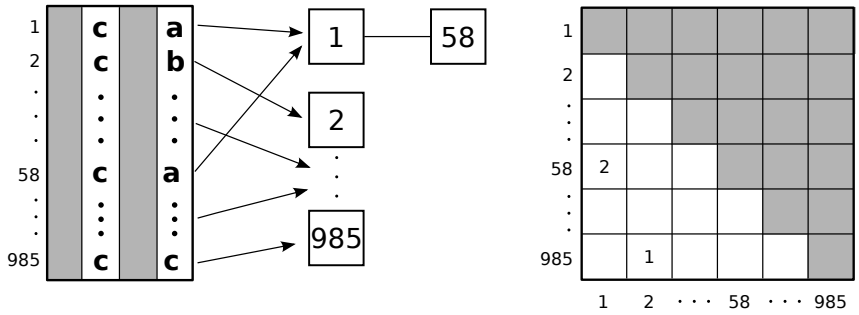


Figura 3.6: Na segunda iteração do algoritmo foi utilizada a máscara $\{2,4\}$ e a posição $(58,1)$ da matriz de colisão é incrementada. Adaptado de [Chiu 03].

A última etapa do algoritmo utiliza a matriz de colisão como um filtro de possíveis candidatos a *motif*. Caso os valores da matriz de colisão possuam uma distribuição uniforme, a série temporal não apresenta *motifs*. Já valores altos (estatisticamente relevantes) na matriz de colisão podem indicar *motifs*. As células que apresentarem valores altos são verificadas no domínio original da série temporal. Primeiro verifica-se se as duas subsequências formam um casamento não-trivial, no caso do exemplo as subsequências S_1 e S_{58}. Em caso positivo, através de uma janela deslizante, adiciona-se ao *motif* as subsequências que também formarem um casamento não-trivial com S_1.

O algoritmo de projeção aleatória é um dos principais algoritmos para a mineração de *motifs* frequentes. Segundo os autores, o algoritmo apresenta complexidade linear com o tamanho da série temporal, apresentando uma resposta de modo rápido e eficiente. Apesar de sua natureza probabilistica, o algoritmo apresenta bons resultados com apenas algumas iterações,

sendo possível melhorar sua precisão com um maior número de iterações. Embora eficiente, há a necessidade de utilizar uma matriz de tamanho m^2 para a matriz de colisão, tornando o método custoso em memória. Em Tang e Liao [Tang 08] foi proposto um aprimoramento no algoritmo de projeção aleatória para minerar *motifs* de tamanhos diferentes. Para tanto, o algoritmo inicia com valores pequenos de n, concatenando os padrões menores para descobrir os padrões maiores.

O trabalho de Udechukwu et al. [Udechukwu 04] apresentou um algoritmo para a descoberta de *motifs* frequentes utilizando uma árvore de sufixos. Assim como na projeção aleatória, é necessário que a série temporal passe por um processo de discretização. Como o problema de descoberta de *motifs* consiste em encontrar padrões que se repetem, é necessário que cada subsequência seja normalizada. Deste modo, não é possível utilizar métodos de discretização baseados nos valores das séries temporais para obter proveito das vantagens de uma árvore de sufixo. Para contornar tal fato, Udechukwu et al. propôs uma discretização simbólica baseada no ângulo de inclinação das retas formadas entre duas leituras consecutivas na série temporal. Um exemplo pode ser visto na Figura 3.7, ao invés de utilizar os valores dos pontos A, B e C, serão utilizados os ângulos ab e bc, formados respectivamentes pelas retas $\overline{AB}$ e $\overline{BC}$. Cada inclinação será representada por um símbolo de acordo com o valor de sua angulação. O tamanho do alfabeto pode ser ajustado de acordo com o grau de detalhamento desejado. Como os ângulos sempre possuem valores no intervalo entre -90° e $+90^o$, não há a necessidade de realizar ajustes específicos para cada domínio de dados. Desse modo, é possível representar padrões similares sem a necessidade de normalizar cada subsequência. Após o processo de discretização, a série temporal é indexada em uma árvore de sufixos e os *motifs* podem ser encontrados através de buscas pela árvore. Não há a necessidade de determinar o tamanho do *motif*. Embora eficiente, o algoritmo pode ser afetado por ruídos [Castro 12].

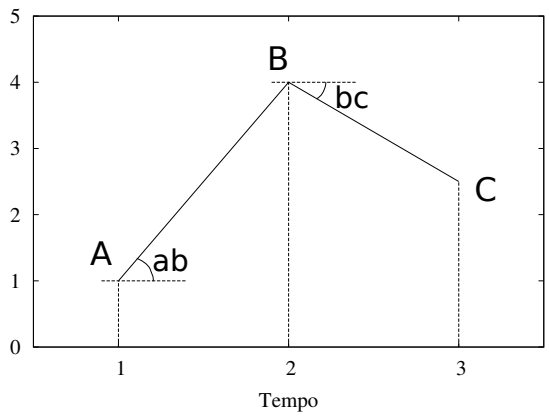


Figura 3.7: Discretização através dos ângulos

Um algoritmo para minerar *motifs* no domínio original dos dados através de amostragem aleatória foi proposto em Catalano et al. [Catalano 06]. O algoritmo retira amostras de janelas da série temporal, armazenando as janelas mais similares. Para que os padrões mais frequentes

sejam encontrados, é necessário retirar uma quantidade suficiente de amostras. O tamanho das janelas é escolhido para que seja possível encontrar qualquer padrão. Para eliminar ruídos, o algoritmo cria uma janela formada por leituras de posições aleatórias da série temporal. Assim, os padrões que forem muito similares à janela de ruídos são descartados. Por ser baseado em amostragens aleatórias e por utilizar pouca memória, o método pode apresentar resultados ruins quando os *motifs* são raros ou esparços entre si [Mohammad 09]. Nesses casos, o método necessita de uma grande quantidade de memória [Minnen 08].

Em Li e Lin [Li 10, Li 12] foi proposto um método para a descoberta de *motifs* de tamanhos variados utilizando um algoritmo baseado em compressão gramatical. O algoritmo extrai subsequências das séries temporais através de uma janela deslizante e convertendo-as para a representação SAX. Em seguida, as subsequências convertidas são concatenadas e transformadas em uma sequência de padrões. A partir dessa sequência de padrões, o algoritmo induz as regras gramaticas, que darão origem aos padrões frequentes. Segundo os autores, por induzir regras gramaticais para a construção da sequência de padrões, o método permite revelar o comportamento hierárquico dos dados, bem como determinar intuitivamente a relação entre padrões. Outra vantagem do método proposto por Li e Lin é a sua capacidade em minerar *motifs* de diferentes tamanhos em uma única passagem pela série temporal. Entretanto, o método apresenta dificuldades em encontrar padrões que possuam subsequências próximas dos intervalos de extração [Li 10].

3.4 Pares de Padrões Mais Próximos

Assim como na descoberta de *motifs* frequentes, o par de padrões mais próximos também é previamente desconhecido. Portanto, é necessário verificar todas as subsequências para encontrar o par que apresente a menor distância e que sejam vizinhos não-triviais. Uma possível solução para esse problema é a busca exaustiva assim como no algoritmo de força bruta para os *motifs* frequentes. O algoritmo de força bruta para o par de vizinhos mais próximos também apresenta complexidade quadrática com o tamanho da série temporal e está representado no Algoritmo 3.2. O algoritmo calcula a distância dos pares de vizinhos não-triviais e armazena o par que apresentar a menor distância. Assim como no algoritmo de força bruta para os *motifs* frequentes, seu uso é impraticável.

Yankov et al. estendeu o algoritmo de projeção aleatória para encontrar o par de vizinhos mais próximos [Yankov 07]. O método proposto por Yankov et al. segue o mesmo algoritmo utilizado em [Chiu 03], porém com algumas modificações. As subsequências da série temporal são extraídas e convertidas na representação simbólica SAX. Após essa etapa, o algoritmo utiliza amostragens aleatórias para contabilizar na matriz de colisão os pares de subsequências que contenham os mesmos valores. Ao final do algoritmo, é realizado o cálculo da distância no domínio original da série temporal apenas para os pares de subsequências que apresenta-

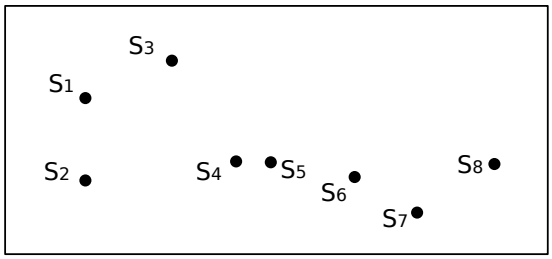
Algoritmo 3.2 Algoritmo de Força Bruta para descobrir o *Motif* de Vizinhos mais Próximos

Entrada: T, n
Saída: L_1, L_2 //Posições do par de vizinhos mais próximos

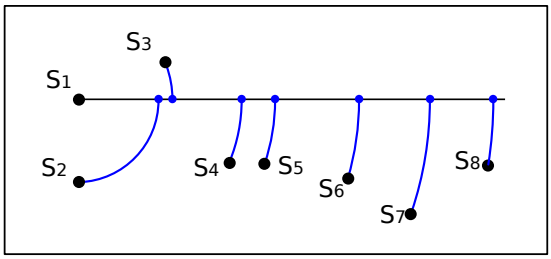
```
menorDistancia = ∞;
L1 = L2 = NULL;
para i = 1 até tamanho(T) - n + 1 faça
  para j = 1 até tamanho(T) - n + 1 faça
    se vizinhoNãoTrivial(Ti, Tj) < menorDistancia então
      menorDistancia = vizinhoNãoTrivial(Ti, Tj);
      L1 = i;
      L2 = j;
    fim se
  fim para
fim para
```

rem valores na matriz de colisão superior a um limiar. Outra característica desse método é a capacidade de encontrar padrões semelhantes mas que possuem escalas diferentes.

Em [Mueen 09b] foi proposto o algoritmo Mueen-Keogh (MK), um algoritmo exato para a descoberta de pares de vizinhos mais próximos. O algoritmo MK utiliza uma subsequência de referência para reduzir o número de cálculos de distância. Através da subsequência de referência, as demais subsequências são ordenadas, criando uma heurística para calcular apenas os pares com maior probabilidade de serem vizinhos próximos. A idéia principal do algoritmo pode ser vista na na Figura 3.8.



(a) Subsequências representadas no espaço 2D.



(b) Projeção das subsequências para o espaço 1D utilizando a subsequência S_1 como referência.

Figura 3.8: Proposta de ordenação de sequências no espaço 2D. Adaptado de [Mueen 09b].

Na Figura 3.8a está ilustrado um conjunto de subsequências representadas no espaço bidimensional, as subsequências S_4 e S_5 representam o par de vizinhos mais próximos. Inicialmente o algoritmo assume que a melhor menor distância entre os vizinhos mais próximos é infinita ($menorDistancia = \infty$). Então, é escolhida aleatoriamente uma subsequência S_R para ser utilizada como ponto de referência, no caso da Figura 3.8b foi escolhida a subsequência S_1. As subsequências serão projetadas em um espaço undimensional utilizando a distância entre elas. As subsequências serão então ordenadas pela distância até a subsequência de referência

S_1, sendo projetadas em um espaço unidimensional. A menor distância até o momento será a distância entre a subsequência de referência e sua subsequência mais próxima, no caso do exemplo, $menorDistancia = D(S_1, S_2)$. Durante a ordenação também são calculadas as distâncias entre duas subsequências adjacentes projetadas no espaço unidimensional. Essas distâncias, embora não correspondam à distância real entre as subsequências, podem ser utilizadas como um limitante inferior, ou seja, $D(S_R, S_{i+1}) - D(S_R, S_i) \leq D(S_i, S_{i+1})$.

No próximo passo, o algoritmo MK percorre a lista ordenada de subsequências calculando a distância real entre duas subsequências adjacentes. Sempre que a distância calculada for inferior à menor distância armazenada até o momento, *menorDistancia* é atualizada. No caso do exemplo da Figura 3.8b, é calculado a distância entre as subsequências S_2 e S_3. É importante notar que, como dito anteriormente, apesar de $D(S_1, S_3) - D(S_1, S_2) < menorDistancia$, a distância real entre elas $D(S_2, S_3)$ é maior que *menorDistancia*. O algoritmo segue calculando a distância $D(S_3, S_4) > menorDistancia$, até calcular $D(S_4, S_5)$. Como $D(S_4, S_5) < menorDistancia$, *menorDistancia* é atualizada para esse valor. A partir do valor de *menorDistancia* é possível percorrer a lista de subsequências na projeção unidimensional utilizando uma janela deslizante para calcular o par de vizinhos mais próximos. Como a distância entre duas subsequências na projeção unidimensional corresponde a um limitante inferior, são verificados apenas os pares que apresentarem distância na projeção inferior à *menorDistancia*. No caso do exemplo, apenas os pares (S_2, S_3) e (S_4, S_5) respeitam essa condição e como já se sabe que $D(S_2, S_3) > menorDistancia$, o par de vizinhos mais próximos encontrados pelo algoritmo MK é par (S_4, S_5), como esperado. Para aumentar ainda mais a velocidade dos algoritmos, os autores recomendam a utilização de mais de uma subsequência de referência, permitindo assim uma melhor definição nos limitantes inferiores e evitando cálculos desnecessários da distância Euclidiana. Entretanto, o uso da distância Euclidiana diretamente no domínio original da série temporal pode apresentar problemas em dados ruidosos [Castro 12], prejudicando o algoritmo.

3.5 Considerações Finais

Nesse capítulo foram apresentados os conceitos relacionados a descoberta de *motifs* em séries temporais. A descoberta de *motifs* pode ser dividida em duas categorias, mineração de padrões frequentes e a descoberta do par de vizinhos mais próximos. Embora [Mueen 09a] considere que o problema de descoberta de pares de vizinhos mais próximos possa ser utilizado como base para encontrar os padrões mais frequentes, esse projeto de mestrado considerou ambos como problemas distintos. O par de vizinhos mais próximos não necessariamente indica que esse seja o padrão mais frequente. Já um padrão mais frequente pode não conter um par de subsequências entre os 10 vizinhos mais próximos [Castro 12]. Nesse projeto de mestrado será dada ênfase ao problema de mineração de padrões mais frequentes (Capítulo 5).

Capítulo 4

O Índice *Telesto*

4.1 Considerações Iniciais

Devido ao recente avanço tecnológico e à crescente capacidade de armazenamento dos computadores, o volume de dados relacionados a séries temporais coletado, armazenado e disponível para análise tem aumentado constante e substancialmente. Como resultado, surge o desafio de pesquisar de forma eficiente bases de dados de séries temporais. A busca por similaridade em bases de dados de séries temporais é uma ferramenta importante para extrair conhecimento dessas bases. Ela é amplamente empregada como sub-rotina em aplicações baseadas em agrupamentos [Kalpakis 01], classificação [Geurts 01] e mineração de regras de associação [Luo 04]. Por exemplo, por meio dessa busca é possível determinar, a partir de dados agrometeorológicos, regiões para as quais um determinado comportamento seja similar ao ocorrido em regiões produtoras de cana-de-açúcar em períodos próximos [Romani 10]. Neste capítulo será apresentado o índice *Telesto* (*Time Series Generalized Suffix Tree*), uma nova abordagem de indexação destinada a consultas por similaridade em bases de dados de séries temporais.

Devido a sua natureza contínua e por apresentar uma alta dimensionalidade, a tarefa de indexar séries temporais pode ser considerada um grande desafio. Para contornar esse problema, o índice *Telesto* reduz a dimensionalidade das séries temporais através de técnicas de discretização, assim, é possível tomar proveito da velocidade de consulta de uma árvore de sufixo generalizada. A partir da árvore de sufixo generalizada, o índice *Telesto* fornece consultas por similaridade em uma base de dados de séries temporais. O índice *Telesto* permite realizar consultas de subsequências por abrangência e por vizinho mais próximo. Em [Chino 12] foi preliminarmente apresentado o *Telesto*, que foi desenvolvido em colaboração com outros pes-

quisadores do GBdI.

O restante deste capítulo está organizado do seguinte modo. Na Seção 4.2 é dada uma descrição de árvore de sufixo generalizada. Na Seção 4.3 será apresentado o índice *Telesto* para realizar consultas por similaridade em uma base de dados de séries temporais. Na Seção 4.4 são apresentados os resultados experimentais utilizando dados obtidos a partir de imagens de sensores remotos. As considerações finais são apresentadas na Seção 4.5.

4.2 Árvore de Sufixo

Uma árvore de sufixo é uma estrutura capaz de indexar todos os sufixos de uma cadeia de caracteres [Weiner 73]. A árvore de sufixo permite determinar de modo rápido e eficiente a ocorrência de padrões na cadeia de caracteres. Assim, é possível utilizá-las para encontrar padrões em comum entre duas ou mais cadeias de caracteres ou repetições de padrões.

Seja $\Sigma = \{\alpha_1, \ldots, \alpha_a\}$ um alfabeto com a caracteres. Seja Σ^* um conjunto com todas as cadeias de caracteres que podem ser construídas a partir de Σ e seja $\hat{S} \in \Sigma^*$ uma cadeia com m caracteres, isto é, $|\hat{S}| = m$. Seja $\hat{S}[i : j]$ $(1 \leq i \leq j \leq m)$ uma subcadeia entre (e inclusive) o i-ésimo e j-ésimo caracteres de $\hat{S}$. Sejam todas as subcadeias $\hat{S}[1 : i]$ prefixos de S e todas as subcadeias $\hat{S}[i : m]$ sufixos de $\hat{S}$, denotado por $\hat{S}[i ::]$. Seja $\$ \notin \Sigma$ um caracter terminal que representa o final de uma cadeia de caracteres. Para que seja possível armazenar todos os sufixos de uma cadeia de caracteres $\hat{S}$ de tamanho m, a árvore de sufixo deve obedecer às seguintes propriedades [Gusfield 97]:

1. A árvore possui m nós folhas numerados de 1 a m;

2. Cada nó interno, exceto a raiz, tem pelo menos 2 filhos;

3. Cada aresta representa, por meio de seu rótulo, uma subcadeia de $\hat{S}$;

4. Duas arestas que saiam do mesmo nó não podem representar subcadeia com prefixo comum;

5. Para cada nó folha i, a concatenação dos rótulos das arestas que estão no caminho da raiz até esse nó folha forma exatamente o sufixo $\hat{S}[i ::]$.

Umas das principais características de uma árvore de sufixo é a sua capacidade em realizar consultas por padrões (subcadeias) em complexidade linear ao tamanho da subcadeia de consulta. Para realizar as consultas na árvore de sufixo, o algoritmo de busca percorre apenas as arestas que contém o mesmo prefixo que a cadeia de consulta. Na Figura 4.1 está ilustrado um exemplo para uma árvore de sufixo contendo todos os sufixos da palavra $S = banana$. No caso do exemplo da Figura 4.1, para se determinar as ocorrências da cadeia $S_q = ana$, o algoritmo de

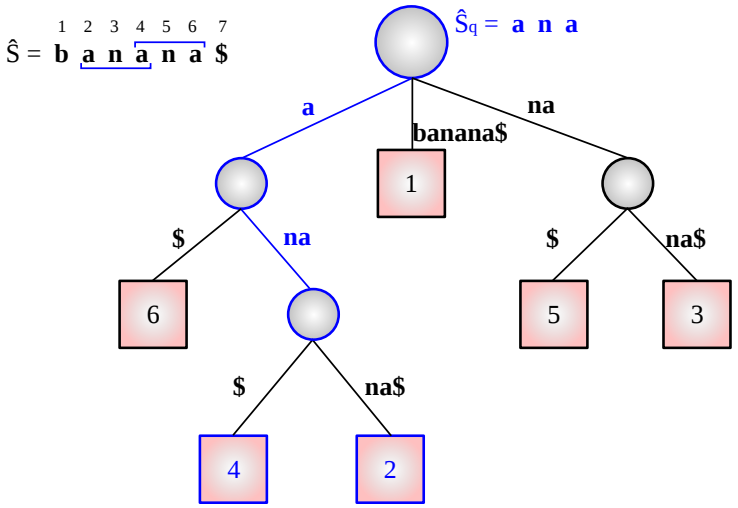


Figura 4.1: Exemplo de uma consulta da subcadeia *ana* em uma árvore de sufixo contendo a palavra *banana*.

busca percorrerá as arestas destacadas em azul e retornará como conjunto resposta os sufixos $\hat{S}[2::]$ e $\hat{S}[4::]$ também destacados em azul.

A árvore de sufixo é capaz de indexar os sufixos de apenas uma cadeia de caracteres, entretanto, é possível indexar mais cadeias de caracteres através de uma árvore de sufixo generalizada. A árvore de sufixo generalizada permite indexar um conjunto n de cadeias de caracteres $\mathcal{S} = \hat{S}_1, \ldots, \hat{S}_n$. Na árvore de sufixo generalizada são indexados todos os sufixos $\hat{S}_i[j::]$ de todas as cadeias de $\hat{S}_i \in \mathcal{S}$. Diferentemente da árvore de sufixo, os nós folhas da árvore de sufixo generalizado são rotulados por um par de inteiros $[i, j]$ que representam o sufixo $\hat{S}_i[j::]$. A Figura 4.2 ilustra uma árvore de sufixo generalizada para o conjunto $\mathcal{S} = \hat{S}_1, \hat{S}_2$, onde $\hat{S}_1 = abca\$$ e $\hat{S}_2 = bcaba\$$. O nó folha destacado em azul possui os pares $[1,4]$ e $[2,5]$, os quais representam os sufixos $\hat{S}_1[4::] = a\$$ e $\hat{S}_2[5::] = a\$$, respectivamente. O caminho a um nó (interno ou folha) X, denotado por $caminho(X)$, representa uma subcadeia formada por meio da concatenação dos rótulos que estão no caminho do nó raiz ao nó X. Por exemplo, o caminho ao nó X na Figura 4.2 representa a subcadeia bca, isto é, $caminho(X) = bca$.

O algoritmo trivial de construção de árvores de sufixo generalizada é realizado da seguinte forma. Inicialmente cria-se um nó raiz, em seguida, para cada *string* $\hat{S}_i \in T$ adicionam-se todos os sufixos de $\hat{S}_i$ na árvore. A cada iteração, conforme necessário, as arestas são divididas de forma a preservar as propriedades da árvore. Por exemplo, o algoritmo trivial para a construção da árvore de sufixo generalizada da Figura 4.2 é realizado do seguinte modo. Primeiro é criado um nó raiz, então os sufixos $\hat{S}_1[1::] = abca\$$, $\hat{S}_1[2::] = bca\$$ e $\hat{S}_1[3::] = ca\$$ são inseridos ao nó raiz, já que não há nenhum caminho na árvore com esses prefixos. Em seguida é inserido o sufixo $\hat{S}_1[4::] = a\$$ ao nó Y (sendo que $caminho(Y) = abca$), como eles compartilham o mesmo prefixo a, o nó Y é dividido e o sufixo $\hat{S}_1[4::] = a\$$ é inserido. O restante dos sufixos é inserido na árvore do mesmo modo.

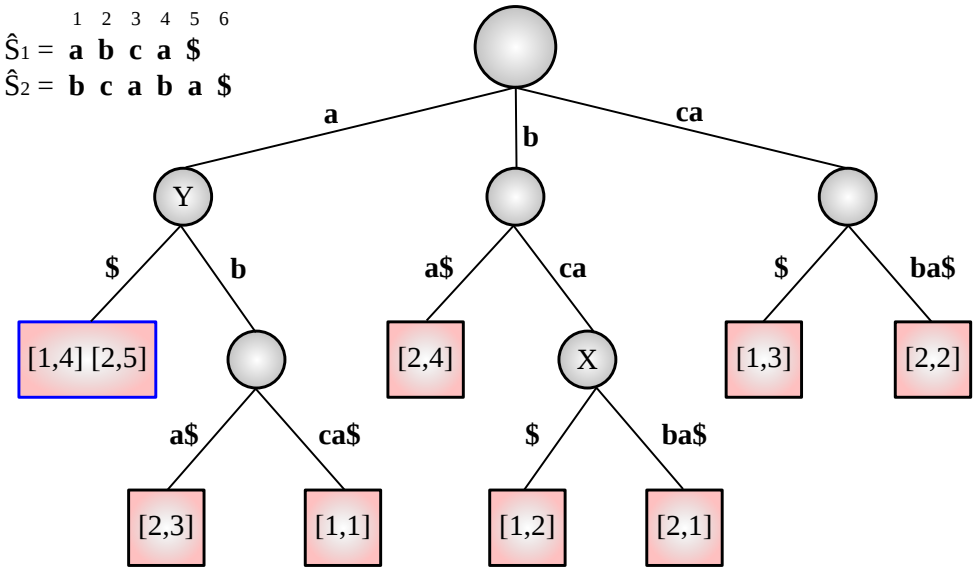


Figura 4.2: Example of a generalized Suffix Tree

As consultas por subcadeias na árvore de sufixo generalizada é realizada de maneira determinística em tempo linear ao tamanho da cadeia de consulta $\hat{S}_q$ [Gusfield 97]. A partir do nó raiz, realiza-se o casamento entre a subsequência de consulta $\hat{S}_q$ e as subcadeias contidas na árvore, até que seja encontrado um nó Y tal que $caminho(Y) = \hat{S}_q$. Todos os nós folhas que descendem de Y também apresentam ocorrências da cadeia de consulta. Por exemplo, uma consulta para a cadeia $\hat{S}_q = a$ na árvore de sufixo generalizada representada na Figura 4.2 é realizada do seguinte modo. Após encontrar o nó Y, tal que $caminho(Y) = a$, o algoritmo percorre a árvore até encontrar todos os nós folhas descendentes de Y, encontrando as ocorrências de $\hat{S}_q = a$ nos sufixos $\hat{S}_1[4::]$, $\hat{S}_2[5::]$ $\hat{S}_2[3::]$ $\hat{S}_1[1::]$.

4.3 O Índice *Telesto*

O índice *Telesto* (acrônimo para *Time Series Generalized Suffix Tree*) é um novo método para realizar consultas por similaridade em bases de dados de séries temporais de modo rápido e eficiente. O *Telesto* se aproveita da redução de dimensionalidade proporcionada por técnicas de discretização de séries temporais e da alta velocidade de consulta das árvores de sufixo. As consultas por similaridade através do índice *Telesto* ocorre em duas etapas: construção e consulta.

Durante a etapa de construção do índice *Telesto*, as séries temporais são convertidas para uma representação simbólica e em seguida indexadas em uma árvore de sufixo generalizada, como ilustra a Figura 4.3. Inicialmente a base de dados de séries temporais $\mathcal{T} = T_1, \ldots, T_n$ é recuperada do disco e em seguida cada série temporal é convertida para uma representação simbólica através da técnica SAX utilizando um alfabeto de tamanho a. Ao final desse processo, obtém-se um conjunto de séries temporais discretizadas $\mathcal{S} = \hat{S}_1, \ldots, \hat{S}_n$. Então, as séries tempo-

rais discretizadas são indexadas em uma árvore de sufixo generalizada. A construção da árvore de sufixo generalizada é realizada por meio do algoritmo de construção trivial apresentado na Seção 4.2.

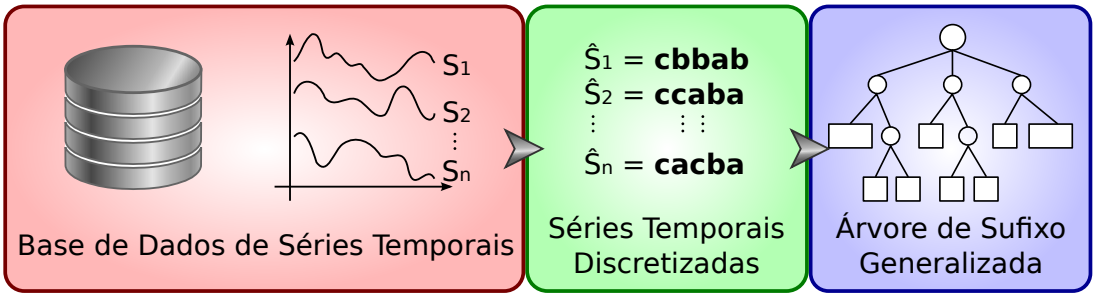


Figura 4.3: Etapa de construção do índice *Telesto*.

A etapa de consulta ilustrada na Figura 4.4 é a responsável por realizar as consultas por similaridade entre as séries temporais discretizadas por meio de casamentos de subsequências. Seja $T_q = \{t_1, \ldots, t_m\}$ uma série temporal de consulta. No primeiro passo dessa etapa, a série temporal T_q é convertida para representação simbólica $\hat{S}_q = s_1 \ldots s_w$ utilizando a técnica SAX com os mesmo parâmetros utilizados na etapa de construção. Em seguida, é realizada uma consulta por similaridade na árvore de sufixo generalizada, gerando um conjunto resposta de séries temporais. Ao realizar uma consulta por abrangência, o índice *Telesto* recupera todas as séries temporais $T_i \in \mathcal{T}$ que estejam a uma distância $D(T_q, T_i) < R$, sendo R o raio de abrangência. Já para a consulta por vizinho mais próximo, o índice *Telesto* recupera a série temporal mais próxima de T_q.

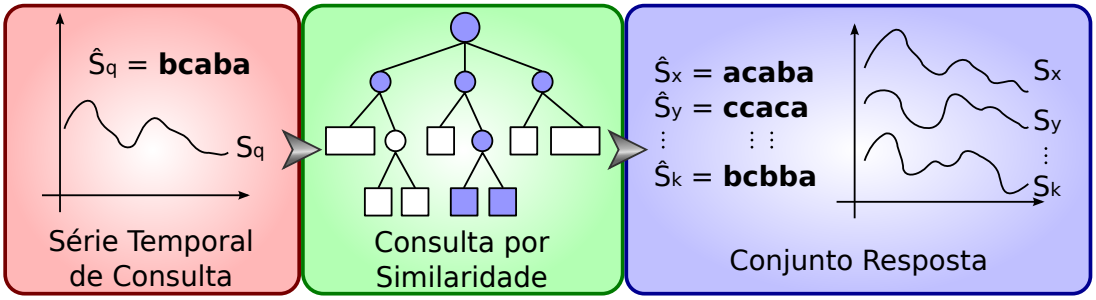


Figura 4.4: Etapa de consulta do índice *Telesto*.

Para comparar séries temporais discretizadas por meio da técnica SAX, isto é, para calcular as distâncias $D(T_i, T_j)$ entre essas séries, *Telesto* usa como base a função de distância $MINDIST(\hat{S}_i, \hat{S}_j)$ (Equação 2.17). Como resultado, a consulta por abrangência em árvores de sufixo generalizadas proposta pelo Telesto estende a busca por subcadeias na árvore de sufixo generalizada da seguinte forma. Seja $\hat{S}_q = s_1 \ldots s_w$ uma série temporal de consulta discretizada. A consulta por abrangência por $\hat{S}_q$ busca todas as subcadeias de tamanho m na árvore de sufixo generalizada. Durante a busca, para cada nó interno X percorrido na árvore, tem-se a subcadeia

$caminho(X) = S_i[1:k]$ correspondente. Caso $MINDIST(\hat{S}_i[1:k], \hat{S}_q[1:k]) > R$, a busca é interrompida, isto é, realiza-se uma poda na busca, e descarta-se $S_i[1:k]$. Caso a busca encontre uma subcadeia com m caracteres $\hat{S}_i[1:m]$ e $MINDIST(\hat{S}_i[1:m], \hat{S}_q[1:m]) \leq R$, $\hat{S}_i[1:m]$ é inserida no conjunto resposta.

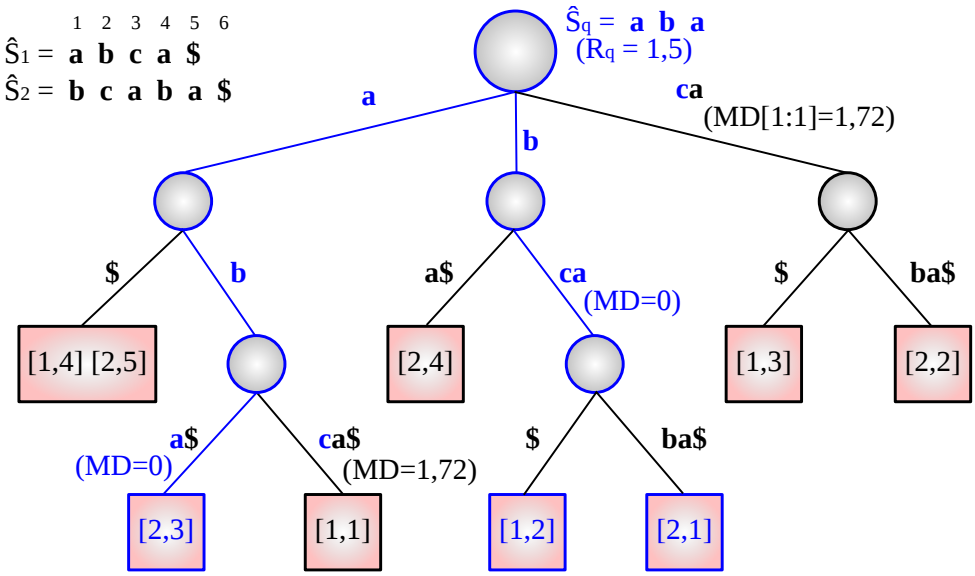


Figura 4.5: Consulta por abrangência utilizada no índice *Telesto*.

Na Figura 4.5 está ilustrado um exemplo de como a consulta por abrangência é realizada. Sejam as séries temporais discretizadas $\hat{S}_1 = abca\$$ e $\hat{S}_2 = bcaba\$$ indexadas no índice Telesto, a série temporal discretizada de consulta $\hat{S}_q = aba$ e um raio de abrangência $R = 1,5$. Durante a consulta por abrangência o algoritmo de busca percorrerá todos os possíveis caminhos, descartando os caminhos em que $MINDIST(\hat{S}_i[1:k], \hat{S}_q[1:k]) > 1,5$. No caso do exemplo, apenas os caminhos destacados em azul são percorridos, visto que $MINDIST(abc, \hat{S}_q[1:3] = aba) > 1,5$ e $MINDIST(c, \hat{S}_q[1:1] = a) > 1,5$. Assim, ao final do processo, serão retornadas as subcadeias $\hat{S}_2[3:5] = aba$ e $\hat{S}_1[2:4] = \hat{S}_2[1:3] = bca$.

Para a consulta por vizinhos mais próximos, o raio de abrangência *RNN* é definido dinamicamente a medida que a consulta é realizada. Inicialmente o raio de abrangência é ajustado para $RNN = \infty$. Similarmente à consulta por abrangência, o algoritmo de busca do *Telesto* percorrerá todos os possíveis caminhos na árvore de sufixo generalizada calculando a $MINDIST(\hat{S}_i[1:k], \hat{S}_q[1:k])$. Sempre que for calculado uma $MINDIST(\hat{S}_i[1:k], \hat{S}_q[1:k])$ que seja menor que o raio de abrangência *RNN*, o raio de abrangência é atualizado para $RNN = MINDIST(\hat{S}_i[1:k], \hat{S}_q[1:k])$. Assim como na consulta por abrangência, os caminhos são podados sempre que $MINDIST(\hat{S}_i[1:k], \hat{S}_q[1:k]) > RNN$.

É importante ressaltar que, como o *Telesto* está baseado na representação SAX e em sua função de distância *MINDIST*, as consultas por similaridade não descartarão falsos negativos no conjunto resposta, visto que a *MINDIST* é um limitante inferior da distância Euclidiana. Em contrapartida, podem ocorrer falsos positivos, criando uma necessidade de uma fase posterior

de refinamento. Portanto, o *Telesto* deve ser considerado como um filtro, gerando apenas um conjunto de séries temporais candidatas. Consequentemente, deve ser realizada uma verificação da distância Euclidiana entre a série temporal de consulta e as séries retornadas pelo *Telesto* no domínio original das séries.

4.4 Teste de Desempenho

Para validar o índice *Telesto*, foram realizados experimentos utilizando uma base de dados reais de séries temporais. Foram utilizadas séries temporais extraídas das imagens de satélite AVHRR/NOAA[1]. Os valores presentes nas séries temporais correspondem às medidas mensais de NDVI[2] contidos nos *pixels* das imagens. Foram utilizadas imagens mensais da região do estado de São Paulo correspondentes ao período de abril de 2001 a março de 2010, totalizando 250.000 séries temporias de tamanho 108.

Foram realizados experimentos para avaliar o tempo de construção do índice *Telesto* e também foram realizados testes de desempenho para as consultas por abragência e vizinho mais próximo. Os experimentos foram realizados variando-se o tamanho da base de dados entre 50.000 e 250.000 séries temporais e variou-se também o tamanho das séries temporais de consulta. Para os testes de desempenho das consultas, foram geradas séries temporais de consulta, sendo essas separadas em 10 grupos. Cada grupo de consulta possui 5 séries temporais discretizadas de tamanho 12, 24, 36, 48 e 60, correspondendo às séries temporais de 1, 2, 3, 4 e 5 anos respectivamente. As séries temporais foram discretizadas utilizando a técnica SAX com $a = 5$ símbolos e $w = 108$. Para a consulta por abrangência, foi utilizada a taxa de erro $\varepsilon = 0,005$ para calcular o raio de abrangência R das consultas, tal que $\varepsilon = R/|S_q|$, sendo $|S_q|$ o tamanho da série de consulta. A etapa de consulta proposta pelo método *Telesto* foi comparada com a busca sequencial na base de dados de séries temporais já discretizadas. Foram analisados os tempos gastos durante a etapa de construção do índice *Telesto* e também o tempos gastos para o processamento das consultas através do índice *Telesto* e da busca sequencial. Todos os testes foram executados 5 vezes, portanto, os valores apresentados e discutidos nessa seção correspondem à média dessas 5 execuções. Os testes foram realizados em um computador executando o sistema operacional Linux (32 bits) com um processador *Intel Core i7* 2,67 Ghz, 12 GB de memória primária e 2 TB de memória secundária. Ambos os métodos, Telesto e busca sequencial, foram implementados para memória primária usando a linguagem de programação C++ considerando apenas a fase de filtragem.

O primeiro experimento visou avaliar o tempo necessário para a construção do índice *Telesto* variando-se o tamanho da base de dados. Na Figura 4.6 é possível observar o tempo gasto durante a etapa de construção do índice *Telesto*. Pode-se observar um crescimento linear do tempo gasto de acordo com o aumento do tamanho da base de dados. A partir desse resultado,

[1] *Advanced Very High Resolution Radiometer/National Oceanic and Atmospheric Administration*
[2] *Normalized Difference Vegetation Index*

é possível afirmar que a etapa de construção do índice *Telesto* é escalável, visto que o aumento no tamanho da base de dados não degradou seu desempenho. É importante notar que, apesar de serem necessários aproximadamente 70 segundos para se construir o índice para uma base de dados de tamanho 250.000, a busca sequencial nessa mesma base leva em média 9 segundos para realizar uma única consulta. Assim, caso seja necessário realizar múltiplas consultas, o tempo gasto na etapa de construção do índice *Telesto* pode ser considerado desprezível.

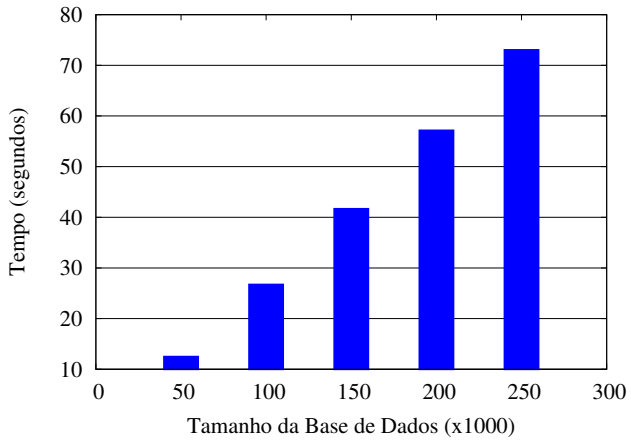
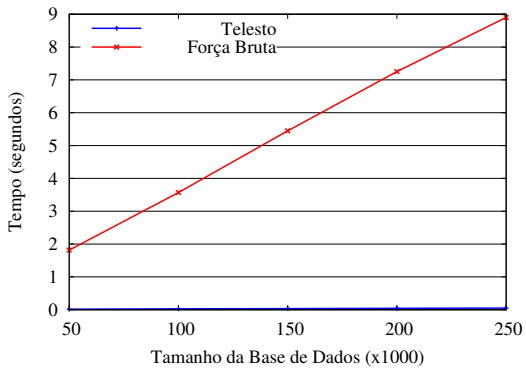


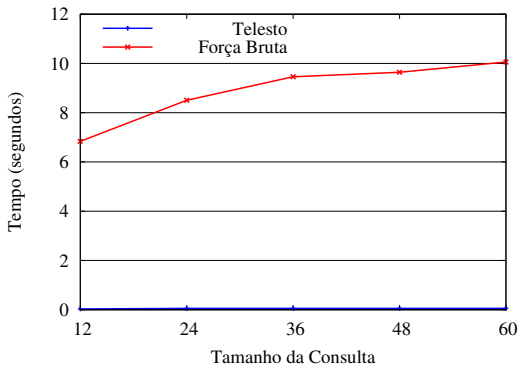
Figura 4.6: Tempo gasto para a construção do índice *Telesto*.

No segundo conjunto de experimentos, foram realizados testes de desempenho para as consultas por abrangência e vizinho mais próximo. Para ambas as consultas foram realizados experimentos para avaliar o impacto causado pela aumento do tamanho da base de dados, como também o aumento do tamanho da consulta. Para os testes de desempenho variando-se o tamanho da série temporal de consulta foi utilizada uma base de dados de tamanho 250.000.

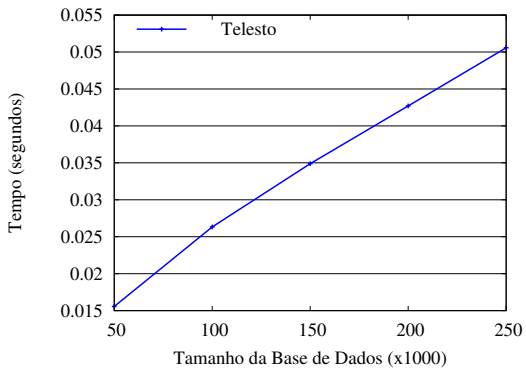
Na Figura 4.7 estão ilustrados os resultados obtidos para o teste de desempenho da consulta por abrangência. Nas Figuras 4.7a e 4.7b está sendo apresentado o resultado comparativo entre as consultas utilizando o índice *Telesto* e a busca sequencial. Como o índice *Telesto* apresentou resultados na ordem de milésimos de segundos, a escala apresentada nas Figuras 4.7a e 4.7b não permite uma análise do comportamento ao se utilizar o índice. Assim, nas Figuras 4.7c e 4.7d estão sendo apresentados apenas os resultados obtidos utilizando o índice *Telesto*. Pode-se observar que em ambos os casos, o desempenho ao se utilizar o índice *Telesto* foi muito superior à busca sequencial. O índice *Telesto* proveu uma melhora de desempenho que variou de 116 a 176 vezes quando considerado com o impacto do aumento do tamanho da base, e que variou de aproximadamente 180 a 220 vezes quando considerado o impacto do aumento do tamanho da consulta. O *Telesto* também apresentou um custo linear em relação ao tamanho da base para realizar consultas por abrangência (Figura 4.7c). Ainda mais importante, com exceção da consulta de tamanho 12 que apresentou tempo gasto bastante reduzido, pode-se observar que o tempo gasto pelo *Telesto* no processamento da consulta foi praticamente indiferente ao tamanho



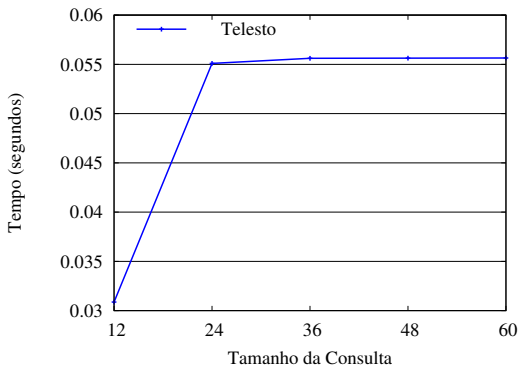
(a) Variando-se o tamanho da base de dados.



(b) Variando-se o tamanho da consulta.



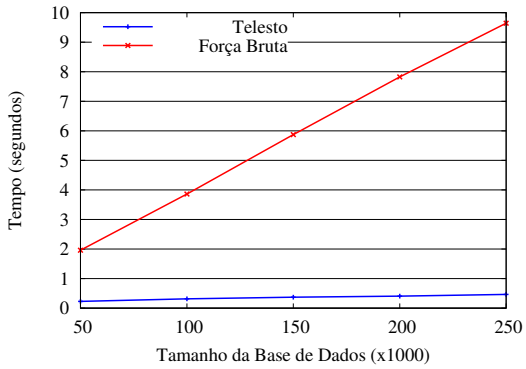
(c) Variando-se o tamanho da base de dados. (Apenas o resultado obtido utilizando-se o índice *Telesto*)



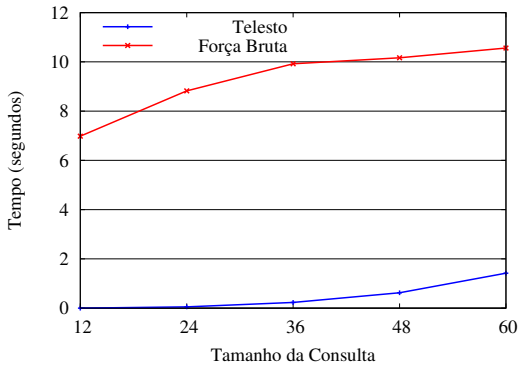
(d) Variando-se o tamanho da consulta. (Apenas o resultado obtido utilizando-se o índice *Telesto*)

Figura 4.7: Tempo de resposta para uma consulta por abrangência utilizando-se o índice *Telesto*.

da consulta utilizada (Figura 4.7d).



(a) Variando-se o tamanho da base de dados.



(b) Variando-se o tamanho da consulta.

Figura 4.8: Tempo de resposta para uma consulta por vizinho mais próximo utilizando-se o índice *Telesto*.

Os resultados obtidos para o teste de desempenho para as consultas por vizinho mais pró-

ximo estão ilustrados na Figura 4.8. Novamente, em ambos os casos o *Telesto* apresentou um desempenho superior à busca sequencial. O índice *Telesto* apresentou uma melhora de até 20 vezes quando considerado o impacto do aumento do tamanho da base de dados e de até 168 vezes quando considerado o aumento do tamanho da série temporal de consulta. Assim como na consulta por abrangência, o índice *Telesto* apresentou um crescimento linear em relação ao tamanho da base para realizar consultas por vizinhos mais próximos (Figura 4.8a). Diferentemente da busca por abrangência, a busca por vizinhos mais próximos não se comportou indiferente ao tamanho da consulta (Figura 4.8b). Esse comportamento para as buscas por vizinho mais próximo se dá ao fato de que o raio de abrangência para essa consulta não é fixo. Desse modo, ao realizar a busca, mais ramos da árvore podem ser visitados até que o raio de abrangência convirja para a distância do vizinho mais próximo. Entretanto, é possível afirmar que o índice *Telesto* é escalável para consultas por vizinhos mais próximos aumentando-se o tamanho da consulta.

4.5 Considerações Finais

Neste capítulo foi apresentado o índice *Telesto*, uma nova abordagem de indexação destinada a consultas por similaridade em séries temporais. O *Telesto* foi descrito em termos de suas principais características:

- **Etapa de construção:** etapa em que as séries temporais são convertidas para uma representação simbólica e então indexadas através de uma árvore de sufixo generalizada;
- **Etapa de consulta:** voltada à realização de consultas por similaridade entre séries temporais utilizando como base a velocidade de consulta de uma árvore de sufixo generalizada.

Foram realizados experimentos utilizando séries temporais sobre bases de dados reais de séries temporais, sendo que o índice *Telesto* se mostrou rápido e eficaz. Os resultados obtidos mostraram que as etapas de construção e consultas providas pelo índice *Telesto* são escaláveis, visto que apresentaram um crescimento linear do tempo gasto de acordo com o aumento do tamanho da base de dados. Os resultados também mostraram que o tempo gasto no processamento de consultas por abrangência é praticamente indiferente ao tamanho da consulta. Comparado à busca sequencial, o índice *Telesto* apresentou um ganho de desempenho de até 220 vezes para as consultas por abrangência e de até 168 vezes para as consultas por vizinho mais próximo. Finalizando, apesar de o índice *Telesto* se apresentar como uma etapa de filtragem para realizar consultas por similaridade em séries temporais. Sua eficácia se mostrou capaz de agilizar técnicas de mineração de padrões em séries temporais.

Capítulo 5

O Algoritmo *TrieMotif*

5.1 Considerações Iniciais

Neste capítulo será apresentado o *TrieMotif*, um novo algoritmo para a descoberta de padrões frequentes (*motifs*). O *TrieMotif* realiza uma seleção de possíveis candidatos a casamento para cada iteração do algoritmo, reduzindo assim a quantidade de cálculos de distâncias realizada. Para tanto, o *TrieMotif* indexa todas as subsequências de uma série temporal e seleciona os candidatos através de uma busca por abrangência. Adicionalmente, o *TrieMotif* se mostrou rápido e apresentou um baixo custo em termos de memória. Este algoritmo será apresentado como artigo científico completo no ICEIS 2014 [Chino 14].

O restante deste capítulo está organizado do seguinte modo. Na Seção 5.3 é apresentado o algoritmo *TrieMotif* para a descoberta de padrões frequentes em séries temporais. Nas Seções 5.4 e 5.5 são apresentados os resultados experimentais utilizando, respectivamente, dados sintéticos de passeio aleatório e dados obtidos a partir de imagens de sensores remotos. As considerações finais são apresentadas na Seção 5.6.

5.2 O Algoritmo *TrieMotif*

5.3 O Algoritmo *TrieMotif*

Como visto no Capítulo 3 o maior custo para um algoritmo de descoberta de *motifs* se encontra no excessivo número de cálculos de distância realizados. Para uma série temporal de tamanho m, o algoritmo exato (força bruta) realiza $O(m)$ cálculos de distância para cada iteração. Com esse objetivo, o *TrieMotif* busca reduzir a quantidade de subsquências a ser comparada em

cada iteração. Para tanto, define-se um espaço de busca, de modo que contenha apenas as subsequências com um comportamento similar à subsequência de consulta, como ilustrado na Figura 5.1. Desse modo, as subsequências comparadas apresentarão uma maior probabilidade de formarem um casamento e aquelas que apresentarem um comportamento muito discrepante da subsequência de consulta serão descartadas.

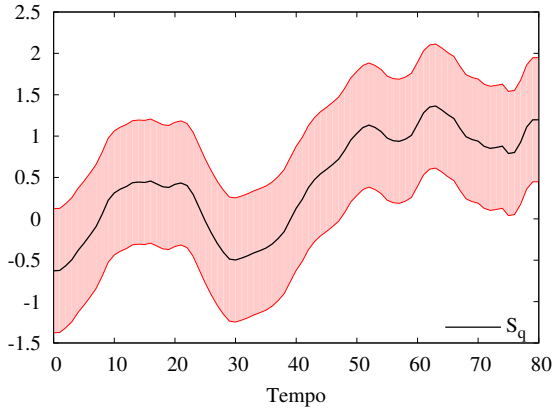


Figura 5.1: As subsequências presentes na região destacada apresentam um comportamento similar à subsequência de consulta S_q.

Entretanto, é necessário que a seleção de candidatos seja realizada de modo rápido e eficiente. Assim, o *TrieMotif* utiliza uma estrutura baseada na *trie* para indexar todos os padrões da série temporal. Para realizar a descoberta de *motifs*, o algoritmo *TrieMotif* particiona o problema em três etapas, como ilustrado na Figura 5.2:

- **Azul**: Primeiro, todas as subsequências são extraídas da série temporal e convertidas para uma representação simbólica;
- **Vermelho**: As subsequências são indexadas através de uma estrutura *trie* e uma lista de possíveis candidatos a casamento é gerada para cada subsequência de consulta;
- **Verde**: Por último, verifica-se no domínio original da série temporal quais candidatos formam de fato um casamento não trivial com a subsequência de consulta.

Na primeira etapa, todas as subsequências S_i de tamanho n são extraídas através de uma janela deslizante. Após a extração, as subsequências são z-normalizadas e sua dimensionalidade é reduzida para $w < n$ através do algoritmo PAA. O último processo dessa etapa consiste na transformação das subsequências para uma representação simbólica. Para obter essa representação, divide-se o intervalo de valores da subsequência em a caixas e para cada caixa é designado um símbolo. Em seguida, os valores da subsequências são convertidos para os símbolos correspondentes à caixa que pertecem. A Figura 5.3 ilustra um exemplo de uma subsequência S de tamanho $n = 128$ possuindo valores no intervalo $[-2,83; 1,36]$ sendo convertida para uma

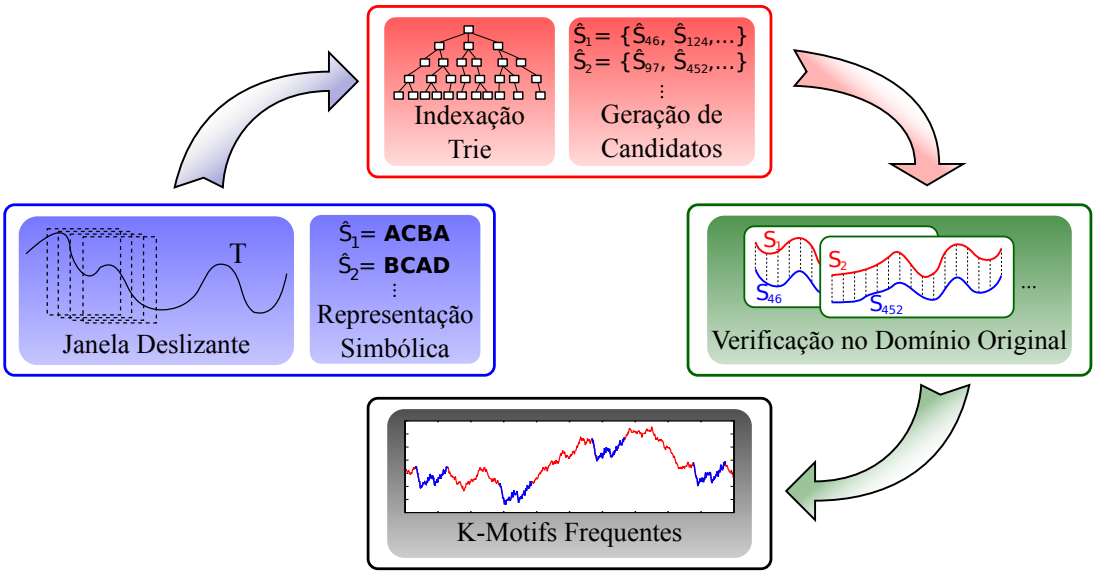


Figura 5.2: As etapas realizadas pelo *TrieMotif* para a descoberta de *motifs*.

representação simbólica de caixas utilizando $w = 8$ e $a = 3$. Inicialmente, é aplicado o algoritmo PAA na subsequência S (em vermelho), que passa a ser representada por uma série de tamanho $w = 8$ (em azul). O intervalo de valores é então dividido em $a = 3$ caixas: $A = [-2.83, -1.43)$, $B = [-1.43, -0.03)$ e $C = [-0.03, 1.36]$. Cada valor da representação PAA é então convertido para o símbolo da caixa correspondente. Ao final do processo, a subsequência S é convertida para a representação $\hat{S} = ABCBCCCB$.

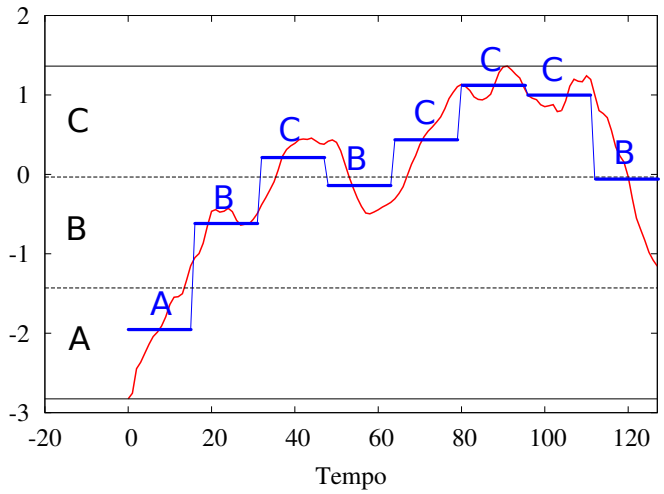
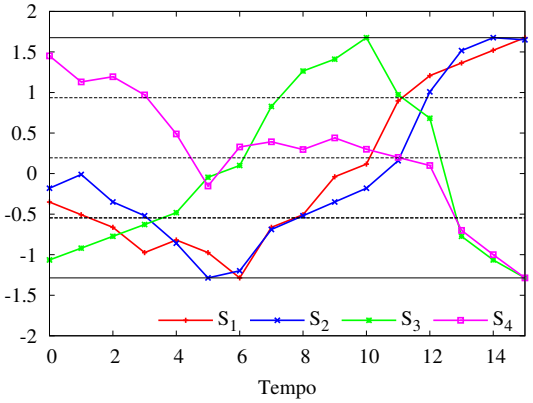
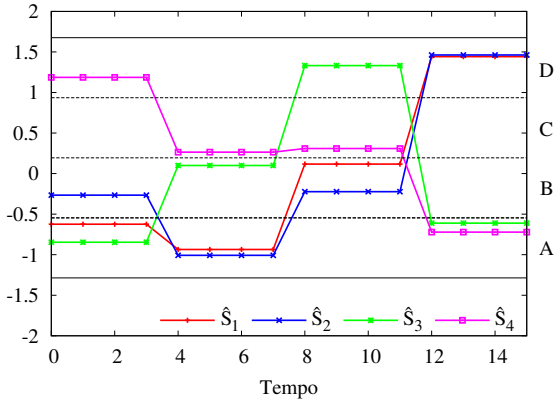


Figura 5.3: Conversão de uma subsequência de tamanho $n = 128$ para a representação simbólica de $a = 3$ caixas de tamanho $w = 8$.

Com as subsequências já extraídas e convertidas para a representação simbólica de caixas, é dado início à segunda etapa do algoritmo *TrieMotif*. Um dos problemas presentes no algoritmo de força bruta é a excessiva quantidade de cálculos de distância (verificação de casamentos) entre as subsequências. Para cada subsequência de consulta S_q é necessário



(a) Subsequências no domínio original



(b) Representação PAA das subsequências e a definição das caixas.

Figura 5.4: As subsequências S_1, S_2, S_3 e S_4 convertidas para as representações simbólica $\hat{S}_1 = AABD$, $\hat{S}_2 = BABD$, $\hat{S}_3 = ABDA$ e $\hat{S}_4 = DCCA$ respectivamente, utilizando como parâmetro $w = 4$ e $a = 4$.

calcular as distâncias com todas as subsequências da série temporal. Para contornar esse problema, o algoritmo *TrieMotif* reduz o número de cálculos de distância através da seleção de candidatos com maior probabilidade de formar um casamento. As subsequências (já em suas representações simbólicas) são indexadas em uma estrutura baseada na *trie* e a escolha de candidatos é realizada de modo eficiente através de uma busca por abrangência na estrutura *trie*.

Uma das grandes vantagens de se utilizar uma estrutura de indexação como a *trie* é a capacidade de realizar buscas de modo eficiente, já que uma busca exata possui complexidade linear em relação ao tamanho da subsequência. Entretanto, a busca exata pode apresentar problemas para o processo de seleção de candidatos, visto que é possível que ocorra o descarte de candidatos que formam um casamento com a subsequência de consulta. Esse descarte ocorre pois após o processo de conversão simbólica, subsequências com comportamentos similares podem apresentar diferentes representações simbólicas. Por exemplo, considere as subsequências S_1, S_2, S_3 e S_4 de tamanho $m = 16$ com valores no intervalo $[-1,28; 1,67]$ (Figura 5.4a). O intervalo de valores será dividido em $a = 4$ caixas $A = [-1.28; -0.54]$, $B = [-0.54; 0.19]$, $C = [0.19; 0.93]$ e $D = [0.93; 1.67]$ que serão utilizadas para converter as subsequências nas representações simbólicas $\hat{S}_1 = AABD$, $\hat{S}_2 = BABD$, $\hat{S}_3 = ABDA$ e $\hat{S}_4 = DCCA$ (Figura 5.4b). As subsequências S_1 e S_2 apresentam um comportamento bastante similar e possivelmente formam um casamento. Entretanto, note que ao serem convertidas para a representação simbólica de caixas, as subsequências passam a ser representadas por cadeias de caracteres diferentes. Embora S_1 e S_2 formem um casamento, suas representações simbólicas são diferentes, logo, uma busca exata para $\hat{S}_1$ não retornaria a subsequência $\hat{S}_2$, que passaria a ser ignorada na etapa seguinte do algoritmo. Para contornar esse problema, a busca exata na *trie* foi modificada para apresentar um comportamento similar a uma busca em abragência.

Na busca exata, quando o algoritmo de busca processa o j-ésimo elemento de uma subsequência $\hat{S}_q$ ($\hat{S}_q[j]$), apenas o caminho em que o símbolo do nó interno da *trie* ($Simbolo_{No}$) for igual ao símbolo de $\hat{S}_q[j]$ é percorrido, ou seja, apenas se $Simbolo_{No} = \hat{S}_q[j]$. No algoritmo de busca por abrangência do *TrieMotif* são associados valores numéricos para os símbolos, por exemplo, $A = 1$, $B = 2$ e assim por diante. Então, ao invés de percorrer apenas o caminho em que o símbolo do nó interno da *trie* for igual ao da subsequência de consulta, o algoritmo também percorre os caminhos com os símbolos que apresentam uma diferença menor que um certo limiar δ, ou seja, $|\hat{S}_q[j] - Simbolo_{No}| \leq \delta$. Por exemplo, se $\hat{S}_q[j] = B$ e $\delta = 1$, o algoritmo percorre os caminhos em que $Simbolo_{No} = A$, $Simbolo_{No} = B$ ou $Simbolo_{No} = C$ (um acima e um abaixo).

Percorrer um maior número de caminhos pode acarretar em um aumento exponencial do número de nós visitados, portanto, faz-se necessário realizar uma poda para que caminhos desnecessários não sejam percorridos. Como as séries temporais são um conjunto de medidas realizadas ao longo do tempo, os valores medidos em curtos intervalos de tempo tendem a apresentar valores similares. Por exemplo, se em uma dada série temporal um valor é representado pelo símbolo B, provavelmente o próximo valor estará entre os símbolos A, B ou C. Assim, pode-se evitar que caminhos desnecessários sejam percorridos através de uma indexação das representações simbólicas de um modo não sequencial. No algoritmo *TrieMotif*, os elementos são indexados intercalando-se entre elementos do ínicio e do final da subsequência, ou seja, primeiro é indexado o elemento inicial, em seguida o último, então o segundo elemento, o penúltimo e assim por diante. Desse modo, evita-se que caminhos de subsequências muito discrepantes mas com comportamento inicial semelhante sejam percorridos logo no início da busca.

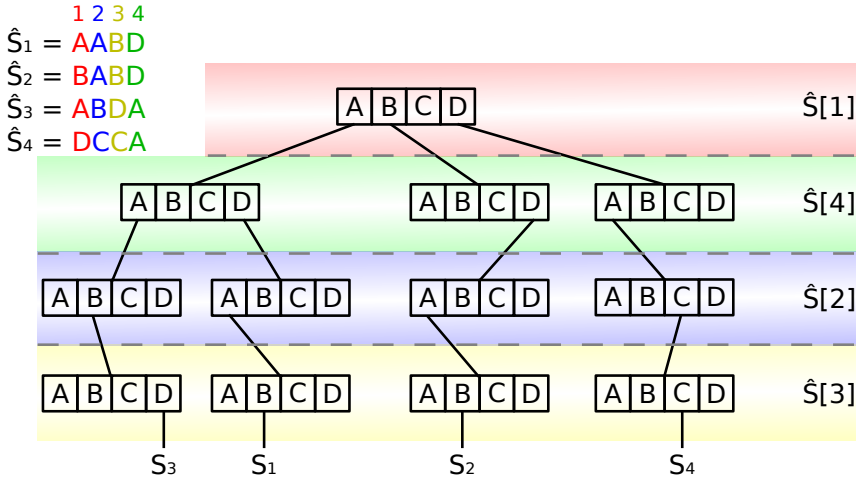


Figura 5.5: As representações simbólicas são indexadas em ordem não sequencial para diminuir a quantidade de caminhos percorridos durante a busca por abrangência.

A Figura 5.5 ilustra o modo como o algoritmo *TrieMotif* indexa as subsequências S_1, S_2, S_3 e S_4 da Figura 5.4a. No primeiro nível da *trie* (vermelho), estão indexados os caminhos

correspondentes ao primeiro símbolo de cada subsequência. No segundo nível (verde) estão os caminhos para o último símbolo, no terceiro nível (azul) estão os caminhos para o segundo símbolo e no último nível (amarelo) os caminhos para o terceiro símbolo. Já na Figura 5.6 é possível observar como a busca por abrangência do *TrieMotif* se comporta para encontrar os candidatos para a subsequência de consulta $\hat{S}_1$. No primeiro nível, como $\hat{S}_1[1] = A$, o algoritmo percorre os caminhos correspondentes aos símbolos A e B, marcados em vermelho. No nível seguinte, $\hat{S}_1[4] = D$ e são percorridos os caminhos de C e D. Esse processo é repetido até que um nó folha seja alcançado e então as subsequências encontradas são retornadas. No caso do exemplo, são retornadas as subsequências $\hat{S}_1$ e $\hat{S}_2$. Note também, que os caminhos para as subsequências $\hat{S}_3$ e $\hat{S}_4$ não são percorridos. É importante ressaltar que se caso as subsequências fossem indexadas em ordem sequencial, como ilustrado na Figura 5.7, essa mesma busca por abrangência também percorreria parte do caminho para a subsequência $\hat{S}_3$, marcado em azul, o que não ocorreu ao se indexar em ordem não sequencial.

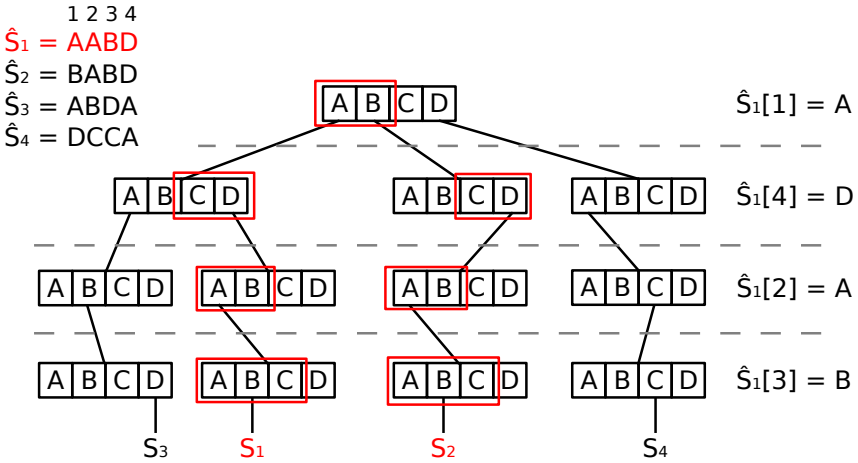


Figura 5.6: Através de uma busca na *trie*, é possível obter apenas candidatos a casamento, descartando as subsequências com comportamentos diferentes da subsequência de consulta.

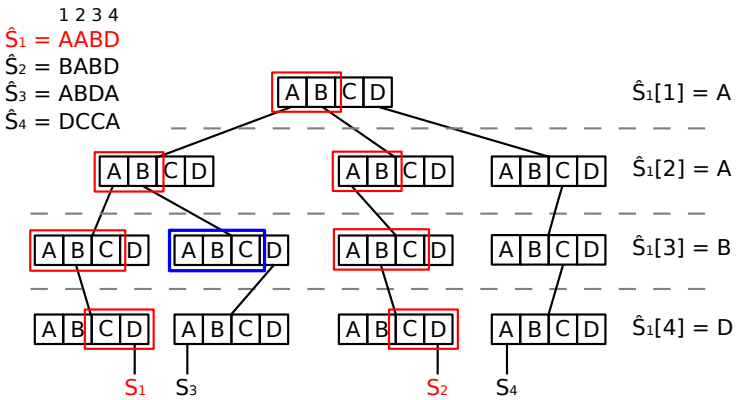


Figura 5.7: Utilizando uma indexação em ordem sequencial, o número de caminhos percorridos durante a busca pode aumentar. Diferentemente da indexação não sequencial, o caminho para a subsequência $\hat{S}_3$ marcado em azul também é percorrido.

A última etapa do algoritmo, consiste em calcular as distâncias para cada subsequência de consulta S_q com seus respectivos candidatos a casamento S_i obtidos na etapa anterior. Durante essa etapa, as distâncias entre as subsequências são calculadas no domínio original da série temporal. Assim, as subsequências que formarem um casamento não-trivial, ou seja, $D(S_q, S_i) \leq R$ e sem sobreposição, são relacionadas à subsequência de consulta S_q, já as que não satisfizerem essa condição são descartadas. Para melhorar a eficiência do algoritmo, os cálculos de distâncias podem ser interrompidos antecipadamente caso a somatória parcial da distância seja superior ao raio de abrangência R. Para cada subsequência S_q, é armazenada a quantidade de casamentos não-triviais e suas respectivas posições na série temporal. As subsequências S_q são então ordenadas em ordem decrescente de acordo com a quantidade de casamentos não-triviais. Ao final do algoritmo *TrieMotif*, são retornados os *K-Motifs* mais frequentes, tomando as devidas providencias para que a Definição 3.4 seja satisfeita.

Embora tenha sido apresentado um método de discretização para as subsequências, o algortimo *TrieMotif* também pode ser utilizado com outras representações simbólicas, como por exemplo a técnica SAX. Nesse caso, e considerando também que a distância utilizada é a Euclidiana, é possível se aproveitar da baixa complexidade da $MINDIST$ e evitar cálculos da distância no domínio original da série temporal sempre que $MINDIST(\hat{S}_q, \hat{S}_i) > R$, visto que a $MINDIST$ é um limitante inferior da distância Euclidiana.

O algoritmo *TrieMotif* para encontrar os *K-Motifs* mais frequentes está ilustrado no Algoritmo 5.1. Primeiro, todas as subsequências S_i são convertidas para uma representação simbólica $\hat{S}_i$. Durante esse processo, as subsequências S_i também são z-normalizadas. Em seguida, as subsequências simbólicas $\hat{S}_i$ são indexadas na estrutra *trie*. Através da *trie* é possível reduzir o número de distâncias calculadas utilizando-se apenas o conjunto $Candidatos$ de possíveis candidatos a casamento para cada subsequência de consulta $\hat{S}_q$. Em seguida, verifica-se no domínio original da série temporal quais subsequências $S_i \in Candidatos$ formam um casamento não-trivial com a subsequência de consulta S_q. Como não é possível saber previamente quais subsequências serão os K padrões mais frequentes, é necessário que todas as subsequências sejam computadas. As subsequências computadas são então armazenadas em uma lista de *motifs* ($ListaDeMotifs$). Na última fase do algoritmo, é também verificado se os *K-Motifs* mais frequentes satisfazem a Definição 3.4. Essa verificação é realizada ordenando-se a $ListaDeMotifs$ em ordem decrescente pela quantidade de casamentos não-triviais, de modo que os *motifs* mais frequentes fiquem no começo da lista. Assim, é possível percorrer a $ListaDeMotifs$ já ordenada e sempre que $F^{\{i\}}$ satisfizer a Definição 3.4, $F^{\{i\}}$ é inserido no resultado $TopKMotif$, caso contrário, $F^{\{i\}}$ é descartado. Esse procedimento é repetido até que $TopKMotif$ possua K *motifs* ou até que não existam mais padrões.

Algoritmo 5.1 Algoritmo *K-TrieMotif*

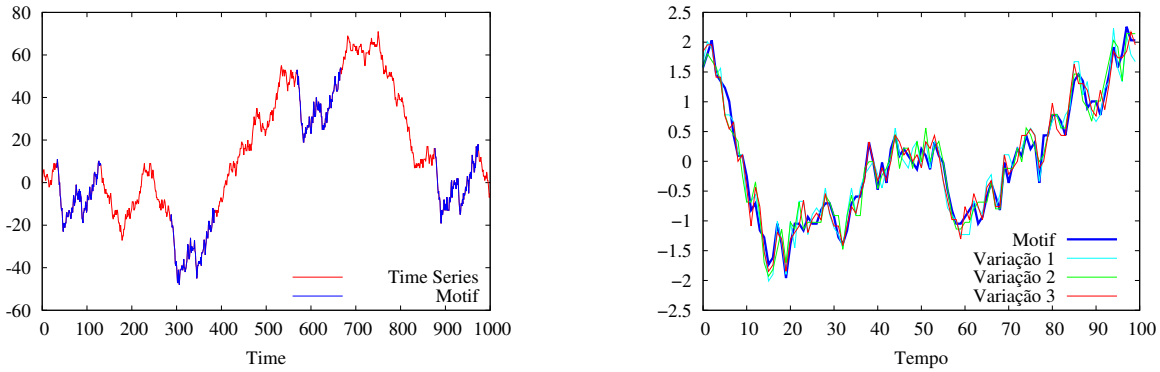
Entrada: T, n, R, w, a, δ, K
Saída: Lista dos *K-Motifs*

1: $ListaDeMotifs \leftarrow \emptyset$
2: **para toda** Subsequência S_i de tamanho n de T **faça**
3: $\hat{S}_i \leftarrow$ ConverterParaSímbolo(S_i, w, a)
4: Inserir $\hat{S}_i$ na estrutura *trie* de indexação
5: **fim para**
6: **para toda** Subsequência $\hat{S}_q$ de T **faça**
7: $Candidatos \leftarrow$ ObterCandidatosNaTrie($\hat{S}_q$, δ)
8: $MotifS \leftarrow \emptyset$
9: **para toda** $S_i \in Candidatos$ **faça**
10: **se** $CasamentoNoTrivial(S_q, S_i, R)$ **então**
11: Adcionar S_i em $MotifS$
12: **fim se**
13: **fim para**
14: Inserir $MotifS$ na $ListaDeMotifs$
15: **fim para**
16: Ordenar $ListaDeMotifs$ por quantidade de casamentos não-triviais em ordem decrescente
17: $TopKMotifs \leftarrow \emptyset$
18: $k \leftarrow 0$
19: **para toda** $(F^{\{i\}} \in ListaDeMotifs)$ **e** $(k < K)$ **faça**
20: **se** $(Distncia(F^{\{i\}}, F^{\{l\}}) \leq 2R)$, $\forall F^{\{l\}} \in TopKMotif$ **então**
21: Descartar $F^{\{i\}}$
22: **senão**
23: Inserir $F^{\{i\}}$ em $TopKMotif$
24: k++
25: **fim se**
26: **fim para**
27: **retornar** $TopKMotif$

5.4 Teste de Desempenho

Para validar o algoritmo *TrieMotif*, foram realizados experimentos com séries temporais geradas sinteticamente através de dados de passeio aleatório. Para determinar a capacidade do algoritmo *TrieMotif* em encontrar *motifs*, foram implantados *motifs* de tamanho $n = 100$ em séries temporais de tamanho $m = 1.000$ (Figura 5.8). Os *motifs* foram implantados em quatro posições da série temporal (32, 287, 568 e 875), como ilustrado na Figura 5.8a. Os *motifs* implantados são variações de um mesmo padrão (Figura 5.8b). Ao executar o algoritmo *TrieMotif* utilizando a representação simbólica de caixas com os mesmo parâmetros utilizados para a projeção aleatória em [Chiu 03] ($n = 100$, $w = 16$, $a = 4$ e $\delta = 1$), o raio de abrangência utilizado foi $R = 2,5$, o algoritmo foi capaz de encontrar os *motifs* nas posições determinadas com sucesso.

Comprovada a capacidade do algoritmo TrieMotif de localizar *motifs*, foi realizada uma



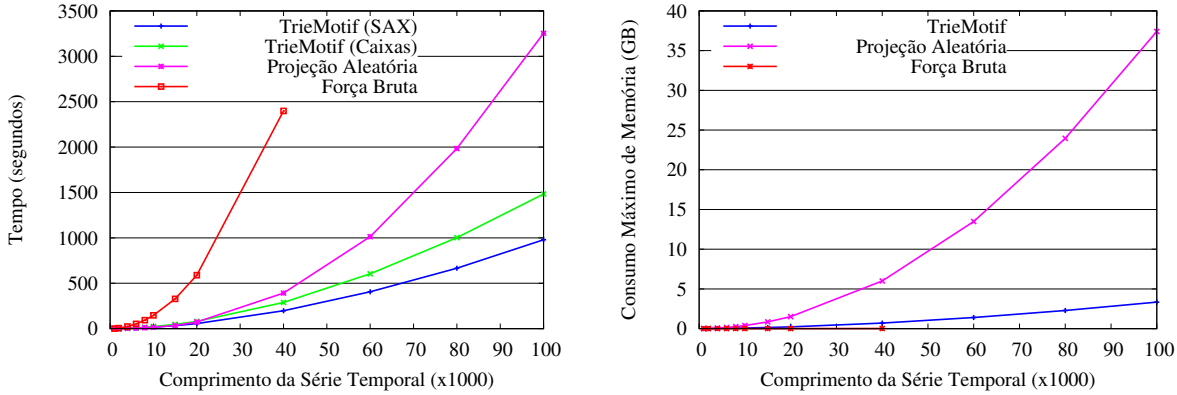
(a) Série temporal de um passeio aleatório com *motifs* (azul) implantados.

(b) *Motifs* implantados e suas variações.

Figura 5.8: Dados sintéticos de passeio aleatório com *motifs* implantados nas posições pré-determinadas 32, 287, 568 e 875.

série de experimentos para avaliar seu desempenho. O desempenho do algoritmo *TrieMotif* foi avaliado comparando com o desempenho dos algoritmos de força bruta e de projeção aleatória. Foram procurados por *motifs* de tamanho $n = 100$ utilizando a distância Euclidiana. Tanto para a projeção aleatória quanto para o *TrieMotif* foram usados os mesmos parâmetros para a representação simbólica, $a = 4$ e $w = 16$. Para a projeção aleatória foram executadas 100 iterações. Para o *TrieMotif* foram utilizadas tanto a representação simbólica de caixas como também a técnica SAX com o parâmetro da busca por abrangência $\delta = 1$. O comprimento das séries temporais foram variadas no intervalo $1.000 \leq n \leq 100.000$. Cada conjunto de parâmetros foi executado usando 5 diferentes sementes para o passeio aleatório e para cada semente foram realizadas 5 execuções, totalizando 25 execuções. Foram medidos tanto o tempo de execução, como também o consumo máximo de memória para cada uma das execuções. Os valores aqui apresentados correspondem a média das 25 execuções. Devido a limitações de tempo, os testes com séries temporais com tamanhos superiores a 40.000 não foram executados para o algoritmo de força bruta, já que para tamanhos maiores seriam necessários dias para rodar a bateria de testes. Os experimentos foram executados em um servidor HP com dois processadores Intel Xeon 5600 com 96 GB de memória primária utilizando o sistema operacional CentOS Linux 6.2. Todos os algoritmos (*TrieMotif*, projeção aleatória e força bruta) foram implementados utilizando a linguangem de programação C++ considerando apenas memória primária.

Na Figura 5.9 estão ilustrados os resultados obtidos nos dois experimentos. Como esperado, tanto o algoritmo de projeção aleatória quanto o *TrieMotif* obtiveram uma melhor perfomance se comparados ao algoritmo de força bruta (Figura 5.9a). Os algoritmos de projeção aleatória e *TrieMotif* se comportaram de forma similar para séries temporais de tamanho inferior a 10.000. Entretanto, a medida que a série temporal aumenta de tamanho, o *TrieMotif* apresenta uma melhor perfomance devido à redução do espaço de busca para cada subsequência de consulta. Também é possível observar que o *TrieMotif* apresenta uma melhor performance ao



(a) Teste de desempenho.

(b) Teste de consumo de memória.

Figura 5.9: Comparações realizadas com os algoritmos *TrieMotif*, força bruta e projeção aleatória.

ser utilizada a técnica SAX. Essa melhora na perfomance ocorre pois ao utilizar a técnica SAX, o número de candidatos retornados em uma busca em abragência é menor. Como a técnica SAX utiliza intervalos equiprováveis de uma distribuição normal para a conversão simbólica, alguns símbolos apresentam um menor intervalo de valores, diminuindo assim o espaço de busca de candidatos, por exemplo, os intervalos para valores próximos de 0 são menores do que os intervalos para valores com módulo maior (Figura 2.7). Na Figura 5.9b é possível observar que o *TrieMotif* também apresentou um menor consumo de memória em relação a projeção aleatória, visto que, para séries temporais de tamanho m, essa necessita de um espaço em memória de pelo menos m^2 para a matriz de colisão. O *TrieMotif* requereu até 10 vezes menos memória do que a projeção aleatória.

n	50	100	150	200	250	300	350	400	450	500
w	8	16	25	33	41	50	58	66	75	83

Tabela 5.1: Tamanho n dos *motifs* procurados na *trie* e os respectivos tamanhos w utilizados para as representações simbólicas. Foi utilizada uma razão de 6 entre n e w para definir o tamanho da representação simbólica.

Também foram realizados experimentos para verificar a capacidade de redução do número de caminhos percorridos na busca por abrangência ao se alterar a ordem de indexação das represetanções simbólicas. Para esse experimento, foi computado o número de nós visitados para cada consulta realizada na *trie* variando o tamanho dos *motifs*. O tamanho n dos *motifs* e o tamanho w das representações simbólicas variaram de acordo com os valores presentes na Tabela 5.1. Além disso, foi utilizada a representação simbólica de caixas com $a = 4$. O algoritmo *TrieMotif* foi executado para as séries temporais de tamanho $m = 10.000$, utilizando o passeio aleatório com cinco sementes diferentes. Para cada tamanho de consulta, foi registrada a média do número de nós visitados. Os resultados obtidos podem ser observados na Figura 5.10.

Ao indexar as representações simbólicas em uma ordem não sequencial houve uma redução no número médio de nós visitados por consulta. Para as consultas de tamanho $w = 50$ houve uma redução de até 14.78%.

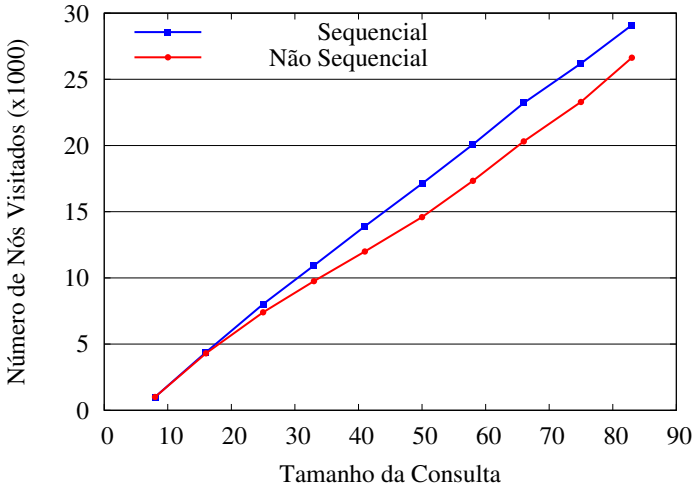


Figura 5.10: Número médio de nós visitados por consulta utilizando indexação sequencial e não sequencial.

5.5 Aplicação em Imagens de Satélite

O algoritmo *TrieMotif* também foi analisado em dados de aplicações reais. Foram utilizadas séries temporais obtidas a partir de imagens dos satélites AVHRR/NOAA[1]. As imagens correspondem a leitura mensal do NDVI[2], que indica o vigor vegetativo do solo, sendo fortemente relacionado com a biomassa. Foram utilizadas imagens da região do estado de São Paulo correspondentes ao perído de abril de 2001 a março de 2010. A partir das imagens foram extraídas 174.034 séries temporais de NDVI de tamanho 108. Cada série temporal corresponde a um par de coordenadas de latitude e longitude (pixel da imagem) do estado de São Paulo, com exceção da região litorânea.

Para que o algoritmo *TrieMotif* possa trabalhar com uma base de dados de séries temporais, foram necessários realizar alguns ajustes. Como o *TrieMotif* utiliza como entrada uma única série temporal, todas as séries temporais da base de dados foram concatenadas para que fosse obtida uma única série. É importante ressaltar que foram tomados os devidos cuidados para que no processo de extração de subsequências não fossem criadas subsequências espúrias, isto é, formadas por duas séries temporais (partes do final de uma série e início de outra). Como o estado de São Paulo é um dos maiores produtores de cana-de-açúcar do país, foram procurados *motifs* de tamanho $n = 12$, que correspondem ao ciclo anual da safra de cana-de-açúcar. O

[1] *Advanced Very High Resolution Radiometer/National Oceanic and Atmospheric Administration*
[2] Normalized Difference Vegetation Index

TrieMotif foi executado utilizando a representação simbólica de caixas com os parâmetros $R = 1,5$, $w = 4$, $a = 4$ e $\delta = 1$. Os experimentos foram executados na mesma infraestrutura computacional utilizada para os experimentos com dados sintéticos.

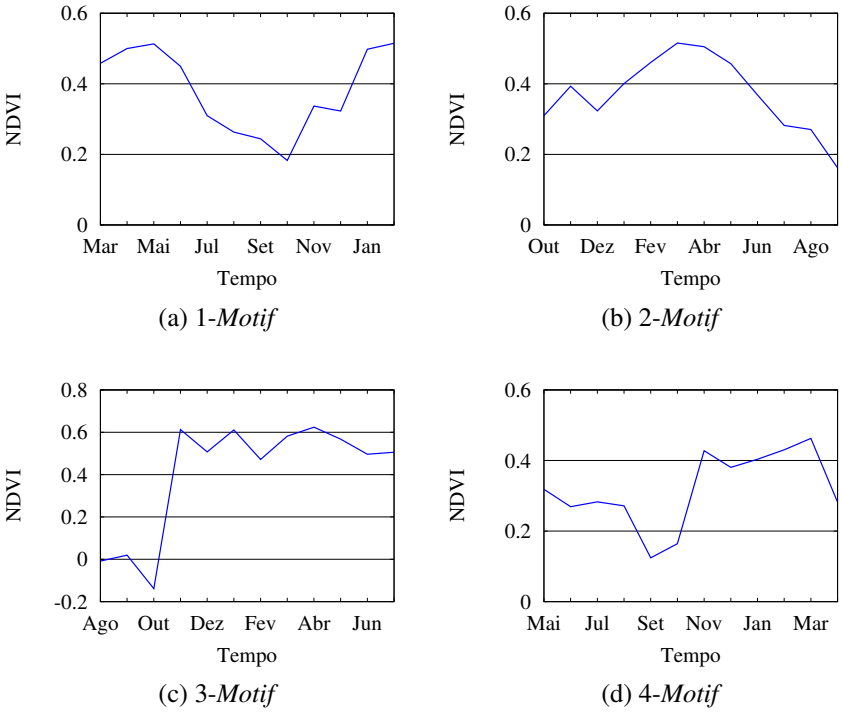


Figura 5.11: Os 4-*Motif* mais frequentes utilizando toda a área do estado de São Paulo.

Os centros de consulta dos 4-*Motifs* mais frequentes obtidos estão ilustrados na Figura 5.11. Para que os especialistas pudessem realizar uma análise semântica dos resultados obtidos, as séries temporais se encontram no domínio original dos dados, ou seja, não estão z-normalizadas. Como esperado, os dois padrões mais frequentes (Figura 5.11a e 5.11b) se assemelham ao comportamento do ciclo da safra de cana-de-açucar, com valores de NDVI elevados no mês de abril e baixos no mês de outubro. Esse comportamento ocorre pois no período de abril as safras de cana-de-açúcar se encontram em sua capacidade máxima, apresentando então altos valores de NDVI. Entre os meses de abril e outubro, é realizada a colheita da cana-de-açúcar, o que acarreta na redução gradual dos valores de NDVI, até alcançar os menores índices no mês de outubro, período em que o solo está exposto. Como a safra é replantada no período entre outubro e abril, os valores de NDVI se elevam até que o ciclo seja reiniciado em abril. Porém, esses padrões foram encontrados em praticamente todas as posições geográficas do estado de São Paulo, tornando o resultado obtido inconsistente. Para contornar esse problema, o estado de São Paulo foi dividido em 5 regiões de interesse (Figura 5.12). As regiões de interesse foram obtidas através de técnicas de agrupamento, sendo em seguida analisadas por especialistas na área de agrometeorologia e sensoriamento remoto. De acordo com os especialistas, a região 1 apresenta regiões urbanas e água, as regiões 2 e 3 apresentam pastagem e uma mistura de culturas (com

exceção da cana-de-açúcar), a região 4 apresenta áreas de cultivo da cana-de-açúcar e por último, a região 5 apresenta regiões florestais. Na Tabela 5.2 está apresentada uma sumarização de cada região de interesse como também quantidade de séries temporais para cada região.

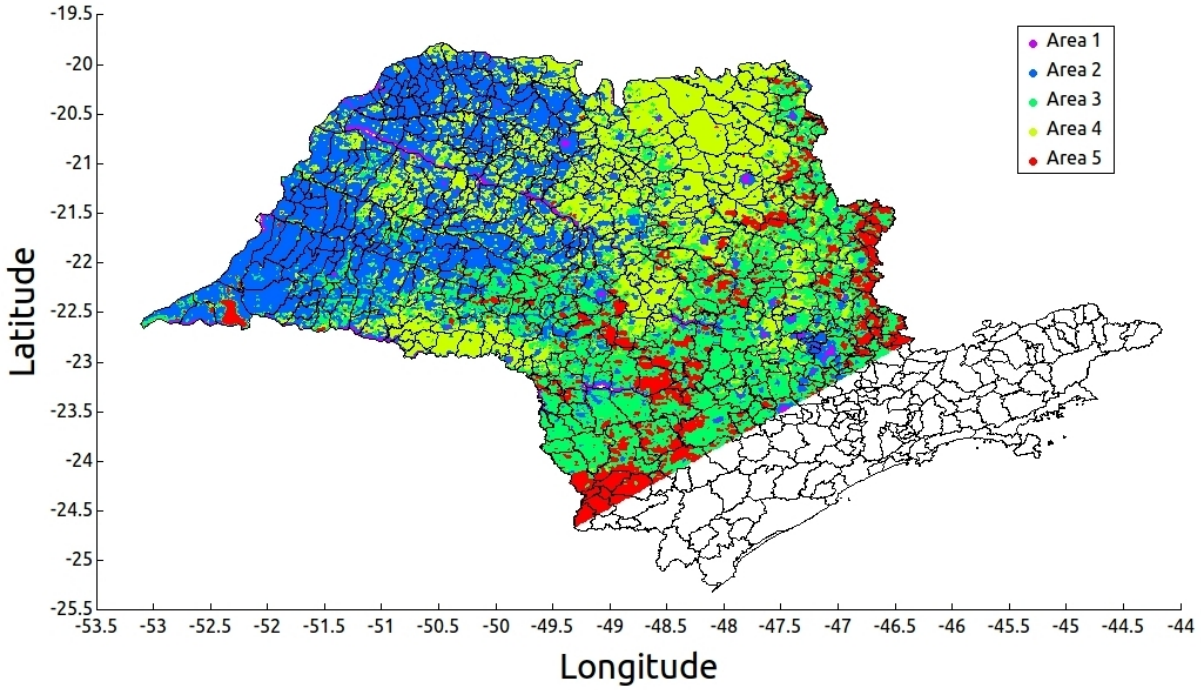


Figura 5.12: O estado de São Paulo foi dividido em 5 regiões de interesse. A região 1 (roxo) corresponde a água e áreas urbanas, as regiões 2 e 3 (azul e verde respectivamente) a pastagem e misto de culturas, a região 4 (amarelo) apresenta áreas de cultivo de cana-de-açúcar e a região 5 a florestas.

Região	Cor	# de Séries Temporais	Apresenta
1	Roxo	2.804	Água e Região Urbana
2	Azul	55.815	Pastagem e Mistura de Culturas
3	Verde	50.785	
4	Amarelo	48.301	Cultura de Cana-de-açúcar
5	Vermelho	16.329	Floresta

Tabela 5.2: Regiões de interesse segundo os especialistas em agrometeorologia e sensoriamento remoto.

Nas Figuras 5.13, 5.14, 5.15, 5.16 e 5.17 estão ilustrados os centros de consulta dos 4-*Motifs* mais frequentes para cada região de interesse. Novamente, as séries temporais se encontram no

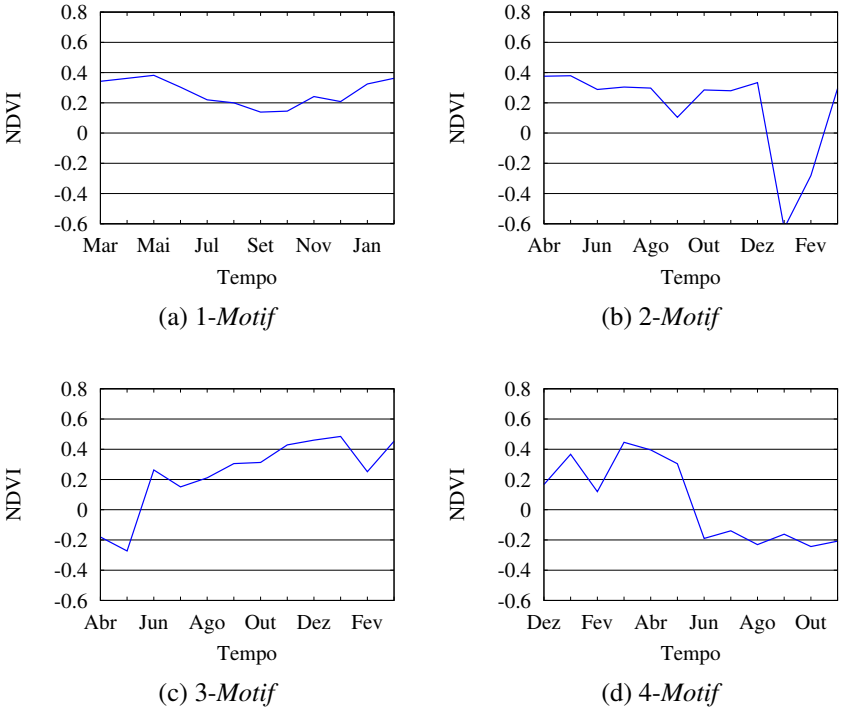


Figura 5.13: Os 4-*Motif* mais frequentes para a região de interesse 1 (Água e Região Urbana).

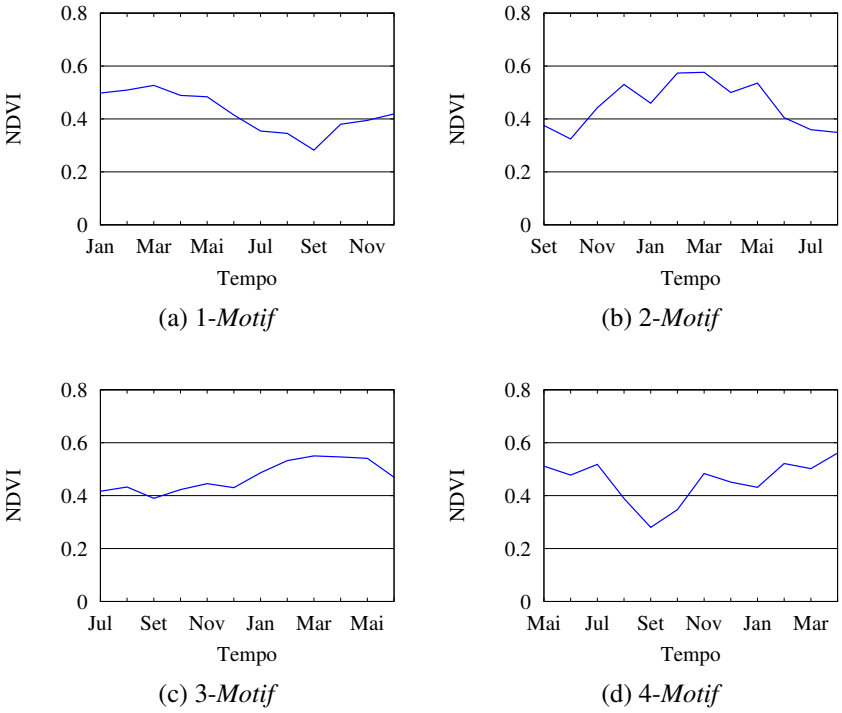
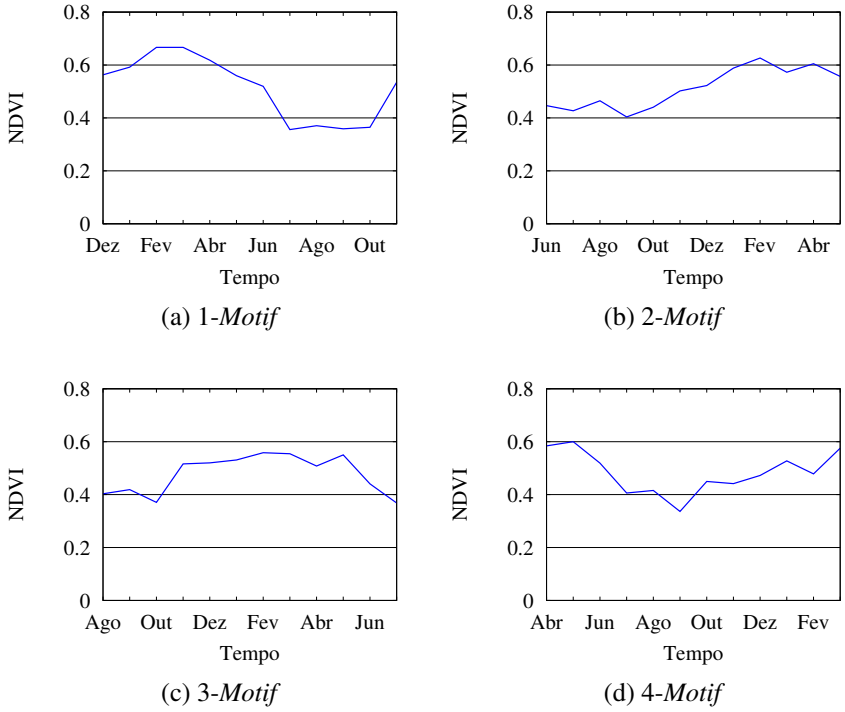


Figura 5.14: Os 4-*Motif* mais frequentes para a região de interesse 2 (Pastagem e Mistura de Culturas).

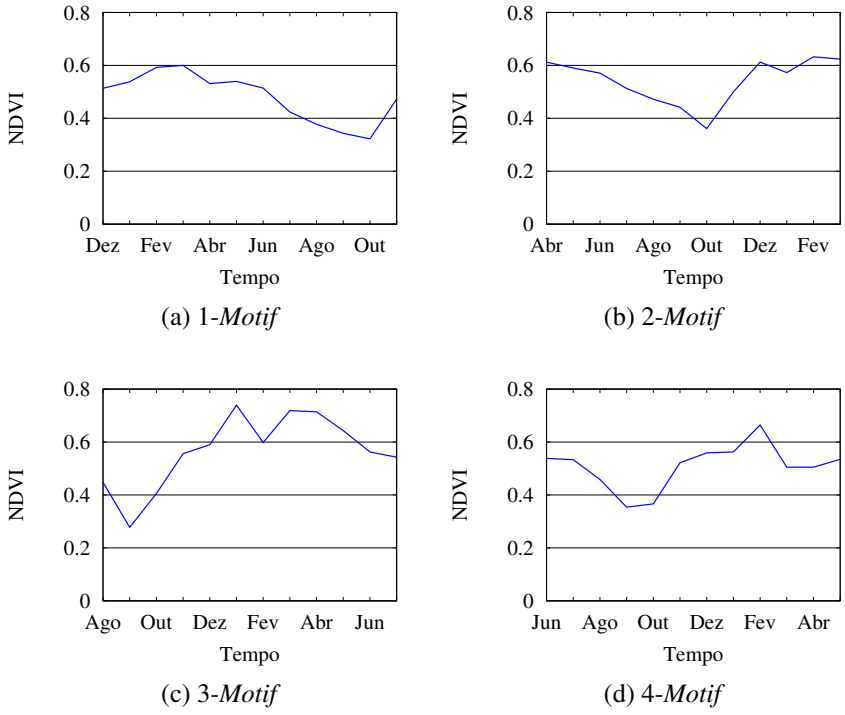
domínio original dos dados. O 1-*Motif* encontrado na região 1 (Figura 5.13a) apresenta um padrão com valores baixos e uma variação pequena, esse comportamento corresponde, segundo



(a) 1-*Motif* (b) 2-*Motif*

(c) 3-*Motif* (d) 4-*Motif*

Figura 5.15: Os 4-*Motif* mais frequentes para a região de interesse 3 (Pastagem e Mistura de Culturas).



(a) 1-*Motif* (b) 2-*Motif*

(c) 3-*Motif* (d) 4-*Motif*

Figura 5.16: Os 4-*Motif* mais frequentes para a região de interesse 4 (Cultura de Cana-de-açúcar).

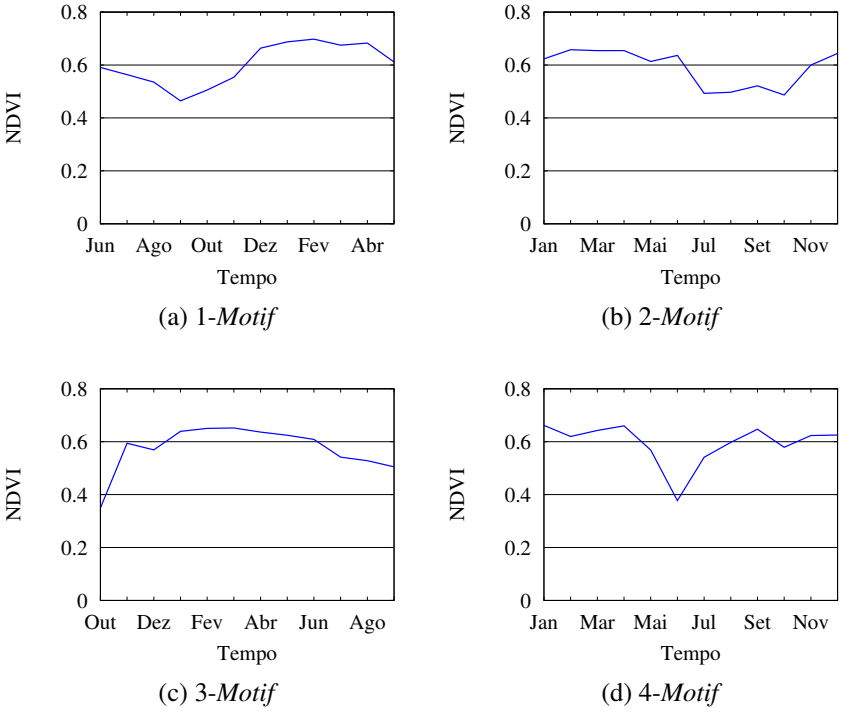


Figura 5.17: Os 4-*Motif* mais frequentes para a região de interesse 5 (Regiões de Floresta).

os especialistas, a áreas urbanas, já que os valores baixos de NDVI indicam a presença de solo com pouca vegetação. Já os outros *motifs* encontrados na região 1 (Figuras 5.13b, 5.13c e 5.13d) apresentam um padrão correspondente a água, um comportamento não definido. Como era esperado, os padrões encontrados nas regiões 2 e 3 (Figuras 5.14 e 5.15) apresentam altos valores de NDVI com uma alta variação, indicando a presença de áreas agrícolas, vegetação rasteira e áreas de solo exposto. Os 3-*Motifs* mais frequentes da região 4 (Figuras 5.16a, 5.16b e 5.16c) foram facilmente reconhecidos pelos especialistas como o ciclo da safra de cana-de-açúcar. Os padrões encontrados na Figura 5.17 apresentam altos valores de NDVI e pouca variação ao longo do ano, o que representa um comportamento esperado para áreas florestais, onde o NDVI acompanha as variações de temperatura da região.

O *TrieMotif* também obteve resultados interessantes ao encontrar padrões que não eram esperados pelos especialistas. De acordo com os especialistas, para algumas regiões o 4-*Motif* encontrado não corresponde ao padrão esperado. O 4-*Motif* encontrado nas regiões 2 e 3 (Figuras 5.14d e 5.15d respectivamente) indicam a presença de cana-de-açúcar, já o 4-*Motif* encontrado na região 4 (Figura 5.16d) não se assemelha ao ciclo da safra de cana-de-açúcar. Esses resultados indicam que há a necessidade de inspencionar as áreas en que esses padrões ocorreram, para assim determinar os fatores que acarretaram nessa discrepância de comportamento.

5.6 Considerações Finais

A descoberta de padrões é uma atividade de extrema relevância para a mineração de dados em séries temporais, pois é possível determinar o comportamento que antecede a ocorrência de eventos extremos. Por exemplo, a partir de dados de clima e de agrometeorologia é possível realizar previsões de produções agrícolas a partir de variações na temperatura e quantidade de precipitação. Atualmente existe uma ampla rede de sensores para a captura de dados, como estações de monitoramento de clima, como também sensores remotos/satélites para a detecção de atividades na superfície terrestre. Entretanto, a grande quantidade e a diversidade dos dados coletados torna a descoberta de padrões capazes de auxiliar nessas análises, que são bastante complexas. Nesse contexto, é muito importante a descoberta de mecanismos rápidos e poderosos para auxiliar na análise desse tipo de dado.

Nesse capítulo foi apresentado o algoritmo *TrieMotif* destinado a descoberta de padrões frequentes em uma série temporal. O *TrieMotif* se aproveita de técnicas de reduçao da dimensionalidade dos dados para aumentar a velocidade das comparações entre as subsequências. As subsequências são indexadas em uma estrutura *trie* e é adotada uma heurística para uma busca por similaridade para descartar de antemão as subsequências que não são de interesse para o resultado final. Desse modo, o *TrieMotif* é capaz de selecionar apenas as subsequências candidatas que apresentem uma alta probabilidade de formarem um casamento.

Os resultados experimentais obtidos a partir dos dados sintéticos e reais mostraram que o *TrieMotif* foi capaz de aumentar a velocidade da descoberta de *motifs* em até 3 vezes comparado ao estado da arte (projeção aleatória). Além do melhor desempenho, o *TrieMotif* também requereu uma menor quantidade de memória, chegando a até 10 vezes menos memória do que a projeção aleatória. Os resultados obtidos a partir dos dados agrometeorológicos analisados pelos especialistas se mostraram corretos, podendo auxiliar em suas atividades para o processamento de dados climáticos.

Capítulo 6

Conclusões

6.1 Considerações Finais

Esta dissertação de mestrado visou contribuir com algoritmos e técnicas de mineração de séries temporais considerando as restrições impostas em grandes bases de dados. Ou seja buscou desenvolver algoritmos de complexidade linear (escaláveis), que permitissem discretizar, indexar e encontrar padrões frequentes em séries temporais para suportar consultas usualmente demandadas pelos especialistas da área de agrometeorologia e climatologia.

Os algoritmos foram testados tanto com dados sintéticos quanto com dados reais dos domínios de agrometeorologia e climatologia, apresentando resultados bastante promissores, como puderam ser observados nos Capítulos 4 e 5.

6.2 Principais Contribuições

Este projeto de mestrado apresentou duas contribuições principais. A primeira contribuição consistiu no índice *Telesto* para realizar consultas por similaridade em séries temporais. O *Telesto* utiliza como base as técnicas de representação simbólica de séries temporais para se aproveitar da alta velocidade das consultas por subcadeias de uma árvore de sufixo. Foram propostas modificações nos algoritmos de busca da árvore de sufixo generalizada para que fosse possível utilizá-la em consultas por similaridade de séries temporais convertidas em representação simbólica. Experimentos mostraram que o índice *Telesto* obteve ótimo desempenho, realizando as consultas por similaridade em complexidade linear com o tamanho da base de séries temporais e realizando consultas em abrangência indiferente ao tamanho da cadeia de consulta. Os resultados obtidos com o índice *Telesto* foram publicados no *Journal of Information and*

Data Management, volume 3, número 2 de junho de 2012 [Chino 12].

A segunda contribuição deste trabalho foi o algoritmo *TrieMotif* para a descoberta de padrões frequentes em séries temporais. O *TrieMotif* converte as subsequências de uma série temporal e as indexa em uma estrutura baseada na *trie*. Através da *trie*, o algoritmo realiza a seleção de candidatos a *motif*, reduzindo assim a quantidade de cálculos realizados no domínio original da série temporal. Experimentos mostraram que o algoritmo *TrieMotif* é capaz de encontrar os *k-motifs* mais frequentes, apresentando um desempenho superior a um dos principais algoritmos para essa tarefa. Ao ser aplicado em dados reais obtidos através de imagens de satélite, o *TrieMotif* foi capaz de encontrar os padrões mais frequentes esperados pelos especialistas em agrometeorologia. O *TrieMotif* também mostrou a existência de padrões inesperados em certas regiões, indicando a necessidade de uma análise mais profunda pelos especialistas. Os resultados obtidos com o *TrieMotif* foram submetidos à 16th *International Conference on Enterprise Information Systems* (ICEIS 2014) sendo aceito para publicação [Chino 14].

No contexto mais geral desta pesquisa, visando suportá-la com ferramentas e algoritmos que envolveram de modo mais próximo o usuário em suas análises utilizando dispositivos de interação inovadores, foram também desenvolvidos mecanismos de interação sobre a plataforma Kinect da Microsoft, que geraram os artigos:

- Romani, Luciana A.S.; Chino, Daniel Y. T.; Avalhais, Letrícia P. S.; Oliveira, Willian D.; Gonçalves, Renata R. V.; Traina Jr., Caetano; Traina, Agma J. M. - "*Involving users in the gestural language definition process for the NInA framework*", *Brazilian Symposium on Human Factors in Computer Systems* (IHC 2013) *short paper*, pp. 280-283, 8-11 de outubro de 2013, Manaus, Brazil.

- Chino, Daniel Y. T.; Romani, Luciana A.S.; Avalhais, Letrícia P. S.; Oliveira, Willian D.; Gonçalves, Renata R. V.; Traina Jr., Caetano; Traina, Agma J. M. - "*The NInA Framework: Using Gesture to Improve Interaction and Collaboration in Geographical Information Systems*", *15th International Conference on Enterprise Information Systems* (ICEIS 2013), pp. 58-66, 4-7 de julho 2013, Angers Loire Valley, France.

6.3 Trabalhos Futuros

Há diversas vertentes de trabalhos futuros que poderão ser realizados como continuação da pesquisa desenvolvida nesta dissertação. Entre elas pode-se citar:

- Aprimoramento de função de distância para comparação das séries temporais.

- Desenvolvimento de algoritmos de agrupamento, utilizando como base tanto a série temporal completa quanto os *motifs* delas obtidos.

- Criação de "dicionários de palavras" que descrevam as séries, utilizando os *motifs* extraídos das séries.
- Análise do algoritmos em séries temporais provenientes de outras regiões do Brasil como também da América do Sul. Além de explorar dados obtidos através de diferentes sensores.
- Análise dos algoritmos em outros contextos de utilização de séries temporais.
- Agilizar o algoritmo *TrieMotif* utilizando propriedades da *trie* para selecionar primeiro os centros de consulta com maior probabilidade de ser um *motif*. Evitando assim cálculos desnecessários da função de distância no domínio original da série temporal.
- Investigar métodos para determinar automaticamente o raio de abrangência para a descoberta de *motifs* frequentes, evitando assim sua passagem como parâmetro.

Referências Bibliográficas

[Agrawal 93] Rakesh Agrawal, Christos Faloutsos & Arun Swami. *Efficient similarity search in sequence databases*. In David Lomet, editeur, Foundations of Data Organization and Algorithms, volume 730 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 1993.

[Alon 09] Jonathan Alon, Vassilis Athitsos, Quan Yuan & Stan Sclaroff. *A Unified Framework for Gesture Recognition and Spatiotemporal Gesture Segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 9, pages 1685–1699, 2009.

[Baum 70] Leonard E. Baum, Ted Petrie, George Soules & Norman Weiss. *A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains*. The Annals of Mathematical Statistics, vol. 41, no. 1, pages 164–171, 1970.

[Buhler 01] Jeremy Buhler & Martin Tompa. *Finding Motifs Using Random Projections*. In Proceedings of the Fifth Annual International Conference on Computational Biology, RECOMB '01, pages 69–76, New York, NY, USA, 2001. ACM.

[Castro 12] Nuno Constantino Castro. *Time series motif discovery*. PhD Thesis, Universidade do Minho, Portugal, 2012.

[Catalano 06] Joe Catalano, Tom Armstrong & Tim Oates. *Discovering Patterns in Real-Valued Time Series*. In Johannes Fürnkranz, Tobias Scheffer & Myra Spiliopoulou, editeurs, Knowledge Discovery in Databases: PKDD 2006, volume 4213 of *Lecture Notes in Computer Science*, pages 462–469. Springer Berlin Heidelberg, 2006.

[Chan 99] Kin-pong Chan & Ada Wai-Chee Fu. *Efficient Time Series Matching by Wavelets*. In International Conference on Data Engineering, pages 126–133, Sydney, Australia, 1999.

[Chan 03] Franky Kin-Pong Chan, Ada Wai-chee Fu & Clement Yu. *Haar Wavelets for Efficient Similarity Search of Time-Series: With and Without Time Warping*. IEEE Transactions on Knowledge and Data Engineering, vol. 15, pages 686–705, 2003.

[Chen 04] Lei Chen & M Tamer Özsu. *Multi-Scale Histograms for Answering Queries over Time Series Data*. In International Conference on Data Engineering, page 838, Boston, MA, USA, 2004. IEEE Computer Society.

[Chino 11] Daniel Yoshinobu Takada Chino, Felipe Alves Louza, Agma Juci Machado Traina, Cristina Dutra Aguiar Ciferri & Caetano Traina Jr. *SatImagExplorer: tornando a mineração de dados de sensores orbitais mais flexível*. In Anais da Sessão de Demos do XXVI Simpósio Brasileiro de Banco de Dados (SBBD)., pages 25–30, Florianópolis, Brazil, 2011.

[Chino 12] Daniel Yoshinobu Takada Chino, Felipe Alves Louza, Agma Juci Machado Traina, Cristina Dutra de Aguiar Ciferri & Caetano Traina Jr. *Time Series Indexing Taking Advantage of the Generalized Suffix Tree*. Journal of Information and Data Management, vol. 3, no. 2, pages 101–109, 2012.

[Chino 14] Daniel Yoshinobu Takada Chino, Renata Ribeiro do Valle Gonçalves, Luciana Alvim Santos Romani, Caetano Traina Jr. & Agma Juci Machado Traina. *TrieMotif: a New and Efficient Method to Mine Frequent K-Motifs from Large Time Series (Aceito para publicação)*. In Proceedings of the 16th International Conference on Enterprise Information Systems, pages 1–10, Lisboa, Portugal, 2014. SciTePress.

[Chiu 03] Bill Chiu, Eamonn Keogh & Stefano Lonardi. *Probabilistic discovery of time series motifs*. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '03, pages 493–498, New York, NY, USA, 2003. ACM.

[Ding 08] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang & E. Keogh. *Querying and mining of time series data: experimental comparison of representations and distance measures*. Proceedings of the VLDB Endowment, vol. 1, no. 2, pages 1542–1552, 2008.

[Faloutsos 94] Christos Faloutsos, M. Ranganathan, Yannis Manolopoulos & Yannis Manolopoulos. *Fast Subsequence Matching in Time-Series Databases.* In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 419–429, Minneapolis, USA, 1994.

[Felipe 05] J.C. Felipe, A.J.M. Traina & Jr. Traina C. *Global warp metric distance: boosting content-based image retrieval through histograms*. In Multimedia, Seventh IEEE International Symposium on, volume 1, pages 295 – 302., Los Alamitos, USA, 2005.

[Geurts 01] Pierre Geurts. *Pattern Extraction for Time Series Classification*. In Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery, pages 115–127, Freiburg, Germany, 2001.

[Gruber 06] Christian Gruber, Michael Coduro, Bernhard Sick*et al.* *Signature verification with dynamic RBF networks and time series motifs*. In Tenth International Workshop on Frontiers in Handwriting Recognition, pages 455 – 460, La Baule, France, 2006.

[Gusfield 97] Dan Gusfield. Algorithms on strings, trees, and sequences - computer science publisher = Cambridge University Press, and computational biology. Cambridge, United Kingdom, 1997.

[Han 11] J. Han, M. Kamber & J. Pei. Data mining: Concepts and techniques. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011.

[IPCC 07] IPCC. *Climate change 2007: Fourth assessment report (AR4)*. In Intergovernmental Panel on Climate Change, volume 34, pages 100–125, 2007.

[Jiang 09] Yu-Feng Jiang, Chun-Ping Li & Jun-Zhou Han. *Stock temporal prediction based on time series motifs*. In 2009 International Conference on Machine Learning and Cybernetics, volume 6, pages 3550–3555, Baoding, China, July 2009.

[Kalpakis 01] Konstantinos Kalpakis, Dhiral Gada & Vasundharu Puttagunta. *Distance Measures for Effective Clustering of ARIMA Time-Series*. Data Mining, IEEE International Conference on, vol. 0, page 273, 2001.

[Keogh 98] Eamonn J Keogh & Michael J Pazzani. *An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering*

and Relevance Feedback. In Forth International Conference on Knowlodge Discovery and Data Mining, pages 239–243, New York, USA, 1998.

[Keogh 00] Eamonn Keogh & Michael Pazzani. *A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases*. In Takao Terano, Huan Liu & Arbee Chen, editeurs, Knowledge Discovery and Data Mining. Current Issues and New Applications, volume 1805 of *Lecture Notes in Computer Science*, pages 122–133. Springer Berlin / Heidelberg, 2000.

[Keogh 01a] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani & Sharad Mehrotra. *Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases*. Knowledge and Information Systems, vol. 3, pages 263–286, 2001.

[Keogh 01b] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani & Sharad Mehrotra. *Locally adaptive dimensionality reduction for indexing large time series databases*. In Proceedings of the 2001 ACM SIGMOD international conference on Management of data, SIGMOD '01, pages 151–162, New York, NY, USA, 2001. ACM.

[Keogh 01c] Eamonn Keogh, Selina Chu, David Hart & Michael Pazzani. *An Online Algorithm for Segmenting Time Series*. IEEE International Conference on Data Mining, pages 289 – 296, 2001.

[Keogh 03] E. Keogh & Shruti Kasetty. *On the need for time series data mining benchmarks: a survey and empirical demonstration*. vol. 7, no. 4, pages 349–371, 2003.

[Keogh 05] Eamonn Keogh & Chotirat Ann Ratanamahatana. *Exact indexing of dynamic time warping*. Knowledge and information systems, vol. 7, no. 3, pages 358–386, 2005.

[Korn 97] Flip Korn, H V Jagadish & Christos Faloutsos. *Efficiently supporting ad hoc queries in large datasets of time sequences*. In Proceedings of the 1997 ACM SIGMOD international conference on Management of data, SIGMOD '97, pages 289–300, New York, NY, USA, 1997. ACM.

[Last 04] M. Last, A. Kandel & H. Bunke. Data mining in time series databases. Series in machine perception and artificial intelligence. World Scientific, Washington, D.C., USA, 2004.

[Li 10] Yuan Li & Jessica Lin. *Approximate variable-length time series motif discovery using grammar inference*. In Proceedings of the Tenth International Workshop on Multimedia Data Mining, MDMKDD '10, pages 10:1—-10:9, New York, NY, USA, 2010. ACM.

[Li 12] Yuan Li, Jessica Lin & Tim Oates. *Visualizing Variable-Length Time Series Motifs.* In Twelfth SIAM International Conference on Data Mining, pages 895–906, Anaheim, USA, 2012. SIAM / Omnipress.

[Lin 02] J Lin, E Keogh, P Patel & S Lonardi. *Finding motifs in time series*. In In the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, International Conference on Knowledge Discovery and Data Mining, pages 53–68, Edmonton, Alberta, Canada, 2002. ACM.

[Lin 03] Jessica Lin, Eamonn Keogh, Stefano Lonardi & Bill Chiu. *A symbolic representation of time series, with implications for streaming algorithms*. In Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, DMKD '03, pages 2–11, New York, NY, USA, 2003. ACM.

[Lin 07] Jessica Lin, Eamonn Keogh, Li Wei & Stefano Lonardi. *Experiencing SAX: a novel symbolic representation of time series*. Data Mining and Knowledge Discovery, vol. 15, no. 2, pages 107–144, 2007.

[Luo 04] Kedong Luo, Jianmin Wang & Jia-Guang Sun. *Rules Discovery from Cross-Sectional Short-Length Time Series*. In Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, pages 604–614, Sydney, Australia, 2004.

[Meng 08] Jingjing Meng, Junsong Yuan, Mat Hans & Ying Wu. *Mining motifs from human motion*. In EUROGRAPHICS, volume 8, pages 71–74, Crete, Greece, 2008.

[Minnen 08] David Minnen. *Unsupervised discovery of activity primitives from multivariate sensor data*. PhD Thesis, Georgia Institute of Technology, Atlanta, USA, 2008.

[Mitsa 09] T. Mitsa. Temporal data mining. Chapman & Hall/CRC data mining and knowledge discovery series. Chapman & Hall/CRC, 2009.

[Mohammad 09] Yasser Mohammad & Toyoaki Nishida. *Constrained Motif Discovery in Time Series*. New Generation Computing, vol. 27, no. 4, pages 319–346, 2009.

[Mueen 09a] Abdullah Mueen, Eamonn Keogh & Nima Bigdely-Shamlo. *Finding Time Series Motifs in Disk-Resident Data*. Data Mining, IEEE International Conference on, vol. 0, pages 367–376, 2009.

[Mueen 09b] Abdullah Mueen, Eamonn J Keogh, Qiang Zhu, Sydney Cash & Brandon Westover. *Exact Discovery of Time Series Motifs*. In SIAM International Conference on Data Mining (SDM09), Sparks, USA, 2009. American Statistical Association (ASA).

[Perng 00] Chang-Shing Perng, Haixun Wang, Sylvia R Zhang & Douglas Stott Parker Jr. *Landmarks: a new model for similarity-based pattern querying in time series databases*. In International Conference on Data Engineering, pages 33–42, San Diego, USA, 2000.

[Pevzner 00] Pavel A. Pevzner & Sing-Hoi Sze. *Combinatorial Approaches to Finding Subtle Signals in DNA Sequences*. In Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, pages 269–278, San Diego, USA, 2000. AAAI Press.

[Rabiner 89] L. Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition*. Proceedings of the IEEE, vol. 77, no. 2, pages 257–286, Feb 1989.

[Ratanamahatana 04] C. Ratanamahatana & E. Keogh. *Everything you know about Dynamic Time Warping is Wrong*. In 3rd Workshop on Mining Temporal and Sequential Data, in conjunction with 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 22 – 25, Seatle, WA, USA, 2004. ACM.

[Ratanamahatana 05] C. Ratanamahatana, E. Keogh, A.J. Bagnall & Stefano Lonardi. *A novel bit level time series representation with implication of similarity search and clustering*. pages 771–777, 2005.

[Romani 10] Luciana A. S. Romani, R. R. V. Goncalves, Jurandir Zullo-Jr., Caetano Traina-Jr. & Agma J. M. Traina. *New DTW-based method to similarity search in sugar cane regions represented by climate and remote sensing time series*. In Proceedings of the IEEE International Geoscience & Remote Sensing Symposium, pages 355–358, Honolulu, USA, 2010.

[Rouse 73] J. W. Rouse, R. H. Haas, J. A. Schell & D. W. Deering. *Monitoring vegetation systems in the Great Plains with ERTS*. In Proceedings of the Third ERTS Symposium, pages 309–317, Washington, DC, USA, 1973.

[Sakurai 05] Yasushi Sakurai, Masatoshi Yoshikawa & Christos Faloutsos. *FTW: fast similarity search under the time warping distance*. In Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '05, pages 326–337, New York, NY, USA, 2005. ACM.

[Salvador 07] Stan Salvador & Philip Chan. *Toward Accurate Dynamic Time Warping in Linear Time and Space*. Intell. Data Anal., vol. 11, no. 5, pages 561–580, October 2007.

[Tang 08] Heng Tang & Stephen Shaoyi Liao. *Discovering original motifs with different lengths from time series*. Knowledge-Based Systems, vol. 21, no. 7, pages 666–671, 2008.

[Truong 12] CaoDuy Truong & DuongTuan Anh. *Time Series Prediction Using Motif Information*. In Chattrakul Sombattheera, NguyenKim Loi, Rajeev Wankar & Tho Quan, editeurs, Multi-disciplinary Trends in Artificial Intelligence, volume 7694 of *Lecture Notes in Computer Science*, pages 110–121. Springer Berlin Heidelberg, 2012.

[Udechukwu 04] Ajumobi Udechukwu, Ken Barker & Reda Alhajj. *Discovering all frequent trends in time series*. In Proceedings of the winter international synposium on Information and communication technologies, WISICT '04, pages 1–6, Cancun, Mexico, 2004. Trinity College Dublin.

[Uppaluri 02] R. Uppaluri, T. Mitsa, E. A. Hoffman, G. McLennan & M. Sonka. *Method and Apparatus for Analyzing CT Images to Determine the Presence of Pulmonary Tissue Pathology*, 10 2002.

[Vlachos 02] Michail Vlachos, Dimitrios Gunopoulos & George Kollios. *Discovering Similar Multidimensional Trajectories*. Data Engineering, International Conference on, vol. 0, pages 673–684, 2002.

[Vlachos 03] Michail Vlachos, Marios Hadjieleftheriou, Dimitrios Gunopulos & Eamonn Keogh. *Indexing multi-dimensional time-series with support for multiple distance measures*. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '03, pages 216–225, New York, NY, USA, 2003. ACM.

[Wang 13] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann & Eamonn Keogh. *Experimental comparison of representation methods and distance measures for time series data*. Data Mining and Knowledge Discovery, vol. 26, no. 2, pages 275–309, 2013.

[Weiner 73] Peter Weiner. *Linear pattern matching algorithms*. 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, vol. 0, pages 1–11, 1973.

[Yankov 07] Dragomir Yankov, Eamonn Keogh, Jose Medina, Bill Chiu & Victor Zordan. *Detecting time series motifs under uniform scaling*. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '07, pages 844–853, New York, NY, USA, 2007. ACM.

[Yi 00] Byoung-Kee Yi & Christos Faloutsos. *Fast Time Sequence Indexing for Arbitrary Lp Norms*. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter & Kyu-Young Whang, editeurs, International Conference on Very Large Data Bases, pages 385–394, Cairo, Egypt, 2000. Morgan Kaufmann.

[Zhang 05] Chengqi Zhang, Qiang Yang & Bing Liu. *Guest Editors' Introduction: Special Section on Intelligent Data Preparation*. IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 9, pages 1163–1165, 2005.

[Zhang 11] Yaodong Zhang & J.R. Glass. *An inner-product lower-bound estimate for dynamic time warping*. In 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 5660–5663, Prague, Czech Republic, May 2011.