UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

OBTENÇÃO DE PADRÕES SEQUENCIAIS EM DATA STREAMS ATENDENDO REQUISITOS DO *BIG DATA*

DANILO CODECO CARVALHO

ORIENTADORA: PROFA. DRA. MARILDE TEREZINHA PRADO SANTOS

São Carlos – SP

Maio/2016

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

OBTENÇÃO DE PADRÕES SEQUENCIAIS EM DATA STREAMS ATENDENDO REQUISITOS DO *BIG DATA*

DANILO CODECO CARVALHO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: ENGENHARIA DE SOFTWARE E BANCO DE DADOS.

Orientadora: Dra. MARILDE TEREZINHA PRADO SANTOS

São Carlos – SP

Maio/2016

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária UFSCar
Processamento Técnico
com os dados fornecidos pelo(a) autor(a)

Carvalho, Danilo Codeco
C331o Obtenção de padrões sequenciais em data streams atendendo requisitos do Big Data / Danilo Codeco Carvalho. -- São Carlos : UFSCar, 2016.
94 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2016.

1. Mineração de dados. 2. Mineração no Big Data. 3. Mineração de data streams. 4. Mineração em fluxos de dados. 5. Processamento de eventos complexos. I. Título.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Dissertação de Mestrado do candidato Danilo Codeco Carvalho, realizada em 06/06/2016.

Profa. Dra. Marilde Terezinha Prado Santos
(UFSCar)

Prof. Dr. Carlos Roberto Valêncio
(Unesp)

Profa. Dra. Marcela Xavier Ribeiro
(UFSCar)

Certifico que a sessão de defesa foi realizada com a participação à distância do membro Prof. Dr. Carlos Roberto Valêncio, e depois das arguições e deliberações realizadas, os participantes à distância estão de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa do aluno Danilo Codeco Carvalho.

Profa. Dra. Marilde Terezinha Prado Santos
Presidente da Comissão Examinadora
(UFSCar)

Agradecimento

A Deus, por me dar forças nos inúmeros momentos que estive longe da minha família e de meus amigos.

A meus pais, que sempre incentivaram, confiaram no meu sucesso e ficaram do meu lado nos momentos mais difíceis da minha vida.

A meu irmão Lucas, por me proporcionar boas risadas mesmo estando longe.

À minha orientadora, Marilde, por me mostrar o caminho da pesquisa científica e a todos os professores que me ajudaram durante a graduação.

Aos colegas do Grupo de Banco de Dados que me auxiliaram.

À minha noiva, Taiani, por estar ao meu lado principalmente nos momentos difíceis do mestrado, me mantendo forte e seguro das minhas decisões.

RESUMO

O crescimento da quantidade de dados produzidos diariamente, tanto por empresas como por indivíduos na web, aumentou a exigência para a análise e extração de conhecimento sobre esses dados. Enquanto nas duas últimas décadas a solução era armazenar e executar algoritmos de mineração de dados, atualmente isso se tornou inviável mesmo em super computadores. Além disso, os requisitos da chamada era do *Big Data* vão muito além da grande quantidade de dados a se analisar. Requisitos de tempo de resposta e complexidade dos dados adquirem maior peso em muitos domínios no mundo real. Novos modelos têm sido pesquisados e desenvolvidos, muitas vezes propondo computação distribuída ou diferentes formas de se tratar a mineração de fluxo de dados. Pesquisas atuais mostram que uma alternativa na mineração de fluxo de dados é unir um mecanismo de tratamento de eventos em tempo real com algoritmos clássicos de mineração de regras de associação ou padrões sequenciais. Neste trabalho é mostrada uma abordagem de mineração de fluxo de dados (*data stream*) para atender ao requisito de tempo de resposta do *Big Data*, que une o mecanismo de manipulação de eventos em tempo real Esper e o algoritmo *Incremental Miner of Stretchy Time Sequences (IncMSTS)*. Os resultados mostram ser possível levar um algoritmo de mineração de dados estático para o ambiente de fluxo de dados e manter as tendências de padrões encontrados, mesmo não sendo possível ler todos os dados vindos continuamente no fluxo de dados.

Palavras-chave: Mineração de dados. Mineração no *Big Data*. Mineração de Data streams. Mineração em Fluxos de Dados. Processamento de eventos complexos. Janelamento. Mineração de padrões sequenciais. Mineração de regras de associação.

ABSTRACT

The growing amount of data produced daily, by both businesses and individuals in the web, increased the demand for analysis and extraction of knowledge of this data. While the last two decades the solution was to store and perform data mining algorithms, currently it has become unviable even to supercomputers. In addition, the requirements of the *Big Data* age go far beyond the large amount of data to analyze. Response time requirements and complexity of the data acquire more weight in many areas in the real world. New models have been researched and developed, often proposing distributed computing or different ways to handle the data stream mining. Current researches shows that an alternative in the data stream mining is to join a real-time event handling mechanism with a classic mining association rules or sequential patterns algorithms. In this work is shown a data stream mining approach to meet the *Big Data* response time requirement, linking the event handling mechanism in real time Esper and Incremental Miner of Stretchy Time Sequences (IncMSTS) algorithm. The results show that is possible to take a static data mining algorithm for data stream environment and keep tendency in the patterns, although not possible to continuously read all data coming into the data stream.

Keywords Data mining. Mining *Big Data*. Data stream mining. Complex event processing. Sliding Window. Sequential pattern mining. Association rule mining.

LISTA DE FIGURAS

LISTA DE TABELAS

Sumário

Capítulo 1
Introdução

Neste capítulo são apresentados os problemas e motivação, a hipótese, o objetivo e a metodologia de pesquisa para a realização deste trabalho, além da justificativa de relevância do assunto abordado.

1.1 Considerações Iniciais

Atualmente vive-se a era da evolução dos dados e da exigência de novas formas de extrair conhecimento útil desses dados. Enquanto ter um grande banco de dados relacional e um algoritmo otimizado de extração de padrões foi considerado o suficiente e gerou enorme quantidade de pesquisas nas duas últimas décadas, a velocidade e o entendimento praticamente imediato das tendências se tornou requisito primordial nos dias de hoje.

O chamado *Big Data*, a nova era em que a quantidade, velocidade e complexidade dos dados atingem níveis nunca antes vistos, é o que desafia a extração de conhecimento atual e traz diferentes abordagens para geração de modelos que solucionem os problemas e respeitem os requisitos do *Big Data*.

As redes sociais, os motores de busca, sensores e computadores embarcados, transações financeiras, todos geram quantidades gigantescas de dados e entender esses dados em tempo real e reagir imediatamente da melhor forma pode ser fundamental para determinar o sucesso de um investimento, de uma campanha de *marketing*, da prevenção de um desastre natural e outros problemas.

1.2 Questões e Motivação

Após a revisão da literatura sobre o tema descrito, as questões que guiaram este trabalho foram:

- Por que são necessárias novas abordagens para a análise de fluxo de dados com requisitos do *Big Data*?
- Os algoritmos clássicos de mineração de dados podem ser úteis com a quantidade e velocidade de dados presentes nos fluxos de dados?
- Como os algoritmos clássicos de mineração de dados podem ser úteis com a quantidade e velocidade de dados presentes nos fluxos de dados?
- É possível unir um algoritmo de mineração de padrões sequenciais com um mecanismo de manipulação de fluxo de dados?
- É possível conseguir desempenho satisfatório com essa união?
- A nova abordagem proposta neste trabalho é relevante para o estado da arte dos assuntos discutidos?

Com o passar dos anos, o problema de gerenciamento de dados para extração de conhecimento útil aumentou devido aos desafios do *Big Data*. Aplicações tradicionais de descoberta de conhecimento e mineração de dados não se adaptam aos requisitos de velocidade, variedade e volume dos dados atuais, para citar apenas alguns empecilhos.

Requisitos de negócios também exigem mais dos modelos de extração de conhecimento e de previsão, além de que aproveitar dados oriundos de diferentes fontes pode ser de grande utilidade para determinadas aplicações.

As abordagens existentes, normalmente, focam em dois caminhos: baseadas em computação distribuída utilizando o modelo de programação *MapReduce* e sua implementação livre, o *Hadoop*, ou em mineração de fluxos de dados.

Apesar de parecer antiquada, a utilização de algoritmos clássicos de mineração de regras de associação e padrões sequenciais unidos aos recursos atuais de mineração em fluxo de dados pode se tornar uma ferramenta importante e de alto desempenho se adaptada de forma que se adeque aos requisitos do *Big Data*. É

possível aliar algoritmos tradicionais e bem fundamentados com o desempenho ótimo exigido pelo ambiente *Big Data*.

1.3 Hipótese

Na revisão bibliográfica de abordagens existentes, foi encontrado apenas um trabalho - *Online Association Rule Mining over Fast Data*, de Ölmezoğulları e Ari (2013) - que se propõe a realizar a adaptação de algoritmos clássicos de mineração de dados para o ambiente *Big Data* com técnicas semelhantes as que foram utilizadas neste trabalho. Os resultados mostraram sucesso na abordagem de Ölmezoğulları e Ari e permitem, naturalmente, a ideia de adaptar outros algoritmos de mineração de dados não destinados, inicialmente, ao *Big Data*.

Os testes e resultados apontaram que essa adaptação pôde proporcionar desempenho suficiente dos algoritmos de mineração de dados que, inicialmente eram exclusivamente para mineração de dados estáticos e *offline*, de forma que se tornem alternativas na mineração de fluxo de dados.

Adicionalmente, o algoritmo *Incremental Miner of Stretchy Time Sequences (IncMSTS)*, de Silveira Junior, Ribeiro e Santos (2013), propõe a mineração de padrões sequenciais para dados esparsos e com ruídos. O algoritmo foi adaptado para a mineração de fluxo de dados, com o apoio do mecanismo Esper, para atender aos requisitos de velocidade exigidos pelo *Big Data*.

Portanto, a hipótese do trabalho foi que é possível unir Esper + IncMSTS e responder de forma satisfatória às questões citados anteriormente.

Os testes e resultados mostraram que a hipótese estava correta e a união de Esper + IncMSTS na obtenção de tendências e padrões sequenciais em mineração de fluxo de dados pode ser realizada.

1.4 Objetivo

Este trabalho visou a adaptação de um algoritmo de mineração de padrões sequenciais para que seja aplicado à mineração de fluxos de dados com desempenho satisfatório no ambiente do *Big Data*.

A abordagem sugerida utilizou técnica de janelamento de dados para se adequar à proposta de alta velocidade e quantidade de dados dos fluxos de dados. O janelamento divide os dados em janelas de tempo ou tamanho para servir de entrada ao algoritmo de mineração de dados sequencial.

A utilização do Esper unido ao algoritmo de mineração de dados estático permitiu alta velocidade de identificação de padrões e a não necessidade de um conjunto prévio de treinamento dos dados. Portanto, o objetivo deste trabalho foi adaptar o algoritmo de extração de padrões sequenciais *Incremental Miner of Stretchy Time Sequences* para as condições atuais de exigência da mineração de fluxo de dados no *Big Data*.

1.5 Metodologia de Desenvolvimento

O desenvolvimento desse trabalho de pesquisa foi pautado nas seguintes atividades:

- levantamento bibliográfico, com o método de revisão sistemática, conforme descrito na próxima subseção,
- reuniões de orientação com a docente orientadora,
- apresentação e participação em seminários organizados pelo Grupo de Banco de Dados do Departamento de Computação da UFSCar,
- estudo e experimentação empírica do algoritmo e ambiente que integraram a abordagem proposta.
- adaptação do algoritmo e ambiente,

- encaminhamento de experimentos para validação da abordagem com dados reais.
- redação de documentos científicos.

1.5.1 Revisão Sistemática

A revisão sistemática é o método de se realizar o levantamento da bibliografia relacionada ao trabalho e identificação do estado da arte, além de trabalhos correlatos. O método possui três etapas bem definidas: Planejamento, Seleção e Extração (Pai et al., 2004).

O planejamento trata de questões iniciais fundamentais para o direcionamento correto, como objetivo da revisão, questões a serem respondidas, resultados esperados, as máquinas de busca que serão utilizadas e critérios para se aceitar ou recusar um artigo, por exemplo.

A etapa de seleção já separa os artigos para leitura de acordo o título, palavras-chave e resumo. Neste trabalho, artigos duplicados foram eliminados manualmente, então a leitura resultou na separação dos artigos apenas em aceitos, que seguiram para a etapa de extração, e rejeitados, que infringiram os critérios de aceitação e não foram considerados. A prioridade de leitura nessa fase também já é determinada.

Por fim, a etapa de extração fornece as informações que se esperavam de cada artigo, novamente classificados em aceitos e recusados. Os artigos aceitos foram parte, definitivamente, da base teórica para o trabalho, enquanto os rejeitados não.

Todo o processo de revisão sistemática citado foi apoiado pela ferramenta StArt (LAPES StArt, 2011), desenvolvida pelo Laboratório de Pesquisa em Engenharia de Software do Departamento de Computação da Universidade Federal de São Carlos.

1.6 Organização do Trabalho

Este trabalho está organizado da seguinte forma: No capítulo 2 é apresentada a conceituação sobre *Big Data*. No capítulo 3 é mostrada a teoria sobre Mineração de Dados. No capítulo 4 é descrita a abordagem proposta Esper + IncMSTS. No capítulo 5, são apresentados os testes e resultados do trabalho. Por fim, no capítulo 6, é mostrada a conclusão do trabalho.

1.7 Considerações Finais

Este capítulo apresentou o problema existente atualmente na abordagem de mineração de fluxo de dados no ambiente do *Big Data*, a motivação para o desenvolvimento deste trabalho, sua hipótese e objetivo. Por fim foram descritos o método para a realização de revisão sistemática e a organização que será encontrada a seguir no decorrer dos capítulos.

Capítulo 2

Big Data

Neste capítulo são apresentados os conceitos e desafios referentes ao Big Data. Além disso, também são mostrados o estado da arte da análise no Big Data e os trabalhos relacionados ao Esper.

2.1 Considerações Iniciais

A IBM estima que 2.5 quintilhões de bytes de dados sejam criados todo dia e 90% dos dados no mundo hoje foram criados nos últimos dois anos (IBM, 2012). Criação de dados e conteúdo *online* não é mais exclusividade de pessoas que possuam algum tipo de autorização para publicação na rede. Qualquer pessoa com acesso à internet pode produzir diferentes quantidades e tipos de dados e exibi-los ao mundo.

Outro ponto a se destacar é a redução do custo de armazenamento de dados ao longo dos anos. Inclusive muitas organizações preferem simplesmente aumentar a quantidade de dados de entrada do que melhorar seus modelos de extração de conhecimento. Isso se mostra perigoso para o futuro, pois tendo por base as ideias que são apresentadas neste capítulo, a utilização de um modelo específico e com desempenho ótimo para o domínio pode gerar vantagens de negócios para seus administradores.

Neste contexto, muitos pesquisadores consideram a era do gerenciamento do *Big Data* como uma das mais importantes e desafiadoras tarefas para os profissionais da tecnologia da informação.

2.2 O que é o *Big Data*?

O conceito de *Big Data* é definido de diferentes formas por diferentes autores. Porém, todos são unânimes em dizer que o *Big Data* trata de conjuntos de dados que continuam crescendo e que torna complicada a tarefa de gerenciamento dos dados utilizando as técnicas e ferramentas existentes (Singh e Singh, 2012; Fan e Bifet 2012).

Gartner (2013) diz que *Big Data* é a informação de alto volume, alta variedade e alta velocidade que demanda processamento de informação inovador para aprimorar a tomada de decisão. É natural perceber, também, que os autores se preocupam com os modelos e com as novas ideias e formas para se gerenciar dados do *Big Data*.

A origem do termo *Big Data* está relacionada ao fato que, atualmente, todos criam uma enorme quantidade de dados todo dia (Fan e Bifet, 2012).

Mas não é só pela quantidade de dados que o *Big Data* deve ser reconhecido, outras características também são importantes e podem ser elencadas como desafios a serem superados, tais como velocidade e variedade dos dados. Um exemplo explorado por Wu et al. (2013), mostra que o debate dos candidatos à presidência dos EUA em 2012 teve mais de 10 milhões de *tweets* em duas horas (Lin e Ryaboy, 2012). Porém, entre todos esses *tweets*, o momento específico que gerou mais discussão foi sobre saúde pública. Esse exemplo mostra que o interesse do público em tempo real pode ser capturado e novas estratégias podem ser pensadas a partir disso. Além disso, o apelo comercial desse tipo de análise de *feedback* é consideravelmente maior que na TV ou no rádio, por exemplo.

2.3 Características e Desafios do *Big Data*

Na tentativa de visualizar o *Big Data* por uma nova perspectiva e tentar diferenciá-lo dos dados clássicos (texto, dados estruturados, transações), os autores sempre tentam enumerar as principais diferenças de forma a exibi-las o mais claramente possível. Assim, uma das descrições mais bem aceitas foi inicialmente proposta por Laney (2001), que cita os 3 V’s do gerenciamento do *Big Data*:

- Volume: existem mais dados que nunca e continua crescendo, mas não cresce o percentual de dados que as ferramentas atuais conseguem processar.

- Variedade: existem muitos tipos diferentes de dados agora para processar, como áudio, vídeo, grafos, postagens em redes sociais, dados de sensores e mais.

- Velocidade: os dados chegam continuamente em fluxos de dados e há interesse em extração de informação útil em tempo real.

Atualmente mais 2 V's foram adicionados às características iniciais (Fan e Bifet, 2012):

- Variabilidade: podem ocorrer mudanças na estrutura dos dados e na forma como os usuários desejam interpretar tais dados.

- Valor: valor de negócio que dê às organizações detentoras de melhor conhecimento vantagem competitiva. Isso acontece devido a habilidade de tomar decisões baseadas em respostas de questões que antes eram consideradas inatingíveis.

Por ser uma nova tendência (Fan e Bifet, 2012), o termo *Big Data* sofre também com alguns pontos de controvérsia. A seguir são listados alguns desses pontos de forma resumida do trabalho de Boyd e Crawford (2012).

- A importância do *Big Data* pode estar sendo exagerada e supervalorizada para se vender sistemas computacionais baseados em *Hadoop* (sistema de computação distribuída para processamento de dados), sendo que não é sempre a melhor plataforma (Lin, 2012), principalmente para pequenas e médias empresas.

- Preocupações éticas sobre acesso aos dados. O anseio por mais dados a se analisar e o modo de conseguir esses dados não podem passar por cima de várias restrições de privacidade existentes. O maior problema é se é ético que pessoas sejam analisadas sem saberem disso.

- Acesso limitado aos dados pode criar barreiras digitais, onde organizações que podem gerenciar melhor o *Big Data* vão conseguir extrair conhecimento e ter vantagens sobre companhias que não poderão ter o mesmo acesso. Pode ser criada a divisão entre ricos e pobres também em relação ao gerenciamento de dados.

- Em análise de tempo real, dados podem estar em constante mudança. Nesse caso, o importante não é o tamanho e sim o caráter recente dos dados.

- Mais dados não quer dizer sempre melhores dados. Depende da quantidade de ruídos dos dados e se são representativos para o propósito buscado. Por exemplo, algumas vezes os usuários do *Twitter* são representativos para o comportamento da população mundial, mas não é sempre esse o caso.

Deixado claro os desafios e controvérsias sobre o *Big Data*, Amatriain (2012) discute se a revolução *Big Data* é apenas falácia. Garante que não é. Mais dados, tanto em termos de mais exemplos como em mais características, pode ser uma benção. A disponibilidade dos dados permite mais e melhores visões e aplicações. Mais dados permite melhores abordagens. Mais que isso: requer melhores abordagens. Em outras palavras, dados são importantes. Mas dados sem tratamento adequado se tornam ruídos.

2.4 Tempo Real

De acordo com o que já foi mencionado neste trabalho, a extração de conhecimento se tornou uma tarefa de extrema importância para pessoas e, principalmente, organizações. Peter Druker disse, em 1992:

> *"De agora em diante, a chave é o conhecimento. O mundo não está se tornando voltado ao trabalho, ao material ou a energia, mas sim voltado ao conhecimento"*

O universo digital em 2007 foi estimado em 281 exabytes ou 281 bilhões de gigabytes para, em 2011, ser 10 vezes maior (Gantz, 2011). Para lidar com essa quantidade de dados de forma responsável, computação verde se tornou uma necessidade.

Computação verde é o estudo e prática de utilizar recursos computacionais de forma eficiente e uma das principais abordagens é baseada em eficiência de algoritmos. A quantidade de recursos computacionais depende da eficiência dos algoritmos usados (Morales, 2013).

Por mais que a mineração de dados atualmente permita manipular grandes bases de dados e tenha se desenvolvido fortemente por pesquisas nas últimas duas décadas, ainda não soluciona o desafio de fluxo de dados, já que, normalmente, após a execução de um algoritmo de mineração de dados estática, os resultados alcançados não podem ser atualizados caso novos dados apareçam. Ou seja, o processo de mineração deve ser repetido para refletir o novo comportamento dos dados.

No paradigma de fluxo de dados, a limitação citada não é permitida. Algoritmos escritos para fluxo de dados podem naturalmente lidar com dados maiores que a memória disponível e atingir o objetivo de extrair conhecimento a partir de dados que são constantemente atualizados. O algoritmo que processa o fluxo de dados não tem controle sobre a ordem que os dados chegam e devem atualizar seu resultado de forma incremental e eficiente (Morales, 2013).

Alguns requisitos precisam ser atingidos para a mineração de fluxo de dados, os mais importantes para este trabalho estão listados a seguir (Morales, 2013):

1. A característica chave para um fluxo de dados é que os dados fluem de um exemplo para outro, ou seja, não há espaço para acesso aleatório.
2. A maior motivação para empregar fluxo de dados é que permite processamento de dados muito maiores que a memória disponível na máquina. Ainda assim, sempre haverá um limite físico de memória em que a máquina poderá trabalhar e o algoritmo deve trabalhar dentro desse limite.
3. Para esse constante aumento de dados, a complexidade do algoritmo deve ser linear em relação ao número de exemplos. A saída perfeita seria um algoritmo capaz de processar os exemplos tão ou mais rápido do que chegam.

Esses requisitos são levados em conta nos testes deste trabalho. Mesmo que os requisitos não sejam atingidos, a discussão pertinente ao motivo desse problema é realizada. Encontrar uma abordagem que solucione todos os problemas é uma tarefa complexa, então na maioria dos casos, ajustar uma boa abordagem ao domínio específico pode ser mais importante que a construção de uma nova solução.

2.5 Análise do *Big Data*

Por mais que a análise de dados tenha evoluído nos últimos anos, com melhores algoritmos e, principalmente, aumento na capacidade de armazenamento de dados, a maioria das ferramentas e tarefas de mineração de dados, aprendizado de máquina e de inteligência artificial não estão preparadas para os V's do *Big Data*.

O aumento da quantidade de dados é ligado ao também aumento da capacidade de armazenamento. Porém, os SGBDs Relacionais atuais não estão preparados para a era do *Big Data* em que milhares de novos nós possam ser adicionados em um *cluster*. Os motivos, de acordo com (Qin et al., 2012), são (i): as restrições ACID (Atomicidade, Consistência, Isolamento, Durabilidade) necessárias para as transações de todo banco de dados e (ii): SGBDs tradicionais não são capazes de manipular dados semi-estruturados e não estruturados, não atingindo, assim, performances aceitáveis.

A análise sobre o *Big Data* tem recebido recentemente grande atenção da comunidade acadêmica (Cuzzocrea, Song e Davis, 2011).

Em 2004, a Google publicou sua técnica de processamento de dados não estruturados paralela, chamado *MapReduce* (Dean e Ghemawat, 2004), causando uma nova onda de estudos nas comunidades de pesquisa e indústria, gerando também um novo cenário de competição na corrida de gerenciamento de dados.

2.5.1 MapReduce

Depois da publicação e do amplo estudo sobre *MapReduce*, geralmente tudo que se refere a análise do *Big Data* hoje remete ao *paradigma de programação* desenvolvido pela Google.

O *MapReduce* provê um modelo de programação usando funções de *map,* para mapear e *reduce,* para reduzir sobre pares chave-valor que podem executar em paralelo em um grande *cluster* com inúmeros nós (Mukherjee et al. 2012). O modelo de *MapReduce* da Google se tornou popular ao longo dos anos na implementação livre do Apache *Hadoop* (Apache Hadoop, 2012). O *Hadoop* também oferece um Sistema de Arquivo Distribuído o *Hadoop Distributed File System* - HDFS.

Para exemplificar o funcionamento do *MapReduce*, um paradigma de computação distribuída inspirado por conceitos de linguagens funcionais, a seguir um exemplo extraído de Morales (2012), com o fluxo de dados mostrado na Figura 2.1.

Os *mappers* leem os dados do Sistema de Arquivo Distribuído, que está, usualmente, no mesmo nó computacional para que a leitura seja, na maior parte do tempo, de dados locais (Mukherjee et al. 2012).

Cada *mapper* lê uma fatia da Entrada, *Input*, aplica a função mapeamento para o par chave-valor e produz um ou mais pares de saída. Os *mappers* ordenam e escrevem essas saídas intermediárias no disco local.

Cada *reducer* busca os dados de diferentes locais. Pares chave-valor intermediários já foram particionados pelos *mappers*, então o *reducer* apenas realiza o merge entre as diferentes partições para agrupar as mesmas chaves juntas. Essa é a fase *Shuffle* e a mais cara em termos de operações de entrada e saída. A *Shuffle* pode ser parcialmente feita ao mesmo tempo da fase inicial de mapeamento, permitindo que resultados intermediários de alguns *mappers* possam ser transferidos tão logo sejam escritos em disco.

Na última fase, cada *reducer* aplica a função de redução para o par chave-valor intermediário e escreve o resultado final, *Output* no sistema de arquivo.

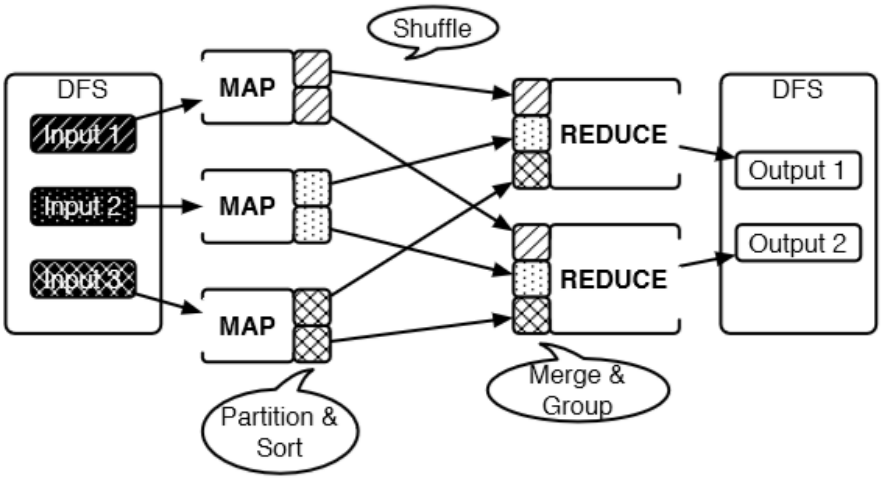


Figura 2.1 – Funcionamento do MapReduce - adaptada de Morales (2012). Note, com o apoio das texturas de bloco, que os *mappers* puxam os dados de entradas específicas e apenas na fase de *shuffle* que os dados são agrupados por seus identificadores e reduzidos (*merge*) para as saídas esperadas.

Aplicando de forma simples para melhorar o entendimento, imagine um índice invertido para um sistema de busca *online*. O algoritmo utilizando o *MapReduce* lê documentos web presentes no DFS e para cada palavra, na fase de mapeamento, forma o par (palavra, id_documento).

A fase de redução agrupa os identificadores de documento associado com a mesma palavra chave (palavra1, [id_documento1, id_documento3, id_documento4, ...]).

2.5.2 SGBDs Relacionais

Desde sua criação na década de 70, o modelo relacional tem sido amplamente estudado e utilizado em bancos de dados acadêmicos e comerciais. SGBDs Relacionais evoluíram e se tornaram a grande força no mercado de banco de dados. Suas características, citadas por Qin et al. (2012), são:

- Altamente estruturado;
- Possível desenvolver índices que melhoram o desempenho;

- Linguagem padrão para acessar os dados - SQL;
- Separação do armazenamento físico e esquema lógico;
- Alta consistência, confiabilidade e performance;
- Dificuldade de expansão compatível com o nível *Big Data* e;
- Dificuldade com dados semi estruturados e não estruturados.

As dificuldades encontradas pelos modelos relacionais praticamente os impossibilitam de serem utilizados de forma satisfatória no *Big Data*, porém, é importante lembrar também que nem sempre é necessário utilizar uma solução baseada em *MapReduce*.

Como é mostrada na wiki do Apache *Hadoop* (APACHE Hadoop, 2012), um número significativo de implementações do *Hadoop* em empresas não ultrapassa o número de 16 nós. Sendo assim, o papel do tradicional modelo de armazenamento compartilhado em um *cluster* pode servir tanto para análise convencional de dados quanto para análise do *Big Data* (Mukherjee et al. 2012).

Na Tabela 2.1, adaptada de Qin et al. (2012), são mostradas as principais diferenças entre os paradigmas de modelo de dados relacional e *MapReduce*.

Tabela 2.1: Comparação entre SGBD Relacional e o paradigma *MapReduce*

Item de comparação	SGBD Relacional	*MapReduce*
Suporte a esquema	Esquema rigoroso	Sem esquema, porém, depois de aplicar algumas estruturas ao HDFS, o M*apReduce* pôde manipular dados estruturados (Kaldewey et al, 2012; He et al. 2011)
Índice	Vários índices disponíveis	Sem índices, porém há pesquisas nesse aspecto
Modelo de programação/ flexibilidade	SQL, uma linguagem declarativa de fácil utilização Análises complexas necessitam de funções definidas pelo usuário usando outra linguagem.	Funções de mapeamento e redução são de baixo nível de interface, não sendo intuitivo de usar. Muito flexível.
Otimização	Otimizador complexo tenta encontrar plano de execução ótimo para as consultas	Precisa ser evoluída, mas já existem trabalhos de otimização de algoritmos complexos de *MapReduce* nos últimos tempos.
Tolerância à falha	Baixa	Alta
Escalabilidade	Baixa, escabilidade limitada para centenas de nós	Alta, pode ser escalável a mais de 1000 nós
Heterogeneidade	Suporte limitado	Suporte completo

2.6 CEP

Fluxo de dados são contínuos e infinitos, portanto, é difícil tratá-los usando os SGBDs Relacionais existentes (Jaein et al., 2012). Fluxo de dados na era do *Big Data* tendem a ser ainda mais trabalhosos de acordo com os V's que o descrevem.

Conforme já explicado, em um ambiente clássico e estático, os dados são primeiramente armazenados e só depois consultados para extração de conhecimento. Porém, o ambiente de fluxo de dados requer novo métodos, pois não há a possibilidade de armazenar todos os dados vindos do fluxo (Arasu et al., 2004).

Um novo sistema discutido atualmente é conhecido como *Complex Event Processing*, CEP, e tem sido utilizado em vários campos para aplicações em tempo real e fluxo de dados. Se em SGBDs Relacionais as consultas em SQL são feitas em dados armazenados, no ambiente de fluxo de dados é difícil aplicar o mesmo método (Jaein et al., 2012).

O sistema CEP é utilizado no desenvolvimento de aplicações que lidam com fluxos de dados de entrada volumosos com o objetivo de encontrar eventos significativos ou padrões de eventos, e responder aos eventos de interesse em tempo real (Bhargavi et al., 2010). Além disso, os sistemas CEP normalmente fornecem uma função de janelamento dos dados e ferramentas que permitem a manipulação da janela (Jaein et al., 2012). Isso pode ter papel fundamental na análise de certos domínios no *Big Data*, contendo da velocidade, quantidade e complexidade dos dados, além de poder filtrar boa parte de dados que se comportam apenas como ruídos, lembrando a citação de Amatriain (2012) de que maior disponibilidade de dados sem tratamento pode não trazer benefício algum.

Como sistema CEP, destaca-se o Esper, ***E****vent* ***s****tream and com****p****lex* ***e****vent* ***p****rocessing,* (Esper website, 2014). De forma mais detalhada, destacam-se os dois pontos de união do nome Esper da seguinte forma, adaptado de (Astrova, Koschel e Schaaf, 2012):

- ESP é a habilidade de capturar conjuntos infinitos de eventos de complexidade arbitrária e reagir em tempo real para situações vindas em

fluxo de eventos, portanto, é comum a utilização de janelas deslizantes para que seja considerado apenas partes menores do fluxo de eventos e;

- CEP é a habilidade de detectar padrões entre eventos, assim como relações entre eventos baseados em causalidade e tempo. Como exemplo de padrão, detectar uma situação onde algum(ns) evento(s) ocorre(m) antes de outro(s).

O Esper provê uma linguagem parecida com SQL chamada EPL, ***E****vent* ***P****rocessing* ***L****anguage*, que, como já foi destacado que os sistemas CEP normalmente possuem, tem funções que permitem a aplicação de técnicas de janelamento sobre fluxos de dados.

O sistema Esper funciona como uma base de dados invertida. Ao invés de armazenar os dados e executar as consultas, o Esper permite aplicações para armazenar consultas, executá-las em dados momentâneos e retorna respostas quando as condições da consulta forem satisfeitas (Bhargavi et al., 2010).

O trabalho de Ölmezoğulları e Ari (2013) mostra a possibilidade efetiva de se adaptar algoritmos clássicos de MD para o ambiente de fluxo de dados por meio do Esper. Os autores incluíram, com sucesso e utilizando as opções de janelamento do Esper, os algoritmos Apriori e FP-Growth. Portanto, também é possível inserir outros algoritmos de MD tradicional no mecanismo Esper. Mais detalhes sobre esse experimento são dados no capítulo 4.

2.6.1 Esper

Analisar e reagir a informações em tempo real requer o desenvolvimento de aplicações personalizadas. Essas aplicações normalmente obtêm e filtram os dados, derivam informações e então as indicam através de algum formulário de apresentação.

Os dados podem chegar com alta frequência e requisitar saídas imediatas ou próximas disso. Pode ser exigido das aplicações serem flexíveis o suficiente para reagir às mudanças enquanto os dados são processados. Esper é um processador

de fluxo de eventos que visa permitir o mínimo de desenvolvimento desde a implantação da ideia até a produção para esses tipos de aplicações (Esper website, 2014).

A Linguagem de Processamento de Eventos, *Event Processing Language – EPL,* é uma linguagem SQL padrão com extensões, oferecendo cláusulas SELECT, FROM, WHERE, GROUP BY, HAVING e ORDER BY. Os fluxos de dados substituem as tabelas como a origem dos dados e os eventos substituem as tuplas como a unidade básica de dados. Já que os eventos são compostos por dados, os conceitos SQL de relação através de junção, filtros e agregação podem ser utilizados. Declarações EPL são utilizadas para derivar e agregar informação de um ou mais fluxo de eventos (Esper website, 2014).

O Esper é um componente para Processamento de Eventos Complexos, *Complex Event Processing – CEP*, disponível em Java como Esper, e em .NET como NEsper.

2.6.2 Por quê o Esper?

O Esper foi o mecanismo de janelamento de fluxo de dados escolhido para este trabalho por ser uma ferramenta livre e com ampla divulgação em trabalhos científicos. Além disso, foi utilizado com sucesso no trabalho de Ölmezoğulları e Ari (2013).

O mecanismo Esper traz diversas bibliotecas e toda a documentação necessária para que o desenvolvedor escolha se prefere inseri-lo em seu projeto em andamento ou criar um projeto que cuide apenas do recebimento do fluxo de dados e seu janelamento. Neste trabalho, optou-se pela segunda opção.

Além disso, o Esper permite:

- Alta vazão – aplicações que processam grandes volumes de eventos, entre 1000 e 100.000 eventos por segundo. Neste trabalho, os eventos serão as tuplas da base de dados.

- Baixa latência – aplicações que reagem em tempo real quando as condições ocorrem. Neste trabalho, a reação exigida do Esper foi empacotar os dados de uma janela em um lote em um arquivo formatado para entrada do IncMSTS.
- Computação complexa – aplicações que detectam padrões e filtram eventos, agregações entre eventos de janelas por tamanho e por tempo, junção de séries de eventos, *triggers* baseadas na falta de eventos. Neste trabalho não foram utilizados recursos desse ponto.

2.6.3 Janelamentos do Esper

O janelamento utilizado no Esper para prover os arquivos de entrada para o algoritmo IncMSTS foi do tipo *Tumbling Window*. Mais detalhes desse tipo de janelamento são encontrados na seção 3.4.2.

No Esper esse tipo de janelamento é chamado de *Time Batch Window* ou Janela por Lotes de Tempo. A Janela por Lotes de Tempo armazena os eventos e os libera em todo intervalo de tempo especificado. A seguir um exemplo do funcionamento desse tipo de janelamento.

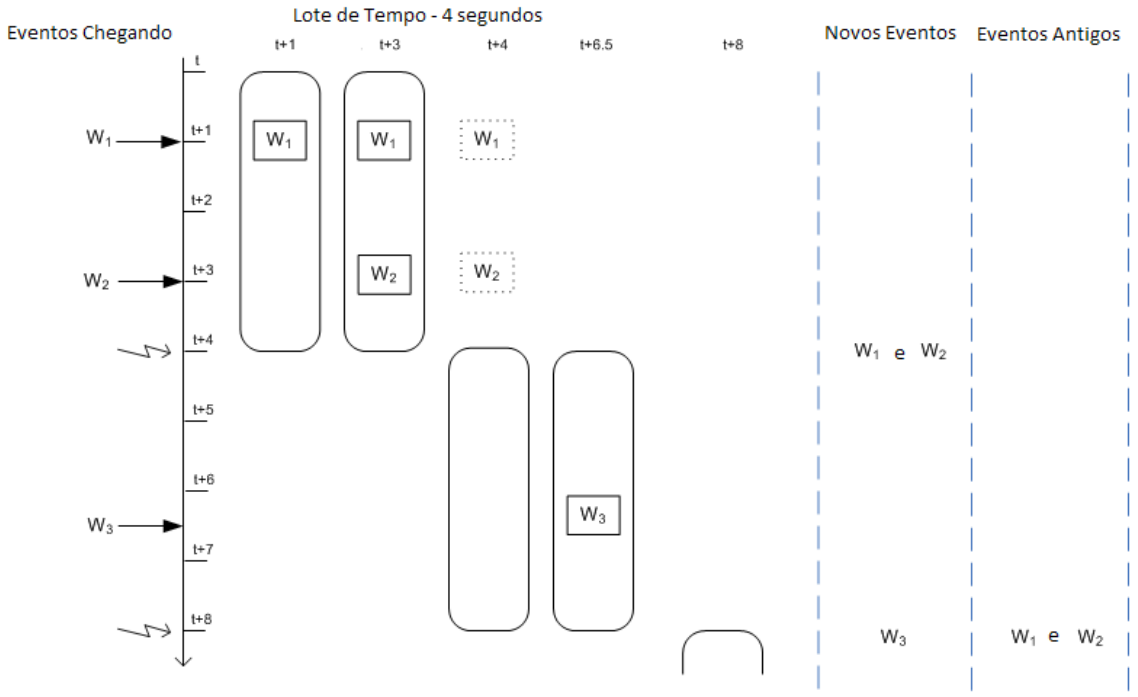


Figura 2.2 - Exemplo de janelamento por lote de tempo (Esper website, 2014).

Na Figura 2.2 é mostrado o funcionamento de uma Janela por Lotes de Tempo. Assume-se uma consulta:

```
select * from Withdrawal.win:time_batch(4 sec)
```

O diagrama inicia num dado tempo t e mostra o conteúdo da janela de dados em t+4, t+5 segundos e assim por diante.

Os passos a serem seguidos são:

1. No tempo t+1 segundos, um evento W1 chega e entra no lote. Não há chamada para informar o *listener*, que é o processo ouvinte que ficará atento a todas as tuplas que satisfazem as condições de janelamento e poderão compor os dados formatados de saída.
2. No tempo t+3 segundos, um evento W2 chega e entra no lote. Não há chamada para informar o *listener*.
3. No tempo t+4 segundos, o mecanismo processa os eventos armazenados no lote e inicia um novo lote. O mecanismo reporta os eventos W1 e W2 para o *listener*.
4. No tempo t+6.5 segundos, um evento W3 chega e entra no lote. Não há chamada para informar o *listener*.

2.6.4 Composição de um projeto Esper

Um projeto Esper deve ter alguns itens que permitam ao mecanismo realizar o objetivo ao qual foi proposto.

A seguir, são listados os itens incluídos neste trabalho, assim como a explicação relacionada.

- Importação de bibliotecas:

```
import com.espertech.esper.client.*;
```

Essa é a importação de todo o pacote client do Esper. Seria possível importar apenas alguns pacotes dentro desse escopo, mas como não houve queda de desempenho da ferramenta, optou-se por deixar a importação um pouco mais genérica.

Detalhes do que foi utilizado dessa importação serão dados a seguir.

- Criação de regra:

  Supondo o mecanismo Esper em funcionamento, é hora de criar a primeira declaração EPL.

  O Esper tem a capacidade de filtrar de forma inicial alguns dados em fase de pré-processamento para melhorar o desempenho. Porém, como é importante para o escopo do trabalho a eficiência do algoritmo IncMSTS, o Esper cumpriu apenas o papel de criar as janelas e organizá-las em diferentes arquivos, compatíveis com a entrada esperada pelo IncMSTS.

```
public static void main(String[] args) {...

     cepConfig.addEventType("EsperIncMSTS", Tick.class.getName());
     EPServiceProvider cep =
          EPServiceProviderManager.getProvider("myCEPEngine", cepConfig);
     EPRuntime cepRT = cep.getEPRuntime();

     EPAdministrator cepAdm = cep.getEPAdministrator();
     EPStatement cepStatement = cepAdm.createEPL("select * from " +
          "EsperIncMSTS(symbol='JANELA').win:time_batch(3 sec)");
            ...

}
```

Nesse caso de exemplo, as janelas são do tipo *batch,* ou seja, a cada 3 segundos, um lote de informações é unido e lançado como saída do mecanismo.

- Criação do *Listener*

  O próximo passo é a criação de um *Listener* e conectá-lo aos eventos gerados pela regra de seleção. O *Listener* imprime o objeto gerado pelo mecanismo. Esse objeto, obviamente, deve ser formatado para servir de entrada para o IncMSTS.

```
public void update (EventBean[] newData, EventBean[] oldData) {...}
```

  O Esper mantém armazenado em oldData a saída anterior, caso ainda seja necessária para manipulação junto aos dados atuais. Novamente,

como o IncMSTS implementa a mineração incremental, os novos dados sempre foram adicionados como incrementos ao algoritmo, não sendo necessária a utilização de oldData.

Esses são os componentes fundamentais de funcionamento do Esper. Outros foram utilizados neste trabalho e muitos outros são contemplados pela ferramenta.

2.6.5 Trabalhos Relacionados ao mecanismo Esper

Não foi encontrado trabalho relacionado a este projeto na literatura, exceto o de Ölmezoğulları e Ari (2013), que é explicado no Capítulo 4. Assim, esta subseção tem alguns trabalhos que utilizam o Esper em abordagens diferentes.

O trabalho de Chen e Chen (2014), utiliza o Esper em um Sistema de Detecção de Intrusão, IDS, para identificar os requisitos de aplicação e reagir a diferentes eventos de segurança em redes IoT, *Internet of Things*. O Esper pode processar consultas estáticas e pré-definidas como entrada para fluxo de eventos IoT. Os resultados do processamento são dados em tempo real, gerados automaticamente e contínuos.

Um IDS integrado com o Esper pode ser usado para lidar com padrões entre eventos e processar grande volume de dados com baixa latência. Ao processar esse fluxo de dados gerado pelas redes IoT, identificar padrões e responder em tempo real, o IDS pode reagir rapidamente em caso de ataques de hackers e atividade maliciosa em IoT.

O trabalho de Jayan e Rajan (2014) também tem foco em segurança de rede. Segurança efetiva de rede visa parar uma variedade de ameaças de invadir ou se espalhar pela rede. No escopo de segurança de rede, os sistemas CEP podem ser usados para relacionar eventos através de diferentes dispositivos de segurança e aplicações para detecção e resposta de ataques. Esses eventos são gravados em arquivos de log, onde milhões de eventos são gerados por cada dispositivo de segurança e, assim, o sistema CEP deve processar uma grande quantidade de logs. Os autores propõem um método de pré-processamento em que grande quantidade

de dados de entrada sejam resumidos em dados relevantes por meio da utilização do Esper.

Um trabalho com foco diferente é o proposto por Anh e Datta (2014), que visa o problema de privacidade dos dados na nuvem, particularmente no controle de acesso em fluxo de dados. Os autores apresentam o Streamforce, um sistema que permite aos donos de dados terceirizar com segurança seus dados por uma nuvem não confiável. O Streamforce é implementado sobre o Esper por seu desempenho e ser de código aberto.

O custo de segurança é grande o suficiente para impedir o sistema de atingir vazão de dados máxima, comparado com a vazão máxima do Esper. Entretanto, segundos os autores, a vazão atingida é suficiente para diversas aplicações do mundo real em que os dados não chegam em alta velocidade.

2.7 Considerações Finais

Neste capítulo foram introduzidos os conceitos relacionados ao *Big Data*, suas características e análise. Além disso, foi detalhado um ponto fundamental deste trabalho, o mecanismo Esper.

O aprendizado gerado a partir desses conceitos guiaram a hipótese e os testes deste trabalho, de forma a se explorar um ambiente similar ao que encontra em *Big Data*, com fluxo de dados, propiciar mineração de dados em tempo real e com a utilização de ferramenta de janelamento de dados e adaptação do algoritmo de mineração de dados estática para o fluxo de dados.

Os capítulos seguintes mostram o conceito sobre o IncMSTS, o desenvolvimento, a execução e a análise dos resultados que foram alcançados durante esta pesquisa de mestrado. Mais especificamente, o capítulo 3 traz em detalhes o IncMSTS e o capítulo 4 a união que recebeu o nome de abordagem Esper + IncMSTS.

Capítulo 3

Mineração de Dados

Neste capítulo são apresentados os conceitos referentes à mineração de dados tradicional, assim como a teoria necessária sobre mineração de fluxo de dados, técnicas de janelamento e o algoritmo IncMSTS.

3.1 Considerações Iniciais

Como citado anteriormente, há um aumento natural da quantidade de dados a serem armazenados em todos os setores que utilizam um banco de dados para suas aplicações. Assim, a maneira de obter informação útil a partir desses dados armazenados passa a ser cada vez mais importante.

O Processo de Descoberta de Conhecimento em Base de Dados, *Knowledge Discovery in Database – KDD*, é o processo de identificação de novos, não triviais, válidos e potencialmente úteis padrões nos dados (Fayyad, Piatetsky-Shapiro e Smyth, 1996).

De acordo com Fayyad, Piatetsky-Shapiro e Smyth (1996), os passos para o processo de KDD são: inicialmente deve-se compreender o domínio da aplicação. Após deve-se identificar as metas a fim de personalizar suas etapas para atingir aos objetivos desejados. Então os dados são pré-processados para gerar uma estrutura que facilite a aplicação de um método de mineração. Levando em conta cada meta definida anteriormente, o método de mineração de dados já pode ser escolhido e resultará em padrões selecionados e classificados de acordo com o método. A interpretação dos padrões resultantes envolve visualização e extração de padrões ou modelos. Neste passo, pode-se retornar a outras etapas caso os resultados não sejam satisfatórios. A consolidação do conhecimento descoberto é uma etapa que inclui a checagem e resolução de possíveis conflitos resultantes das etapas anteriores.

Na Figura 3.1 a seguir são mostrados os passos do processo de KDD até o usuário ter condições de usar o conhecimento adquirido para tomar ações corretas baseadas nas características dos dados.

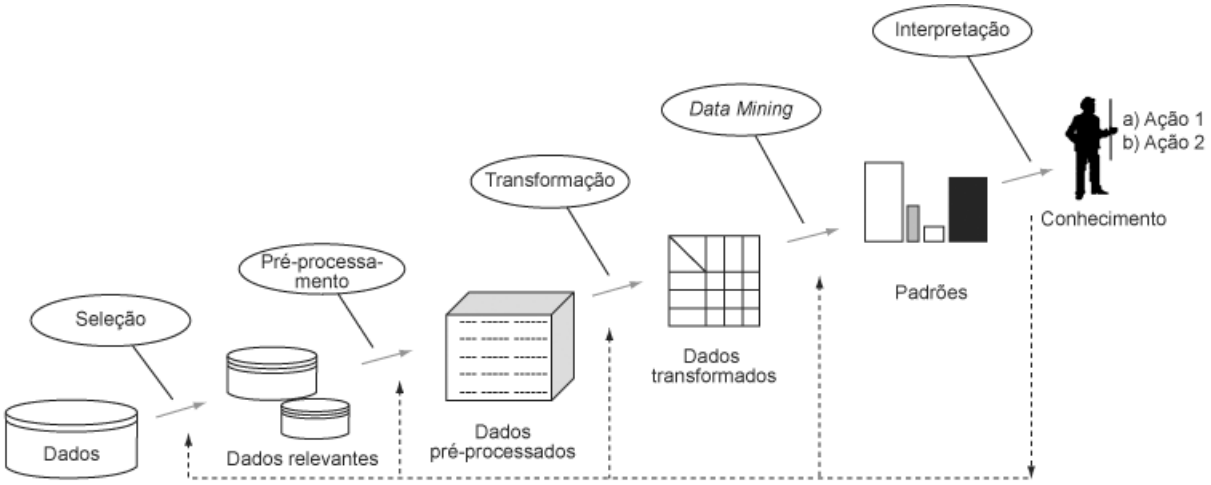


Figura 3.1 – O processo de Descoberta de Conhecimento Útil (adaptada de Fayyad, Piatetsky-Shapiro e Smyth, 1996).

Após completar o processo, pode-se garantir que o conhecimento obtido a partir da base de dados é válido já que as etapas têm grande interferência humana, o que também exige atenção. Como visto na Figura 3.1, após o final do processo é possível retornar aos passos anteriores (Han e Kamber, 2006).

3.2 Mineração de Dados (MD)

A Mineração de Dados (MD) ou *Data Mining* é o processo de extração de padrões que se repetem frequentemente em uma base de dados por meio de aplicação de algoritmos. A mineração de dados pode ser aplicada a um conjunto de dados visando dois objetivos: verificação e descoberta de conhecimento (Fayyad, Piatetsky-Shapiro e Smyth, 1996).

Por ser feita por meio de algoritmos, a MD é a etapa de maior automatização computacional e também se torna uma etapa complicada do processo de KDD.

Em Silberschatz, Korth e Sudarshan (2006) é apresentada uma explicação, a partir de uma analogia com o ambiente comercial, sobre como os conceitos mineração de dados e *Business Intelligence* estão inter-relacionados. O papel da Mineração de Dados é comparado com o papel de um balconista que conhece bem sua freguesia e tenta sempre melhorar seu atendimento a partir do conhecimento que possui, enquanto *Business Intelligence,* Inteligência de Negócios, é comparado a um gerente de estoque que, baseado em dados, procura não deixar faltar material e mercadoria no estoque da empresa.

Para revelar os padrões contidos nas bases de dados existem diferentes tarefas de mineração de dados, estas são diferenciadas pelo tipo de padrão revelado, as tarefas mais comuns são: extração de regras de associação (RA), extração de padrões sequenciais, extração de séries temporais, classificação e agrupamento (Han e Kamber, 2006; Elmasri e Navathe, 2005). O foco deste trabalho está nas tarefas de extração de padrões sequenciais, detalhado na subseção 3.2.2.

3.2.1 Regras de Associação

São padrões do tipo {antecedente} => {consequente}. A obtenção de regras de associação é uma tarefa muito aplicada em domínios comerciais. É baseado no fato de encontrar relações que ocorrem frequentemente no conjunto de dados. Por exemplo, suponha uma base de dados que armazene os dados de compra de um supermercado em um dia. Estes dados podem apresentar o seguinte padrão: muitas pessoas que compram pão também compram leite gerando a regra *Pão* => *Leite*, então uma estratégia de venda poderia ser colocar produtos relacionados entre pão e leite, como margarina e achocolatado, maximizando a venda destes produtos.

A mineração por regras de associação extrai a informação *Pão* => *Leite* da base de dados se esta ocorrer com uma frequência maior ou igual à mínima necessária (ajustada pelo usuário). A MD por descoberta de regras de associação é amplamente utilizada em domínios comerciais (Silberschatz, Korth, Sundarshan, 2006).

O exemplo anterior é simplista e não diz qual o critério para uma relação ser considerada frequente. Existem medidas associadas às regras de associação, tais

como suporte e confiança, que dão apoio à escolha da relação equivalente ao desejo do usuário.

A medida de Suporte representa o percentual de vezes que os itens presentes na regra aparecem no conjunto de transações.

Suporte (X→Y) = (ocorrências de X U Y)/número total de transações.

A medida de Confiança indica o percentual de ocorrência da regra.

Confiança (X→Y) = Suporte(X U Y)/Suporte(X).

Tabela 3.1: Exemplos de transações realizadas em um supermercado.

ID	Itens comprados
01	Pão, Açúcar, Manteiga
02	Leite, Pão, Açúcar, Cerveja, Manteiga
03	Chocolate, Iogurte, Leite, Pão
04	Manteiga, Leite, Pão, Açúcar, Iogurte
05	Pão, Refrigerante, Fralda, Chocolate, Açúcar, Manteiga
06	Iogurte, Leite, Refrigerante

Na Tabela 3.1 são mostradas as transações de um comércio e os itens comprados em cada uma. Para esse caso, o cálculo do suporte e confiança é a seguinte:

Suporte (Pão→Açúcar} = (Ocorrências (Pão U Açúcar) / Total de Transações).
Suporte (Pão→Açúcar) = 4/6 = 0.666.

Confiança (Pão→Açúcar) = (suporte(Pão U Açúcar) / suporte (Pão)).
Confiança (Pão→Açúcar) ≈ 0.8.

O conceito de mineração de fluxos de dados não existia antes dos anos 2000 (Ölmezogullari e Ari, 2013). Assim, algoritmos clássicos de Mineração de Regras de Associação como o Apriori (Agrawal e Srikant, 1994) e o FP-Growth (Han, Pei e Yin, 2000) foram desenvolvidos para mineração estática ou tradicional.

Portanto, pode ser avaliada a possibilidade de adaptar esses algoritmos, assim como tantos outros, para realizar mineração de dados no *Big Data*.

3.2.2 Padrões Sequenciais

A mineração de padrões sequenciais descobre uma subsequência frequente em uma base de dados, suas aplicações incluem, principalmente, a análise de padrões de compras dos consumidores ou padrões de acesso na Web (Pei et al., 2001; Pei, Han e Wang, 2002). A mineração de sequências traz um novo fator à extração de conhecimento útil em uma base de dados. Não é necessário considerar a data, mas sim a ordem em que os elementos aparecem.

Tabela 3.2: Exemplo de transações de loja de eletrônicos - adaptada de Amo (2003)

IdCl	Itemsets	Data
1	{TV, ferro-elétrico}	10/02/2002
2	{sapato, aparelho-de-som, TV}	01/03/2002
1	{sapato, lençol}	03/03/2002
3	{TV, aparelho-de-som, ventilador}	04/03/2002
2	{lençol, Vídeo}	05/03/2002
3	{Vídeo, fitas-de-vídeo}	07/03/2002
1	{biscoito, açúcar}	10/04/2002
4	{iogurte, suco}	14/04/2002
4	{telefone}	21/04/2002
2	{DVDPlayer, fax}	23/04/2002
3	{DVDPlayer, liquidificador}	28/04/2002
4	{TV, Vídeo}	30/04/2002

O exemplo presente em Amo (2003) e mostrado na Tabela 3.2 trata de um problema de negócio relacionado às compras de consumidores. Uma sequência ou padrão sequencial de tamanho k (ou k-sequência) é uma coleção ordenada de itemsets $< i_1,i_2,...,I_n >$. Por exemplo, s = {TV, aparelho de som}, {Vídeo}, {DVDPlayer} é um padrão sequencial. Note que o padrão s ocorre com os clientes identificados com os IdCL 2 e 3. Ambos os clientes compram, num primeiro momento (não importa quando), TV e aparelho de som (em conjunto), depois um Vídeo Cassete e tempos mais tarde um DVDPlayer. Suponha que você, como gerente, decide que um padrão sequencial que se manifesta em pelo menos 50% dos clientes registrados será considerado frequente. Neste caso, o padrão s acima será considerado frequente, pois ocorre em 6 de 12 transações. Caso você seja muito exigente e decida que o mínimo para ser considerado frequente é que pelo menos 70% dos clientes manifestem tal comportamento. Então o padrão s acima não será considerado frequente.

Definição: Sejam s e t duas sequências, s =< $i_1 i_{2...} i_k$ > e t =< $j_1 j_2 ... j_m$ >. Diz-se que s está contida em t se existe uma subsequência de itemsets em t, $l_1,...,l_k$ tal que $i_1 \subseteq l_1,...,i_k \subseteq l_k$. Por exemplo, sejam t =<{1 3 4} {2 4 5} {1 7 8}> e s =<{3} {1 8}>. Então, é claro que s está contida em t, pois <{3}> está contido no primeiro itemset de t e <{1 8}> está contido no terceiro itemset de t. Por outro lado, a sequência s0 =< {8};{7} > não está contida em t, pois indica que <{8}> vem antes de <{7}>, o que não acontece na sequência t.

Para facilitar o entendimento, na Tabela 3.3 é mostrado um exemplo retirado de Agrawal e Srikant (1994).

Tabela 3.3: Transações de exemplo - adaptada de Agrawal e Srikant (1994)

Id de usuário	Sequência de compra
1	<{30} {90}>
2	<{10 20} {30} {40 60 70}>
3	<{30 50 70}>
4	<{30} {40 70} {90}>
5	<{90}>

O processo de descoberta das sequências que atendem ao suporte mínimo é iterativo, porém, apenas as sequências de tamanho máximo aparecerão como respostas no final.

Exemplo: Supondo um suporte mínimo de 25%, ou seja, sequências que apareçam em pelo menos dois usuários, o início é dado decidindo quais sequências com apenas um item cumprem essa regra de suporte. No caso mostrado, as sequências <{30}>, <{40}>, <{70}> e <{90}> superam o suporte exigido.

O próximo passo n da iteração é tentar formar sequências de tamanho n e que, obrigatoriamente, sejam formadas pelas sequências de tamanho menor n-1 já identificadas. Nesse caso, as sequências de tamanho n=2 são: <{30} {40 }>, <{30} {70}>, <{30} {90}> e <{40 70}> . Note que a sequência <{30} {70}> é formado a partir dos clientes 2 e 4. O cliente 3 possui a sequência <{30 70}>, que é diferente e aparece apenas em um cliente e, portanto, não atende ao suporte mínimo.

A seguir, a iteração 3 gera a nova sequência <{30} {40 70}> e mantém a sequência <{30} {90}> da iteração anterior, pois o {90} não é contemplado na sequência de tamanho maior.

3.3 Mineração de Fluxo de Dados (*Data Stream*)

O rápido crescimento em busca de novas informações trouxe desafios para a tarefa de mineração de dados. Muitas fontes, atualmente, produzem dados continuamente. Por exemplo, redes de sensores, gravações telefônicas, dados multimídia e científicos, transações online, redes sociais, etc. (Gama e Gaber, 2007).

Um fluxo de dados é uma sequência contínua de itens em tempo real onde (Golab e Ozsu, 2003):

- não é possível controlar a ordem que os itens chegam.
- não é inteligente armazenar todos os dados do fluxo localmente.

Adicionalmente, Gama e Gaber (2007) definem fluxo de dados como uma sequência ordenada de instâncias que podem ser lidas apenas uma ou poucas vezes usando computação e armazenamento limitados. Lembrando que a velocidade e a complexidade dos fluxos de dados continuam a aumentar, é importante pensar em novos modelos e estratégias que visem maior eficiência para minerar esses dados, ao invés de apenas melhorar o hardware com servidores e máquinas mais potentes.

A distinção de dados atuais e dados antigos é a alimentação automática. Além de pessoas, existem computadores e sistemas autônomos inserindo e trocando informações entre si (Muthukrishnan, 2005).

A Mineração de Dados permite manipular grandes conjuntos de dados, mas não resolve o problema de chegada contínua de dados (Bifet et al, 2011). Normalmente, o conjunto de dados está isolado e não permite a chegada de novos dados depois que se decide aplicar a MD.

Normalmente se diferenciando bastante das tarefas de mineração de dados já descritas, a Mineração de Fluxo de Dados não visa analisar dados armazenados

por, possivelmente, longos períodos de tempo. Lembrando os exemplos citados anteriormente, é fácil notar que alguns domínios de aplicação necessitam de análise em tempo real. Imagine, por exemplo, armazenar dados de redes sociais pra extrair algum tipo de informação. Além da quantidade, outro problema é que as informações podem se tornar obsoletas rapidamente e acabarem sendo inúteis caso passe o momento específico da tendência a ser identificada. Outro ponto importante diz respeito aos Sistemas Gerenciadores de Banco de Dados (SGBD) tradicionais, que não foram desenvolvidos para dar suporte direto para consultas contínuas realizadas por um sistema de mineração de fluxo de dados (Babcock et al, 2002).

Na Figura 3.2 a seguir é mostrada, de forma simplificada e adaptada de Rajaraman e Ullman (2012), como seria o esquema de um Sistema Gerenciador de Fluxo de Dados, em analogia com o conhecido Sistema Gerenciador de Banco de Dados (SGBD).

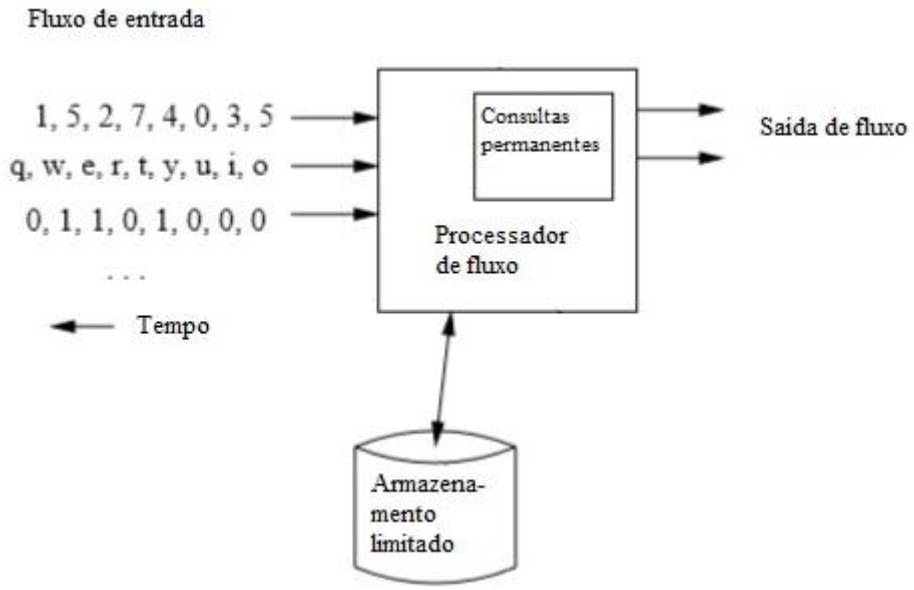


Figura 3.2 - Esquema simplicado de um Sistema Gerenciador de Fluxo de Dados - adaptado de Rajaraman e Ullman (2012).

O número de fluxos de entrada não é conhecido, ou seja, podem chegar em intervalos de tempo e formatos diferentes. O fato de a taxa de chegada dos fluxos de dados não estar sob o controle do sistema é o que distingue o processamento de fluxos de dados do trabalho realizado dentro de um SGBD tradicional. O SGBD tem à sua disposição os dados em um disco e não corre o risco de perdê-los enquanto executa uma consulta. O armazenamento limitado, que pode ser tanto em memória

principal quanto secundária, pode guardar partes de fluxos para responder algumas consultas. Porém, dificilmente terá capacidade para armazenar todos os fluxos de entrada. O armazenamento limitado pode ser usado como suporte, mas não como ferramenta de armazenamento como as que são utilizadas na Mineração de Dados.

3.4 Técnica de Janelamento da Mineração de Dados

Janelamento é uma técnica bastante conhecida e discutida em MD (Ahmed, Tanbeer e Jeong, 2009; Niranjan et al., 2011). O janelamento pode ser feito de diferentes formas nas abordagens descritas na literatura, porém visa, em geral, selecionar um conjunto de dados específicos para a aplicação dos algoritmos de extração de padrões. O caso a ser discutido neste texto relaciona a técnica de janelamento com algoritmos que realizam a mineração em fluxos de dados no *Big Data*.

Por mais que quantidade e variedade tenham sido sempre um desafio para o gerenciamento de dados, é verdade também que a velocidade de entrada dos dados se tornou o maior desafio atualmente (Ölmezogullari e Ari, 2013). Tentar armazenar os dados para depois analisar traz custo extra e ainda pode ser inútil para domínios em que o tempo de análise é fundamental para decisões de momento, como no mercado financeiro.

Por esse motivo, tem-se buscado novas alternativas para adaptar ideias e ferramentas já existentes ao modelo veloz atual. Uma das alternaticas é a utilização de janelamento nos dados, embutido em diferentes algoritmos, para se priorizar, isolar e focar o desempenho da tarefa de mineração de dados na parte correta e melhor aproveitável dos dados de entrada. Portanto, o janelamento delimita os dados mais relevantes para a aplicação (Xu, Chen e Bie, 2009), evitando, assim, perda de tempo ou de desempenho com dados desnecessários para a tarefa solicitada.

Nesse contexto, os dois tipos de janelamento mais utilizados são *Sliding Window* e *Tumbling Window*.

3.4.1 *Sliding Window* (Janela Deslizante)

Os algoritmos que implementam Janela Deslizante podem adotar as seguintes estratégias:

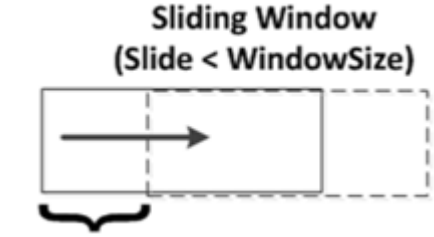
- por tempo: Trabalham com os dados mais recentes e vai excluindo os antigos quando seu tempo de entrada na janela estoura. Com o "deslizamento" da janela durante o tempo, é possível que dois eventos de diferentes momentos se encontrem na janela.
- por evento: Trabalham com a quantidade de eventos presentes na janela. Armazenam os últimos X eventos e quando a capacidade está prestes a exceder o valor X, descartam os antigos para receber os mais recentes.

3.4.2 Tumbling Window

Tumbling Window, por outro lado, não tem sobreposição de eventos de períodos diferentes, pois move a janela em quantidade igual ao seu tamanho anterior.

Na Figura 3.3 é mostrada a diferença entre as duas estratégias de janelamento. Na técnica de *Tumbling Window* não há sobreposição de dados entre as janelas, enquanto na Janela Deslizante é possível um dado estar na janela atual e também na próxima.

Sliding Window
(Deslizamento < Tamanho da Janela)



Deslizamento

Tumbling Window
(Deslizamento = Tamanho da Janela)

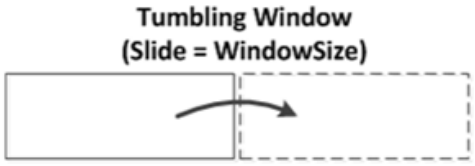


Figura 3.3 - Diferenças entre *Tumbling Window* e *Sliding Window* - adaptada de Ölmezoğulları e Ari (2013).

A proposta de Kahn et al. (2009) trabalha com *Emerging and Jumping Pattern* (JEP). Um *Emerging Pattern* (EP) é definido como um conjunto de itens que o suporte aumenta com o tempo de acordo com a taxa de chegada de novos dados. Um *Jumping Pattern* é um conjunto de itens que o suporte muda muito mais rapidamente que o EP. O algoritmo proposto, *Dual Support Apriori for Temporal Data*, extrai padrões sequencias de fluxo de dados utilizando o conceito de janela deslizante e explorando dados temporais já minerados previamente, por isso, requer menos memória para execução.

Também em aplicações de fluxo de dados, a proposta de Chen et al. (2012) faz uso de janelas deslizantes de tamanho variável para minerar padrões frequentes de forma eficiente. Para destacar os padrões frequentes no fluxo de dados, é utilizado um modelo de decaimento de tempo para diferenciar os padrões que recentemente geraram transações dos mais antigos. Para tornar a mineração eficiente, a árvore SWP proposta é podada periodicamente identificando padrões insignificantes. Por causa dessa árvore e do decaimento de tempo, é possível atingir tanto 100% de precisão ou de revocação.

3.5 Algoritmo Incremental Miner of Stretchy Time Sequences (IncMSTS)

3.5.1 – Descrição do Trabalho

O trabalho de Silveira Junior, Ribeiro e Santos (2013) traz três itens diferenciados e que foram amplamente utilizados neste projeto de mestrado.

Primeiro, o algoritmo IncMSTS faz uso da Mineração de Dados Incremental, permitindo o incremento de novos dados ao invés de considerar todo dado de entrada como uma nova base de dados.

Segundo, traz um tipo de janelamento alternativo para manipular dados com ruído, característica frequente em dados coletados por sensores de ambiente. Essa abordagem é mais apropriada para capturar a evolução de fenômenos naturais e de

desenvolvimento de plantações. Os erros podem ser resolvidos se forem considerados como intervalos de tempo, assim o algoritmo resolve os erros ignorando os momentos em que acontecem. Esse janelamento alternativo presente no IncMSTS não deve ser confundido com o janelamento de dados de entrada realizado pelo Esper.

Terceiro, o algoritmo permite a identificação de itens semi frequentes da base de dados, aumentando a possibilidade de se gerar padrões a partir de itens que não foram considerados frequentes ainda, mas ficaram perto dessa situação.

O IncMSTS é baseado no algoritmo GSP (Srikant e Agrawal, 1996). O algoritmo descarta as sequências candidatas que não são frequentes, pois estas nunca poderão gerar sequências frequentes; como é provado pela propriedade anti–monotônica. A propriedade anti–monotônica garante que a partir de um itemset ou padrão não frequente é impossível gerar um padrão frequente (Han, Pei e Yan, 2005). O número de iterações do algoritmo é igual ao tamanho da maior sequência encontrada.

3.5.2 – Algoritmo Incremental

A Mineração de Dados Incremental (MDI) visa a manutenção dos padrões frequentes já previamente encontrados, mediante o incremento de novos dados (Mabroukeh e Ezeife, 2010; Niranjan et al., 2011). Em outras palavras, a MDI objetiva manter a consistência dos padrões frente à evolução da base, evitando reprocessamentos (Silveira Junior, Ribeiro e Santos, 2013).

Na Figura 3.4 é mostrado o esquema de uma base incremental sofrendo várias adições de conteúdo e, por isso, sempre reprocessada para atualizar o conhecimento extraído.

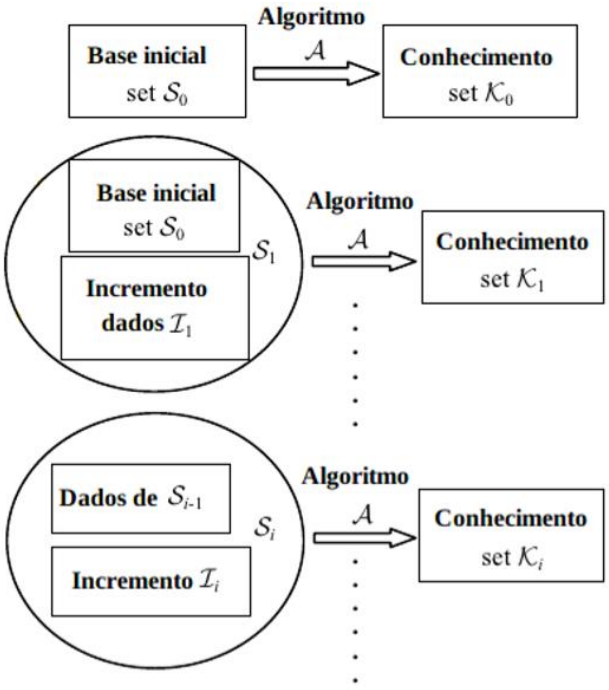


Figura 3.4 - Modelo de uma base de dados incremental em diversas etapas, adaptada de (Jiancong, Yongquan e Jiuhong, 2009).

Note que o Conhecimento extraído das bases é sempre apenas um elemento, atualizado a partir do Conhecimento anterior mais o Conhecimento adquirido no último incremento.

3.5.3 – Janelamento do IncMSTS

O algoritmo de janelamento do IncMSTS, chamado *Stretchy Time Windows* (STW), permite a ocorrência de padrões esparsos e calcula o número de tuplas que podem ser checadas para encontrar um novo item. Assim, por mais que existam elementos indesejados entre itens procurados na base de dados, o algoritmo é capaz de ignorá-los (considera que são intervalos de tempo) e encontra o item desejado caso tenha suporte e esteja dentro da janela de busca máxima estipulada pelo usuário. Esse método tem importante papel na proposta deste trabalho e o algoritmo de janelamento é mostrado na Figura 3.5.

Para compreensão correta de como foram feitos os testes deste trabalho, é fundamental entender que o janelamento do IncMSTS, interno ao algoritmo, nada tem a ver com o janelamento do Esper explicado anteriormente. O janelamento do

Esper separa e gera um lote de dados vindos do fluxo de dados para entregar como entrada ao IncMSTS. A partir daí o IncMSTS pode fazer uso do seu janelamento para procurar por padrões esparsos. Portanto, são dois janelamentos distintos em dois pontos distintos do trabalho.

```
   Input: pattern p, itemset ι, μ
   Output: count /* Number of occurrence of pattern p          */
1  function checkingOccurrence
2  count ← 0 ;
3  p′ ← concat(p, ι) ;
4  foreach occurrence occ of pattern p do
       /* Calculating the number of tuples where ι can be found
          to consider the occurrence.                           */
5      numberOfTimeGaps ← (timeOfLastElement(occ) −
       timeOfFirstElement(occ) + 1) − sizeOf(p) ;
6      window ←
       MIN(μ − numberOfTimeGaps, Δ(nextOcurrence(p)));
       /* Searching for ι in calculated tuples.               */
7      for k ← 1 to window do
8          if happen(ι, tupleAtTime(timeOfLastElement(occ) + k))
           then
               /* ι found.                                     */
9              count ← count + 1 ;
10             break ;
```

Figura 3.5 - Algoritmo Stretchy Time Windows (Silveira Junior, Ribeiro e Santos, 2013)

O método STW usa o parâmetro μ para dizer quão longo o maior intervalo de tempo pode ser.

De acordo com os autores, o funcionamento é o seguinte: sempre que um padrão *p'* =< i1...in in+1 > é gerado baseado em um padrão anterior *p* =< $i_1...i_n$ > (linha 3), a existência de *p*' é checada nas linhas de 4 a 10. Perceba que *p* é parte de *p'*.

O modo de checagem é: para todas as ocorrências de *p*, o algoritmo verifica quão esparsa é a ocorrência (linha 5) e calcula o número de tuplas que podem ser checadas para encontrar um novo item. A janela de busca deve ser o valor mínimo entre (*μ - número de intevalos de tempo*) e o Δ(*próxima ocorrência de p*, que é a distância entre o último item da ocorrência analisada do padrão *p* e o último item da próxima ocorrência de *p*), pois a janela de busca não pode ser grande o suficiente para cobrir as próximas ocorrências de *p* (linha 6).

O algoritmo mantém um vetor com todos os momentos (*timestamps*) onde *p* ocorreu e um vetor para guardar todos os itens frequentes. Portanto, para encontrar o valor do contador de *p'*, é necessário calcular quantas vezes i_{n+1} (itens frequentes concatenados em p para gerar *p'*) aconteceram após *p* (linhas de 7 a 10).

Para exemplificar, Silveira Junior, Ribeiro e Santos (2013) mostram uma situação hipotética onde existem dois padrões não frequentes s_1 =< *a b c* > e s_2 =< *a d c*>. O padrão < *a c* > é frequente embora os eventos não aconteçam um imediatamente após o outro. O MSTS encontra *s* = < *a c* > se o valor máximo da janela de busca (*μ*) consegue cobrir as ocorrências de *c* em quantidade suficiente para considerar *s* frequente.

Os testes deste projeto de mestrado abordam o impacto do janelamento do IncMSTS nos resultados.

3.5.4 – Semi frequentes

O algoritmo IncMSTS armazena os padrões semi frequentes assim como o algoritmo IncSpan (Cheng, Yan e Han, 2004).

Um padrão é dito ser semi frequente caso seu valor de suporte esteja entre suporte mínimo e suporte mínimo × δ. Assim, o parâmetro δ é utilizado para ajustar o quão próximo do suporte mínimo um padrão deve ser para considerá-lo semi frequente.

A escolha do valor de δ influencia da seguinte forma: um valor de δ pequeno faz com que mais padrões sejam armazenados como semi frequente. Isso gera uma queda de desempenho do algoritmo e maior uso de memória.

Por outro lado, a escolha de um δ grande faz com que poucos padrões sejam armazenados como semi frequentes. Isso pode acarretar uma queda de precisão do algoritmo.

Os testes deste projeto de mestrado abordam o impacto dos itens semi frequentes nos resultados do trabalho.

3.6 Considerações Finais

A mineração de dados tem papel indiscutível no processo de descoberta de conhecimento útil e tem sido foco de várias pesquisas por alguns anos.

O janelamento fornece uma importante alternativa de utilização no *Big Data* para os algoritmos tradicionais de mineração de dados que realizam a extração de regras de associação e padrões sequencias, pois permite que esses algoritmos sejam aplicados de forma eficiente mesmo em mineração de fluxo de dados, que vai ao encontro do comportamento usual dos algoritmos que trabalham armazenando e analisando seus dados *offline*.

O algoritmo *Incremental Miner of Stretchy Time Sequences* realiza a clássica mineração de padrões sequenciais utilizando técnicas de janelamento para manipular dados esparsos e com ruídos. Dada sua importância e possibilidade de adaptação, o IncMSTS foi explicado em detalhes e sua participação no projeto será mostrada no capítulo 5.

Nos capítulos a seguir são mostradas a união entre Esper e IncMSTS, criando a abordagem Esper + IncMSTS e os testes e resultados alcançados no desenvolvimento do projeto.

Capítulo 4

Esper + IncMSTS

Neste capítulo é apresentada a abordagem Esper + Incremental Miner of Stretchy Time Sequences. Essa abordagem foi proposta com a finalidade de permitir ao algoritmo de extração de padrões sequenciais estático funcionar em ambiente de dados em tempo real, apoiado pelo janelamento contido no Esper. Este capítulo encontra-se dividido da seguinte forma: na Seção 4.1 são apresentadas as considerações iniciais. Na Seção 4.2 é apresentada a arquitetura que une e permite o funcionamento das diferentes partes do projeto. Na Seção 4.3 são apresentados os trabalhos relacionados. Por fim, na Seção 4.4, o capítulo é finalizado com as considerações finais.

4.1 Considerações Iniciais

As características do algoritmo IncMSTS são mantidas para este trabalho. Suas capacidades diferenciadas na tarefa de extração de padrões sequenciais continuam válidas para a sua adaptação ao ambiente de fluxo de dados. Essas capacidades são (i) algoritmo incremental, (ii) a mineração de dados esparsos e (iii) os itens semi frequentes.

O IncMSTS, assim como o Esper, são partes fundamentais deste trabalho. Porém, não é discutido aqui sobre a eficácia e eficiência do IncMSTS, que já foi comprovada por seu autor. O foco é nas maiores contribuições deste trabalho, que são a produção do código do Esper, os diversos testes realizados, as tendências e padrões alcançados e sua aplicação no ambiente de fluxo de dados.

4.2 Arquitetura

Nas Figuras 4.1, 4.2 e 4.3 são mostradas a arquitetura da abordagem proposta, representada pelo diagrama *Structured Analysis and Design Technique* (SADT) (Ross, 1977). De acordo com a notação sugerida pelo diagrama, os retângulos

representam cada uma das atividades relevantes ao desenvolvimento da abordagem e as setas possuem significados distintos, dependendo da sua posição.

As setas que ocorrem do lado esquerdo de cada retângulo representam os dados de entrada da atividade. As setas que partem de cada retângulo, à direita, se referem às saídas da atividade. Setas incidentes na parte superior dos retângulos correspondem aos controles (linguagens e técnicas) e setas na parte de baixo representam as ferramentas utilizadas em determinada atividade.

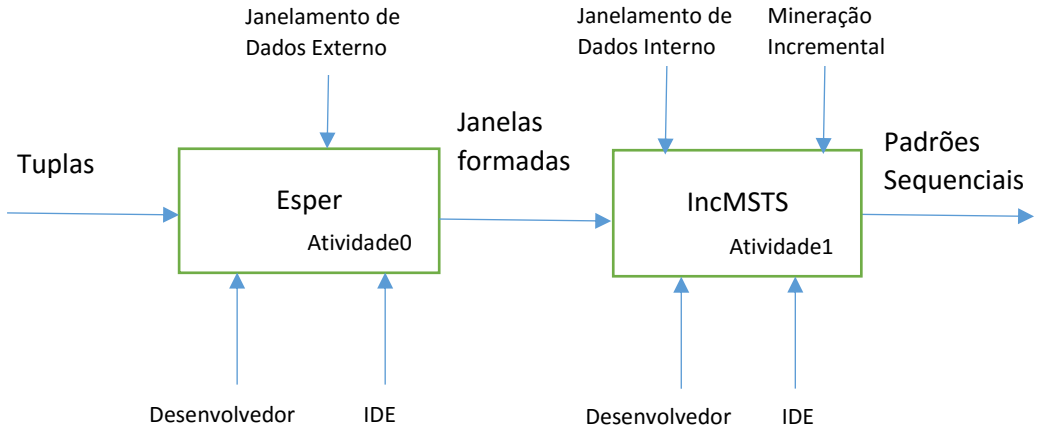


Figura 4.1 – Arquitetura geral da abordagem proposta utilizando o diagrama SADT.

Para facilitar a compreensão das diferentes atividades fundamentais deste projeto, a seguir são mostradas as duas atividades em maior nível de detalhe, nas Figuras 4.2 e 4.3, ainda seguindo a ideia do diagrama SADT.

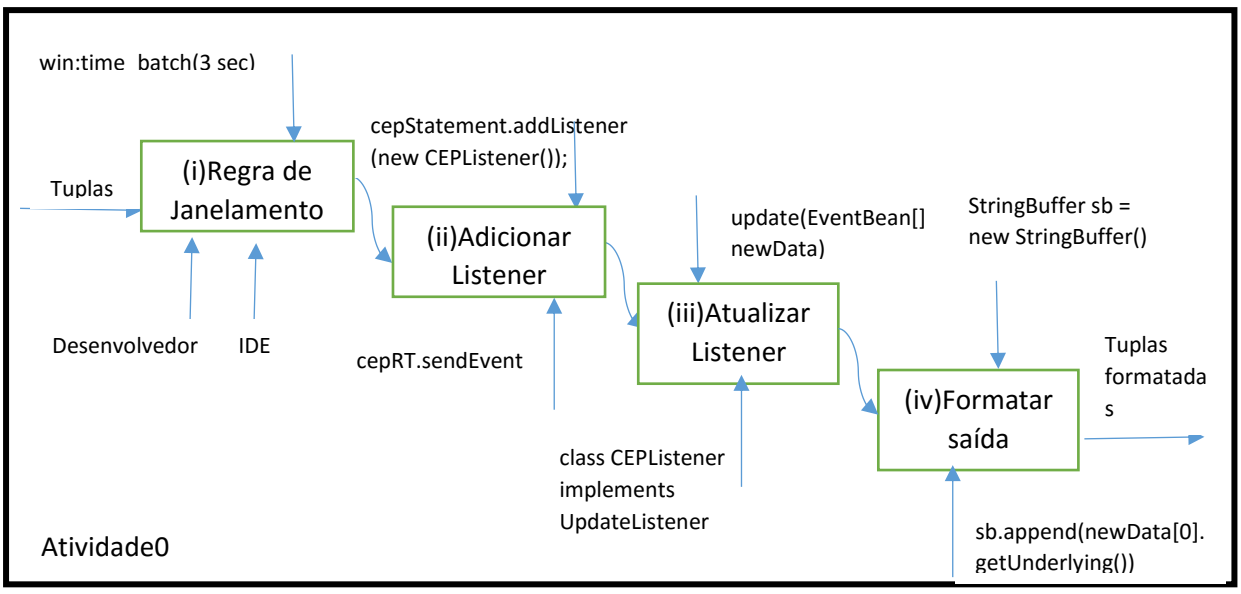


Figura 4.2 – Atividade0, por dentro da utilização do mecanismo Esper.

A Atividade0 engloba exclusivamente os procedimentos realizados pelo mecanismo Esper, da chegada dos dados, janelamento, atualização e preparação das tuplas para se tornarem a entrada do algoritmo IncMSTS.

Conforme é mostrado na Figura 4.2, as tuplas chegam de forma ininterrupta ao Esper e o janelamento é necessário para entregar pacotes de dados ao IncMSTS, pois o algoritmo não trabalha com dados que chegam em fluxo contínuo.

Conforme visto no Capítulo 3, há diferentes tipos de janelamento possíveis no Esper, na Figura 4.2 é mostrado o tipo de janelamento por lotes de 3 em 3 segundos em (i). Ou seja, todas as tuplas que chegaram nos últimos 3 segundos serão agrupadas em um lote e enviadas para o algoritmo IncMSTS.

Os passos para isso acontecer, seguindo a Figura 4.2, passam pelo janelamento por lotes de tempo (i), já explicado. Após é realizada a adição do *Listener* (ii), que é o processo ouvinte que ficará atento a todas as tuplas que satisfazem as condições de janelamento e poderão compor os dados formatados de saída. Depois é feita a atualização do *Listener* (iii), passo necessário para identificar quais são os novos itens e quais se tornaram antigos. Por fim, o passo (iv) deixa as tuplas do janelamento atual formatadas de modo que o algoritmo IncMSTS possa receber como entrada e realizar a mineração de dados.

Uma observação importante é que, no caso deste trabalho, o passo (ii) anterior sobre o *Listener* não faz nenhum tipo de filtragem nos dados, a fim de eliminar possíveis ruídos. Este trabalho não foi feito nesta fase pois o algoritmo IncMSTS tem seu próprio tratamento de dados esparsos como forma de aliviar os ruídos vindos da entrada de dados.

Outra observação é que no item (iii), o Esper é capaz de fazer uma filtragem de forma a eliminar possíveis dados repetidos antigos para o novo pacote de dados. Essa opção também não foi utilizada, já que o IncMSTS trata sequências semi frequentes também. Dessa forma, a eliminação de tuplas ditas “antigas” ou repetidas poderia impactar no cálculo dos semi frequentes.

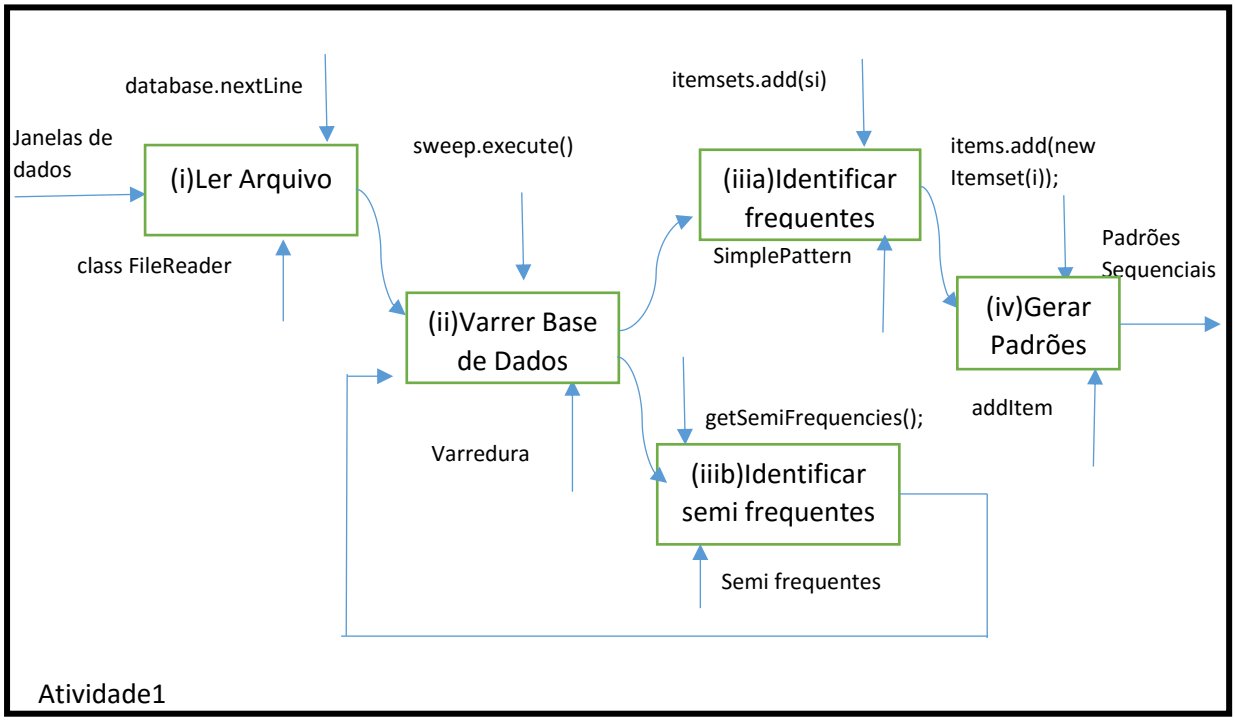


Figura 4.3 – Atividade1, por dentro do algoritmo IncMSTS

O algoritmo IncMSTS faz a mineração de dados sequencial inclusive de dados esparsos e de forma incremental. A intenção deste trabalho é provar a hipótese de que o algoritmo, inicialmente construído e testado com dados estáticos, pode ser uma alternativa satisfatória também em ambiente *Big Data*, com fluxo de dados contínuo. Dito isso, nenhuma modificação foi feita no código original do IncMSTS para ajudá-lo a trabalhar com fluxo de dados.

Na figura 4.3 são mostrados os passos de execução do algoritmo. O primeiro passo é a leitura do arquivo de entrada linha por linha. A seguir é feita a varredura (ii), que busca os itens frequentes que geram padrões (iiia), sendo esse o caminho padrão do algoritmo de mineração de dados sequenciais IncMSTS. O passo (iiib) adiciona a identificação de itens semi frequentes, que no próximo incremento, podem se tornar frequentes.

Assim, itens que não são tão frequentes quanto os definidos inicialmente como frequentes, mas que se aproximam desse patamar, ficam reservados para o próximo incremento que pode fazer dos itens não tão frequentes, agora, itens frequentes.

Neste trabalho, o papel do IncMSTS é buscar um arquivo de entrada de tempos em tempos, em repetição infinita, e dar como saída os padrões corretos em tempo de assegurar que o próximo arquivo de entrada seja lido imediatamente após

ser criado, a fim de evitar acúmulo de entradas e atraso na identificação dos padrões, que não é permitida na mineração de dados em fluxo de dados. Esses detalhes são discutidos no Capítulo 5.

4.3 Trabalhos Relacionados a abordagem Esper + IncMSTS

Os Trabalhos Relacionados nesta Seção são equivalentes no sentido de realizarem a extração de padrões sequenciais em fluxo de dados, assim como a abordagem Esper + IncMSTS, mas não necessariamente utilizam as mesmas ideias e ferramentas.

O trabalho de (Choi, Kim e Hwang, 2014) traz uma abordagem para encontrar padrões sequenciais em fluxo de dados baseada em aplicar pesos dinâmicos. O método reduz o número de padrões candidatos ao utilizar peso dinâmico de acordo com a sequência em um tempo determinado. A proposta também reduz o uso de memória e tempo de processamento. Sem considerar os pesos dos eventos em determinados momentos em que possam ser de importância crítica, os métodos existentes de mineração de dados calculam sua frequência, mas isso pode não ser uma técnica de mineração eficiente. No caso do peso dinâmico, um evento ou item que pode ter diferentes pesos durante o tempo pode se beneficiar do método. Isso significa que os usuários podem ter a informação que procuram rapidamente e pode ser aplicado em tarefas do mundo real. O trabalho de (Choi et al., 2014) pode ser um possível trabalho futuro ao se associar ao IncMSTS, principalmente ao reduzir o tempo de processamento por meio dos pesos dinâmicos.

O trabalho de (Kaushal e Singh, 2015) mostra uma comparação entre algoritmos de mineração de padrões sequenciais em fluxo de cliques em páginas web. Por mais que não seja trabalho relacionado a este em relação ao desenvolvimento, se assemelha ao utilizar dados semelhantes, pois este trabalho utiliza a sequência de cliques nas páginas do site da FIFA durante a Copa do Mundo de 1998. Os autores testaram os algoritmos SPADE, SPAM, LAPIN, CM_SPADE e CM_SPAM, que são considerados dos melhores dentre os que realizam a mineração de padrões sequenciais. Os testes mostraram que o SPADE consome menos tempo e o SPAM e CM_SPAM consomem menos memória. A comparação

de Kaushal e Singh (2015) mostra que os dados vindos de sequências de cliques de usuários em páginas web são importantes para se avaliar o comportamento das pessoas que acessam o site e, por isso, a mesma ideia foi utilizada neste trabalho. Ainda, os algoritmos citados pelos autores podem ser adaptados para a abordagem deste projeto e se utilizar do janelamento do Esper.

A proposta de (Zihayat et al., 2015) define o problema de minerar padrões sequenciais de alta utilidade (HUSP) por meio de fluxo de dados de alta velocidade e propõe um algoritmo eficiente para essa mineração. Em mineração HUSP, cada item tem um peso (preço, lucro) e pode aparecer mais que uma vez em alguma transação (quantidade de compra) e o objetivo é encontrar sequências que a utilidade na base de dados não seja menor que um valor mínimo definido pelo usuário. Os autores propõem o algoritmo HUSP-Stream que poda o espaço de procura para gerar um modelo eficiente. Os resultados mostraram que a abordagem melhorou a performance em relação ao estado da arte no número de potenciais HUSPs gerados, tempo de execução e uso de memória.

O trabalho chamado *Online Association Rule Mining over Fast Data*, de Ölmezoğulları e Ari (2013) é o mais correlato em relação a este projeto. A ideia é utilizar algoritmos tradicionais que realizam mineração de regras de associação, como o Apriori e FP-Growth, e adaptá-los para a realização de mineração online sobre fluxo de dados contínuo de alta velocidade. A estratégia inclui nos algoritmos citados uma estratégia de janelamento. Assim, os autores conseguiram gerenciar a quantidade de dados no tamanho da janela e podem repetir o processo de mineração para a próxima janela.

Durante o trabalho, os autores detectaram ser impossível utilizar o Apriori quando a quantidade de dados aumentava, chegando a ser 75 vezes mais lento que a implementação usada do FP-Growth. Isso se dá por questões de estratégias e o modo de funcionamento de cada algoritmo.

Ainda na escolha dos algoritmos, uma versão atualizada do FP-Growth foi comparada com a original e também mostrou resultados superiores em relação ao tempo de execução.

Os dados coletados da LastFM passaram por um pré-processamento de forma a simplificar as tuplas, foram mantidos apenas nome da música e do cantor como

atributos. Além disso, o pré-processamento também retirou tuplas em que as informações não estavam completas. Assim, o trabalho se resumiu a tuplas simples que continham, basicamente, nome da música e cantor. Esse fato é um facilitador em relação a mineração. Domínios que permitem essa simplificação tendem a facilitar o trabalho dos algoritmos de mineração. Para efeito de comparação, os dados de acesso do site da FIFA utilizado na abordagem Esper + IncMSTS tinham tuplas com tamanho médio de 34,74 itens.

Os resultados dos testes mostraram que o FP-Growth consegue ótimo desempenho em termos de extração de regras e tempo de processamento.

Tabela 4.1: Comparação do trabalho relacionado com a abordagem proposta.

	Ölmezoğulları e Ari (2013)	Esper + IncMSTS
Extração de Padrões Sequenciais		X
Extração de Regras de Associação	X	
Técnicas de Janelamento Esper para fluxo de dados	X	X
Pré processamento dos dados	X	
Trata dados com ruídos		X
Mineração de Dados Incremental		X

Na tabela 4.1 são mostradas as semelhanças e diferenças entre o trabalho relacionado e a abordagem proposta Esper + IncMSTS.

As duas primeiras linhas tratam das estratégias de mineração de dados, onde o primeiro realiza a extração de regras de associação e o segundo a extração de padrões sequenciais. O próximo ponto é o que une os dois trabalhos: a utilização do Esper para janelar o fluxo de dados. O janelamento permite que algoritmos voltados à mineração de dados estática possam ser aproveitados também em ambiente de fluxo de dados.

Por fim, Esper + IncMSTS não realiza nenhum pré processamento para melhorar as condições das tuplas, o algoritmo IncMSTS, trata dados com ruídos, ou seja, busca padrões esparsos utilizando janelamento interno e realiza mineração de dados incremental, que é um ponto importante para se melhorar a eficiência de algoritmos complexos em ambiente de fluxo de dados.

4.4 Considerações Finais

Neste capítulo foram apresentados a arquitetura geral e específica do projeto desenvolvido, a proposta de validação e o trabalho relacionado. O relacionamento entre as partes (Esper e IncMSTS) e a avaliação do seu desempenho em ambiente *Big Data* são os motivos deste trabalho. O capítulo a seguir mostra os testes e conclusões alcançadas a partir do estudo e execução dos testes.

Capítulo 5
Testes e Resultados

Neste capítulo são apresentados os testes realizados e os resultados alcançados por este projeto de mestrado. Mas, antes disso, também é apresentado um resumo do que foi visto até aqui e um problema que pode ocorrer durante a execução do Esper + IncMSTS. Este capítulo encontra-se dividido da seguinte forma: na Seção 5.1 são apresentadas as considerações iniciais. Na Seção 5.2 é apresentada a estratégia de teste, que resume os dois tipos de leitura utilizadas nos testes. Na Seção 5.3 são apresentadas condições gerais para os testes e na seção 5.4 os testes de forma detalhada. Na Seção 5.5 são mostrados os resultados alcançados por este projeto e, por fim, na Seção 5.6, o capítulo é finalizado com as considerações finais.

5.1 Considerações Iniciais

Conforme dito no Capítulo 1, este trabalho levanta a hipótese de utilização de um algoritmo de mineração de dados sequencial para o ambiente *Big Data*.

Para comprovar a aplicação com sucesso, ou não, da hipótese, diversos testes foram realizados, divididos em alterações no incremento, nos itens semi frequentes, no suporte e no tamanho da janela do IncMSTS. Todos os testes são comentados individualmente e, no final, uma tabela comparativa é mostrada.

É importante lembrar que os testes de eficácia e eficiência para a operação normal do algoritmo IncMSTS já foram exibidos no trabalho original. Portanto, aqui é tratado apenas o que puder afetar o rendimento do algoritmo para as respostas em tempo real.

5.2 Estratégias de Teste

Existem duas estratégias de testes presentes neste trabalho, uma de leitura total e outra de leitura parcial. As próximas seções descrevem as duas estratégias, com suas características, vantagens e desvantagens.

5.2.1 Leitura Total

Características:

A Leitura Total sugere o comportamento teoricamente perfeito da execução de um algoritmo de mineração de dados sequencial em ambiente *Big Data*, ou seja, independentemente das características da chegada dos dados (velocidade e tamanho dos incrementos), o algoritmo IncMSTS passa por todos os incrementos e retorna seus resultados. É a transição direta do algoritmo desenvolvido para ambiente estático para o ambiente *Big Data*.

Vantagens:

A vantagem dessa estratégia é a garantia de que o algoritmo retorna todos os padrões identificados nos dados e seus incrementos. A saída do algoritmo, nesse caso, é a mesma caso os dados de um fluxo de dados fossem “transformados” em uma base estática e seus incrementos. Assim, há a garantia de que o IncMSTS retornou tudo que é possível em relação aos padrões identificados nos dados.

Desvantagens:

Essa estratégia possui duas desvantagens. A primeira, e mais importante, é clara e explicada a seguir: o tempo.

- Esper
    - Esper recebe os dados de forma ininterrupta e de tempos em tempos fecha uma janela de dados (*batch*), a formata corretamente e a envia a um local determinado para salvar o arquivo.
    - O Esper não para de receber os dados em nenhum momento e faz as janelas de dados externa sempre no mesmo tempo ou no mesmo tamanho, dependendo de como foi configurado. No caso do tempo, assume-se que o tempo de formação da janela do Esper seja *tWindEsper*.
- IncMSTS

- IncMSTS aguarda em repetição infinita no local em que o Esper jogará seus arquivos. O IncMSTS sempre executará as janelas de dados externas do Esper tão logo cheguem, a não ser que ocorra a demora excessiva na execução da janela anterior pelo IncMSTS. Assume-se o tempo de execução do IncMSTS como *tIncMSTS*,

- Esper + IncMSTS
  - Se *tIncMSTS* > *tWindEsper*, então há atraso na execução das janelas de dados externa pelo IncMSTS, que pode gerar acúmulo.
  - Nesse caso é necessário notar o comportamento do Esper + IncMSTS na chegada das janelas seguintes em relação ao tempo, pois caso as próximas janelas tenham comportamento parecido a anterior, a tendência é o acúmulo de arquivos gerados pelo Esper que não estão sendo lidos de imediato pelo IncMSTS, o que descaracteriza a mineração de dados em tempo real. Observe a tabela 5.1 a seguir com um exemplo de acúmulo.

Tabela 5.1 – Exemplo de execução e de formação de acúmulo.

Saída do Esper	Tempo Decorrido	Entrada do IncMSTS	Tempo de Execução do IncMSTS
1ª Janela de Dados	5 segundos	Aos 5 segundos - imediata	8 segundos
2ª Janela de Dados	10 segundos	13 segundos – depois de 3 segundos de disponibilidade da janela de dados vinda do Esper.	8 segundos
3ª Janela de Dados	15 segundos	21 segundos – Acúmulo, já existem 2 saídas do Esper aguardando.	8 segundos
4ª Janela de Dados	20 segundos	29 segundos – Acúmulo, já existem 2 saídas do Esper aguardando.	8 segundos
5ª Janela de Dados	25 segundos	37 segundos – Acúmulo, já existem 3 saídas do Esper aguardando.	8 segundos
...	...	...	...

Na tabela é mostrada a importância de se analisar o comportamento dos dados e a possível presença de acúmulo. Se for uma ocorrência aleatória não acarreta problemas, porém, caso se torne uma tendência, a dupla Esper + IncMSTS não cumpre a função estipulada neste trabalho, que é a mineração de dados em tempo real.

O segundo problema é a restrição excessiva nos parâmetros do IncMSTS normalmente necessária para diminuir o tempo de execução. Caso o usuário escolha a estratégia de Leitura Total, provavelmente terá que diminuir o tamanho da janela e a quantidade de semi frequentes. Se isso for feito ao extremo, o IncMSTS não terá o reconhecimento de padrões esparsos (tamanho de janela igual a 1) e não irá armazenar os itens semi frequentes (delta igual a 1). Porém, uma execução do IncMSTS com essas características nada mais é que um GSP incremental, fugindo dos diferenciais implementados pelo IncMSTS.

5.2.2 Leitura Parcial

A Leitura Parcial foi a estratégia utilizada prioritariamente nos testes deste trabalho de mestrado. Essa estratégia agrega tanto as características do ambiente *Big Data* como os diferenciais do algoritmo IncMSTS.

Características:

Na estratégia de Leitura Parcial, os dados do fluxo de dados janelados pelo Esper continuam vindo de forma ininterrupta e sendo divididos em lotes de arquivos de tempos em tempos. Porém, a diferença fundamental está no que é utilizado de entrada para o IncMSTS, ou seja, o que o IncMSTS utiliza como dados de entrada.

Nesse caso, o IncMSTS toma como entrada sempre o último arquivo modificado presente na pasta local, ou seja, o acúmulo causado pela estratégia anterior é ignorado. Nesse momento pode-se imaginar que a perda de alguns incrementos fosse decisiva para o fracasso da extração de padrões, mas não é o que ocorre, fato corroborado pelos testes realizados.

Vantagens:

A principal vantagem é a manutenção das características que tornam o IncMSTS um algoritmo de extração de padrões sequenciais diferenciado. Por mais que o tempo de execução ainda seja um item de fundamental importância, não há a necessidade de se restringir o IncMSTS ao ponto de que se torne um GSP incremental para evitar o acúmulo. O acúmulo continua ocorrendo, mas agora é desconsiderado para não atrasar a execução do IncMSTS.

A outra vantagem é a possibilidade de se ajustar e encontrar uma sintonia fina entre todos os parâmetros do IncMSTS, como suporte, janelamento e semi frequentes, de forma a encontrar a melhor configuração para os dados do domínio.

Por fim, os testes mostraram que, por mais que agora o IncMSTS tenha como entrada menos lotes de dados, que podem chegar a apenas 33% dos lotes, os padrões sequenciais mostrados na saída tem pouca diferença em relação a execução nos dados completos.

Desvantagens

Embora a estratégia de Leitura Parcial tenha se mostrado uma boa alternativa para a execução do Esper + IncMSTS, pode acontecer que alguns padrões fiquem de fora da saída do algoritmo, principalmente em fluxo de dados que tenham algumas tendências pontuais. Por exemplo, é possível que uma tendência importante seja ignorada caso apareça justamente nos incrementos ignorados pelo Esper + IncMSTS. Não é o comportamento padrão, mas pode acontecer dependendo das características do fluxo de dados.

5.3 Condições gerais

Os dados utilizados para os testes são compostos por sequências de cliques nas páginas do site da FIFA durante a Copa do Mundo de 1998. O conjunto de dados possui 20450 sequências, tem 2990 itens distintos e a média de tamanho de cada sequência é de 34,74 itens.

Em relação aos algoritmos utilizados, vale destacar a presença do algoritmo GSP na execução dos testes. O GSP é importante por mostrar o impacto causado pela inserção dos dados com características do *Big Data* na abordagem Esper + IncMSTS. Quando o teste engloba a execução do GSP, isso foi deixado claro.

Neste ponto é necessário explicar uma das restrições deste trabalho. O algoritmo IncMSTS exige a montagem de um arquivo com as sequências que utiliza como incremento. Então, por mais que nenhum dado seja perdido ao se montar um arquivo de incremento, um certo tempo é perdido para realizar essa montagem. Portanto, cada incremento é entregue pelo Esper ao IncMSTS a cada três segundos, com 2000 sequências em cada um, em média. Cada sequência do incremento tem tamanho médio de 34,74 itens.

Os dados foram divididos em 9 incrementos, de teste0.txt até teste8.txt. Então foram 27 segundos para a geração desses incrementos.

5.4 Execução dos testes

Os testes são divididos prioritariamente entre a presença ou não de incrementos. Dentro dessa divisão, há a distinção entre as abordagens: GSP, IncMSTS e Esper + IncMSTS. Sempre são destacadas as diferenças ocorridas entre as abordagens, facilitando, assim, a compreensão e desenvolvimento lógico dos testes.

Na Seção seguinte há uma tabela resumindo os testes descritos em detalhes.

5.4.1 Sem incremento

Os dados tem comportamento semelhante a uma base de dados estática nesse caso de teste. As sequências oriundas do acesso às páginas da FIFA foram agrupadas em um único arquivo e passadas pelos algoritmos GSP e IncMSTS. Para o GSP esse é o procedimento natural, já que não faz mineração de dados incremental. Para o IncMSTS, é a execução ignorando sua característica

incremental. Além disso, o janelamento e os semi frequentes do IncMSTS também foram ignorados.

Os testes para o caso explicado servem principalmente para nortear como devem se comportar os próximos testes e comprovar que o IncMSTS, se tratado com o máximo de simplicidade, corresponde ao GSP.

Ainda, é necessário observar que o suporte utilizado nesse primeiro teste foi 0.02. Lembrando a fórmula para cálculo do suporte mínimo:

suporte = (número de ocorrências da sequência) / (total de sequências da base)

O valor de 0.02 é baixo se comparado com os testes seguintes. A explicação para a diferença de valor de suporte é a quantidade de sequências presentes nesse caso. Nos casos seguintes, o suporte é calculado a cada incremento que chega, ou seja, o número *total de sequências da base* é muito menor do que o utilizado agora, em que este atinge seu valor máximo.

Por exemplo, nos testes explicados nesta subseção, o *total de sequências da base* tem valor 20450. Nos testes com incremento, esse valor aumenta em torno de 2000 a partir do segundo incremento e atinge 20450 sequências apenas ao executar o último incremento.

Por fim, a estratégia de testes utilizadas neste trabalho é de avançar do menor para o maior, ou seja, iniciar com o comportamento básico tanto do GSP como do IncMSTS e evoluir os dados e parâmetros dos algoritmos, de forma a trilhar um caminho lógico que cada vez mais explicite as diferenças que vão distanciando o início (GSP) do intermediário (IncMSTS) e da proposta deste trabalho (Esper + IncMSTS).

5.4.1.1 GSP

A execução do algoritmo GSP, com suporte 0.02, resulta na imagem a seguir.

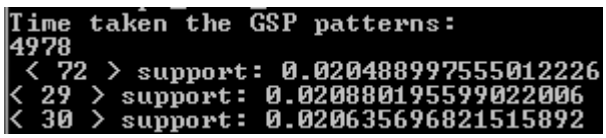


Figura 5.1 - Resultado GSP para suporte 0.02 com base de dados completa

Conforme mostrado na Figura 5.1, o GSP foi capaz de encontrar três padrões unitários (<72>, <29> e <30>) com suporte mínimo de 0.02. Caso o suporte suba

para 0.03, nenhum padrão é identificado. Conforme dito, esse teste não mostra nenhuma surpresa, serve apenas para ser o marco inicial de onde começa a evolução dos testes.

5.4.1.2 IncMSTS

A execução do algoritmo IncMSTS, com suporte 0.02, janelamento 1 e semi frequentes 0.9999 resulta na Figura 5.2 a seguir.

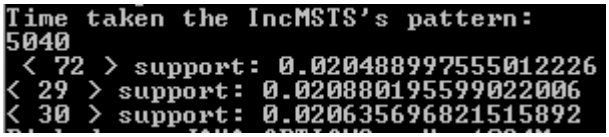


Figura 5.2 - Resultado IncMSTS para suporte 0.02, sem janelamento e semi frequentes para base de dados completa.

Conforme mostrado na Figura 5.2, a execução é semelhante ao GSP, retornando exatamente os mesmos padrões e elevando ligeiramente o tempo de execução.

O principal interesse na execução deste teste é mostrar a semelhança esperada do algoritmo GSP com o IncMSTS caso a base de dados tenha o comportamento estático. A partir do princípio que ambos os algoritmos buscam os mesmo padrões praticamente no mesmo tempo, os próximos testes atacam os principais pontos que diferenciam as abordagens utilizadas neste trabalho.

5.4.2 Com Incremento

Os dados agora se comportam como um fluxo de dados. Não há mais o conceito de base de dados e mineração de dados estática. Os dados são os mesmos presentes na base inicial da FIFA, mas agora divididos em incrementos que são passados como entrada para os algoritmos.

Conforme explicado anteriormente, o suporte sobe para o valor 0.04 e 0.05.

5.4.2.1 GSP

A execução do algoritmo GSP, com suporte 0.04 e realizada apenas no primeiro incremento (teste0.txt), resulta na imagem a seguir.

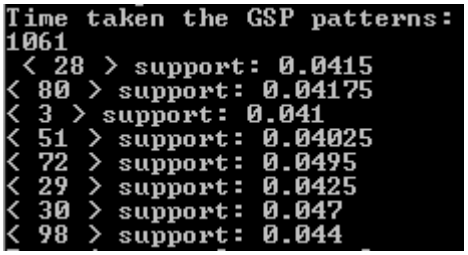


Figura 5.3 - Resultado GSP para suporte 0.04 no primeiro incremento.

Na Figura 5.3, o GSP, que sem incremento não encontrava padrões caso o suporte fosse maior que 0.02, agora encontra 8 padrões unitários com suporte mínimo 0.04. Além disso, seu tempo de execução despencou de quase 5 segundos para pouco mais que 1 segundo. Isso se deve ao incremento tratado aqui ser de tamanho 80% inferior em relação a base completa, utilizada no teste anterior. A desvantagem é que, como o GSP não é um algoritmo incremental, simplesmente ignora os 8 incrementos gerados a seguir, perdendo mais de 88% dos dados gerados.

Ainda sobre o GSP, para permitir a comparação futura mais fiel com o IncMSTS e Esper + IncMSTS, foi realizada a sua execução para o primeiro incremento novamente, mas com suporte mínimo elevado para 0.05. O resultado é a Figura 5.4 a seguir.

Figura 5.4 - Resultado GSP para suporte 0.05 no primeiro incremento.

Neste ponto, na Figura 5.4, o GSP não é capaz de encontrar padrões no primeiro incremento e ignora os próximos incrementos de dados.

5.4.2.2 IncMSTS

Ainda sem utilizar todo o potencial do IncMSTS, novamente são mostrados os testes semelhantes ao que fez o algoritmo GSP. O IncMSTS utiliza apenas o primeiro incremento, com suporte 0.04, janelamento 1 e semi frequentes 0.9999.

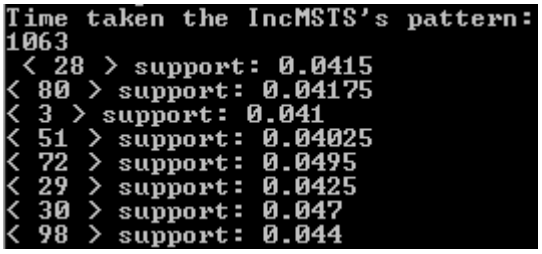


Figura 5.5 - Resultado IncMSTS para suporte 0.04 no primeiro incremento.

Conforme a Figura 5.5, o resultado é idêntico ao GSP tanto nos padrões encontrados como no suporte. Além disso, o aumento no tempo de execução é insignificante.

A repetição do teste é mostrada na Figura 5.6, com a alteração de que o suporte mínimo agora é 0.05.

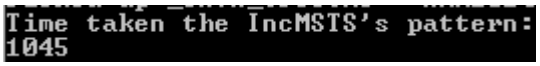


Figura 5.6 - Resultado IncMSTS para suporte 0.05 no primeiro incremento.

O IncMSTS se equivale ao algoritmo GSP e também não encontra nenhum padrão no primeiro incremento.

O algoritmo IncMSTS utilizado com todas as suas características é o alvo dos testes a partir desse momento. As diversas configurações de testes mostradas a seguir permitem a comparação fiel tanto para o lado do GSP como para o lado Esper + IncMSTS. Assim, é possível realizar várias análises e chegar a algumas conclusões.

O primeiro teste do IncMSTS com todos os incrementos tem:

- suporte mínimo 0.05,
- janelamento 1 e
- semi frequentes 0.9999.

Nesse caso, o IncMSTS faz uso de sua parte incremental, mas ainda não utiliza janelamento e semi frequentes. A saída do algoritmo é mostrada na Figura 5.7.

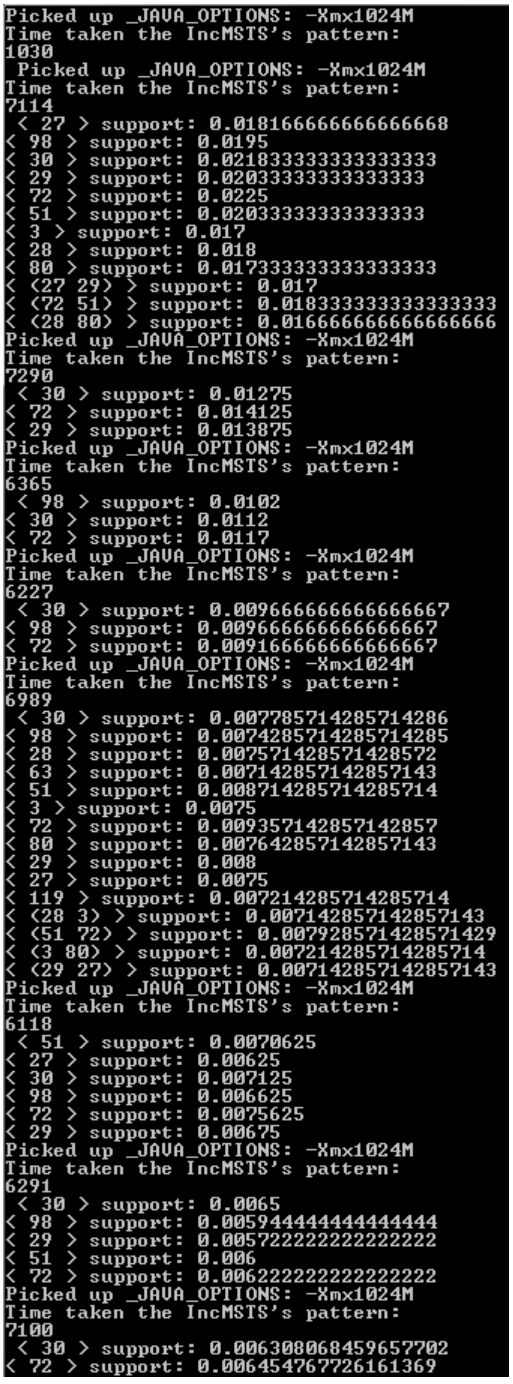


Figura 5.7 - Resultado do IncMSTS com todos os incrementos, suporte mínimo 0.05, sem janelamento e semi frequentes.

O IncMSTS mostra as saídas para cada incremento individualmente, com os padrões encontrados, o tempo decorrido para cada incremento e os itens que são frequentes para aquele incremento, mesmo que não seja para todos os dados.

Para o suporte mínimo definido como 0.05, por exemplo, no primeiro incremento nenhum padrão foi identificado. Com o segundo incremento, diversas sequências atingem o suporte mínimo, porém ao se somar com as sequências do

primeiro incremento, perdem força. São essas as sequências mostradas nos incrementos e que não atingem o suporte mínimo para todos os dados, mas para aquele momento são relevantes.

Para os parâmetros do teste anterior, nenhum padrão é identificado nos dados após executar o algoritmo IncMSTS em todos os incrementos. Nesse ponto existem duas análises. A primeira é que o resultado é igual ao GSP quando executa na base inteira, não retornando padrões. A segunda é que, por mais que não retorne padrões, o IncMSTS já mostra alguns itens com potencial a se tornar um padrão, que em determinados incrementos se destacaram. Além disso, falta ainda utilizar o janelamento e o fator de semi frequentes e verificar se, assim, algum padrão é identificado.

O próximo teste do IncMSTS tem:

- suporte mínimo 0.05,
- janelamento 50 e
- semi frequentes 0.9999.

Nesse caso, o IncMSTS faz uso de sua parte incremental, do janelamento mas não de semi frequentes. A saída do algoritmo é mostrada na Figura 5.8.

Por conta do janelamento, vários potenciais padrões cresceram e exibem sequências maiores agora, destacadas nas figuras por meio de setas azuis. O janelamento não teve efeito no reconhecimento de novos padrões, ou seja, ainda ocorre a mesma saída do teste anterior. Além disso, o tempo de execução de cada incremento subiu ligeiramente ao aumentar o tamanho da janela do IncMSTS.

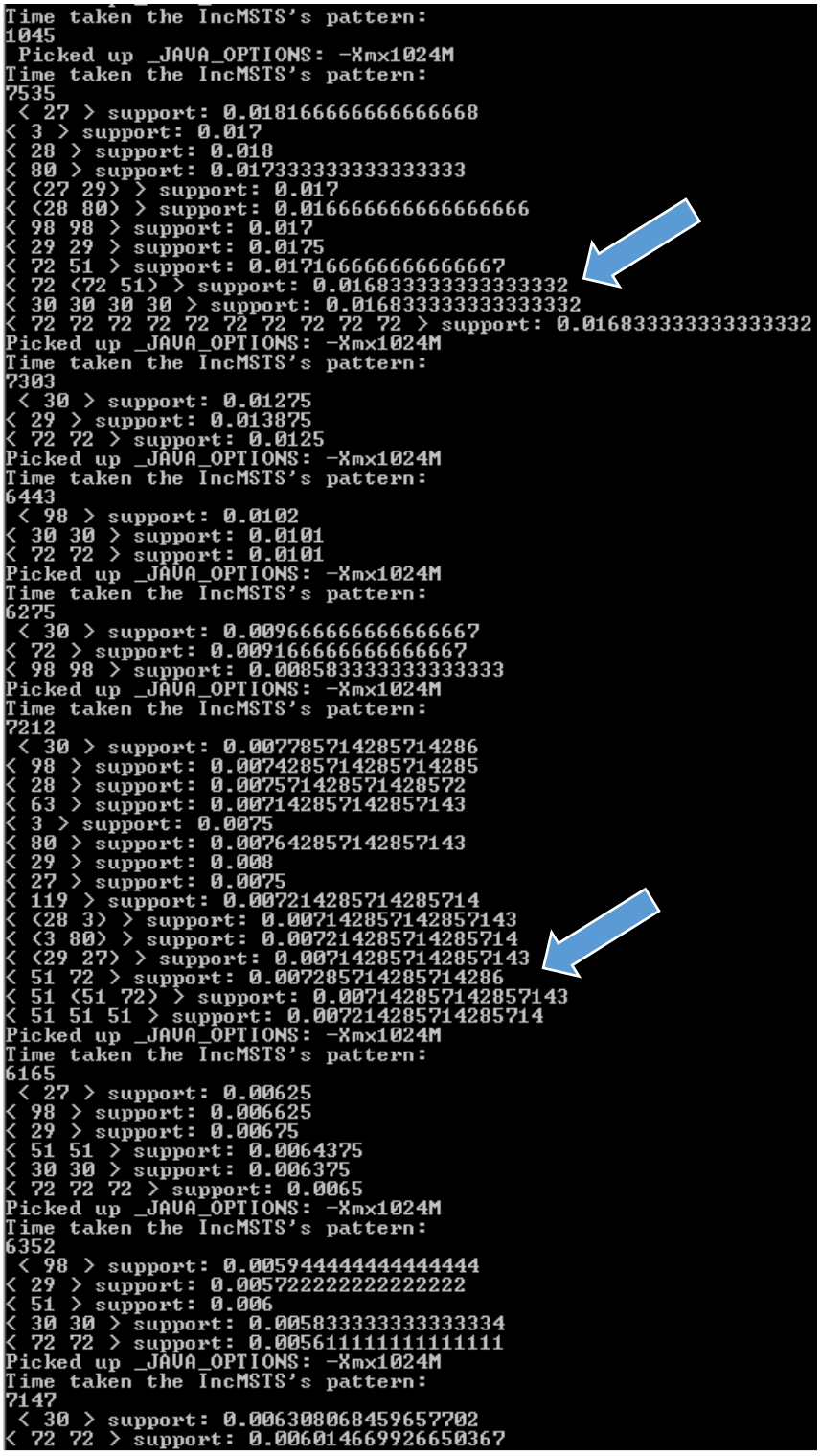


Figura 5.8 - Resultado do IncMSTS com todos os incrementos, suporte mínimo 0.05, com janelamento e sem semi frequentes.

O próximo teste do IncMSTS tem:

- suporte mínimo 0.05,
- janelamento 1 e
- semi frequentes 0.1.

Nesse caso, o IncMSTS faz uso de sua parte incremental, dos semi frequentes mas não do janelamento. A saída do algoritmo é mostrada na Figura 5.9. O primeiro incremento que retorna vazio não é exibido.

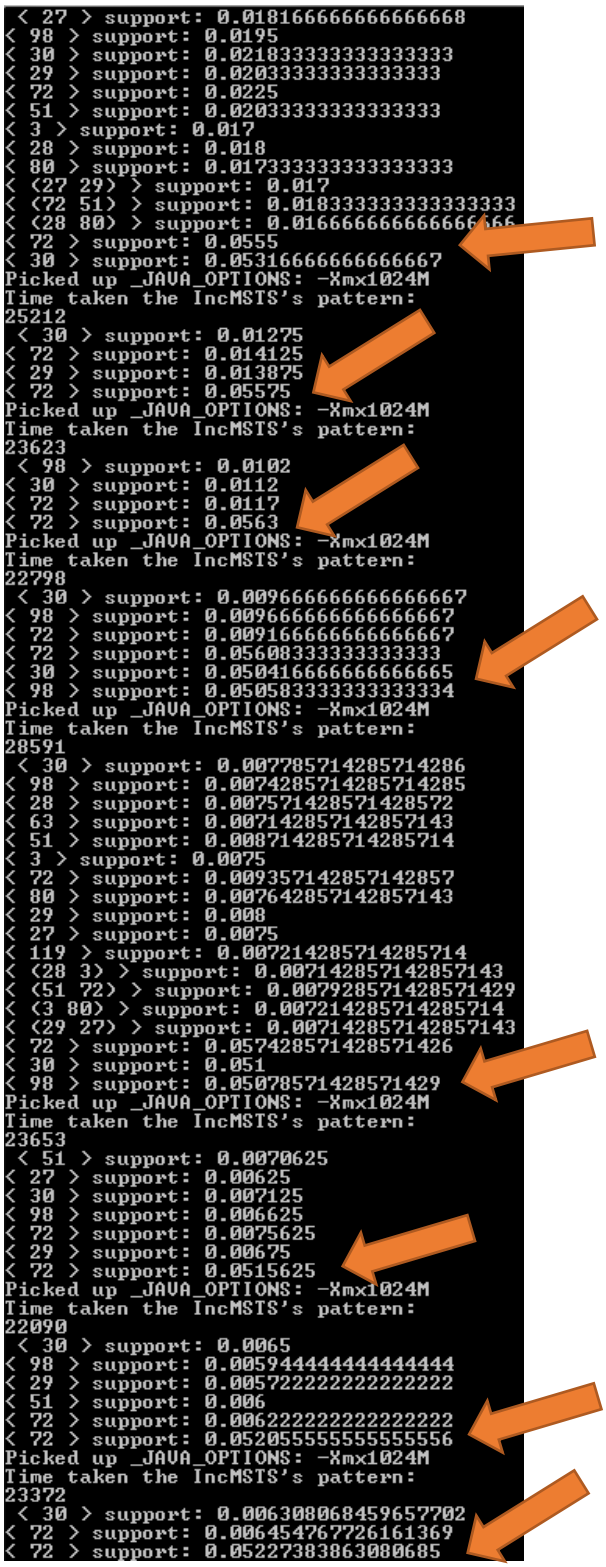


Figura 5.9 - Resultado do IncMSTS com todos os incrementos, suporte mínimo 0.05, sem janelamento e com semi frequentes.

Na Figura 5.9 são mostrados dois grandes impactos da adoção dos semi frequentes para os padrões encontrados pelo IncMSTS. Se por um lado é possível ver alguns itens destacados com setas alaranjadas (<72>, <30> e <98>) que atingem o suporte mínimo em diferentes incrementos, também é verdade o grande salto ocorrido no tempo de execução, em alguns casos um tempo quase quatro vezes maior.

Na execução desse teste, o IncMSTS consegue mostrar itens semi frequentes que se tornaram frequentes quando o usuário favorece essa característica do algoritmo.

O próximo teste, Figura 5.10, explora tudo que o IncMSTS permite:

- suporte mínimo é de 0.05,
- o janelamento é 50 e
- semi frequentes 0.01.

Dessa forma a execução engloba incrementos, uma janela de 50 sequências e um intervalo considerável abaixo do suporte mínimo que seleciona itens semi frequentes.

Seguindo a lógica esperada, novamente o janelamento aumentou o tamanho de algumas sequências, por mais que nenhuma tenha atingido o suporte mínimo. O tempo de execução também teve um ligeiro aumento.

Após executar o IncMSTS, variar todos os seus parâmetros e analisar seu comportamento, fica claro o caminho percorrido: se buscar mais padrões com maior semântica, então maior tempo de execução. O IncMSTS não é um algoritmo pensado para realizar mineração em ambiente de fluxo de dados, portanto, o tempo de execução desempenhado nos testes anteriores são aceitáveis.

No escopo deste trabalho, por outro lado, o tempo de execução é tão ou mais importante que a quantidade e qualidade dos padrões encontrados. Portanto, na próxima seção é mostrado e analisado o comportamento da abordagem Esper + IncMSTS, que adapta o algoritmo para o ambiente de mineração de fluxo de dados.

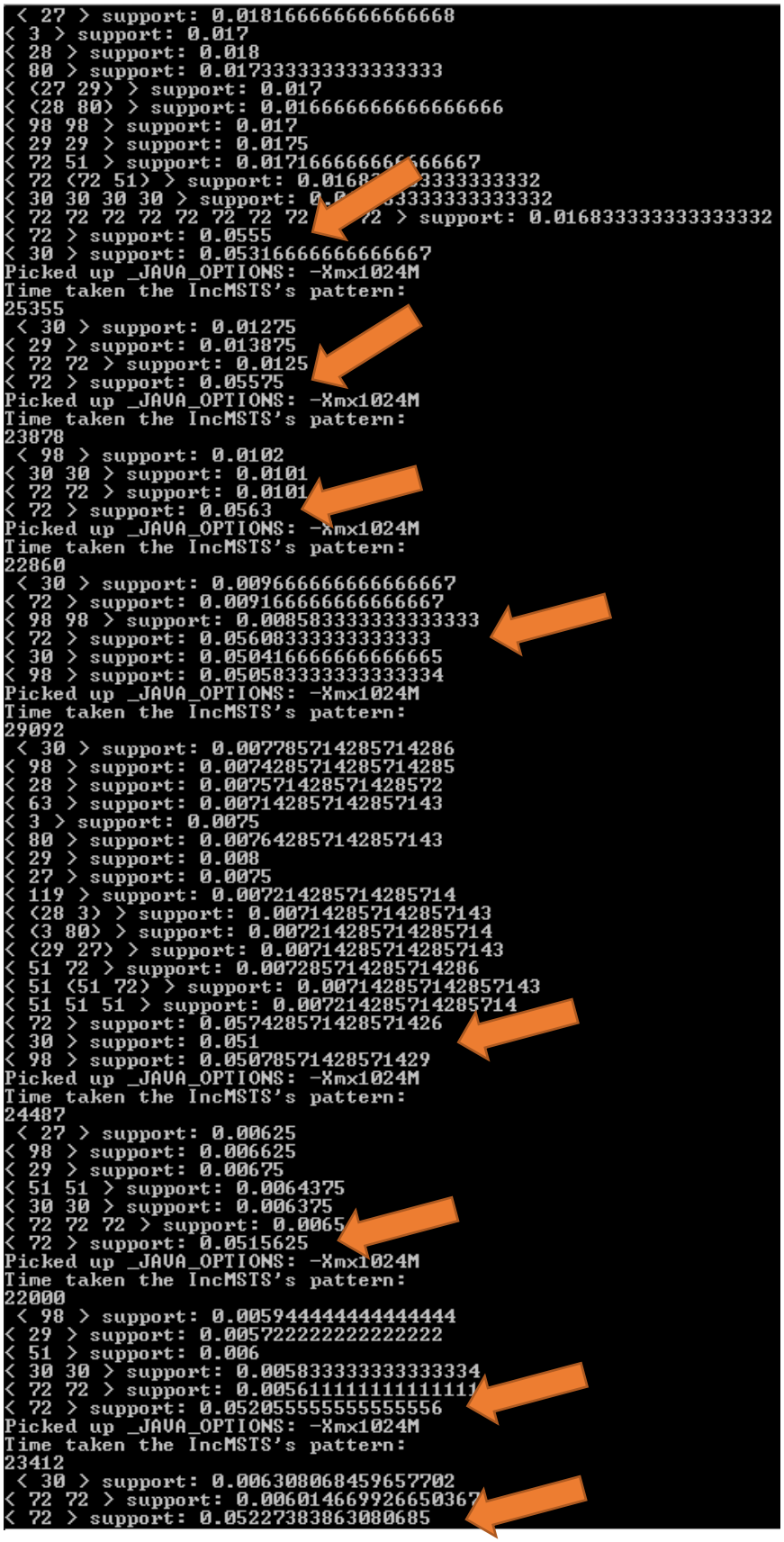


Figura 5.10 Resultado do IncMSTS com todos os incrementos, suporte mínimo 0.05, com janelamento e semi frequentes.

5.4.2.3 Esper + IncMSTS

A lógica utilizada nos testes a seguir é a de executar o último arquivo gerado pelo janelamento do Esper. Assim, o IncMSTS apenas utiliza o último arquivo modificado e ignora completamente quaisquer arquivos que tenham sido gerados enquanto o algoritmo ainda processava o incremento anterior.

Essa lógica vai de encontro ao que foi feito na seção anterior, em que o IncMSTS lia sistematicamente incremento por incremento, em ordem crescente, independentemente do tempo de execução.

Nesta seção o que mais importa é o tempo. As análises foram feitas em cima do quão próximo a abordagem Esper + IncMSTS foi capaz de chegar em relação aos padrões do IncMSTS, além da explicação do motivo da sua proximidade.

Como foi dito anteriormente, a base de dados foi dividida em 9 incrementos, de teste0.txt até teste8.txt. Então foram 27 segundos para a geração desses incrementos, com o IncMSTS tendo permissão para ler os últimos arquivos modificados durante 30 segundos.

Os parâmetros da abordagem Esper + IncMSTS utilizados aqui são idênticos aos utilizados nos diversos testes do IncMSTS, a fim de permitir a comparação mais justa possível.

O primeiro teste, na Figura 5.11, tem:

- suporte mínimo de 0.05,
- janelamento 1 e
- semi frequentes 0.9999.

Portanto, aqui Esper + IncMSTS se aproxima do GSP, mas utiliza incrementos e ignora semi frequentes e janelamento. Além disso, os arquivos de incremento lidos são:

- teste0.txt,
- teste1.txt,
- teste3.txt,
- teste5.txt e
- teste7.txt

Esses são os últimos arquivos gerados pelo Esper no momento que o IncMSTS acaba sua execução anterior.

Note que nos testes a partir de agora, a coluna da direita tem a saída do IncMSTS para efeito de comparação. Além disso, a coluna da direita sempre tem a saída para o último incremento (teste8.txt), que não é realizada pelo Esper + IncMSTS em nenhum dos testes realizados. Por mais que o teste8.txt esteja sempre disponível ao Esper + IncMSTS, foi considerado que a abordagem só o tomaria como entrada caso ocorresse no tempo correto de acordo com a lógica definida anteriormente. Como o Esper + IncMSTS sempre tomou conhecimento do teste8.txt após os 30 segundos de tempo de execução, o incremento acabou desconsiderado.

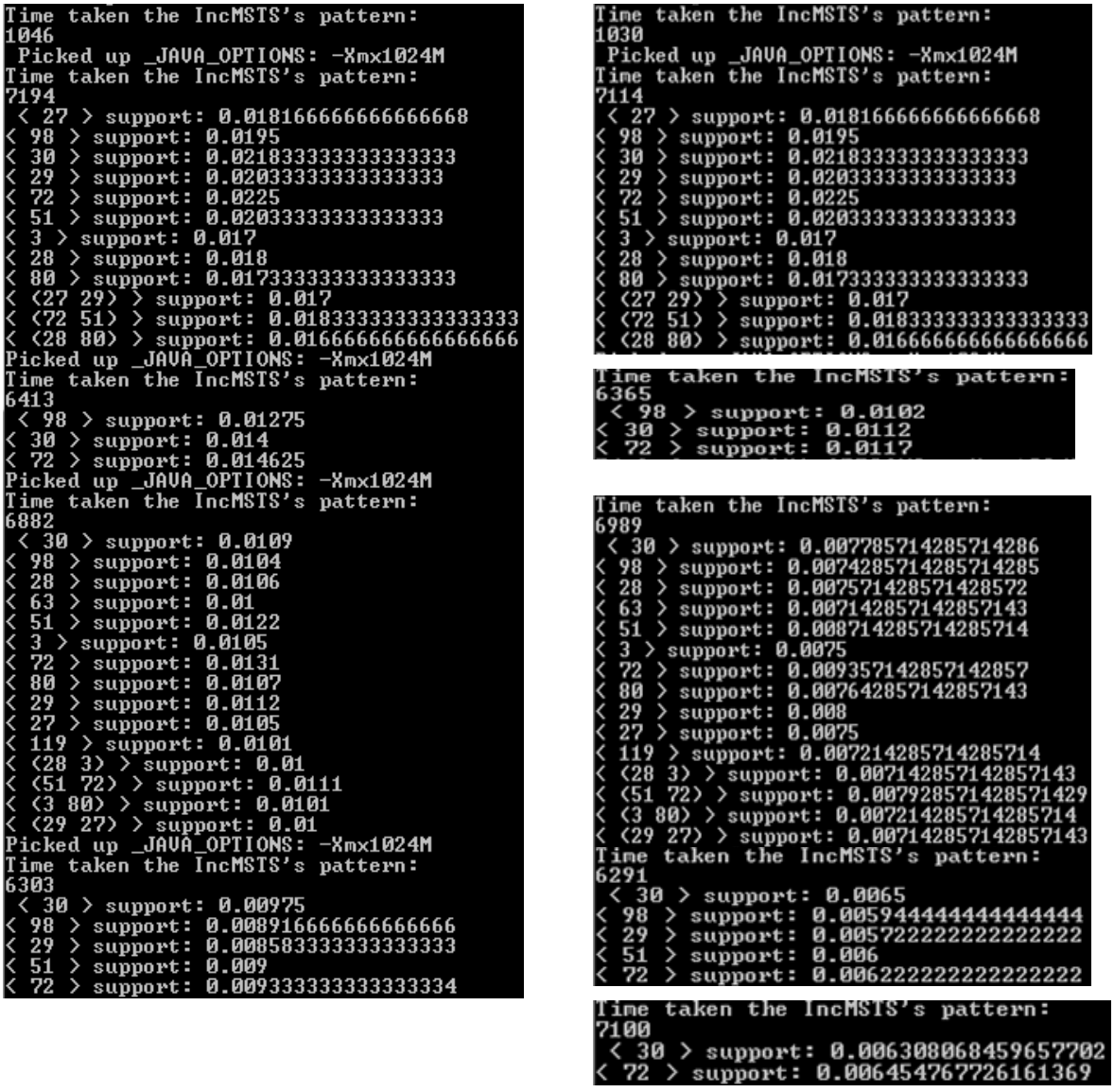


Figura 5.11 - Do lado esquerdo a saída Esper + IncMSTS e do direito a saída IncMSTS para comparação. A saída da direita inclui o incremento final não presente na abordagem Esper + IncMSTS.

Conforme mostrado na Figura 5.11, a equivalência entre as duas abordagens é idêntica enquanto executam o mesmo caminho (teste0.txt e teste1.txt), apenas com um tempo de execução ligeiramente superior para o Esper + IncMSTS.

A partir do terceiro incremento da coluna da esquerda (teste3.txt, com teste2.txt ignorado), as sequências encontradas permanecem as mesmas em todos os incrementos, com tempo maior quase sempre para o Esper + IncMSTS. Além disso, o suporte da abordagem da esquerda é maior também, pois conta com menos dados e, consequentemente, o valor do suporte sobe.

Esse primeiro teste mostrou que Esper + IncMSTS encontra as mesmas sequências potenciais que o IncMSTS nos incrementos selecionados, mas agora com limite de tempo.

O próximo teste, na Figura 5.12, tem:

- suporte mínimo de 0.05,
- janelamento 50 e
- semi frequentes 0.9999.

Portanto, aqui o Esper + IncMSTS utiliza incrementos e janelamento, ignora semi frequentes. Além disso, os arquivos de incremento lidos são:

- teste0.txt,
- teste1.txt,
- teste3.txt,
- teste5.txt e
- teste7.txt:

O tempo de execução nesse caso teve um ligeiro aumento para o Esper + IncMSTS, mas nada que alterasse os incrementos a serem lidos.

Novamente há a equivalência total de sequências potenciais encontradas no Esper + IncMSTS com o IncMSTS. O suporte também mantém sua tendência de aumento, mas ainda assim nenhum padrão sequencial foi encontrado em nenhuma das duas abordagens.

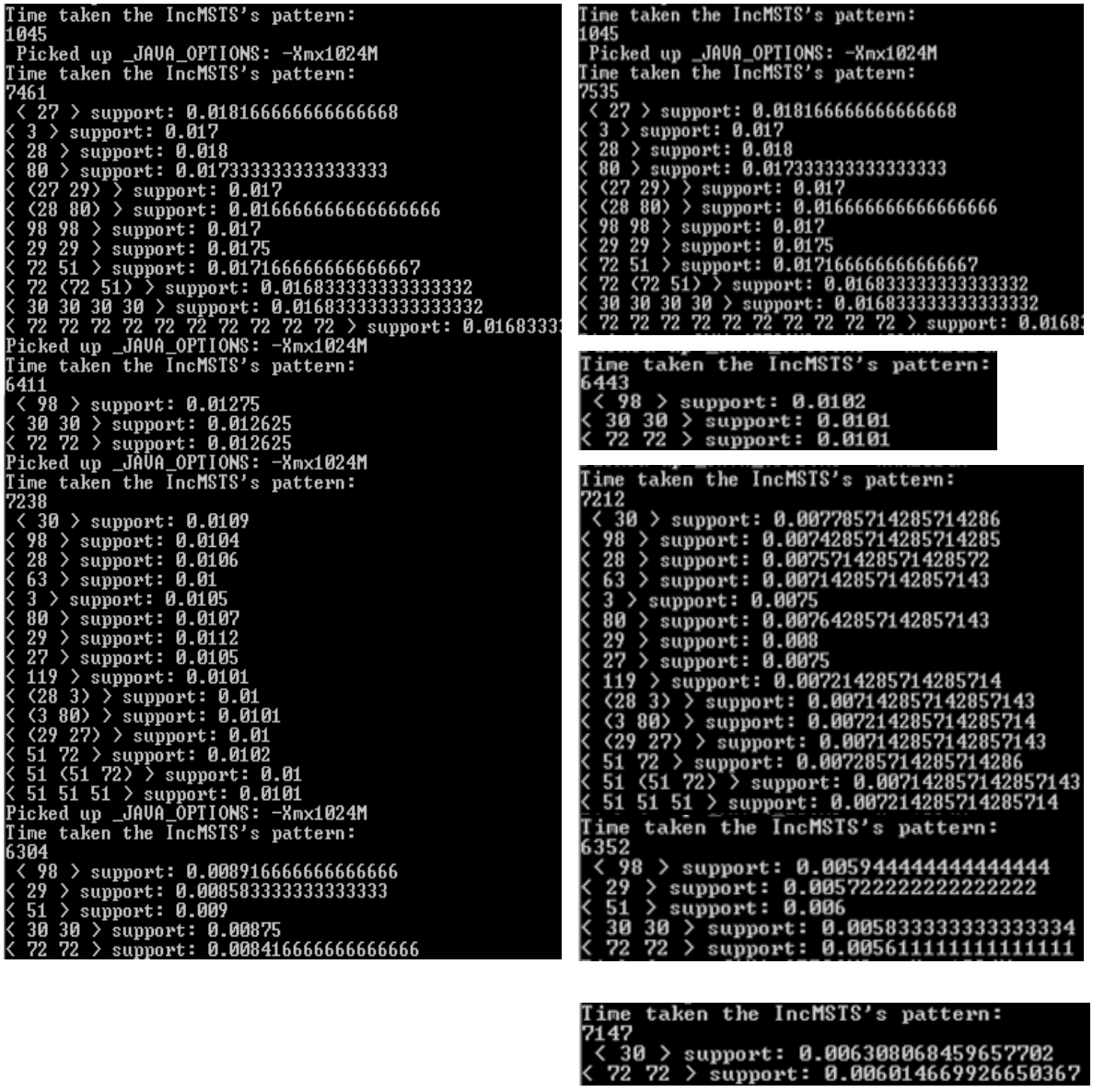


Figura 5.12 - Do lado esquerdo a saída Esper + IncMSTS e do direito a saída IncMSTS para comparação. A saída da direita inclui o incremento final não presente na abordagem Esper + IncMSTS.

O próximo teste, na Figura 5.13, tem:

- suporte mínimo de 0.05,
- janelamento 1 e
- semi frequentes 0.1.

Portanto, aqui o IncMSTS utiliza incrementos e semi frequentes, ignora o janelamento. Além disso, os arquivos de incremento lidos são:

- teste0.txt,
- teste1.txt e

- teste4.txt:

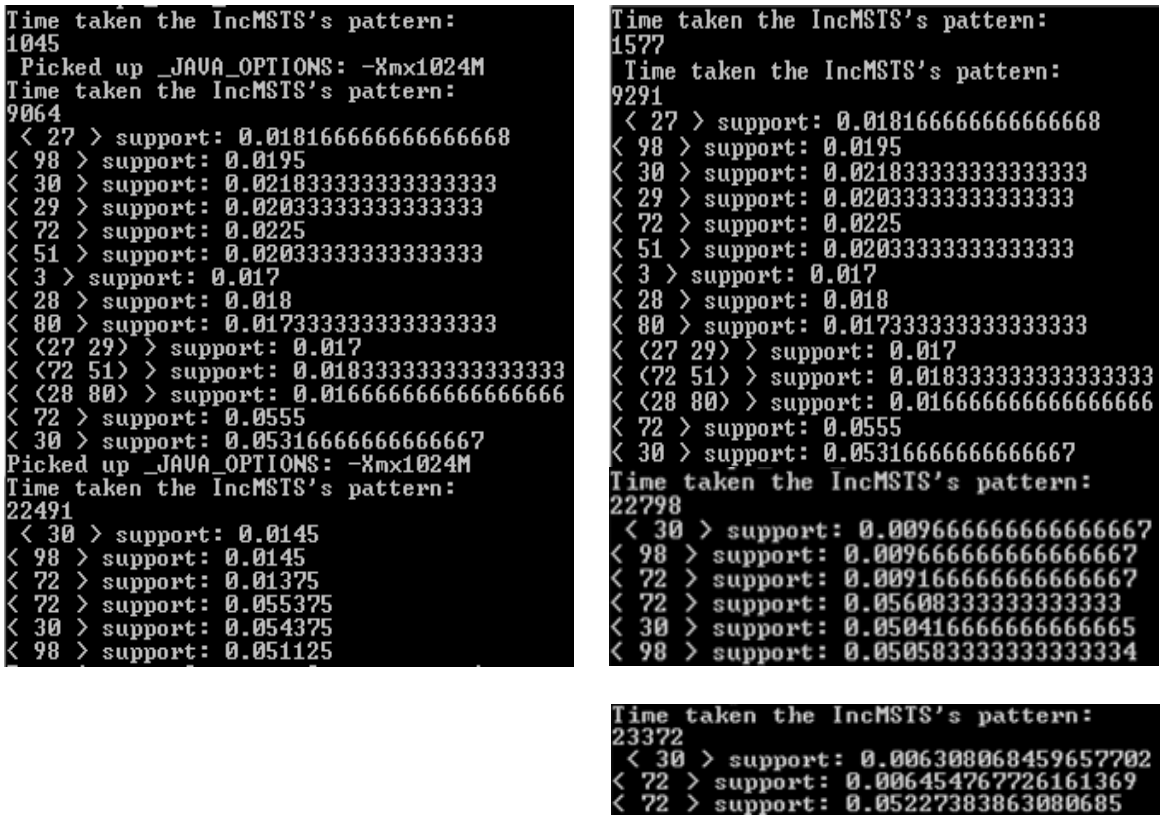


Figura 5.13 - Do lado esquerdo a saída Esper + IncMSTS e do direito a saída IncMSTS para comparação. A saída da direita inclui o incremento final não presente na abordagem Esper + IncMSTS.

Ao aumentar os semi frequentes, o tempo de execução cresceu consideravelmente, o que levou a ler menos arquivos. Agora o teste4.txt é o último do conjunto de incrementos a ser lido.

Agora o IncMSTS, à direita da Figura 5.13, já é capaz de identificar padrões que atingem o suporte mínimo nos incrementos 1 e 4 (<72> e <30>; <72>, <30> e <98>, respectivamente). O Esper + IncMSTS também foi capaz de identificar os mesmos padrões, inclusive com o suporte mais elevado.

Se levado em conta que no incremento final do IncMSTS o padrão <72> foi o único considerado frequente, o Esper + IncMSTS o reconheceu mesmo fazendo a leitura de apenas de 33% do fluxo de dados.

O próximo teste, na Figura 5.14, tem:

- suporte mínimo de 0.05,
- janelamento 50 e

- semi frequentes 0.1.

Portanto, aqui o IncMSTS utiliza incrementos, janelamento e semi frequentes. Além disso, os arquivos de incremento lidos são:

- teste0.txt,
- teste1.txt e
- teste4.txt:

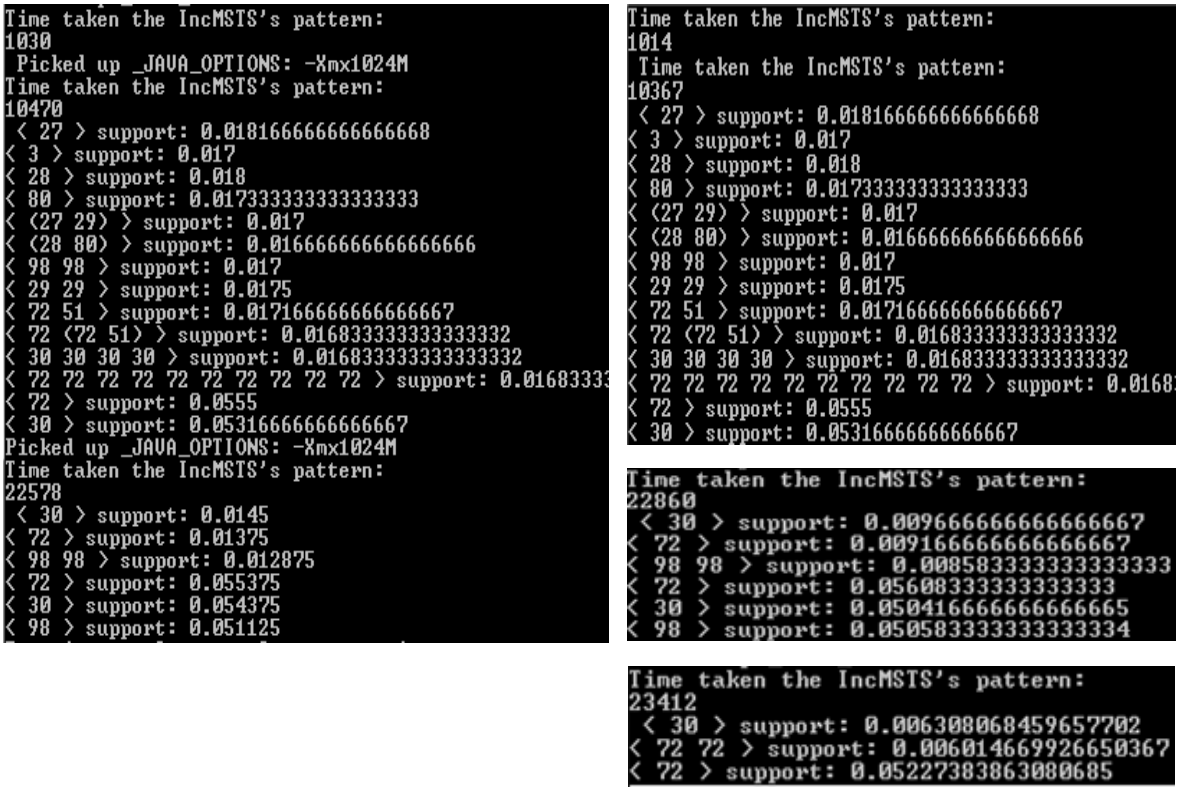


Figura 5.14 - Do lado esquerdo a saída Esper + IncMSTS e do direito a saída IncMSTS para comparação. A saída da direita inclui o incremento final não presente na abordagem Esper + IncMSTS.

Novamente, a abordagem Esper + IncMSTS é capaz de identificar exatamente os mesmos padrões e sequências com potencial nos incrementos lidos.

Se levado em conta que no incremento final do IncMSTS o padrão <72> foi o único considerado frequente, o Esper + IncMSTS o reconheceu mesmo fazendo a leitura de apenas de 33% do fluxo de dados.

Assim como no teste anterior, entretanto, a abordagem Esper + IncMSTS tem um problema mesmo ignorando incrementos. A execução do incremento teste4.txt, após ignorar 2 e 3, é lenta se observado o ambiente de *Big Data* e mineração em tempo real proposto. Cabe ao conhecedor de domínio estabelecer a aceitação ou

não desse desempenho. Além disso, sempre se tem a medida do suporte que pode ser aumentada para minimizar o tempo de execução em detrimento da quantidade de padrões identificados.

5.5 Resultados

A seguir serão pontuados alguns itens importantes concluídos a partir dos testes.

- O módulo incremental do IncMSTS se mostrou essencial por conta da sua facilidade de agregar dados a uma base já definida. Mesmo sendo pensado para o ambiente estático, essa característica traz flexibilidade ao IncMSTS, permitindo que, ao contrário de outras abordagens, consiga tratar de todos os dados que chegam ao fluxo de dados. Os incrementos deste trabalho chegavam de 3 em 3 segundos com pouco mais de 2000 sequências e o módulo incremental se mostrou uma saída conceitualmente simples e com resultado satisfatório.
- A base de dados utilizada (páginas do site da FIFA na Copa da França de 1998) não tirou proveito da mineração de dados esparsos, um dos diferenciais do IncMSTS. O fato de não haver padrões que surgissem a partir do janelamento do IncMSTS, que estivessem dentro do suporte mínimo, favoreceu a equivalência total nos padrões entre IncMSTS e Esper + IncMSTS.
- Ao mesmo tempo em que a abordagem Esper + IncMSTS favorece o aparecimento de mais padrões, já que tende a elevar o suporte, também pode ser prejudicial no caso de fluxos de dados com tendências de tempo curto, pois pode acontecer dos incrementos contendo essas tendências serem ignorados pelo Esper + IncMSTS, de forma a garantir a velocidade de execução.
- Deve-se tomar cuidado com o tempo de execução que Esper + IncMSTS pode alcançar. Mesmo ao ignorar incrementos para equilibrar velocidade e extração de padrões, com o aumento do janelamento e, principalmente, dos semi frequentes, o tempo de execução do

algoritmo sobe de maneira agressiva, o que pode, em muitos casos, inviabilizar sua execução em ambiente de fluxo de dados.

- O fato de que em alguns casos apenas 33,3% dos dados foram utilizados por Esper + IncMSTS e, ainda assim, sua saída esteve de acordo com a versão estática, comprova a hipótese que a abordagem Esper + IncMSTS tem espaço no ambiente de fluxo de dados.

De acordo com o que foi mostrado nesse e nos capítulos anteriores, a hipótese de que o algoritmo IncMSTS seria capaz de realizar a mineração de padrões sequenciais em ambiente *Big Data* se mostrou válida, mas com ressalvas importantes em relação aos seus parâmetros de entrada.

Se por um lado pode-se utilizar esse algoritmo com fluxo de dados, diferente da proposta inicial de trabalhar com dados estáticos, é verdade, também, que devem ser observadas limitações importantes em relação ao tempo de execução do algoritmo.

Uma execução que utilize todos os diferenciais do IncMSTS e ainda valor de suporte baixo fatalmente encontrará dificuldades no tempo de execução, exceto casos de domínios específicos.

O caso mais comum é também o que fornece a maior possibilidade de manipulação. Baseado no que se recebe de entrada e qual o objetivo da saída, pode-se fazer um IncMSTS mais ou menos preciso na formação dos padrões e dessa forma atingir a velocidade de execução pretendida. Isso também vale para o tamanho e tipos de janela externa do Esper, em que cada domínio pode tirar proveito de um tipo, mesmo que, em nossos testes, janelas do Esper com menor quantidade de dados e com incrementos constantes tenham mostrado melhor desempenho do que grandes quantidades de dados em cada janela, com maior tempo.

Na tabela 5.2 é mostrado um resumo da discussão anterior.

Tabela 5.2 – Resultados das diferentes execuções do GSP, IncMSTS e Esper + IncMSTS.

	Incremento	% de incremento utilizado	Janelamento	Semi frequentes	Resultado
GSP	Não	-	-	-	Padrões idênticos. Tempo semelhante
IncMSTS	Não	-	Não	Não	
GSP	Sim (teste0.txt)	-	-	-	Padrões idênticos. Tempo semelhante.
IncMSTS	Sim (teste0.txt)	-	Não	Não	
IncMSTS	Sim	100	Não	Não	Esper + IncMSTS capaz de retornar sequências equivalentes a sua versão estática IncMSTS com tempo semelhante.
Esper + IncMSTS	Sim (teste0.txt, 1, 3, 5 e 7)	55,5			
IncMSTS	Sim	100	Sim	Não	Esper + IncMSTS capaz de retornar sequências equivalentes a sua versão estática IncMSTS com tempo semelhante.
Esper + IncMSTS	Sim (teste0.txt, 1, 3, 5 e 7)	55,5			
IncMSTS	Sim	100	Não	Sim	Esper + IncMSTS capaz de retornar padrões equivalentes a sua versão estática IncMSTS. Fica a ressalva para o tempo de execução, que pode impedir execução em ambiente de fluxo de dados.
Esper + IncMSTS	Sim (teste0.txt, 1 e 4)	33,3			
IncMSTS	Sim	100	Sim	Sim	Esper + IncMSTS capaz de retornar padrões equivalentes a sua versão estática IncMSTS. Fica a ressalva para o tempo de execução, que pode impedir execução em ambiente de fluxo de dados.
Esper + IncMSTS	Sim (teste0.txt, 1 e 4)	33,3			

Portanto, a abordagem Esper + IncMSTS é capaz de retornar padrões semelhantes ao uso original em dados estáticos. Mesmo ao utilizar em alguns momentos apenas 33% da quantidade de dados, a abordagem Esper + IncMSTS

conseguiu exibir a sequência de cliques dos usuários do site da FIFA assim como o IncMSTS fez lendo 100% dos dados em tempo consideravelmente superior.

5.6 Considerações Finais

Neste capítulo foram apresentados os testes e resultados que foram alcançados com este projeto de mestrado. A hipótese de utilização do conjunto Esper + IncMSTS em ambiente *Big Data* foi comprovada por meios de diversos testes, que também mostraram a necessidade de um ajuste fino na relação a configuração dos parâmetros de Esper + IncMSTS versus respostas em tempo real. Por fim, foram mostrados os resultados e uma tabela para facilitar a interpretação dos resultados por parte do leitor. No capítulo a seguir será apresentada a conclusão para o fechamento desta dissertação de mestrado.

Capítulo 6

Conclusão

Neste capítulo estão as conclusões sobre este trabalho, assim como as contribuições e os trabalhos futuros. Na seção 6.1 é realizada as considerações iniciais, na 6.2 as contribuições e trabalhos futuros e na 6.3 o encerramento da seção e da dissertação.

6.1 Considerações Iniciais

O processo de descoberta de conhecimento útil tem sido utilizado há pelo menos duas décadas e tem diversas pesquisas realizadas. Ainda assim, é possível repensar algumas formas de entendimento e adaptá-las, na medida do possível, para atender as necessidades do mundo atual.

A frase dita no trabalho de Lu et al. (1995), já falava “dados ricos, mas conhecimento pobre”. Se há mais de 20 anos essa frase fez sentido e motivou diversas pesquisas, seu valor hoje tende a ser ainda maior.

Em 1995 se vivia a fase do armazenamento de dados e o desenvolvimento de algoritmos em diversas tarefas de mineração de dados a fim de reconhecer padrões e extrair conhecimento dessa base. Porém, a característica principal que marca esse modelo é o armazenamento, a premissa de que os dados estão lá à espera de serem analisados e, mesmo que demorem, algoritmos consigam extrair conhecimento útil a partir desse armazém.

Pois essa premissa foi alterada em diversas áreas nos últimos anos. Em um mundo em que tudo é dinâmico, chega a ser natural pensar que os dados a serem analisados também sejam. Os dados estão chegando, passando e se tornando obsoletos e passaram-se apenas alguns segundos no mundo real.

Assim, diversas novas abordagens, assim como há 20 anos, vem discutindo como extrair conhecimento útil e aplicar a mineração de dados em tempo real, em fluxo de dados, em ambiente *Big Data*.

Uma ideia realizada com sucesso foi a adaptação de um algoritmo de mineração de regras de associação, desenvolvido para minerar bases de dados estáticas, para o ambiente de fluxo de dados. Esse trabalho conseguiu extrair padrões de dados que chegavam através de fluxo, em alta velocidade, com tempo de resposta ao que se espera de uma aplicação de tempo real. Portanto, essa é uma abordagem promissora e que pode ser também adaptada para outros tipos de algoritmos.

6.2 Contribuições e Trabalhos Futuros

A principal contribuição é a conclusão de que a abordagem Esper + IncMSTS pode ser utilizada em ambiente *Big Data*, o que confirma a hipótese levantada no início do trabalho, ou seja, a hipótese de que é possível unir Esper + IncMSTS e responder de forma satisfatória aos desafios da mineração de dados em ambiente *Big Data*.

Os problemas levantados no início deste trabalho e que fizeram parte da motivação agora podem ser respondidos:

- Por que são necessárias novas abordagens para a análise de fluxo de dados com requisitos do *Big Data*?
    - Porque as abordagens tradicionais não atendem aos V's que caracterizam o *Big Data*.
- Os algoritmos clássicos de mineração de dados podem ser úteis com a quantidade e velocidade de dados presentes nos fluxos de dados?
    - São úteis, como foi comprovado, mas necessitam do apoio de novas ferramentas, como as ferramentas CEP.
- Como os algoritmos clássicos de mineração de dados podem ser úteis com a quantidade e velocidade de dados presentes nos fluxos de dados?
    - Ao realizar a mineração de dados em janelas de dados vindas do fluxo.
- É possível unir um algoritmo de mineração de padrões sequenciais com um mecanismo de manipulação de fluxo de dados?

    - Sim, é possível. O mecanismo Esper e o algoritmo IncMSTS foram unidos para formar a abordagem Esper + IncMSTS de mineração em fluxo de dados.
- É possível conseguir desempenho satisfatório com essa união?
    - É possível, pois a abordagem Esper + IncMSTS mostrou não ser necessário ler todos os dados para encontrar as tendências encontradas quando se lê todos os dados e respeita, assim, o tempo de criação dos incrementos de dados e não utiliza dados obsoletos.
- A nova abordagem proposta neste trabalho é relevante para o estado da arte dos assuntos discutidos?
    - A abordagem Esper + IncMSTS é relevante pois abre um leque de opções a serem pesquisadas e permite diversos trabalhos futuros.

Além disso, os testes mostraram a grande gama de opções oferecidas por essa abordagem, tanto do lado do Esper, que foi explorado apenas em relação ao janelamento, como do lado do IncMSTS, com seus parâmetros de suporte, janelamento para minerar dados com ruídos e adição de itens semi frequentes.

Os testes mostraram, por exemplo, o grande impacto causado pelo aumento dos itens semi frequentes na execução do IncMSTS. Porém, isso foi o que permitiu que a abordagem encontrasse padrões mesmo com o suporte um pouco mais alto, ou seja, são padrões importantes que, com a mineração comum sem inclusão dos semi frequentes, não seriam encontrados.

Portanto, cabe ao usuário encontrar a melhor sintonia para seus dados e sua abordagem.

Como trabalho futuro, este projeto pode ser uma motivação a se explorar mais a fundo as possibilidades e resultados que o Esper pode trazer no tratamento dos dados, ou seja, quais alternativas existem ao se pensar sobre o pré-processamento no Esper antes da geração das janelas.

Por exemplo, o trabalho de (Choi et al., 2014) tem potencial para se associar ao IncMSTS, principalmente ao reduzir o tempo de processamento por meio dos pesos dinâmicos, reduzindo os padrões a serem identificados.

Ainda, outros algoritmos podem ser adaptados ao Esper para se buscar uma mineração de dados distinta, os algoritmos SPADE, SPAM, LAPIN, CM_SPADE e CM_SPAM, que são alguns dos melhores dentre os que realizam a mineração de padrões sequenciais, são bons candidatos a compor uma abordagem junto com o Esper.

Até mesmo outras bases de dados específicas podem tirar proveito das características do IncMSTS, que são, relembrando, a extração de padrões esparsos por meio de janelamento (janelamento interno do IncMSTS, não confundir com o janelamento do Esper) e a inclusão de itens semi frequentes na busca pelos padrões.

6.3 Considerações Finais

Neste capítulo final foram relembrados os conceitos que levaram a se levantar a hipótese para se realizar este trabalho. A evolução dos dados, o ambiente *Big Data* dos dias de hoje foram a motivação desta pesquisa. Após foram citadas as contribuições alcançadas com este trabalho e os trabalhos futuros que podem enriquecer o que foi pesquisado até aqui.

REFERÊNCIAS

AGRAWAL, R.; SRIKANT, R. **Fast algorithms for mining association rules in large databases**. In: BOCCA, J. B.; JARKE, M.; ZANIOLO, C. (Ed.). VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile. [S.l.]: Morgan Kaufmann, 1994. p. 487–499. ISBN 1-55860-153-8.

AHMED, C.; TANBEER, S.; JEONG, B. S. **Efficient mining of weighted frequent patterns over data streams**. In: High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on. [S.l.: s.n.], 2009. p. 400 –406.

AMATRIAIN, X. **Mining Large Streams of User Data for Personalized Recommendations**. In: SIGKDD Explorations, Volume 14, Issue 2, 2012. p.37-48.

AMO, S. A. de. **Curso de data mining**. Universidade Federal de Uberlândia. 102 p. Agosto 2003.

ANH, D. T. T., DATTA, A. **Streamforce: Outsourcing Access Control Enforcement for Stream Data to the Clouds**. CODASPY'14, March 3–5, 2014, San Antonio, Texas, USA. 2014.

APACHE Hadoop. Disponível em: <http://hadoop.apache.org/>. Acessado em: março de 2014.

ARASU, A. et al. **STREAM -- The Stanford Data Stream Management System**, Technical Report, Stanford InfoLab, 2004.

ASTROVA, I.; KOSCHEL, A.; SCHAAF, M. **Automatic Scaling of Complex-Event Processing Applications in Eucalyptus**. 2012 IEEE 15th International Conference on Computational Science and Engineering.

BABCOCK, B. et al. **Models and issues in data stream systems**. In: P.G. Kolaitis (Ed.), Proceedings of the 21nd Symposium on Principles of Database Systems, pp. 1–16. ACM Press, New York, 2002.

BHARGAVI, R. et al. **Complex Event Processing for Object Tracking and Intrusion Detection in Wireless Sensor Networks**. 2010 11th Int. Conf. Control, Automation, Robotics and Vision Singapore, 7-10th December 2010

BIFET, A. et al. **DATA STREAM MINING A Practical Approach**. MOA - The University of Waikato, 2011. p. 6.

BOYD, D.; CRAWFORD, K. **Critical Questions for *Big Data***. Information, Communication and Society, 15(5):662–679, 2012.

CHEN, H. et al. **Mining frequent patterns in a varying-size sliding window of online transactional data streams**. Information Sciences, v.215, n.0, p. 15–36,2012.ISSN0020-0255. 2012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025512003349>.

CHEN, J., CHEN, C. **Design of Complex Event-Processing IDS in Internet of Things.** Sixth International Conference on Measuring Technology and Mechatronics Automation. 2014.

CHENG, H.; YAN, X.; HAN, J. Incspan: incremental mining of sequential patterns in large database. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. New York, NY, USA: ACM, 2004. (KDD '04), p. 527–532. ISBN 1-58113-888-1.

CHOI, P., KIM, H., HWANG, B. **A Sequential Pattern Mining Using Dynamic Weight in Stream Environment**. ICOIN 2014.

CUZZOCREA, A.; SONG, I. Y.; DAVIS, K. C. **Analytics over Large-Scale Multidimensional Data: The *Big Data* Revolution!** DOLAP'11, October 28, 2011, Glasgow, Scotland, UK. ACM 978-1-4503-0963-9/11/10.

DEAN, J., GHEMAWAT, S. ***MapReduce*: simplified data processing on large clusters**. OSDI 2004, pp.137-150.

DRUCKER, P. F. **Managing for the Future: The 1990s and Beyond**. Dutton Adult, 1992.

ELMASRI, R.; NAVATHE, S. B. Sistemas de Banco De Dados. 4a. ed. [S.l.]: Pearson Addison Wesley, 2005. 744 p. Título original: Fundamentais of database systems.

ESPER website. Disponível em: <http://www.espertech.com/>. Acessado em: março de 2016.

FAN, W.; BIFET, A. **Mining *Big Data*: Current Status, and Forecast to the Future**. SIGKDD Explorations Volume 14, Issue 2. 2012.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. **Knowledge discovery and data mining: Towards a unifying framework**. In: SIMOUDIS, E.; HAN, J.; FAYYAD, U. (Ed.). KDD-96 Conference Proceedings. [S.l.]: AAAI Press, 1996. p. 82-88.

GARTNER, http://www.gartner.com/it-glossary/big- data. Último acesso em 09/12/2015.

GAMA, J.; GABER, M. M. Learning from Data Streams Processing Techniques in Sensor Networks. ACM Classification: H.3, C.2, C.3, I.2 ISBN 978-3-540-73678-3 Springer Berlin Heidelberg New York. 2007.

GANTZ, J. F. **The Diverse and Exploding Digital Universe**. An IDC White Paper - sponsored by EMC. 2011. Disponível em: http://www.ifap.ru/library/book268.pdf. Acessado em: maio de 2016.

GOLAB, L.; ÖZSU, M. T. **Data Stream Management Issus - A Survey. School of Computer Science** - University of Waterloo, Canada. 2003. p. 1-18.

HAN, J.; KAMBER, M. **Data Mining Concepts and Techniques**. 2a. ed. [S.l.]: Diane Cerra, 2006. 743 p.

HAN, J.; PEI, H.; YIN Y. **Mining Frequent Patterns without Candidate Generation**. In: Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX). ACM Press, New York, NY, USA 2000.

HE, Y. et al. **RCFile: A Fast and Space-efficient Data Placement Structure in *MapReduce*-based Warehouse Systems**. ICDE 2011, pp. 1199-1208.

IBM. IBM *Big Data* and analytics platform. Disponível em: <http://www-01.ibm.com/software/data/bigdata/>. 2012. Acessado em: março de 2016.

JAEIN et al. **A Study on CEP Performance in Mobile Embedded System**. 9 78-1-4673-4828-7/12/$31.00 ©2012 2 IEEE. ICTC 2012.

JAYAN, K., RAJAN, A. K. **Preprocessor for Complex Event Processing System in Network Security**. Fourth International Conference on Advances in Computing and Communications. 2014.

JIANCONG, F.; YONGQUAN, L.; JIUHONG, R. **An evolutionary mining model in incremental data mining**. In: Natural Computation, 2009. ICNC '09. Fifth International
Conference on. [S.l.: s.n.], 2009. v. 3, p. 114 –118.

KALDEWEY T.; SHEKITA, E. J.; CLYDESDALE S. T. **Structured Data Processing on *MapReduce***. EDBT 2012, pp.15-25.

KAUSHAL, C., SINGH, H. **Comparative Study of Recent Sequential Pattern Mining Algorithms on Web Clickstream Data.** IEEE Power, Communication and Information Technology Conference (PCITC) Siksha 'O' Anusandhan University, Bhubaneswar, India. 2015.

KHAN, M. S. et al. **A Sliding Windows based Dual Support Framework for Discovering Emerging Trends from Temporal Data**. In: Bramer, M and Ellis, R and Petridis, M (Ed.). RESEARCH AND DEVELOPMENT IN INTELLIGENT SYSTEMS XXVI: INCORPORATING APPLICATIONS AND INNOVATIONS IN INTELLIGENT SYSTEMS XVII. [S.l.], 2010. p. 35–48. ISBN 978-1-84882-982-4. 29th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, ENGLAND, DEC 15-17, 2009.

LANEY, D. **3-D Data Management: Controlling Data Volume, Velocity and Variety**. META Group Research Note, February 6, 2001.

LAPES StArt. 2011. Disponível em <http://lapes.dc.ufscar.br/tools/start_tool>. Acessado em: março de 2016..

LIN, J. ***MapReduce* is Good Enough? If All You Have is a Hammer, Throw Away Everything That's Not a Nail!** CoRR, abs/1209.2191, 2012.

LIN, J.; RYABOY, D. **Scaling *Big Data* Mining Infrastructure: The Twitter Experience.** SIGKDD Explorations Volume 14, Issue 2. 2012. p. 6-19.

LU, H.; SETIONO, R.; LIU, H. Neurorule: **A connectionist approach to data mining**. In: DAYAL, U.; GRAY, P. M. D.; NISHIO, S. (Ed.). VLDB'95, Proceedings of

21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland. [S.l.]: Morgan Kaufmann, 1995. p. 478–489. Isbn 1-55860-379-4.

MABROUKEH, N. R.; EZEIFE, C. I. **Using domain ontology for semantic web usage mining and next page prediction**. In: Proceeding of the 18th ACM conference on Information and knowledge management. New York, NY, USA: ACM, 2009. (CIKM ’09), p. 1677–1680. ISBN 978-1-60558-512-3.

MORALES, G. F. ***Big Data*** **and the Web: Algorithms for Data Intensive Scalable Computing**. PhD Program in Computer Science and Engineering. IMT Institute for Advanced Studies - Lucca, Italy, 2012.

MORALES, G. F. **SAMOA: A Platform for Mining *Big Data* Streams**. WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil. ACM 978-1-4503-2038-2/13/05, 2013.

MUTHUKRISHNAN, S. **Data streams: algorithms and applications**. Now Publishers, 2005.

MUKHERJEE et al. **Shared disk *Big Data* analytics with Apache Hadoop**. High Performance Computing (HiPC), 2012. 978-1-4673-2371-0/12/$31.00 ©2012 IEEE.

NIRANJAN, U. et al. **Developing a dynamic web recommendation system based on incremental data mining**. In: Electronics Computer Technology (ICECT), 2011 3rd International Conference on. [S.l.: s.n.], 2011. v. 3, p. 247–252.

ÖLMEZOGULLARI, E; ARI, I. **Online Association Rule Mining over Fast Data**. 978-0-7695-5006-0/13 © 2013 IEEE. DOI 10.1109/BigData.Congress.2013.77.

PAI, M. et al. **Systematic reviews and meta-analyses: an illustrated, step-by-step guide**. The National Medical Journal of India, v. 17, n. 2, p. 86–95, 2004.

PEI, J. et al. **Prexspan: Mining sequential patterns e-ciently by prex-projected pattern growth**. In: ICDE '01: Proceedings of the 17th International Conference on Data Engineering. Washington, DC, USA: IEEE Computer Society, 2001. p. 215.

PEI, J.; HAN, J.; WANG, W. **Mining sequential patterns with constraints in large databases**. In: CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management. New York, NY, USA: ACM, 2002. p. 1825. ISBN 1-58113-492-4.

QIN, X. et al. **Beyond Simple Integration of RDBMS and *MapReduce* – Paving the Way toward a Unified System for *Big Data* Analytics: Vision and Progress.** 2012 Second International Conference on Cloud and Green Computing. 978-0-7695-4864-7/12 $26.00 © 2012 IEEE. p. 716-725.

RAJARAMAN, A.; ULLMAN, J. D. **Mining of Massive Datasets**, 2012. p. 112. Disponível em <http://infolab.stanford.edu/~ullman/mmds/booka.pdf>. Acessado em: fevereiro de 2014.

ROSS, D. T. Structured analysis: A language for communications ideas. IEEE Transactions on Software Engineering, IEEE Press, v.3, n. 1, p. 16 - 34, 1977. ISSN 0098-5589.

SILBERSCHATZ, A.; KORTH, H. F.; SUNDARSHAN, S. Sistema de Banco de Dados. 5. ed. [S.l.]: Daniel Vieira, 2006. 808 p. Tradução de Database system concepts, 5th ed.

SILVEIRA JUNIOR, C. R.; RIBEIRO, M. X.; SANTOS, M. T. P. **Stretchy Time Pattern Mining: A Deeper Analysis of Environment Sensor Data.** Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference, 2013. p. 468-473.

SILVEIRA Jr., C. R.; CARVALHO, D. C.; RIBEIRO, M. X. ; Santos, Marilde T. P. Incremental Mining of Frequent Sequences in Environmental Sensor Data. In: The 28th International FLAIRS Conference, 2013, Florida, USA. Annals of Florida Artificial Intelligence Research Society - FLAIRS 2015 v. 1, p. 452-456, 2015.

SINGH, S.; SINGH, N. ***Big Data* Analytics**. 2012 International Conference on Communication, Information & Computing Technology (ICCICT), Oct. 19-20, Mumbai, India. p. 1-4.

SRIKANT, R.; AGRAWAL, R. **Mining sequential patterns: Generalizations and performance improvements**. In: EDBT '96: Proceedings of the 5th International Conference on Extending Database Technology. London, UK: Springer-Verlag, 1996. p. 3–17.

WU, X. et al. **Data Mining with *Big Data*.** IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. 2013.

XU, C.; CHEN, Y.; BIE, R. **Sequential pattern mining in data streams using the weighted sliding window model**. In: Paralleland Distributed Systems (ICPADS), 2009 15th International Conference on. [S.l.: s.n.], 2009. p. 886 –890. ISSN 1521-9097.

ZIHAYAT, M., WU, C. W., AN, A., TSENG, V. S. **Mining High Utility Sequential Patterns from Evolving Data Streams**. ASE BD&SI 2015, October 07-09, 2015, Kaohsiung, Taiwan. 2015.