UNIVERSIDADE FEDERAL FLUMINENSE
DEPARTAMENTO CIÊNCIA DA COMPUTAÇÃO

PÓS-PROCESSAMENTO DE PADRÕES SEQUENCIAIS NA MINERAÇÃO DE DADOS

MONOGRAFIA DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Priscila de Jesus Bonvini
Rafael Santos de Abreu

Niterói
2010

Priscila de Jesus Bonvini
Rafael Santos de Abreu

PÓS-PROCESSAMENTO DE PADRÕES SEQUENCIAIS NA MINERAÇÃO DE DADOS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal Fluminense, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação sob a orientação do Professor Inhaúma Neves Ferraz.

Universidade Federal Fluminense

Niterói
2010

Priscila de Jesus Bonvini
Rafael Santos de Abreu

PÓS-PROCESSAMENTO DE PADRÕES SEQUENCIAIS NA MINERAÇÃO DE DADOS

Aprovada por:

_____

Prof. Inhaúma Neves Ferraz/IC-UFF

_____

Prof. Luiz Valter Brand Gomes/IC-UFF

_____

Profa. Regina Célia Paula Leal Toledo/IC-UFF

Niterói
2010

RESUMO

Atualmente, o grande volume de informações armazenadas em banco de dados de organizações vai além da capacidade dos tradicionais métodos de análise dos dados baseados em consultas, pois eles se tornaram insuficientes para analisar o conteúdo em relação a algum conhecimento implícito que pode estar contido na grande massa de dados. Devido a esse fato, a mineração de dados passou a ser um processo importante de pesquisa, pois ela provê técnicas e ferramentas capazes de nos auxiliar a analisar grandes quantidades de dados e extrair conhecimento encoberto pelos demais dados. Existem fontes de informação que geram dados de natureza intrinsecamente sequencial, ou seja, compostos de eventos discretos que possuem uma ordenação espacial ou temporal. Apesar de existirem diversos métodos eficientes para a mineração desse tipo de padrões, geralmente eles são restritos a problemas específicos e tem dificuldade em lidar com a grande diversidade dos dados, o que pode ocasionar a perda de algum conteúdo relevante na análise das informações envolvidas, já que os padrões sequenciais mais complexos são muitas vezes escondidos atrás de sequências. Sendo assim, no presente trabalho se propõe a implementação de processos de pós-processamento de Conjuntos de Padrões Sequenciais, obtidos de bases de dados nas quais seja conhecida a origem do domínio, com a finalidade de filtrar o referido conjunto.

Palavras-chave: Mineração de Dados, Padrões Sequenciais, Pós-processamento, Filtragem.

ABSTRACT

Nowadays, the large volume of information stored in organizations is beyond the capacity of traditional analysis methods of data based on queries because they become insufficient to analyze the content in the report to any implied knowledge quote that may be contained in the great mass of data. Due to this fact, the data mining has become an important research because it provides techniques and tools that help us analyze large amounts of data and extracting knowledge hidden by other data. There are sources of information which generate data intrinsically sequential, i.e., composed of discrete events that have a spatial or temporal assortment. Although there are several efficient methods for mining this type of patterns, they are often restricted to specific problems and have difficulty in dealing with a great diversity of data, which is lost in involved information analysis, as the most complex sequential patterns are often hidden behind sequences Thus, the present work intends to implement processes of post-processing of Standard Patterns Sets, obtained from databases in which it is known the origin of the domain in order to filter that set, giving us as a result a detailed analysis of the data.

Keywords: Data Mining, Sequential Patterns, Post-processing, Filtering.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

Com a grande quantidade de dados armazenados em bancos de dados e data warehouses, torna-se cada vez mais importante desenvolver ferramentas potentes para análise e extração de conhecimentos de interesse sobre os dados. Existem fontes de informação que geram dados de natureza essencialmente sequencial, ou seja, dados compostos por eventos que possuem uma ordenação espacial ou temporal. Mineração de Dados é um processo de inferência de conhecimento de grande quantidade de dados e um dos seus principais problemas é a descoberta de padrões que ocorrem frequentemente em dados sequenciais. Para isso, utiliza-se a técnica de mineração de padrões sequenciais.

A mineração de padrões sequenciais tem como objetivo encontrar subsequências frequentes em um banco de dados de sequências. A busca de padrões sequenciais ocorre sobre uma base de dados de transações, onde a mesma possui o identificador do seu proprietário, a informação de quando ela foi realizada e os itens que a compõem. Esta é uma tarefa desafiadora, já que essa busca pode exigir a análise de um número imenso de padrões de subsequências possíveis [1].

Entre as aplicações da mineração de padrões sequenciais estão: a análise do comportamento de clientes, os padrões de acesso à Web, a análise do processo de experimentos científicos, a previsão de desastres naturais, o tratamento de doenças, entre outros [2].

Muitos estudos ao longo dos últimos anos contribuíram para a mineração eficiente de padrões sequenciais ou outros padrões frequentes em dados relacionados com o tempo [1], levando ao desenvolvimento de diversos algoritmos,

nos quais a maioria visa evitar a geração de todas as combinações possíveis de sequências, a fim de trazer como resultado a informação relevante. Em geral, os algoritmos propostos para este problema podem ser categorizados em três classes: 1 - métodos baseados no Apriori [10], como o GSP [11] e o AprioriAll [8]; 2 - métodos que projetam e particionam recursivamente a base de dados, como o FreeSpan [1] e o PrefixSpan [1]; e 3 - métodos que exploram a base de dados em um formato verticalizado, como o SPADE [12] e o SPAM [1].

O algoritmo GSP (Generalized Sequential Patterns) [11] foi uma das primeiras estratégias propostas para solução deste problema, e será o algoritmo utilizado para mineração neste trabalho. Ele possui a seguinte classificação: abordagem de geração e teste de candidatos. Essa categoria é baseada diretamente na propriedade Apriori: se um padrão com k itens não é frequente, nenhum de seus super padrões com k+1 ou mais itens pode ser frequente. Dessa forma, assim como o algoritmo Apriori, o GSP realiza várias iterações a fim de encontrar todos os padrões sequenciais.

Apesar desse tipo de mineração estar sendo muito estudado, não é necessariamente fácil entender a sua utilização. Existem algumas questões que sempre são levantadas a respeito da mineração dos padrões sequenciais. Entre elas está o questionamento sobre qual seria a relação intrínseca entre padrões sequenciais, ou então a incerteza se existiria alguma outra relação de padrões que poderia ser descoberta baseada nos padrões sequências já encontrados. Estas questões apontam para alguns obstáculos dentro da mineração convencional de padrões sequenciais, os quais serviram de motivação para realização desse trabalho.

O intuito do mesmo é realizar através de um pós-processamento a filtragem de um conjunto de padrões sequenciais, ou seja, trazer como resultado as informações que realmente são interessantes e que podem estar “camufladas” por trás de outras informações.

1.1 PROBLEMA

Apesar de existirem diversos métodos eficientes para a mineração de padrões sequenciais, onde a informação temporal é embutida tanto na regra (quando A ocorrer, então B ocorrerá em certo intervalo de tempo), quanto no processo de mineração sob a forma de restrições temporais, é eles supõem que os dados de entrada sejam sequenciais de eventos discretos incluindo apenas informação de ordenação, normalmente o tempo. Porém, os eventos podem ser associados a outros atributos.

A maioria dos métodos de mineração tem dificuldade em tratar com esta multidimensionalidade dos dados, perdendo a informação adicional envolvida, ou seja, informações que poderiam ser de grande interesse e utilidade podem não ser encontradas. Há também outro problema, a maioria dos métodos existentes tem direcionamento restrito a problemas específicos não sendo adequados a diferentes tipos de dados sequenciais.

1.2 HIPÓTESE, OBJETO, OBJETIVO E LIMITAÇÕES

A hipótese no trabalho é de que dado um conjunto de padrões sequenciais, é possível que existam informações implícitas nos mesmos e que somente a análise do conjunto de padrões gerados pode não ser suficiente para a sua descoberta.

O objeto de estudo nesse trabalho são os conjuntos de padrões sequenciais gerados pelo minerador escolhido, no caso o GSP. Sendo assim, o objetivo geral deste trabalho é a implementação de processos de pós-processamento de Conjuntos de Padrões Sequenciais, obtidos de bases de Dados nas quais seja conhecida a ontologia do domínio, com a finalidade de filtrar/podar o referido conjunto. Isso se deve ao fato de que não é interessante para o usuário

receber uma quantidade imensa de informação quando o que geralmente se deseja é um pequeno número de informação que seja interessante.

Uma limitação que encontramos na realização do trabalho foi a de encontrar base de dados de padrões sequenciais que fossem de domínio público para ser utilizada na mineração.

1.3 METODOLOGIA DE PESQUISA

A metodologia de pesquisa nesse trabalho foi definida baseada no cronograma de atividades fornecido pelo orientador mostrado na tabela abaixo:

		Dias																	
Item	Descrição	20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360
1	Busca de Base de Dados de Padrões Sequenciais																		
2	Busca de Minerador de Padrões Sequenciais																		
3	Mineração de Padrões Sequenciais																		
4	Pesquisa bibliográfica																		
5	Busca de filtro de Pós-processamento																		
6	Filtragem de Pós Processamento																		
7	Elaboração da primeira versão da monografia																		
8	Redação das conclusões																		
9	Revisão da monografia																		

Tabela 1.1: Cronograma

Baseado no mesmo, inicialmente foi feito um levantamento de informações buscando bases de dados disponíveis na internet, e em seguida uma análise dos atributos e a relevância dos dados para o problema proposto. Além disso, foram realizadas pesquisas bibliográficas que permitiram que se tomasse

conhecimento de material relevante, tendo por base o que já havia sido publicado em relação ao tema. Dessa forma foi possível criar uma nova abordagem sobre o mesmo, chegando a conclusões que poderão servir de base para pesquisas futuras. As pesquisas abordaram os mineradores de padrões sequenciais, seus conceitos e também os filtros de pós-processamento de padrões sequenciais.

Para realização do trabalho, foi feito um planejamento das atividades que seriam realizadas, como a escolha do minerador a ser utilizado, e em seguida a realização de testes de mineração de dados para a compreensão do problema e do algoritmo, assim como a identificação do pré-processamento de dados que seria necessário.

Após o planejamento foi iniciada a execução dos planos de ação, onde se realizou o pré-processamento dos dados da base, para posterior mineração dos padrões sequenciais através do algoritmo escolhido, e a filtragem de pós-processamento.

Finalmente, com os resultados em mãos, foi realizada uma análise detalhada dos mesmos para tirada de conclusões, de acordo com os objetivos definidos para o trabalho em questão.

1.4 ORGANIZAÇÃO DO TRABALHO

Neste capítulo são apresentados o problema, a motivação, a proposta e estrutura deste trabalho.

No segundo capítulo são apresentados os conceitos fundamentais sobre mineração de dados, com ênfase para na tarefa de mineração de padrões sequenciais.

No terceiro capítulo serão explicados os algoritmos e as ferramentas utilizadas para o pré-processamento dos dados, para a realização da mineração de padrões sequenciais e para o pós-processamento dos mesmos.

No quarto capítulo serão explicados os experimentos realizados, a base de dados utilizada e os resultados obtidos.

No quinto capítulo são apresentadas as considerações resultantes com este trabalho e as contribuições futuras.

CAPÍTULO 2

MINERAÇÃO DE DADOS

Neste capítulo são apresentados alguns dos principais conceitos sobre Mineração de Dados, e também sobre a técnica de mineração de padrões sequenciais. A definição formal do problema de padrões sequenciais é apresentada na seção 2.2.

2.1 CONCEITOS

A Mineração de Dados é uma etapa do processo de Descoberta de Conhecimento em Base de Dados que consiste me efetuar análise nos dados através da execução de algoritmos que, sob limitações aceitáveis de eficiência computacional, produzem uma enumeração de padrões sobre um conjunto de dados [4].

O termo Mineração de Dados é considerado por algumas pessoas como sinônimo de Knowledge Discovery in Databases (KDD) ou Descoberta de Conhecimento em Banco de Dados (DCBD). Porém, KDD é um processo mais amplo que consiste nas seguintes etapas, como ilustrado na Figura 2.1 abaixo:

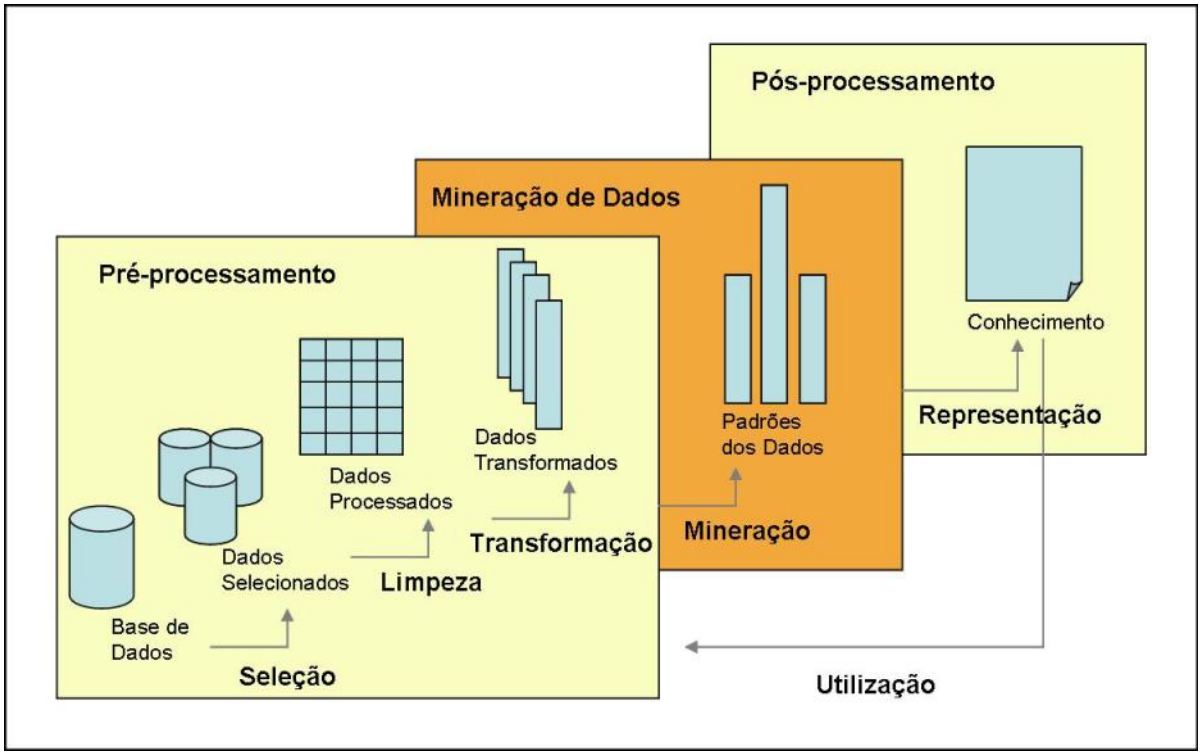


Figura 2.1: As etapas do processo KDD

Fonte: FAYYAD, 1996. P. 10 com adaptações.

- Na primeira etapa, a **seleção**, o objetivo é selecionar os dados importantes e identificar o objetivo do KDD. Para isso, é necessário o entendimento do problema através da compreensão do domínio da aplicação e do conhecimento prévio relevante.
- Na segunda etapa, a **limpeza**, são eliminados os ruídos e dados inconsistentes.
- Na terceira etapa, a **transformação,** os dados são transformados em um formato apropriado para aplicação de algoritmos de mineração.
- A quarta etapa, a **mineração de dados**, é essencial ao processo, consistindo na aplicação de técnicas inteligentes a fim de se extrair os padrões de interesse.
- A quinta etapa, a **representação**, consiste na representação dos padrões descobertos através de um modelo de conhecimento, como gráficos, a fim de apresentar ao usuário o conhecimento minerado. Esse passo envolve a interpretação dos padrões descobertos.

- Finalmente, a sexta etapa, a **utilização**, é onde se tem a utilização do conhecimento. Nela, o conhecimento é incorporado a fim de melhorar o seu desempenho, ou simplesmente os seus interesses são comprovados e documentados.

A Mineração de Dados se caracteriza por ser um conjunto de técnicas que envolvem métodos matemáticos, algoritmos e heurística para descobrir padrões e regularidades em grandes conjuntos de dados. Ela se preocupa em ajustar modelos ou determinar padrões a partir dos dados observados, e pode ser vista como uma forma de selecionar, explorar e modelar grandes conjuntos de dados para detectar padrões de comportamento. Os padrões ajustados representam o conhecimento inferido, o que a torna uma poderosa ferramenta de auxílio à tomada de decisão.

O objetivo da mineração de dados a partir de um banco de dados é gerar uma organização autônoma de aprendizado que faça o uso da informação gerada de forma ótima [5].

É importante distinguir o que é uma tarefa e o que é uma técnica de mineração. A tarefa consiste na especificação do que se quer buscar nos dados, que tipo de regularidades ou categoria de padrões tem-se interesse em encontrar, ou que tipo de padrões poderia surpreender. A técnica de mineração consiste na especificação de métodos que garantam como descobrir os padrões que interessam.

Uma das tarefas da Mineração de Dados é a análise de padrões sequenciais, que é o processo que identifica ocorrências frequentes ordenadas por evento, ou subsequências de padrões nos dados em uma base de dados. Uma sequência consiste de subsequências de elementos, ou eventos ordenados, armazenados com ou sem uma noção concreta de tempo [7].

A mineração de padrões sequenciais consiste na descoberta de subsequências frequentes como padrões, em uma base de sequências. Esse é um importante problema da mineração de dados com uma vasta aplicabilidade [1]. São exemplos dessas aplicações: uso da bioinformática na identificação de padrões em sequências de DNA, auxílio na identificação de sintomas para ajudar no diagnóstico

médico, além das bases de domínio comercial onde a extração de padrões sequenciais é de grande utilidade em campanhas de marketing. Existe também o uso de estratégias para conquistar a fidelidade do cliente, como a organização de prateleiras de supermercados de acordo com os padrões encontrados nas compras, as campanhas promocionais entre várias outras aplicações [7].

2.2 PADRÕES SEQUENCIAIS NA MINERAÇÃO DE DADOS

Pode-se caracterizar um padrão sequencial por eventos que se sucedem no tempo e que podem ser utilizados para prever um evento futuro baseados nos anteriores. Por exemplo, um padrão sequencial que poderia ser retirado de uma base de dados de uma loja seria o seguinte: um cliente que comprou um computador há cinco meses atrás, é um potencial comprador de uma impressora dentro de dois meses. A representação desse padrão seria <{computador}, {impressora}>.

A partir de padrões seqüenciais é possível obter regras de associação temporal, por exemplo, na forma $X \xrightarrow{T} Y$, que indica que se X ocorrer então Y ocorrerá dentro do período de tempo T [6]. Regras de associação temporal são úteis para entender o comportamento de entidades de um domínio e para prever o impacto da ocorrência de certos eventos quando ocorre(m) outro(s) evento(s).

A busca de padrões sequenciais ocorre sobre uma base de dados de transações, onde cada transação possui o identificador do seu cliente, os itens que a caracterizam e a informação de quando ela foi realizada. Em uma analogia com a base de dados de transações de um supermercado, a identificação do cliente poderia ser o número do seu cartão de crédito, os itens seriam os artigos que foram adquiridos pelo cliente e o momento de realização da transação seria a data e hora da compra.

Para exemplificar um problema de busca de padrões sequenciais será usado aqui o modelo de um supermercado, onde a gerência do supermercado estaria interessada em saber a evolução das compras dos clientes para fazer campanhas promocionais. Dessa forma, ao descobrir que uma sequência de produtos <P1; P2; P3> é frequentemente comprada nesta ordem pelos clientes, o supermercado poderia oferecer promocionalmente o produto P3 ou P1 aos clientes que compram o produto P2, evitando desperdícios com campanhas de marketing.

Para descobrir tais sequências, é necessário que exista um banco de dados de transações dos clientes, onde os mesmos sejam identificados para que se possam identificar as sequências de produtos que aparecem sucessivamente nas faturas de um mesmo cliente. Também é necessário que as transações do cliente sejam datadas. A Tabela 2.1 abaixo representa uma base de dados de transações de um supermercado, onde cada registro possui um identificador do cliente (IdCliente), a lista de itens comprados (Itemsets) e a data da compra.

IdCliente	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
2	{Dvd player, Fax}	23/04/2002
3	{ Dvd Player, liquidificador}	28/04/2002

Tabela 2.1 : Um exemplo de base de dados de transações

Como o interesse é minerar as sequências frequentes em um banco de dados de transações de clientes no formato da Tabela 2.1, algumas informações podem ser eliminadas, como a data em que as compras foram feitas. O mais importante é saber a ordem em que os produtos foram comprados. Dessa forma as transações podem ser agrupadas por cliente, formando uma base de dados de sequências do cliente, em que cada sequência é formada pela lista ordenada de transações de um determinado cliente. Um exemplo de base de sequências de

clientes, referente à base de dados transacional do exemplo anterior, está apresentado na Tabela 2.2 abaixo:

IdCliente	Sequências de itemsets
1	{tv, ferro elétrico}, {sapato, lençol}, {biscoito, açúcar}
2	{sapato, aparelho de som, tv}, {lençol, vídeo}, {Dvd Player, Fax}
3	{aparelho de som, tv, ventilador}, {vídeo, fitas de vídeo}, {Dvd Player, liquidificador}

Tabela 2.2: Um exemplo de base de dados de sequências de clientes

Uma sequência é uma lista ordenada de itemsets, sendo que cada itemset é um conjunto não vazio de itens. Denota-se uma sequência s por <s1s2...sn>, onde sj é um itemset. Um elemento de uma sequência s1, por exemplo, é definido por (x1, x2, ..., xm), onde xj é um item. Um item pode ocorrer apenas uma vez em um elemento de uma sequência, porém, podem ocorrer múltiplas vezes em elementos diferentes. Um itemset pode ser considerado como uma sequência com um único elemento. Considera-se que itens em um elemento de uma sequência estão em ordem lexicográfica [9].

Um exemplo de sequência de cliente é a lista de transações do consumidor 1: <{tv, ferro elétrico}, {sapato, lençol}, {biscoito, açúcar}>, na qual cada transação representa um conjunto de itens adquiridos.

O número de instâncias de itens em uma sequência é denominado tamanho da sequência. Por exemplo, a sequência acima <{tv, ferro elétrico}, {sapato, lençol}, {biscoito, açúcar}> tem tamanho é 6. Uma sequência de tamanho k possui k tens, sendo que um determinado item pode aparecer inúmeras vezes na sequência, mas uma única vez por elemento da sequência.

Define-se que uma sequência s=<a1a2...an> é uma subsequência de outra sequência t=<b1b2...bm> se existir inteiros i1 < i2 <...< in tal que $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_n \subseteq b_{i_n}$. Por exemplo, a sequência <(3), (4,5), (8)> é uma subsequência de, <(7), (3,8), (9), (4,5,6), (8)> já que $(3) \subseteq (3,8)$, $(4,5) \subseteq (4,5,6)$ e

$(8) \subseteq (8)$. Porém, a sequência <(3), (5)> não é uma subsequência de <(3,5)> e vice-versa. Assim, a sequência t=<b1b2...bm> é chamada super sequência de s=<a1a2...an>, ou seja, s está contida em t.

Um banco de dados de sequências S é um conjunto de tuplas <(*id_s, s*)>, onde *id_s* é um identificador de sequência e "*s*" é uma sequência [4]. Uma tupla <(id_s, s)> contém uma sequência "α" se "α" é uma subsequência de "*s*".

Assim, padrões sequenciais são todas as sequências que estão contidas em um número mínimo preestabelecido de sequências de clientes, também chamado de suporte mínimo. Um cliente suporta uma sequência s se ela estiver contida na sua sequência.

O suporte de um padrão sequencial *s* com relação a um banco de dados de sequências de clientes *D* é definido como sendo a porcentagem de sequências de clientes que suportam *s*, isto é:

$$sup(s) = \frac{\text{Número de sequências de clientes que suportam } s}{\text{Número total de sequências de clientes}}$$

Um padrão sequencial é dito frequente se possuir o suporte maior ou igual ao suporte mínimo. Por exemplo, supondo que o suporte mínimo seja α=0.2. Então o padrão <{aparelho de som, tv}> é frequente, pois seu suporte de 0.22 (2/9) é maior do que 0.2, definido como suporte mínimo.

Define-se da seguinte forma o problema da mineração de padrões sequenciais:

- Entrada: são dados um banco de dados de sequências de clientes *D*, tendo um nível mínimo de suporte α.
- Saída: todos os padrões sequenciais frequentes com relação a *D* e α.

Dessa forma será estudado nesse projeto o Pós-processamento de Padrões Sequenciais, que será feito após o processo de mineração de padrões sequenciais, ou seja, o processo de identificação de todas as sequências que possuem suporte maior ou igual ao suporte mínimo, especificado pelo usuário.

CAPÍTULO 3

SEQUÊNCIA DE PROCESSAMENTO DOS DADOS

Neste capítulo serão apresentadas as etapas seguidas referentes ao processo KDD neste trabalho. Será descrito como foi feito o pré-processamento dos dados utilizados, o conceito de mineração de padrões sequenciais e a sua prática utilizando o algoritmo GSP, assim como a aplicação e entendimento do mesmo no nosso projeto. Também será explicado como foi feito o pós-processamento desses padrões minerados.

3.1 PRÉ-PROCESSAMENTO DOS DADOS

Seguindo o processo KDD, na etapa de pré-processamento foi selecionada a base de dados *NorthWind.mdb* ou *NWind.mdb* (detalhada na seção 4.1).

O minerador GSP que foi utilizado na etapa de mineração recebe como entrada um arquivo ou base no formato da Tabela 3.1 abaixo:

idCliente	idPedido	timeStamp	Nº de Itens	Lista<Itens>

Tabela 3.1 : Formato da Base de Dados de Entrada no GSP

Como a base de dados escolhida não possui esse padrão específico, houve a necessidade da limpeza e transformação dos dados. Para isso foi criado um programa que recebesse como entrada a base de dados e retornasse como saída um arquivo no formato descrito acima.

Foi desenvolvida uma pequena aplicação na linguagem Visual Basic 6.0 que teve como objetivo simplesmente ler e formatar os dados da base em questão. A tela inicial da aplicação é mostrada na Figura 3.1 a seguir:

Figura 3.1 : Tela Inicial do Formatador da Base Nwind

A tela inicial, como mostra a figura acima, disponibiliza os filtros *CustomerID* (carregado com todos os ids dos clientes), *OrderID* (carregado com todos os ids dos pedidos) e ProductID (carregado com todos os ids dos produtos). Para visualizar todos pedidos realizados contidos na base em questão, basta clicar em [Visualizar]. Ao optar por visualizar todos os dados, a aplicação simplesmente exibe uma lista contendo todos os pedidos com seus respectivos produtos como mostra a Figura 3.2 a seguir:

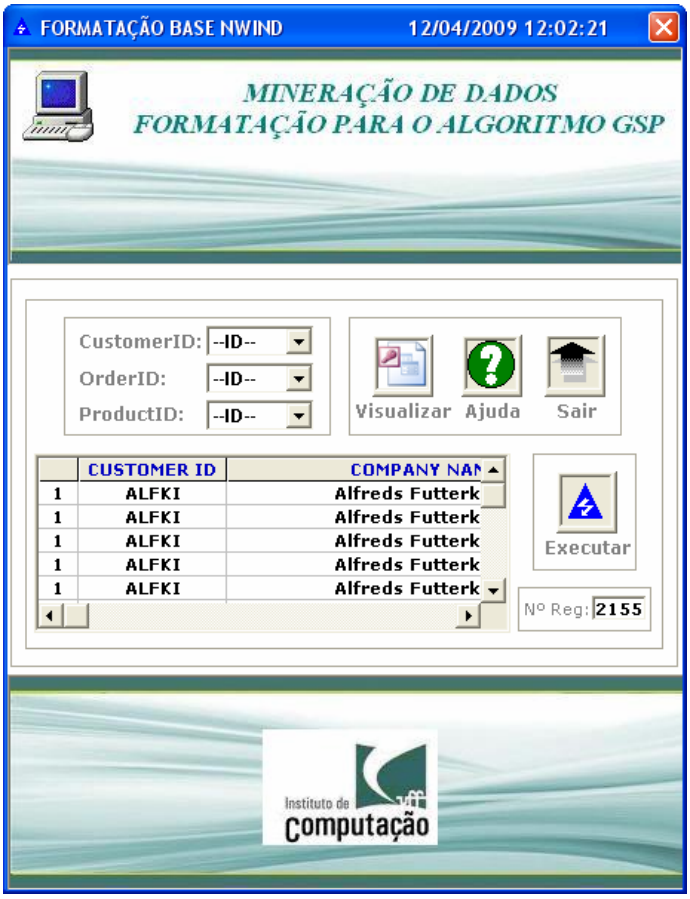


Figura 3.2 : Visualização dos Pedidos da Base NWind

Para uma melhor visualização dos pedidos, basta dar um duplo clique na lista. A aplicação então mostra claramente que os pedidos são exibidos separados por item de pedido ou produto, ou seja, um par < *Pedido* – *Produto* > por linha, de acordo com a Figura 3.3 abaixo:

Figura 3.3 : Visualização Expandida dos Pedidos da Base Nwind

Nota-se que o *CustomerID* é um campo do tipo *string* (conjunto de caracteres), sendo que o GSP o interpreta como um inteiro. Portanto, no momento da consulta ou visualização, a aplicação já associa automaticamente cada *CustomerID* com um inteiro, como já mostra a Figura 3.3 acima.

Após a consulta realizada, a aplicação disponibiliza o comando [Executar] (Figura 3.2) que tem como objetivo justamente agrupar os dados exibidos na lista por pedido e produto (*OrderID* e *ProductID*), de acordo com o formato aceito pelo GSP. Ao optar por executar a formatação, a aplicação realiza então o agrupamento dos dados e gera os mesmos em 2(dois) arquivos de saída diferentes:

- Um deles considerando apenas os ids dos clientes, os ids dos pedidos, os *timestamps*, o número de produtos no pedido em questão e os ids dos produtos, conforme o formato aceito pelo GSP (Figura 3.4)
- O outro com o nome dos clientes, os ids dos pedidos, os *timestamps*, o número de produtos no pedido em questão e os nomes dos produtos, simplesmente traduzindo o arquivo aceito pelo GSP para uma melhor visualização dos dados (Figura 3.5).

O *Timestamp* (definição contida mais a frente na seção 3.3.1) considerado em cada pedido ou transação, foi o número de dias dado pela a diferença entre a data do pedido (*Order Date*) e a data de embarcação (*Shipped Date*), caso existisse. Caso contrário, a data considerada seria a data requerida do pedido (*Required Date*).

Dessa forma, o valor do *Timestamp* é associado a cada um dos padrões gerados pelo GSP, resultando em eventos do tipo evento <ID-Tempo> no arquivo de saída gerado pelo programa.

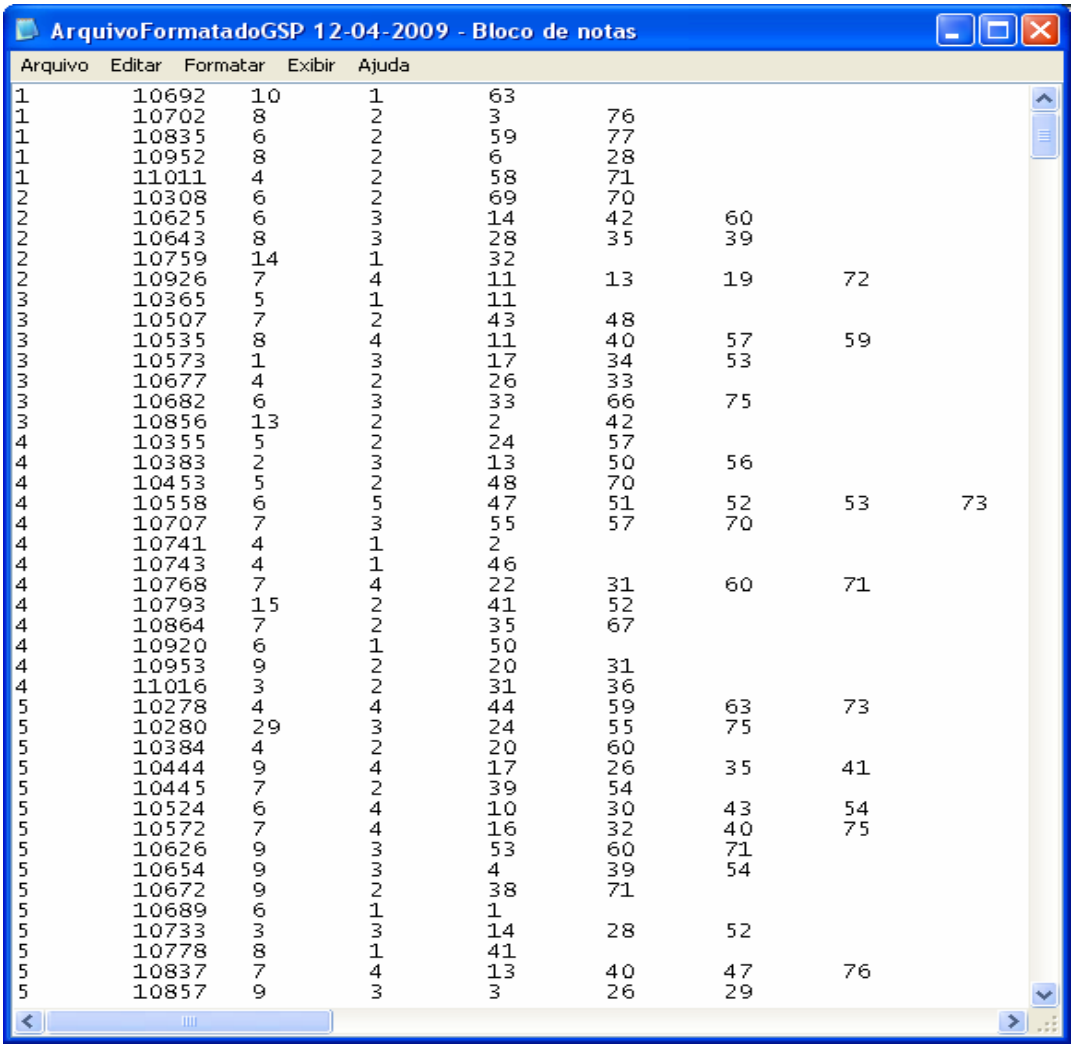


Figura 3.4 : Base Nwind Formatada para o GSP

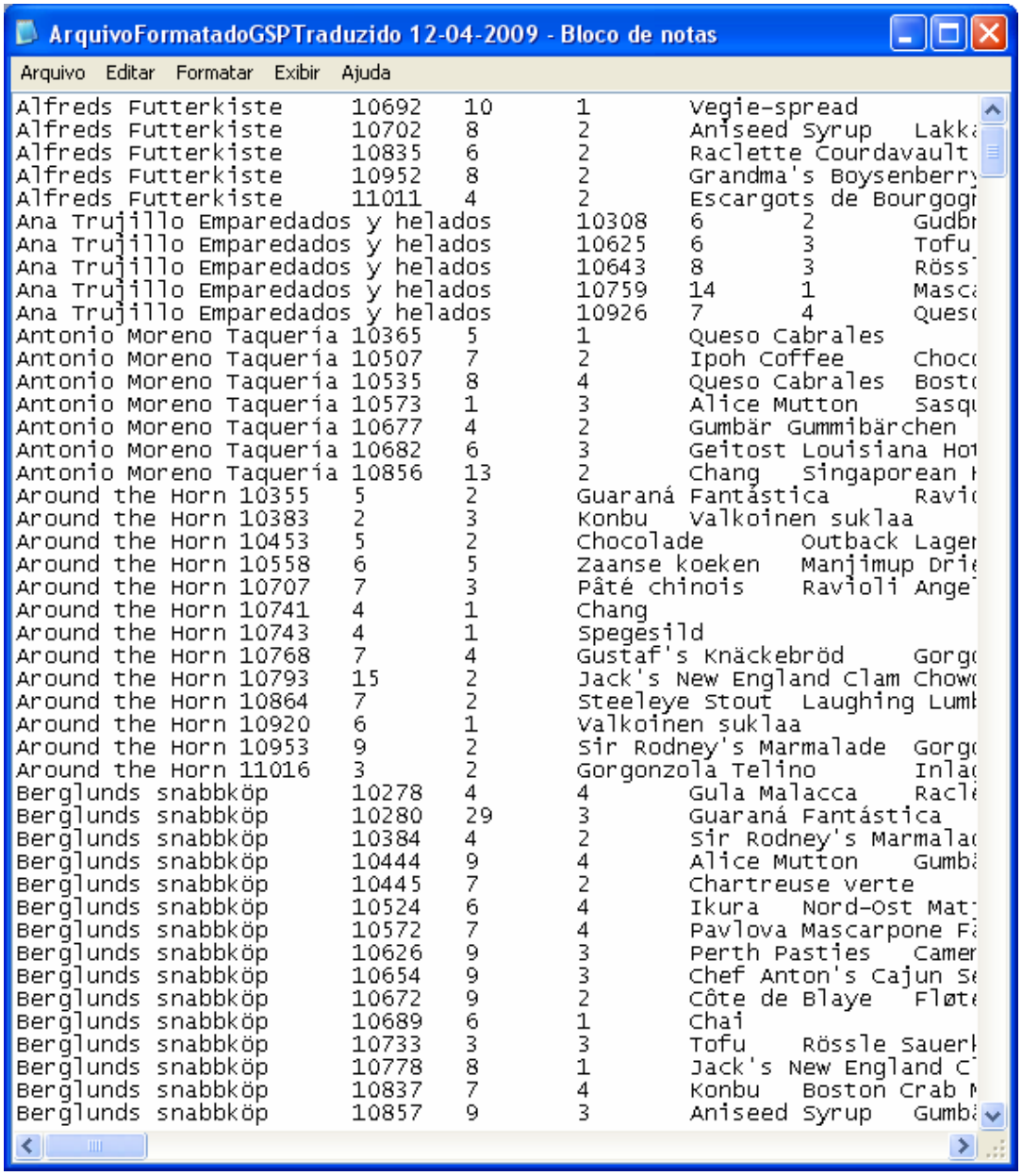


Figura 3.5 : Base Nwind Formatada e Traduzida para o GSP

3.2 MINERAÇÃO DE PADRÕES SEQUENCIAIS

O objetivo da mineração de padrões sequenciais é encontrar subsequências frequentes em um banco de dados de sequências. Trata-se de uma tarefa desafiadora, uma vez que a busca pode exigir o exame de um enorme número de padrões de subsequências possíveis. Essa técnica possui diversas aplicações, incluindo a análise do comportamento de clientes, padrões de acesso a Web,

análise do processo de experimentos científicos, previsão de desastres naturais, tratamento de doenças, teste de medicamentos, análise de DNA, entre outras coisas [2].

A maioria dos métodos desenvolvidos para a mineração de padrões sequenciais tem como finalidade evitar a geração de todas as combinações possíveis de sequências. Muitos estudos contribuíram para a mineração eficiente de padrões sequenciais ou outros padrões frequentes em dados relacionados com o tempo [1] e levaram ao desenvolvimento de diversos algoritmos, por exemplo, os de abordagem de geração e teste de candidatos como o GSP, que no caso foi minerador utilizado neste trabalho.

3.2.1 O ALGORITMO GSP

O GSP (Generalized Sequential Pattern ou Padrão Sequencial Generalizado) é um algoritmo para mineração de padrões sequenciais capaz de podar muitos candidatos na fase de podagem e, como resultado, leva para a fase de validação muito menos elementos a serem testados. A vantagem é que o algoritmo GSP elimina o máximo possível de candidatos que não são potencialmente frequentes, o que otimiza bastante o processo de mineração.

- Definição: uma k-sequência é uma sequência com *k* itens. Um item que aparece em itemsets distintos é contado uma vez para cada itemset onde ele aparece. Por exemplo: < {1, 4} >, < {1}, {4} > e < {1}, {1} > são 2-sequências [8].

No GSP, em cada iteração *k,* os conjuntos F_k e C_k são constituídos de sequências de *k* itens. Dessa forma, o algoritmo GSP gera as *k*-sequências frequentes (sequência com *k* itens) na iteração *k* (Figura 3.6 abaixo). Cada iteração é composta pelas fases de geração, de poda e de validação (cálculo do suporte).

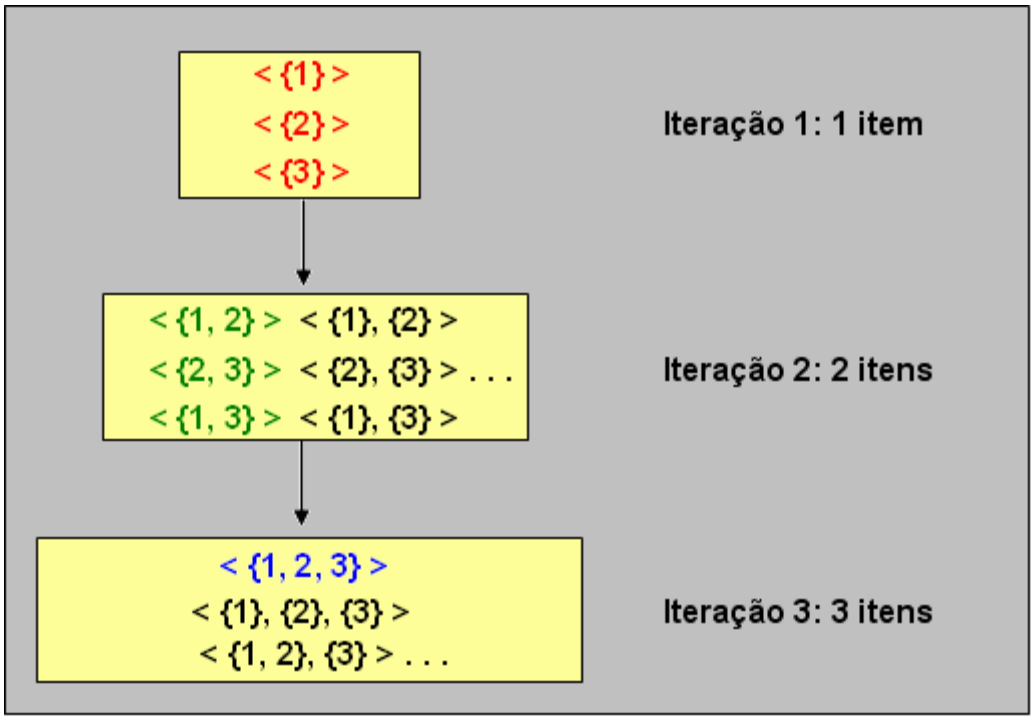


Figura 3.6: Geração Refinada de Candidatos no GSP

3.2.1.1 FASE DA GERAÇÃO DOS CANDIDATOS

- Caso $k \geq 3$

Suponhamos que F_{k-1} já tenha sido gerado na etapa $k - 1$. Duas sequências, $s = < s_1, s_2 ..., s_n >$ e $t = < t_1, t_2, ..., t_m >$ de F_{k-1} são ditas *ligáveis* se, retirando-se o primeiro item de s (s_1) e o último item de t (t_m) as sequências resultantes são iguais. Neste caso, s e t podem ser ligadas e produzir uma sequência r da seguinte maneira:

1. se t não é unitário: $r = < s_1, s_2, ..., s_n \cup t' >$, onde t' é o último item de t.
2. se t é unitário: $r = < s_1, s_2, ..., s_n, t >$.

A Figura 3.7 abaixo mostra o processo de junção de duas sequências de $k - 1$ itens com o objetivo de produzir uma sequência de k itens, ou seja, como

acrescentar um item a mais na primeira sequência de modo que o resultado tenha chance de ser frequente.

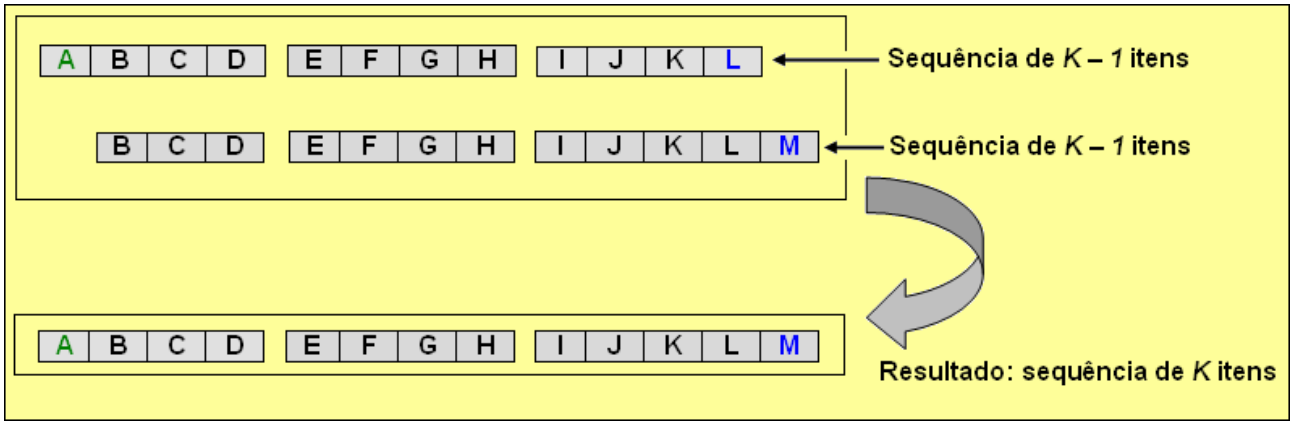


Figura 3.7: Junção de Sequências

Exemplo: Sejam s = < {**1**, 3}, {5}, {7, 9} > e t = < {3}, {5}, {7, 9, **11**} >. Retirando-se o primeiro item de s (o item **1**) e o último item de t (o item **11**) obtemos a mesma sequência < {3}, {5}, {7, 9} >. Logo, s e t são ligáveis e sua junção produz a sequência: s = < {**1**, 3}, {5}, {7, 9, **11**} >.

Definimos o conjunto dos pré-candidatos C'_k como sendo o conjunto obtido ligando-se todos os pares ligáveis de sequências de F_{k-1}. Veja o exemplo da Figura 3.8 abaixo:

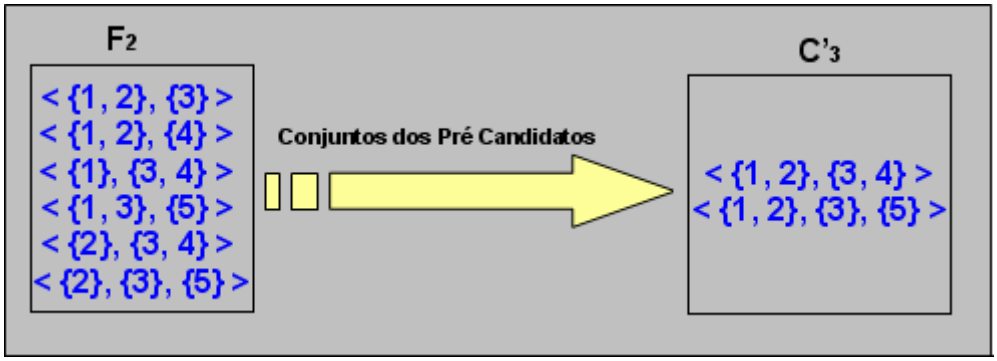


Figura 3.8: Conjunto dos Pré Candidatos C'3

Propriedade: a partir da junção de 2(duas) sequências s_1 e s_2, temos que, se eliminarmos o primeiro item do primeiro itemset da junção, será obtida a própria sequência s_2 [8].

- Caso $k = 2$

Para juntar duas sequências $s_1 = < 1 >$ e $s_2 = < 2 >$ de 1 item com o objetivo de produzir uma de 2(dois) itens, é necessário adicionar o item 2 de s_2 em s_1 tanto como parte do itemset {1} quanto como um itemset isolado. Portanto, a junção de s_1 com s_2 produz 2(duas) sequências de 2(dois) elementos: < {1}, {2} > e < {1, 2} >.

A propriedade acima mencionada se confirma para as 2(duas) sequências obtidas como resultado da junção de s_1 e s_2: nas 2(duas) sequências, ao eliminarmos o primeiro item do primeiro itemset, obtemos a sequência $s_2 = < \{2\} >$.

- Caso $k = 1$

O cálculo de C_1 considera todas as sequências de 1 item < {i} > e testa o suporte para cada uma delas. As que são frequentes constituem o conjunto F_1.

3.2.1.2 FASE DA PODA DOS CANDIDATOS

Suponhamos s uma k-sequência. Se s for frequente, então, pela Propriedade Apriori, sabemos que toda subsequência de s deve ser frequente. Suponhamos agora t uma subsequência qualquer obtida de s suprimindo-se 1(um) item de algum itemset. Se t não estiver em F_{k-1} então s não tem chance nenhuma de ser frequente e, portanto pode ser podada.

Exemplo: Consideremos o mesmo exemplo da Figura 3.5. A sequência < {1, 2}, {3}, {5}> será podada, uma vez que se retiramos o item 2 do primeiro itemset,

a sequência resultante < {1}, {3}, {5} > não está em F_2. Dessa forma, após a fase da poda, o conjunto C_3 resultante é { < {1, 2}, {3, 4} > }.

Propriedade: se uma *k*-sequência é frequente, ela tem necessariamente que estar presente em C_k.

3.2.1.3 FASE DA CONTAGEM DO SUPORTE

A cada iteração, cada sequência de cliente *c é* lida 1(uma) vez e incrementa-se o contador dos candidatos de C_k que estão contidos em *c*. Portanto, dado um conjunto C_k de sequências candidatas de uma sequência de um cliente c, precisamos encontrar todas as sequências em *C* que estão contidas em *c*. Dessa forma, 2(duas) técnicas são usadas para contornar este problema:

1. Usamos uma estrutura de árvore-hash para diminuir o número de candidatos de *C* que serão testados para o cliente *c*.

2. Transformamos a representação da sequência do cliente *c* de tal modo que possamos testar de forma eficiente se um determinado candidato de *C* é suportado (ou seja, está contido) em *c*.

Para poder realizar a mineração de padrões sequenciais da base de dados obtida, foi utilizado o algoritmo GSP implementado na linguagem Java. A ideia desse algoritmo consiste em utilizar uma árvore hash para armazenar as sequências candidatas. As folhas dessa árvore armazenam os conjuntos de padrões sequenciais (sequências de itemsets) e os nós intermediários (inclusive a raiz) armazenam tabelas-hash contendo pares do tipo (número, ponteiro). Para armazenar uma sequência *s* na árvore, o algoritmo aplica uma função hash a cada item da sequência.

Dessa forma o programa gera como saída um conjunto de padrões sequenciais, os quais finalmente são usados como entrada na etapa de pós-processamento, para que possam ser manipulados e filtrados.

3.3 PÓS-PROCESSAMENTO DE PADRÕES SEQUENCIAIS

A execução do algoritmo GSP para geração dos padrões sequenciais irá gerar um número muito grande de padrões e boa parte deles será de pouco valor. Para diminuir este número de padrões, utilizamos as restrições temporais no pós-processamento dos padrões gerados pelo GSP.

Antes de mostrar como foi implementada a etapa de pós-processamento nos dados, serão explicado alguns conceitos fundamentais sobre as restrições temporais, que são a base para a filtragem do conjunto de padrões sequenciais.

3.3.1 RESTRIÇÕES TEMPORAIS

As restrições temporais são condições impostas pelo usuário as quais os padrões sequenciais devem satisfazer a fim de serem filtrados. O programa utilizado para o pós-processamento dos dados foi implementado tendo por base estas restrições.

Dentre os parâmetros considerados, um deles, o *minsup* ou *suporte mínimo,* não se trata de uma restrição temporal. As restrições temporais são de fato explicadas abaixo:

- *Span*: intervalo de observação, ou seja, é o intervalo no qual está sendo estudado o comportamento das transações.

Exemplo: compra de um consumidor durante 60 dias na Tabela 3.2 abaixo.

Span = 60		
IdCliente	**Itemsets**	**Data**
1	{arroz, feijão, manteiga, macarrão, leite}	**10/02/2002**
1	{açúcar, arroz, biscoito}	17/02/2002
1	{ biscoito, macarrão, banana, maça}	20/02/2002
.	.	.
.	.	.
.	.	.
1	{arroz, feijão, açúcar, biscoito, manteiga}	02/04/2002
1	{açúcar, arroz, biscoito, leite, maça}	05/04/2002
1	{banana, maça, açúcar, manteiga}	**10/04/2002**

Tabela 3.2 : Um exemplo de Span aplicado em 60 dias de compras de um cliente

- *Mingap*: intervalo de tempo mínimo entre duas transações independentes.

  Exemplo: 2(duas) compras com um intervalo de 2(dois) dias sendo mingap igual a 1(um) dia são consideradas como fazendo parte de uma mesma transação (Tabela 3.3 abaixo).

Mingap = 1		
IdCliente	**Itemsets**	**Data**
1	{arroz, feijão, manteiga, macarrão, leite}	10/02/2002
1	{açúcar, arroz, biscoito}	11/02/2002

Tabela 3.3 : Um exemplo de Mingap aplicado em 2 dias de compras de um cliente

- *Maxgap*: duração máxima de duas transações consecutivas ou janelas mais o intervalo entre elas.

  Exemplo: 2(duas) compras de um cliente com um determinado intervalo de tempo (Tabela 3.4 abaixo).

Maxgap = 10		
IdCliente	**Itemsets**	**Data**
1	{arroz, feijão, manteiga, macarrão, leite}	10/02/2002
1	{açúcar, arroz, biscoito}	19/02/2002

Tabela 3.4 : Um exemplo de Maxgap aplicado em 2 dias de compras de um cliente

- *Janela ou time window:* é o intervalo de tempo durante o qual os eventos constituem uma transação.

  Exemplo: 4(quatro) compras alternadas de um cliente dentro de uma janela de 20 dias (Tabela 3.5).

Janela = 20		
IdCliente	**Itemsets**	**Data**
1	{arroz, feijão, manteiga, macarrão, leite}	10/02/2002
1	{açúcar, arroz, biscoito}	12/02/2002
1	{biscoito, macarrão, banana, maça}	14/02/2002
1	{biscoito, macarrão, banana, maça}	20/02/2002

Tabela 3.5 : Um exemplo de Janela aplicado em 4 compras de um cliente

Um intervalo de observação ou *span* é composto de diversas transações, ou seja, eventos que ocorrem dentro de uma janela e que ficam entremeados com eventos isolados. As restrições temporais consideradas podem ser resumidas da seguinte maneira (Figura 3.9):

- intervalo de observação (span);
- maior distância permitida entre o início de uma transação e o final da transação subseqüente (maxgap);
- menor distância permitida entre o final de uma transação e o início da transação subseqüente (mingap);
- maior distância permitida entre o início de uma transação e o final dessa transação (janela).

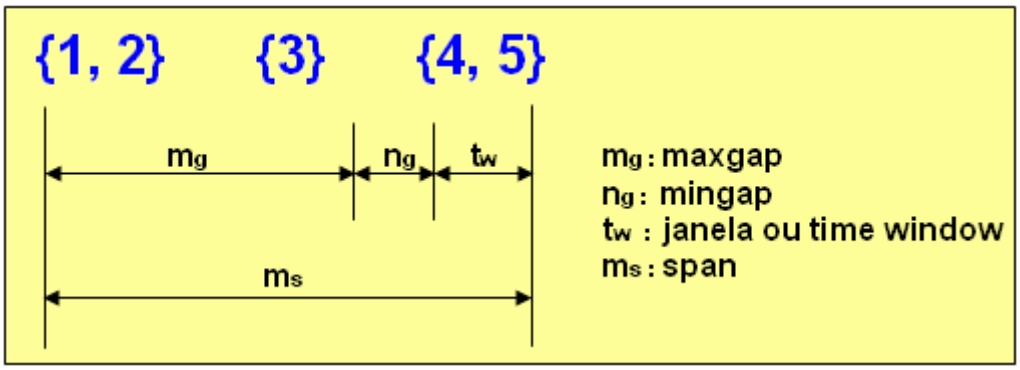


Figura 3.9: Restrições Temporais em Padrões Sequenciais

Para integrar todas essas restrições temporais, é necessário que o banco de dados minerado armazene também os tempos de cada itemset, visto que estes são essenciais no momento de verificar as restrições. Esse tempo é chamado de *Timestamp* (como mencionado na seção 3.1) que nada mais é do que o tempo no qual ocorreu o evento (marca de tempo).

3.3.2 O ALGORITMO

Para realização do pós-processamento do que foi minerado pelo GSP, construímos o programa *PosProcessGSP.*

O programa utilizou como base os algoritmos abaixo. Em sua implementação ele possui uma função principal chamada *RotulaTransa*, que chama as outras funções *VarreSpam* e *ProximoEvento*. *VarreSpan* chama *MarcaTransa* e *ProximoEvento*. Finalmente, *MarcaTransa* chama *ProximoEvento* e *Concatena*.

Em seguida serão descritos os algoritmos que serviram de base para a implementação das respectivas funções no *PosProcessGSP*.

O algoritmo *RotulaTransa* é responsável por resolver o problema da rotulação das transações de uma sequência de dados de acordo com restrições temporais especificadas. Ele recebe como parâmetros de entrada uma sequência de dados *D,* considerada padrão sequencial por um processo de mineração sem o uso

de restrições temporais, o parâmetro *span* ou intervalo de observação, o parâmetro *janela* ou intervalo de transação, os parâmetros *maxgap* e *mingap* caracterizando restrições temporais e o parâmetro *tamanho* identificando o menor número de eventos admitidos em um intervalo de observação.

Inicialmente é extraído o primeiro evento da sequência *D*. A seguir uma repetição infinita chama o procedimento *VarreSpan*, que percorre a sequência de dados de entrada aplicando as restrições temporais. Então, verifica-se se o número de eventos é suficiente comparando o tamanho do padrão sequencial até então obtido com o parâmetro *tamanho*, que especifica o menor número aceitável de eventos em um intervalo de observação. Caso o número obtido de eventos for satisfatório o algoritmo se encerra. Em caso contrário, sempre que o número de eventos descobertos em um intervalo de observação for menor do que o parâmetro *tamanho*, *despreza-se* o primeiro elemento do intervalo de observação corrente e busca-se outro intervalo de observação até atingir o patamar *tamanho* ou terminar a sequência de dados. Para tanto, a função *ProximoEvento* aplicada duas vezes a *R* busca o segundo item de *D.*

PróximoEvento é uma função que busca o próximo evento em uma sequência de dados que sucede ao evento corrente retornando false quando acabarem os eventos da sequência de dados. Quando aplicada pela primeira vez a uma sequência de dados retorna o primeiro item da sequência.

Como resultado, o algoritmo retorna o padrão sequencial *R* obtido pela aplicação das restrições temporais a *D*.

Algoritmo RotulaTransa

```
Require: <D,R,span,janela,maxgap,mingap,tamanho>
1: E = ProximoEvento(D)
2: tis := E.tempo
3: while true
4:     VarreSpan(E,D,R,span,janela,maxgap,mingap)
5:     if (R.size > tamanho)
6:         break
```

```
7:      else
8:          E = ProximoEvento(R) { primeiro item de R}
9:          E = ProximoEvento(R) {segundo item de R}
10:     end if
11: end while
12: return R
```

O algoritmo *VarreSpan* recebe como parâmetros de entrada um evento *E*, uma sequência de dados *D* considerada padrão sequencial por um processo de mineração sem o uso de restrições temporais, o parâmetro *span* ou intervalo de observação, o parâmetro *janela* ou intervalo de transação e os parâmetros *maxgap* e *mingap* caracterizando restrições temporais.

Inicialmente são estabelecidas as condições iniciais dos tempos do evento corrente, do início e fim da transação predecessora. Um ciclo infinito chama as funções *MarcaTransa* e *ProximoEvento*. *MarcaTransa* é uma função que extrai a próxima transação dentro das restrições temporais de *D* e a concatena a *R*. *PróximoEvento* é uma função que busca o próximo evento, em *D*, que sucede ao último evento concatenado a *R* retornando false quando acabarem os eventos da sequência de dados.

Como resultado o algoritmo retorna o padrão sequencial *R* obtido pela aplicação a *D* das restrições temporais.

Algoritmo VarreSpan

```
Require: <E,D,R,span,janela,maxgap,mingap, tis>
1: tec := E.tempo
2: tip := tec – mingap  {tec é o tempo de E}
3: tfp := tec – mingap
4: tic := E.tempo
5: tfc := tic
6: while (true)
7:    MarcaTransa(E, D, R, span, janela, maxgap, mingap, tec, tip, tfc, tfp, tis, tic)
```

8: E := ProximoEvento(D) *{ false quando acabarem os eventos}*
9: tec := E.Tempo
10: tip := tic
11: tfp := tfc
12: end while
13: return R

O algoritmo *MarcaTransa* recebe como parâmetros de entrada um conjunto *D* de padrões sequenciais, as restrições temporais *span*, *janela*, *maxgap* e *mingap*, o tamanho máximo, em número de itens que se pode considerar dentro de um span. Recebe também como parâmetros os timestamps *tec*, *tif*, *tfc* e *tfp* correspondendo ao evento corrente, início e final da transação predecessora e final da transação corrente.

Ele se inicia criando um “array” de itens e um contador de eventos em uma transação. A seguir o timestamp do primeiro evento é testado contra *mingap* para verificar se pode ocorrer o início de uma nova transação ou se o evento corrente é um evento isolado. Neste caso, é atribuído o valor 0 ao indicador de posição e o item é inserido no “array”. Se o timestamp permitir o início de nova transação, será disparada uma repetição na qual o item corrente é inserido no “array” e avança-se para o próximo evento de *D*. O processo se repetirá até se atingir o limite de janela ou de intervalo de observação.

O passo seguinte é o de rotulação, no qual são atribuídos os valores dos indicadores de posição aos itens, caracterizando se o item é isolado (0), se é início de uma sequência (1), meio de sequência (2) ou se é último de uma sequência (3). Finalmente, o “array” contendo a transação corrente ou evento isolado é concatenado ao padrão sequencial rotulado *R* que é, então, retornado.

Concatena é uma função que recebe uma sequência de dados e uma transação (ou evento isolado), retornando a sequência de entrada com a transação (ou o evento isolado) concatenada ao final da sequência.

O algoritmo retorna um conjunto *R* de padrões sequenciais rotulado, no qual as transações e eventos isolados que atendam as restrições temporais estejam identificados.

Algoritmo MarcaTransa

```
Require: <E,D,R,span,janela,maxgap,mingap,tec,tip,tfc,tfp >
1: New(A)  { criar um "array" de itens }
2: i:= 0
3: if (tec < tfp + mingap)
4:        E.ind = 0   {evento isolado}
5:        A[i++] = E  {Adicionar E ao "array" }
6:   else
7:       while ( (tec ≤ tic + janela) e (tec ≤ tis + span) e (tec ≤ tip + maxgap) )
8:            A[i++] = E    { Adicionar E ao "array" }
9:            E = ProximoEvento(D)
10:           if (E = null) break
11:           tec := E.tempo
10:      end while
11: end if
12: if (i > 1) {o número de elementos do "array" for maior do que 1}
13:     A[0].ind = 1   {início de transação}
14:     tic = A[0]..tempo
15:     A[i - 1].ind = 3   {final de transação}
16:     tfc = A[i - 1].tempo
17:      for j variando de 1 até i – 2  do   { meio de transação }
18:         if (A[j].ind <> 0)
19:             A[j].ind = 2
20:        end if
21:     end for
22:  else
23:     if I = 1
24:         A[0].ind = 0   {evento isolado}
25:     end if
```

```
26:  end if
27: Concatena(R, A)
28: return R
```

3.3.3 O FUNCIONAMENTO DO PROGRAMA

O programa *PosProcessGSP* foi desenvolvido na linguagem C Sharp (C#), utilizando o Visual Studio 2005. O objetivo principal foi implementar os métodos RotulaTransa, VarreSpan e MarcaTransa citados na seção anterior. A tela inicial da aplicação é mostrada na Figura 3.10 a seguir:

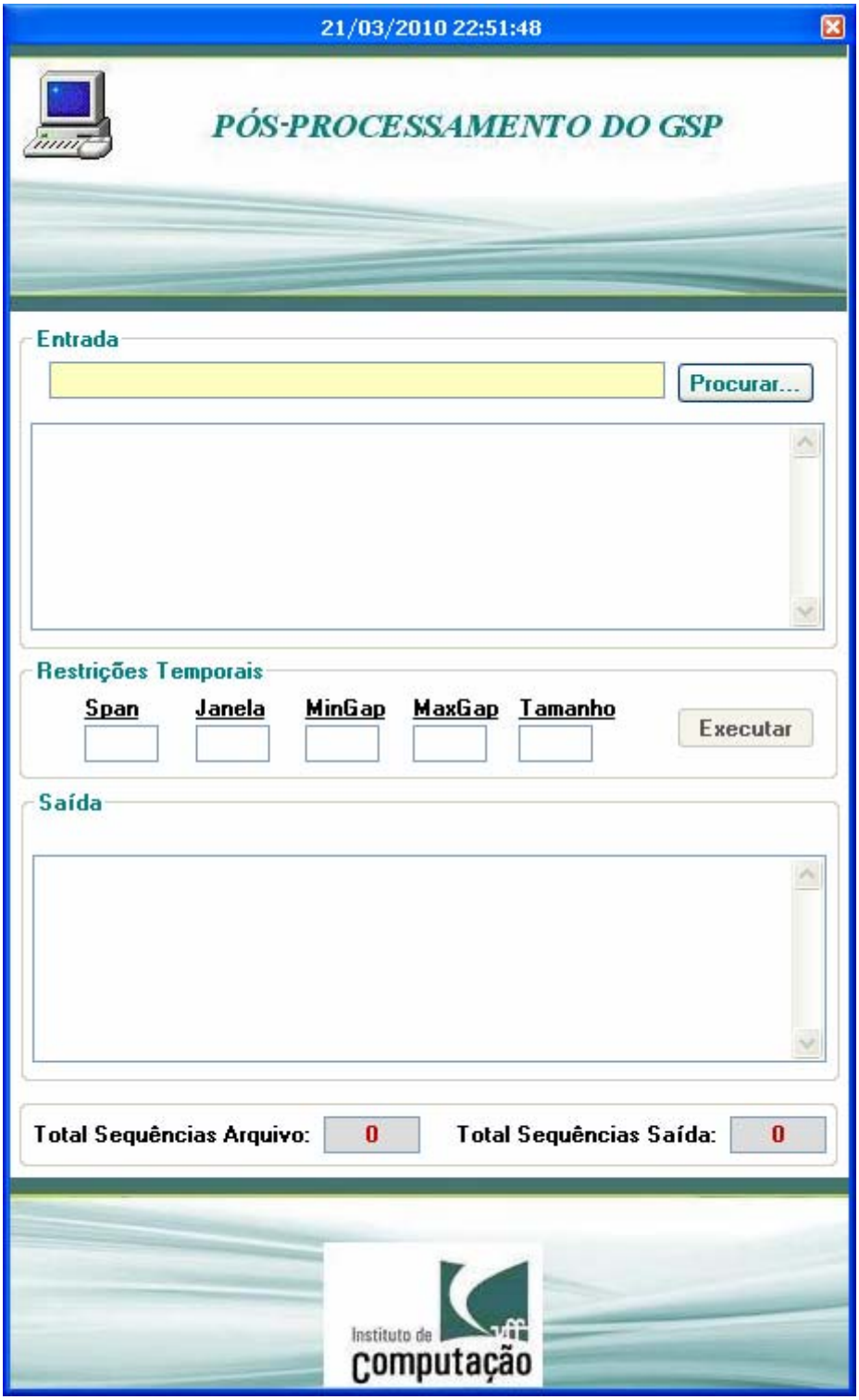


Figura 3.10 : Tela Inicial do Programa de Pós Processamento

A tela inicial, como mostra a figura acima, mostra as informações de entrada e de saída, além das restrições temporais. O botão [Procurar...] serve para buscar o arquivo de entrada com os padrões sequenciais minerados pelo GSP sem

as restrições temporais, mas com o valor do timestamp associado a cada evento contido no padrão sequencial gerado. Ao selecionar o arquivo de entrada, o programa exibe o diretório de onde se encontra o respectivo arquivo a ser processado. A tela a seguir mostra exatamente essa situação (Figura 3.11):

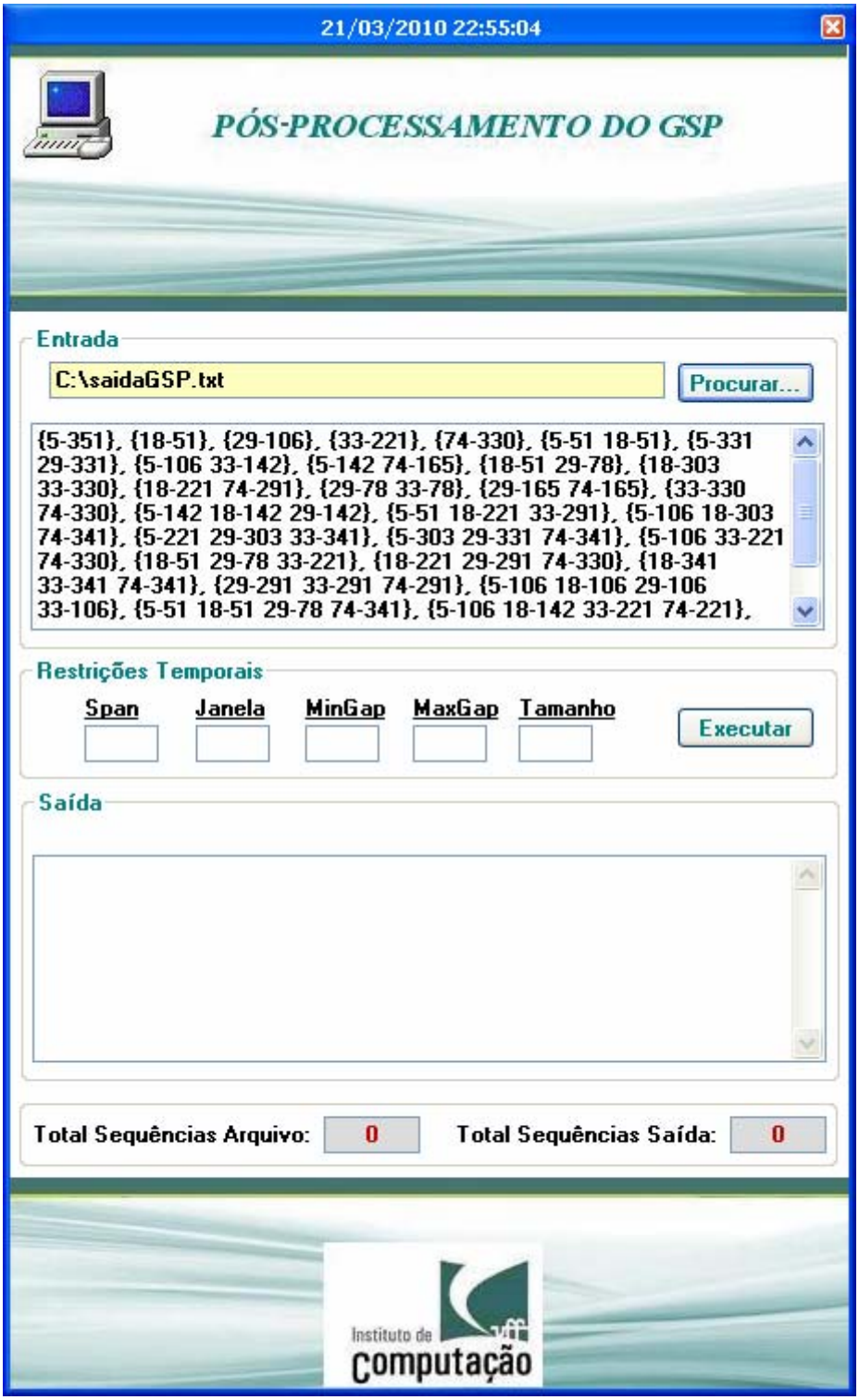


Figura 3.11 : Diretório do Arquivo de Entrada

Logo abaixo do diretório do arquivo de entrada, são exibidos os parâmetros de entrada do programa que representam as restrições temporais a serem utilizadas no pós-processamento: Span, Janela, Mingap, Maxgap e Tamanho. As definições desses parâmetros se encontram nas seção 3.3.1 e na seção 3.3.2.

O programa realiza as seguintes validações considerando os parâmetros de entrada:

- não podem ficar em branco (todos são obrigatórios)
- todos devem ser maiores do que 0 (zero)
- o parâmetro Mingap deve ser menor do que o MaxGap
- o parâmetro Span deve ser maior do que Janela, Mingap e Maxgap
- o parâmetro Maxgap deve ser maior ou igual a (Janela + Mingap) e menor do que (2 X Janela + Mingap)

Para mostrar a execução do programa, será utilizado como exemplo o seguinte arquivo de entrada contendo parte dos padrões minerados sem restrições temporais pelo GSP aplicado em nossa base de dados (Figura 3.12):

Figura 3.12 : Arquivo de Entrada utilizado como exemplo

Repare que cada evento presente no arquivo acima está associado à um tempo (ou timestamp), ou seja, cada evento é do tipo <ID-Tempo> (exemplo: {5-351}, sendo “5” o ID do evento e “351” o respectivo tempo).

Após informar todos os parâmetros, é necessário clicar no botão [Iniciar] para começar a execução do pós-processamento. O programa separa cada evento presente no arquivo de entrada e atribui ao mesmo uma estrutura de dados contendo os seguintes atributos (Tabela 3.6):

Atributo	Descrição
Item	“Id” do evento representado por um número inteiro
Tempo	timestamp ou marca de tempo associado ao evento
Ind	indicador de posição do evento dentro de uma transação

Tabela 3.6 : Estrutura de Dados representando um Evento

Usando como exemplo o padrão **{5-351}** presente no arquivo, o evento representado pelo número “5” terá um timestamp igual a “351” e terá o seu atributo “IND” inicializado com o valor “-1”.

Após atribuir a estrutura de dados para cada evento presente no arquivo, o programa realiza a classificação ou ordenação da lista de eventos pelo atributo “Tempo”.

Para poder atender às restrições temporais, um evento tem que atender a uma das seguintes condições:

- **TEC – TFP < MINGAP** → a diferença entre o tempo do evento corrente (TEC) e o tempo do fim da transação predecessora (TFP) tem que ser menor do que MINGAP. Nesse caso, o evento é caracterizado como um evento isolado (exemplos: {1}, {3}, etc).
- **(TEC – TIC <= JANELA) e (TEC – TIS <= SPAN) e (TEC – TIP <= MAXGAP)** → se o evento não atendeu a primeira condição, ou seja, a condição de ser um evento isolado, então o mesmo tem que ser verdadeiro nas seguintes condições, respectivamente:
    1. a diferença entre o tempo do evento corrente (TEC) e o tempo de início da transação corrente (TIC) tem que ser menor ou igual a JANELA;
    2. a diferença entre o tempo do evento corrente (TEC) e o tempo de início da seqüência (TIS) tem que ser menor ou igual ao SPAN;
    3. a diferença entre o tempo do evento corrente (TEC) e o tempo de início da transação predecessora (TIP) tem que ser menor ou igual a MAXGAP.

A Figura 3.13 abaixo mostra os detalhes das condições acima em função das restrições temporais, utilizando como evento corrente o item “**42**” contido na segunda janela:

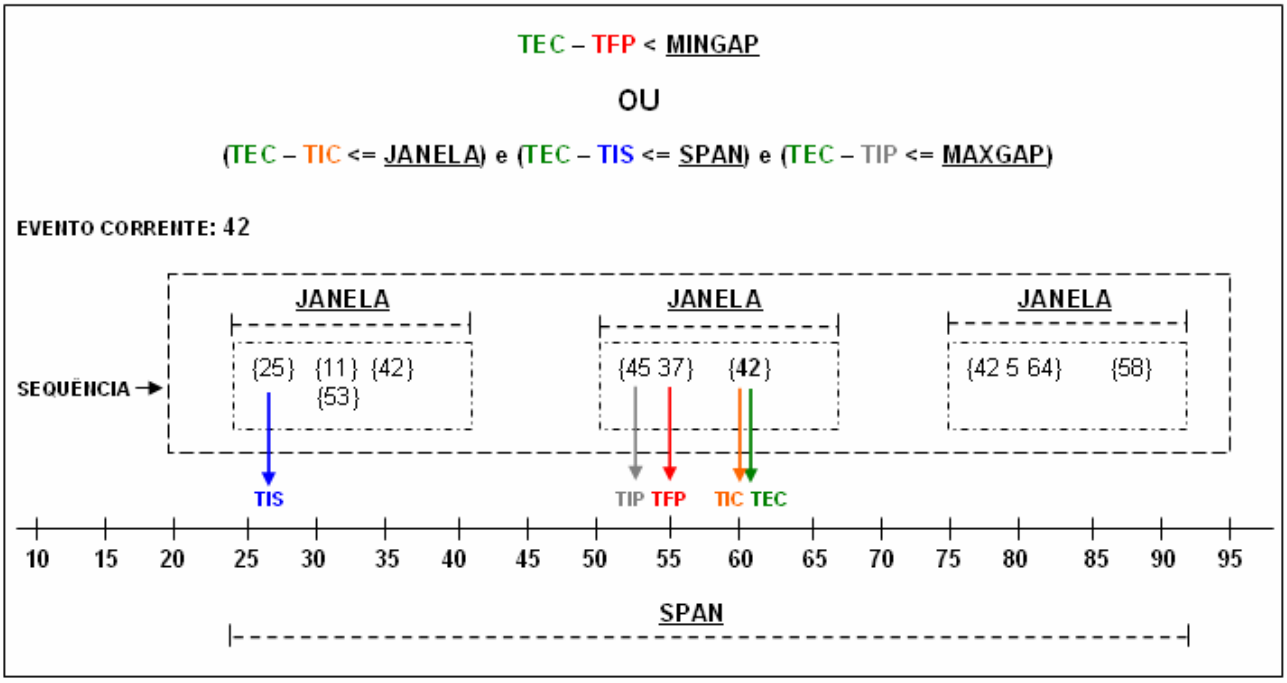


Figura 3.13 : Restrições Temporais

Caso o evento analisado não atender à nenhuma das duas condições acima, então o mesmo é descartado, uma vez que não atendeu às restrições temporais consideradas.

Ao finalizar a execução, o programa exibe uma mensagem informando o fim do processamento e o local onde se encontra a saída do programa com os resultados. Além disso, aplicação exibe todas as transações ou eventos isolados que atenderam às restrições temporais de entrada, o total de sequências contidas no arquivo de entrada e o total de sequências de saída. A Figura 3.14 abaixo mostra o final da execução do processamento do arquivo da Figura 3.12:

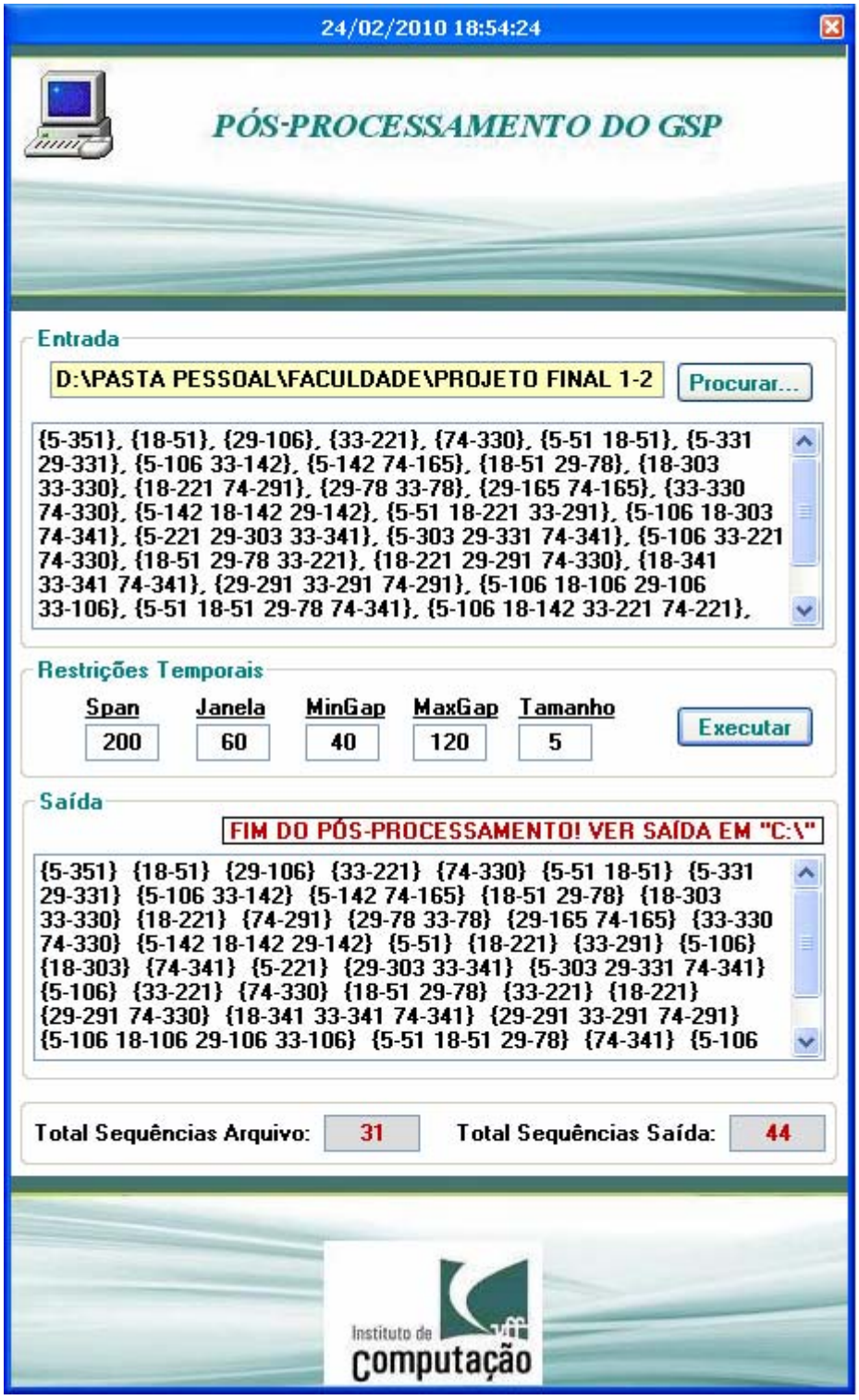
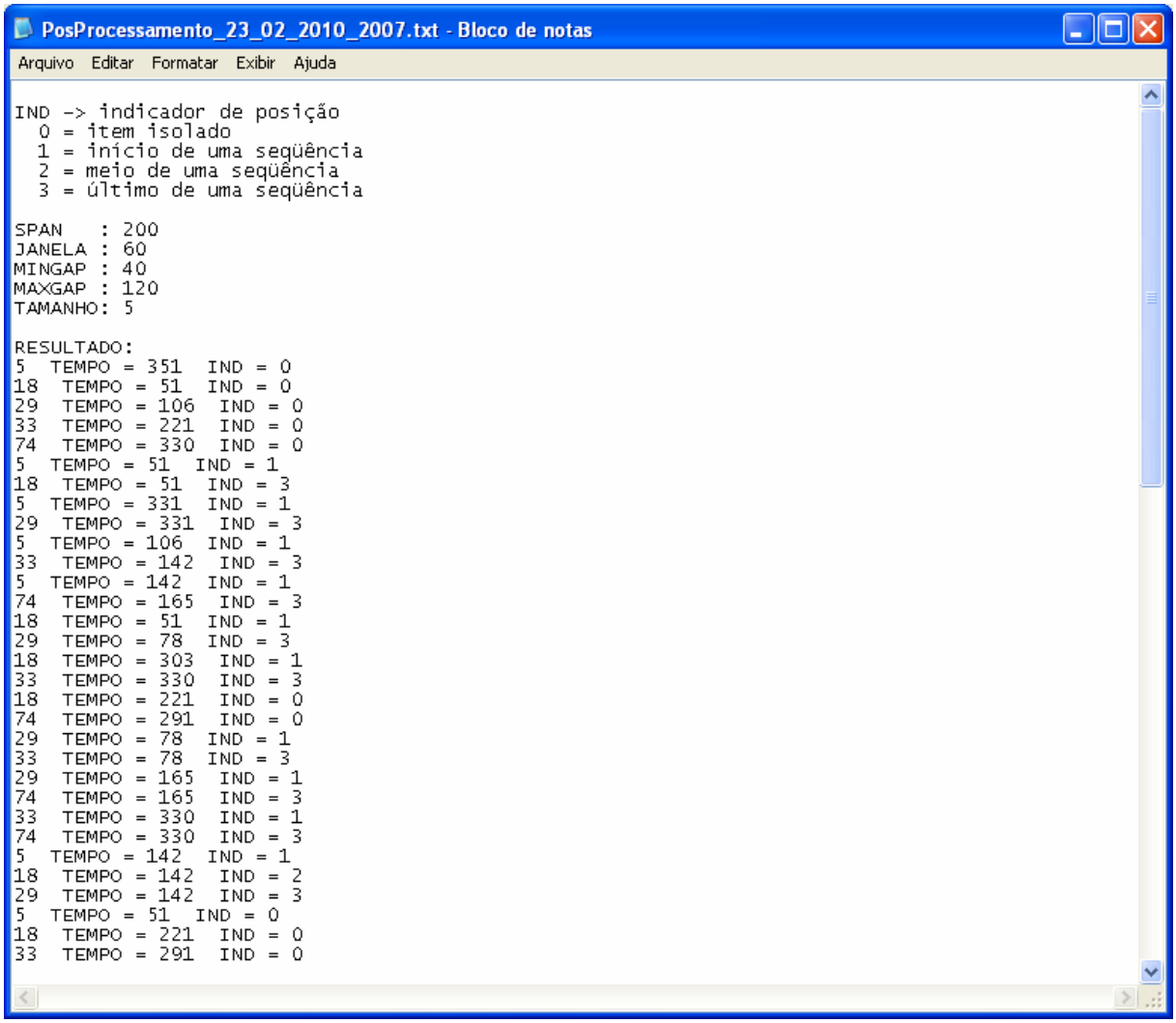


Figura 3.14 : Final do Pós Processamento

A figura acima indica que a saída do programa se encontra na raiz do drive “C:\”. O programa gera 2 (dois) arquivos de saída:

- PosProcessamento_[DD]_[MM]_[AAAA]_HHMM.txt: contém as informações mais simples do pós processamento, ou seja, uma breve explicação do atributo “IND” de um evento, os valores dos parâmetros utilizados, as informações de cada evento (“id” do evento, tempo ou timestamp e o valor da variável “IND”) e as estatísticas do resultado em questão (Figura 3.15)
- PosProcessamentoDetalhes_[DD]_[MM]_[AAAA]_HHMM.txt: contém as informações mais detalhadas do pós processamento, ou seja, uma breve explicação do significado dos parâmetros de entrada e das variáveis de tempo associadas aos eventos, os valores dos parâmetros utilizados e as sequências de entrada e de saída (Figura 3.16).



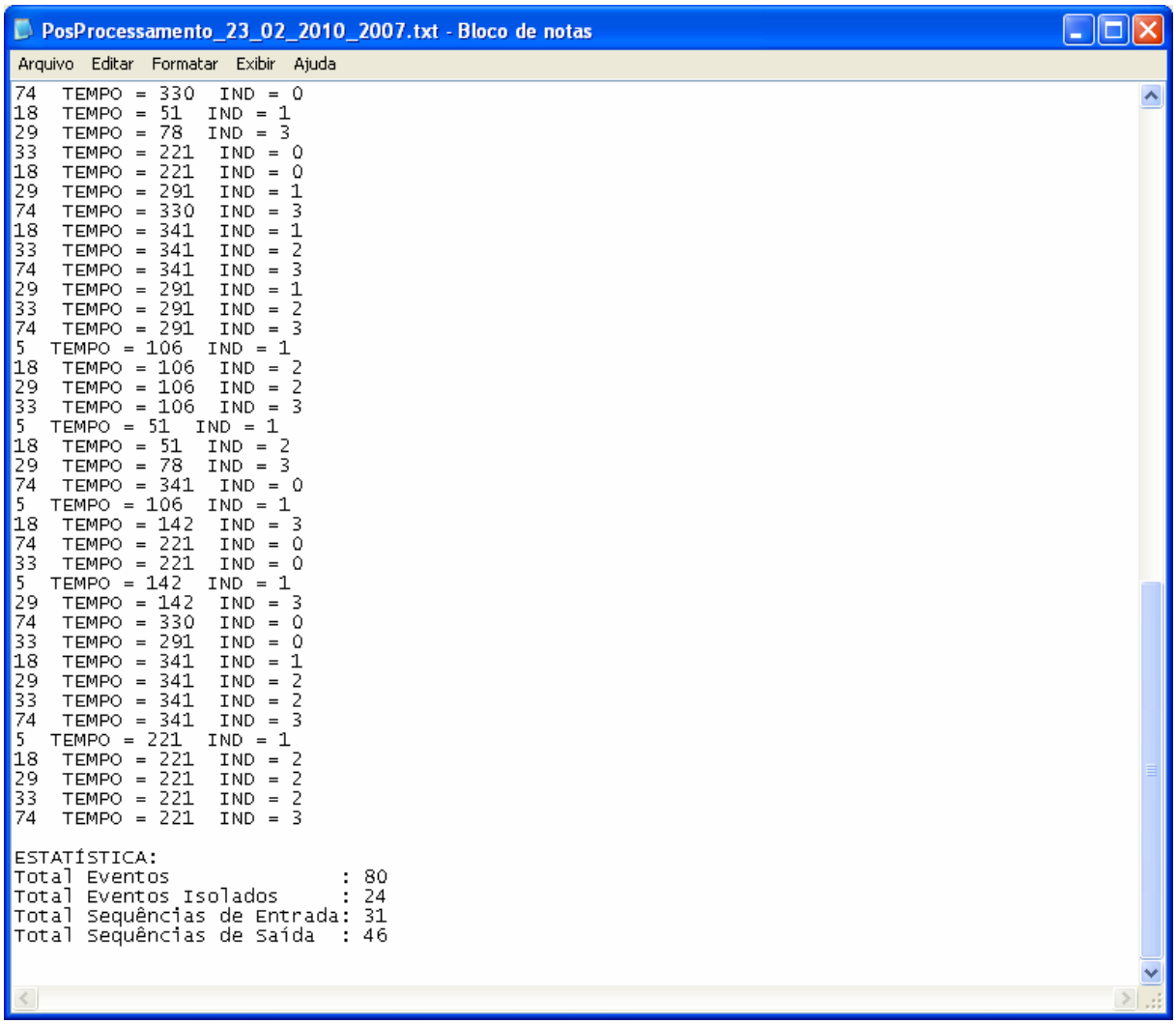
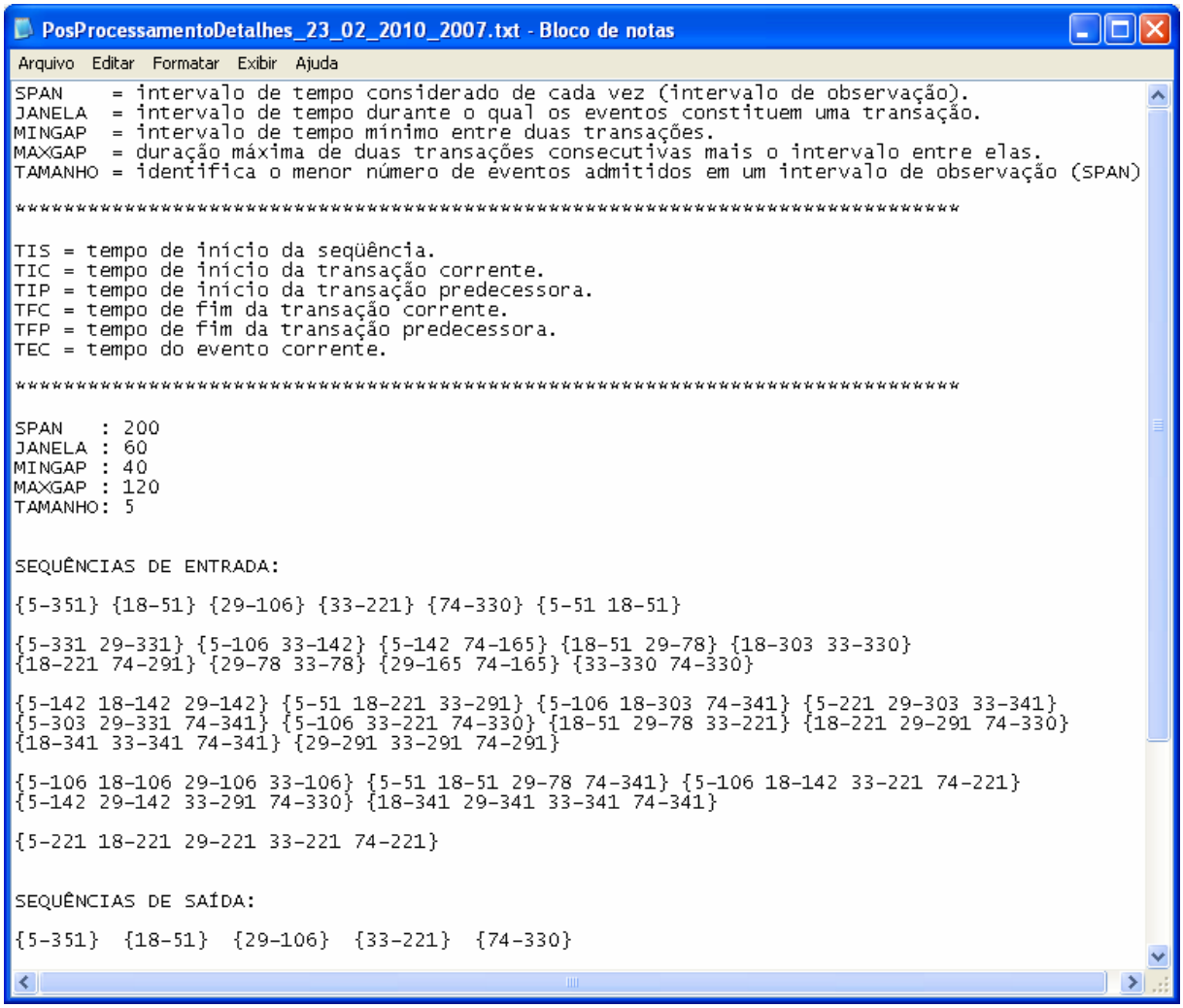


Figura 3.15 : Arquivo PosProcessamento_[DD]_[MM]_[AAAA]_HHMM



PosProcessamentoDetalhes_23_02_2010_2007.txt - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

```
SPAN    = intervalo de tempo considerado de cada vez (intervalo de observação).
JANELA  = intervalo de tempo durante o qual os eventos constituem uma transação.
MINGAP  = intervalo de tempo mínimo entre duas transações.
MAXGAP  = duração máxima de duas transações consecutivas mais o intervalo entre elas.
TAMANHO = identifica o menor número de eventos admitidos em um intervalo de observação (SPAN)

*******************************************************************************

TIS = tempo de início da seqüência.
TIC = tempo de início da transação corrente.
TIP = tempo de início da transação predecessora.
TFC = tempo de fim da transação corrente.
TFP = tempo de fim da transação predecessora.
TEC = tempo do evento corrente.

*******************************************************************************

SPAN   : 200
JANELA : 60
MINGAP : 40
MAXGAP : 120
TAMANHO: 5


SEQUÊNCIAS DE ENTRADA:

{5-351} {18-51} {29-106} {33-221} {74-330} {5-51 18-51}

{5-331 29-331} {5-106 33-142} {5-142 74-165} {18-51 29-78} {18-303 33-330}
{18-221 74-291} {29-78 33-78} {29-165 74-165} {33-330 74-330}

{5-142 18-142 29-142} {5-51 18-221 33-291} {5-106 18-303 74-341} {5-221 29-303 33-341}
{5-303 29-331 74-341} {5-106 33-221 74-330} {18-51 29-78 33-221} {18-221 29-291 74-330}
{18-341 33-341 74-341} {29-291 33-291 74-291}

{5-106 18-106 29-106 33-106} {5-51 18-51 29-78 74-341} {5-106 18-142 33-221 74-221}
{5-142 29-142 33-291 74-330} {18-341 29-341 33-341 74-341}

{5-221 18-221 29-221 33-221 74-221}


SEQUÊNCIAS DE SAÍDA:

{5-351}  {18-51}  {29-106}  {33-221}  {74-330}
```

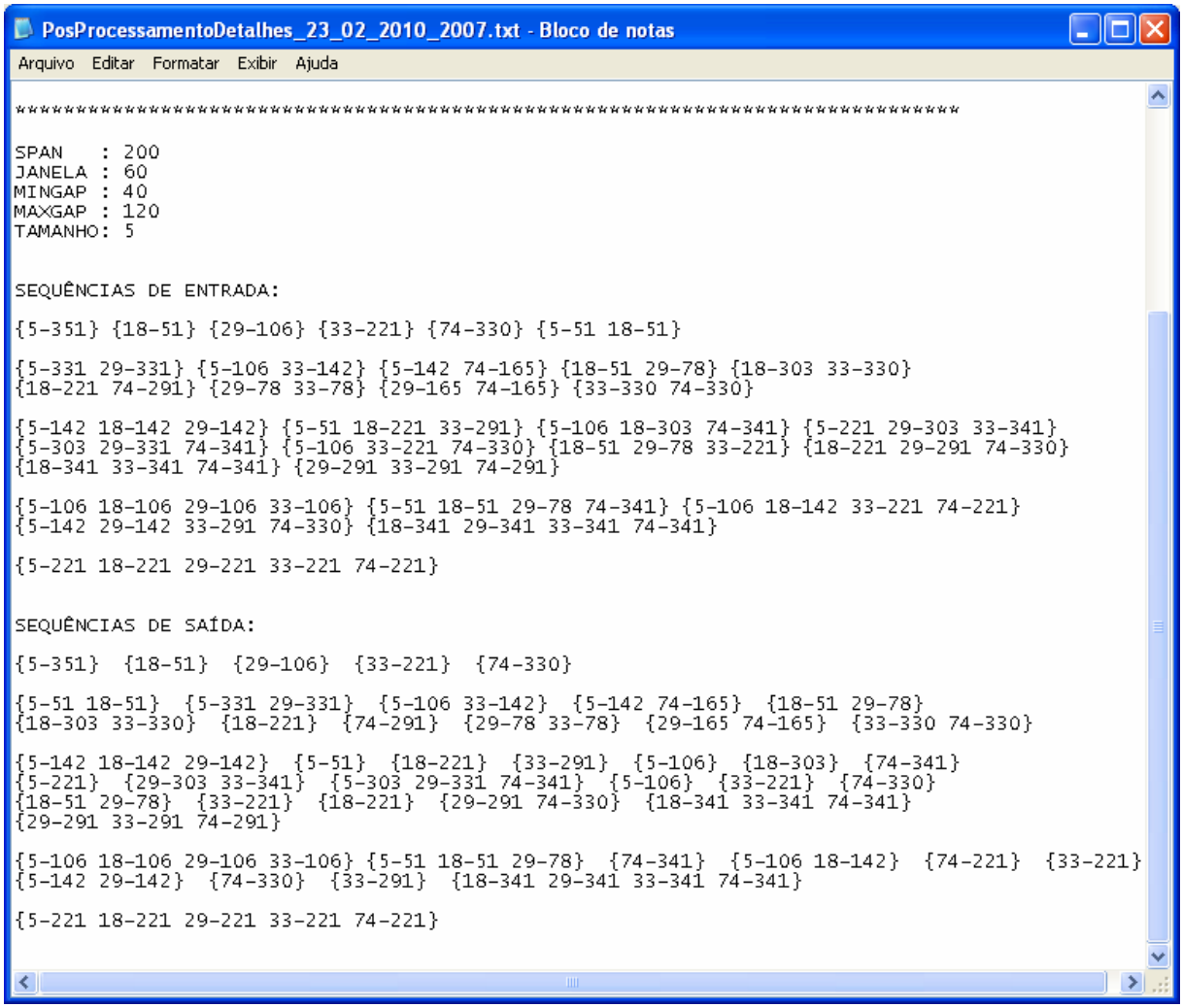


Figura 3.16 : Arquivo PosProcessamentoDetalhes_[DD]_[MM]_[AAAA]_HHMM

CAPÍTULO 4

EXPERIMENTOS REALIZADOS

Neste capítulo será descrita a base de dados que foi utilizada e será apresentado uma análise dos resultados obtidos pela aplicação do programa *PosProcessGSP* nos padrões sequenciais minerados pelo GSP relativos a essa base de dados.

4.1. BASE DE DADOS UTILIZADA

O banco de dados escolhido para esse projeto foi Northwind, que contêm dados de vendas de uma empresa fictícia chamada Northwind Traders, a qual importa e exporta alimentos especiais de todo o mundo.

O banco de dados Northwind cobre as principais atividades da empresa. Nas tabelas estão incluídos os produtos, os fornecedores desses produtos e os clientes que compram esses produtos. Além disso, há os funcionários, que vendem ou ajudam a vender. Também há tabelas com os itens gerados pelo processo de vendas: pedidos, notas fiscais, transportadoras, entrada e saída de estoques etc.

As tabelas desse banco são as seguintes: Categories**,** Customers**,** Employees**,** Orders**,** Order Details, Products**,** Shippers e Suppliers**.**

Para a realização desse trabalho, foi necessária a utilização e manipulação dos dados dessas tabelas a fim de criar um arquivo de entrada para o nosso minerador, como descrito na seção 3.1. Logo abaixo, temos o modelo físico do banco em questão apresentado na Figura 4.1:

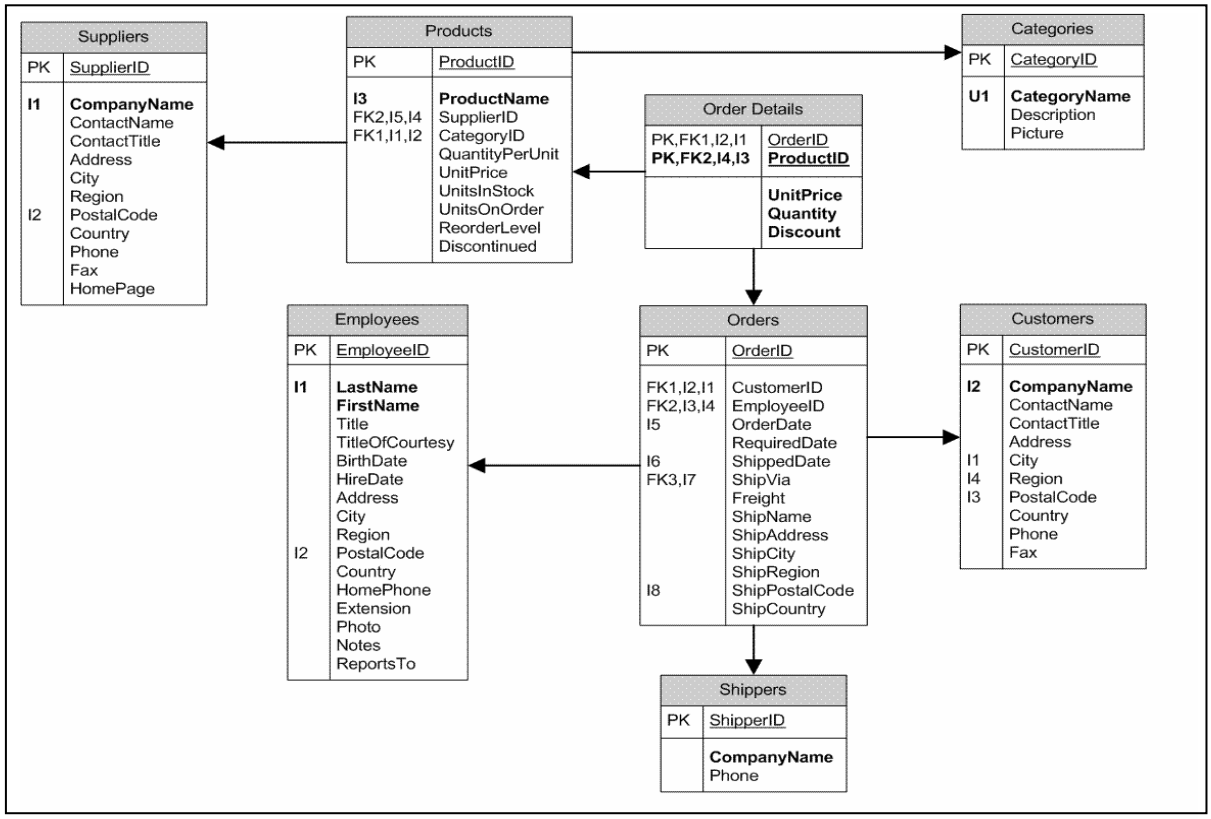


Figura 4.1 : Modelo Físico de Dados da base Northwind

4.2. RESULTADOS OBTIDOS

Nessa seção serão apresentados alguns dos resultados obtidos a partir da execução do programa de pós-processamento *PosProcessGSP*. Será utilizado 1 (um) arquivo na explicação dos resultados obtidos em função da variação dos parâmetros.

A Tabela 4.1 abaixo, exibe os padrões minerados sem restrições temporais pelo GSP, utilizados e presentes no arquivo de entrada pós-processado:

Nº	Nº EVENTOS	SEQUÊNCIA ENTRADA
1	1	{5-351}
2	1	{18-51}
3	1	{29-106}
4	1	{33-221}
5	1	{74-330}
6	2	{5-51 18-51}
7	2	{5-331 29-331}
8	2	{5-106 33-142}
9	2	{5-142 74-165}
10	2	{18-51 29-78}
11	2	{18-303 33-330}
12	2	{18-221 74-291}
13	2	{29-78 33-78}
14	2	{29-165 74-165}
15	2	{33-330 74-330}
16	3	{5-142 18-142 29-142}
17	3	{5-51 18-221 33-291}
18	3	{5-106 18-303 74-341}
19	3	{5-221 29-303 33-341}
20	3	{5-303 29-331 74-341}
21	3	{5-106 33-221 74-330}
22	3	{18-51 29-78 33-221}
23	3	{18-221 29-291 74-330}
24	3	{18-341 33-341 74-341}
25	3	{29-291 33-291 74-291}
26	4	{5-106 18-106 29-106 33-106}
27	4	{5-51 18-51 29-78 74-341}
28	4	{5-106 18-142 33-221 74-221}
29	4	{5-142 29-142 33-291 74-330}
30	4	{18-341 29-341 33-341 74-341}
31	5	{5-221 18-221 29-221 33-221 74-221}

Tabela 4.1 : Padrões Pós Processados

Antes de mostrarmos os resultados obtidos, é necessário ressaltarmos algumas observações:

- em sequências com apenas um 1 item/evento não precisam ser aplicadas as restrições temporais;
- em sequências contendo eventos com o mesmo "timestamp" não precisam ser aplicadas as restrições temporais;
- condições aplicadas às sequências com 2 eventos → (TEC - TFP < MINGAP) OU (TEC <= TIC + JANELA) E (TEC < = TIS + SPAN), ou seja, não existe o conceito de MAXGAP para sequências que possuem 2 eventos;

- condições aplicadas às sequências com mais de 2 eventos → (TEC - TFP < MINGAP) OU (TEC <= TIC + JANELA) E (TEC < = TIS + SPAN) E (TEC < = TIP + MAXGAP)

O primeiro resultado obtido é mostrado pela Tabela 4.2 abaixo a partir dos parâmetros de entrada considerados (SPAN = 200, JANELA = 60, MINGAP = 40, MAXGAP = 120 e TAMANHO = 5):

SPAN = 200; JANELA = 60; MINGAP = 40; MAXGAP = 120; TAMANHO = 5			
Nº	**Nº EVENTOS**	**SEQUÊNCIA ENTRADA**	**SEQUÊNCIA SAÍDA**
1	1	{5-351}	{5-351}
2	1	{18-51}	{18-51}
3	1	{29-106}	{29-106}
4	1	{33-221}	{33-221}
5	1	{74-330}	{74-330}
6	2	{5-51 18-51}	{5-51 18-51}
7	2	{5-331 29-331}	{5-331 29-331}
8	2	{5-106 33-142}	{5-106 33-142}
9	2	{5-142 74-165}	{5-142 74-165}
10	2	{18-51 29-78}	{18-51 29-78}
11	2	{18-303 33-330}	{18-303 33-330}
12	2	{18-221 74-291}	{18-221} {74-291}
13	2	{29-78 33-78}	{29-78 33-78}
14	2	{29-165 74-165}	{29-165 74-165}
15	2	{33-330 74-330}	{33-330 74-330}
16	3	{5-142 18-142 29-142}	{5-142 18-142 29-142}
17	3	{5-51 18-221 33-291}	{5-51} {18-221} {33-291}
18	3	{5-106 18-303 74-341}	{5-106 18-303 74-341}
19	3	{5-221 29-303 33-341}	{5-221} {29-303 33-341}
20	3	{5-303 29-331 74-341}	{5-303 29-331 74-341}
21	3	{5-106 33-221 74-330}	{5-106} {33-221} {74-330}
22	3	{18-51 29-78 33-221}	{18-51 29-78} {33-221}
23	3	{18-221 29-291 74-330}	{18-221} {29-291 74-330}
24	3	{18-341 33-341 74-341}	{18-341 33-341 74-341}
25	3	{29-291 33-291 74-291}	{29-291 33-291 74-291}
26	4	{5-106 18-106 29-106 33-106}	{5-106 18-106 29-106 33-106}
27	4	{5-51 18-51 29-78 74-341}	{5-51 18-51 29-78} {74-341}
28	4	{5-106 18-142 33-221 74-221}	{5-106 18-142} {33-221 74-221}
29	4	{5-142 29-142 33-291 74-330}	{5-142 29-142} {33-291 74-330}
30	4	{18-341 29-341 33-341 74-341}	{18-341 29-341 33-341 74-341}
31	5	{5-221 18-221 29-221 33-221 74-221}	{5-221 18-221 29-221 33-221 74-221}
Nº Total de Eventos = 80			
Nº Total de Eventos Isolados = 24			
Nº Total de Sequências de Entrada = 31			
Nº Total de Sequências de Saída = 44			

Tabela 4.2 : Primeiro resultado obtido

Considerando a sequência de entrada da linha 8 (oito) **{5-106 33-142}**, dado o valor de MINGAP (40), os eventos "5" e "33" não poderiam ficar isolados, uma vez que a diferença entre os seus timestamps é menor do que MINGAP, ou seja, 142 - 106 = 36. Dessa forma, pelos valores da JANELA e do SPAN (60 e 200, respectivamente), os eventos em questão constituem uma transação de 2 itens em uma mesma janela.

Já a sequência de entrada da linha 12 (doze) **{18-221 74-291}**, dado o valor de MINGAP (40), os eventos "18" e "74" já poderiam ficar isolados, uma vez que a diferença entre os seus timestamps é maior do que MINGAP, ou seja, 291 - 221 = 70. O que define a separação dos mesmos é que essa diferença ultrapassa o valor da JANELA (60), ou seja, os eventos em questão constituem eventos isolados.

A figura 4.2 abaixo mostra a tela do programa de pós-processamento com o respectivo resultado acima:

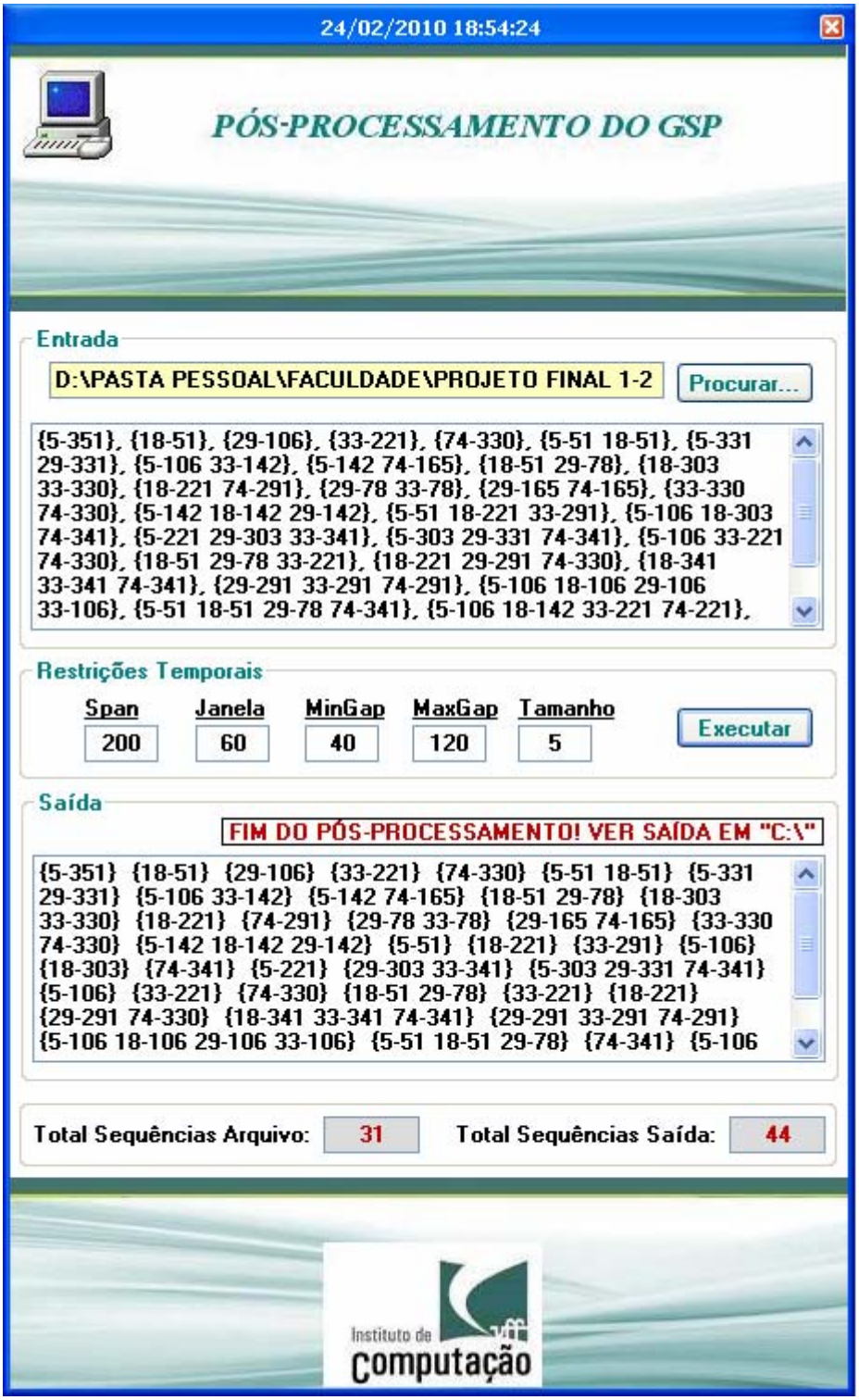


Figura 4.2 : Interface com o 1º resultado

Com o objetivo de obter um novo resultado, foi executado novamente o pós-processamento dos padrões contidos na Tabela 4.1, variando os valores dos parâmetros referentes às restrições temporais (SPAN = 250, JANELA = 90, MINGAP = 40, MAXGAP = 150 e TAMANHO = 5). O resultado mostrou que o aumento da

Janela (de 60 para 90) possibilitou que eventos que estavam isolados no resultado anterior ficassem juntos, constituindo uma mesma transação de tamanho maior. A Tabela 4.3 abaixo mostra esse resultado:

SPAN = 250; JANELA = 90; MINGAP = 40; MAXGAP = 150; TAMANHO = 5			
Nº	**Nº EVENTOS**	**SEQUÊNCIA ENTRADA**	**SEQUÊNCIA SAÍDA**
1	1	{5-351}	{5-351}
2	1	{18-51}	{18-51}
3	1	{29-106}	{29-106}
4	1	{33-221}	{33-221}
5	1	{74-330}	{74-330}
6	2	{5-51 18-51}	{5-51 18-51}
7	2	{5-331 29-331}	{5-331 29-331}
8	2	{5-106 33-142}	{5-106 33-142}
9	2	{5-142 74-165}	{5-142 74-165}
10	2	{18-51 29-78}	{18-51 29-78}
11	2	{18-303 33-330}	{18-303 33-330}
12	2	{18-221 74-291}	{18-221 74-291}
13	2	{29-78 33-78}	{29-78 33-78}
14	2	{29-165 74-165}	{29-165 74-165}
15	2	{33-330 74-330}	{33-330 74-330}
16	3	{5-142 18-142 29-142}	{5-142 18-142 29-142}
17	3	{5-51 18-221 33-291}	{5-51} {18-221} {33-291}
18	3	{5-106 18-303 74-341}	{5-106} {18-303 74-341}
19	3	{5-221 29-303 33-341}	{5-221 29-303} {33-341}
20	3	{5-303 29-331 74-341}	{5-303 29-331 74-341}
21	3	{5-106 33-221 74-330}	{5-106} {33-221} {74-330}
22	3	{18-51 29-78 33-221}	{18-51 29-78} {33-221}
23	3	{18-221 29-291 74-330}	{18-221 29-291} {74-330}
24	3	{18-341 33-341 74-341}	{18-341 33-341 74-341}
25	3	{29-291 33-291 74-291}	{29-291 33-291 74-291}
26	4	{5-106 18-106 29-106 33-106}	{5-106 18-106 29-106 33-106}
27	4	{5-51 18-51 29-78 74-341}	{5-51 18-51 29-78} {74-341}
28	4	{5-106 18-142 33-221 74-221}	{5-106 18-142} {33-221 74-221}
29	4	{5-142 29-142 33-291 74-330}	{5-142 29-142} {33-291 74-330}
30	4	{18-341 29-341 33-341 74-341}	{18-341 29-341 33-341 74-341}
31	5	{5-221 18-221 29-221 33-221 74-221}	{5-221 18-221 29-221 33-221 74-221}
Nº Total de Eventos = 80			
Nº Total de Eventos Isolados = 20			
Nº Total de Sequências de Entrada = 31			
Nº Total de Sequências de Saída = 41			

Tabela 4.3 : Segundo resultado obtido

Já nesse resultado, a sequência de entrada na linha 12 (doze) **{18-221 74-291}** manteve-se igual à saída. Como o valor da *Janela* aumentou (de 60 para 90), os eventos "18" e "74" apareceram juntos dessa vez em uma mesma transação,

uma vez que a diferença entre os seus timestamps é menor do que a nova *Janela*, ou seja, 291 - 221 = 70.

Já considerando a sequência de entrada na linha 19 (dezenove) **{5-221 29-303 33-341}**, dessa vez, o evento "5" e "29" ficaram juntos constituindo uma transação de 2 ventos, em função do aumento da *Janela* para 90. Já o evento "33" não se juntou a sequência, visto que a diferença entre o início da sequência (tempo do evento "5", ou seja, 221) e o seu timestamp (341) é de 120, sendo maior do que a *Janela*.

A figura 4.3 abaixo mostra a tela do programa de pós-processamento com o respectivo resultado acima:

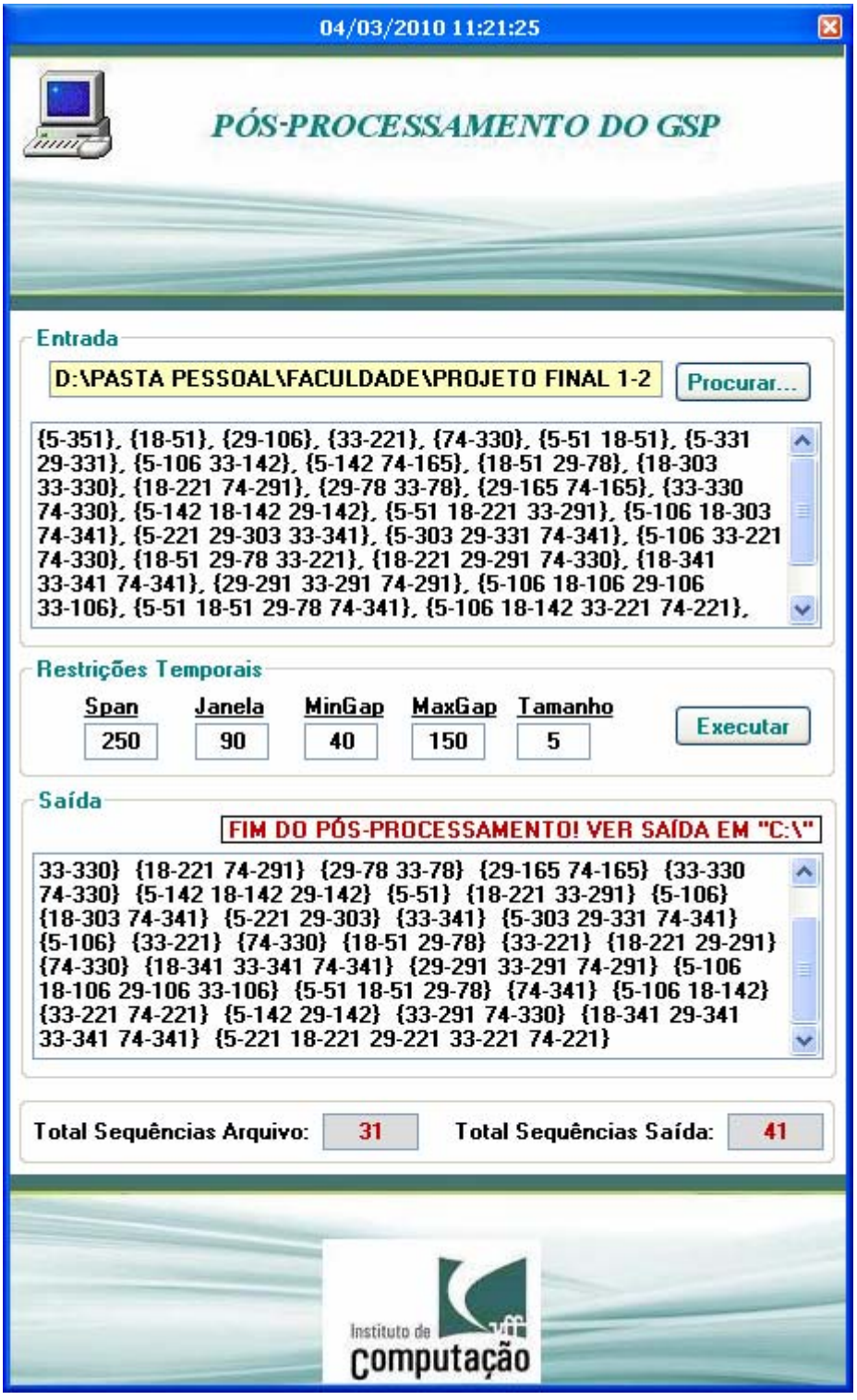


Figura 4.3 : Interface com o 2º resultado

CAPÍTULO 5

ANÁLISE DOS RESULTADOS E CONCLUSÕES

A partir de todo o trabalho realizado de análise dos resultados obtidos, foi concluído facilmente que o desafio da mineração de padrões não está associado apenas à busca de um conjunto de itens ou eventos mais freqüentes, e sim a encontrar padrões bem mais complexos. Dessa forma, nosso trabalho se propôs a apresentar técnicas de mineração de dados através do pós-processamento de padrões sequenciais.

Com os padrões gerados pelo GSP, foi realizado um pós-processamento dos mesmos utilizando restrições impostas pelo usuário como uma maneira de filtragem dos dados. Para cumprir com esse objetivo, foi criado um novo programa, adicionando algumas limitações críticas chamadas restrições temporais no intuito de fazer com que os padrões sequênciais pudessem ser úteis em aplicações reais, uma vez que os mesmos por si só minerados não tem muita aplicação.

Em particular, foi generalizado a definição de padrões sequênciais admitindo restrições temporais do tipo *MinGap* e *MaxGap* entre elementos adjacentes de um padrão sequêncial. Além disso, foi imposta a restrição de que todos os itens de um padrão sequêncial devem pertencer a uma mesma transação, permitindo assim o uso específico de uma *Janela* onde esses itens devem estar presentes, ou seja, concluímos que a mesma trata-se de um intervalo de tempo onde esses itens ou eventos constituem uma transação.

Com o objetivo de estabelecer um intervalo de observação dessas transações, foi acrescentado também um parâmetro definido como *Span*. E para garantir um patamar que é o número mínimo de eventos admitidos em um intervalo de observação ou *Span*, utilizamos o parâmetro *Tamanho*.

A partir dos resultados obtidos e apresentados na seção 4.2, chegamos as seguintes conclusões:

- a variação das restrições temporais é capaz de trazer como resultado novos conjuntos de padrões sequenciais, fornecendo assim um resultado mais interessante para o usuário final;
- como estávamos querendo caracterizar o comportamento de um comprador, o conceito de *Span* foi fundamental para um intervalo de observação, uma vez que não faz sentido querer comparar as compras de hoje com as de 5 (cinco) anos atrás, por exemplo;
- os parâmetros *Mingap* e *Maxgap* são determinantes para saber se possíveis eventos isolados podem ou não fazer parte de um mesmo contexto ou até mesmo de duas moléstias distintas ocorrendo em intervalos pequenos, por exemplo;
- a utilização da *Janela* serve para configurar um padrão de classificação "humano", como por exemplo, compras de mês, possíveis eventos que poderiam ter causado uma doença, um problema mecânico, entre outros;
- como o resultado obtido pode dar origem a novas perguntas, para possibilitar essa iteração, o tomador de decisão precisa ter acesso aos recursos da exploração de dados;
- através dos resultados obtidos, é possível realizar previsões futuras, baseando-se no comportamento passado dos dados.

No nosso caso de estudo, os eventos foram representados pelos itens de pedido que constituiam compras realizadas por um determinado cliente.

A Tabela 5.1 abaixo mostra e exemplifica exatamente as observações descritas acima, considerando os resultados obtidos (os padrões em amarelo são exatamente aqueles que sofreram modificações em função da variação das restrições temporais):

SPAN: de “200” para “250” ; JANELA: de “60” para “90” ; MAXGAP: de “120” para “150”				
Nº	Nº EVENTOS	SEQUÊNCIA ENTRADA	1º RESULTADO	2º RESULTADO
1	1	{5-351}	{5-351}	{5-351}
2	1	{18-51}	{18-51}	{18-51}
3	1	{29-106}	{29-106}	{29-106}
4	1	{33-221}	{33-221}	{33-221}
5	1	{74-330}	{74-330}	{74-330}
6	2	{5-51 18-51}	{5-51 18-51}	{5-51 18-51}
7	2	{5-331 29-331}	{5-331 29-331}	{5-331 29-331}
8	2	{5-106 33-142}	{5-106 33-142}	{5-106 33-142}
9	2	{5-142 74-165}	{5-142 74-165}	{5-142 74-165}
10	2	{18-51 29-78}	{18-51 29-78}	{18-51 29-78}
11	2	{18-303 33-330}	{18-303 33-330}	{18-303 33-330}
12	2	{18-221 74-291}	{18-221} {74-291}	{18-221 74-291}
13	2	{29-78 33-78}	{29-78 33-78}	{29-78 33-78}
14	2	{29-165 74-165}	{29-165 74-165}	{29-165 74-165}
15	2	{33-330 74-330}	{33-330 74-330}	{33-330 74-330}
16	3	{5-142 18-142 29-142}	{5-142 18-142 29-142}	{5-142 18-142 29-142}
17	3	{5-51 18-221 33-291}	{5-51} {18-221} {33-291}	{5-51} {18-221} {33-291}
18	3	{5-106 18-303 74-341}	{5-106} {18-303} {74-341}	{5-106} {18-303 74-341}
19	3	{5-221 29-303 33-341}	{5-221} {29-303 33-341}	{5-221 29-303} {33-341}
20	3	{5-303 29-331 74-341}	{5-303 29-331 74-341}	{5-303 29-331 74-341}
21	3	{5-106 33-221 74-330}	{5-106} {33-221} {74-330}	{5-106} {33-221} {74-330}
22	3	{18-51 29-78 33-221}	{18-51 29-78} {33-221}	{18-51 29-78} {33-221}
23	3	{18-221 29-291 74-330}	{18-221} {29-291 74-330}	{18-221 29-291} {74-330}
24	3	{18-341 33-341 74-341}	{18-341 33-341 74-341}	{18-341 33-341 74-341}
25	3	{29-291 33-291 74-291}	{29-291 33-291 74-291}	{29-291 33-291 74-291}
26	4	{5-106 18-106 29-106 33-106}	{5-106 18-106 29-106 33-106}	{5-106 18-106 29-106 33-106}
27	4	{5-51 18-51 29-78 74-341}	{5-51 18-51 29-78} {74-341}	{5-51 18-51 29-78} {74-341}
28	4	{5-106 18-142 33-221 74-221}	{5-106 18-142} {33-221 74-221}	{5-106 18-142} {33-221 74-221}
29	4	{5-142 29-142 33-291 74-330}	{5-142 29-142} {33-291 74-330}	{5-142 29-142} {33-291 74-330}
30	4	{18-341 29-341 33-341 74-341}	{18-341 29-341 33-341 74-341}	{18-341 29-341 33-341 74-341}
31	5	{5-221 18-221 29-221 33-221 74-221}	{5-221 18-221 29-221 33-221 74-221}	{5-221 18-221 29-221 33-221 74-221}

Tabela 5.1 : Conclusão dos Resultado Obtidos

Com isso, atingiu-se o objetivo desse trabalho que era de analisar e entender o complexo processo de Pós-processamento da Mineração de Padrões Seqüenciais, utilizando restrições temporais no intuito de filtrar os dados obtidos.

Pensando em um trabalho futuro, sugere-se a construção de um minerador GSP que seja capaz de aplicar as restrições temporais aos dados minerados em uma mesma aplicação, ou seja, o pós-processamento estaria

embutido na aplicação, trazendo um resultado final mais próximo às necessidades reais do usuário em um único programa.

Sugere-se também a criação de um programa que receba como entrada esse resultado final e o transforme nos dados reais da base de dados, o que facilitará o entendimento do resultado para o usuário final, que não mais será exibido em função de números, mas sim de dados concretos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Pei, J.; Han, J.; Mortazavi-Asl, B.; Pinto, H. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In Proceedings of the 2001 International Conference on Data Engineering (ICDE'01), Heidelberg, Germany, 2001. Disponível em: http://citeseer.nj.nec.com/cache/papers/cs/27046/http:zSzzSzwww-faculty.cs.uiuc.eduzSz~hanjzSzpdfzSzspan01.pdf/pei01prefixspan.pdf.

[2] Pinto, H.; Han, J.; Pei, J.; Wang, K. Multi-dimensional Sequential Pattern Mining, Proceedings of the 27th International Conference on Very Large Data Base (VLDB'01), Roma, Italy, 2001. Disponível em: http://citeseer.nj.nec.com/cache/papers/cs/22412/ftp:zSzzSzfas.sfu.cazSzpubzSzcszSztheseszSz2001zSzHelenPintoMSc.pdf/pinto01multidimensional.pdf. Acesso em: Agosto de 2003.

[3] FAYYAD, U.M. et al. Data mining and Knowledge Discovery an International Journal. Redmond, WA: Kluwer Academic Publishers, 1997.

[4] ADRIAANS, P.; ZANTINGE, D. Data Mining. Harlow: Addison-Wesley, 1997.

[5] DAS G.; LIN K., MANNILA, H.; RENGANATHAN, G.; SMYTH, P. **Rule discovery from time series**. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD-98), 4., 1998. pp 16-22.

[6] Han,J.; Kamber, M. (2006). “Mining Sequence Patterns in Trasactional Databases. In: Data Mining: Concepts and Techniques”. 2. ed. San Francisco: Elsevier. p. 23-498.

[7] AGRAWAL, R., AND SRIKANT, R. Mining sequential patterns. In Proceedings of the 11th International Conference on Data Engineering (1995), pp. 3–14.

[8] Srikant, R.; Agrawal, R. Mining Sequential Patterns Generalizations and Performance Improvements. In Proceedings of the Fifth Int'l Conference on Extending Database Technology (EDBT). Avignon, France, 1996. Disponível em: http://www.almaden.ibm.com/software/quest/Publications/papers/edbt96.pdf . Acesso em: agosto de 2003.

[9] AGRAWAL, R., AND SRIKANT, R. Fast algorithms for mining association rules. In Proceedings of the the 20th VLDB International Conference on Very Large Databases (1994), pp. 487–489.

[10] SRIKANT, R., AND AGRAWAL, R. Mining sequential patterns: Generalizations and performance improvements. In Proceedings of the 5th International Conference on Extending Database Technology (1996), pp. 3–17.

[11] ZAKI, M. J. SPADE: An efficient algorithm for mining frequent sequences. Machine Learning Journal 42, 1 (2001), 31–60.

[12] HAN, J., PEI, J., MORTAZAVI-ASL, B., CHEN, Q., DAYAL, U., AND HSU, M. C. FreeSpan: Frequent pattern-projected sequential pattern mining. In Proceedings of the International Conference on Knowledge Discovery and Data Mining (2000), pp. 355–359.