Descoberta de Padrões Sequenciais Utilizando Árvores Orientadas

Luís Cavique
DCET, Univ. Aberta
lcavique@univ-ab.pt

José Silva Coelho
DCET, Univ. Aberta
jcoelho@univ-ab.pt

Resumo

Hoje em dia, a descoberta de padrões sequenciais em grandes bases de dados é um assunto de grande interesse. A maior parte dos algoritmos de padrões sequenciais usam estruturas de memória muito grandes no espaço de soluções e geram um número enorme de regras. Com a utilização do modelo das cadeias de Markov é possível ter uma visão global, já que todos os itens são tomados em consideração. Contudo, para grandes matrizes nas cadeias de Markov, a complexidade do problema cresce muito rapidamente. Neste artigo pretendemos manter a visão global dos itens e evitar tempos computacionais não-polinomiais. Usando heurísticas baseadas no algoritmo de Prim, árvores e poli-árvores podem ser encontradas em redes cíclicas. Os resultados computacionais são apresentados para grandes bases de dados, criadas com um conhecido gerador artificial de dados de teste.

Palavras-chave: análise de dados, poli-árvores, descoberta de padrões sequenciais

Abstract

Nowadays, the detection of sequential patterns in large databases is a very relevant issue. Most of the Sequence Pattern algorithms use lattice structures in the search space and return a huge number of rules. With the Markov Chain Model a global view is possible since all items are taken into consideration. However, for large Markov Chain matrixes, the complexity of the problem increases very quickly. In this paper we want to maintain the global view of the items and avoid the non-polynomial computational times. Using heuristics based on the Prim algorithm, trees and poly-trees can be found in cyclic networks. Computational results are presented for very large datasets, created with a well-known artificial dataset generator.

Keywords: data mining, poly-trees, sequence pattern discovery

1 Introdução

A Descoberta de Padrões Sequenciais é um tema muito importante em "Data Mining", dado o grande número de aplicações que incluem a análise do cabaz de compras, "web mining", sequência de ADN, entre outros. A análise de sequências é usada para encontrar padrões de dados escondidos numa sequência de estados temporais, utilizando técnicas que vão desde a "machine learning", optimização, estatística e funcionalidade de bases de dados.

A maior parte das técnicas de descoberta de padrões sofre de três desvantagens: a necessidade de vários parâmetros, os problemas associados à escalabilidade dos algoritmos e o enorme número de regras geradas que evita uma visão global dos padrões.

i) Parâmetros: o utilizador tem de especificar o limite do suporte mínimo (minsup) para encontrar os padrões desejados. O valor de minsup pode gerar um corte excessivo ou pelo contrário muito reduzido de números de itens, obtendo-se assim um resultado sem aplicabilidade. O processo de sintonização do minsup deve ser repetido e analisado, o que torna esta tarefa muito demorada para grandes volumes de dados.

ii) Escalabilidade: dado que a maior parte dos algoritmos realiza uma busca exaustiva, em forma de cristal, e precisa de ler a base de dados mais do que uma vez, torna-se pouco compatíveis com grandes volumes de dados.

iii) Visualização: os sistemas com regras associativas geram um enorme número de regras, tornando a visualização e o processo de decisão difíceis por parte do utilizador.

Utilizando os modelos de Cadeias de Markov, não existe a necessidade de parâmetros e é possível ter uma visão global, já que todos os itens são tomados em consideração. Contudo, para matrizes de Cadeia de Markov com ordem elevada, surgem problemas de escalabilidade.

Tal como o algoritmo das Cadeias de Markov, a nossa abordagem utiliza uma matriz inicial semelhante, mas usa uma heurística polinomial baseada no algoritmo de Prim. A representação de padrões sequenciais com árvores foi utilizada em [Cavique 2007]. Neste artigo, desenvolvemos o trabalho prévio, mas com uma abordagem para a detecção de padrões sequenciais ao usar poli-árvores.

Na secção 2, é apresentado o trabalho relacionado com a área de "temporal mining". Na secção 3, é descrito o algoritmo proposto. Na secção 4, são discutidos os resultados computacionais alcançados com os dados gerados pelo IBM Quest Synthetic. Finalmente, na secção 5, são retiradas algumas conclusões.

2 Trabalho Relacionado

O problema de descoberta de padrões é a capacidade de extrair padrões interessantes dos dados originais. Contudo, é difícil definir quais são os ingredientes de um padrão interessante. Um parâmetro usual é dado pelos padrões que são frequentes. A prospecção de padrões temporais pode ser dividida em quatro diferentes abordagens: os padrões periódicos [Wang et al 2001], descoberta de sequências [Agrawal, Srikant 1995], episódios frequentes [Mannila et al 1997] e modelos das cadeias de Markov [Borges, Levene 2007].

2.1 Padrões Periódicos

A detecção de padrões periódicos em dados temporais é um assunto de extrema importância em aplicações reais. Dado um conjunto de literais D= {d(1), d(2), …, d(n)} onde d(i) foi colectado no instante t(i), o problema para a detecção de padrões periódicos é determinar se um determinado padrão ocorre na sequência.

Dado o padrão P= {p(1), p(2), … p(m)} e uma sequência de literais D, dizemos que P combina com D, ou que D suporta P, se p(k)=d(i+k) para $1 \leq k \leq m$. A primeira abordagem a este problema foi dada pelo algoritmo KMP "string matching" [Knuth 1977], onde o objectivo é encontrar um pequeno padrão num texto longo.

O conceito de meta-padrão foi apresentado com o propósito de representar um conjunto de padrões elementares numa forma sucinta [Wang et al. 2001]. O modelo do meta-padrão funciona numa estrutura semelhante a uma gramática, onde os meta-padrões são compostos de padrões simples ou de outros meta-padrões.

2.2 Descoberta de sequências

A descoberta de sequências envolve uma sequência de passos no tempo e é apresentada nesta secção utilizando o exemplo da base de dados de transacções de clientes. Os padrões temporais frequentes exprimem os padrões de compras de vários clientes. Uma sequência de compras é iniciada quando um cliente compra, por exemplo, um computador portátil. Existe um conjunto de itens, que podem ser posteriormente comprados, como uma rato, depois uma caixa de CDs, um saco, uma memória "flash", uma impressora, um "router" e vários itens ligados à multimédia.

A maior parte dos algoritmos de descoberta de sequências frequente utiliza pesquisas exaustivas. Estes algoritmos incluem para os algoritmos de pesquisa em largura, o algoritmo AprioriAll [Agrawal, Srikant 1995] e o (Generalized Sequential Pattern) [Srikant, Agrawal 1996] e para a pesquisa em profundidade o algoritmo SPADE (Sequential PAttern Discovery using Equivalence classes) [Zaki 2001].

2.3 Episódios Frequentes

Uma outra abordagem para descobrir padrões sequenciais em dados temporais é a descoberta de episódios frequentes apresentado por [Mannila et al 1997]. Nesta estrutura de padrões sequenciais, é dado um conjunto de eventos e o objectivo é descobrir sequências frequentes, ou episódios frequentes, que ocorrem ao longo do tempo.

Os dados referidos como sequência de eventos são expressos por E = {(e(1),t(1)), (e(2),t(2)), …(e(n), t(n))}, onde e(i) corresponde ao conjuntos dos eventos tipo e t(i) é um inteiro que denota o instante de tempo do evento i. O conjunto seguinte é um exemplo de uma sequência de eventos: {(A,3), (B,5), (C,9), (A,11), (B,12), (B, 14), (C, 15) , (A,17), (C,20)}. O resultado do método, conhecido como episódios, é um grafo acíclico de eventos ordenado na sequência temporal. Quando um episódio está ordenado no tempo é chamado de episódio em série, por exemplo $(A \rightarrow B \rightarrow C)$. Pelo contrário, quando não existe relação no tempo, o episódio é chamado de episódio em paralelo e representado por (A; B; C).

2.4 Modelos das Cadeias de Markov

As cadeias de Markov são grafos acíclicos com um conjunto de estados associados com um conjunto de transições entre estados. No caso da análise do cabaz de compras cada estado corresponde a um item e, no caso da navegação da 'web', cada estado é uma página. Os modelos de Markov foram usados para representar e analisar os utilizadores que navegam na 'web' em [Borges, Levene 2007].

A cada intervalo de tempo o sistema pode passar de um estado para um estado seguinte. A transição dos estados é quantificada utilizando probabilidades. Para cada estado na matriz de transições, P(i,j) é a probabilidade de o estado i transitar para o estado j, sendo a soma das probabilidades em cada estado igual a um. A cadeia de primeira ordem, é usualmente representada por uma matriz quadrada com as probabilidades de transição de estado para o estado seguinte. A cadeia de Markov de segunda ordem toma em consideração o estado anterior e o estado corrente na transição para o próximo estado. A matriz de ordem n, com n>1, cresce para novas dimensões, deixando de ser quadrada, tal como a matriz de primeira ordem. As cadeias de ordem superior consideram grandes conjuntos de estados anteriores, levando à expansão da matriz original e também à complexidade do problema.

3- Modelo da Árvore das Sequências

A nossa abordagem tem duas fases. Na primeira fase, dá-se a transformação da base de dados numa rede. Os dados (cliente, tempo, item) começam por ser ordenados. Para cada linha um novo atributo é criado, o próximo item. Em seguida é criada uma rede, i.e., um grafo com uma fonte (ou raiz) e um nó final chamado sumidouro. Nesta rede, os ciclos são permitidos, condensando a informação da base de dados e incorporando todas as sequências de compras dos clientes. Na rede construída, cada nó corresponde a um item e a transição representa uma sequência de um item para o próximo item. O peso de cada arco corresponde ao número de vezes que um item antecede um próximo item. A transformação da base de dados numa rede é idêntica à abordagem das Cadeias de Markov. A segunda fase, com base na rede cíclica é criada a árvore das sequências mais frequentes.

O modelo da Árvore das Sequências frequentes é apresentado no seguinte algoritmo de duas fases: a fase de transformação da base de dados numa rede cíclica e a procura da árvore das sequências frequentes.

Tabela 1- Algoritmo de Duas Fases para o Modelo da Árvore de Sequências

Input: base de dados (cliente, tempo, item);
Output: árvores (ou poli-árvore) de itens;
1) Fase de Transformação em Rede:
- Ordenar os dados por cliente, tempo e item;
- Criar um novo atributo próximo item;
- Construir a rede com base nas sequências do cliente;
2) Fase de Procura
- Procurar na rede cíclica, a árvores de itens mais frequente, utilizando uma heurística.

A abordagem da Árvores das Sequências tem fortes ligações às Cadeias de Markov. No modelo das Cadeias de Markov cada estado é um item e cada transição na matriz de primeira ordem, P(i,j), é a probabilidade de transitar de um estado i para um estado j.

Na nossa abordagem, tal como nas Cadeias de Markov, também permite ter uma visão global, já que todos os itens são tomados em consideração. No nosso modelo, em vez de utilizar frequências relativas, são utilizadas as frequências absolutas. Uma outra diferença, entre os modelos, é que a nossa abordagem permite grafos cíclicos, ao contrário dos modelos de Markov que só trabalham com grafos acíclicos. Como foi dito anteriormente, iremos utilizar heurísticas, baseadas no algoritmo de Prim, que permitem grande escalabilidade, evitando o aumento da dimensão do problema e dos tempos computacionais.

Neste artigo, são apresentadas duas heurísticas: Heurística Forward, que gera uma árvore de itens, e a Heurística Backward-and-Forward que é capaz de criar uma poli-árvore.

3.1 Conceitos de Redes

Nesta sub-secção é apresentado uma breve revisão bibliográfica sobre teoria dos grafos para ser utilizada nas próximas secções.

Dado um grafo não orientado e conexo, a árvore geradora ("spanning tree") é um sub-grafo que contém todos os vértices. A árvore geradora mínima é a árvore com peso inferior ou igual a qualquer outra árvore geradora. Este problema é de fácil resolução, isto é, existem algoritmos polinomiais que encontram a solução óptima. Os algoritmos propostos por Kruskal e Prim são dois exemplos bem conhecidos. No algoritmo de Kruskal, os arcos são escolhidos pelo seu peso, sem a preocupação da ligação a outros arcos, mas evitando os ciclos. No algoritmo de Prim a árvore cresce a partir de uma raiz arbitrária.

Num grafo G orientado e conexo, um sub-grafo acíclico B ("branch") é um ramo de G se o grau interno de cada vértice é no máximo igual a 1. A soma dos pesos dos arcos do ramo B é representada por w(B). O problema do máximo ramo ponderado é um problema da optimização que encontra o ramo mais pesado, cuja resolução foi proposta por [Edmonds 1967] e conhecido como o algoritmo de ramificação de Edmonds. Este algoritmo pode ser descrito em dois passos: o passo de condensação do grafo, para remover os ciclos e o passo de descompressão onde o ramo é encontrado.

Fulkerson [Fulkerson 1974] apresenta o mesmo problema com uma restrição adicional, o ramo deve ter início num determinado vértice, chamado raiz. Este algoritmo é também descrito em dois passos.

A poli-árvore é um grafo orientado acíclico, com um arco entre cada par de nós no máximo. O grau interno dos vértices de uma árvore é zero (a raiz) ou um. Por sua vez, o grau interno dos vértices de uma poli-árvore pode ser maior que um.

Os dois algoritmos, de [Edmonds 1967] e [Fulkerson 1974], geram árvores, com grau interno dos vértices de zero ou um. Para encontrar a poli-árvore ponderada de maior peso não existe nenhum algoritmo polinomial conhecido, pelo que iremos recorrer a heurísticas.

3.2 A Heurística Forward

Em trabalhos anteriores [Cavique 2007] a Árvore de Sequências resultava da aplicação do algoritmo do ramo com raiz mais pesado de Fulkerson [Fulkerson 1974] à rede que condensava os dados da base de dados. Tal como o algoritmo de ramificação de [Edmonds 1967], o algoritmo de Fulkerson tem, para o pior caso, uma complexidade temporal $\Theta(N^2)$, onde N representa o número de vértices. Podemos encontrar uma árvore idêntica, utilizando uma heurística baseada no algoritmo de Prim, com a heurística Forward.

Tabela 2- Heurística Forward

```
Input: Rede G;
Output: Árvore S;
Iniciar S;
Para cada vértice em G
   Para cada arco em G
        x = arg_max(vértice ponderado à frente, não visitado em G e ligado a S)
   Fim-para;
   Actualizar S com x;
Fim-para;
```

3.3 Heurística Backward-and-Forward

Para a geração de poli-árvores, desenvolveu-se a Heurística Backward-and-Forward, também baseada no algoritmo de Prim.

Tabela 3- Heurística Back-and-Forward

```
Input: Rede G;
Output: Árvore S;
Iniciar S;
Para cada vértice em G
   Para cada arco em G
        x = arg_max(vértice ponderado à frente, não visitado em G e ligado a S;
                    vértice ponderado atrás, não visitado em G e ligado a S)
   Fim-para;
   Actualizar S com x;
Fim-para;
```

Na figura 1 é apresentado um exemplo numérico. A figura 1.a mostra a rede cíclica. A figura 1.b mostra a árvore de maior peso que resulta da aplicação da heurística Forward, com um peso total de 136. A figura 1.c apresenta a poli-árvore de maior peso, obtida pela heurística Backward-and-Forward. Note que o vértice B tem um grau interno de dois e que o arco (A,B) foi encontrado utilizando o modo de pesquisa para trás. Note ainda, que a soma dos pesos é igual a 151, sendo maior que o resultado anterior, mostrando que a poli-árvore representa padrões mais frequentes.

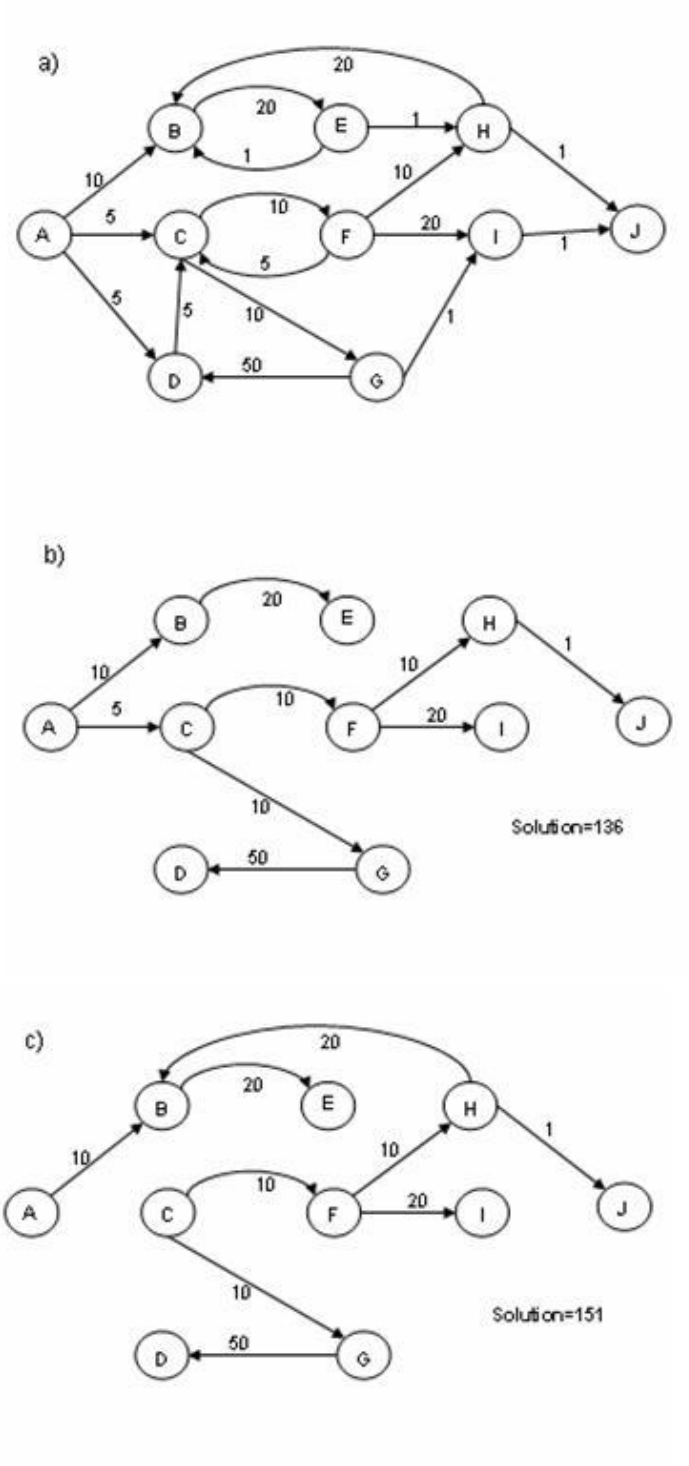

Figura 1. (a) Rede original (b) Heurística Forward devolve uma árvore (c) Heurística Back-and-forward devolve uma poli-árvore

As heurísticas Forward e Backward-and-Forward são utilizadas em circunstâncias diferentes. A heurística Forward pode ser utilizada se é fornecido um vértice inicial. Por exemplo, a maior parte dos problemas de prospecção de dados na "web" têm início num vértice de partida. Se, por outro lado, não existir informação acerca do nó de início, deve ser escolhida a heurística Backward-and-Forward.

4 Resultados Computacionais

Na validação de um algoritmo têm de ser tomadas algumas decisões, respeitantes aos dados de teste, ao ambiente computacional e às medidas de desempenho.

Os dados artificiais forma gerados pelo IBM Quest Synthetic [Agrawal, Srikant 1995]. O código do gerador pode ser descarregado de http://www.almaden. ibm.com/ software/ quest/ Resources/ datasets/ syndata.html #assocSynData. A tabela seguinte mostra os parâmetros do gerador e o seu significado.

Tabela 4 – Parâmetros do IBM Data Generator

Símbolo	Significado
D	Número de clientes em 1000s
C	Média de transacções por clientes
T	Média itens por transacção
S	Média do comprimento de sequência máxima
N	Número de diferentes itens em 1000s

O programa de computador para a heurística de Backward-and-Forward foi escrito em linguagem C e compilado no Microsoft Visual Studio 2005 C++. Os resultados computacionais foram obtidos com um processador Intel Core 2 Duo CPU T5600, 1.8 GHz com 1 GB de memória e num sistema operativo Windows XP.

As experiências computacionais foram levadas a cabo num sistema integrado de teste, chamado "Engine Tester", desenvolvido por [Coelho 2006].

Foram testados diferentes parâmetros dos ficheiros de teste variando os parâmetros de gerador da IBM, apresentado na tabela 5. O ficheiro D?C20T20S20N1 mostra o comportamento do algoritmo ao variar o número de clientes. O ficheiro D15C?T20S20N1 é usado para estudar a variação do número médio de transacções por cliente. Os ficheiros D10C15T?S20N1, D10C15T10S?N1 and D8C15T10S20N? são usados para estudar o impacto dos parâmetros T, S e N respectivamente.

O algoritmo mostra uma grande capacidade de escalabilidade, correndo em cerca de 20 segundos para quaisquer dos exemplos de teste. A robustez da heurística Backward-and-Forward deve-se à utilização dos dados condensados e às vantagens dos algoritmos polinomiais na procura de árvores em redes.

Tabela 5 – Variação dos parâmetros dos ficheiros de teste usando a heurística Backward-and-Forward Heuristic

D?C20T20S20N1	Número de clientes	Tempo CPU (segundo)
D1c20t20s20n1	1	19,7
D2c20t20s20n1	2	19,8
D3c20t20s20n1	3	20,1
D4c20t20s20n1	4	19,8
D5c20t20s20n1	5	19,7
D6c20t20s20n1	6	20,0

D15C?T20S20N1	Média de transacções por clientes	Tempo CPU (segundo)
D15c1t20s20n1	1	20,4
D15c3t20s20n1	3	20,2
D15c5t20s20n1	5	20,0
D15c7t20s20n1	7	20,1
D15c9t20s20n1	9	19,7

D10C15T?S20N1	Média itens por transacção	Tempo CPU (segundo)
D10c15t1s20n1	1	19,9
D10c15t2s20n1	2	19,7
D10c15t3s20n1	3	19,8
D10c15t4s20n1	4	19,7
D10c15t5s20n1	5	19,8
D10c15t6s20n1	6	19,7

D10C15T10S?N1	Média do comprimento de sequência máxima	Tempo CPU (segundo)
D10c15t10s4n1	4	5,0
D10c15t10s6n1	6	6,5
D10c15t10s8n1	8	8,8
D10c15t10s10n1	10	10,9
D10c15t10s12n1	12	12,5

D8C15T10S20N?	Número de diferentes itens em 1000s	Tempo CPU (segundo)
D8c15t10s20n1	1	20,1
D8c15t10s20n2	2	23,9
D8c15t10s20n3	3	24,4
D8c15t10s20n4	4	25,0
D8c15t10s20n5	5	24,9

5 Conclusões

Neste artigo apresentámos uma nova abordagem uma nova abordagem para a detecção de padrões sequenciais utilizando poli-árvores, desenvolvendo o trabalho de [Cavique 2007]. A nossa abordagem, o Modelo de Árvore de Sequências, tal como o Modelo das Cadeias de Markov, apresentam uma visão global dos dados, visto que condensam a informação em redes. Na nossa abordagem, em vez de frequências relativas, usamos frequências absolutas. São apresentadas duas heurísticas que descobrem árvores. A Heurística Forward descobre árvores clássicas e a Heurística Backward-and-Forward encontra poli-árvores. Ambas as heurísticas são inspiradas no algoritmo de Prim.

Para testar a nova abordagem, foram usados ficheiro de grandes dimensões. As experiências foram implementadas utilizando os ficheiros gerados pelo IBM Quest Synthetic. O problema de extracção de padrões interessantes dos dados utiliza geralmente algoritmos exaustivos. Dado que utilizamos condensações e algoritmos polinomiais, a comparação com qualquer dos outros algoritmos pode parecer injusta.

É nossa convicção que em "Data Mining" a complexidade algorítmica é extremamente importante, dado que são processados grandes volumes de dados. Para descobrir padrões sequenciais propomos a Heurística Backward-and-Forward para encontrar poli-árvores numa rede, baseado nas seguintes três vantagens:

i) Parâmetros: a maior parte dos algoritmos para detecção de sequências utiliza o suporte mínimo (minsup) com parâmetro para controlar a explosão combinatória. Para o algoritmo proposto não há necessidade de qualquer parâmetro.

ii) Escalabilidade: em comparação com os demais algoritmos, a nossa abordagem não faz uma procura exaustiva, contudo utiliza os dados condensados numa rede, tal como o Modelo das Cadeias de Markov. O procedimento que devolve o resultado da árvore tem uma complexidade polinomial e apresenta uma óptima escalabilidade.

iii) Visualização: usualmente, os pacotes de software mais conhecidos geram um grande número de regras, perdendo-se portanto a visão global. Na nossa abordagem todos os itens são tomados em consideração e a visualização das árvores mais pesadas é extremamente clara.

Bibliografia

Agrawal R., Srikant R., Mining sequential patterns, Proceedings 11th International Conference Data Engineering, ICDE, 3–14, IEEE Press (1995)

Borges J., Levene M., Evaluating Variable-Length Markov Chain Models for Analysis of User Web Navigation Sessions. IEEE Trans. Knowl. Data Eng. 19(4): 441-452 (2007)

Cavique L., Network Algorithm to Discover Sequential Patterns, in Progress in Artificial Intelligence, J.Neves, M.Santos and J.Machado (Eds.), EPIA 2007, LNAI 4874, Springer-Verlag Berlin Heidelberg, 406-414 (2007)

Coelho J.S., A model for making empirical tests - Engine Tester, working paper IST-Cesur (2006)

Edmonds J., Optimum branchings, J. Research of the National Bureau of Standards 71B, 233-240 (1967)

Fulkerson D.R., Packing rooted directed cuts in a weighted directed graph, Mathematical Programming 6, 1-13 (1974)

Knuth D.E., Morris J.H., Pratt V.R., Fast pattern matching in strings, SIAM Journal on Computing 6(1), 323-350 (1977)

Mannila H., Toivonen H., Verkamo I., Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1, 3, 259-289 (1997)

Srikant R., Agrawal R., Mining sequential patterns: Generalizations and performance improvements, Proceedings 5th International Conference Extending Database Technology, EDBT, 1057, 3–17 (1996)

Wang W., Yang J., Yu P., Meta-Patterns: revealing hidden periodic patterns, IEEE International Conference on Data Mining (ICDM) 550-557 (2001)

Zaki M.J., Spade: An efficient algorithm for mining frequent sequences, Machine Learning, 42, 31–60 (2001)