Halite$_{ds}$: Agrupamento de Dados em Subespaços de Séries Temporais Multidimensionais *

Afonso E. da Silva[1], **Robson L. F. Cordeiro**[1]

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
Avenida Trabalhador São-Carlense, 400 – Centro – São Carlos – SP – Brasil

aexpedito@grad.icmc.usp.br, robson@icmc.usp.br

Abstract. *Given a data stream with many attributes, how to cluster similar events? For example, how to cluster measurements of tens of climatic attributes to aid in forecasting the climate and extreme events? The task of clustering data with many attributes is known as* **subspace clustering***. Today, there exists a need for algorithms of this type well-suited to process data streams. This paper proposes the new algorithm* Halite$_{ds}$ *for subspace clustering in data streams. The new algorithm improves upon one existing technique, the method* Halite*, which was originally designed to process static datasets. Compared to using the base algorithm in data streams, the new algorithm takes advantage of the knowledge obtained from clustering past data to easy clustering data in the present, thus shrinking the runtime. Experiments using a synthetic stream, as well a real climatic stream indicate that the new algorithm is in average* 4.2 *times faster than the base algorithm, still obtaining similar accuracy of results.*

Resumo. *Dada uma série temporal com muitos atributos, como agrupar eventos similares? Por exemplo, como buscar grupos em medições de dezenas de atributos climáticos para previsão climática e de eventos extremos? O agrupamento de dados com muitos atributos é conhecido como* **agrupamento em subespaços***. Há hoje uma carência de algoritmos adequados a séries temporais. Este artigo propõe o novo algoritmo* Halite$_{ds}$ *para agrupamento em subespaços de séries temporais. É utilizada como base a técnica* Halite*, originalmente voltada à análise de dados estáticos. Em comparação ao uso do algoritmo base em séries temporais, o novo algoritmo permite que o conhecimento obtido dos dados do passado facilite o agrupamento dos dados no presente, diminuindo o tempo de análise. Experimentos em uma série sintética e em uma série climática real indicam que o novo algoritmo é em média* $4,2$ *vezes mais rápido do que o algoritmo base, e ainda obtém acurácia similar de resultados.*

1. Introdução

O volume de informação gerada ou coletada em formatos digitais nas mais diversas áreas de aplicação vem crescendo não somente na quantidade de objetos, mas também na quantidade e na complexidade dos atributos que descrevem cada objeto [Kang et al. 2014, Cordeiro et al. 2013a, Cordeiro et al. 2011]. Além disso, a coleta de dados é em muitos casos um processo contínuo, repetitivo e potencialmente infinito, no qual são registrados os valores dos atributos de interesse seguindo intervalos de tempo determinados. O conjunto de valores resultante forma uma **série temporal multidimensional**[1]. Nesse cenário, a busca por agrupamentos é um dos principais meios que nos auxiliam nas tarefas de análise, compreensão e extração de conhecimento de tais dados. Por exemplo, como analisar décadas de medições freqüentes de dezenas de atributos climáticos, como temperatura, precipitação de chuva,entre outros, a fim de identificar grupos de medições similares voltado ao auxílio à previsão climática e de eventos extremos?

*Este trabalho foi financiado por FAPESP, CAPES e CNPq.

[1]Assume-se aqui que uma série temporal multidimensional é formada por n séries temporais (unidimensionais), em que n é o número de atributos ou dimensões, cujos valores são coletados simultaneamente em todas as séries.

Nota-se nesse cenário que, sempre que um período de tempo finito para análise é especificado, qualquer série temporal pode ser vista como uma base de dados estática. Assim, é possível a análise de séries temporais feita por algoritmos tradicionais, originalmente voltados à mineração de dados estáticos, em que são analisados dados correspondentes a janelas (i.e., intervalos) de tempo passado específicas, de forma que a análise de janelas consecutivas permita entender a evolução temporal dos dados. Entretanto, uma limitação relevante desta abordagem é o fato de que os resultados da análise de cada janela são comumente ignorados durante a análise de janelas temporais subsequentes, de forma que o conhecimento prévio obtido dos dados no passado não é utilizado para facilitar a análise dos dados no presente, aumentando assim o custo computacional da análise e impossibilitando e/ou dificultando, por exemplo, a análise das séries em tempo real.

Como conseqüência, têm sido desenvolvidos novos algoritmos de mineração específicos para a análise de séries temporais com o intuito principal de minimizar e/ou eliminar as limitações encontradas na abordagem anteriormente descrita. Em específico, para a tarefa de agrupamento de dados com muitos atributos – conhecida na literatura como **agrupamento em subespaços** [Kriegel et al. 2009], nota-se hoje uma carência de algoritmos voltados à análise de séries temporais multidimensionais, visto que são raros os trabalhos que abordam este tema na literatura.

Este artigo propõe um novo algoritmo de agrupamento de dados em subespaços bem adequado à análise de séries temporais de média a alta dimensionalidade – o algoritmo *Halite*$_{ds}$. É utilizada como base a técnica *Halite* [Cordeiro et al. 2013a, Cordeiro et al. 2013b], a qual integra o estado da arte para agrupamento em subespaços de dados estáticos de média a alta dimensionalidade por permitir a análise precisa, rápida e escalável de tais dados. O algoritmo *Halite* analisa por multiresolução os dados de entrada identificando variações na densidade dos dados representados em um espaço multidimensional por meio de varreduras em uma estrutura de dados que armazena contagens de pontos incidentes em hipercubos multidimensionais de variados tamanhos – uma *quad-tree* multidimensional. Nesse contexto, a principal contribuição do novo algoritmo *Halite*$_{ds}$ é permitir a análise de janelas consecutivas de séries temporais, sem reconstruir, para cada nova janela, a estrutura de dados utilizada. Permite-se assim que o conhecimento prévio obtido dos dados agrupados no passado facilite o agrupamento dos dados no presente, diminuindo consideravelmente o custo computacional total da análise.

São reportados experimentos realizados em uma série temporal sintética, e em uma grande série de dados climáticos reais referente a quase um século de medições diárias de atributos climáticos. Para analisar tais séries, o novo algoritmo *Halite*$_{ds}$ demandou tempo de processamento em média $4,2$ vezes menor do que o tempo requerido pelo o algoritmo base *Halite*, e, ainda assim, obteve acurácia de resultados similar.

O restante deste artigo segue uma organização tradicional: os principais trabalhos relacionados são discutidos na Seção 2; o novo método *Halite*$_{ds}$ é descrito na Seção 3, e; experimentos são reportados na Seção 4. Por fim, a Seção 5 apresenta conclusões.

2. Trabalhos Relacionados

Nota-se hoje uma carência de algoritmos de **agrupamento em subespaços** voltados à análise de séries temporais multidimensionais, visto que são raros os trabalhos que abordam este tema na literatura. Em [Hassani et al. 2013], é proposta a medida *SubCMM* (*Subspace Cluster Mapping Measure*) para a avaliação de acurácia de resultados, a qual detecta erros ocasionados por surgimento, movimentação e divisão do subespaço de cada grupo da série analisada. Em [Miller and Hu 2012], é proposta a técnica de agrupamento em subespaços *StreamPreDeCon* com foco específico na análise de séries de pacotes de dados em redes de comunicação para detecção de pacotes anômalos. Em [Zhang et al. 2007], é proposto um algoritmo incremental de agrupamento em subespaços sobre janelas deslizantes, o qual inclui um modelo para descrever grupos e detectar mudanças de

subgrupos em séries temporais em relação a subconjuntos específicos de eventos. Por fim, em [Lee and Lee 2008, Park and Lee 2007] é proposto um algoritmo de mineração em subespaços que combina o agrupamento por *grids* com a mineração de conjuntos freqüêntes em séries temporais.

Nota-se ainda a existência de um trabalho correlato não referente a agrupamento de dados. Em [Sousa et al. 2006] utiliza-se uma *quad-tree* multidimensional para analisar séries temporais, em que também são propostas operações de auto-ajuste da estrutura. Entretanto, o foco do trabalho é a análise fractal e de dimensão intrínseca dos dados.

2.1. O Algoritmo *Halite*

O algoritmo *Halite* [Cordeiro et al. 2013b] permite a busca precisa, rápida e escalável por agrupamentos em subespaços de dados estáticos de média a alta dimensionalidade. O algoritmo possui duas fases principais: a primeira é responsável por representar os objetos de entrada em uma estrutura de dados de tipo *quad-tree* multidimensional; na segunda etapa, a estrutura criada é analisada em busca de grupos de objetos similares formados em subespaços do espaço original.

É importante notar que as mudanças propostas neste artigo para permitir a análise eficiente de séries temporais afetam apenas a primeira fase do algoritmo base *Halite*. Por esse motivo, apresenta-se aqui uma descrição detalhada da construção da estrutura de dados utilizada – primeira fase – e uma descrição sucinta da fase subsequente que se mantém inalterada – a busca por grupos. Detalhes são encontrados em [Cordeiro et al. 2013b].

2.1.1. Primeira fase

A fase inicial do algoritmo envolve a construção em memória primária de uma estrutura de dados de tipo *quad-tree* multidimensional, referenciada como a *Árvore de Contagens*. A estrutura representa uma base de dados ${}^dS$ contendo $n = |\ {}^dS\ |$ objetos com d atributos cada. Cada atributo refere-se a um valor real normalizado para o intervalo $[0, 1)$. Portanto, assume-se que a base de dados de entrada está inclusa no hipercubo unitário $[0, 1)^d$.

O número H de níveis (ou resoluções) da árvore é dado de acordo com a necessidade da aplicação. O valor sugerido pelos autores é $H = 4$. O nó raiz (nível 0) representa a base de dados como um todo, o qual é subdividido em 2^d hipercubos no próximo nível (nível 1), dividindo-se ao meio o espaço de dados em relação a cada uma das d dimensões de forma que cada novo hipercubo tenha lados com a metade do tamanho dos lados do hipercubo original. Novamente, o nível 2 apresenta 2^d hipercubos cada, tendo lados com a metade do tamanho dos lados dos hipercubos do nível anterior. Segue-se a divisão até o último nível $H - 1$. Cada hipercubo é representado como uma célula de contagem. A estrutura da célula contém os campos loc, n, $P[d]$, $usedCell$ e ptr. O campo loc é um identificador único para uma célula, em relação a todas as células que compartilham a mesma "célula pai" no nível anterior. n é o número total de pontos incidentes na célula, e $P[d]$ é um vetor que armazena a contagem de pontos na metade inferior da célula, em relação a cada uma das d dimensões. Por fim, $usedCell$ é uma variável booleana, utilizada apenas na fase de agrupamento, e ptr é o ponteiro para o próximo nível da árvore.

Os objetos de dados são representados por pontos no espaço d-dimensional, sendo contabilizados nas respectivas células em que incidem em cada um dos níveis da árvore. Uma célula a em um nível h é indicada por a_h. Para exemplificar a divisão do espaço, a Figura 1 ilustra o espaço 2-dimensional dividido em células de diferentes níveis da árvore, além de seus respectivos valores para o campo loc. A célula em destaque – com textura em linhas diagonais – está no nível 3, e contém campo $loc = 11$. Nos níveis 2 e 1, encontram-se sua "célula mãe" com campo $loc = 11$ e sua "célula avó" com campo $loc = 01$, respectivamente. Portanto, um ponto incidente sobre a célula em destaque $a_3.loc = 11$ é também contabilizado nas células correspondentes em todos os demais níveis, nesse

caso, nas células identificadas por $a_1.loc = 01$ e $a_2.loc = 11$.

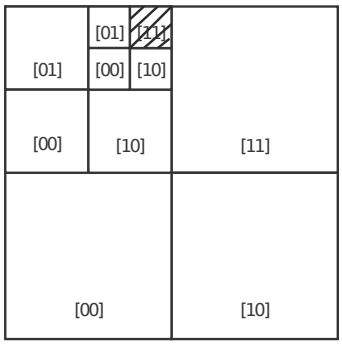

Figura 1. Células no espaço 2-dimensional.

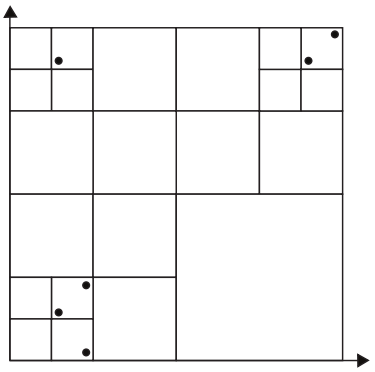

Figura 2. Pontos incidentes em células no espaço 2-dimensional.

A contagem de pontos é ilustrada nas Figuras 2 e 3. Seis pontos são apresentados no espaço 2-dimensional da Figura 2. A *Árvore de Contagens* correspondente é ilustrada na Figura 3, em que omite-se o campo $usedCell$ de cada célula para não poluir a ilustração. Têm-se células representadas em nós de três níveis distintos da árvore, em que cada nó apresenta $2^d = 4$ células.

2.1.2. Segunda fase

A segunda fase do algoritmo *Halite* busca por grupos de objetos similares em subespaços do espaço original dos dados de entrada. Apenas as contagens presentes na *Árvore de Contagens* são avaliadas nesta fase – não há novas leituras aos objetos de dados de entrada. O algoritmo encontra grupos com base na variação de densidade dos dados em subespaços. Uma análise por convolução é aplicada a cada nível da árvore, por meio de filtros de Laplace, a fim de localizar concentrações expressivas de pontos no espaço com todas as dimensões, as quais podem indicar grupos de dados existentes em subespaços do espaço original. Por ainda não terem sido validados, estes grupos em potencial são chamados de β-grupos. Em seguida, são analisados em separado os pontos de cada β-grupo, projetados em dimensões individuais, a fim de determinar os subespaços onde a concentração de pontos é mais evidente. Um teste estatístico é então aplicado para validar cada β-grupo, em relação ao seu subespaço de maior evidência, e os β-grupos com maior possibilidade de terem sido formados por tendências reais de agrupamento de dados são identificados e reportados.

3. Método proposto

Nesta seção é descrito o novo método *Halite*$_{ds}$ de agrupamento em subespaços de séries temporais multidimensionais. O método proposto adiciona três novas funcionalidades à *Árvore de Contagens* do algoritmo base, as quais são descritas nas subseções seguintes.

3.1. Inclusão de janelas de eventos

O algoritmo base *Halite* obtém os objetos de um banco de dados estático, de forma que é necessário ler cada objeto e inseri-lo na *Árvore de Contagens* sequencialmente, e só então o processo de agrupamento pode ser iniciado. Entretanto, séries temporais são potencialmente infinitas. Para analisá-las, deve-se permitir o agrupamento de subconjuntos dos dados, ao invés de aguardar a leitura do conjunto completo. Assume-se neste artigo que a série de dados de entrada é composta por uma seqüência de janelas de eventos, comumente referentes a intervalos temporais, como, por exemplo, janelas mensais, semestrais ou anuais, e que a memória primária disponível permite o armazenamento de M janelas. O novo algoritmo *Halite*$_{ds}$ lê e insere na *Árvore de Contagens* as M janelas iniciais, e então

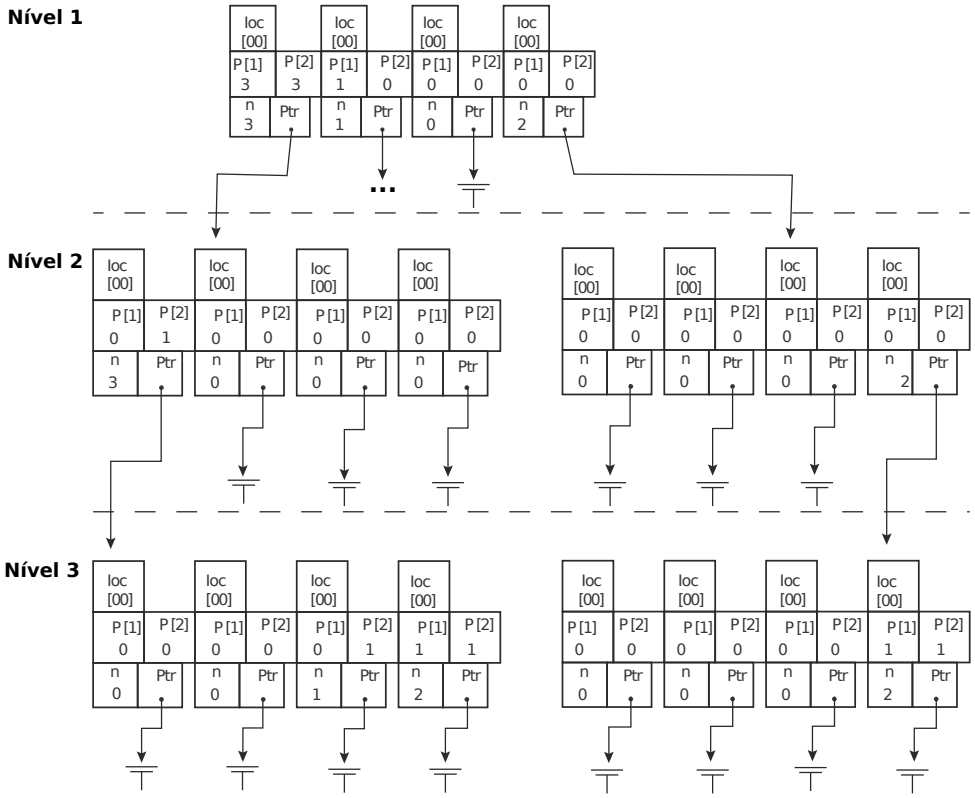

Figura 3. *Árvore de Contagens* referente aos pontos da Figura 1.

as analisa por agrupamento. Cada nova janela lida sobrescreve a janela mais antiga na árvore, e uma nova análise por agrupamento é realizada nas M janelas mais recentes. Por exemplo, se $M = 3$ janelas cabem em memória, o algoritmo lê e armazena a primeira janela, lê a segunda janela e a armazena, lê a terceira e a armazena, e só então busca por grupos nas três janelas lidas. Ao ler a quarta janela, os dados da primeira janela são sobrescritos e um novo processo de agrupamento é iniciado, considerando apenas a segunda, terceira e quarta janelas. Efetuam-se nesse processo análises por agrupamento conforme novas janelas são lidas, sempre em referência às M janelas mais recentes da série de dados. Permite-se assim a análise eficiente da evolução temporal dos dados.

O modelo de janelas empregado permite o suporte eficiente a aplicações que recebem eventos intermitentemente para análise sem a necessidade de realizar operações relativamente caras, originalmente propostas para a análise de bases de dados estáticas. Especificamente, o novo algoritmo *Halite*$_{ds}$ permite analisar as M janelas mais recentes de uma série temporal sem a necessidade de reconstruir por completo a *Árvore de Contagens* a cada nova janela lida. Vale notar que experimentos reportados na Seção 4 indicam que a construção da arvore é a fase com maior custo computacional do algoritmo base.

São propostas neste artigo alterações na árvore descrita na Subseção 2.1.1, algumas delas referentes aos campos de suas células. O campo n, que armazena a contagem de pontos incidentes em uma célula, passa a considerar também a janela temporal a que esses pontos pertencem. Na implementação atual do novo algoritmo *Halite*$_{ds}$ esse campo é representado por um vetor com posições referentes às contagens de pontos de janelas específicas. Adiciona-se também mais uma dimensão ao campo $P[i]$ para representar janelas, o qual armazena a contagem de pontos na metade inferior de cada célula em relação a cada atributo i. A implementação atual desse campo é feita por uma matriz 2-dimensional $P[i][j]$, em que i representa um atributo e j uma janela.

O Algoritmo 1 descreve a operação de inserção de uma janela na *Árvore de Contagens*. As

operações de expansão e contração são descritas nas seções posteriores.

Algoritmo 1: insereJanela($eventosDaJanela, idDaJanela$)

Entrada: Eventos da nova janela e número identificador da nova janela
Saída: Árvore de contagem com a nova janela

```
1  início
2      Descarta dados da janela mais antiga na árvore;
3      para cada evento e da janela faça
4          se e está fora do hipercubo representado por raiz então
5              Expande(e); // expande a árvore para representar o evento e
6          fim
7          Insere o evento e na árvore, com referência à janela atual;
8      fim
9      se há uma única célula no nível 1 da árvore então
10         Contrai( ); // contrai a árvore
11     fim
12 fim
```

3.2. Análise de dados não normalizados

O algoritmo base *Halite* recebe objetos de dados com valores de atributos normalizados entre 0 e 1, e os insere na árvore com seu nó raiz sempre representando o hipercubo unitário $[0, 1)^d$. Os limites espaciais mínimo e máximo de cada célula da árvore são sempre invariáveis e pré-definidos, além de serem sempre os mesmos para todas as dimensões. No entanto, em diversas séries temporais reais os valores limites mínimo e máximo de cada atributo não são previamente conhecidos, inviabilizando qualquer normalização. É então necessário capturar e representar na árvore eventos não normalizados, assumindo que seus atributos podem conter qualquer valor real.

Para permitir a análise de séries temporais é proposta neste artigo mais uma alteração na árvore descrita na Subseção 2.1.1. No novo algoritmo *Halite*$_{ds}$ o nó raiz da árvore representa um hipercubo com tamanho variável $r0$ para cada lado. O valor inicial de $r0$ é definido pelo domínio ativo dos atributos da primeira janela de dados lida. Especificamente, $r0 = max(maior_i - menor_i)$, em que $maior_i$ e $menor_i$ se referem respectivamente ao maior e ao menor valor do domínio ativo do atributo i em relação à primeira janela. A primeira janela também define valores iniciais para os atributos L_i e U_i, os quais se referem respectivamente aos valores limites mínimo e máximo do espaço de dados representado pelo nó raiz da árvore para cada atributo i. A inicialização de L_i é dada por $L_i = menor_i$ e, após isso, U_i é inicializado com $U_i = L_i + r0$. Note que os limites do espaço de representação da árvore são variáveis e dinâmicos, e, portanto, assim também são os valores de L_i e U_i para cada atributo i e de $r0$, os quais são sempre armazenados junto à árvore para correta interpretação do espaço de dados representado.

3.3. Expansão e contração do espaço de representação

Dado que atributos de séries temporais podem assumir quaisquer valores dos números reais, propõe-se para o algoritmo *Halite*$_{ds}$ o auto-ajuste do espaço de representação da *Árvore de Contagens*, por meio de duas novas operações de expansão e contração de espaço. Em uma expansão, um novo nó raiz é criado na árvore para representar um espaço maior, enquanto o antigo raiz se torna "filho" do novo nó. De modo contrário, em uma contração o nó raiz é excluído por ter apenas um nó "filho", enquanto o "filho" assume o papel de nova raiz para representar um espaço menor. Permite-se assim inserir eventos que não incidem sobre o espaço até então representado.

No novo algoritmo *Halite*$_{ds}$, a primeira janela de eventos determina o intervalo inicial que o nó raiz da árvore irá adotar para cada dimensão. Na chegada de um evento e de uma nova janela,

Algoritmo 2: Expande(e)

Entrada: Coordenadas e_i de um evento e **Saída:** Árvore expandida

```
1  início
2      se e está fora do hipercubo representado por raiz então
3          // incrementa o nível inicial de busca por grupos (para a segunda fase)
4          nivelInicial = nivelInicial + 1;
5          H = H + 1; // incrementa a altura da árvore
6          para cada coordenada e_i do evento e faça
7              A_i = L_i; // valor antigo de L_i para uso posterior
8              se e_i < L_i então
9                  L_i = L_i − r0;
10             senão
11                 U_i = U_i + r0;
12             fim
13         fim
14         r0 = r0 * 2;  aux = raiz;  raiz = new(nó);  raiz.ptr = aux;
15         para cada dimensão i faça
16             para cada janela j faça
17                 se e_i < A_i então
18                     raiz.P[i][j] = aux.n[j];
19                 senão
20                     raiz.P[i][j] = 0;
21                 fim
22             fim
23         fim
24         Expande(e); // chamada recursiva para uma possível nova expansão
25     fim
26 fim
```

se e contiver um valor de atributo e_i tal que $e_i < L_i$ ou $e_i > U_i$, em relação a pelo menos um atributo i, é realizada uma operação de expansão na árvore para possibilitar sua representação, e só então, o evento é inserido. Como pode ser visto no Algoritmo 2, a operação de expansão é realizada recursivamente até que todas as coordenadas do evento possam ser representadas. Por outro lado, a contração (Algoritmo 3) realiza o procedimento contrário à expansão para a eliminar da memória todas as células vazias da árvore, causadas por remoção de uma janela antiga. Exemplos de expansão e de contração são ilustrados nas Figuras 4 e 5 respectivamente.

4. Experimentos

Esta seção reporta os experimentos realizados com o novo método *Halite*$_{ds}$. Em comparação ao uso do método base *Halite* em séries temporais, visa-se responder:

- **Q1** – Qual é o ganho em tempo de processamento obtido pelo novo método?
- **Q2** – Qual é o correspondente impacto na acurácia dos resultados obtidos?

Os experimentos utilizaram um processador de $2,0$ GHz de uma máquina com SO Linux e 2 GB de memória principal. Ambas as técnicas *Halite* e *Halite*$_{ds}$ foram configuradas com os valores padrão para seus parâmetros de entrada, $H = 4$ e $\alpha = 1.0E-10$, seguindo o que sugere a proposta original de *Halite*. Note que α é utilizado apenas na busca por grupos – segunda fase – que se mantém inalterada neste artigo.

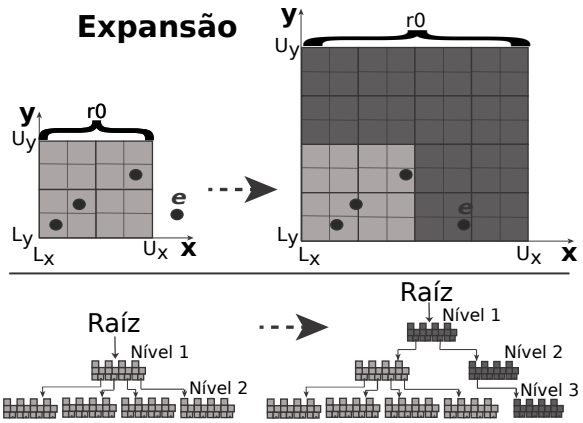

Figura 4. Exemplo de expansão.

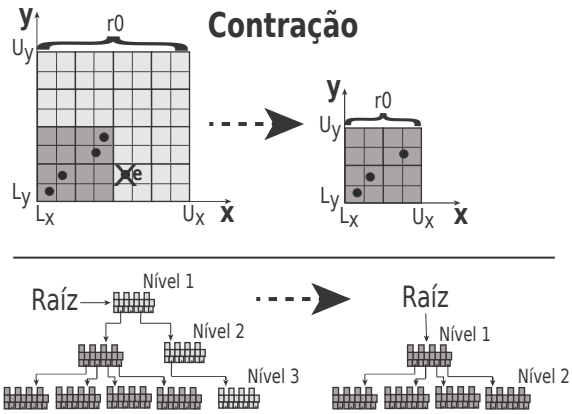

Figura 5. Exemplo de contração.

Algoritmo 3: Contrai( )

Saída: Árvore contraída

início
  se *há uma única célula no nível* 1 *da árvore* $\wedge$
    $H >$ *valor de* H *originalmente informado pelo usuário* **então**
      // decrementa o nível inicial de busca por grupos (para a segunda fase)
      $nivelInicial = nivelInicial - 1$;
      $H = H - 1$; // decrementa a altura da árvore
      $c =$ célula única do nível 1 da árvore; $r0 = r0/2$;
      para *cada dimensão* i **faça**
        se c *está na metade superior de* $raiz$*, em relação a* i **então**
          $L_i = L_i + r0$;
        senão
          $U_i = U_i - r0$;
        fim
      fim
      para *cada dimensão* i **faça**
        para *cada janela* j **faça**
          $raiz.P[i][j] = c.P[i][j]$;
        fim
      fim
      $aux = raiz$; $raiz = c.ptr$; $delete(aux)$;
  fim
  $Contrai(\,)$; // chamada recursiva para uma possível nova contração
fim

4.1. Experimentos com dados sintéticos

Foi gerada uma série temporal multidimensional sintética de 15 atributos (i.e., 15 séries unidimensionais com eventos simultâneos) dividida em 100 janelas, cada janela contendo $\sim 10k$ eventos. Cada janela inclui 5% de *outliers* e 10 grupos existentes apenas em subespaços do espaço original de 15 dimensões, com subespaços escolhidos aleatóriamente. Cada grupo segue uma distribuição normal em seu correspondente subespaço, definida com média e desvio padrão aleatórios. Visto que as janelas devem representar evoluções temporais de uma mesma base de dados, cada grupo da primeira janela tem seu par correspondente em cada uma das janelas posteriores, seguindo a mesma média e

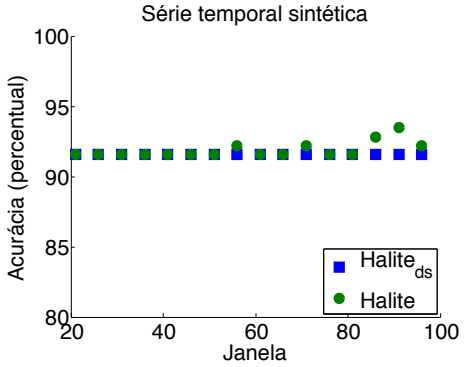

Figura 6. Acurácia em série sintética.

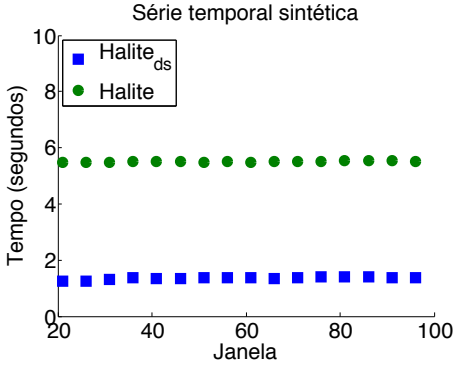

Figura 7. Tempo em série sintética.

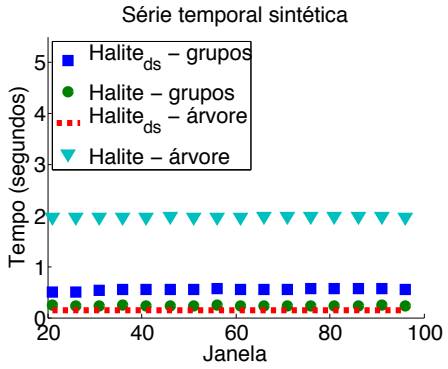

Figura 8. Tempo em série sintética – árvore e grupos.

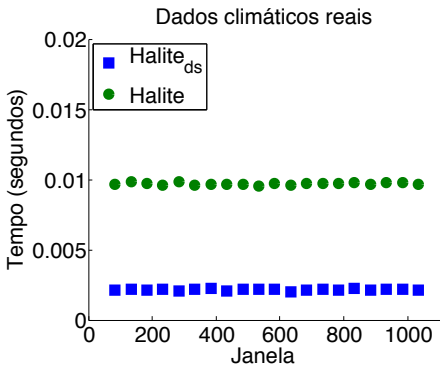

Figura 9. Tempo em série climática real.

desvio padrão na geração de dados. Utilizou-se aqui o algoritmo de geração de dados do algoritmo base, o Algoritmo 6 em [Cordeiro et al. 2013b].

Os resultados em dados sintéticos são vistos nas Figuras 6, 7 e 8. $M = 20$ foi utilizado. A Figura 6 reporta a acurácia (percentual) dos resultados em relação à cada janela lida. Como pode ser visto, a acurácia se mantém similar em todos os casos para ambos os algoritmos. Para computar a acurácia, utilizou-se a mesma estratégia baseada em valores de precisão e revocação definida na proposta original do algoritmo base, Seção 8.1 em [Cordeiro et al. 2013b]. Nesse processo, valores de precisão e de revocação foram computados comparando-se os resultados de agrupamento apresentados por cada algoritmo com o resultado ideal (i.e., "verdade absoluta" ou *ground truth*) previamente conhecido apenas nos dados sintéticos.

A Figura 7 reporta os respectivos tempos de processamento. Como pode ser visto, o novo método *Halite*$_{ds}$ é em média 4 vezes mais eficiente do que o método base *Halite*. Esses resultados são detalhados na Figura 8 para mostrar o tempo gasto em duas fases específicas: a construção da árvore e a busca por grupos. Como pode ser visto, o tempo de construção da árvore do algoritmo base *Halite* equivale em média a $89,3\%$ do tempo total da análise, sendo este o grande gargalo no processamento. Justificam-se assim os ganhos expressivos da nova abordagem proposta.

4.2. Experimentos com dados reais

Uma grande série climática real multidimensional foi estudada contendo quase um século (i.e., anos entre 1917 e 2010) de medições diárias de três de atributos climáticos, temperaturas máxima e mínima e precipitação de chuva, efetuadas em uma estação climática real – a estação da ESALQ-USP localizada no município de Piracicaba[2]. A série temporal completa contém 33.217 eventos. Os resultados obtidos são reportados na Figura 9. Janelas mensais e $M = 60$ foram utilizados. É reportado o tempo

[2] Dados gentilmente fornecidos por colaboradores do CEPAGRI-UNICAMP e CPTEC-INPE.

de processamento para cada algoritmo e cada janela. Como pode ser visto, o novo método *Halite*$_{ds}$ é novamente mais eficiente do que o método base *Halite*, sendo neste caso $4,5$ vezes mais rápido.

5. Conclusão

Este artigo propõe um novo algoritmo de agrupamento de dados em subespaços de séries temporais de média a alta dimensionalidade – o algoritmo *Halite*$_{ds}$. É utilizada como base a técnica *Halite*. A principal contribuição do novo algoritmo *Halite*$_{ds}$ é permitir a análise de janelas consecutivas de séries temporais, sem reconstruir, para cada nova janela, a estrutura de dados utilizada. Permite-se assim que o conhecimento prévio obtido dos dados agrupados no passado facilite o agrupamento dos dados no presente, diminuindo consideravelmente o custo computacional total da análise. São reportados resultados de experimentos realizados uma série temporais sintética, e também em uma grande série de dados climáticos reais referente quase um século de medições diárias de atributos climáticos. Para analisar tais séries, o novo algoritmo *Halite*$_{ds}$ foi em média $4,2$ vezes mais rápido do que o algoritmo base *Halite*, e, ainda assim, obteve acurácia de resultados similar.

Referências

Cordeiro, R., Faloutsos, C., and Traina Jr., C. (2013a). *Data Mining in Large Sets of Complex Data*. SpringerBriefs in Computer Science Series. Springer-Verlag New York Incorporated.

Cordeiro, R. L. F., Jr., C. T., Traina, A. J. M., López, J., Kang, U., and Faloutsos, C. (2011). Clustering very large multi-dimensional datasets with mapreduce. In Apté, C., Ghosh, J., and Smyth, P., editors, *KDD*, pages 690–698. ACM.

Cordeiro, R. L. F., Traina, A. J. M., Faloutsos, C., and Traina Jr., C. (2013b). Halite: Fast and scalable multiresolution local-correlation clustering. *IEEE TKDE*, 25(2):387–401.

Hassani, M., Kim, Y., Choi, S., and Seidl, T. (2013). Effective evaluation measures for subspace clustering of data streams. In Li, J., Cao, L., Wang, C., Tan, K., Liu, B., Pei, J., and Tseng, V., editors, *Trends and Applications in Knowledge Discovery and Data Mining*, volume 7867 of *Lecture Notes in Computer Science*, pages 342–353. Springer Berlin Heidelberg.

Kang, U., Meeder, B., Papalexakis, E. E., and Faloutsos, C. (2014). Heigen: Spectral analysis for billion-scale graphs. *IEEE Trans. Knowl. Data Eng.*, 26(2):350–362.

Kriegel, H.-P., Kröger, P., and Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM TKDD*, 3(1):1:58.

Lee, J. W. and Lee, W. S. (2008). A coarse-grain grid-based subspace clustering method for online multi-dimensional data streams. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 1521–1522, New York, NY, USA. ACM.

Miller, Z. and Hu, W. (2012). Data stream subspace clustering for anomalous network packet detection. *J. Information Security*, 3(3):215–223.

Park, N. H. and Lee, W. S. (2007). Grid-based subspace clustering over data streams. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 801–810, New York, NY, USA. ACM.

Sousa, E. P., Traina, A. J., Traina Jr., C., and Faloutsos, C. (2006). Measuring evolving data streams' behavior through their intrinsic dimension. *New Generation Computing*, 25(1):33–60.

Zhang, Q., Liu, J., and Wang, W. (2007). Incremental subspace clustering over multiple data streams. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 727–732.