CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA (CEFET/RJ)

MÉTODOS DE AGRUPAMENTO DE ACESSOS BASEADOS EM FLUXOS DE REQUISIÇÕES DE SERVIDORES WEB

Riccardo Campisano

Professores orientadores: Eduardo Soares Ogasawara, D.Sc
Raphael Carlos Santos Machado, D.Sc

Rio de Janeiro, RJ
Dezembro / 2014

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA (CEFET/RJ)

MÉTODOS DE AGRUPAMENTO DE ACESSOS BASEADOS EM FLUXOS DE REQUISIÇÕES DE SERVIDORES WEB

Riccardo Campisano

Projeto Final II apresentado em cumprimento às
normas do Departamento de Educação Superior
do CEFET-RJ, como parte dos requisitos para obtenção
do título de Tecnólogo em Sistemas para Internet

Professores orientadores: Eduardo Soares Ogasawara, D.Sc
Raphael Carlos Santos Machado, D.Sc

Rio de Janeiro, RJ
Dezembro / 2014

À minha esposa e parceira de sempre,
que me incentivou ao longo de toda a graduação.

RESUMO

Embora não seja possível garantir a inviolabilidade de um sistema computacional, as técnicas de detecção de intrusões podem se tornar um recurso complementar para defender estes sistemas das tentativas de ataque provenientes da rede de computadores. Diferentes fontes de dados podem ser analisadas para este fim. A mais tradicional fonte de dados é obtida monitorando diretamente os pacotes trafegados pela rede. Recentemente, o estudo dos fluxos de requisições como forma de apoiar o monitoramento da rede passou a despertar o interesse da comunidade na tentativa de resolver os problemas computacionais causados pelo grande número de pacotes trafegados. O fluxo de requisições pode ser analisado como uma série temporal. O presente trabalho tem como objetivo transformar os dados monitorados em séries temporais de fluxos de requisições e agrupá-las para medir o potencial de classificação. Para avaliar a proposta, foi estudado o comportamento combinado de usuários e *crawlers* dos motores de busca em um servidor WEB durante um mês. Os resultados obtidos pelo agrupamento destas séries temporais mostram que os perfis característicos dos *crawlers* se diferenciam dos demais acessos ao se apresentarem em grupos distintos. Esse resultado indica que há um potencial de exploração de métodos de classificação a partir destes grupos formados para fins de detecção de intrusões.

Palavras-chave: Detecção de intrusões, requisições WEB, fluxos, mineração de dados, agrupamentos, séries temporais.

ABSTRACT

Even though it be not possible to guarantee the inviolability of a computer system, techniques for intrusion detection may become a complementary resource to defend these systems from computer network attack attempts. Different sources of data can be analyzed for this purpose. The more traditional source of data is obtained monitoring directly packets passing over the network. Recently, the study of requests flows as a way to support network monitoring has started to awaken the community interest in an attempt to solve computational problems caused by the large number of trafficked packages. The flow of requests can be analyzed as time series. The present work aims to transform the monitored data in time series of requests flows and grouping them to measure the classification potential. To evaluate the proposal, the combined behavior of users and search engines crawlers on a WEB server has been studied for a month. The results obtained by grouping these time series show that the characteristics profiles of crawlers are different from other access when presenting in different groups. This result indicates that there is a potential of exploitation of classification methods for intrusion detection purposes from these groups.

Keywords: Intrusions detection, WEB requests, flows, data mining, clustering, time series.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ANN – Artificial Neural Network (Rede Neural Artificial)

AR – Association Rules (Regras de Associação)

DT – Decision Trees (Árvore de Decisão)

FTP – File Transfer Protocol (Protocolo de Transferência de Arquivos)

Gbps – Gigabit per second

HMM – Hidden Markov Models (Modelos Ocultos de Markov)

HTTP – Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)

IP – Internet Protocol (Protocolo de Internet)

KDD – Knowledge discovery in databases (Descoberta de Conhecimento em. Bancos de Dados)

PHP – PHP: Hypertext Preprocessor

SSH – Secure Shell protocol

SSW – Sum-of-squares within cluster (soma dos quadrados dentro dos grupos)

SVM – Support Vector Machine (Máquinas de Vetor de Suporte)

WEB – Abreviação para World Wide Web

1 Introdução

As técnicas tradicionais de proteção são a primeira linha de defesa para um sistema informático, porém podem ser vulneráveis ao fator humano, quer seja pela utilização errada da ferramenta ou da sua configuração, quer seja por erros de programação (PEDDABACHIGARI et al., 2007). A recente falha no mais usado software de criptografia *OpenSLL*, denominada *Heartbleed (NETWORK SECURITY, 2014)* demonstra como essas defesas podem não ser suficientes. Um invasor pode explorar falhas conhecidas, configurações inadequadas ou erros humanos para infiltrar-se em um sistema e cumprir atividades ilícitas que podem causar danos econômicos significantes para empresas e clientes.

Técnicas de detecção de intrusões foram concebidas a fim de contrastar tais ameaças. Estas monitoram as atividades do sistema protegido com o objetivo de identificar tentativas de intrusões (WU; YEN, 2009). A detecção de intrusões pode ser baseada no monitoramento de pacotes trafegados pela rede ou de fluxos de requisições (SPEROTTO; PRAS, 2011). Técnicas de detecção em nível de pacote, mais tradicionais, baseiam-se no monitoramento de grandes volumes de dados constituídos pelo tráfego de pacotes na rede do sistema monitorado. Neste contexto, faz-se necessário o uso de recursos computacionais consideráveis para o monitoramento desta grande quantidade de dados monitorada e coletada (SPEROTTO; PRAS, 2011). A detecção baseada em fluxos de requisições é mais recente. Esta agrupa em fluxos os dados obtidos, monitorando pacotes de rede ou arquivos de *logs* de requisições aos serviços acessados (TANASA; TROUSSE, 2004) e reduzindo consideravelmente o volume de dados que as técnicas de detecção de intrusões devem processar. Considerando a recente difusão dos dispositivos de redes de alta velocidade (1-10 Gbps) e o aumento progressivo do número de ataques (SPEROTTO; PRAS, 2011) é compreensível entender os motivos pelos quais se investe cada vez mais nas técnicas de detecção baseadas em fluxos.

Uma forma de representar os fluxos de dados que variam com o tempo é através de séries temporais. Estas são uma ferramenta poderosa para descrever alterações nos padrões da rede e são comumente consideradas como o método natural de observar o tráfego de rede em forma de fluxo (SPEROTTO; PRAS, 2011). Existem vários métodos de agrupamento, de medidas de similaridade e de avaliações dos resultados aplicados às séries temporais (WARREN LIAO, 2005). As técnicas de agrupamento de dados permitem a redução da quantidade de processamento necessária, identificando grupos de registros similares entre si e

diferentes se comparados com outros grupos identificados. Estas técnicas não precisam de dados previamente classificados e não têm o objetivo de classificá-los, porém seus resultados podem ser de grande auxílio quando se visa o emprego posterior de técnicas de classificação.

No presente trabalho foi investigado o uso de técnicas de agrupamento aplicadas a séries temporais constituídas a partir de fluxos de requisições, avaliando seu potencial como ferramenta de apoio a classificadores, tendo como objeto da análise os arquivos de *logs* dos servidores WEB. Duas alternativas foram identificadas no monitoramento e obtenção dos dados a serem analisados. A primeira consistia na análise de pacotes, o que permitiria o acesso integral ao conteúdo da comunicação entre o servidor e os usuários. Por outro lado, esta alternativa implicaria no processamento de um grande volume de dados e, além disso, o emprego de alguma técnica de criptografia na comunicação entre o servidor e o usuário poderia dificultar a identificação de ameaças. Foi, portanto, escolhida a segunda abordagem, que consistiu na análise dos fluxos de requisições obtidos com o processamento dos arquivos de *logs* das requisições a servidores WEB.

Este trabalho está estruturado como se segue: no capítulo 2 descreve-se o estado da arte abrangendo as técnicas de detecção de intrusões, a análise dos acessos por pacote e dos fluxos de requisições, bem como o agrupamento das séries temporais. No capítulo 3 propõe-se um processo genérico para a geração dos fluxos de requisições de cada sessão de usuário, a extração das séries temporais e seu agrupamento a partir dos dados provenientes dos arquivos de *logs* das requisições ao servidor WEB, descrevendo, desta forma, a metodologia empregada. O capítulo 4 tem como objetivo a avaliação da proposta através de um experimento real, isto é, analisando os *logs* das requisições de usuários e *crawlers* dos motores de busca em um servidor WEB Apache e comparando-se os resultados dos agrupamentos através da entropia. Por fim, no capítulo 5 os resultados são discutidos e as conclusões e as sugestões para a elaboração de trabalhos futuros são apresentados.

2 Requisições e Fluxo de Requisições

Este capítulo aborda a revisão bibliográfica no que diz respeito à análise de requisições bem como às técnicas de agrupamento de dados que variam no tempo. Desta forma, serão apresentadas as principais metodologias de detecção de intrusões, bem como os trabalhos relacionados à geração de fluxos de dados, à obtenção de séries temporais e às técnicas de agrupamento.

Ao longo da história da informática, várias metodologias foram propostas para detectar ameaças a sistemas de computadores com o objetivo comum de identificar padrões de comportamento e classificá-los. A grande maioria dos trabalhos existentes na literatura emprega alguma técnica de mineração de dados. Existe uma ampla variedade de algoritmos empregados nesta área. Estes envolvem as áreas de inteligência artificial, aprendizado de máquina, reconhecimento de padrões e de bases de dados (WENKE LEE; STOLFO; MOK, 1999). A mineração de dados é uma etapa do processo KDD (do inglês: *knowledge discovery in databases*) que tem como objetivo a descoberta do conhecimento sobre bases de dados. As demais etapas do KDD, desde o pré-processamento dos dados até a interpretação dos resultados, têm a finalidade de extrair informações úteis a partir dos dados analisados (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996). A Figura 1 mostra de forma esquemática este processo.

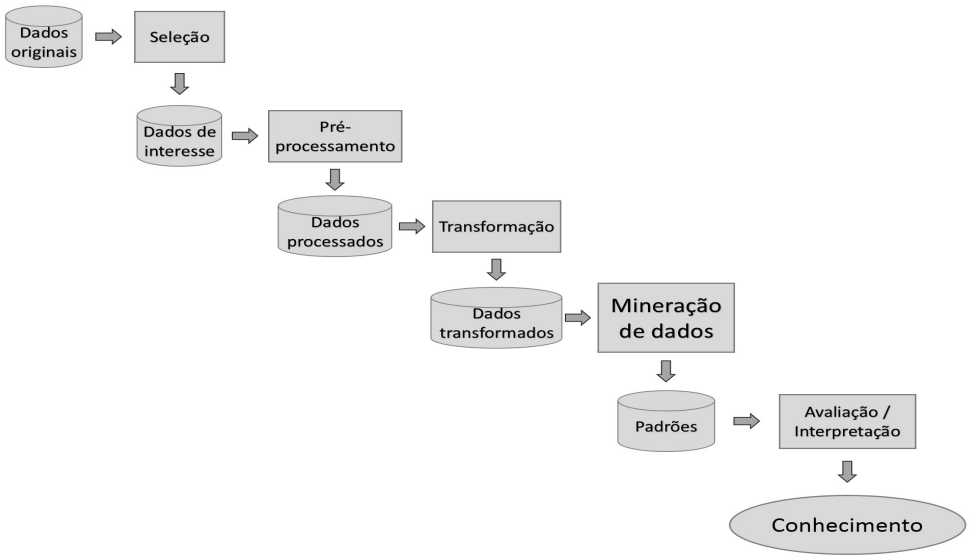

Figura 1 - Processo de Knowledge discovery database (KDD).

Uma abordagem a detecção de intrusões empregando mineração de dados é apresentada na Figura 2. Os dados monitorados são inicialmente convertidos em informações de eventos e, posteriormente, transformados em registros de conexão ou sessão, junto às informações do tipo de serviço, duração, etc. Essas informações são consumidas pelos algoritmos de mineração de dados para criarem modelos que detectem intrusões (LEE; STOLFO, 2000).

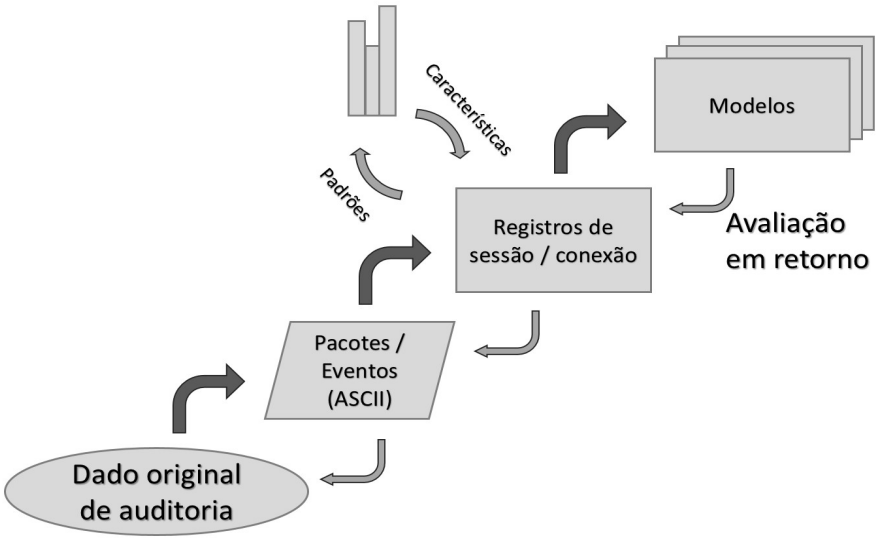

Figura 2 - Exemplo de processo de detecção de intrusões usando mineração de dados, adaptado de Lee e Stolfo (2000).

Desde 1999, o conjunto de dados kddcup 99 é o mais empregado para avaliar os métodos de detecção, tendo sido gerado pelo MIT Lincoln Labs em 1998. O mesmo laboratório gerou uma versão melhorada em 2000 baseada nas avaliações de detecção ocorridas em 1998 e 1999 e apresentada por Cieslak *et al*. (CIESLAK; CHAWLA; STRIEGEL, 2006), Thing *et al.* (L. THING; SLOMAN; DULAY, 2009) e Mirkovic *et al.* (MIRKOVIC; REIHER, 2005).

Outro conjunto de dados usado difusamente nas avaliações de detecção de intrusões é o Defcon 9. Este foi gerado durante uma competição de *hackers* em uma das conferências anuais denominada Defcon. Estes *datasets* podem ser considerados desatualizados pois as tecnologias de rede e os métodos de ataques mudaram de forma significativa (WU; YEN,

2009). Além disso, estes *datasets* perderam grande parte das informações importantes por terem sido anonimizados e a veracidade de seus dados foi reduzida por conterem dados de tráfego sintético adicionados (SHIRAVI et al., 2012).

Exemplos de ataques importantes que não podem ser contidos nestes *datasets* são aqueles que desfrutam vulnerabilidades mais recentes como a *"Integer Overflow in IPP Service Vulnerability"* que, através uma requisição HTTP POST, permite executar um código arbitrário como, por exemplo, instalar programas, apagar arquivos ou criar conta de usuários (Maiores informações estão disponíveis no endereço https://technet.microsoft.com/library/security/ms08-062). Um exemplo mais simples deste tipo de ataque é o *"Web Server Folder Traversal"*, onde qualquer usuário WEB pode executar um qualquer comando existente no servidor vulnerável (Maiores detalhes estão disponíveis em https://technet.microsoft.com/library/security/ms00-078). Este ataque desfruta de um erro no controle dos caracteres *unicode* pertencentes ao percurso da requisição WEB nos servidores IIS. Normalmente os servidores WEB apresentam páginas e outros conteúdos disponíveis apenas a partir de uma pasta dedicada a conter estas informações. Para evitar que os usuários possam acessar outras pastas do sistema operacional, os servidores WEB analisam o percurso da requisição do usuário para verificar que o conteúdo requerido esteja contido na pasta dedicada ou em suas subpastas. Porém, se o usuário utilizasse os caracteres *unicode* correspondentes à combinação "../" que indica a pasta anterior da atual, este usuário conseguiria executar programas existentes no sistema operacional sob ataque, como é apresentado na Tabela 1, na qual o usuário está tentando executar o comando *"dir"* que lista o conteúdo da pasta atual.

Tabela 1 - Exemplo de ataque para execução de um comando através o servidor IIS.

Percurso acessado	**Resposta do servidor**
http://site.com/..\../winnt/system32/cmd.exe?/c+dir	HTTP 404 - File not found
http://site.com/..%c1%9c../winnt/system32/cmd.exe?/c+dir	Directory of c:\inetpub 01/01/2002 03:45p <DIR> . 01/01/2002 03:45p <DIR> .. 0 File(s) 0 bytes 2 Dir(s) 1,234,567,890 bytes free

2.1 Análise de fluxos de Requisições

A detecção de intrusões é tradicionalmente baseada no monitoramento dos pacotes de rede que podem ser analisados por meio de assinaturas de ataques conhecidos ou empregando modelos de detecção de anomalias e técnicas de mineração de dados. Exemplos destas técnicas são as Regras de Associação (AR, do inglês: *Association Rules*) (LEE; STOLFO, 1998), as Máquinas de Vetor de Suporte (SVM, do inglês: *Support Vector Machines*) (PEDDABACHIGARI et al., 2007) (SUBBULAKSHMI et al., 2011) e as Árvores de Decisão (DT, do inglês: *Decision Trees*), também conhecidas como Árvores de Classificação (CT, do inglês: *Classification Trees*) (WU; YEN, 2009). Também existem abordagens híbridas para a detecção de anomalias como aquela que foi experimentada utilizando o *dataset* kddcup 99 analisado através DT, SVM e combinações das duas metodologias (PEDDABACHIGARI et al., 2007).

Fluxo é um conjunto de dados que passam por um ponto de observação em um determinado intervalo de tempo e que possuem propriedades em comum (QUITTEK et al., 2004). *Sperotto* et al. (2009) descreve uma metodologia para a obtenção de dados a fim de gerar *datasets* organizados em fluxos foi idealizada utilizando um servidor-isca deixado acessível por seis dias na internet por meio de um IP fixo da University of Twente. Este servidor, que foi denominado pote de mel (do inglês: *honeypot)*, foi preparado para oferecer serviços amplamente disponibilizados na internet, tais como SSH, FTP e HTTP, os quais foram alterados para oferecer vulnerabilidades conhecidas e *logging* reforçado das requisições. Como resultado, 14 milhões de fluxos de dados foram organizados e 98% foram classificados. Esta metodologia foi aprimorada, dados gerados automaticamente foram acrescentados e classificados como ataque ou normal através do *framework* de modelagem oferecido por Modelos Ocultos de Markov (HMM, do inglês: *Hidden Markov Models*). O *dataset* assim produzido foi disponibilizado no endereço http://traces.simpleweb.org.

Uma abordagem diferente foi desenvolvida com base na detecção de perfis de comportamento que podem seguir padrões de distribuição ou terem distribuição particular de acordo com o serviço (SHIRAVI et al., 2012). O objetivo dos autores foi definir e disponibilizar padrões de comportamento normal e de ataques para que estes pudessem ser usados na detecção de intrusões, bem como para que fossem reprodutíveis gerando *datasets* que pudessem ser compartilhados sem qualquer preocupação acerca dos problemas de privacidade. Os mesmos autores montaram uma rede de computadores na qual cada máquina

estava conectada por meio de um *switcher* que permitia o monitoramento de todo o tráfego da rede. Desta forma, a rede do próprio centro de pesquisa foi monitorada por quatro semanas, gerando manualmente cenários reais de ataque. O *dataset* resultante foi disponibilizado livremente no endereço http://iscx.ca/datasets. De acordo com Shiravi *et al.*, a maioria dos trabalhos na literatura elabora a modelagem das requisições HTTP com base em distribuições estatísticas bem conhecidas (SHIRAVI et al., 2012). Porém, os autores observaram que ao analisar as atividades diárias de cada usuário não era possível observar uma distribuição estatística qualquer.

Um particular tipo de fluxo pode ser obtido a partir dos *logs* das requisições dos serviços. Uma pesquisa específica sobre a mineração de dados em âmbito WEB foi apresentada por *Tanasa* e *Trousse* (2004). Nesta, os autores aplicaram técnicas de mineração de dados para processar informações de vários servidores. Em uma primeira etapa, os dados de *logs* provenientes dos servidores WEB foram reunidos, adicionando-se o nome do servidor de origem a cada registro de requisição, bem como eventuais diferenças entre os relógios de sistema dos servidores foram corrigidas. Após esta etapa, os registros menos interessantes, tais como as requisições por recursos de multimídia e atividades de *scanning* de motores de buscas, foram removidos. Em seguida, uma etapa de estruturação dos dados identificou usuários e sessões através de dados de login, IP e *browser-agent*. A partir das requisições e do mapa dos sítios WEB foram também identificados os contextos das atividades, tais como *page view*, visitações e episódios. Finalmente, os dados classificados foram gravados em uma base de dados a fim de permitir a análise de um conjunto de informações específicos sem que fosse necessário processar o *dataset* completo.

2.2 Agrupamento de Séries Temporais

O objetivo das técnicas de agrupamento é identificar estruturas entre grupos de dados não classificados, organizando os dados em grupos homogêneos que são formados de forma a maximizar a similaridade entre os itens do mesmo grupo e minimizar a similaridade entre os itens de um grupo e os itens dos outros grupos (WARREN LIAO, 2005). A maioria das ferramentas de agrupamento trabalha apenas com dados que não variam ao longo do tempo, denominados de dados estáticos.

Cada agrupamento pode ser puro (do inglês: *crisp*), se cada elemento é contido apenas em um grupo, ou nebulosa (do inglês: *fuzzy*) se cada elemento é contido em mais de um

grupo. Entre as técnicas mais conhecidas para criar agrupamentos puros o *k-means* e o *k-medoids* podem ser evidenciadas. No primeiro, cada grupo é representado pelo valor médio dos dados nele contidos, enquanto que no segundo cada grupo é representado pelo valor central do grupo. As respectivas técnicas para o agrupamento *fuzzy* são o *fuzzy c-means* e o *fuzzy c-medoids*. Métodos hierárquicos agrupam os dados em uma árvore de grupos. Este método pode iniciar com a criação de grupos de dois elementos e, em seguida, reunir os grupos criados em grupos maiores, ou ainda colocar todos os elementos em um único grupo e dividi-lo até chegar na condição desejada. O *dbscan* é um método baseado em densidade que acrescenta itens ao grupo até que o número de itens esteja próximo de alcançar uma determinada condição. Métodos baseados em grade determinam a posição do objeto em um espaço finito a fim de empregar essa informação como base para determinar o grupo do objeto. Por fim, métodos baseado em modelos definem um modelo para cada grupo do agrupamento, buscando designar os dados ao melhor modelo correspondente, utilizando, para tal, estatísticas ou redes neurais.

Diferentemente da análise de dados estáticos, as técnicas de agrupamento de dados que variam no tempo devem considerar as características dos dados, sejam eles discretos ou contínuos, uniformemente amostrados ou não, univariados ou multivariados, bem como a possibilidade de comparar séries temporais de tamanhos diferentes. Os métodos de comparação entre séries temporais buscam adaptar os algoritmos existentes para lidar com as características dos dados não estáticos, ou tentam converter os dados não estáticos para uma forma estática para que os algoritmos atualmente empregados possam manipulá-los. A primeira abordagem utiliza o dado de forma imediata e, por isso, é chamada de *raw-data-based*. A segunda abordagem é chamada de *feature-based* ou *model-based*, onde a conversão é feita extraindo atributos ou gerando parâmetros de modelos, respectivamente.

O trabalho de *Warren Liao* (2005) apresenta diferentes métodos de agrupamento, medidas de similaridade e avaliações dos resultados, com ênfase nos métodos mais usados em agrupamentos de séries temporais, organizando-os entre *raw-data-based*, *feature-based* e *model-based*. De acordo com o autor, existe apenas um trabalho que considera dados contendo erros no processo de agrupamento, há ainda poucos trabalhos que consideram dados amostrados de forma desigual e nenhum que considere séries temporais multivariadas com tamanho diferente para cada variável. Em todos os modelos, o objetivo principal é determinar um método para definir a similaridade entre os itens a serem agrupados. A principal diferença

entre o agrupamento de dados estáticos e dados de séries temporais é a forma como a similaridade é calculada. Uma vez que as similaridades tenham sido calculadas, diferentes técnicas gerais podem ser aplicadas para o agrupamento. Assim, o principal passo na compreensão das técnicas de agrupamento entre séries temporais consiste em entender as características dos dados e definir a metodologia de cálculo da similaridade.

3 Processo de Agrupamento de Fluxos de Requisições

Neste capítulo será descrita a abordagem empregada neste trabalho que propõe um processo para o agrupamento de séries temporais obtidas a partir de fluxos de requisições a serviços WEB com o objetivo de extrair informações relevantes para técnicas de classificação de intrusões. O processo é inspirado no KDD, porém é especializado na análise baseada em fluxo de séries temporais. O esquema do processo é apresentado na Figura 3.

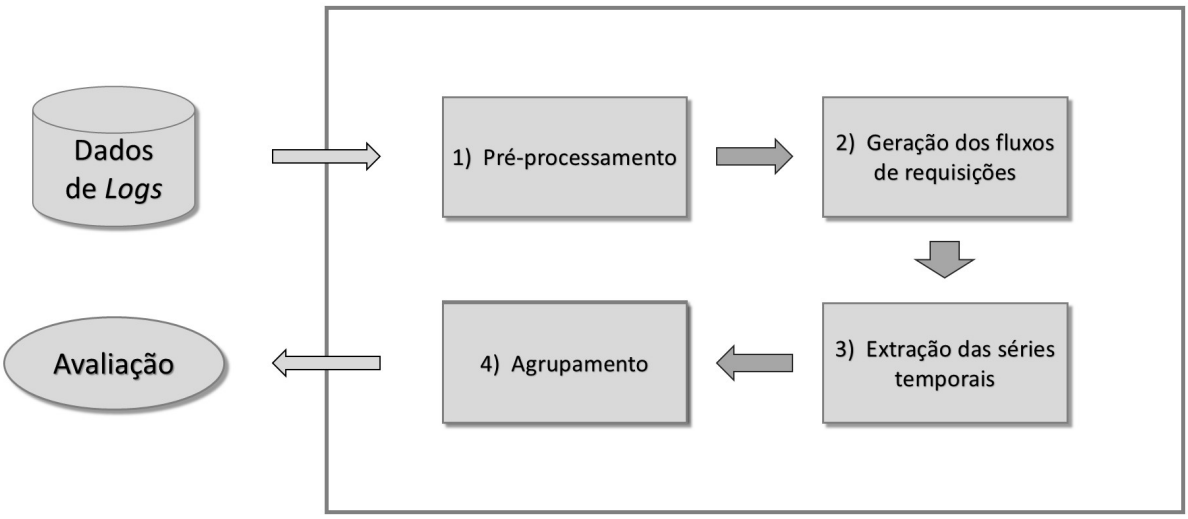

Figura 3 - Processo de agrupamento de fluxos de requisições.

A etapa 1 do processo consiste no pré-processamento dos dados. Os servidores WEB são comumente configurados para dividir os dados das requisições em diferentes arquivos de *logs*, um para cada período de tempo, como por exemplo um arquivo por semana. Este procedimento tem como objetivo facilitar o manuseio e o arquivamento destes registros. Desta forma, esta etapa é constituída pela reunião de todos os arquivos em apenas um. Além disso, a fim de garantir a consistência dos dados durante a reunião dos arquivos são aplicados procedimentos para manter a ordenação temporal, preencher dados faltantes e remover dados inconsistentes. Exemplos do conteúdo de arquivos de *logs* de servidores Apache e IIS são apresentados nas Figuras 4 e 5, respectivamente. Nota-se que cada requisição é registrada em uma linha e que os dois arquivos contêm informações parecidas.

```
1 187.53.x.x - - [24/Mar/2014:20:26:51 -0300] "GET /?page_id=xxx HTTP/1.1"
  200 31439 "https://www.google.com.br/" "Mozilla/5.0 (Windows NT 5.1)
  AppleWebKit/537.22 (KHTML, like Gecko) Chrome/xxx"
2 187.53.x.x - - [24/Mar/2014:20:26:53 -0300] "GET /wp-content/themes/xxx.png
  HTTP/1.1" 200 849 "http://coinfo.cefet-rj.br/?page_id=xxx" "Mozilla/5.0
  (Windows NT 5.1) AppleWebKit/537.22 (KHTML, like Gecko) Chrome/xxx"
3 157.55.x.x - - [24/Mar/2014:20:27:20 -0300] "GET /?feed=rss2 HTTP/1.1" 200
  8188 "-" "Mozilla/5.0 (compatible; bingbot/2.0;
  +http://www.bing.com/bingbot.htm)"
4 187.15.x.x - - [24/Mar/2014:20:30:37 -0300] "POST /moodle/login/index.php
  HTTP/1.1" 200 23288 "http://coinfo.cefet-rj.br/moodle/login/index.php"
  "Mozilla/5.0 (Windows NT 6.1; rv:27.0) Gecko/xxx"
5 186.24.x.x - - [24/Mar/2014:20:30:40 -0300] "GET /moodle/ HTTP/1.1" 200
  17130 "https://www.google.com.br/" "Mozilla/5.0 (iPhone; CPU iPhone OS 7_1
  like Mac OS X) AppleWebKit/537.51.2 (KHTML, like Gecko) Version/xxx"
```

Figura 4 - Exemplo do conteúdo de um arquivo de log do servidor Apache.

```
1 2012-01-06 09:09:27 xxx xxx 10.211.146.27 GET /blog - 80 - 94.245.127.11
  HTTP/1.1
  Mozilla/5.0+(compatible;+MSIE+9.0;+Windows+NT+6.1;+WOW64;+Trident/5.0) xxx
  http://site.supersimple.fr/cuisine-japonaise/assortiment-de-makis
  site.supersimple.fr 200 0 0 3972 544 2406
2 2012-01-06 09:09:30 xxx xxx 10.211.146.27 GET /blog/marmiton - 80 -
  94.245.127.11 HTTP/1.1
  Mozilla/5.0+(compatible;+MSIE+9.0;+Windows+NT+6.1;+WOW64;+Trident/5.0) xxx
  http://site.supersimple.fr/blog site.supersimple.fr 200 0 0 5214 519 718
3 2012-01-06 09:09:49 xxx xxx 10.211.146.27 GET /ustensiles - 80 -
  94.245.127.11 HTTP/1.1
  Mozilla/5.0+(compatible;+MSIE+9.0;+Windows+NT+6.1;+WOW64;+Trident/5.0) xxx
  http://site.supersimple.fr/blog/marmiton site.supersimple.fr 200 0 0 6897
  525 2859
```

Figura 5 - Exemplo do conteúdo de um arquivo de log do servidor IIS.

A etapa 2 envolve a geração dos fluxos de requisições com o intuito de representar a atividade de cada usuário. O fluxo de requisições é composto por um conjunto de requisições WEB geradas pelo mesmo usuário e é identificado utilizando o conceito de sessão. Este conceito é empregado pelas aplicações que rodam em servidores WEB para identificar os usuários e relacionar suas requisições durante um determinado intervalo de tempo. É comum que um usuário se conecte a um sítio WEB e faça requisições múltiplas, uma vez que o navegador acessa páginas que contêm normalmente múltiplos objetos, isto é, além da requisição da página que está sendo visitada é feita uma requisição adicional para cada imagem, script de código, folhas de estilos e outros objetos contidos nesta página. Se a aplicação WEB não gravasse as informações do usuário por um determinado tempo, sua identificação deveria ser solicitada a cada requisição.

As requisições provenientes de uma mesma conexão possuem a mesma informação de

origem (IP) e mesma descrição de navegador (*browser-agent*). O IP de origem da requisição e o *browser-agent* declarado pelo próprio navegador são registrados nos *logs* dos servidores WEB. No entanto, cada aplicação pode implementar um método próprio para gerenciar os dados da sessão e pode existir um valor de *timeout* específico para cada aplicação. Por isso, os fluxos de requisições são definidos associando as requisições provenientes do mesmo IP e contendo a mesma informação de *browser-agent* a um mesmo identificador de sessão. Além disso, cada requisição pertencente ao mesmo fluxo de requisições e, portanto, ao mesmo identificador de sessão, deve apresentar uma diferença máxima de tempo com a requisição anterior definida de acordo com o *timeout* de sessão da aplicação que está sendo monitorada. As informações usadas para a detecção dos fluxos de requisições são evidenciados nas Figuras 6 e 7. Um exemplo de *dataset* gerado na identificação dos fluxos de requisições é apresentado na Figura 8, onde cada requisição é registrada em uma linha e o identificador da sessão é evidenciado.

```
1  187.53.x.x - - [24/Mar/2014:20:26:51 -0300] "GET /?page_id=xxx HTTP/1.1"
   200 31439 "https://www.google.com.br/" "Mozilla/5.0 (Windows NT 5.1)
   AppleWebKit/537.22 (KHTML, like Gecko) Chrome/xxx"
2  187.53.x.x - - [24/Mar/2014:20:26:51 -0300] "GET /wp-content/themes/xxx.png
   HTTP/1.1" 200 849 "http://coinfo.cefet-rj.br/?page_id=xxx" "Mozilla/5.0
   (Windows NT 5.1) AppleWebKit/537.22 (KHTML, like Gecko) Chrome/xxx"
3  157.55.x.x - - [24/Mar/2014:20:27:20 -0300] "GET /?feed=rss2 HTTP/1.1" 200
   8188 "-" "Mozilla/5.0 (compatible; bingbot/2.0;
   +http://www.bing.com/bingbot.htm)"
4  187.15.x.x - - [24/Mar/2014:20:30:37 -0300] "POST /moodle/login/index.php
   HTTP/1.1" 200 23288 "http://coinfo.cefet-rj.br/moodle/login/index.php"
   "Mozilla/5.0 (Windows NT 6.1; rv:27.0) Gecko/xxx"
5  186.24.x.x - - [24/Mar/2014:20:30:40 -0300] "GET /moodle/ HTTP/1.1" 200
   17130 "https://www.google.com.br/" "Mozilla/5.0 (iPhone; CPU iPhone OS 7_1
   like Mac OS X) AppleWebKit/537.51.2 (KHTML, like Gecko) Version/xxx"
```

Figura 6 - Exemplo de detecção de fluxo de requisições no log do servidor Apache.

```
1 2012-01-06 09:09:27 xxx xxx 10.211.146.27 GET /blog - 80 - 94.245.127.11
  HTTP/1.1
  Mozilla/5.0+(compatible;+MSIE+9.0;+Windows+NT+6.1;+WOW64;+Trident/5.0) xxx
  http://site.supersimple.fr/cuisine-japonaise/assortiment-de-makis
  site.supersimple.fr 200 0 0 3972 544 2406
2 2012-01-06 09:09:30 xxx xxx 10.211.146.27 GET /blog/marmiton - 80 -
  94.245.127.11 HTTP/1.1
  Mozilla/5.0+(compatible;+MSIE+9.0;+Windows+NT+6.1;+WOW64;+Trident/5.0) xxx
  http://site.supersimple.fr/blog site.supersimple.fr 200 0 0 5214 519 718
3 2012-01-06 09:09:49 xxx xxx 10.211.146.27 GET /ustensiles - 80 -
  94.245.127.11 HTTP/1.1
  Mozilla/5.0+(compatible;+MSIE+9.0;+Windows+NT+6.1;+WOW64;+Trident/5.0) xxx
  http://site.supersimple.fr/blog/marmiton site.supersimple.fr 200 0 0 6897
  525 2859
```

Figura 7 - Exemplo de detecção de fluxo de requisições no log do servidor IIS.

```
1 000004→37.59.xx.xx→24/Feb/2014:07:18:42 -0300→Mozilla/5.0 (compatible;
  PaperLiBot/2.1; http://support.paper.li/entries/20023257-what-is-paper-li)→
  304→GET /?feed=rss2 HTTP/1.1→-→-
2 000005→179.86.xxx.xx→24/Feb/2014:07:22:24 -0300→Mozilla/5.0 (Linux; U;
  Android 4.3; en-us; GT-N7100 Build/JSS15J) AppleWebKit/xxx→404→GET
  /favicon.ico HTTP/1.1→293→http://coinfo.cefet-rj.br/moodle/login/index.php
3 000005→179.86.xxx.xx→24/Feb/2014:07:22:26 -0300→Mozilla/5.0 (Linux; U;
  Android 4.3; en-us; GT-N7100 Build/JSS15J) AppleWebKit/xxx→200→GET
  /moodle/login/index.php HTTP/1.1→23167→
  http://coinfo.cefet-rj.br/moodle/login/index.php
4 000006→66.249.xx.xxx→24/Feb/2014:07:30:14 -0300→SAMSUNG-SGH-E250/1.0
  Profile/MIDP-2.0 Configuration/CLDC-1.1 UP.Browser/xxx (compatible;
  Googlebot-Mobile/2.1; +http://www.google.com/bot.html)→200→GET
  /?page_id=1255 HTTP/1.1→28923→-
```

Figura 8 - Exemplo de detecção de dataset gerado na identificação do fluxo de requisições.

A etapa 3 do processo trata da extração das séries temporais a partir dos fluxos de requisições. Os atributos disponíveis nos fluxos gerados permitem a determinação de séries temporais de tipo diferente de acordo com os atributos considerados, tais como o número de requisições, tamanho de dados enviados, número de erros, entre outros. No intuito de comparar as séries temporais de usuários diferentes, os dados originais com as datas de inicio e término das sessões não auxiliam. Pelo contrário, a manutenção desta informação no formato original pode piorar a qualidade do resultado das técnicas de agrupamento. Por isso, o atributo de tempo de cada requisição foi alterado para representar os segundos transcorridos desde a primeira requisição da sessão.

Mesmo assim, os fluxos de requisições geradas na etapa 2 possuem uma duração total variável entre si o que dificulta a comparação entre as respectivas séries temporais. Com o

intuito de definir uma duração igual para todas as séries temporais, um valor de duração máxima fui determinado de forma que as requisições ocorridas após este tempo pudessem ser consideradas *outliers*. Para a identificação de tal valor foi utilizada a mesma regra empregada nos diagramas *box-plot* para representar os valores *outliers*. Esta regra está apresentada na equação (3.1) e é definida como o valor do terceiro quartil mais 1.5 vezes a distância inter-quartil.

$$3Q + 1.5\ IQR \tag{3.1}$$

Além disso, os fluxos de requisições gerados na etapa 2 não são correspondentes entre si no que diz respeito aos instantes de tempo de suas requisições. Isso originaria séries temporais de difícil comparação, como aquelas apresentadas na Figura 9. Para resolver este problema, as séries temporais foram discretizadas dividindo o tempo de duração total em dez intervalos e compondo os dados das requisições pertencentes a cada intervalo. Com isso, são finalmente obtidas séries temporais com o mesmo número de observações e a mesma duração e, por isso, facilmente comparáveis entre si. A Figura 10 mostra as duas séries temporais do exemplo anterior discretizadas.

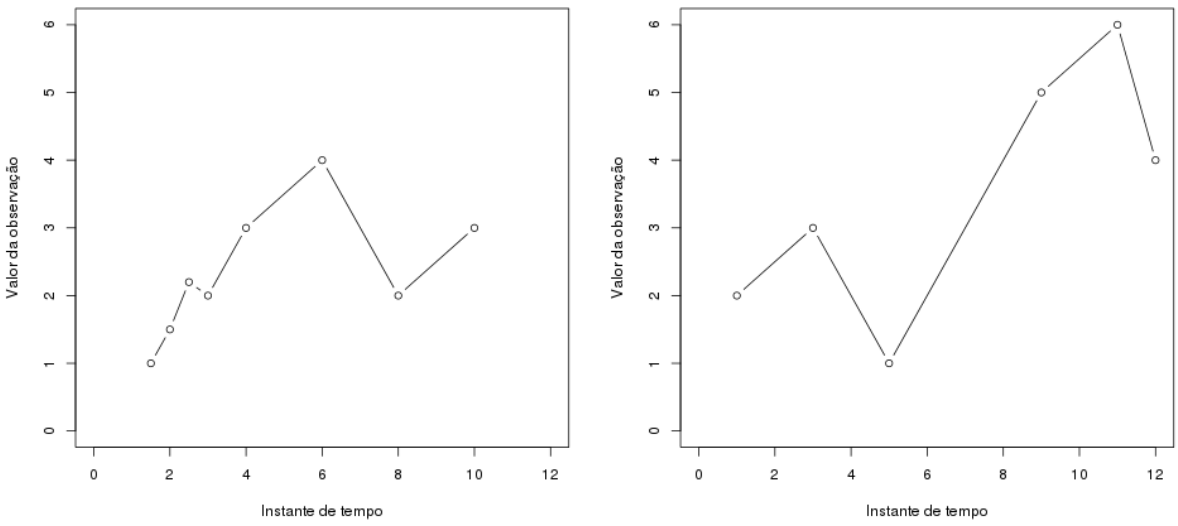

Figura 9 - Exemplo de duas séries temporais com diferente duração, número de observações e instantes de tempo das observações.

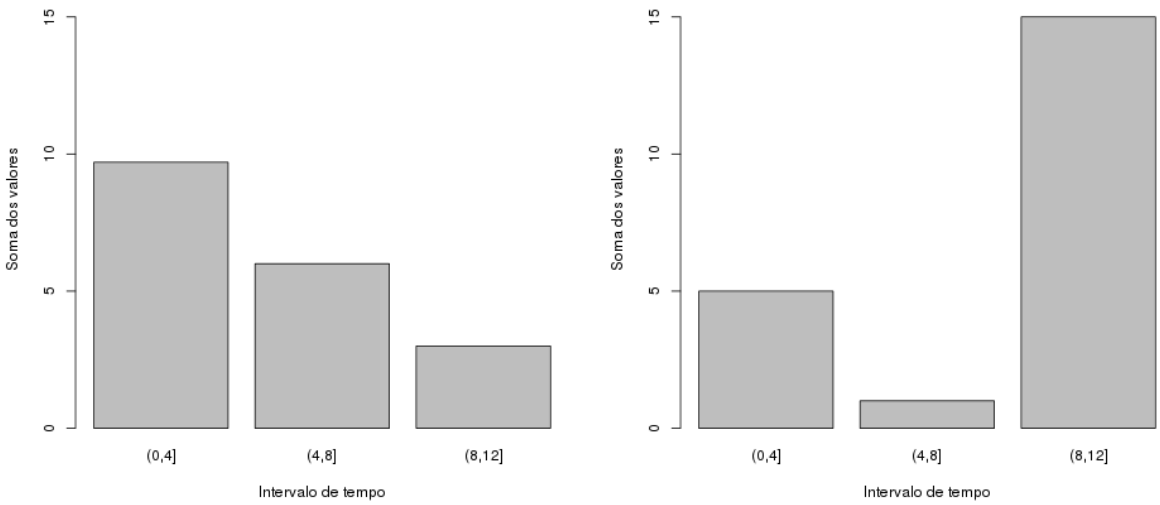

Figura 10 - Exemplo de duas séries temporais discretizadas usando a soma das observações e duração fixa para ter igual duração, número de observações e intervalos de tempo das observações.

Enfim, a etapa 4 tem como finalidade agrupar as séries temporais utilizando técnicas de agrupamentos empregadas na mineração de dados. Para os fins deste trabalho, foram escolhidas o *k-means* e o *dbscan*. O *k-means* é uma das técnicas de agrupamento mais usada e seu método é baseado na divisão em *k* partições dos objetos a serem agrupados. A escolha do valor de *k* deve ser feita previamente, já que este é parâmetro de entrada desta técnica. Uma das formas empregadas para escolher o número ideal de *k* é o método do cotovelo (do inglês: *elbow method*) que emprega o índice de soma dos quadrados dentro dos grupos (SSW, do inglês: sum-of-squares within cluster) (TIBSHIRANI; WALTHER; HASTIE, 2001). Este é um método visual que mostra uma curva formada pelo número de grupos e o respectivo valor de SSW de cada agrupamento avaliado. Desta forma, o *k-means* é empregado para gerar agrupamentos com um número variável de grupos e o valor ideal de *k* é escolhido considerando o *k* do agrupamento que tem a variação de SSW mais significativa quando relacionado aos agrupamentos com *k* imediatamente maior ou menor, como apresentado na Figura 11.

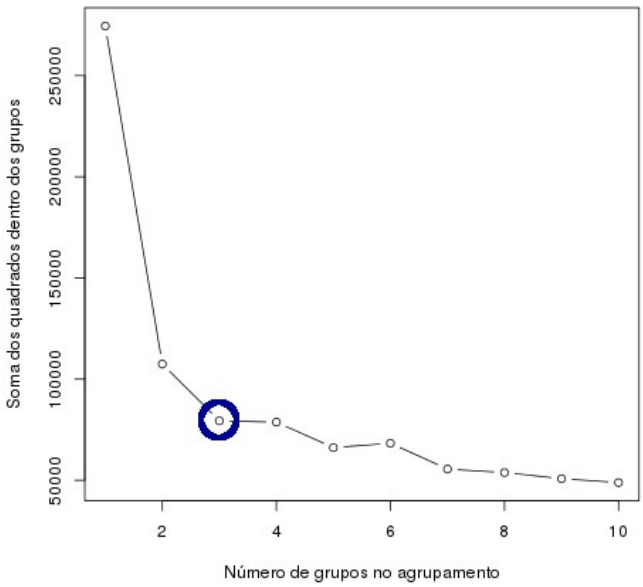

Figura 11 - Exemplo de identificação com o método do cotovelo, empregando SW.

Para os fins deste trabalho, foram avaliados os agrupamentos gerados empregando a técnica *k-means* com *k* variando entre o menor valor e duas vezes o valor ideal, valor este identificado através o método do cotovelo.

No que diz respeito ao *dbscan*, esta técnica possui características diferentes da anterior, já que não é baseada em partições e, por isso, não precisa definir a priori o número de grupos que devem ser gerados. Ao invés disto, o *dbscan* realiza o agrupamento diferenciando regiões de alta densidade separadas por regiões com baixa densidade. Os dois parâmetros de entrada do *dbscan*, *eps* e *minPts*, o influenciam diretamente nos critérios de identificação dos grupos. Estes parâmetros definem, respectivamente, o conceito de vizinhança e de densidade dos grupos a serem gerados, isto é, cada item do grupo deve estar próximo de uma quantidade mínima de itens (*minPts*) contidos em uma distância máxima determinada (*eps*). Para a escolha dos valores a serem utilizados neste trabalho foram feitas as considerações a seguir.

O valor do *minPts* foi escolhido de forma que cada grupo contivesse uma percentagem mínima de séries temporais. Neste trabalho, cada série temporal é concetualmente ligada a uma sessão de um usuário. Em um servidor onde foram identificadas, por exemplo, 1000 sessões de usuário, pode se usar um valor de *minPts* igual a 10 para que o *dbscan* não gere

grupos contendo menos de 1% de todas as sessões extraídas. Para os fins deste trabalho foi empregado o valor de 1% a fim de que o *minPts* não fosse limitante aos resultados do *dbscan*.

O valor do *eps* é escolhido considerando a distribuição das distâncias euclideanas existente entre todas as séries temporais, isto é, calculando a distância entre todas as combinações possíveis de duas das N séries temporais é obtido um conjunto de distâncias com tamanho de ordem N^2. A partir deste conjunto é possível avaliar sua distribuição e identificar um intervalo de valores de *eps* que incluam as percentagens de distâncias totais desejadas. Para os fins deste trabalho foi utilizado um intervalo de percentuais entre os valores do primeiro e do terceiro quartil, isto é, foram identificados os valores de *eps* que estivessem entre 25% e 75% das distâncias obtidas.

Finalmente, os resultados apresentados foram avaliados utilizando a entropia. A entropia é amplamente utilizada como método externo para a medição da qualidade dos grupos de acordo com o valor da característica específica que está sendo avaliada (HUI XIONG; JUNJIE WU; JIAN CHEN, 2009). Seu valor é uma forma de avaliar o grau de incerteza do grupo, de forma que quanto maior for o valor da entropia, menor será a qualidade do grupo. O valor empregado neste trabalho para a avaliação de cada agrupamento é a soma das entropias de cada grupo do agrupamento multiplicada pelo número de itens do grupo e dividida pelo número total de itens, de forma que a importância da qualidade de um dos grupos identificados é proporcional ao número de itens de cada grupo.

A probabilidade de que um item do grupo *k* seja do tipo *t* foi obtida conforme a equação (3.2) onde p_{kt} é a probabilidade, n_k é o número de itens contidos no grupo *k* e n_{tk} é o número de itens do tipo *t* contidos no grupo *k*.

$$p_{kt} = n_{tk} / n_k \tag{3.2}$$

O valor da entropia do grupo *k* foi obtido pelo somatório negativo, para cada tipo possível, da probabilidade calculada pela equação (3.2) multiplicada pelo logaritmo na base 2 da mesma probabilidade, conforme a equação (3.3).

$$e_k = - \Sigma_t \, p_{kt} \log_2 p_{kt} \tag{3.3}$$

Finalmente, a entropia total foi calculada pelo somatório das entropias de cada grupo *k* multiplicado pelo número de itens do grupo (n_k) e dividido pelo número de itens totais (n_{tot}), conforme apresentado na equação (3.4).

$$e_{tot} = \Sigma_k \, e_k \, n_k \, / \, n_{tot} \qquad (3.4)$$

Diferentemente do *k-means*, o *dbscan* não associa obrigatoriamente todos os itens a algum grupo, isto é, o resultado do *dbscan* pode considerar um conjunto de itens como *outliers* e não associá-lo a nenhum dos grupos gerados. Isso poderia invalidar a comparação entre a qualidade dos agrupamentos das duas técnicas pois a entropia do resultado do *dbscan* estaria sendo calculada apenas para os conjuntos de itens associados em grupos, descartando os *outliers*. Por causa disto, um grupo 'artificial' foi adicionado em cada agrupamento resultante do *dbscan*, associando a este grupo todos os itens que resultaram *outliers*. Isto ajuda não apenas na avaliação dos resultados através da entropia, como também auxilia na avaliação visual dos gráficos gerados a partir dos resultados dos agrupamentos.

4 Avaliação Experimental - detecção de *crawlers* Web

Neste capítulo, foi estudada a eficácia da abordagem proposta quanto ao objetivo de detectar requisições geradas pelos programas que navegam na WEB de forma automatizada e que são denominados *crawlers* WEB (STEVANOVIC; AN; VLAJIC, 2012). Exemplos de tais programas são os algoritmos de varredura automática empregados pelos motores de busca. Estes programas vasculham os sítios WEB para catalogar cada página encontrada de forma que elas possam ser acessadas pelos usuários dos motores de busca.

Os dados resultantes da análise das requisições dos *crawlers* conhecidos poderiam ser de grande importância para detectar padrões de requisições de varredura. Estes padrões podem ser muitos parecidos com aqueles gerados por *crawlers* desconhecidos, de modo que tornar-se-ia mais fácil identificar os *crawlers* mal-intencionados que operam na tentativa de ataque ou a tentativa de detecção de falhas de segurança nos servidores WEB.

Não foi possível encontrar na literatura uma lista de *crawlers* conhecidos, por isso foi empregada uma lista disponível de forma aberta no endereço http://user-agent-string.info/list-of-ua/bots. Existem outras listas parecidas na internet como, por exemplo, aquela disponível no endereço http://www.user-agents.org/allagents.xml. De qualquer forma, o foco deste experimento consistiu na identificação de um padrão de comportamento que pudesse ser característico de programas de varredura automática, sendo a lista dos *crawlers* mais comuns suficiente para a validação do resultado.

Neste experimento foram analisados os arquivos de *logs* gerados ao longo de um mês pelo servidor WEB Apache empregado em uma das máquinas da coordenação da Escola de Informática e Computação do CEFET-RJ (http://eic.cefet-rj.br). Nestes arquivos, nomeados *access_log*, o servidor Apache registra os detalhes das requisições, tais como: o *timestamp*, o IP de origem da requisição, o objeto da requisição identificado através do percurso no *filesystem*, o código HTTP resultado da requisição e o *browser-agent*, isto é, um atributo que fornece uma descrição identificativa do navegador e do sistema operacional declarada pelo usuário. O Apache não registra o conteúdo transmitido entre os dois lados da conexão em nenhum arquivo. (Maiores informações sobre estes formatos estão disponíveis no endereço http://httpd.apache.org/docs/current/mod/mod_log_config.html). Os dados dos registros dos arquivos *access_log* estão apresentados na Tabela 2.

Tabela 2 - Dados de exemplos do arquivo access_log.

IP	L	U	Instante de tempo	Requisição	Estado	Tamanho	*Referer*	*Browser-agent*
10.150.x.xx1	-	-	[24/Mar/2014:19:28:32 -0300]	"GET /moodle/pluginfile.php/2661/mod_resource/content/6/Aula-XXX.doc HTTP/1.1"	200	61952	"http://coinfo.cefet-rj.br/moodle/course/view.php?id=80"	"Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.154 Safari/537.36"
66.249.x.xx4	-	-	[24/Mar/2014:19:28:53 -0300]	"GET /robots.txt HTTP/1.1"	404	209	"-"	"Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"
66.249.x.xx2	-	-	[24/Mar/2014:19:28:53 -0300]	"GET / HTTP/1.1"	200	31074	"-"	"SAMSUNG-SGH-E250/1.0 Profile/MIDP-2.0 Configuration/CLDC-1.1 UP.Browser/6.2.3.3.c.1.101 (GUI) MMP/2.0 (compatible; Googlebot-Mobile/2.1; +http://www.google.com/bot.html)"
177.205.x.xx8	-	-	[24/Mar/2014:19:29:49 -0300]	"GET /moodle/mod/assignment/type/uploadsingle/upload.php?contextid=xxxx&userid=xxx HTTP/1.1"	200	41332	"http://coinfo.cefet-rj.br/moodle/mod/assignment/view.php?id=1753"	"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.154 Safari/537.36"
177.205.x.xx8	-	-	[24/Mar/2014:19:30:02 -0300]	"GET /moodle/theme/yui_image.php?file=2.8.2/sprite.png HTTP/1.1"	200	3745	"http://coinfo.cefet-rj.br/moodle/mod/assignment/type/uploadsingle/upload.php?contextid=xxxx&userid=xxx"	"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.154 Safari/537.36"

O Apache é configurado por padrão de forma a dividir os registros das requisições em um arquivo separado por semana, nomeados adicionando-se um sufixo de numeração que informa a ordem de criação. Desta forma, a etapa 1 do processo foi constituída pela concatenação na ordem alfabética de todos os arquivos, o que resultou em apenas um arquivo. A partir do conteúdo deste arquivo foi possível determinar a sequência de requisições geradas pelos usuários do serviço, utilizando o conceito de sessão explicado no capítulo 3. A escolha do tempo máximo entre as requisições da mesma sessão foi determinada explorando-se as características específicas do servidor WEB escolhido, bem com das aplicações hospedadas por este servidor.

A escolha do tempo máximo pode ser realizada considerando o valor de *timeout* das conexões do Apache. Um conjunto de requisições feita por um mesmo usuário ao longo de uma navegação WEB chega do mesmo IP e do mesmo navegador, utilizando a mesma conexão. Em cada página acessada são feitas múltiplas requisições. O procedimento padrão do Apache é manter as conexões abertas, mesmo que estejam inativas, por um tempo mínimo padrão de cinco segundos, com o intuito de que sejam utilizadas por outras requisições (Maiores informações estão disponíveis no endereço http://httpd.apache.org/docs/current/mod/core.html). Contudo, a maioria dos usuários acessa o mesmo sítio WEB por um tempo maior do que cinco segundos, independentemente do escopo

do acesso, então este valor foi descartado. Uma outra forma para a escolha do tempo máximo pode ser realizada considerando o módulo que gerencia as sessões e que é interno ao próprio servidor Apache. Infelizmente este módulo permanece desabilitado como procedimento padrão, o que também ocorreu com o servidor deste experimento (No endereço a seguir encontram-se maiores detalhes http://httpd.apache.org/docs/current/mod/mod_session.html).

Uma forma alternativa para a escolha do tempo máximo entre as requisições da mesma sessão pode considerar o funcionamento interno das aplicações hospedadas no servidor. O servidor empregado neste trabalho contém apenas duas aplicações WEB: o *Moodle* e o *Wordpress*. O *Moodle* é utilizado por alunos e professores a fim de planejar as atividades acadêmicas e trocar arquivos, tais como o material dos cursos e os trabalhos dos alunos. O *Wordpress* é utilizado para apresentar em detalhes o curso e as disciplinas oferecidas. Não existe um valor de *timeout* de sessão próprio do Moodle ou do Wordpress. Estas aplicações utilizam a linguagem PHP, que oferece um módulo que gerencia as sessões e que, como padrão, define em 24 minutos o tempo limite entre duas requisições (É possível obter maiores informações no endereço http://www.php.net/manual/en/session.configuration.php). Considerando estas informações, torna-se possível definir o escopo da atividade de usuário como uma sequência de requisições que chegam do mesmo computador (IP) e com o mesmo tipo de navegador (*browser-agent*) em um intervalo de tempo de no máximo 24 minutos entre as requisições subsequentes.

Uma aplicação JAVA foi desenvolvida para ler as requisições contidas em cada linha do arquivo concatenado na etapa 1, extrair os dados nele contidos usando uma expressão regular e gerar um arquivo de saída contendo as mesmas requisições rotuladas com um identificador sequencial correspondente à sessão detectada. No entanto, para que a análise sequencial fosse possível, a aplicação desenvolvida esperava que os registros do arquivo estivessem exatamente em ordem cronológica. O próprio servidor Apache acrescenta os dados em ordem cronológica de acordo com a ordem de chegada dos mesmos. Infelizmente, nem todo o conteúdo do arquivo respeitou a ordem cronológica, motivação pela qual foi necessário um pré-processamento inicial dos dados com o objetivo de ignorar algum registro inconsistente.

Diversas causas podem ter ocasionado esta inconsistência. Entre as origens mais comuns para este problema estão a troca do horário do relógio do sistema operacional e a passagem entre o horário legal e o horário solar. Uma outra possibilidade é a ocorrência de

atrasos de sincronização entre o fechamento de um arquivo semanal e a abertura do sucessivo, o que ocasionaria inconsistência ao reunir os arquivos. Estas inferências poderiam ser facilmente verificadas analisando detalhadamente os erros encontrados, porém a ocorrência deste erro foi baixa o suficiente para que os registros inconsistentes fossem ignorados.

Após garantir a ordem cronológica dos registros, foi possível aplicar a regra descrita anteriormente que define a duração da sessão do usuário como um período de tempo no qual as requisições iniciadas pelo mesmo usuário não tenham intervalos de tempo superior a 24 minutos. Conforme anteriormente explicado, um mesmo usuário é identificado através da mesma combinação que consiste no *browser-agent* e no endereço IP. Desta forma, cada requisição que respeitasse o mesmo critério foi rotulada com o mesmo identificativo de sessão. Com êxito, foi possível gerar o arquivo de saída com as sequências de requisições associadas por sessão, uma vez que cada requisição foi classificada com o seu identificativo de sessão. O formato deste arquivo de saída está apresentado na Tabela 3, onde o identificador de sessão foi evidenciado. Notam-se que os dados apresentados nas Tabelas 2 e 3 estão relacionados entre si por terem sido originados das mesmas requisições.

Tabela 3 - Dataset com informações de sessões geradas a partir dos arquivos do Apache.

Sessão	**IP**	**Instante de tempo**	***Browser-agent***	**Estado**	**Requisição**	**Tamanho**	***Referer***
8084	66.249.x.xx4	24/Mar/2014:19:28:53 -0300	Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	404	GET /robots.txt HTTP/1.1	290	-
8085	66.249.x.xx2	24/Mar/2014:19:28:53 -0300	SAMSUNG-SGH-E250/1.0 Profile/MIDP-2.0 Configuration/CLDC-1.1 UP.Browser/6.2.3.3.c.1.101 (GUI) MMP/2.0 (compatible; Googlebot-Mobile/2.1; +http://www.google.com/bot.html)	200	GET / HTTP/1.1	31074	-
8072	177.205.x.xx8	24/Mar/2014:19:29:49 -0300	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.154 Safari/537.36	200	GET /moodle/mod/assignment/type/uploadsingle/upload.php?contextid=3109&userid=844 HTTP/1.1	41332	http://coinfo.cefet-rj.br/moodle/mod/assignment/view.php?id=1753
8072	177.205.x.xx8	24/Mar/2014:19:30:02 -0300	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.154 Safari/537.36	200	GET /moodle/theme/yui_image.php?file=2.8.2/sprite.png HTTP/1.1	3745	http://coinfo.cefet-rj.br/moodle/mod/assignment/type/uploadsingle/upload.php?contextid=3109&userid=844

Uma ferramenta *ad-hoc* também foi implementada com o intuito de apresentar os dados gerados anteriormente sob a forma de histogramas a fim de permitir uma melhor visualização que possibilitasse verificar a existência de padrões de utilização. Os resultados apresentados nas Figuras 12, 13 e 14 se referem à análise temporal das sessões de usuários apresentadas na Tabela 3. Os gráficos de histogramas foram gerados a partir da biblioteca

JAVA JfreeChart, disponível no endereço http://www.jfree.org/jfreechart/.

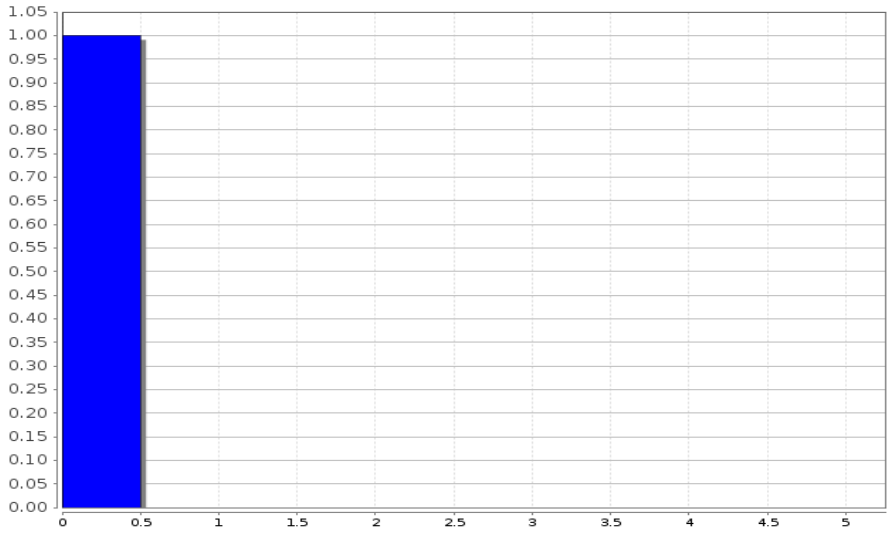

Figura 12 -Histograma da sessão 8084, em requisições versus minutos.

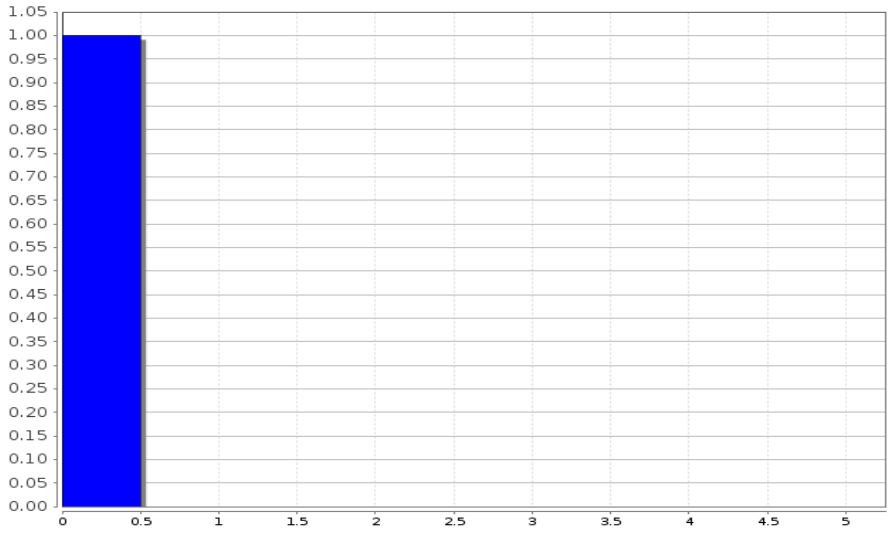

Figura 13 - Histograma da sessão 8085, em requisições versus minutos.

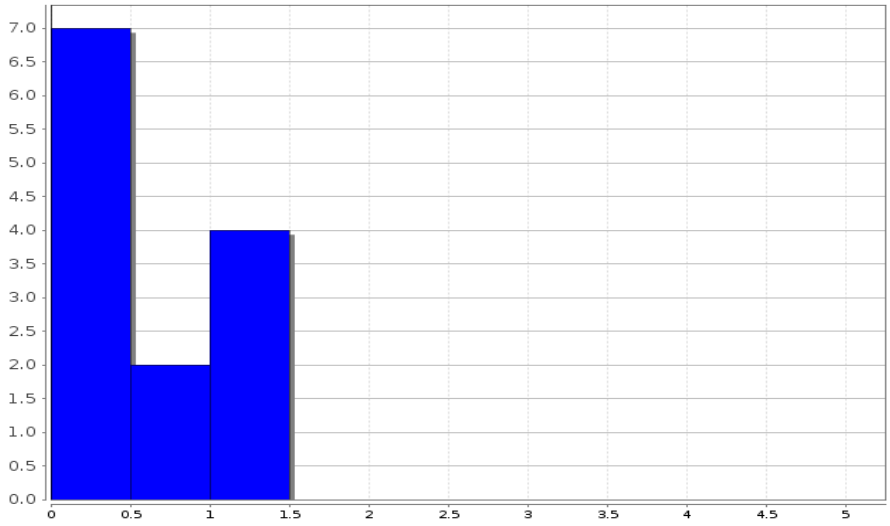

Figura 14 - Histograma da sessão 8072, em requisições versus minutos.

A grande maioria das sessões contém uma quantidade de requisições muito baixa e distribuída entre os primeiros minutos do acesso do usuário, conforme mostrado nas Figuras 12, 13 que representam as sessões de dois *crawlers* diferentes, enquanto na Figura 14 é apresentada uma sessão de usuário. A Figura 15 mostra uma das sessões que apresenta mais dados de requisições do que as apresentadas anteriormente.

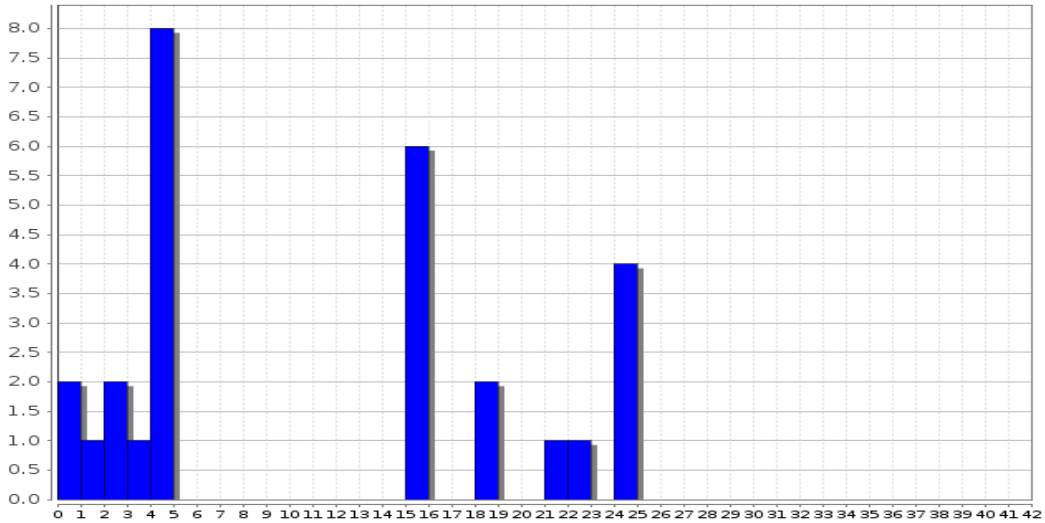

Figura 15 - Histograma de uma sessão com duração de 25 minutos, em requisições versus minutos.

Para a determinação das séries temporais foi empregada a linguagem R. Esta é uma linguagem de programação especializada em tratamentos de dados estatísticos e fornece uma grande quantidade de módulos adicionais para a mineração de dados. Esta ferramenta possibilitou uma rápida experimentação de diferentes análises e a imediata visualização dos resultados. Os dados de sessão, gerados pela aplicação anteriormente descrita, foram importados e o tipo dos atributos foi convertido, onde possível, de texto para data ou número, enquanto o atributo de tempo de cada requisição foi alterado para que informasse os segundos passados desde a primeira requisição da sessão. Para que fosse obtida a identificação de um valor de duração igual para todas as séries temporais foi gerado o *box-plot* da duração das sessões que está apresentado na Figura 16.

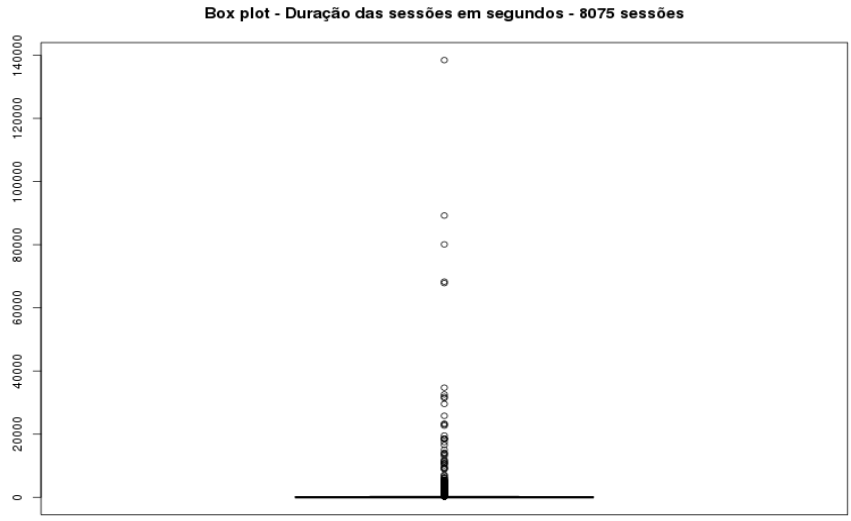

Figura 16 - Box-plot da duração das sessões, em segundos.

A Figura 17 apresenta um gráfico de barras representando o número de sessões por intervalo de duração em segundos. Esta figura possibilita uma melhor compreensão da distribuição da duração das sessões, evidenciando que a grande maioria das sessões é de breve duração.

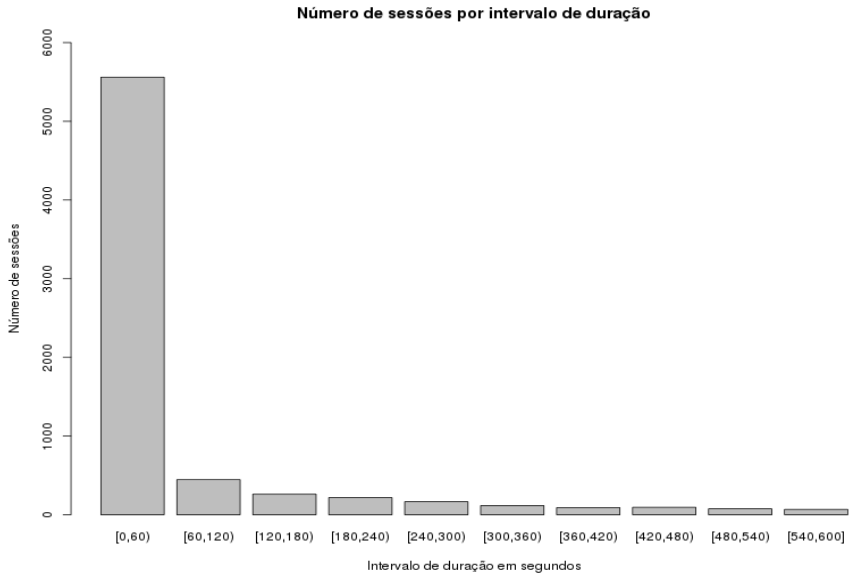

Figura 17 - Número de sessões por intervalo de duração.

Os *outliers* foram removidos seguindo a mesma regra empregada para representá-los nos *box-plots*, isto é, utilizando a fórmula apresentada na equação (3.1), o que resultou em uma duração máxima de 325 segundos.

Uma vez definido o limite de duração das séries temporais a serem geradas, foi necessário discretizar os intervalos de tempo para ter uma correspondência de tempo entre as requisições de diferentes séries temporais. Para resolver este problema, o tempo de duração das sessões foi dividido em dez intervalos de tempo de 36 segundos, no qual determinou-se como último intervalo aquele composto pelo somatório dos atributos de cada requisição existente a partir do 324º segundo (9 vezes 36) dividido pelo tempo faltante por sua vez dividido pelos 36 segundos de duração de cada intervalo. O cálculo é apresentado na equação (4.1), onde st_{10} é o decimo valor da série temporal, r_i se refere a cada requisição feita após 324 segundos e *t* é a duração total da sessão.

$$st_{10} = (\Sigma_{324 \rightarrow t}\, r_i\,) / (\,(t - 324) / 36) \qquad (4.1)$$

A relação entre o número de sessões e sua duração em segundos, de acordo com a atual definição de intervalos de tempo, é representada no gráfico de barras apresentado na Figura 18.

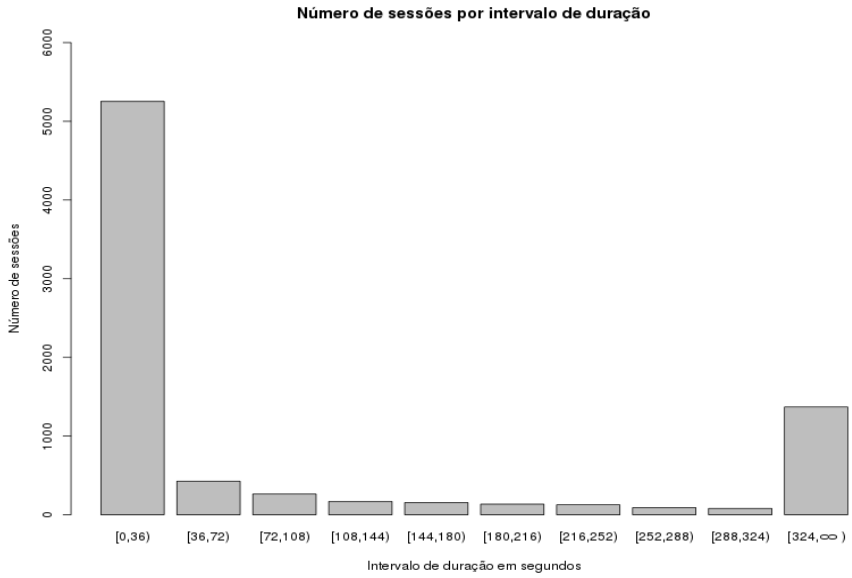

Figura 18 - Gráfico de barras com o número de sessões por intervalo de duração de 36 segundos.

Com base nestas definições foram extraídas 8075 séries temporais correspondentes às sessões geradas na etapa 2. Uma única estrutura de dados, representada na Figura 19, foi gerada para que fosse esta de fácil manuseio no processo de agrupamento usando a linguagem R. Nesta estrutura, cada elemento que representa uma sessão de usuário contém o identificador numérico sequencial da sessão, o número total de suas requisições, o total de duração em segundos, o tempo inicial da sessão, o tempo final da sessão e o código de usuário detectado, isto é, 0 se não for um *crawler* conhecido, 1 para o crawler do Google, 2 para o crawler do Bing, 3 para o crawler do Yahoo e 4 para os outros crawlers conhecidos. Por fim, os últimos 10 atributos representam a série temporal usada neste experimento, os quais contêm o número de requisições por intervalo de tempo (atributos de a01 até a10). O atributo a10 representa o número de requisições a partir do 324º segundo até o final da sessão dividido pelo intervalo de tempo, conforme a equação (4.1). Observa-se que apenas os atributos de a01 até a10 foram utilizados como entrada para as técnicas de agrupamentos. O atributo '*robot*' nunca foi utilizado, pois seu emprego foi reservado para verificar e avaliar a qualidade dos agrupamentos gerados.

	session	tot_req	tot_dur	ini_tim	end_tim	robot	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10
1	1	12	1369	0	1369	1	1	0	0	0	0	0	0	0	0	0.379
2	2	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0.000
3	3	6	36	77	113	0	5	1	0	0	0	0	0	0	0	0.000
4	4	1	0	256	256	4	1	0	0	0	0	0	0	0	0	0.000
5	5	2	2	478	480	0	2	0	0	0	0	0	0	0	0	0.000
6	6	2	49	948	997	1	1	1	0	0	0	0	0	0	0	0.000
7	7	1	0	1097	1097	4	1	0	0	0	0	0	0	0	0	0.000
8	8	14	754	1905	2659	0	3	0	0	0	0	0	0	0	0	0.921
9	9	8	3185	2176	5361	1	2	0	0	0	0	0	0	0	2	0.050
10	10	4	58	2457	2515	0	3	1	0	0	0	0	0	0	0	0.000
11	11	2	21	2458	2479	0	2	0	0	0	0	0	0	0	0	0.000
12	12	14	70	2501	2571	0	9	5	0	0	0	0	0	0	0	0.000
13	13	2	2	2789	2791	2	2	0	0	0	0	0	0	0	0	0.000
14	14	3	24	2895	2919	4	3	0	0	0	0	0	0	0	0	0.000
15	15	1	0	3147	3147	0	1	0	0	0	0	0	0	0	0	0.000
16	16	1	0	3148	3148	0	1	0	0	0	0	0	0	0	0	0.000
17	17	2	175	3199	3374	2	1	0	0	0	1	0	0	0	0	0.000
18	18	4	8	3916	3924	0	4	0	0	0	0	0	0	0	0	0.000
19	19	6	25	4184	4209	0	6	0	0	0	0	0	0	0	0	0.000
20	20	7	509	4510	5019	0	4	1	0	0	0	0	0	0	0	0.389
21	21	26	674	4599	5273	0	1	3	2	2	4	2	0	4	0	0.823
22	22	2	1	4692	4693	0	2	0	0	0	0	0	0	0	0	0.000
23	23	25	105	4879	4984	0	15	6	4	0	0	0	0	0	0	0.000
24	24	115	2630	6104	8734	0	9	6	13	1	11	3	3	4	2	0.984
25	25	1	0	6158	6158	4	1	0	0	0	0	0	0	0	0	0.000
26	26	1	0	6445	6445	4	1	0	0	0	0	0	0	0	0	0.000
27	27	2	5	6463	6468	0	2	0	0	0	0	0	0	0	0	0.000
28	28	4	141	6777	6918	0	2	0	0	2	0	0	0	0	0	0.000
29	29	3	661	7018	7679	1	1	0	1	0	0	0	0	0	0	0.107
30	30	17	31	7369	7400	0	17	0	0	0	0	0	0	0	0	0.000

Figura 19 - Data-frame usado na linguagem R como estrutura de dados para extração das séries temporais. Os atributos correspondentes às séries temporais compostas pelo número de requisições no intervalo de tempo são evidenciados a direita.

Na etapa 4 foram gerados os agrupamentos empregando as séries temporais compostas pelo número de requisições de cada sessão nos intervalos de tempo definidos anteriormente. No que diz respeito ao *k-means*, o gráfico do cotovelo foi gerado para identificar o parâmetro de *k* mais significativo. Conforme apresentado na Figura 20, o valor 5 resultou ser o mais indicado como parâmetro de entrada do agrupamento. Conforme a metodologia definida no capítulo 3, foram gerados agrupamentos de 2 até 10 grupos para avaliar as diferentes qualidades dos agrupamentos resultantes.

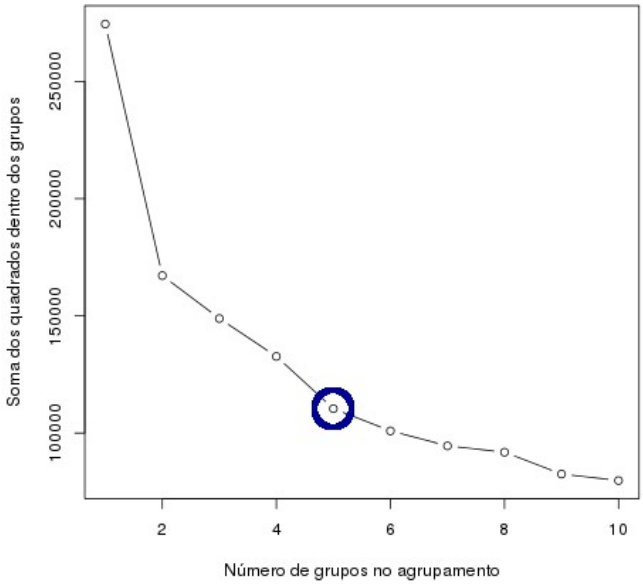

Figura 20 - Identificação do melhor número de k através o método do cotovelo, empregando SW.

Os resultados foram representados de forma a possibilitar a visualização das séries temporais associadas a cada grupo. Os gráficos representados nas Figuras 21, 22 , 23, 24, 25, 26, 27, 28 e 29 apresentam respectivamente os agrupamentos com 2, 3, 4, 5, 6, 7, 8, 9 e 10 grupos. Em cada gráfico foram representadas todas as séries temporais divididas em cores de acordo com o grupo de associação, isto é, todas as séries temporais do mesmo grupo foram representadas com a mesma cor. O cálculo do 1º quartil, mediana e 3º quartil de cada grupo também foi representado para cada intervalo de tempo, utilizando três linhas em destaque com a mesma cor do grupo.

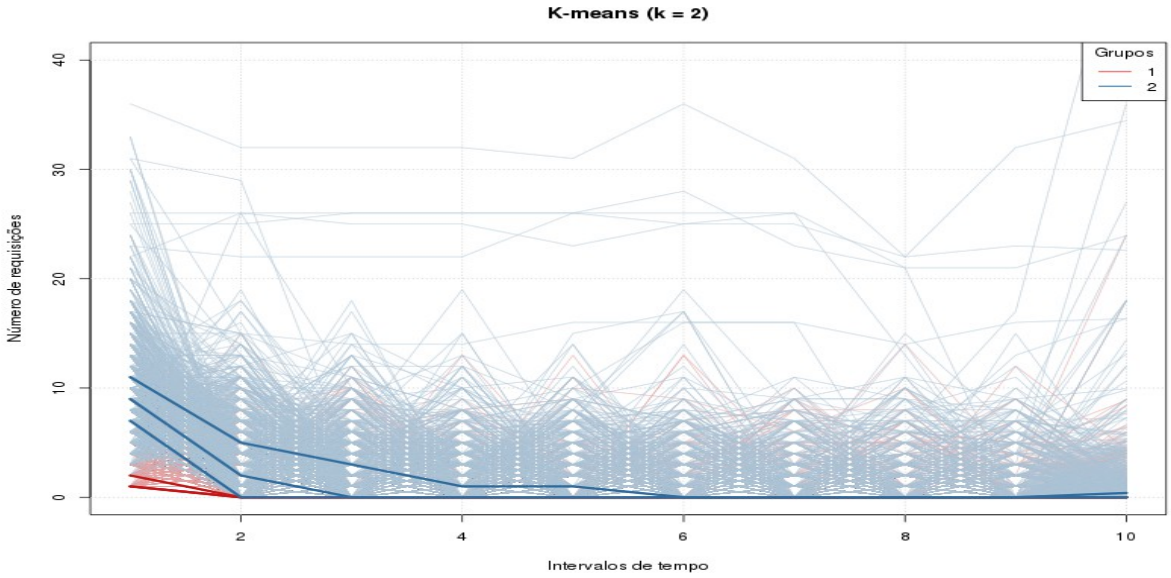

Figura 21 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. O agrupamento foi gerado usando k-means com k = 2.

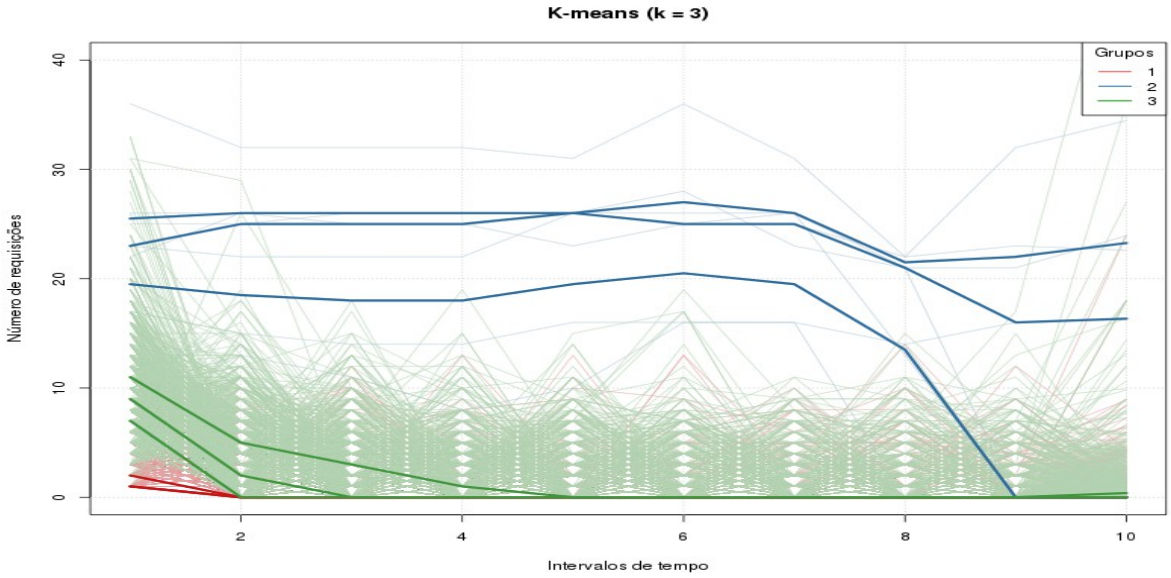

Figura 22 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. O agrupamento foi gerado usando k-means com k = 3.

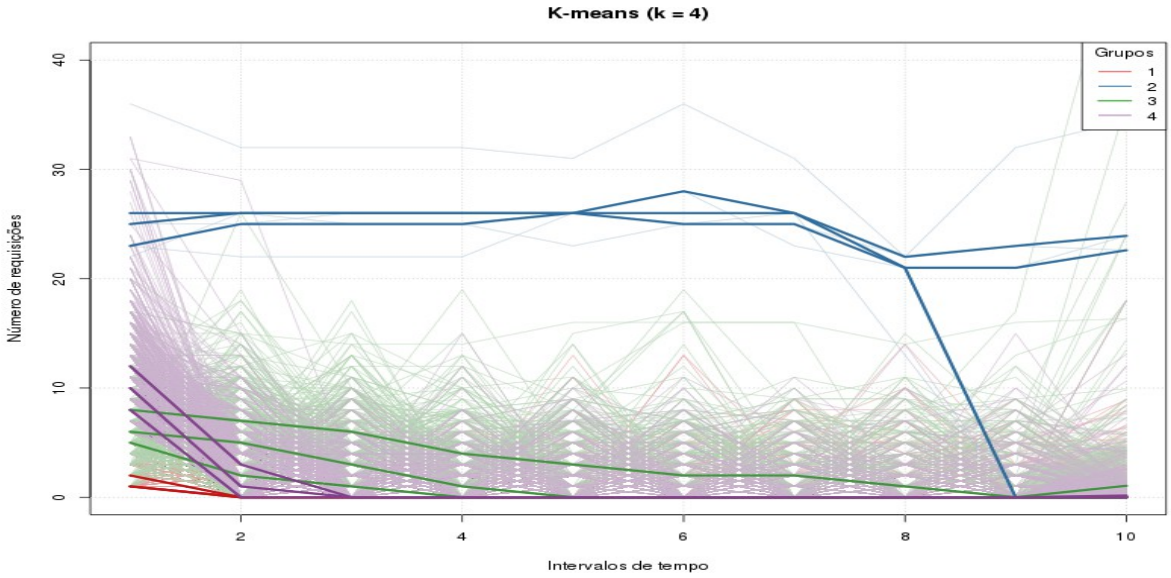

Figura 23 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. O agrupamento foi gerado usando k-means com k = 4.

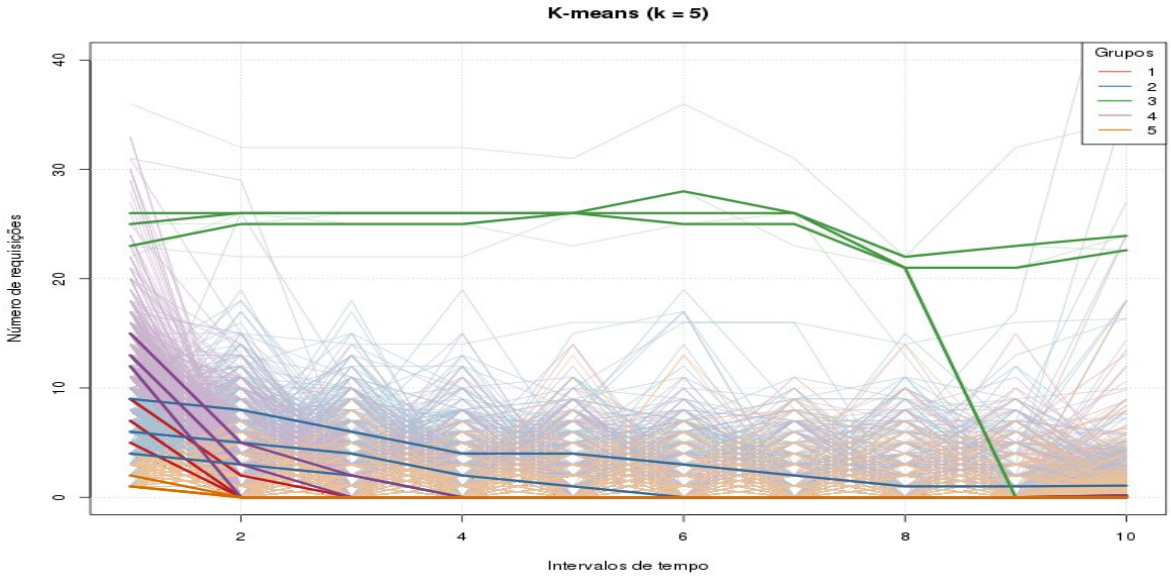

Figura 24 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. O agrupamento foi gerado usando k-means com k = 5.

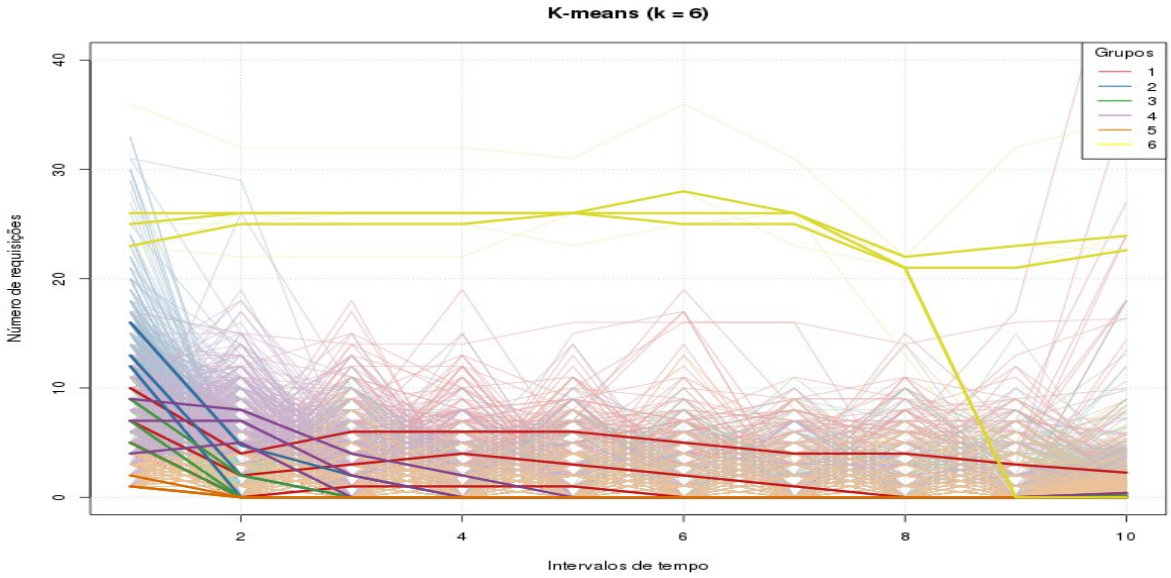

Figura 25 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. O agrupamento foi gerado usando k-means com k = 6.

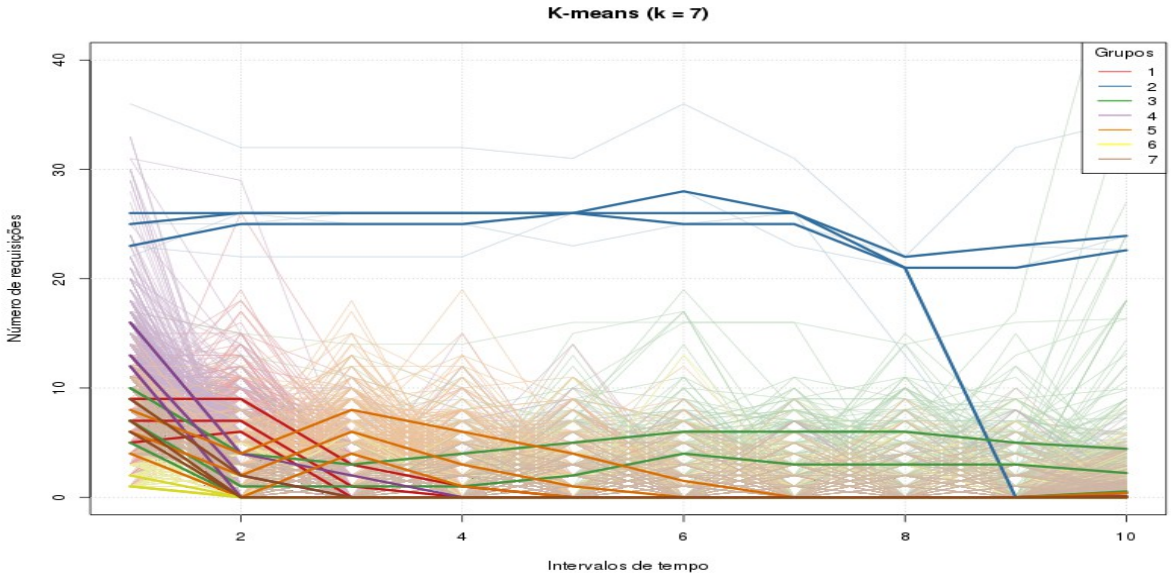

Figura 26 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. O agrupamento foi gerado usando k-means com k = 7.

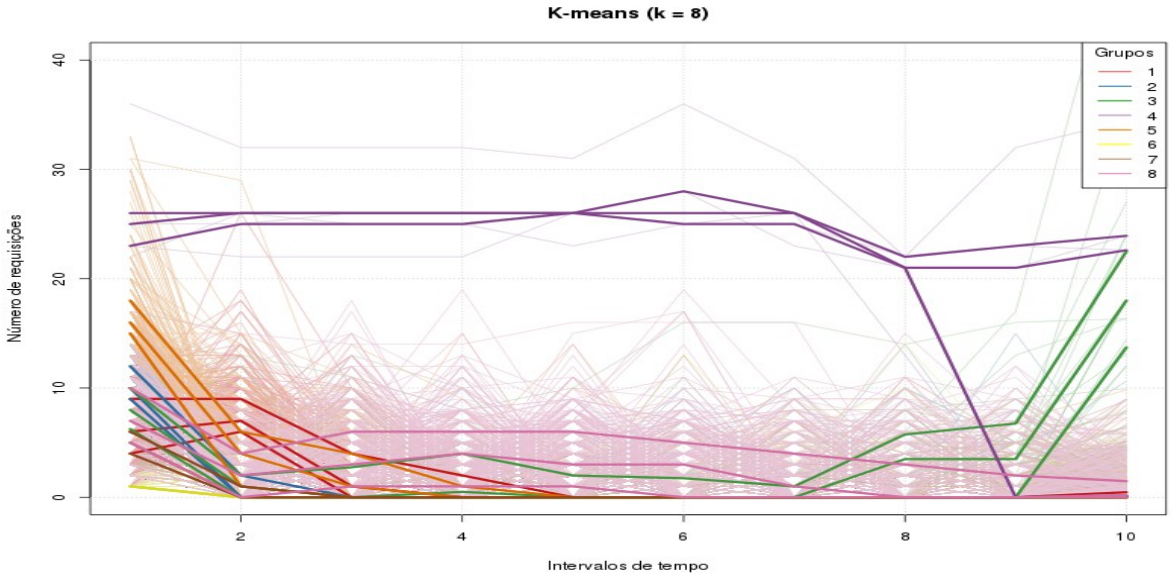

Figura 27 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. O agrupamento foi gerado usando k-means com k = 8.

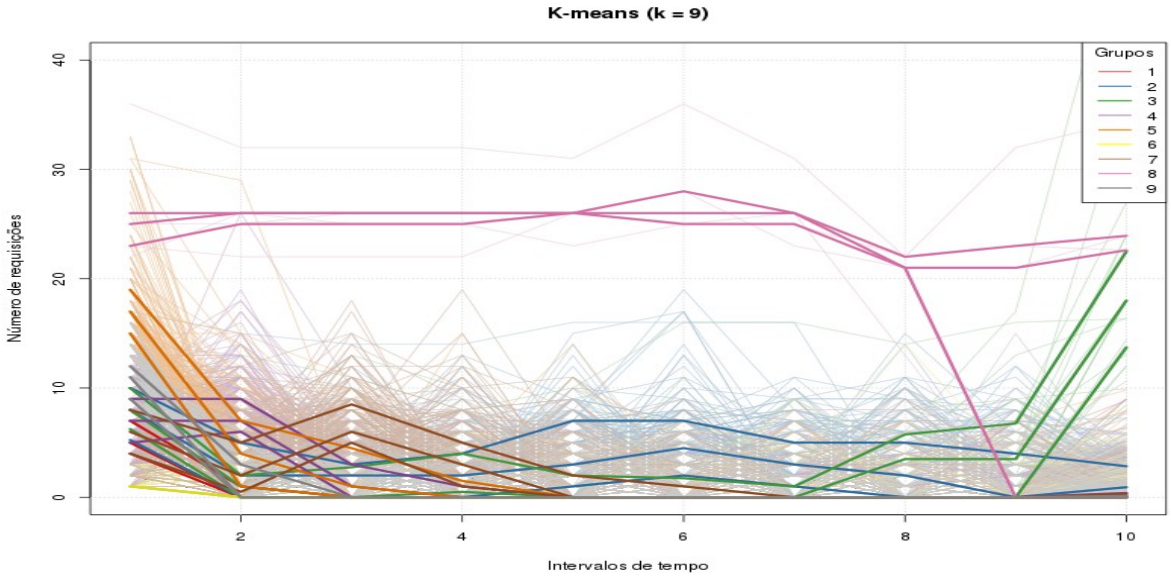

Figura 28 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. O agrupamento foi gerado usando k-means com k = 9.

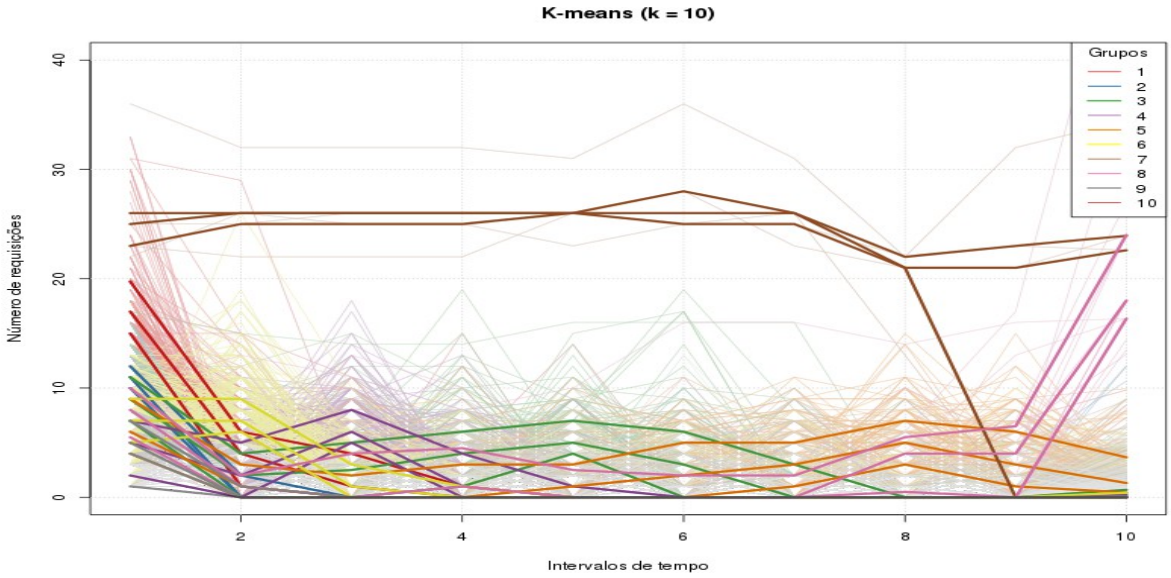

Figura 29 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. O agrupamento foi gerado usando k-means com k = 10.

Para avaliar o agrupamento em relação à detecção do perfil de comportamento dos *crawlers* conhecidos, os dados foram representados mostrando a proporção tanto do número de sessões incluídas em cada grupo quanto a quantidade de *crawlers* que foram incluídos em cada grupo. Os resultados apresentados na Figura 30 mostram a proporção dos *crawlers* nos agrupamentos de 2 até 10 grupos.

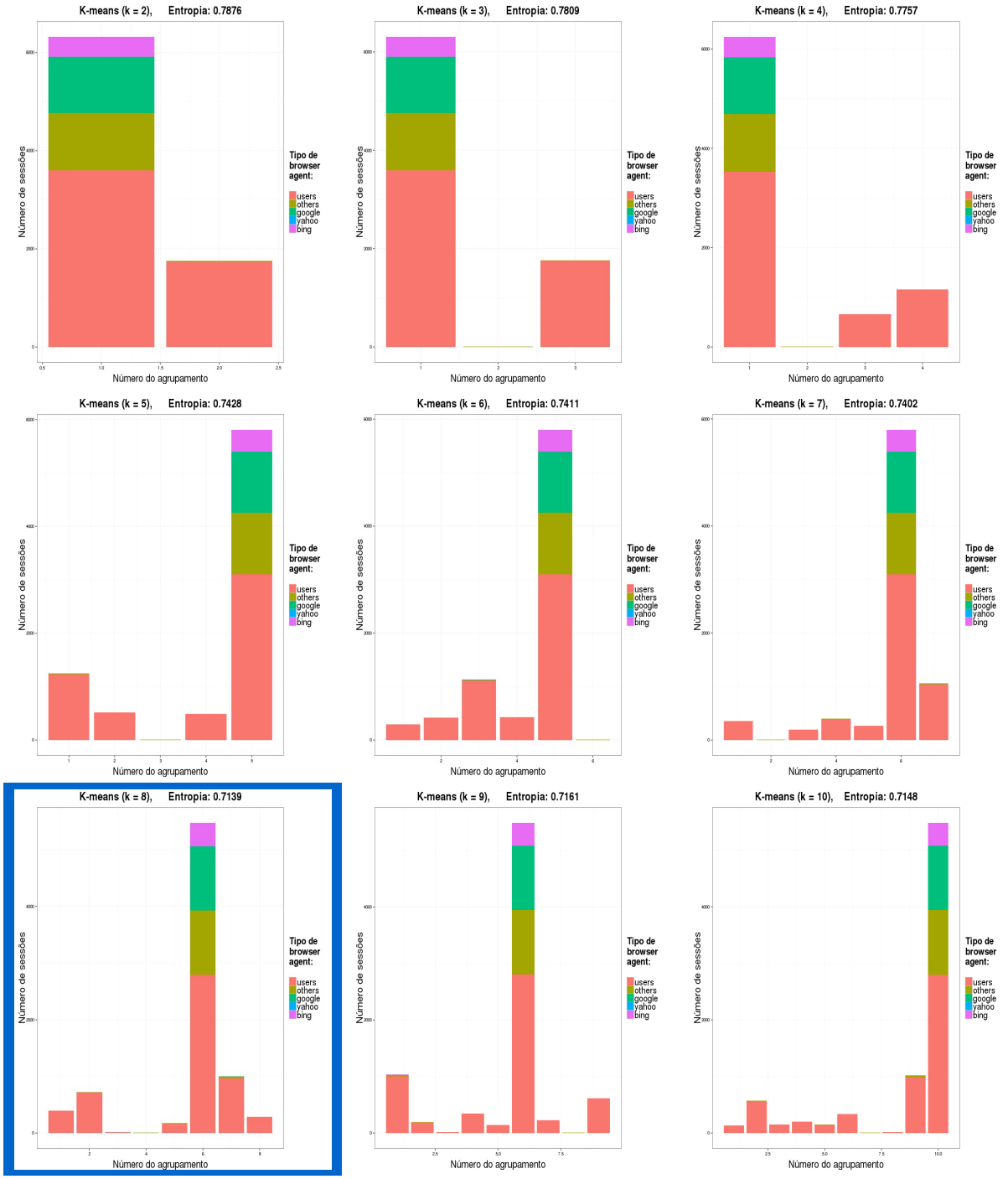

Figura 30 - Proporção de crawlers conhecidos detectados no agrupamento gerado usando k-means com k de 2 até 10. O resultado evidenciado possui a melhor entropia.

Todos os agrupamentos gerados usando o *k-means* associaram a quase totalidade dos *crawlers* conhecidos ao mesmo grupo. O aumento do valor de k de 2 até 10, a fim de aumentar o número de grupos a serem gerados pelo *k-means*, pareceu ocasionar apenas a sub-

divisão adicional dos grupos que continham as sessões dos *crawlers* conhecidos.

Outra técnica de agrupamento avaliada neste trabalho é o *dbscan*. Os parâmetros de entrada *eps* e *minPts* foram escolhidos conforme a metodologia definida no capítulo 3. Conforme explicitado anteriormente, a escolha do valor de *minPts* igual a 1% foi definida como a percentagem mínima das séries temporais que o *dbscan* deve considerar para que um grupo possa ser criado. Esta percentagem corresponde a um *minPts* do valor de 80. No que diz respeito ao *eps*, foi analisada a distribuição das distâncias euclideanas calculadas para todas as combinações de duas séries temporais, apresentadas no gráfico de barras da Figura 31, onde nota-se a existência de um grande número de combinações de distâncias com valor nulo.

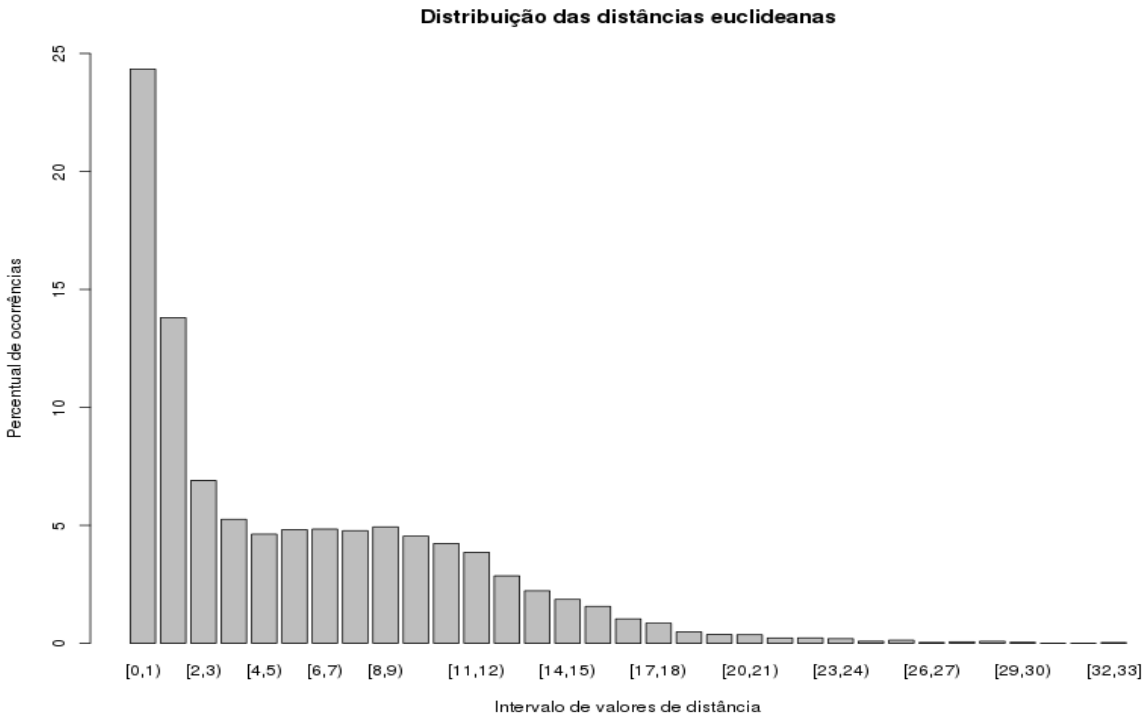

Figura 31 - Distribuição das distâncias euclidianas existentes entre as combinações de séries temporais.

Da mesma forma que foi feito para o *k-means*, os valores de *eps* foram escolhidos para gerar nove agrupamentos, conforme a metodologia definida no capítulo 3, isto é, foram gerados valores de *eps* correspondentes a nove intervalos de percentagens distribuídas

uniformemente entre os valores de 25% e 75%, obtendo-se os valores representados na Tabela 4. Nesta tabela, observa-se que o valor de *eps* correspondente à 25% é igual ao valor do percentil posterior e, por isso, o valor de *eps* correspondente à 25% foi alterado para ser 0.5.

Tabela 4 - Valores de eps obtidos de acordo com os correspectivos percentis das diferencias euclideanas.

Percentil	**25%**	**31.25%**	**37.5%**	**43.75%**	**50%**	**56.25%**	**62.5%**	**68.75%**	**75%**
eps	1	1	1.433117	2.249669	3.741657	5.000168	6.385171	7.810250	9.001840

Os resultados são apresentados nas Figuras 32, 33, 34, 35, 36, 37, 38, 39, 40 e 41.

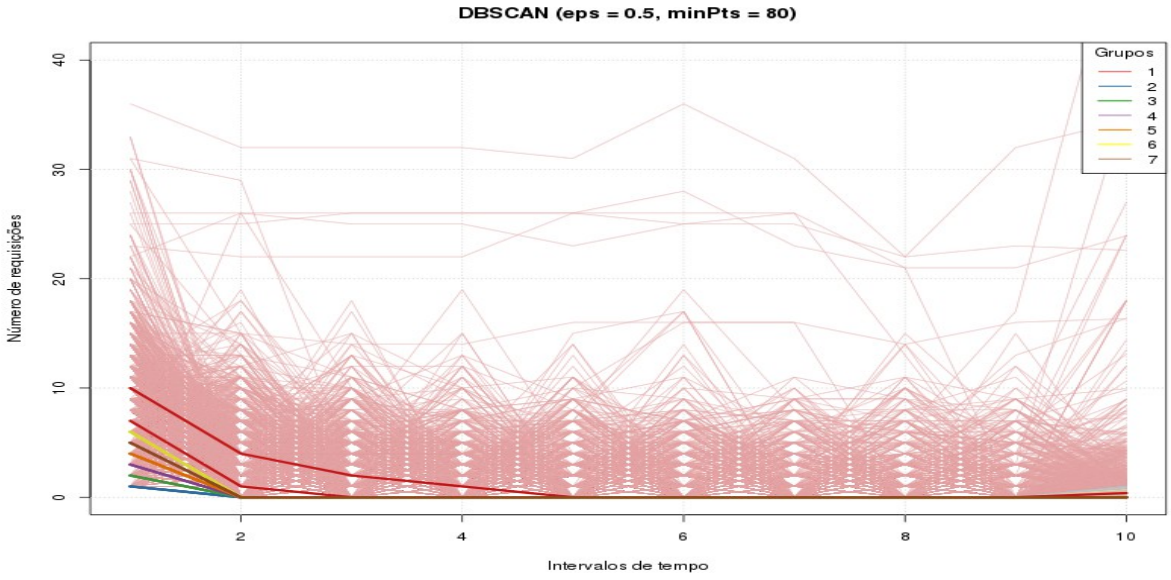

Figura 32 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. Dbscan obtido com eps = 0.5 e minPts = 80.

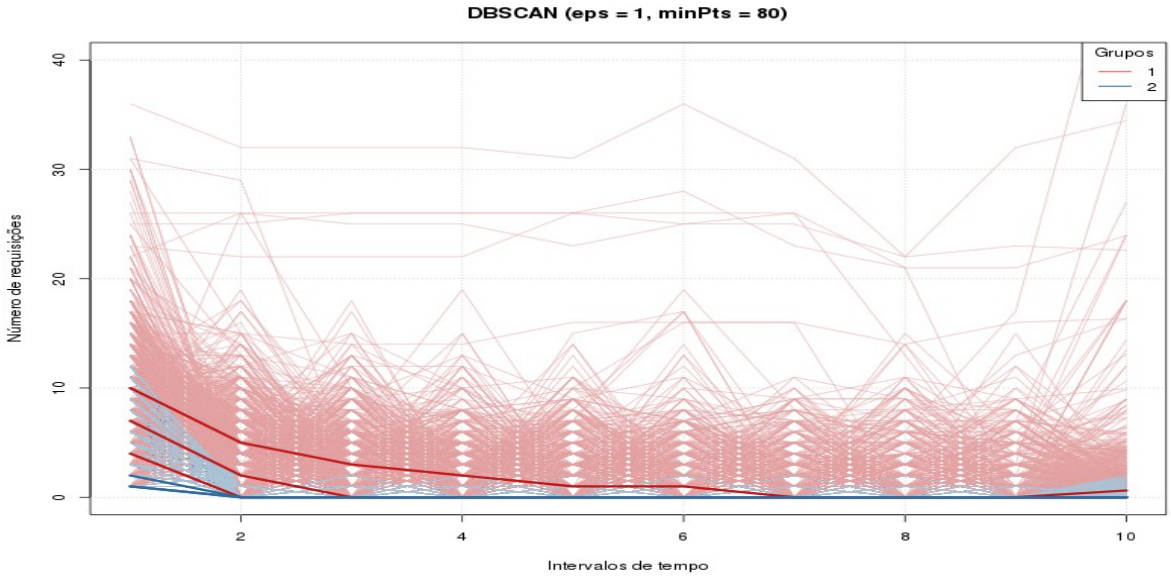

Figura 33 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. Dbscan obtido com eps = 1 e minPts = 80.

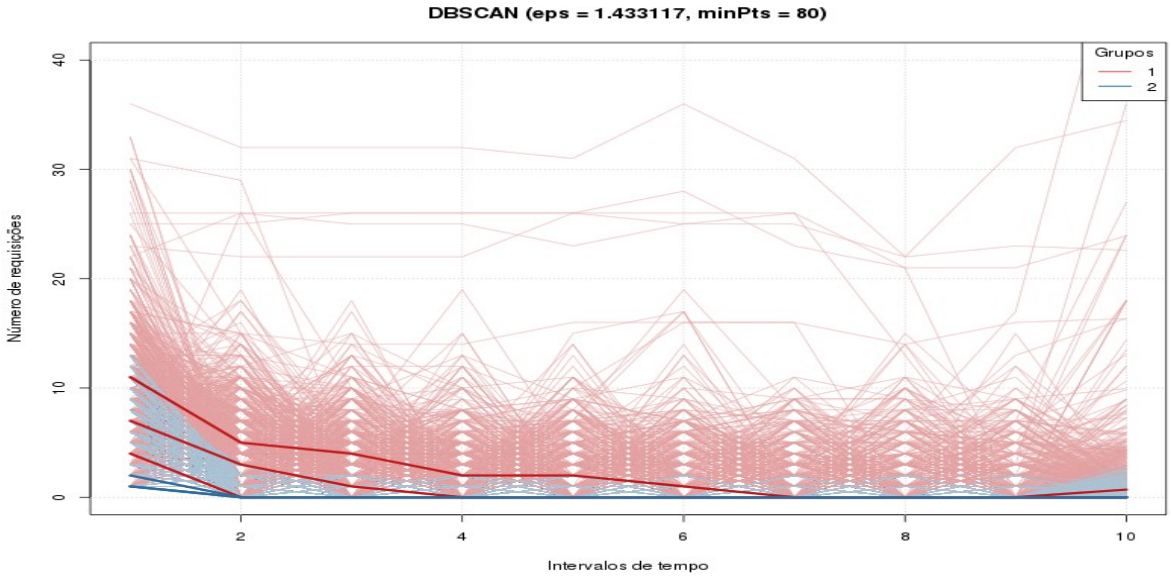

Figura 34 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. Dbscan obtido com eps = 1.433117 e minPts = 80.

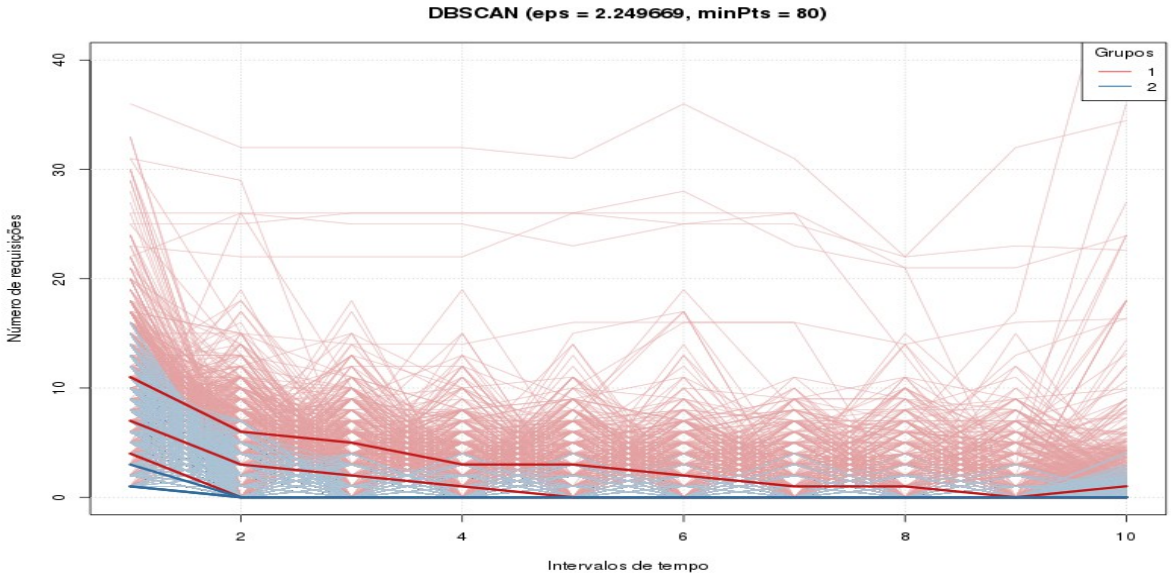

Figura 35 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. Dbscan obtido com eps = 2.249669 e minPts = 80.

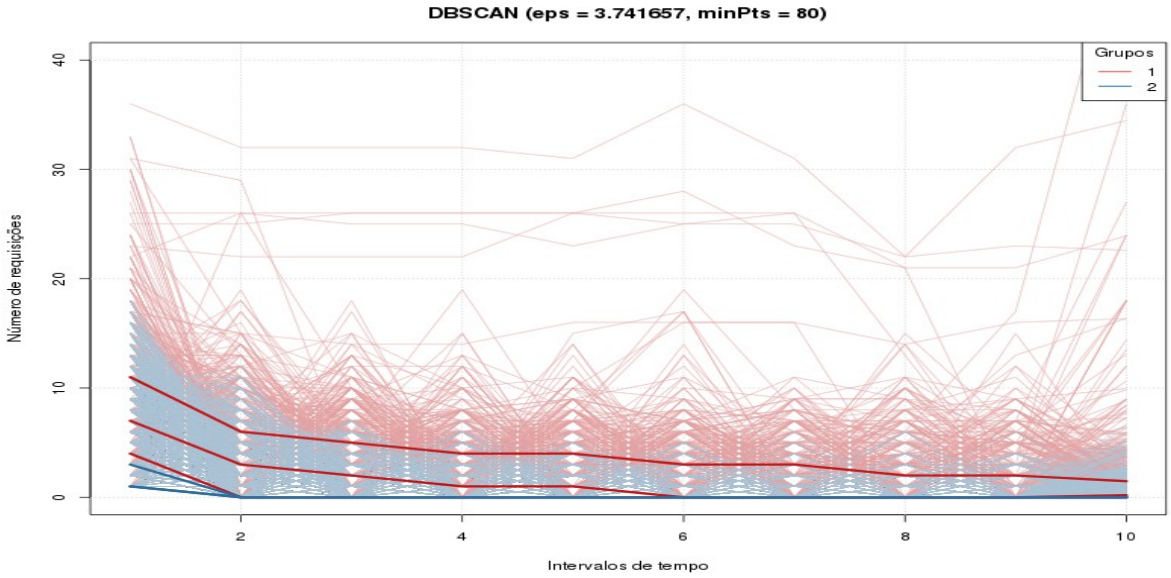

Figura 36 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. Dbscan obtido com eps = 3.741657 e minPts = 80.

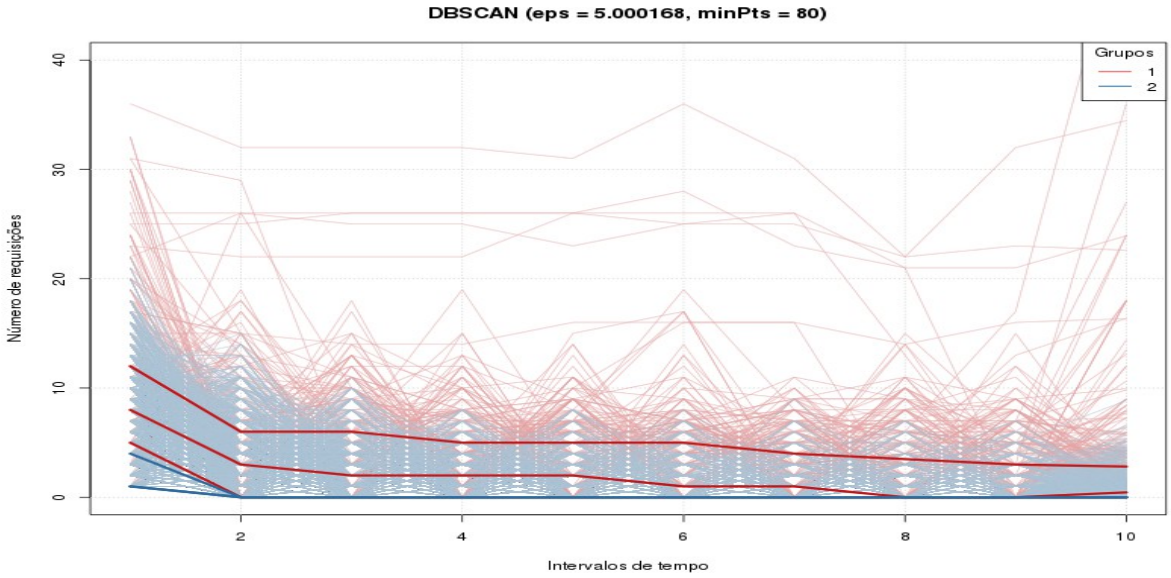

Figura 37 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. Dbscan obtido com eps = 5.000168 e minPts = 80.

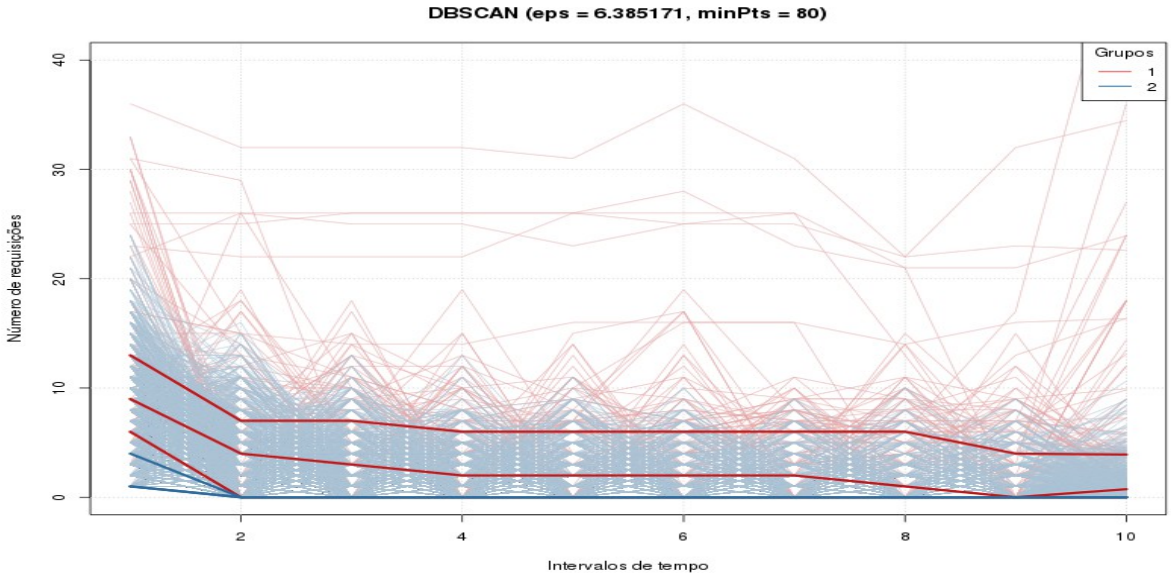

Figura 38 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. Dbscan obtido com eps = 6.385171 e minPts = 80.

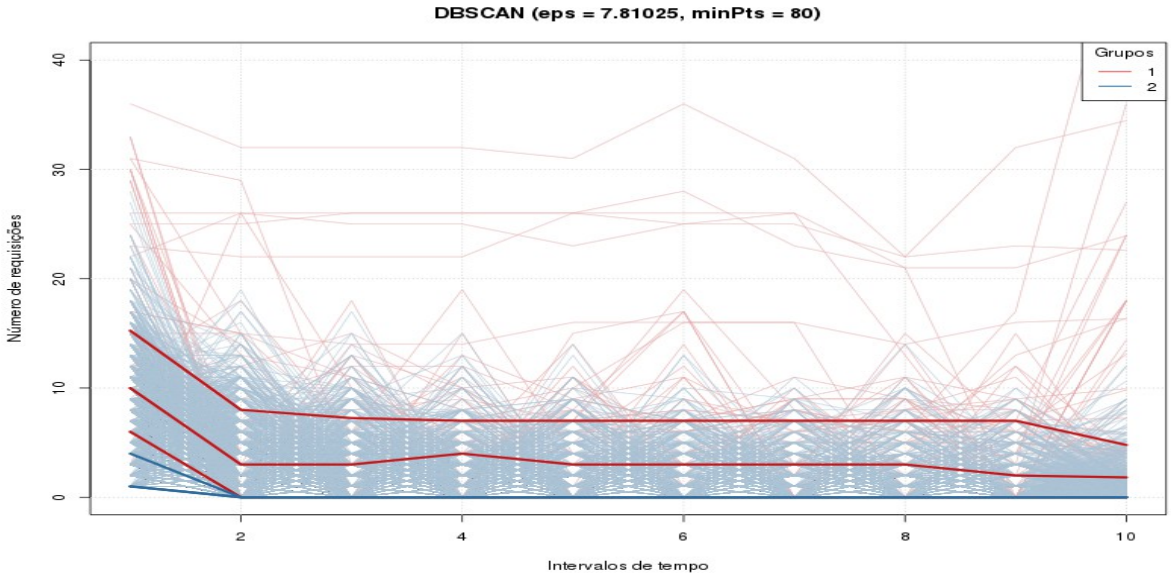

Figura 39 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. Dbscan obtido com eps = 7.81025 e minPts = 80.

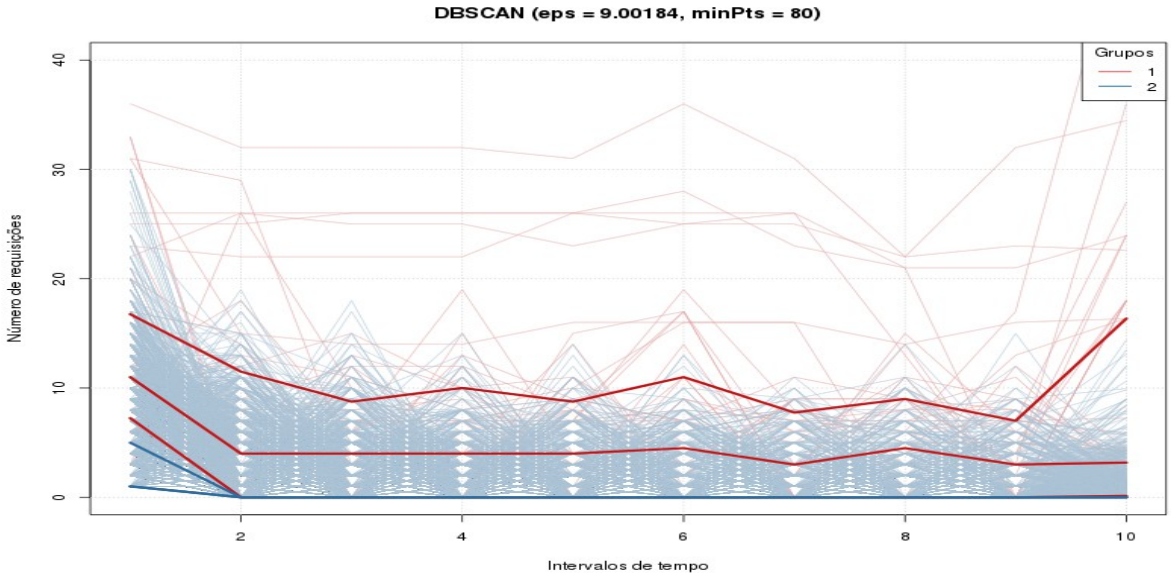

Figura 40 - Séries temporais, índice de dispersão e posição representadas em função do tempo e das cores de cada grupo. Dbscan obtido com eps = 9.00184 e minPts = 80.

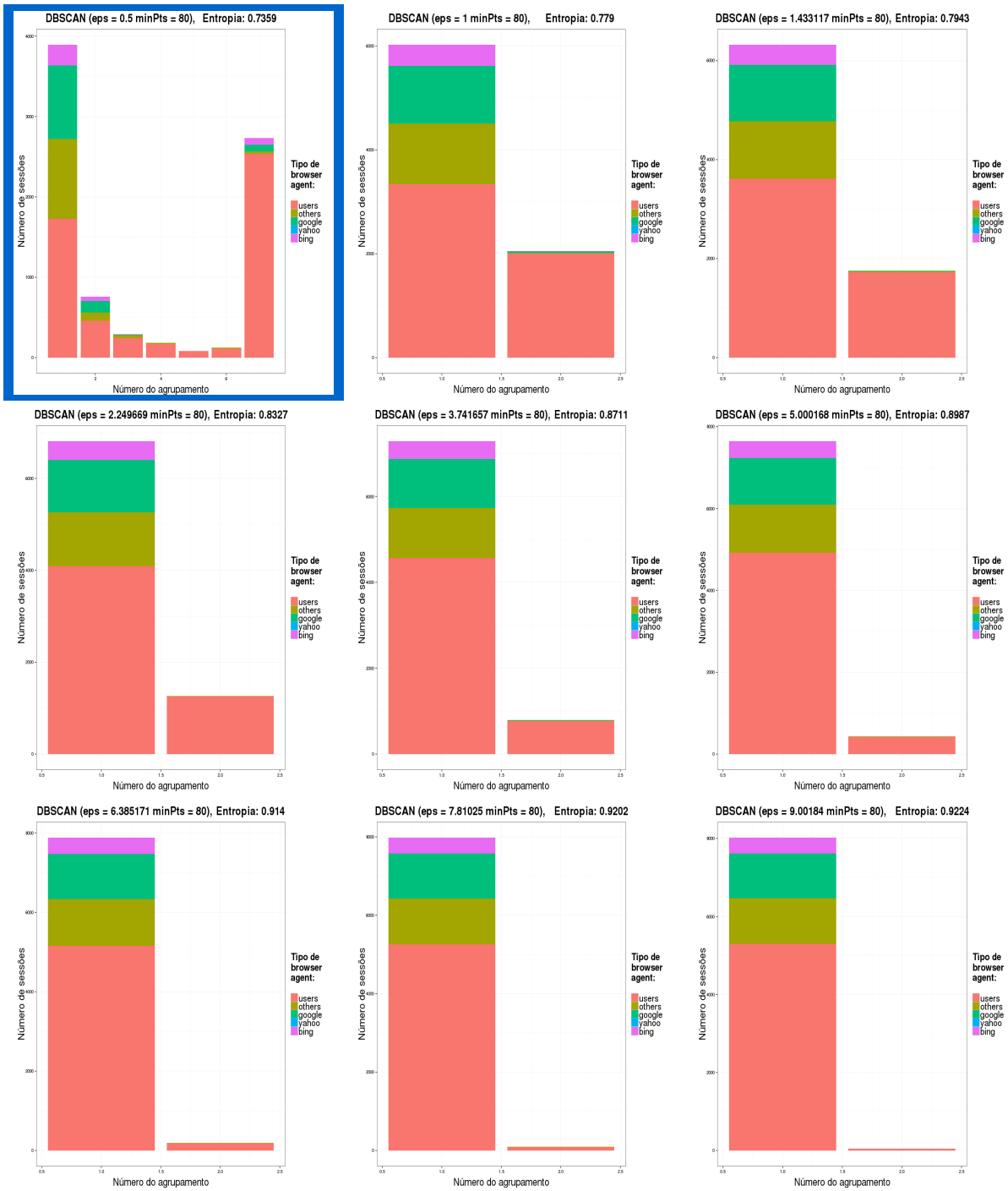

Figura 41 - Proporção de crawlers conhecidos detectados no agrupamento gerado usando dbscan com eps de 0.5, 1, 1.433117, 2.249669, 3.741657, 5.000168, 6.385171 7.810250, 9.001840 e minPts de 80. O resultado evidenciado possui a melhor entropia.

Os resultados apresentados foram avaliados utilizando o cálculo da entropia conforme apresentado na equação (3.4). Neste experimento, a entropia foi calculada considerando o tipo

de usuário que pode ser avaliado apenas como *crawlers* ou não *crawlers*, tendo sido utilizado para tal avaliação o atributo *robot* apresentado na Figura 19. Os valores de entropia total calculados para cada agrupamento são apresentados na Tabela 5, onde os valores em destaque possuem a melhor entropia para cada tipo de agrupamento.

Tabela 5 - Entropia total ponderada, calculada por cada agrupamento apresentado.

Tipo agrupamento	***Parâmetros***	***Entropia total***
k-means	K = 2	0.7876
	K = 3	0.7809
	K = 4	0.7757
	K = 5	0.7428
	K = 6	0.7411
	K = 7	0.7402
	K = 8	0.7139
	K = 9	0.7161
	K = 10	0.7148
dbscan	Eps = 0.5, MinPts = 80	0.7359
	Eps = 1, MinPts = 80	0.779
	Eps = 1.433117, MinPts = 80	0.7943
	Eps = 2.249669, MinPts = 80	0.8327
	Eps = 3.741657, MinPts = 80	0.7208
	Eps = 5.000168, MinPts = 80	0.7075
	Eps = 6.385171, MinPts = 80	0.8711
	Eps = 7.810250, MinPts = 80	0.8987
	Eps = 9.001840, MinPts = 80	0.914

Neste contexto foi possível avaliar os resultados apresentados anteriormente, tendo sido observado que os agrupamentos obtidos pelo *k-means* apresentaram uma tendência de melhora com o aumento do número de k, enquanto que os agrupamentos obtidos pelo *dbscan* apresentaram a mesma tendência ao diminuir o valor de *eps*. Comparando os resultados de cada técnica de agrupamento é possível inferir que a qualidade dos melhores resultados do *k-means* e do *dbscan* é similar.

5 Conclusões

Os resultados mostraram que as técnicas de agrupamento empregadas neste trabalho detectaram o padrão de comportamento dos programas automáticos de varredura, pois na maioria dos resultados obtidos todos os *crawlers* conhecidos foram associados ao mesmo grupo. Este resultado indica que há um potencial de exploração dos grupos gerados pelas técnicas de classificação como, por exemplo, redes neurais artificiais (ANN) ou máquinas de vetor de suporte (SVM), aplicadas a fluxos de dados construídos a partir de arquivos de *logs*.

Um número relevante de acessos de usuários desconhecidos também foi incluído no mesmo grupo dos *crawlers*. Estes acessos podem ser falsos positivos, ou seja, usuários legítimos que foram classificados erroneamente com o perfil de *crawler*, ou podem ser programas maliciosos disfarçados como usuários normais e que acessaram o servidor na tentativa de ataque ou de descoberta de vulnerabilidades, portanto, apresentando comportamento semelhante ao de um *crawler*. Esta informação poderia ser obtida através da análise de *datasets* sintéticos, onde todas as requisições sejam previamente rotulados como *crawlers* ou acesso normal. Caso contrário, não seria possível determinar o número de falsos positivos que foram associados ao grupo dos *crawlers*, como aconteceu neste experimento, pois as outras sessões associadas a este grupo podem ter sido originadas por programas automáticos maliciosos que podem estar disfarçados como navegadores normais.

As séries temporais identificadas se caracterizaram por apresentarem perfis muito heterogêneos entre si, de forma que não foi possível reconhecer uma distribuição padrão, como já havia sido identificado por *Shiravi* et al (2012). A grande maioria das séries temporais extraídas é de breve duração. Isto pode ser uma característica da atividade de navegação nas páginas da internet, bem como ser concernente ao tipo de aplicações usadas neste servidor (Moodle e Wordpress), ou ainda estar relacionado ao tipo de usuário ou ao escopo particular de sua navegação. Provavelmente os usuários deste servidor são em grande parte estudantes em busca de informações pontuais sobre os cursos ou acerca dos trabalhos a serem entregues e, neste caso, é provável que suas atividades no sítio WEB sejam de curta duração. As poucas séries temporais de longa duração poderiam ser formadas pelo acesso de professores que utilizam a aplicação por mais tempo, pois precisam configurar os sítios e colocar o material a ser disponibilizado.

O conjunto de dados empregado como base para esse trabalho é relativamente pobre

de informações. A falta de uma base de dados rotulada com a informação prévia de quais são as requisições maliciosas não permitiu avaliar outros aspectos do resultado dos agrupamentos. Inicialmente, o objetivo deste trabalho era analisar dados já utilizados na literatura, tais como o *dataset* NASA ou os relacionados à mesma pesquisa (ARLITT; WILLIAMSON, 1996) Porém, este *dataset*, bem como os outros que foram encontrados na literatura, não fornecem a informação de *browser-agent* necessária para a identificação dos *crawlers* conhecidos. Além disto, o *dataset* NASA se revelou grande demais para algumas das análises desenvolvidas. A utilização de métodos de agrupamento de séries temporais empregando técnicas de indexação e redução de dimensionalidade também poderia ser uma outra área interessante para trabalhos futuros.

Assim, visando dar continuidade às pesquisas no tema do presento trabalho, outras formas de geração de séries temporais podem ser experimentadas. O somatório dos valores de um atributo em uma faixa de tempo é apenas uma forma de gerar uma série temporal de tamanho fixo. Também existem diversas formas de comparação entre séries temporais que podem ser usadas para tratar os dados antes de utilizar as técnicas de agrupamento. Existem trabalhos, por exemplo, que utilizam *wavelets* para transformar os dados antes de empregar as técnicas de agrupamento (FU, 2011).

REFERÊNCIAS BIBLIOGRÁFICAS

ARLITT, M. F.; WILLIAMSON, C. L. Web server workload characterization: the search for invariants. **ACM SIGMETRICS Performance Evaluation Review**, v. 24, n. 1, p. 126–137, 15 maio 1996.

CIESLAK, D. A.; CHAWLA, N. V.; STRIEGEL, A. **Combating imbalance in network intrusion datasets**IEEE, 2006Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1635905>. Acesso em: 12 maio. 2014

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The KDD process for extracting useful knowledge from volumes of data. **Communications of the ACM**, v. 39, n. 11, p. 27–34, 1 nov. 1996.

FU, T. A review on time series data mining. **Engineering Applications of Artificial Intelligence**, v. 24, n. 1, p. 164–181, fev. 2011.

HUI XIONG; JUNJIE WU; JIAN CHEN. K-Means Clustering Versus Validation Measures: A Data-Distribution Perspective. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 39, n. 2, p. 318–331, abr. 2009.

LEE, W.; STOLFO, S. J. Data mining approaches for intrusion detection. **SSYM'98 Proceedings of the 7th conference on USENIX Security Symposium**, v. 7, p. 6 – 6, 1998.

LEE, W.; STOLFO, S. J. A framework for constructing features and models for intrusion detection systems. **ACM Transactions on Information and System Security**, v. 3, n. 4, p. 227–261, 1 nov. 2000.

L. THING, V.; SLOMAN, M.; DULAY, N. Locating network domain entry and exit point/path for DDoS attack traffic. **IEEE Transactions on Network and Service Management**, v. 6, n. 3, p. 163–174, 2009.

MIRKOVIC, J.; REIHER, P. D-WARD: A Source-End Defense against Flooding Denial-of-Service Attacks. **IEEE Transactions on Dependable and Secure Computing**, v. 2, n. 3, p. 216–232, mar. 2005.

NETWORK SECURITY. **"Heartbleed" flaw leaves millions of websites, email servers and other services vulnerable to attack**. [s.l: s.n.]. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1353485814700364>. Acesso em: 4 maio. 2014.

PEDDABACHIGARI, S. et al. Modeling intrusion detection system using hybrid intelligent systems. **Journal of Network and Computer Applications**, v. 30, n. 1, p. 114–132, jan. 2007.

QUITTEK, J. et al. Requirements for IP Flow Information Export. **RFC3917**, out. 2004.

SHIRAVI, A. et al. Toward developing a systematic approach to generate benchmark datasets

for intrusion detection. **Computers & Security**, v. 31, n. 3, p. 357–374, maio 2012.

SPEROTTO, A. et al. **A Labeled Data Set for Flow-Based Intrusion Detection**Proceedings of the 9th IEEE International Workshop on IP Operations and Management. **Anais**... In: 9TH IEEE INTERNATIONAL WORKSHOP ON IP OPERATIONS AND MANAGEMENT. Venice, Italy: 29 out. 2009

SPEROTTO, A.; PRAS, A. **Flow-based intrusion detection**IEEE, maio 2011Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5990529>. Acesso em: 20 abr. 2014

STEVANOVIC, D.; AN, A.; VLAJIC, N. Feature evaluation for web crawler detection with data mining techniques. **Expert Systems with Applications**, v. 39, n. 10, p. 8707–8717, ago. 2012.

SUBBULAKSHMI, T. et al. **Detection of DDoS attacks using Enhanced Support Vector Machines with real time generated dataset**IEEE, dez. 2011Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6165212>. Acesso em: 20 abr. 2014

TANASA, D.; TROUSSE, B. Advanced data preprocessing for intersites Web usage mining. **IEEE Intelligent Systems**, v. 19, n. 2, p. 59–65, mar. 2004.

TIBSHIRANI, R.; WALTHER, G.; HASTIE, T. Estimating the number of clusters in a data set via the gap statistic. **Journal of the Royal Statistical Society: Series B (Statistical Methodology)**, v. 63, n. 2, p. 411–423, maio 2001.

WARREN LIAO, T. Clustering of time series data—a survey. **Pattern Recognition**, v. 38, n. 11, p. 1857–1874, nov. 2005.

WENKE LEE; STOLFO, S. J.; MOK, K. W. **A data mining framework for building intrusion detection models**IEEE Comput. Soc, 1999Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=766909>. Acesso em: 29 maio. 2014

WU, S.-Y.; YEN, E. Data mining-based intrusion detectors. **Expert Systems with Applications**, v. 36, n. 3, p. 5605–5612, abr. 2009.

GLOSSÁRIO

Box-plot – Diagrama de caixa, representação gráfica que descreve distribuições através índices de dispersão e posição

Browser-agent – Identificação do navegador

Crawler – Rastreador da Web, programa de computador que navega de forma automatizada

Dataset – Conjunto de dados

Dbscan – Método de agrupamento por densidade (Density-Based Spatial Clustering of Applications with Noise)

Defcon 9 – Conjunto de dados gerado na competição de hackers ocorrida na nona das conferências anuais chamadas Defcon

Elbow method – Método do cotovelo para avaliação visual de gráficos

Kddcup 99 – Conjunto de dados mais conhecido para os métodos de detecção de anomalias

K-means – Método de agrupamento por particionamento

Logging – Atividade de auditoria

Logs – Arquivos de auditoria

R – Linguagem de programação para computação estatística

Wavelet – Ondaleta, função capaz de decompor e descrever ou representar outra função