Entendendo e melhorando a capacidade de detecção de estratégias de busca de similaridade em investigações forenses

João P. B. Velho[1], **Vitor H. G. Moia**[1], **Marco A. A. Henriques**[1]

[1]Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
Campinas, SP, Brazil 13083-852

{bizzi,vhgmoia,marco}@dca.fee.unicamp.br

Abstract. *Digital forensic practitioners face two main challenges: The increase in the number of digital devices in use and the difficulties in analysing them. Approximate Matching (AM) can be used to find relevant data by efficiently assessing the similarity of objects. However, commonly used procedures to assess the similarity between data sets (e.g., brute force) consume too much time and resources. To tackle this problem, the Similarity Digest Search Strategies allow faster comparisons by performing smart searches with AM. This paper compares some strategies in the literature and related brute force approaches, showing their precision and recall rates. We present the time taken by the strategies, analyze the impact of file types on similarity, and propose improvements.*

Resumo. *Peritos em forense digital têm dois grandes desafios: o aumento no número de dispositivos digitais em uso e as dificuldades em analisá-los. Funções de Pareamento Aproximado (PA) são utilizadas para encontrar dados relevantes através da avaliação de similaridade entre objetos de forma eficiente. Contudo, métodos tradicionais (força bruta) consomem muito tempo e recursos. Para mitigar este problema, as estratégias de busca de similaridade permitem rápidas comparações por meio das funções de PA. Este artigo compara algumas destas estratégias e métodos de força bruta, mostrando suas taxas de precision e recall. Também apresentamos o tempo de execução das estratégias, o impacto do tipo de arquivo na similaridade e propomos melhorias.*

1. Introdução

O avanço tecnológico, apesar de todos os seus benefícios para a sociedade, torna as investigações no âmbito da forense digital cada vez mais desafiadoras. O aumento na quantidade e também na capacidade de armazenamento de dispositivos digitais faz com que o trabalho de análise destes dispositivos pelos peritos forenses se torne cada vez mais árduo. Normalmente, são utilizadas funções hash para gerar resumos dos arquivos de uma mídia e facilitar a localização de artefatos já conhecidos e de interesse em uma investigação, como exemplares de malware, arquivos de listas negras etc. Estas funções são conhecidas por serem eficientes e conseguirem lidar com grandes quantidades de dados em um curto período de tempo. Contudo, devido ao efeito avalanche presente nestas funções (alteração de um bit na entrada gera uma saída completamente diferente), uma limitação se torna clara neste contexto: a ineficiência para busca de similaridade.

Para solucionar o problema envolvendo a identificação de similaridade, foram desenvolvidos nos últimos anos as funções de Pareamento Aproximado (PA). Assim como

as funções hash, as técnicas de PA tem como objetivo criar resumos para os arquivos e, através da simples comparação destes resumos, avaliar a similaridade de forma eficiente, na qual arquivos similares apresentam resumos similares. Além de identificar artefatos de interesse em uma investigação, estas funções também podem ser empregadas em sistemas anti-vírus para identificar malwares de forma mais eficiente que os métodos de assinatura tradicionais.

O problema que ainda permanece com as funções de PA é que, dados dois conjuntos com muitos arquivos, ainda é inviável a comparação destes conjuntos utilizando métodos convencionais como o de força bruta, onde uma comparação arquivo por arquivo entre os conjuntos é realizada. Dado a grande quantidade de dados presentes em investigações de hoje em dia, métodos simples de busca como o citado se tornam impraticáveis. Por esta razão, foram desenvolvidas na literatura as estratégias de busca de similaridade, que são formas mais eficientes da utilização das funções de PA para a realização de comparações entre grandes conjuntos de arquivos [Moia and Henriques 2017b]. Utilizando os resumos de um conjunto de arquivos gerados pelas funções de PA, as estratégias criam uma estrutura de armazenamento para estes resumos, seja uma tabela ou um filtro de Bloom, por exemplo, de forma a organizar os resumos criados e permitir que buscas de similaridade entre dois conjuntos sejam realizadas de forma mais eficiente.

No entanto, as estratégias de busca de similaridade ainda carecem de uma avaliação e comparação conjuntas, analisando como tais abordagens se comportariam na prática. Deste forma, este trabalho busca avaliar e comparar algumas das estratégias existentes, como o `MRSH-NET` e o `F2S2` (escolhidas por serem as precursoras da área de PA) além de métodos de força bruta relacionados. Mostramos sob a ótica das métricas de *precision* e *recall* como estas estratégias lidam com dados reais. Também avaliamos o tempo de execução e o impacto do tipo de arquivo na similaridade. Melhorias são propostas na forma de uma modificação na estratégia `MRSH-NET`, a fim de melhorar sua capacidade de detecção de similaridade, como mostrado e comprovado em nossos experimentos, onde os resultados se mostram similares aos das demais estratégias e até superiores em alguns casos. Este trabalho está estruturado da seguinte forma: Seção 2 apresenta trabalhos relacionados; Seção 3 apresenta o problema estudado, seguido de nossa modificação do `MRSH-NET`. Os dados e métricas utilizadas no estudo são apresentados em seguida. Finalizamos com a discussão dos resultados obtidos e conclusões.

2. Trabalhos Relacionados

Nesta seção, apresentamos os conceitos básicos e alguns dos métodos existentes na literatura para entendimento tanto das funções de pareamento aproximado, como das estratégias de busca de similaridade. Para uma visão mais completa da área de PA, consultar os trabalhos de Harichandran et al. [Harichandran et al. 2016] e de [Moia and Henriques 2017b].

As funções de Pareamento Aproximado destinam-se a criar representações compactas para os arquivos (resumos) a fim de avaliar a similaridade entre dois arquivos de forma simples e eficiente. Tais algoritmos podem operar em três diferentes níveis: bytes, sintático e semântico [Breitinger et al. 2014b]. Este trabalho visa avaliar apenas as funções que operam no nível de bytes, ou seja, consideram apenas os bytes de um arquivos para geração do resumo, sem buscar interpretá-los. Os motivos para tal escolha são

devidos à maior eficiência em gerar e comparar resumos, e da capacidade destas funções em serem independentes de formato, diferentemente dos outros níveis.

Para identificar os arquivos similares de dois conjuntos, comumente é utilizado o método de força bruta, onde compara-se todos os resumos de um conjunto contra todos os resumos do outro conjunto. Contudo, este processo se torna impraticável a medida que a quantidade de arquivos destes conjuntos aumenta. Visando minimizar este problema, foram desenvolvidas as estratégias de busca de similaridade, que realizam uma busca inteligente utilizando os resumos gerados pelas funções de PA.

2.1. Funções de Pareamento Aproximado

2.1.1. `ssdeep`

Desenvolvida por Kornblum, J. [Kornblum 2006], a ferramenta `ssdeep` foi baseada no detector de spam de Tridgell (`Spamsum`) [Tridgell 2002] e pode ser considerada uma das precursoras da área de PA. Para a criação de um resumo de similaridade, a ferramenta divide um arquivo em blocos de tamanhos variados usando uma função de *rolling hash*, que define onde um bloco começa e termina. Primeiro, é utilizada uma janela deslizante de tamanho fixo (sete bytes) que se move byte-a-byte através do arquivo e, para cada iteração, é gerado um valor com a função *rolling hash* da janela e este é comparado a uma constante derivada do tamanho do arquivo; caso este valor seja igual, o fim de um bloco (e início de outro) é definido e o processo se repete até o fim do arquivo. Uma segunda etapa consiste em calcular o hash (com a função FNV [Noll 2012]) de todos os blocos gerados no passo anterior, onde os seis bits menos significativos de cada hash são convertidos em um caractere (base 64) e utilizado para representar aquele hash; o resumo de similaridade consiste na concatenação de todos os caracteres gerados (máximo 64). Para avaliação da similaridade entre dois resumos é utilizada uma função de *edit distance*, que conta o número mínimo de operações necessárias para transformar os caracteres de um resumo no outro. Um valor no intervalo de zero a 100 é retornado, onde zero representa que os arquivos são diferentes e 100 que são iguais ou muito similares.

2.1.2. `sdhash`

Proposta por Roussev [Roussev 2010], a ferramenta `sdhash` é talvez uma das mais conhecidas e utilizadas de PA, alvo constante de trabalhos na área. Esta ferramenta extrai *features* (pedaços de β bytes de um arquivo - padrão: 64 bytes) dos arquivos e faz uso da entropia de Shannon para selecionar os que melhores representam o arquivo como único. Após a seleção, cada *feature* é comprimida utilizando a função de hash SHA-1 e armazenada em um filtro de Bloom. Um filtro tem capacidade máxima para 160 features e, quando atinge seu limite, um novo filtro é criado. No final do processo, o resumo de similaridade é a concatenação de todos os filtros de Bloom criados. Em linhas gerais, a comparação de dois resumos consiste em comparar o primeiro filtro do primeiro arquivo contra todos os filtros do segundo arquivo e selecionar o valor cuja comparação resultou na maior quantidade de bits em comum. O processo se repete para todos os demais filtros do primeiro arquivo contra o segundo e no fim uma média é computada, variando entre zero (arquivos diferentes) a 100 (arquivos iguais ou muito parecidos).

2.2. Estratégias de busca de similaridade

2.2.1. F2S2

O F2S2 foi desenvolvido por Winter et al. [Winter et al. 2013] e utiliza os resumos de similaridade gerados pela ferramenta ssdeep para organizá-los de maneira a realizar buscas de similaridade de forma mais eficiente. Para isso, é utilizado um tipo especial de tabela hash para armazenar os resumos. Primeiro, uma janela deslisante (byte-a-byte) de tamanho *n* é aplicada a cada resumo a ser inserido na estrutura. O valor da janela (n-grama) é dividido em duas partes, onde a primeira é usada para definir a posição na tabela (índice) e a segunda como identificador do resumo. Após mapear um dado conjunto na estrutura, podemos realizar buscas de similaridade dado um resumo de um arquivo de interesse. Os n-gramas extraídos do resumo são utilizados para encontrar candidatos a similaridade com o arquivo em questão, de forma que são retornados todos os resumos que possuem os mesmos n-gramas do item consultado; um passo adicional consiste em verificar a similaridade destes candidatos utilizando-se o ssdeep, a fim de determinar se cada um dos registros retornados são similares ou apenas falso positivos da estratégia.

2.2.2. MRSH-NET

Desenvolvido por Breitinger et al. [Breitinger et al. 2014a], o MRSH-NET faz uso da ferramenta de PA mrsh-v2 [Breitinger and Baier 2013] para extrair as *features* (ou blocos) dos arquivos de um dado conjunto. Estas *features* são armazenadas em um único filtro de Bloom. Dado um arquivo de interesse, é possível determinar se este possui similaridade com algum dos arquivos do conjunto mapeado na estrutura ou não. Para realizar esta consulta, basta extrair as features do arquivo e verificar a presença destas no filtro. Diferentemente das outras ferramentas que retornam se existe algum arquivo similar e qual(is) arquivo(s) é(são) similar(es), o MRSH-NET retorna uma resposta binária, indicando apenas se o arquivo possui similaridade com outro do conjunto mapeado ou não.

3. Descrição do Problema e Direções da Pesquisa

As funções de Pareamento Aproximado apresentam altos tempos de execução quando submetidas a grandes volumes de dados. Dado dois conjuntos, sendo o primeiro composto por arquivos de referência de posse de um investigador de tamanho r e o segundo por arquivos de interesse (a serem analisados) de tamanho q, compará-los resulta em uma complexidade de busca de $O(qr)$. Para minimizar esta alta complexidade, foram propostas diferentes estratégias de busca de similaridade que conseguem reduzir a complexidade para até $O(q)$ em alguns casos.

As estratégias analisadas neste trabalho, isto é, o F2S2 e o MRSH-NET, apresentam particularidades em relação a forma de apresentar seus resultados, o que pode ser suficiente ou até limitar uma investigação. Por utilizar um filtro de Bloom para mapear todos os arquivos de um conjunto, o MRSH-NET retorna apenas uma resposta binária a uma consulta, sendo capaz de afirmar se um dado arquivo está presente no conjunto mapeado em sua estrutura ou não; ele não indica a qual arquivo do conjunto o item consultado é similar. Contudo, seu benefício está em sua menor complexidade: $O(q)$. Já o F2S2 é capaz de indicar a qual arquivo do conjunto mapeado o item consultado é similar. Por outro

lado, sua complexidade é $O(qr)$, apesar de testes mostrarem que na prática a redução do tempo é significativa [Winter et al. 2013].

É crucial levar em consideração o tipo de resposta, assim como a complexidade de busca, no momento de escolher qual estratégia utilizar. Além disso, a capacidade de detecção também deve ser um fator indispensável para a tomada de decisão. Desta forma, a primeira contribuição deste trabalho é a proposta de uma nova versão da abordagem `MRSH-NET`, com melhorias focadas em diminuir a complexidade de busca e aumentar a capacidade de detecção desta estratégia.

Outra contribuição deste trabalho é a comparação das estratégias em relação à capacidade de detecção, em especial analisando as taxas de *precision* e *recall*. Até onde sabemos, nenhum trabalho na literatura realizou tal análise. São comparadas as estratégias mencionadas acima (com adição da versão que propomos) e são analisadas as taxas de *precision* e *recall*, mostrando o impacto dos diferentes tipos de arquivos na similaridade. Uma análise dos tempos de execução das estratégias também é realizado. É importante destacar que, como não existiam implementações de código-aberto da estratégia `F2S2` (até onde sabemos), foi desenvolvida e disponibilizada uma prova de conceito desta estratégia em `https://github.com/regras/f2s2`. O código desenvolvido realiza as principais funções da estratégia: a criação da estrutura, inserção dos resumos e a consulta de um resumo na estrutura.

4. Aprimorando o `MRSH-NET`

A estratégia `MRSH-NET` utiliza a ferramenta `mrsh-v2` para extração de features dos arquivos. Apesar de possuir características interessantes para investigações forenses, [Breitinger and Roussev 2014] mostram que esta ferramenta gera muitos falsos positivos (diminuindo sua taxa *precision*), além de ter um desempenho geral menor que o `sdhash`. Neste trabalho, decidimos realizar a substituição do módulo de extração de features do `mrsh-v2` pelo módulo do `sdhash`, a fim de melhorar as capacidades de detecção da estratégia `MRSH-NET`. Batizamos a versão modificada de `MRSH-SD`; nas próximas seções serão discutidas as vantagens e desvantagens de nossa alteração.

É importante destacar que a mudança realizada não altera a forma de interpretação dos resultados da nova estratégia. Outro ponto a ser enfatizado é que a versão original da estratégia impõe uma condição para considerar um arquivo como similar. Esta condição requer que o arquivo consultado tenha, no mínimo, seis (6) features (em comum) consecutivas encontradas em sua estrutura durante uma busca; caso contrário, o arquivo será considerado diferente por não haver similaridade suficiente. Esta decisão foi mantida a princípio no `MRSH-SD`, porém testes variando este valor de inicial de *threshold* serão realizados a fim de observar o impacto nas taxas de *precision* e *recall*. Ressaltamos ainda que os parâmetros utilizados na extração e seleção de features são os mesmos utilizados pelo `sdhash` original. O código-fonte da implementação de nossas mudanças se encontra disponível em `https://github.com/regras/mrsh-sd`.

5. Dados e métricas utilizados nos experimentos

Para realizar os experimentos deste trabalho, utilizamos dois conjuntos de arquivos extraídos da base de dados *t5-corpus* [Roussev 2011]. Estes conjuntos foram denominados como *target* e *known*, e suas estatísticas são apresentadas na Tabela 1.Foram uti-

lizados conjuntos relativamente pequenos para facilitar a análise manual dos resultados. Assim é possível saber quais arquivos são realmente similares para calcular as métricas *precision* e *recall*.

Tabela 1. Relação dos tipos de arquivos e suas quantidades para cada conjunto

Conjunto	Tipo de arquivo								$\sum$
	html	*text*	*pdf*	*doc*	*ppt*	*xls*	*jpg*	*gif*	
known	1073	701	1053	513	358	240	357	62	4357
target	20	10	20	20	10	10	5	5	100

Além de avaliar as estratégias, adicionamos também duas abordagens que utilizam o método de força bruta usando as ferramentas de Pareamento Aproximado `sdhash` e `ssdeep`. Foi gerado um resumo para cada arquivo dos conjuntos *known* e *target* e realizada uma comparação entre os conjuntos (todos-contra-todos) para cada ferramenta a fim de determinar os pares similares entre os conjuntos. Já para as estratégias de busca de similaridade, o conjunto *known* foi mapeado na estrutura da estratégia, enquanto os arquivos do conjunto *target* tiveram seus resumos calculados e consultados um a um na estrutura criada.

Neste trabalho, avaliamos a capacidade de detecção das estratégias através de três métricas recorrentes no cenário de pareamento aproximado e que são utilizadas na área de *Recuperação de Informação*: *precision*, *recall* e *F-score*. A métrica *precision* avalia a proporção de acertos da estratégia, ou seja, pares de arquivos realmente similares (tp) em relação a todos os resultados retornados, incluindo falsos positivos ($tp + fp$). O *recall* se refere a proporção de arquivos similares encontrados (tp) em relação à todos os arquivos similares existentes entre os conjuntos ($tp + fn$). Já o *F-score* é a média harmônica entre as duas métricas anteriores, sendo um o melhor resultado que pode ser alcançado.

$$precision = \frac{t_p}{tp+fp} \qquad recall = \frac{tp}{tp+fn} \qquad F-score = 2 * \frac{precision*recall}{precision+recall}$$

Para calcular as métricas citadas, é necessário saber quando uma estratégia acerta ou erra ao comparar dois arquivos. É importante ressaltar que existem diferentes formas de similaridade. [Moia et al. 2020] define que a similaridade encontrada entre arquivos pode ser de três tipos: conteúdo gerado por usuário - *User Generated Content* (UGC), conteúdo gerado por aplicação - *Application Generated Content* (AGC) ou conteúdo gerado por template - *Template Content* (TC). Neste trabalho, adotaremos a mesma classificação de similaridade. Além do mais, consideraremos como similar, ou verdadeiro positivo (tp), apenas resultados de comparações relacionadas a UGC ou TC, visto que são as duas normalmente de maior interesse em uma investigação. Similaridades do tipo AGC serão consideradas falsos positivos (fp); verdadeiros negativos (tn) serão as similaridades não encontradas de AGC, enquanto falsos negativos (fn) são similaridades não encontradas de UGC e TC. Cenários em que os tipos de similaridades de interesse são outros (por exemplo, apenas TC e AGC), serão abordados em trabalhos futuros.

Para classificar as similaridades retornadas pelas estratégias dentre os três tipos, utilizaremos os resultados de uma classificação manual disponível em `https://github.com/regras/cbamf/blob/master/Some_matches_t5-corpus_per_similarity_class.txt`. Estes dados foram organizados em

forma de lista e têm grande parte das comparações entre os conjuntos analisados que possuem algum nível de similaridade, além do seu tipo (UGC, AGC ou TC). Qualquer similaridade reportada pela ferramenta entre arquivos que não constem nesta lista será considerada como não-similar. Iremos nos referir a esta lista como `lista-verdade` .

Outro ponto a ser destacado é que, devido ao funcionamento de cada estratégia, alguns procedimentos para a avaliação mudam de uma estratégia para outra. Como a estratégia `F2S2` não indica para quais arquivos não há similaridade quando realizamos uma consulta, neste caso assumimos que todos os arquivos do conjunto *known* são diferentes e calculamos o tn e o fn nos baseando na `lista-verdade` . Já quando um arquivo é dito similar, calculamos o tp e o fp também nos baseando na `lista-verdade` mas agora para saber se houve acerto ou erro. Ressaltamos que o `F2S2` pode retornar nenhum ou vários arquivos com similaridade para cada item consultado.

Para as estratégias, `MRSH-NET` e `MRSH-SD`, o procedimento de avaliação é diferente. Como estas estratégias apenas nos fornecem uma resposta binária para cada consulta, calculamos apenas um único valor relativo ao acerto ou erro destas abordagens com base na `lista-verdade` . Assim, apenas um dos quatro possíveis valores é retornado para cada consulta: tp, fp, tn ou fn.

A última observação é em relação aos parâmetros internos utilizados pelas estratégias. O `F2S2` requer a definição do número de entradas da tabela hash a ser criada. Utilizamos três bytes do *n-gram* para criação do índice da tabela, o que leva a uma tabela com 2^{24} entradas, mais do que suficiente para a quantidade de dados disponíveis. Já para o tamanho do filtro de Bloom das estratégias `MRSH-NET` e `MRSH-SD`, utilizamos a Eq. 1 com os seguintes valores: probabilidade de ocorrência de um falso positivo $p_f = 10^{-6}$, número de subhashes $k = 5$, e número mínimo de features subsequentes em comum (para uma comparação ser considerada similar) $r = 6$. O tamanho do conjunto mapeado na estratégia é de $s = 1.7GB$. Com base nos parâmetros fornecidos, o tamanho mínimo do filtro é de 33 MB.

$$m = -\frac{k \cdot s \cdot 2^{14}}{\ln(1 - \sqrt[kr]{p_f})} \tag{1}$$

6. Resultados e Discussões

Iniciamos esta seção com uma análise do valor de *threshold* da estratégia `MRSH-SD`. Em seguida, apresentamos os resultados da busca de similaridade utilizando as estratégias apresentadas na Sec. 3 e os conjuntos de dados da Sec. 5. Para cada estratégia, calculamos o *precision*, *recall* e *F-score*. O impacto do tipo de arquivo na similaridade também é analisado, assim como o tempo gasto na comparação dos conjuntos.

6.1. Escolhendo o valor de *threshold* para o `MRSH-SD`

Para avaliar o impacto do *threshold* (número mínimo de features (em comum) consecutivas a serem encontradas para uma comparação ser considerada similar) na estratégia `MRSH-SD`, realizamos a comparação entre os conjuntos *known* e *target* e calculamos as taxas de *precision* e *recall* de acordo com a variação do valor de *threshold*. Os resultados são apresentados na Fig. 1, no qual podemos observar três faixas de valores: a primeira, onde *recall* é maior do que *precision* (*threshold* entre 6 e 50); a segunda, com estabilização do *recall* e aproximação do valor de *precision* (*threshold* entre 51 e 130); e

a terceira faixa, com menor valor de *recall* e relativa estabilidade de *precision* (*threshold* acima de 130).

As três faixas de valores observadas no gráfico nos mostram que: (1) valores menores de *threshold* (entre 6 e 50) nos permitem encontrar comparações de arquivos com pequenos níveis de similaridades; (2) valores médios nos proporcionam um equilíbrio entre a quantidade de comparações detectadas e falsos positivos; e (3) altos valores para o *threshold* (maiores que 130) não são recomendados, pois não há aumento considerável em *precision*, enquanto *recall* apenas diminui de forma expressiva.

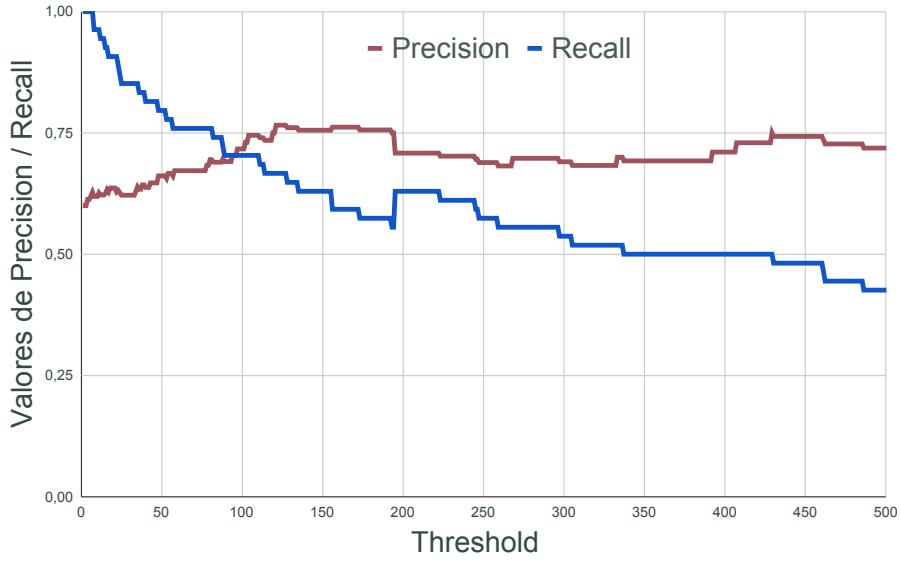

Figura 1. Impacto do valor de *threshold* da estratégia `MRSH-SD` nas métricas *Precision* e *Recall*

6.2. Analisando a capacidade de detecção das estratégias

Nesta seção, discutimos os resultados das comparações dos conjuntos analisados pelas estratégias de busca de similaridade. Para realização dos experimentos, foram utilizados os parâmetros padrão das estratégias `MRSH-NET` e `F2S2`, com variação apenas do tamanho de suas estruturas. Já para o `MRSH-SD`, utilizamos três versões, onde em cada uma alteramos o valor de *threshold (t)* de acordo com os resultados da seção anterior para representar diferentes situações que privilegiam determinada métrica. Os valores considerados foram: 6, 50 e 130. Os resultados são apresentados na Tabela 2.

Podemos notar pelos resultados que diferentes estratégias se sobressaíram em diferentes métricas. A abordagem força bruta utilizando a ferramenta `ssdeep` obteve o melhor valor de *precision*, apresentando a menor quantidade de falsos positivos. Por outro lado, apresentou um dos piores *recall*, onde várias comparações similares não foram detectadas. Já a estratégia que conseguiu retornar todas as comparações similares foi o `MRSH-SD` ($t = 6$), porém devido ao baixo valor de *threshold* utilizado, apresentou vários falsos positivos; contudo, mesmo com esta limitação, obteve a melhor relação de *precision* e *recall*, representada pela métrica *F-score*. Portanto, nossa decisão em substituir a ferramenta de PA da estratégia `MRSH-NET` se mostrou benéfica para aumentar a capacidade de detecção de similaridade entre arquivos pela nova versão.

Tabela 2. Resultado das comparações das estratégias de busca de similaridade e alguns métodos de força bruta para diferentes métricas

Estratégia	*Precision*	*Recall*	*F-score*
`ssdeep`	**0,778**	0,686	0,729
`sdhash`	0,249	0,983	0,397
`F2S2`	0,759	0,694	0,725
`MRSH-NET`	0,629	0,944	0,755
`MRSH-SD` ($t = 6$)	0,620	**1,000**	**0,765**
`MRSH-SD` ($t = 50$)	0,662	0,796	0,723
`MRSH-SD` ($t = 130$)	0,760	0,648	0,699

É importante ressaltar que os baixos valores de *precision* obtidos com todas as estratégias, principalmente com o `sdhash`, são devido à similaridade gerada por conteúdo de aplicação que foi considerado de não interesse em nossos experimentos. Este tipo de dado, referido na literatura como *blocos comuns* [Moia et al. 2020], será objeto de estudos futuros, onde esperamos que, após identificar e remover estes blocos, obtenhamos uma melhora significativa nos resultados de *precision* das estratégias.

Outro ponto a ser destacado em nossos resultados é em relação a capacidade de detecção de similaridade da ferramenta em comparação com a estratégia levando em conta as métricas analisadas. Gostaríamos de saber se a estratégia limita a capacidade de detecção da ferramenta por ela utilizada. Para responder a esta pergunta, comparamos as estratégias `F2S2` e `MRSH-SD` com as respectivas ferramentas por elas utilizadas (`ssdeep` e `sdhash`) sob a abordagem força bruta. No primeiro caso, percebemos que ambas tiveram resultados semelhantes. Se por um lado `ssdeep` teve uma pequena vantagem em *precision*, `F2S2` se saiu melhor em *recall*, porém a diferença foi mínima. Em relação ao segundo caso, podemos comparar `sdhash` com as três versões do `MRSH-SD`. A estratégia obteve resultados melhores para *precision* e *F-score* nas três versões. Além disso, na configuração com menor valor de *threshold*, a estratégia obteve também um valor maior de *recall*, onde conseguiu encontrar todas as comparações similares. A melhoria dos resultados das estratégias em relação às ferramentas pode ser explicada pela alteração do modo de avaliação da similaridade, que apesar de utilizar o mesmo processo de extração de features, mudou a forma de armazená-las e compará-las. Desta forma, podemos concluir que não há redução nas capacidades de detecção de similaridade das funções de PA por parte das estratégias (há até melhorias, em alguns casos).

6.3. Avaliando o impacto de diferentes tipos de arquivos na similaridade

Outra avaliação realizada neste trabalho foi verificar o impacto dos diferentes tipos de arquivos na detecção de similaridade pelas estratégias. Nesta análise, descartamos os métodos de força bruta pois a capacidade de detecção das estratégias é próxima (ou até superior) à deles. A Tabela 3 apresenta a quantidade de comparações similares por tipo de similaridade já conhecida entre os conjuntos de dados analisados. Novamente, será considerado que apenas similaridades entre arquivos com conteúdo gerado por usuário ou template. Além disso, serão descartadas as comparações dos tipos de arquivo `gif` e `jpg` devido ao baixo número de comparações similares das classes consideradas.

As Figuras 2, 3 e 4 mostram os resultados das comparações das estratégias `F2S2`,

Tabela 3. Quantidade de comparações por tipo de arquivo e de similaridade

Tipo de Similaridade	Tipo de arquivo								$\sum$
	html	*doc*	*pdf*	*ppt*	*text*	*xls*	*jpg*	*gif*	
AGC	6	19	19	5	0	21	29	3	102
TC	36	16	4	4	1	3	1	0	65
UGC	30	8	5	3	4	3	0	0	53
$\sum$	72	43	28	12	5	27	30	3	220

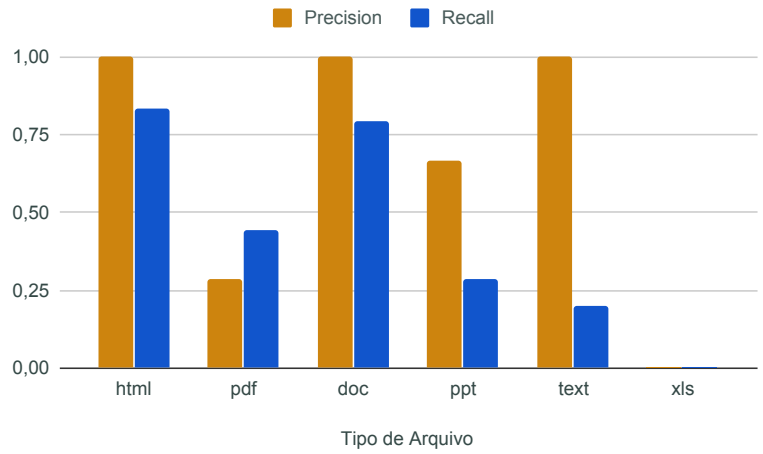

Figura 2. Taxas de *Precision* e *Recall* separadas por tipo de arquivo para a estratégia `F2S2`

`MRSH-NET` e `MRSH-SD` ($t = 6$), separados por tipo de arquivo. Para o cenário considerado em nossos experimentos, notamos que a taxa de *precision* se mantém alta para a maioria dos tipos de arquivos para a estratégia `F2S2`. A exceção são os arquivos do tipo `pdf`, com muitos casos de falsos positivos (similaridade por AGC), em razão dos *blocos comuns*. Já em relação a *recall*, o `F2S2` não conseguiu encontrar nenhuma das seis comparações do tipo `xls` existentes, mas também não acusou nenhum falso positivo (não retornou nenhuma comparação deste tipo). O baixo nível de *recall* pode ser explicado pela ferramenta utilizada (`ssdeep`), sendo esta vulnerável a modificações aleatórias em arquivos, como no caso de comparações do tipo `text` e `xls`. Este último é desafiador devido à organização dos dados, que podem estar em diferentes abas, linhas e colunas.

Já o `MRSH-NET` obteve altas taxas de *precision* e *recall* para arquivos `html`, `doc` e `ppt`. Porém, uma razoável quantidade de falsos positivos gerados por arquivos `pdf` e `xls` foi observada novamente, apesar de esta estratégia já ter identificado todas as similaridades existentes. Destacamos uma melhora em *recall* para arquivos `text` em relação à estratégia anterior, porém ainda não satisfatória. Aqui fica claro que a mudança da ferramenta usada pelas estratégias (de `ssdeep` em `F2S2` para `mrsh-v2` em `MRSH-NET`) fez a diferença no quesito *recall*.

Por fim, analisamos os resultados do `MRSH-SD` ($t = 6$) e novamente vemos o impacto da troca da ferramenta, agora usando o `sdhash` ao invés do `mrsh-v2`. Esta escolha, aliada a um bom valor de *threshold* (seis, na versão considerada aqui), fez com que a estratégia encontrasse todos os itens similares; porém, isto reflete em uma pequena

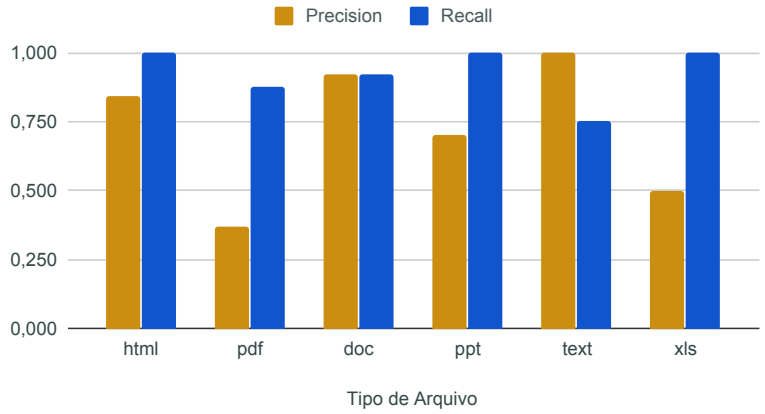

Figura 3. Taxas de *Precision* e *Recall* separadas por tipo de arquivo para a estratégia `MRSH-NET`

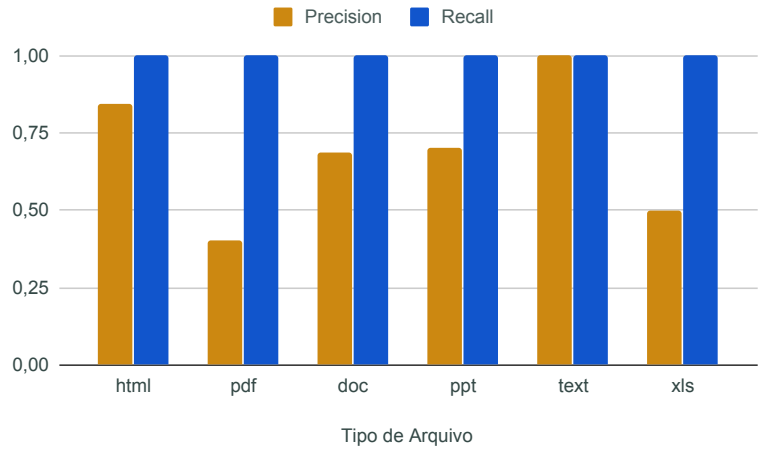

Figura 4. Taxas de *Precision* e *Recall* separadas por tipo de arquivo para a estratégia `MRSH-SD`$_6$

redução em *precision*, em particular para arquivos do tipo `doc` e `ppt`. Tal redução se dá devido à presença dos *blocos comuns* gerados por aplicações, que são encontrados em maiores quantidades justamente nestes tipos de arquivos.

Observando os três gráficos apresentados, nota-se que a maior dificuldade das estratégias foi com os arquivos do tipo `pdf`, que geram grandes quantidades de falsos positivos. O mesmo se repetiu para arquivos do tipo `xls`, onde o `F2S2` não conseguiu encontrar nenhuma comparação similar e as estratégias `MRSH-NET` e `MRSH-SD` $(t = 6)$ apresentaram altas taxas de falsos positivos devido a similaridades AGC. As informações similares neste tipo de arquivo são fragmentadas facilmente, o que torna o trabalho das ferramentas mais desafiador. Ressaltamos também que todas as estratégias se saíram bem buscando similaridades de arquivos do tipo `html` e `doc` no geral, apresentando boas taxas de *precision* e *recall*.

6.4. Tempos de criação e comparação dos conjuntos de dados

A última análise realizada foi em relação ao tempo de execução das estratégias nas etapas de criação de resumos e comparações. Para esta avaliação, foi utilizada a ferramenta *hyperfine*[1]. Cada experimento foi repetido dez vezes e a média dos resultados foi calculada. Outros cuidados, como limpeza de cache e isolamento da aplicação durante a execução, também foram tomados. Todos os testes foram realizados em uma máquina rodando o S.O. *Elementary OS Hera* 5.1.6, com 8 GB de memória RAM e processador i5-8300H 2.3GHZ.

Os testes realizados foram divididos em duas fases. Na primeira fase, de preparação, os resumos de cada arquivo do conjunto *known* foram gerados e organizados na estrutura de cada estratégia (ou em um arquivo, para os métodos de força bruta). Na segunda fase, referida como busca, foi criado o resumo de cada arquivo do conjunto *target* e realizada a comparação com os resumos armazenados pelas estratégias. Nesta etapa, somamos os tempos de geração de resumos (do conjunto *target*) com os de comparação. É importante ressaltar que, para as estratégias, é necessário um tempo adicional para criar suas estruturas e, quando realizamos o processo de busca, temos um tempo adicional de carregar a estrutura (armazenada no disco) para a memória.

Um segundo teste foi realizado, utilizando a mesma base de dados *t5* tanto como *known* (4357 objetos) como também *target* (100 objetos), a fim de avaliar como o tempo de cada estratégia seria afetado para um cenário com muitas comparações. Os resultados obtidos de ambos os testes podem ser observados na Tabela 4. Os resultados de busca da estratégia `F2S2` correspondem apenas à busca na estrutura, não sendo feita uma comparação com o `ssdeep` para verificar a similaridade dos resultados, permitindo uma avaliação apenas das capacidades da estratégia.

Tabela 4. Tempos médios em segundos de criação e comparação dos resumos e respectivos desvios padrão (entre parênteses)

Ferramentas	Target vs Known				t5 vs t5			
	Preparação		Busca		Preparação		Busca	
sdhash	54,12	(3,27)	3,25	(0,45)	55,78	(1,19)	176,79	(1,53)
ssdeep	69,44	(0,33)	4,29	(0,14)	69,68	(0,38)	82,52	(0,58)
F2S2	70,44	(0,48)	3,06	(0,18)	72,09	(0,50)	71,03	(0,47)
MRSH-NET	211,24	(2,04)	7,38	(0,36)	215,78	(1,33)	228,38	(3,02)
MRSH-SD	168,31	(3,63)	8,37	(0,27)	171,99	(1,00)	170,04	(0,52)

A comparação entre as ferramentas e estratégias mostra que, em geral, as ferramentas levam vantagem na fase de preparação, onde são criados os resumos, principalmente por não dependerem da criação de nenhuma estrutura adicional ou processamento dos resumos para serem armazenados em suas estruturas. Contudo, na fase de busca, os tempos das estratégias se sobressaem, onde a estratégia `F2S2` obteve o menor dos tempos. Além disso, a nossa proposta de melhoria (a estratégia `MRSH-SD`) ficou com tempos inferiores tanto para criar como comparar resumos em relação a versão original, destacando outra vantagem da mudança. Acreditamos que as estratégias `MRSH-NET` e

[1] <https://github.com/sharkdp/hyperfine> (último acesso em 2020-07-17)

MRSH-SD ficaram com tempos superiores devido ao processo de criação de resumos das ferramentas por elas utilizadas serem mais complexos do que o ssdeep.

Podemos perceber pelos resultados que as estratégias reduzem consideravelmente o tempo de comparação em relação às ferramentas. Esta afirmação pode ser constatada quando comparamos os tempos de preparação e busca do segundo conjunto comparado (*t5 vs. t5*), que, para as estratégias (com exceção do MRSH-NET), foram ligeiramente superiores. Vale lembrar que este segundo experimento comparou os 4457 arquivos da t5 contra ela mesma. Desta forma, foram gerados 4457 resumos na fase de preparação e, em seguida, foi gerada novamente a mesma quantidade de resumos, porém, com uma etapa adicional de comparação dos mesmos. Ainda na comparação de *t5* com *t5*, constatamos que houve um aumento no tempo de busca em relação ao tempo de preparação de 18,41% (ssdeep) e 216,96% (sdhash), mostrando um alto custo da função de comparação das ferramentas, principalmente, de sdhash. Destacamos que as implementações das estratégias, principalmente do F2S2, tentaram produzir apenas uma prova de conceito e não focaram na eficiência do código. Já ssdeep e sdhash passaram por várias revisões e aprimoramentos e, por isso, consideramos que estão otimizadas, o que pode ser considerado uma leve vantagem em relação aos demais.

Espera-se que as diferenças observadas nos tempos das ferramentas em relação às estratégias sejam consideravelmente maiores se conjuntos com mais arquivos forem utilizados, destacando ainda mais a vantagem das estratégias.

7. Conclusões e trabalhos futuros

Neste artigo, apresentamos estratégias de busca de similaridade capazes de realizar comparações de forma eficiente. Também propusemos melhorias a uma estratégia já existente resultando em menores tempos para gerar e comparar resumos, além da capacidade de encontrar todos os itens similares em nossos experimentos.

Os resultados obtidos neste trabalho nos permitem concluir que as estratégias não limitam a capacidade de detecção de similaridade das funções de pareamento aproximado por elas implementadas. Em alguns casos, mostramos que as estratégias fazem melhor uso dos resumos gerados, detectando mais similaridades do que as próprias ferramentas. Além disso, mostramos que o processo de comparação de resumos foi, nos experimentos realizados, o fator de maior custo durante as buscas, e que as estratégias reduzem consideravelmente o tempo das comparações entre resumos. Por fim, foi analisado o impacto do tipo de arquivo na capacidade de detecção das estratégias, onde se identificou uma dificuldade em encontrar arquivos do tipo texto com pequenos trechos de similaridades espalhados pelos arquivos. Além disto, verificamos um alto número de falsos positivos de arquivos pdf, devido aos *blocos comuns*, com conteúdo gerado por aplicação.

Trabalhos futuros devem envolver a análise do impacto da remoção dos blocos comuns na detecção de similaridade, assim como a avaliação de outras estratégias, como o HBFT [Lillis et al. 2017] e o FSDS [Moia and Henriques 2017a].

Agradecimentos

Este trabalho foi parcialmente financiado com recursos do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), processo número 120240/2019-0.

Referências

Breitinger, F. and Baier, H. (2013). Similarity preserving hashing: Eligible properties and a new algorithm mrsh-v2. In *Digital Forensics and Cyber Crime: 4th International Conference, ICDF2C 2012, Lafayette, IN, USA, October 25-26, 2012, Revised Selected Papers*, pages 167–182, Berlin, Heidelberg. Springer Berlin Heidelberg.

Breitinger, F., Baier, H., and White, D. (2014a). On the database lookup problem of approximate matching. *Digital Investigation*, 11:S1–S9.

Breitinger, F., Guttman, B., McCarrin, M., Roussev, V., and White, D. (2014b). Approximate matching: definition and terminology. *NIST Special Publication*, 800:168.

Breitinger, F. and Roussev, V. (2014). Automated evaluation of approximate matching algorithms on real data. *Digital Investigation*, 11:S10–S17.

Harichandran, V. S., Breitinger, F., and Baggili, I. (2016). Bytewise approximate matching: The good, the bad, and the unknown. *The Journal of Digital Forensics, Security and Law: JDFSL*, 11(2):59.

Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. *Digital investigation*, 3:91–97.

Lillis, D., Breitinger, F., and Scanlon, M. (2017). Expediting mrsh-v2 approximate matching with hierarchical bloom filter trees. In *International Conference on Digital Forensics and Cyber Crime*, pages 144–157. Springer.

Moia, V. H. G., Breitinger, F., and Henriques, M. A. A. (2020). The impact of excluding common blocks for approximate matching. *Computers & Security*, 89:101676.

Moia, V. H. G. and Henriques, M. A. A. (2017a). Fast similarity digest search: a new strategy for performing queries efficiently with approximate matching. *XVII Brazilian Symposium on information and computational systems security, Brazilian Computer Society (SB)*.

Moia, V. H. G. and Henriques, M. A. A. (2017b). Similarity digest search: A survey and comparative analysis of strategies to perform known file filtering using approximate matching. *Security and Communication Networks*, pages 1–17.

Noll, L. C. (2012). Fowler/Noll/Vo (FNV) hash. Disponível em: `http://www.isthe.com/chongo/tech/comp/fnv/index.html`. Acess. em 15 Set 2020.

Roussev, V. (2010). Data fingerprinting with similarity digests. In *IFIP International Conf. on Digital Forensics*, pages 207–226. Springer.

Roussev, V. (2011). An evaluation of forensic similarity hashes. *Digital investigation*, 8:34–41.

Tridgell, A. (2002). Spamsum. Disponível em: `http://samba.org/ftp/unpacked/junkcode/spamsum`. Acess. em 15 Set 2020.

Winter, C., Schneider, M., and Yannikos, Y. (2013). F2s2: Fast forensic similarity search through indexing piecewise hash signatures. *Digital Investigation*, 10(4):361–371.