Análise do Uso de Diversas Funções de Similaridade no Agrupamento de Casos de Teste Baseados em Modelos de Estado

Eliane Martins[1], Gustavo Machado[2]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 15.064 – 13083-852 – Campinas – SP – Brasil

[2] Instituto de Matemática, Estatística e Computação Científica – Universidade Estadual de Campinas (UNICAMP)
Rua Sérgio Buarque de Holanda, 651 -13083-859 – Campinas – SP – Brasil.

eliane@ic.unicamp.br, gusta_machado@hotmail.com

Abstract. *This paper describes a preliminary empirical study to evaluate the influence of similarity measures on clustering of model-based test cases. The goal was to understand the existing functions with the aim to apply them in the future in a similarity-based test-case prioritization method.*

Resumo. *Este artigo descreve um estudo empírico preliminar com o intuito de avaliar a influência de medidas de similaridade no agrupamento de casos de teste baseados em modelos. O objetivo foi compreender melhor as funções existentes para poder aplicá-las futuramente em um método para priorização de casos de teste baseados em similaridade.*

1. Introdução

Contexto. Um dos problemas práticos que se tem hoje em dia na realização de testes é o grande volume de casos de teste. Isto acontece, por um lado, pelo uso de ferramentas de geração automática de casos de teste, e por outro, pela evolução dos sistemas, pois novos testes são criados para as novas versões, juntando-se aos casos de testes já existentes. A cada vez que um software é modificado, é necessário realizar **testes de regressão**, para determinar se as modificações não introduziram novos *bugs*. No entanto, é inviável aplicar a totalidade dos casos de teste existente, porque a realização de testes tem custos que podem ser elevados, tanto em termos de tempo quanto de recursos necessários. Além disso, em muitos casos, a análise dos resultados é feita manualmente; grande volume de testes inviabiliza uma análise mais acurada. A literatura apresenta algumas técnicas para diminuir o esforço com a realização dos testes de regressão. São elas [Yoo et al. 2012]: i) **minimização ou redução**, que visam eliminar redundância, para com isso diminuir o número de casos de testes a serem executados; ii) **seleção**, que procuram identificar os casos de teste relevantes para as modificações realizadas, e iii) **priorização**, em que se determina a ordem em que os casos de teste devem ser realizados de forma que os defeitos possam ser revelados mais cedo. Neste trabalho o interesse está na priorização de casos de teste. Apesar do objetivo principal não ser a eliminação dos casos de testes, a priorização permite reduzir o volume de testes a serem reaplicados quando não há tempo ou recursos suficientes, pois os casos de teste mais relevantes são executados mais cedo.

O problema. A questão é: quais casos de teste priorizar? O objetivo é realizar primeiro os casos de teste que tenham maior potencial de revelar defeitos. Saber se um caso de

teste pode revelar um defeito antes de aplicá-lo não é tarefa fácil, e geralmente é realizada manualmente por especialistas que precisam analisar a base de testes existente. Porém, pessoas têm limitações quanto ao número de casos de teste que são capazes de analisar. De acordo com alguns estudos empíricos, é possível fazer até 100 comparações manualmente, em média; além deste número, o desempenho começa a diminuir. Em outros termos, um especialista seria capaz de comparar no máximo até 14 casos de teste, o que está longe de corresponder ao tamanho de conjuntos de testes de sistemas reais [Yoo et al. 2009]. Sendo assim, é necessário um apoio automatizado aos especialistas.

A proposta. Nossa proposta é usar técnicas de agrupamento (*clustering*) para ajudar na priorização dos casos de teste. Estas técnicas da análise estatística multivariada são muito usadas em mineração de dados e visam organizar dados em grupos [Vicini 2005]. Os dados que tenham maior semelhança entre si, de acordo com um dado critério, são alocados em um mesmo grupo. Ao final, o que se espera é que exista homogeneidade dentro de um grupo, e heterogeneidade entre grupos. Assim, casos de teste serão agrupados de forma que o especialista possa escolhê-los a partir de grupos diferentes, para obter um subconjunto de testes diversificado. Estudos empíricos mostraram que a diversidade de casos de teste melhora a taxa de detecção de defeitos [Chen et al. 2010, Cartaxo et al.2011, Hemmati et al. 2013]. O uso de agrupamento permite que, ao invés de fazer uma seleção dentre centenas ou milhares de casos de teste, o testador tenha apenas alguns grupos para focar, sendo que o número de grupos pode ser escolhido de acordo com a quantidade de casos de teste que podem ser realizados dentro dos prazos e custos estabelecidos para esta atividade. Para avaliar quão (dis)similares são os casos de teste, existem inúmeras funções de similaridade, e a avaliação de seu impacto sobre técnicas de seleção de casos de testes vem sendo estudada [Hemmati et al. 2013; Coutinho et al. 2016].

Resultados. Neste estudo apresentamos um trabalho inicial que avalia algumas funções para determinar seu impacto sobre o agrupamento de casos de teste. Com isso, procuramos entender melhor estas diferentes funções para determinar quais usar e em que circunstâncias.

Estrutura do texto. O artigo está organizado da seguinte forma: a Seção 2 aborda a priorização de casos de teste e apresenta alguns trabalhos relacionados ao deste estudo. A Seção 3 apresenta alguns fundamentos sobre técnicas de agrupamento, enquanto a Seção 4 apresenta as funções de similaridade utilizadas. A Seção 5 descreve o estudo empírico, cujos resultados estão na Seção 6. A Seção 7 conclui o trabalho.

2. Priorização de casos de teste

A priorização busca reordenar o conjunto de testes de modo que mais defeitos (*bugs*) sejam encontrados mais cedo. No entanto, informações sobre quais casos de teste revelam defeitos geralmente só são conhecidas depois de executá-los. Portanto, é necessário o uso de alguma informação existente antes dos testes que forneçam uma boa indicação do potencial para a revelação de defeitos. Inúmeras técnicas podem ser encontradas na literatura. O mais comum é utilizar informações sobre o código: cobertura de comandos, de blocos básicos, de métodos, por exemplo. Neste caso, assume-se que, quanto maior a cobertura de um caso de teste, maior a chance de ele revelar algum defeito. Por questão de espaço, não podemos detalhar todas as técnicas aqui, mas um levantamento mais detalhado delas pode ser visto em [Yoo et al.2012], e as ferramentas existentes podem ser vistas em [Khan et al. 2016].

Nosso interesse, por estar mais relacionado com o estudo aqui descrito, está nas técnicas baseadas em similaridade. Neste caso, o objetivo é maximizar a diversidade do conjunto de testes, pois assume-se que, quanto maior a diversidade, maior a chance de revelar defeitos. Muitos destes trabalhos propõem o uso de técnicas de agrupamento para reduzir o número de comparações que um especialista humano deve realizar. Os estudos variam conforme o critério utilizado para o agrupamento, bem como a função utilizada para comparar os casos de teste, assim como a técnica de agrupamento utilizada. Leon e Podgurski (2003) utilizaram o perfil de execução dos casos de teste como critério para o agrupamento. Para determinar a similaridade entre casos de teste, foi utilizada a distância Euclidiana; os casos de teste foram representados como contadores do número de execuções de cada bloco básico de execução do programa. O trabalho de Yoo et al. (2009) cria grupos utilizando como critério os rastros de execução (*execution traces*) de programas; estes rastros são representados na forma de cadeias binárias, em que cada bit representa um comando do código fonte que foi executado pelo caso de teste. Para a comparação entre os casos de teste, foi utilizada a distância de Hamming. Os casos de teste foram agrupados utilizando técnica aglomerativa hierárquica (a ser abordada na Seção 3.3). Diferentemente dessas técnicas, no estudo aqui apresentado, os casos de teste são baseados em modelos, e não no código ou em perfil de execução dos mesmos. Os casos de teste são representados como cadeias de caracteres, e nosso objetivo não é, ainda, propor uma técnica de priorização, mas sim, comparar as possíveis funções de similaridade existentes.

Neste sentido, trabalhos mais relacionados com o nosso estudo são os de Coutinho et al., bem como de Hemmati et al., que utilizam técnicas baseadas em similaridade para casos de teste baseados em modelos. Apesar de não serem especificamente voltados para a priorização, estes trabalhos analisam a influência das funções de similaridade na seleção dos casos de teste. No primeiro trabalho os autores mostram, dentre outros, um extenso estudo empírico para investigar a influência de parâmetros tais como representação de casos de teste e funções de similaridade na efetividade da detecção de defeitos. Foram investigadas quatro representações diferentes dos casos de teste, oito funções de similaridade e dez algoritmos de minimização, dentre os quais, algoritmos de agrupamento tanto hierárquicos quando particionais [Hemmati et al. 2013]. Coutinho et al. (2016) analisaram a influência de seis funções de similaridade na taxa de detecção de defeitos. Como dito anteriormente, apresentamos neste trabalho um estudo preliminar, que, portanto, não é tão extenso quanto os trabalhos citados. O diferencial está em que algumas das funções de similaridade estudadas não constam de nenhum dos trabalhos anteriores.

3. Análise de Agrupamentos

A Análise de Agrupamentos (AA) envolve várias técnicas e algoritmos com o objetivo de organizar uma coleção de objetos em grupos ou classes de acordo com características que estes objetos possuem [Linden 2009]. A finalidade é que objetos pertencentes ao mesmo grupo sejam semelhantes entre si e diferentes de objetos pertencentes a outros grupos, com relação a um certo critério de similaridade.

A aplicação de AA consiste, tipicamente, na realização das seguintes etapas [Jain et al. 1999]: i) definição da forma de representação dos dados de entrada a serem agrupados; ii) escolha de uma medida adequada de proximidade (ou similaridade) entre os dados; iii) agrupamento (ou *clustering*) dos dados; iv) definição de uma abstração dos

dados, se necessário e v) avaliação dos resultados.

As subseções a seguir descrevem os passos dados, à exceção de ii), que é apresentado em detalhes na Seção 4, e do passo iv), que é relativo à forma de apresentação dos resultados de forma a ser tratado por um algoritmo ou por um ser humano. Este passo não será considerado aqui, pois consideramos a saída na forma de árvore (visto em 3.2) como adequada aos nossos propósitos. A apresentação é breve, e contém somente os elementos necessários para a compreensão do estudo aqui apresentado. Para uma leitura mais detalhada sobre o assunto, podemos recomendar [Jain et al. 1999] [Vicini et al. 2005].

3.1. Representação dos dados

Os dados a serem agrupados são denominados de padrões ou tuplas [Linden 2009]. Uma tupla x é representada por um vetor com d características (ou atributos): $x = <x_1, x_2, \ldots, x_d>$. O valor **d** é designado como a dimensão da tupla. Um conjunto de tuplas é denotado por $X = \{t_1, t_2, t_3, \ldots, t_n\}$.

Definir quais tuplas e suas características mais apropriadas para o problema não é uma tarefa trivial. No entanto, esta tarefa é crucial, pois esta definição serve de base para a próxima etapa, onde se estabelece a similaridade entre os padrões. No contexto deste trabalho consideramos que um caso de teste é representado de forma textual. Na Seção 5.1 essa descrição é apresentada em detalhes.

3.2. Agrupamentos

As técnicas de agrupamento utilizam as medidas de similaridade para construir grupos de dados similares. Existem inúmeros algoritmos para a realização de agrupamentos, assim, podemos ter diferentes agrupamentos para os mesmos dados. Estes algoritmos podem ser classificados como hierárquicos ou particionais [Jain et al. 1999]. As técnicas hierárquicas permitem criar grupos aninhados, que são representados em uma árvore conhecida como dendrograma. Nesta árvore, uma folha representa um dado, e um nó representa um grupo. Já as técnicas particionais (ou não-hierárquicas), não constroem grupos aninhados. Os mais utilizados são os algoritmos hierárquicos, sendo populares principalmente na área de Bio-informática. Uma vantagem dos algoritmos hierárquicos é que o dendrograma permite visualizar diretamente a forma como os dados se interligam: quanto mais alta a linha ligando dois nós ou folhas, mais tarde foi feito o agrupamento. Uma outra vantagem das técnicas hierárquicas é que não é necessário estabelecer o número de grupos a priori; assim, o especialista de testes pode determinar o número de grupos de acordo com o tempo e recursos disponíveis.

As técnicas hierárquicas podem ser aglomerativas ou divisivas. As técnicas **aglomerativas** criam conjuntos a partir de elementos isolados, enquanto as **divisivas** começam com um grande conjunto, contendo todos os elementos, e divide-o, formando novos conjuntos, e assim continua o processo até chegar a elementos isolados ou até que um critério de parada tenha sido atingido. Neste trabalho, utilizamos a técnica aglomerativa, pois a divisiva é mais custosa [Linden 2009].

Para construir o dendrograma é necessário calcular, a cada iteração, a distância entre os grupos formados, denominada de **distância de ligação** (*linkage distance*). Os

métodos de agrupamento diferem de acordo com a forma como esta distância é calculada. A Figura 1 mostra um exemplo de uma matriz de similaridade e o dendrograma obtido a partir dela. No eixo vertical temos indicadas as distâncias de ligação, representadas pela distância mínima entre grupos (método do vizinho mais próximo). A construção do dendrograma não pressupõe nenhum número de grupos. Para determiná-los, é necessário estabelecer os cortes na árvore. As linhas pontilhadas representam estes cortes. Por exemplo, a linha tracejada indicada pela letra A que corta o eixo vertical na distância de 0,6, permite identificar dois grupos: o primeiro é formado pelo par (1, 5), o segundo pelos elementos (2, 4, 3). Já o corte B, na distância 0,4, leva à formação de 3 grupos: (1,5), (2, 4) e (3). Não há uma forma exata de calcular o melhor número de grupos, mas o ideal é não fazer cortes que incluam nós internos muito distantes. No entanto, no contexto de priorização de casos de teste, cabe à equipe decidir, de acordo com o tempo e recursos disponíveis, qual o melhor corte a ser feito.

Ind. (n)	1	2	3	4	5
1	0	5	10	7	1
2		0	5	2	6
3			0	3	11
4				0	8
5					0

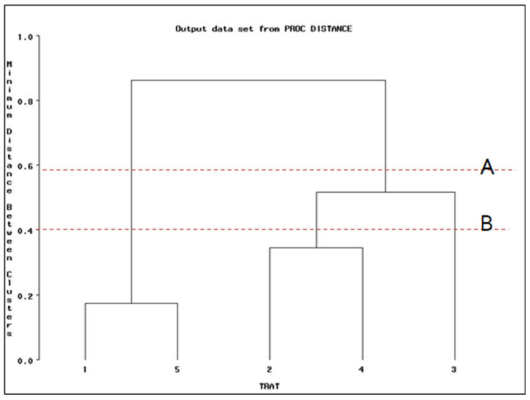


Figura 1. Exemplo de agrupamento para a matriz de similaridade mostrada.

Neste trabalho foi utilizado o método da distância de ligação média (*average linkage*). Mais especificamente, foi utilizado o algoritmo UPGMA (*Unweighted Pair Group Method with Arithmetic Mean*), para a construção da árvore. Este algoritmo é aglomerativo, em que grupos mais próximos são combinados para formar uma hierarquia. A escolha do mais próximo se dá pela média aritmética entre as distâncias entre pares de elementos dos dois grupos. Os pares de grupos que apresentarem a menor média são mais similares, e, portanto, são representados mais próximos na árvore.

3.3. Como avaliar um agrupamento

Pelo que já foi apresentado até então, pode-se constatar que existem inúmeros fatores que influenciam na obtenção de um agrupamento. Como saber se o resultado está bom ou não?

Determinar a validade do agrupamento é importante, pois os algoritmos produzem um resultado, mesmo que os dados sejam totalmente aleatórios, i.e., não formem grupos [Jain et al. 1999]. Existem inúmeras propostas de validação de agrupamentos na literatura, como pode ser visto, por exemplo, em [Vicini et al. 2005]. Uma forma de fazê-lo é avaliando o chamado coeficiente de correlação cofenético (CCC), que serve para medir o grau de ajuste entre a matriz de distância original e a matriz reconstituída com base no dendrograma. Quando ccc > 0,7 conclui-se que o método de agrupamento foi adequado.

4. Funções de similaridade

Um dos fatores fundamentais para a construção de agrupamentos é a definição de uma medida adequada da similaridade entre os dados. Esta medida é calculada com o uso de funções de similaridade (FS), que calculam o grau de semelhança entre dois objetos, medido através de um escore [Hemmati et al. 2013].

De forma simplificada, dados dois objetos x e y de um universo U, a função de similaridade, $sim(x,y) \rightarrow [0,1]$, indica que x tem um certo grau de semelhança com y dentro do intervalo [0,1]. Na verdade, o valor pode estar em qualquer intervalo, mas, para simplificar, aqui considera-se o valor normalizado. O grau de semelhança é chamado de escore de similaridade; se o escore é igual a zero significa que os objetos são totalmente diferentes, e se for igual a 1, que eles são idênticos. Existem diversas funções de similaridade para calcular a semelhança entre os mais variados tipos de objetos: dados numéricos, textos simples (por exemplo, uma palavra) ou longos (e.g. documentos XML ou documentos não estruturados), estruturas de dados complexas (ex.: tuplas), e até imagens, sons e vídeos.

Tendo em vista que consideramos os casos de teste como sendo cadeias de caracteres, foram usadas funções genéricas de similaridade para textos simples, que podem ser classificadas em dois tipos [Gomaa et al. 2013]: (i) baseadas em termos (ou tokens) e (ii) baseadas em caracteres. No primeiro grupo, a cadeia é considerada como um vetor, e cada dimensão do vetor representa a ocorrência ou a frequência de um termo ou palavra presente na cadeia. A distância é medida com base nos elementos comuns e nas diferenças entre os dois vetores.

No segundo tipo, que é o nosso interesse neste estudo, não é necessário gerar uma representação vetorial; as cadeias são comparadas de forma direta. As medidas de distância se baseiam na distância de edição entre as duas cadeias, i.e., no número de operações necessárias para transformar uma cadeia na outra. Estas operações podem ser de inserção, deleção ou substituição de caracteres, bem como a transposição de caracteres adjacentes. Nem todas as funções consideram todas estas operações. Algumas podem ser simuladas usando-se outras, como por exemplo, uma transposição pode ser simulada como duas operações de substituição consecutivas. Para o cálculo da distância, associa-se um custo a estas operações, e o valor da distância entre duas cadeias é dado pela soma dos custos das operações necessárias para transformar uma na outra. Os algoritmos deste grupo consideram a ordem em que os termos aparecem na cadeia, o que é interessante quando um caso de testes representa a ordem temporal em que operações podem ser realizadas, por exemplo.

Neste estudo consideramos as seguintes funções de similaridade: Levenshtein (LEV), Damerau – Levenshtein (DL), Optimal String Alignment (OSA), Jaro-Winkler (JW) e Normalized Compression Distance (NCD). Uma breve descrição de cada uma das distâncias estudadas é dada a seguir [Gomaa et al.2013]:

- Levenshtein (LEV): baseada na distância de edição e considera um custo unitário (valor 1) associado a cada operação de edição (inserção, deleção e substituição). Em outros termos, é baseada na contagem do número de operações necessárias para transformar uma cadeia na outra.
- Damerau – Levenshtein (DL): similar à LEV, só que considera a operação de transposição de símbolos adjacentes na cadeia.

- Optimal String Alignment (OSA): similar à DL, ou seja, conta o número de operações de edição necessárias para transformar uma cadeia na outra, mas com a condição de que nenhuma sub-cadeia seja editada mais do que uma vez [Boytsov 2011].
- Jaro-Winkler (JW): se baseia na frequência e na ordem dos termos em comum nas duas cadeias. A extensão da JW consiste em dar peso maior a cadeias que coincidam desde o início, para um tamanho de prefixo estabelecido.
- Normalized Compression Distance (NCD): a distância normalizada de compressão, proposta por [Cilibrasi 2006], como o nome indica, usa compressores de dados (e.g., gzip, zlib). A ideia é que, ao se concatenar as duas cadeias muito semelhantes entre si, maior a redundância encontrada pelo compressor na cadeia concatenada, x•y. Seja |C(x•y)| o tamanho da cadeia concatenada e comprimida. Assim, quanto maior a semelhança entre x e y, mais |C(x•y)| se aproxima de |C(x)|. NCD é calculada com base nestes tamanhos, sendo normalizada para obter valores entre [0, 1]. Assim, quanto mais próxima de 0, mais semelhantes são as cadeias.

5. Estudo empírico

Nesta seção apresentamos o estudo inicial sobre o uso de técnicas de agrupamento hierárquico aglomerativo para a priorização de casos de teste. Dado que os agrupamentos obtidos são influenciados pela função de distância empregada para determinar quão próximos (ou distantes) estão os casos de teste, neste estudo procuramos avaliar algumas funções de distância para melhor entender quais são mais adequadas para o nosso caso. Os itens a seguir descrevem o estudo realizado.

5.1. Conjunto de testes utilizados

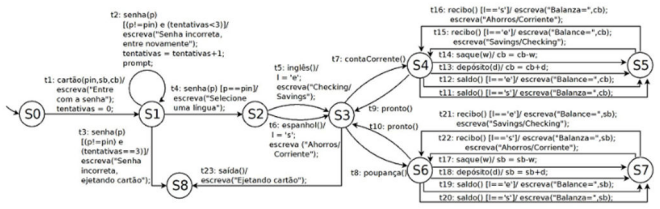
Para este estudo inicial, utilizamos um modelo de estados de um caixa eletrônico, obtido da literatura [Androutsopoulos et al. 2009]. O modelo contém 9 estados e 23 transições, conforme mostra a Figura 2. Os conjuntos de teste gerados a partir deste modelo foram produzidos pela ferramenta StateMutest [Cardoso 2015]. Os casos de teste são representados como um caminho no modelo de estados. Existem várias formas de representar estes caminhos (ver, por exemplo, [Hemmati et al. 2013]), em diferentes níveis de detalhes. Neste estudo inicial, consideramos que um caso de teste é representado como:

<ct> ::= id-tr | id-tr “,”<ct>

em que id-tr ∈ T= {t1, t2, …, tn}, conjunto de transições do modelo de estados.

No lado direito da Figura 2 podemos ver exemplos de casos de teste gerados, representados no formato selecionado.

Para o estudo inicial escolhemos um grupo de controle de 70 casos de teste que, ao serem analisados por uma especialista no assunto, foi constatado que existiam de fato testes muito parecidos e outros muito diferentes, o que seria adequado para nosso estudo. Destes, eliminamos um caso de teste, contendo somente uma transição, que consideramos como um dado atípico (*outlier*) para o contexto. Os demais têm tamanhos entre 2 e 25.



Caso de Teste	Caminho de transições
V1	t1
V2	t1 t2
V3	t1 t2 t2 t2 t3
V4	t1 t4 t5 t8 t10 t7 t9 t7 t14 t15 t13 t15 t12 t15 t14 t15 t14 t15 t12 t15

Figura 2. Modelo de estados do caixa eletrônico e exemplos de casos de teste gerados usando a notação escolhida.

5.2. Funções de similaridade

Para a implementação das distâncias consideramos cada caso de teste como se fosse uma cadeia de texto e utilizamos a ferramenta Harry[1], configurada para normalizar os resultados para o intervalo [0,1], o que facilitou o estudo e a comparação entre eles. Obtivemos assim cinco matrizes de similaridade correspondendo às funções apresentadas na Seção 4.

Além do fato de que os casos de teste derivados de modelos de estado (casos de teste abstrato) são representados por sequências de transições, o uso de funções de distância baseadas em caracteres (c.f. Seção 4) tem a vantagem de não requerer o pré-processamento dos casos de teste, o que seria o caso para funções baseadas em termos. Uma outra vantagem é que a ordem em que as transições são realizadas têm impacto no potencial de detecção de bugs dos casos de teste. No entanto, estas funções são mais custosas em termos de tempo de processamento, dependendo do tamanho das cadeias comparadas. A NCD tem limitações no tamanho das cadeias de acordo com o compressor de dados utilizado. No caso específico da ferramenta Harry, este tamanho é de 16Kb. De um modo geral, os casos de teste gerados não são longos, portanto, as vantagens de usar estas funções foram o fator determinante para a escolha.

6. Resultados

Nosso objetivo neste trabalho foi determinar a influência da escolha de diferentes funções de distância sobre o agrupamento produzido.

Para isso, começamos comparando a distribuição dos valores de distância obtidos pelas diferentes funções, para o mesmo conjunto de dados. A Figura 3 mostra o diagrama de caixas (*box-whisker plot*) representando as distribuições das distâncias calculadas pelas funções estudadas; foi utilizado RStudio[2] para a confecção destes diagramas. Em estatística, este diagrama representa a variação dos dados numéricos em termos de 5 valores: o mínimo, o primeiro quartil, Q1 (25%), a mediana, o terceiro quartil, Q3 (75%) e o máximo. A amplitude da caixa representa o intervalo entre Q1 (lado inferior da caixa) e Q3 (lado superior), ou seja, a caixa representa a distribuição de 50% dos dados. A linha que secciona a caixa representa a mediana. As semi-retas que saem da

[1] http://www.mlsec.org/harry/example1.html
[2] https://www.r-project.org/

caixa (chamadas de *whiskers*) ligam o primeiro quartil ao mínimo, e o terceiro quartil ao valor máximo. Os valores atípicos ou discrepantes (*outliers*), são representados como pontos individuais acima ou abaixo dos valores máximo e mínimo. Esta ferramenta foi utilizada pois permite comparar a variação de uma variável (no caso, a distância entre casos de teste), entre diferentes conjuntos de valores (no caso, obtidos pelas diferentes funções de distância). Além disso, não se pressupõe nenhuma distribuição estatística que os dados deveriam satisfazer [Massart et al 2005].

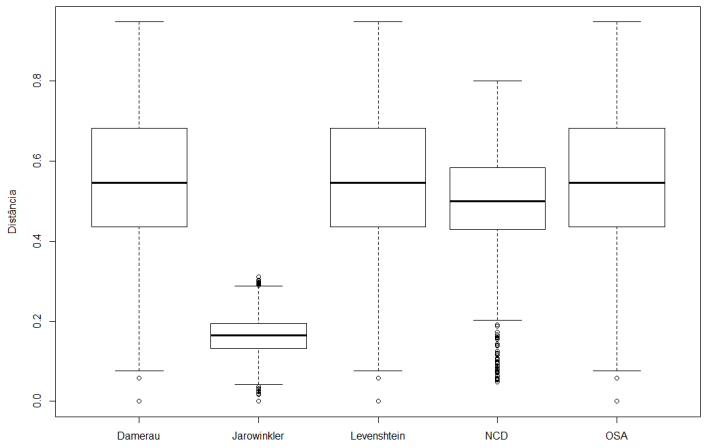


Figura 3. Distribuição dos valores de distância normalizados calculados pelas diferentes funções de distância utilizadas no estudo.

Observando-se os diagramas de caixas lado a lado, pode-se ter uma ideia aproximada se há ou não diferença entre os conjuntos de distâncias geradas. Se as caixas se sobrepõem, é uma indicação de que não há grande diferença entre os conjuntos. Na Figura 3, pode-se notar que as distâncias calculadas pela JW são diferentes das demais. Apesar das caixa ter pequena amplitude, indicando que não há muita dispersão nas distâncias calculadas, há muitos valores discrepantes. Como a JW tende a favorecer casos de teste que tenham prefixo em comum, caso isso não aconteça, eles são classificados como atípicos. Casos de teste gerados a partir de modelos de estado, em geral, têm prefixos em comum, pois todos os caminhos de teste partem do estado inicial, que é único. No entanto, pode haver máquinas de estado que, mesmo sendo inicialmente conexas (i.e., qualquer estado é alcançável a partir do estado inicial), contêm partições a partir de um dado ponto do modelo. É o caso do modelo de estados utilizado, mostrado na Figura 2, para o qual, as transições entre os estados S4 e S5 formam duas partições, dependendo da língua escolhida (inglês ou espanhol). Isso é uma indicação de que a JW não é a distância mais adequada para este tipo de modelo.

A função NCD também apresenta muitos dados discrepantes, em especial, abaixo da distância mínima. Uma razão para isto é que alguns casos de teste menores podem estar contidos em outros maiores, o que faz com que a distância normalizada acabe por ser zero.

Informalmente, analisando os gráficos, podemos constatar que LEV, DL e OSA são similares, como seria de se esperar, pois todas são variações da LEV. A Tabela 1

contém as estatísticas apresentadas nos gráficos; a área em cinza mostra os valores obtidos para estas três funções. Como se pode constatar, os valores obtidos foram exatamente os mesmos para as três funções.

Tabela 1. Estatísticas obtidas para a criação dos *boxplots*.

Dados boxplot	LEV	DL	OSA	JW	NCD
Min	0.00	0.00	0.00	0.00	0.05
Q1	0.44	0.44	0.44	0.13	0.43
Mediana	0.55	0.55	0.55	0.16	0.50
Média	0.55	0.55	0.55	0.16	0.50
Q3	0.68	0.68	0.68	0.19	0.58
Max	0.95	0.95	0.95	0.31	0.80

Uma outra forma de avaliar as funções estudadas foi usando o dendrograma. Nosso objetivo foi determinar se há diferenças nos grupos formados para cada uma das funções. Os dendrogramas foram construídos de acordo com o método UPGMA, descrito brevemente na Seção 3. A Figura 4 apresenta os dendrogramas obtidos.

Para avaliar a qualidade do agrupamento obtido, foi calculado o coeficiente de correlação cofenética (CCC), que mede o grau de preservação das distâncias emparelhadas pelo dendrograma resultante do agrupamento em relação às distâncias originais (c.f. Seção 3.3). A Tabela 2 mostra os coeficientes obtidos para cada função de distância utilizada. Para todas obtivemos um valor superior a 0,7 o que indica que o dendrograma reflete as distâncias calculadas.

Tabela 2. Cálculo do coeficiente de correlação cofenético para as árvores construídas a partir das diferentes funções estudadas.

Distâncias	DL	JW	LEV	NCD	OSA
CCC	0.7844	0.7850	0.7786	0.7346	0.7844

Foi feita também uma comparação visual dos dendrogramas gerados pelas diferentes funções, conforme descrito em [Tendeiro 2005]. As árvores LEV, NCD e OSA são topologicamente distintas, pois possuem esquemas de bifurcação diferentes. Isto é uma indicação de que as diferentes funções realmente produzem agrupamentos diferentes. Já as árvores de DL e OSA são topologicamente idênticas, e também o são do ponto de vista da etiquetagem das folhas, que representam os casos de teste.

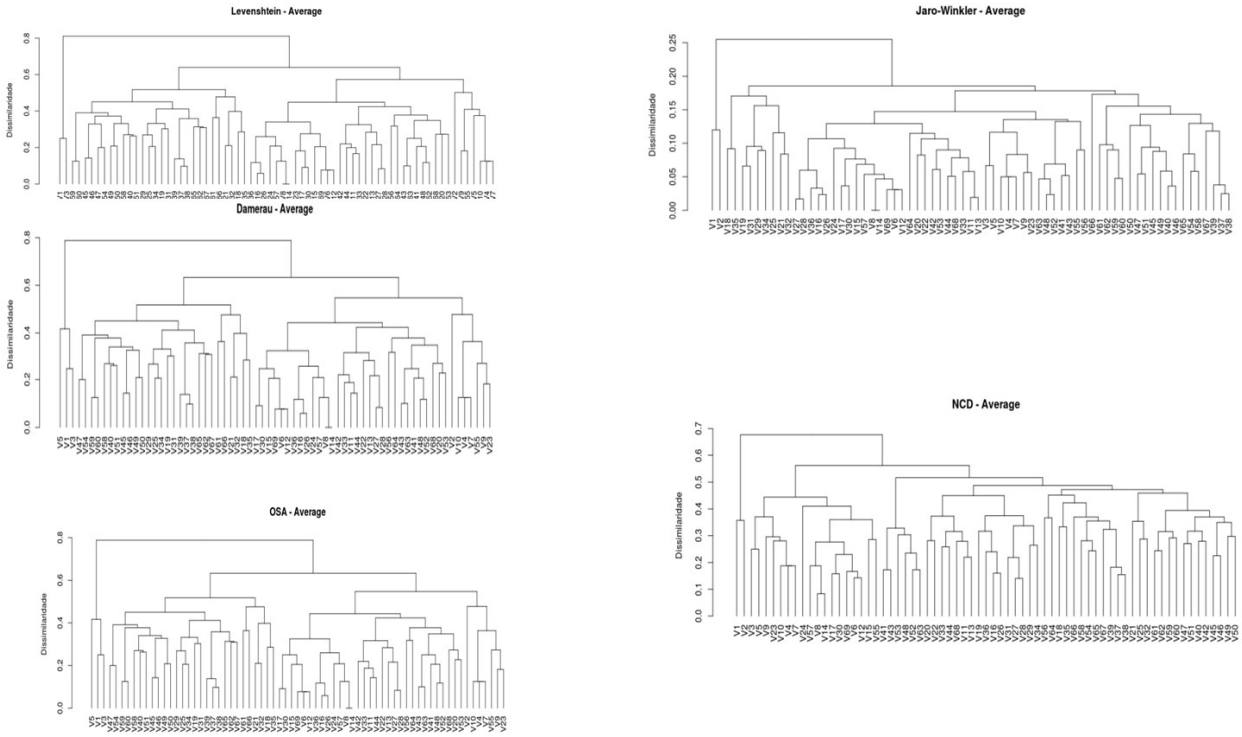


Figura 4. Dendrogramas gerados para as funções de similaridade estudadas.

7. Conclusões e Trabalhos Futuros

Este trabalho é um passo inicial para a proposta de um método de priorização baseada em similaridade para auxiliar equipes de teste na escolha de um subconjunto de testes reduzido, mas eficaz em termos de detecção de defeitos. O objetivo é agrupar casos de teste similares, de forma a reduzir custos com a priorização. Para realizar o agrupamento é, portanto, necessário utilizar uma função para medir o quão (dis)similares são os casos de teste. A literatura contém inúmeras funções, que variam conforme a representação dos casos de teste.

Neste estudo mostramos uma análise inicial de algumas funções similaridade baseadas na comparação de cadeias de caracteres. Estas funções levam em conta a ordem em que os caracteres aparecem na cadeia, o que foi adequado para este estudo, pois casos de teste foram representados como sequências de transições. No entanto, estas funções não são adequadas caso se use uma representação mais complexa dos casos de teste, por exemplo, na forma de entradas e saídas esperadas, pois as funções não são capazes de distinguir as entradas das saídas. Outras funções e outras representações de casos de teste estão sendo estudadas.

Este estudo preliminar serviu para estabelecer o método de análise a ser realizado em trabalhos subsequentes. Uma limitação do estudo foi o pequeno volume dos casos de teste analisados. Por isso, como próximo passo, o método será aplicado a um conjunto maior de casos de teste, tendo como base modelos utilizados em aplicações reais. Além das métricas utilizadas neste estudo para avaliar a qualidade dos agrupamentos produzidos, analisaremos também o potencial de revelação de defeitos para determinar a

homogeneidade dos agrupamentos obtidos.

Bibliografia

[Androutsopoulos et al. 2009] Androutsopoulos, K.; Gold, N. ; Harman, M. ; Li, Z. ; and Tratt, L. A theoretical and empirical study of EFSM dependence. In Proc. ICSM'09: 25th IEEE Int. Conf. on Software Maintenance, pages 287–296, September 2009.

[Boytsov 2011] Boytsov, Leonid (May 2011). "Indexing methods for approximate dictionary searching". Journal of Experimental Algorithmics. Association for Computing Machinery (ACM). 16. Site: http://boytsov.info/pubs/jea2011.pdf.

[Cardoso 2015] CARDOSO, W. F. F. StateMutest: uma ferramenta de apoio ao teste baseado em modelos de estado estendidos. Dissertação de Mestrado, Instituto de Computação, Unicamp. 2015.

[Chen et al. 2010] Chen, T. Y., Kuo, F. C., Merkel, R. G., & Tse, T. H. (2010). Adaptive random testing: The art of test case diversity. Journal of Systems and Software, 83(1), 60–66.

[Cilibrasi 2006] Cilibrasi, R. (2006) "Statistical Inference Through Data Compression". Institute for Logic, Language and Computation, Universiteit van Amsterdam.

[Coutinho et al. 2016] Coutinho, A. E. V. B., Cartaxo, E. G., & de Lima Machado, P. D. (2016). Analysis of distance functions for similarity-based test suite reduction in the context of model-based testing. Software Quality Journal, 24(2), 407-445.

[Gomaa et al.2013] Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. International Journal of Computer Applications, 68(13). Site: https://pdfs.semanticscholar.org/5b5c/a878c534aee3882a038ef9e82f46e102131b.pdf

[Hemmati et al. 2013] Hemmati, H., Arcuri, A., & Briand, L. (2013). Achieving scalable model-based testing through test case diversity. ACM Transactions on Software Engineering and Methodology (TOSEM), 22(1), 6.

[Jain et al. 1999] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. ACM computing surveys (CSUR), 31(3), 264-323. Site: http://eprints.library.iisc.ernet.in/273/1/p264-jain.pdf

Khan, S. U. R., Lee, S. P., Ahmad, R. W., Akhunzada, A., & Chang, V. (2016). A survey on Test Suite Reduction frameworks and tools. International Journal of Information Management, 36(6), 963-975.

[Leon & Podgurski 2003] Leon, David, and Andy Podgurski. "A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases." In Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on, pp. 442-453. IEEE, 2003. Site (visto em março/2018): https://pdfs.semanticscholar.org/1e93/bad85c9037304dbd247c61a7c7031b06618f.pdf

[Linden 2009] Linden, R. Técnicas de Agrupamento. Tutorial. Revista de Sistemas de Informação da FSMA n. 4 (2009) pp. 18-36. Site: http://www.fsma.edu.br/si/edicao4/FSMA_SI_2009_2_Tutorial.pdf

[Massart et al 2005] D.L. Massart,a J. Smeyers-Verbeke,a X. Caprona and Karin Schlesie. Visual Presentation of Data by Means of Box Plots. LC•GC Europe 18(4) 215–218 (2005).

[Tendeiro 2005] Tendeiro, J. N. Comparação de Dendrogramas: Obtenção de Distribuições Empíricas de Alguns Coeficientes. Dissertação de mestrado do Departamento de Engenharia Civil da Faculdade de Engenharia da Universidade do Porto. Janeiro/2005.

[Vicini et al. 2005] Vicini, L., & SOUZA, A. M. (2005). Análise multivariada da teoria à prática. Santa Maria: UFSM, CCNE. Site: http://w3.ufsm.br/adriano/livro/Caderno%20dedatico%20multivariada%20-%20LIVRO%20FINAL%201.pdf

[Yoo et al. 2009] Yoo S, Harman M, Tonella P, Susi A. Clustering test cases to achieve effective & scalable prioritisation incorporating expert knowledge. Proceedings of International Symposium on Software Testing and Analysis (ISSTA 2009), ACM Press, 2009; 201–211.

[Yoo et al.2012] Yoo, S., & Harman, M. (2012). Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability, 22(2), 67-120.