MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

AVALIAÇÕES EXPERIMENTAIS DE HIPER-HEURÍSTICAS PARA TESTE DE SOFTWARE ESPACIAL

RELATÓRIO DE FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA

(PIBIC/CNPq/INPE)

Aluno: Gabriel de Sousa Matsumura (IFSP-CAR, Bolsista PIBIC/CNPq)

E-mail: sousa.matsumura@gmail.com

Orientador: Dr. Valdivino Alexandre de Santiago Júnior
(LABAC/COCTE/INPE)

E-mail: valdivino.santiago@inpe.br

Junho de 2019

RESUMO

Em sistemas críticos, tais como satélites e aplicações de balões estratosféricos desenvolvidos no INPE, o processo de teste de software é vital para identificar defeitos, aumentar a qualidade do software e prevenir a ocorrência de atrasos e prejuízos. No entanto, realizar testes eficientes em um prazo limitado é uma tarefa árdua e desafiadora. A Otimização em Teste de Software (OTS) é uma subárea da Otimização em Engenharia de Software (OES), e tal subárea formula o problema de testar um software como um problema de otimização. OTS tem chamado atenção da comunidade acadêmica de teste de software, onde Meta-Heurísticas, como Algoritmos Evolucionários (e.g. Algoritmos Genéticos), têm sido usadas abordando teste de interação combinatória, teste baseados em modelos, entre outros. No entanto, Meta-Heurísticas sofrem da baixa habilidade de generalização e, sendo assim, um dos caminhos que a comunidade de Pesquisa Operacional tem perseguido, e que a comunidade de OTS começa a investigar, é o uso de Hiper-Heurísticas. As Hiper-Heurísticas possuem como um de seus apelos uma maior capacidade de generalização onde tais soluções objetivam resolver uma classe de problemas ao invés de um problema específico. Dada a crescente quantidade de Hiper-Heurísticas sendo propostas, é conveniente realizar avaliações experimentais rigorosas para evidenciar quais Hiper-Heurísticas são mais adequadas para tratar do problema de geração de casos/dados de teste, especialmente quando o sistema sob teste é um software espacial. Esse projeto de pesquisa possui dois objetivos específicos: a.) Realizar avaliações experimentais rigorosas de Hiper-Heurísticas de seleção no contexto de Teste de Software, considerando métricas como custo e efetividade, para identificar qual Hiper-Heurística de seleção possui melhor desempenho; b.) Considerar, no contexto das avaliações experimentais, softwares espaciais que o INPE vem desenvolvendo como estudos de caso, para melhor caracterizar o desempenho das Hiper-Heurísticas de seleção nesse domínio. Esse relatório apresenta as atividades desenvolvidas no período de 01 de agosto de 2018 a 30 de junho de 2019.

1.) INTRODUÇÃO

A Engenharia de Software Experimental é uma área que procura determinar evidências sobre a adequação, qualidade, custo, e riscos associados às metodologias, métodos e ferramentas que são usados no desenvolvimento de sistemas de software [Shull et al. 2008]. O objetivo final dessa área de pesquisa é formar um corpo de conhecimento validado experimentalmente para dar apoio a tomada de decisão, no contexto da Engenharia de Software. É uma área de pesquisa muito relevante pois pode fornecer aos pesquisadores, engenheiros e cientistas da computação, diretivas e sugestões para escolher, com base em experimentações e procedimentos sistemáticos, uma certa metodologia, método, técnica para melhorar o desenvolvimento de produtos de software.

Teste de Software é, provavelmente, o processo mais adotado, na prática, entre todos relacionadas à área de Verificação & Validação da Engenharia de Software. O objetivo de testar um produto de software é encontrar defeitos no código-fonte do mesmo. Inúmeras teorias, metodologias, abordagens têm sido propostas e/ou usadas para as diversas atividades do processo de Teste de Software. Uma das atividades do processo de Teste mais estudada, mas que ainda apresenta diversos desafios, é a geração de casos/dados de teste. No fundo, dado que a execução de teste exaustivo não é viável, a ideia é utilizar de formas para selecionar, de infinitas possibilidades, um conjunto de dados de entrada de teste do domínio de entrada de um programa P, de forma a detectar o maior número possível de defeitos.

Dentre as diversas abordagens que estão sendo propostas para gerar casos/dados de teste, uma das que se destaca no momento é a Otimização em Teste de Software (OTS). Por meio da formulação do problema de testar software como um problema de otimização, OTS, uma subárea da Otimização em Engenharia de Software (OES), têm atraído bastante atenção da comunidade acadêmica nos últimos anos [Harman et al. 2015]. A razão para isso são os benefícios que se pode obter em função da aplicação de meta-heurísticas já consagradas na solução de problemas reais de busca em diversos campos do conhecimento, tais como Pesquisa Operacional e Inteligência Artificial. Atualmente, existe uma quantidade considerável de estudos abordando vários tipos de testes, tais como testes funcionais [Wegener

e Bühler 2004], testes de segurança [Everson e Fieldsend 2006], teste de interação combinatória [Garvin et al. 2011][Petke et al. 2015], testes baseados em modelos de estados [Asoudeh e Labiche 2014], entre outros.

A comunidade de OTS tem dedicado esforços a problemas multiobjetivos, onde muitos e possivelmente conflitantes objetivos de teste são considerados [Ferrer et al. 2012][Mondal et al. 2015][Shahbazi e Miller 2016]. Além disso, a maioria dos estudos em OTS têm adotado o uso de meta-heurísticas tais como Algoritmos Evolucionários (e.g. Algoritmo Genético (AG) [Petke et al. 2015]), Particle Swarm Optimization (PSO) [Mahmoud e Ahmed 2015] e Simulated Annealing (SA) [Garvin et al. 2011]. No entanto, apesar do histórico bem sucedido da aplicação de meta-heurísticas na solução de problemas de busca reais em diversos domínios de aplicação, tais como agendamento e alocação de espaço, ainda não é tão trivial aplicar meta-heurísticas a novos problemas de otimização ou mesmo novas instâncias de problemas similares [Burke et al. 2013]. Algumas das razões de tais dificuldades são o número usualmente alto de parâmetros ou escolhas de algoritmos que o profissional deve definir, assim como a ausência de orientações adequadas para selecioná-los. Consequentemente, meta-heurísticas são técnicas que possuem baixa capacidade de generalização.

Uma solução para os problemas mencionados acima é pelo uso de hiper-heurísticas. De acordo com [Burke et al. 2013], uma hiper-heurística é “um método de busca ou mecanismo de aprendizagem para selecionar ou gerar heurísticas para resolver problemas de busca computacional”. Assim, em hiper-heurística, a busca é realizada em um espaço de busca de heurísticas (ou componentes de heurística) em vez de ser realizada diretamente no espaço de decisão (espaço de soluções). Assim, as hiper-heurísticas são mais adequadas para resolver uma classe de problemas, em vez de um problema específico, baseadas em um conjunto de Heurísticas de Baixo Nível (HBNs). A ideia é selecionar ou gerar automaticamente LLHs para resolver problemas e, ao mesmo tempo, não exigir do testador um conhecimento amplo sobre parametrização de heurísticas, de modo que é mais fácil aplicar o método a uma gama variada de situações.

Hiper-heurísticas podem ser classificadas de acordo com a natureza do espaço de busca das heurísticas [Burke et al. 2010][Burke et al. 2013]. De

acordo com essa dimensão, existem as hiper-heurísticas de seleção e as hiper-heurísticas de geração. Hiper-heurísticas de seleção se referem àquelas abordagens para selecionar HBNs existentes, enquanto as hiper-heurísticas de geração consideram criar novas heurísticas a partir dos componentes de heurísticas já existentes.

A comunidade de OTS tem gradualmente adotado abordagens baseadas em hiper-heurísticas como Cohen recentemente sugeriu [Cohen 2017]. No entanto, existem muitos poucos trabalhos publicados nesse sentido. Propósitos tais como a geração de dados para teste de interação combinatória [Jia et al. 2015][Zamli et al. 2016], problema de ordem de teste e integração [Mariani et al. 2016][Guizzo et al. 2017], e derivação de produtos para teste de Linhas de Produto de Software [Strickler et al. 2016][Ferreira et al. 2017] têm sido estudados. Porém, essas iniciativas ainda abordam muito pouco o espectro de atividades relacionado a Teste de Software.

Uma outra observação é que as hiper-heurísticas de seleção têm preponderado sobre as de geração no contexto de OTS. Uma explicação para esse fato é que as hiper-heurísticas de seleção têm menor complexidade para serem implementadas se comparadas às hiper-heurísticas de geração. Esse projeto de pesquisa é incluído no contexto de hiper-heurísticas de seleção. E para Teste de Software, os métodos de seleção das hiper-heurísticas de seleção que têm preponderado são Choice Function (CF) [Cowling et al. 2001] e versões modificadas de tal método, e abordagens baseadas em Multi-Armed Bandit (MAB) [Fialho et al. 2009].

Como é comum no contexto de otimização, indicadores de qualidade tais como hipervolume e distância geracional invertida têm sido usados para avaliar o desempenho de hiper-heurísticas na perspectiva de OTS. No entanto, a comunidade de Teste de Software está acostumada a usar outras métricas que são de suma importância para avaliar o desempenho de abordagens que estão sendo propostas. Por exemplo, se uma nova metodologia é proposta para gerar suites de teste (conjuntos de casos de teste), é importante saber quão custosa (em termos, por exemplo, da quantidade de casos de teste gerados) é tal metodologia se comparada a outras. A princípio, quanto menor o conjunto de casos de teste gerados por uma abordagem, melhor pois isso demanda, de uma maneira geral, menos tempo para ser executado.

Outro aspecto é em termos de efetividade, que pode ser definida como a habilidade de uma abordagem em encontrar defeitos nos produtos de software. Quanto maior a efetividade, melhor. Enquanto alguns trabalham no contexto de Linhas de Produto de Software tinham como objetivo a minimização do número de mutantes vivos [Strickler et al. 2016][Ferreira et al. 2017], e isso de alguma forma se relaciona a efetividade, não está totalmente claro qual é a efetividade de diversas hiper-heurísticas de seleção levando em consideração a atividade de geração de casos/dados de teste.

Para resumir, os resultados fornecidos pelos trabalhos publicados pela comunidade de Teste de Software para evidenciar o custo e efetividade de hiper-heurísticas de seleção relacionados à atividade de geração de casos/dados de teste ainda não são suficientes para que seja possível afirmar, com convicção, que uma certa hiper-heurística de seleção é mais adequada do que outra. Desse modo, é muito interessante que novas avaliações rigorosas sejam conduzidas, de acordo com as diretivas da Engenharia de Software Experimental, para que essa conclusão possa ser tirada de forma a dar subsídios aos profissionais para a escolha da melhor opção. Portanto, experimentos controlados ou quasiexperimentos [Zannier et al. 2006][Balera e Santiago Júnior 2016][Balera e Santiago Júnior 2017] devem ser realizados para esse propósito.

Software embarcado em computadores de sistemas espaciais, tais como foguetes, satélites e balões estratosféricos, são críticos. Um defeito em tais produtos de software pode ocasionar a perda da missão inteira. Existem exemplos clássicos na literatura de destruição de aplicações espaciais devido a defeitos de software, como no caso do foguete Ariane-5 da ESA [Dalmau e Gigou 1997] e o Mars Climate Orbiter da NASA [Isbell e Savage 1999]. O INPE vem desenvolvendo diversas aplicações espaciais, tais como os satélites CBERS e Amazônia, e, desse modo, é muito importante garantir que os softwares que estarão embarcados em computadores de tais sistemas espaciais tenham uma alta qualidade. No contexto dessa pesquisa que almeja avaliar experimentalmente hiper-heurísticas de seleção para Teste de Software, um dos estudos de caso que será levado em consideração será o Software for the Payload Data Handling Computer (SWPDC), que foi desenvolvido no contexto de projeto de pesquisa do INPE, fomentado pela

FINEP e com participação de uma empresa de software nacional [Santiago Júnior e Vijaykumar 2012]. O SWPDC foi tomado como base para o desenvolvimento do software embarcado no computador do Subsistema de Gestão de Bordo (SGB) do projeto experimento científico protoMIRAX, da Coordenação-Geral de Ciências Espaciais e Atmosféricas (CGCEA) do INPE. O objetivo principal do protoMIRAX, um protótipo da missão Monitor e Imageador de Raios X (MIRAX) e que voará por meio de um balão estratosférico, é testar, em ambiente quase espacial, diversos subsistemas da carga útil associada à missão MIRAX. Outros estudos de caso do INPE poderão ser também considerados no escopo dessa pesquisa.

Portanto, os objetivos específicos desse projeto são:

a.) Realizar avaliações experimentais rigorosas de hiper-heurísticas de seleção no contexto de Teste de Software, considerando métricas como custo e efetividade, para identificar qual hiper-heurística de seleção possui melhor desempenho;

b.) Considerar, no contexto das avaliações experimentais, softwares espaciais que o INPE vem desenvolvendo como estudos de caso, para melhor caracterizar o desempenho das hiper-heurísticas de seleção nesse domínio.

Esse relatório apresenta as atividades desenvolvidas no período de **01 de agosto de 2018 a 30 de junho de 2019.** Esse é o relatório final do projeto.

2.) CRONOGRAMA DE ATIVIDADES E ETAPAS CONCLUÍDAS

Conforme mostrado no “Formulário para Solicitação de Bolsa PIBIC”, a metodologia a ser empregada para atender aos objetivos do projeto está descrita a seguir.

1. Estudar a fundamentação teórica relativa ao projeto. Especificamente, se familiarizar com os conceitos relacionados à Teste de Software (com particular ênfase em Otimização em Teste de Software via hiper-heurísticas de seleção) e Engenharia de Software Experimental;

2. Definir quais hiper-heurísticas de seleção serão usadas para fazer parte das avaliações experimentais;

3. Definir os produtos de software da área espacial para serem usados nas avaliações experimentais. Simulador de software embarcado em computadores de satélite e de aplicação de balão estratosférico de projetos de pesquisa do INPE é um exemplo de estudo de caso que será considerado;

4. Adaptar as hiper-heurísticas de seleção para que seja possível realizar as avaliações experimentais. Com isso, será desenvolvida uma ferramenta computacional, que implementará as hiper-heurísticas de seleção escolhidas, para dar apoio as avaliações;

5. Realizar avaliação experimental rigorosa considerando como métrica custo (tamanho das suites de teste);

6. Realizar avaliação experimental rigorosa considerando como métrica efetividade (habilidade de encontrar defeitos);

7. Submeter artigo para conferência e/ou workshop e/ou simpósio na área de Engenharia de Software e/ou Pesquisa Operacional e/ou Inteligência Artificial, e elaborar relatório final de atividades.

O cronograma para desenvolvimento das atividades da metodologia está mostrado na Figura 2 a seguir. O número das atividades está de acordo com os números mostrados acima. Cada uma das colunas de Ano I e II representa um mês.

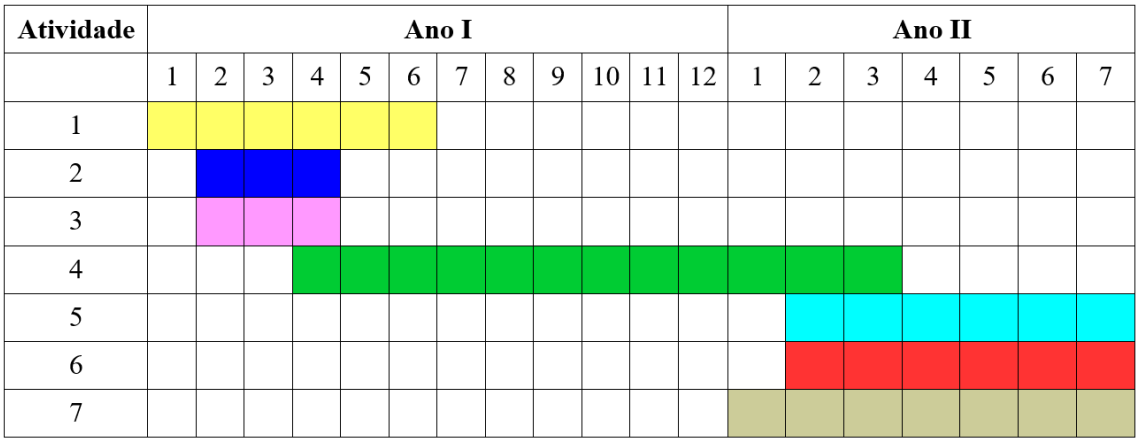

Figura 1 - Cronograma de atividades

Portanto, esse relatório compreende o **mês 1 do Ano I (agosto/2018) ao mês 11 do Ano I (junho/2019)**. Considerando as atividades previstas para serem desenvolvidas, mostradas na Figura 2, a Tabela 1 a seguir mostra as

atividades concluídas considerando o período total do projeto, a que se refere esse relatório de acompanhamento.

Tabela 1 - Etapas Concluídas e a Concluir

	Atividades da Metodologia	**Previsão**	**Realização**
1	Estudar a fundamentação teórica relativa ao projeto. Especificamente, se familiarizar com os conceitos relacionados à Teste de Software (com particular ênfase em Otimização em Teste de Software via hiper-heurísticas de seleção) e Engenharia de Software Experimental.	100%	100%
2	Definir quais hiper-heurísticas de seleção serão usadas para fazer parte das avaliações experimentais.	100%	100%
3	Definir os produtos de software da área espacial para serem usados nas avaliações experimentais. Simulador de software embarcado em computadores de satélite e de aplicação de balão estratosférico de projetos de pesquisa do INPE é um exemplo de estudo de caso que será considerado.	100%	100%
4	Adaptar as hiper-heurísticas de seleção para que seja possível realizar as avaliações experimentais. Com isso, será desenvolvida uma ferramenta computacional, que implementará as hiper-heurísticas de seleção escolhidas, para dar apoio as avaliações.	75%	40%
5	Realizar avaliação experimental rigorosa considerando como métrica custo (tamanho das suites de teste).	0%	0%
6	Realizar avaliação experimental rigorosa considerando como métrica efetividade (habilidade de encontrar defeitos).	0%	0%
7	Submeter artigo para conferência e/ou workshop e/ou simpósio na área de Engenharia de Software e/ou Pesquisa Operacional e/ou Inteligência Artificial, e elaborar relatório final de atividades.	0%	0%

Na Tabela 1 acima, a coluna **Previsão** mostra a porcentagem prevista para a realização da atividade, e a coluna **Realização** mostra a porcentagem realmente realizada da atividade, considerando o período a que se refere esse relatório (01 de agosto de 2018 a 30 de junho de 2019).

Na atividade 1, foram estudados os fundamentos teóricos relacionados ao projeto, conceitos relacionados à Teste de Software, Análise Experimental Rigorosa, Hiper-heurísticas, Algoritmos Evolucionários Multiobjetivo (MOEAs) e Software Espacial. Esta tarefa foi totalmente concluída (100%).

A atividade 2 foi totalmente concluída (100%), onde hiper-heurísticas de seleção foram selecionadas, tais como as que possuem métodos de seleção Choice Function, Simple Random e Simulated Annealing com Reinforcement

Learning, com métodos de aceitação All Moves e Great-Deluge. Para as Heurísticas de Baixo Nível (HBNs) foram selecionadas as meta-heurísticas NSGA-II, MOMBI-II e MOEA-D/STM.

A atividade 3 foi totalmente concluída (100 %), onde o produto de sofware espacial SWPDC, que é um simulador de software embarcado em computadores de satélite e de aplicação de balão estratosférico de projeto de pesquisa do INPE, foi selecionado como estudo de caso. Outros produtos de software do INPE estão sendo investigados para serem usados, assim como softwares do benchmark SF100.

Sobre a atividade 4, a mesma não era para ser totalmente concluída no período a que se refere esse relatório técnico. Essa é uma das atividades mais desafiadoras do projeto e, atualmente, as seguintes tarefas já foram desenvolvidas relacionadas a essa atividade:

i.) Desenvolvimento de um mecanismo de leitura de código-fonte Java, que obtém e estrutura os dados necessários para a geração dos testes unitários via código-fonte, utilizando a ferramenta ANTLR, versão 4.7.2;

ii.) Definição dos parâmetros de configuração das HBNs. A codificação das soluções (cromossomos) será do tipo inteira;

iii.) Geração de códigos mutantes dos estudos de caso via a ferramenta MuJava, para dar apoio as avaliações experimentais.

As demais atividades (5, 6, 7) não estavam previstas para serem realizadas no período a que se refere esse relatório de acompanhamento. Os detalhamentos do desenvolvimento das atividades estão apresentados a seguir.

3.) ATIVIDADE 1: FUNDAMENTAÇÃO TEÓRICA

Nesta atividade o objetivo foi ler e realizar o fichamento de 13 referências selecionadas, visando o conhecimento de conceitos relacionados à Otimização em Teste de Software e Engenharia de Software Experimental. Alguns dos principais assuntos, escolhidos arbitrariamente, encontram-se na

Tabela 2 e as obras selecionadas estão na Tabela 3. Além destas 13 obras, outras também foram lidas.

Tabela 2 - Numeração dos Assuntos

Nº	Assuntos
1	Teste de Software
2	Análise Experimental Rigorosa
3	Hiper-heurísticas
4	Algoritmos Evolucionários Multiobjetivo

Tabela 3 - Referências Selecionadas para Fichamento

	Referências	Assuntos	Status
1	[Shull et al. 2008]	2.	Corrigido
2	[Balera e Santiago Júnior 2017]	1, 2.	Corrigido
3	[Burke et al. 2013]	3.	Corrigido
4	[Jia et al. 2015]	1, 3.	Corrigido
5	[Zamli et al. 2016]	1, 2, 3.	Corrigido
6	[Zamli et al. 2017]	1, 2, 3.	Lido
7	[Carvalho et al. 2015]	1, 3, 4.	Lido
8	[Guizzo et al. 2017]	1, 2, 3, 4.	Lido
9	[Ferreira et al. 2017]	1, 2, 3, 4.	Lido
10	[Deb et al. 2002]	4.	Lido
11	[Gómez e Coello 2015]	2, 4.	Lido
12	[Li et al. 2014]	2, 4.	Lido
13	[Santiago Júnior e Vijaykumar 2012]	1.	Fichado

Os fichamentos podem ser considerados uma forma de resumo das informações e dados mais relevantes de uma obra, a partir da compreensão e interpretação do leitor, facilitando aprendizado do conteúdo e o seu uso como fundamentação teórica.

Os fichamentos foram estruturados nas seguintes seções: Referência; Classificação; Motivação; Objetivo; Descrição de Metodologias/Técnicas/Métodos; Domínio de aplicação; e Contexto de validação.

Os status das obras, que podem ser observados na Tabela 3, são: lido, fichado e corrigido. Todos os artigos foram lidos. Os trabalhos com status de lido possuem pelo menos as seções referência, classificação e domínio de aplicação já fichados e foram contadas como 50% prontas. No caso do artigo com status fichado falta apenas realizar a correção e por isso foi considerado como 90% pronto. Por último, as obras corrigidas estão 100% prontas. Portanto, pela média das porcentagens, a Atividade 1 está 80% concluída.

No resto desta seção encontram-se todos os fichamemtos realizados.

3.1. FICHAMENTO 1 [Shull et al. 2008]

Referência:

[Rosenberg 2008] J. Rosenberg. Chapter 6 - Statistical Methods and Measurement. In Forrest Shull, Janice Singer, Dag I.K. Sjøberg, editors, Guide to Advanced Empirical Software Engineering, pages 155-184, Springer-Verlag, London, 2018.

Classificação (nova abordagem, revisão, survey): Revisão / Livro de Referência.

Motivação do artigo: A principal motivação deste capítulo é a confusão existente quanto as melhores maneiras de definir, coletar e utilizar medições estatísticas durante o processo de engenharia de software.

Objetivo do artigo: O objetivo deste capítulo é apresentar um conjunto de técnicas para a definição e análise de métricas, cada uma aplicável em determinados contextos tendo como objetivos específicos possibilitar que o leitor seja capaz de:

- Definir métricas realísticas e válidas que possam ser coletadas tendo em vista as regras e prazos limites envolvidos;

- Tomar decisões baseadas em formas precisas de análise dos dados mensurados estatisticamente;

- Discutir o contexto e alguns problemas fundamentais de mensuração;

- Discutir a definição e avaliação de métricas;

- Cobrir métodos de descrição, comparação e predição de medições de dados qualitativos e quantitativos;

- Discutir o contexto da atividade de mensuração em relação ao tópico de qualidade de dados.

Rápida descrição sobre as metodologias/técnicas/métodos relacionados ao artigo: Sobre este trabalho, destaco alguns tópicos que selecionei a respeito das metodologias, técnicas e métodos apresentados por Rosenberg (2008), que podem ser úteis no futuro: Criação de Métricas Efetivas; Análise da Medição de Dados (Estáticos e Dinâmicos) e Qualidade de Dados.

Sobre o tópico **Criação de Métricas Efetivas** (pag. 158-164), o autor inicia dividindo as métricas em simples (que remetem a quatro escalas da Teoria da Medição Clássica – Nominais, Ordinais, Intervalar e Razão) e compostas (combinações, geralmente aritméticas, das métricas simples).

As quatro escalas de medição são definidas de acordo com o conjunto de operações matemáticas adequadas a estes tipos de dados, de forma que da escala nominal até a escala razão existe um aumento de operações matematicamente significativas, sendo que:

$$\theta(X) \subset \theta(R) \wedge \theta(N) \subset \theta(X)$$

Onde X é uma escala qualquer, N é a escala nominal, R a razão eθ a função cujo parâmetro é uma escala e que tem como imagem o conjunto de operações permitidas. Desta forma, as próprias escalas de medição seguem uma escala ordinal.

É necessário definir métricas com o objetivo de responder a questões especificamente levantadas para a tomada de decisões. No entanto, as escalas só terão suas características suficientemente levantadas para a aplicação da Teoria da Medição após a realização da pesquisa empírica para avaliar se esta metodologia é adequada ou não ao conjunto de dados.

Ainda sobre este tópico, é indicado que não basta que uma métrica seja bem definida matematicamente para que seja uma métrica efetiva. É essencial

que ela possua a tríade de características *precisão*, *confiabilidade* e *validade* em quantidade adequada ao contexto.

Quanto a precisão, é destacada a quantidade de dígitos significativos (número de 'zeros' após a vírgula). Uma falha frequente é o esquecimento de que uma métrica composta deve ter uma quantidade de dígitos significativos menor ou igual a menor quantidade de dígitos entre as métricas que a compõe. Isso se deve ao fato de que uma medida composta tende a ser menos precisa do que as medidas que a compõe.

A confiabilidade se trata do quanto as medidas amostrais são consistentes em relação a repetidas observações sob as mesmas circunstâncias. A necessidade de confiabilidade é ainda maior ao se manipular medidas contínuas, sendo comum o uso desses testes para quantificar essa característica, um exemplo é o *coeficiente alfa de Cronbach*, sendo dado pela seguinte equação:

$$\alpha = \frac{k \cdot r}{1 + (k-1) \cdot r}$$

Um coeficiente$\alpha \geq 0{,}7$ é considerado o mínimo aceitável, no entanto, isto não é o suficiente para afirmar que os seus dados são confiáveis, pois a confiabilidade varia de amostra para amostra. Uma característica limitante do coeficiente alfa de Cronbach é o fato de assumir que os dados analisados são unidimensionais e, por isso, é uma medição restrita. O coeficiente ômega supera esta limitação, mas não é abordado neste trabalho.

Já a validade é um conceito multifacetado que, a grosso modo, diz respeito ao aspecto da medida ser congruente com a propriedade medida na amostra. A validade pode ser compreendida em três níveis: conteúdo, critério e construto. A validade em nível de conteúdo busca determinar se a métrica reflete adequadamente o domínio estudado. O nível de critério consiste no grau de eficácia que uma medição de certo objeto tem em predizer um determinado comportamento. A validade de construção se refere ao quanto uma medição é válida em determinar um conceito não observável (por exemplo: inteligência, amor, etc), estabelecendo uma correspondência entre medidas observáveis e uma determinada teoria. O nível de validade de construto é frequentemente usado em estudos psicométricos. A validade também possui modelos

matemáticos para quantificá-la, mas esses modelos não foram abordados neste trabalho.

O autor destaca a importância de se levar em conta como uma métrica é utilizada em relação a outras para se obter efetividade são levantadas três armadilhas: o uso de métricas redundantes como se não fossem redundantes; a comparação de métricas com um componente em comum sem considerar a correlação entre elas; e falhar em perceber que certa métrica não é de interesse primário, mas é necessária para ajustar outras métricas uma vez que existe covariança entre elas, sendo este tipo de variável chamada de fator de exposição.

Esses tipos de variáveis são utilizados em Análises Multivariadas como a Análise de Covariança (ANCOVA) ou regressão múltipla para ajustar os efeitos de exposição e mostrar o verdadeiro efeito dos valores restantes. Por exemplo, em um caso em que hajam duas métricas ‘A’ e ‘B’ que, simultaneamente não se afetam e estejam relacionadas ao aumento de uma métrica “C”, não podemos verificar a correlação entre A e C sem considerar B, pois isso retornaria uma probabilidade de correlação enviesada, pois A e B covariam sobre C, assim considerar apenas A afetaria a precisão do resultado.

Sobre a Análise/Inferência Estatística de dados ‘*limpos’* e cujas escalas métricas estejam bem definidas há três tarefas principais: Descrição, Comparação e Predição. Os aspectos envolvidos nessas atividades de análise variam se os dados são dinâmicos, envolvendo o aspecto temporal ou estáticos.

Ainda sobre a análise estatística, o autor discute em separado os dados a análise de dados dinâmicos (ou temporais) e estáticos (que não variam temporalmente). Um fator implícito aos dados dinâmicos, que os difere dos dados estáticos, e deve a ser analisado é a tendência temporal, que pode ser monotônica, cíclica de forma aperiódica e/ou de mudança sazonal. Outro ponto é que a análise dos dados pode ser seriamente restringida caso a quantidade de dados seja pequena.

A análise estatística é fundamentada em modelos do fenômeno latente que cria os dados. Estes modelos, frequentemente imperfeitos se baseiam em axiomas fazendo poucas ou várias suposições. Muitas suposições implicam em

modelos necessariamente paramétricos que permitem fortes deduções sobre fenômenos específicos. Poucas suposições possibilitam o uso de modelos não-paramétricos que têm como vantagem ser de aplicação genérica e desvantagem permitir inferir muito pouco sobre os dados.

Neste tópico indica-se que um valor de probabilidade de 0.05 quer dizer que 5% dos valores extremos – 2,5% em cada lado se a hipótese nula for uma igualdade ou 5% em um único lado se for uma desigualdade – são considerados outliers. Para aumentar a precisão estatística é indicado aumentar a quantidade de testes e considerar um valor de probabilidade menor que 0.05. Quando se trata de testes múltiplos indica-se o procedimento de Bonferroni para a diminuição de outliers, que implicam no erro do tipo I, entre os testes. No entanto, existem trabalhos recentes que afirmam tais procedimentos podem não ser interessantes para a análise dos dados.

Análise da Medição de Dados (páginas 164-178), inicia-se com a análise descritiva, tratando os conceitos de: medidas de tendência central, medidas de dispersão, medidas de associação, tratamentos especiais para as escalas nominais e ordinais.

As medidas de tendência central são a moda, mediana e média, cada uma adequada a uma situação específica. Para distribuições gaussianas se tem preferência pela média. Em distribuições não normais se opta pela mediana. A moda é mais utilizada na escala nominal.

Ao escolher uma medida de dispersão deve-se ter definida a medida de tendência central, sendo também interessante conhecer o intervalo da métrica. Quando se adota a média aritmética é natural utilizar o desvio padrão – raiz quadrada da variância – ou o coeficiente de variação – desvio padrão dividido pela média - como índices de dispersão. No caso da mediana, opta-se pelo intervalo semi-interquartil, que é definido pela diferença entre o 3º quartil (75% da probabilidade) e o 1º quartil (25% da probabilidade).

As medidas de associação entre duas métricas são quantificadas por coeficientes de correlação. Existem várias formas de calcular o coeficiente de correlação, a escolha dentre elas é dependente principalmente da escala de medição e do tamanho das amostras. Seja qual for o cálculo mais adequado, o resultado sempre estará compreendido no intervalo [-1, 1], sendo que quanto

mais próximo de zero menor é a correlação entre as medidas e o sinal indica se a relação é direta ou inversamente proporcional.

Os dados da escala nominal costumam ser descritos pela proporção do total de cada categoria. Quanto aos dados da escala ordinal, além de poderem ser descritos como dados da escala nominal, também podem ser adequadamente descritos centralmente pela mediana e correlacionados a partir de modelos estatísticos não paramétricos. As principais formas de cálculo são o coeficiente ρ de Spearman, o τ de Kendall e a estatística κ.

Quanto aos dados dinâmicos, a primeira opção de descrição se dá através de gráficos de decomposição de série temporal, que, ao detalhar em separado os dados observados e sua tendência temporal implícita, permite que a história do processo observado seja enxergada com tal clareza que é possível identificar padrões que apontem a suposições prováveis.

Um ponto a se destacar é que, para uma análise dinâmica é preferível utilizar métricas da escala razão ou intervalar, pois é predominante a suposição de que neste tipo de análise não há erros, e se há são mínimos, pois, caso sejam considerados o nível de significância e o poder da estatística, o padrão temporal será enviesado e impreciso. Ao realizar a medição nestas escalas é possível garantir maior precisão e confiabilidade as métricas, tornando a sua análise mais efetiva.

Outra questão que deve ser considerada é se as medições das séries temporais são realmente equivalentes. Um ótimo exemplo dado por Rosemberg é que, ao considerar distribuições de dados amostrais conforme uma série temporal em meses deve-se ter em mente que os meses não possuem quantidades iguais de dias, sendo que a diferença é agravada ao considerar que a quantidade de dias úteis varia ainda mais. Indo além, a quantidade de horas válida para os dados coletados possui variação ainda maior. Toda essa variação acumulada, se negligenciada, causa um viés de inconsistência e incerteza a respeito dos resultados e diminuem as suposições que podem ser levantadas.

A tarefa de comparação se baseia no teste de hipóteses, sendo a hipótese nula (H_0) uma proposição significando que determinada métrica possui um determinado intervalo, seja este intervalo de valores reais ou ideais, enquanto

que a hipótese alternativa (H_a) é, no mínimo, uma proposição que pertença ao conjunto complementar à H_0, ou seja, $H_a \in H_0^C$.

A comparação indica precisamente qual das hipóteses tem maior probabilidade de ser verdadeira, mas esta análise sempre estará sujeita a dois tipos de erros que devem ser considerados:

Erro do Tipo I (α): rejeitar aH_0 incorretamente quando ela deveria ser aceita e admitir incorretamente que a diferença é real;

Erro do Tipo II (β): não rejeitar a H_0 quando ela deveria ser rejeitada e admitir que a diferença não é real quando ela de fato é.

A probabilidade do erro α é o nível de significância do teste, sendo a probabilidade de encontrar outliers na distribuição e tem como valor aceitável 0.05. Destaca-se o fato de o valor de probabilidade (valor-p) ser diferente da probabilidade α, sendo definido que o valor-p é a probabilidade mínima de erro ao se deduzir que existe significância estatística, portanto para um resultado estatisticamente significativo, p-valor < α.

A probabilidade do erro β é a probabilidade de que os valores amostrais encontrados sejam em sua maioria outileirs e, portanto, está sendo considerada uma distribuição imprecisa. 1 - β é o poder da estatística, é a probabilidade de que o erro do tipo II seja reconhecido quando ocorrer, ou seja, que a H_0 seja aceita quando deveria ser rejeitada. O tamanho da amostra é o fator mais importante quando se trata de poder estatístico, sendo necessário calcular o tamanho adequado antes de coletar os dados.

Existem diversos testes estatísticos que podem ser utilizados para realizar comparações. A escolha do teste mais adequado depende de alguns fatores como a escala de medição, se a distribuição da população é normal ou não, se vai ser usado um valor hipotético na comparação, se as variáveis são dependentes ou independentes, pareadas ou não pareadas e a quantidade de amostras.

Quanto a tarefa de predição, os modelos estatísticos se fundamentam na Regressão Linear, podendo ser equações de predição linear, modelos lineares genéricos e método dos mínimos quadrados. O caso mais simples é aquele em que a equação de regressão se limita a uma reta:

$$Y = a + bX + erro$$

No caso de dados da escala nominal, a predição de resultados dicotômicos pode ser realizada através do método de predição diagnóstica, no qual existem quatro possíveis resultados: verdadeiro positivo (VP), falso positivo (FP), verdadeiro negativo (VN) e falso negativo (FN). Para diferenciar entre valores falsos e verdadeiros é comum o uso do *padrão de ouro,* um valor conhecido como verdadeiro no contexto do teste, contudo, frequentemente este valor não é conhecido.

Importantes medições deste tipo de predição são a especificidade e a sensibilidade. Ambas medições são probabilidades inversamente proporcionais. A sensibilidade é a probabilidade de detecção de verdadeiros positivos. Já a especificidade é a probabilidade de detecção de verdadeiros negativos. Sendo as equações:

$$Sensibilidade = \frac{VP}{VP + FP} \wedge Especificidade = \frac{VN}{VN + FN}$$

Uma forma típica de representação gráfica dessas medições é a curva de Característica de Operação do Receptor (ROC), ela é representada por um plano cartesiano cujos eixos se encontram no intervalo [0, 1], ou seja, representam probabilidades. O eixo vertical representa a sensibilidade e o eixo horizontal representa 1-especificidade. Desta forma, se um teste for representado pela curva ROC e a sua imagem for a de um segmento que começa na origem e termina nas coordenadas (1, 1), isso significa que há 50% de chance de previsão de VP e 50% de chance previsão de VN, assim este teste na verdade não é capaz de prever nada.

Essa relação entre especificidade e sensibilidade pode ser mensurada pela Área Sob a Curva (AUC). Retomando o exemplo do segmento novamente, tem-se que a área sob a curva gera forma um triângulo de área 0.5. O melhor caso representável para uma curva ROC é aquela em que a área sob a curva cobre totalmente o plano, no entanto, o que será encontrado no mundo real é que os testes terão um UAC < 1 e representarão uma relação inversa entre a especificidade e a sensibilidade.

Do ponto de vista dos dados dinâmicos, a tarefa de predição tem como fim a previsão de uma ou mais observações futuras a partir dos dados já obtidos. Diferente da tarefa de descrição, nesta é possível inferir com poucos dados através de técnicas como o *preditor ingênuo*, no qual se assume que o futuro valor de uma variável será o mesmo que o atual. Esta técnica não permite fortes suposições, mas possibilita uma previsão aproximada na maioria dos casos, além de servir como métrica de base para avaliar a acurácia de técnicas mais elaboradas de previsão, principalmente quando a série temporal é cíclica.

Obviamente existem muitas formas de quantificar a precisão de uma previsão, cada uma com suas vantagens e desvantagens, que neste momento não precisam ser detalhadas individualmente. Sobre a escolha do método de previsão, é comum que se pondere entre a acurácia e a complexidade da computação.

Outro ponto interessante sobre o tema é se a métrica temporal é adiantada, atrasada ou coincidente com as medições em tempo real. Se por um lado as mais desejadas são as adiantadas, por outro são também são as menos encontradas.

Quando se trata de métricas dinâmicas existe uma relação entre tempo real de medição e o tempo em que as métricas são obtidas. O autor classifica as métricas dinâmicas de acordo com essa relação, sendo as três classificações: leading indicator (tempo de obtenção da métrica adiantado em relação ao tempo de medição); lagging indicator (tempo de obtenção da métrica atrasado em relação ao tempo de medição); e coinciding indicator (tempo de obtenção da métrica coincidente em relação ao tempo de medição).

Um uso importante das métricas temporais é no ramo de controle de processo estatístico, que consiste em determinar se as variáveis observadas em determinado processo variam apenas quando se deseja e se, quando sem interação, se mantém estáveis, mesmo frente a variações aleatórias, avaliando se o processo é efetivo.

Uma forma de quantificar e monitorar se um processo é estável é através do gráfico de controle. Este gráfico se assemelha ao gráfico de tendência, com diferença de que possui duas linhas horizontais tracejadas,

uma acima e outra abaixo, com iguais distâncias de um de uma linha de controle que expressa a tendência esperada, sendo as linhas tracejadas limites que não devem ser ultrapassados pela variável observada para que o processo seja considerado efetivo e estável.

Uma variação interessante do gráfico de controle é o gráfico de soma acumulada, que ao acumular o desvio da tendência esperada, sendo mais sensitivo a variação. Quando a variação não é aleatória os desvios da tendência não se equilibram passam a ter um viés, indicando que há um fator influenciando na estabilidade do processo.

Agora, por fim, é necessário descrever as principais causas dos problemas envolvendo o tópico da **Qualidade de Dados** (páginas 178-182), pois dados ruins invalidam quaisquer conclusões que possam ser inferidos a partir deles.

As principais fontes desses problemas são: problemas organizacionais, falta de definições precisas, falta de validação de dados, dados perdidos e amostragem enviesada.

Os problemas organizacionais se referem ao fato de que com frequência não é a pessoa que toma decisões que armazena os dados, e sim pessoas que simplesmente não entendem o significado dos dados e como eles direcionam a tomada de decisões, por isso, para alcançar as metas do trabalho acabam produzindo dados sem significado verdadeiro.

A falta de definições concretas quanto as métricas a serem medidas podem ser resolvidas se antes da medição a métrica for detalhada e precisamente conceituada, assim como outros aspectos envolvidos com a mesma, como aspectos temporais, responsabilidades, dentre outros.

A falta de validação dos dados deve ser resolvida através de auditoria. Claro que é bom que se criem condições nas quais se previne que dados inválidos sejam armazenados, no entanto, a prevenção não garante que ocorram situações em que dados inválidos possam infectar nosso conjunto de dados, dessa forma, a auditoria deve ser realizada e boa parte do problema pode ser resolvido automaticamente a partir de algoritmos de validação/limpeza dos dados.

Sobre os dados perdidos, é aconselhável estar atento aos valores de registros omissos, principalmente quando os dados forem utilizados por mais de uma pessoa, quando se utiliza novos algoritmos ou novas tecnologias no uso dos dados, pois utilizar algo novo abre a possibilidade de que os valores reais não sejam considerados omissos e se acabe produzindo resultados estatísticos enviesados. Sobre este assunto também cabe ponderar o quanto os dados omissos podem afetar os resultados estatísticos e buscar as formas mais adequadas para lidar com isso.

A ocorrência de amostragem enviesada acontece quando o processo de medição afere somente uma parte da população. Isso compromete totalmente os resultados estatísticos, mas mesmo sendo um erro grave para análise frequentemente é algo de difícil identificação, pois geralmente tem origem em momentos antes de os dados serem colhidos.

Em específico, a amostragem enviesada pode ser mais facilmente percebida ao se analisar a documentação buscando por valores de dados que são raramente ou nunca mensurados. Além disso, uma forma objetiva de resolver o problema é identificar especificamente qual a parcela da população que seus dados representam. No geral, defeitos envolvendo a qualidade de dados são comuns e, ao serem identificados devem ser analisados em sua individualidade.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …): Engenharia de Software; Métodos Estatísticos e Medição; Estatística (Descritiva e Indutiva).

Contexto de validação (quais estudos de caso foram aplicados, …): Neste caso o autor validou sou trabalho unicamente através da citação de referências;

3.2. FICHAMENTO 2 [Balera e Santiago Júnior 2017]

Referência:

[Balera e Santiago Júnior 2017] BALERA, J. M.; SANTIAGO JÚNIOR, V. A. 2017. An algorithm for combinatorial interaction testing: definitions and rigorous evaluations. Journal of Software Engineering Research and Development 5, 1 (28 Dec 2017), 41. https://doi.org/10.1186/s40411-017-0043-z.

Classificação (nova abordagem, revisão, survey):

Nova abordagem/algoritmo.

Motivação do artigo:

A motivação deste artigo é o teste de software através de casos de teste gerados via *design* combinatório, pois este *design* tem chamado a atenção da comunidade desta área pela sua capacidade de gerar conjuntos de teste com menor custo e maior eficiência.

Objetivo do artigo:

A partir da motivação acima, foi desenvolvido o algoritmo guloso e irrestrito TTR (T-Tuple Reallocation). O objetivo geral foi contribuir com o campo de TIC (Teste de Interação Combinatória), verificando a capacidade do TTR frente a outros algoritmos/ferramentas de teste de software via design combinatório, que já eram utilizados por praticantes e pesquisadores da área.

Os objetivos principais foram: comparar 2 versões do TTR através de experimento controlado para verificar qual das duas é mais eficiente em termos de quantidade de casos de teste gerados (custo) e tempo de execução (eficiência); comparar a versão do TTR de melhor desempenho com outras soluções de interação combinatória através de experimento controlado, sendo IPOG-F e jenny em termos de custo/tempo de execução, e PICT, IPO-TConfig e ACTS em termos de custo.

A análise dos resultados se deu para cada um dos graus de interação entre os fatores (*strength*), sendo os graus baixo (t = 2), médio ($3 \leq t \leq 4$), alto ($5 \leq t \leq 6$) e todos ($2 \leq t \leq 6$), onde t é o grau de *strength.*

Rápida descrição sobre as metodologias/técnicas/métodos relacionados ao artigo:

Houve 2 experimentos controlados e, por uma questão de simplicidade, trataremos como se fosse um único experimento. Embora o segundo experimento seja decorrente dos resultados do primeiro, ambos os experimentos seguiram um mesmo padrão.

Cada algoritmo/ferramenta foi submetida ao teste de uma amostra de 80 instâncias, uma de cada vez, sendo que cada instância possui um *strength* t, tal que $2 \leq t \leq 6$. Além disso, cada instância possui uma quantidade de atributos (parâmetros), que aqui chamaremos de *p*, tal que $4 \leq p \leq 16$, sendo que este intervalo para *p* não é diretamente mencionado no artigo, mas pode ser conferido nas tabelas de amostras em Balera e Santiago Júnior (2017).

Deve se destacar que, para abordar o não-determinismo de cada solução, houve o cuidado de se avaliar cada uma das instâncias cinco vezes, cada uma delas com os parâmetros ordenados de maneiras diferentes. Assim, cada instância foi avaliada a partir da média aritmética das suas 5 avaliações.

Os algoritmos/ferramentas avaliados foram TTR (versões 1.1 e 1.2), IPOG-F, jenny, IPO-TConfig, PICT, e ACTS. Para ser breve, foi feita a descrição apenas das hipóteses nulas ($H_{0.x}$), deixando claro que as hipóteses alternativas ($H_{1.x}$) para cada caso correspondem à negação da respectiva $H_{0.x}$. Tendo como *custo* a quantidade de casos de teste gerada e como *eficiência* o tempo de execução, seguem:

$H_{0.1}$ - Não há diferença quanto ao custo-eficiência entre TTR 1.1 e TTR 1.2;

$H_{0.2}$ - Não há diferença quanto ao custo-eficiência entre TTR 1.2 (em Java) e IPOG-F (em Java);

$H_{0.3}$ - Não há diferença quanto ao custo-eficiência entre TTR 1.2 (em C) e jenny (em C);

$H_{0.4}$ - Não há diferença quanto ao custo entre TTR 1.2 (em Java) e PICT;

$H_{0.5}$ - Não há diferença quanto ao custo entre TTR 1.2 (em Java) e IPO-TConfig;

$H_{0.6}$ - Não há diferença quanto ao custo entre TTR 1.2 (em Java) e ACTS.

Nos casos que se avaliou as soluções em termos de custo-eficiência foi interessante transformar os dados de um contexto multiobjetivo para um mono-objetivo. Para cumprir esta tarefa, tomou-se o plano cartesiano, tendo como

eixo das abcissas o custo e o das ordenadas o tempo de execução. Tomando-se ainda como ponto ideal o 0 (0, 0), pois o que se busca é o menor custo e tempo possíveis - apesar de essas variáveis nunca chegarem a zero - e o ponto $P_i(cA_i, tAi)$, onde cA_i é o custo médio para uma instância *i* e tA_i é o tempo médio de execução. Assim, o módulo do vetor $\overrightarrow{0P_i}$ representa a distância euclidiana, sendo:

$$\left|\overrightarrow{0P_i}\right| = \sqrt{cA_i^2 + tA_i^2} = d_i$$

Tal que $d_i \in D_A$, sendo D_A o conjunto de distâncias euclidianas de um algoritmo A qualquer. Lembrando que esta é apenas uma das formas de se calcular a distância, neste caso, do ponto de vista vetorial. Assim, D_A é um conjunto de métricas unidimensionais, tornando-se mais interessante para a análise estatística do que a relação custo-eficiência.

Agora, trabalhando com distribuições unidimensionais, sendo a distância euclidiana para os algoritmos envolvidos nas hipóteses nulas $H_{0.1}, H_{0.2} e H_{0.3}$ e o custo para os algoritmos/ferramentas envolvidos nas hipóteses $H_{0.4}, H_{0.5} e H_{0.6}$, deseja-se realizar o teste estatístico mais apropriado para o tipo de distribuição dos dados.

Para tanto, primeiro verifica-se a assimetria/viés da distribuição. Foram utilizados 3 métodos estatísticos para verificar isto: o teste de Shapiro-Wilk com um nível de significância α = 0.05; checando a obliquidade *O* da distribuição de frequência (considerando que, $-0.1 \leq O \leq 0.1$ leva a uma distribuição normal); e pela verificação visual de histogramas e gráficos Quantil-Quantil.

Consequentemente, se a distribuição for gaussiana o teste mais adequado seria teste t de Student e no caso de distribuições não normais o mais adequado seria o teste de Wilcoxon. De fato, as verificações da normalidade das distribuições não retornaram distribuições normais, muito devido à constante presença de outliers. Por conta das características dos dados destas distribuições foi utilizado o teste de Wilcoxon para amostras pareadas com distribuições assintóticas de dois lados.

Deste modo, para rejeitar uma hipótese nula $H_{0.x}$, deve-se verificar, através do teste de Wilcoxon mencionado acima, que a proposição

p-value < 0.05 ^ |z-score| > 1.96

é verdadeira. Destaca-se que os testes estatísticos foram realizados utilizando a linguagem R.

Inicialmente, os resultados nos levam a rejeitar a hipótese nula $H_{0.1}$, pois de fato o TTR 1.2 apresenta um resultado em relação ao custo-eficiência melhor que o TTR 1.1, sendo por isso escolhido nos próximos experimentos.

Os resultados dos próximos experimentos levam à suposição, a grosso modo, de que o IPOG-F possui uma ótima relação custo-eficiência frente aos demais algoritmos/ferramentas em todos os níveis de strength e por isso é muito interessante compará-lo com TTR 1.2. Já o TTR 1.2 possui uma grande melhoria na relação custo-eficiência conforme o nível de strength aumenta, tendo resultados piores que os outros quando a t=2, menos quando comparado com o IPO-TConfig, tendo diferença insignificante. No caso de $5 \leq t \leq 6$, o TTR 1.2 tem resultados significativamente melhores que os demais, menos o IPOG-F que não apresenta resultados com diferença significativa em relação ao TTR 1.2.

É muito provável que o TTR 1.2 venha a superar o IPOG-F com o aumento dos valores de strength para além dos analisados. Contudo, sem experimentos com t>6 não podemos afirmar isso. De todo modo, o TTR 1.2 se mostrou um algoritmo guloso de design combinatório, muito interessante para a geração de casos de teste, principalmente quando se deseja um alto grau de interação entre os atributos da instância a ser testada.

Este trabalho define bem a validação das métricas envolvidas em seus experimentos, em termos de validade de conclusão, validade interna, validade de construto e validade externa.

A validade de conclusão se trata de verificar se o tratamento usado nos experimentos realmente tem relação com os resultados observados. Uma forma de buscar por validade de conclusão é garantir uma alta confiabilidade das medições. Como as medições foram obtidas automaticamente, através de algoritmos/ferramentas e foram adotados os métodos estatísticos mais adequados para analisar a normalidade dos dados e inferir sobre a se há

diferença estatística sobre a validade de conclusão, considera-se este um estudo de alta validade de conclusão.

A validade interna tem o objetivo de analisar se o tratamento experimental realmente foi a causa dos resultados, ou se existe alguma covariável relacionada aos resultados e que não foi tratada. Como os participantes destes experimentos são amostras aleatórias - compostas por parâmetros, valores e graus de interação - e não ocorreram eventos inesperados durante a coleta das medições, afirma-se que os experimentos possuem alta validade interna.

A validade de construto também é alta, uma vez que, por definição, esta visa garantir que o tratamento reflita a construção da causa e o resultado a construção do efeito, pois os algoritmos/ferramentas são a causa que atuou sobre as instâncias e originaram os casos de teste como resultado parcial. Os testes estatísticos fornecem a base para que se possa realizar suposições acerca da performance das soluções avaliadas, sendo estas suposições o resultado final, a partir do qual se pode tomar decisões sobre qual das soluções é mais adequada para determinada situação.

Já a validade externa, tendo em vista que se trabalhou com amostras, busca determinar o quanto que os resultados do estudo podem ser generalizados para outras amostras da mesma população, sob a perspectiva temporal. No caso deste experimento, uma ameaça à validade externa a ser observada é o fato de que quando se considera o intervalo de strength, a quantidade de parâmetros e quantos valores cada parâmetro pode assumir, a população de instâncias possíveis tem infinitos elementos. Portanto, não se espera que os resultados obtidos neste trabalho se apliquem a todas as amostras possíveis da população.

No entanto, ao se considerar as possíveis instâncias que podem ser encontradas no cenário da prática de teste de software, considera-se que a configuração de strengths, parâmetros e valores é significativa, assim como o tamanho da amostra. Além disso, as configurações foram escolhidas de forma aleatória dentro de intervalos determinados e ainda o não-determinismo foi preservado ao se executar a mesma instância 5 vezes, com a entrada dos

atributos em ordens diferentes. Desta forma, através da aleatoriedade, evitou-se que ocorresse viés durante a amostragem.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …):

Teste de Software, Teste de Interação Combinatória.

Contexto de validação (quais estudos de caso foram aplicados, …):

Para a validação experimental de cada algoritmo/ferramenta, foi utilizada uma amostra de 80 instâncias, cada uma possuindo um strength t, tal que $2 \leq t \leq 6$, uma quantidade de atributos p (parâmetros) tal que $4 \leq p \leq 16$ com valores v variando de $2 \leq v \leq 12$. Para lidar com o não-determinismo, cada uma das instâncias foi rodada com os parâmetros ordenados em cinco formas diferentes por cada um dos algoritmos/ferramentas integrantes do experimento. Para mensurar o custo de cada algoritmo/solução em relação a cada uma das instâncias, adotou-se a média dos resultados obtidos em cada uma das cinco ordens diferentes de parâmetros.

3.3. FICHAMENTO 3 [Burke et al. 2013]

Referência:

[Burke et al. 2013] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society, 64(12):1695-1724, Dec 2013.

Classificação (nova abordagem, revisão, survey):

Survey, Revisão de Literatura.

Motivação do artigo:

O conceito por trás das hiper-heurísticas é relativamente novo, sendo que a sua ideia inicial aparece pela primeira vez na literatura em 1963, nos estudos de Fisher e Thompson e Crowston et al. Apenas em 2010, no artigo de Burke et al, a definição de hiper-heurísticas teve suas subclassificações unificadas a partir das classificações anteriormente produzidas neste campo de estudo.

Como se trata de um campo de estudo novo e complexo, a necessidade de esclarecimento sobre tudo o que vem sendo produzido, as vezes com diferentes termos sendo utilizados em diferentes trabalhos com o mesmo significado implícito, é um fator motivador para a produção deste trabalho que, por meio da análise e discussão da literatura, acaba por, de certa forma, organizar e categorizar um conjunto de obras selecionadas que transmitem as tendências deste campo.

Objetivo do artigo:

Este artigo tem como objetivo geral a análise e discussão crítica da literatura científica sobre as hiper-heurísticas produzida até o ano de 2012. Tem como objetivos específicos: descrever a origem e raízes intelectuais das hiper-heurísticas; relatar detalhadamente os principais tipos de abordagens; uma visão geral de algumas áreas relacionadas; e mencionar sobre o desafio de se construir frameworks de desenvolvimento de hiper-heurísticas que se apliquem de forma genérica a vários domínios e que permitam aos praticantes utilizá-las com a menor quantidade de conhecimento específico sobre o problema.

Rápida descrição sobre as metodologias/técnicas/métodos relacionados ao artigo:

Os trechos deste artigo que detalham as metodologias hiper-heurísticas podem ser classificadas nos tópicos: classificação das abordagens hiper-heurísticas; metodologias de seleção de heurísticas ou hiper-heurísticas de seleção; metodologias de geração de heurísticas ou hiper-heurísticas de geração; e áreas relacionadas.

Uma hiper-heurística é uma heurística de alto nível que gerencia um conjunto de heurísticas de baixo nível, que usa informações limitadas de determinado problema para buscar métodos de solução eficientes (Chakhlevitch e Cowling, 2008, apud this). Outra forma de definição é proposta por Burke et al (2010, apud this), sendo uma hiper-heurística um método de busca ou mecanismo de aprendizado para selecionar ou gerar heurísticas que solucionem problemas de busca computacional difíceis.

Assim sendo, uma hiper-heurística pode ser classificada de acordo com as heurísticas que fazem parte do espaço de busca, que podem ser ou heurísticas construtivas ou perturbativas. O método para se chegar a nova solução pode ser por seleção ou geração. Além disso, ainda é possível classificar uma hiper-heurística de acordo com a existência ou não de um mecanismo de aprendizagem e, caso exista, se o aprendizado é online ou offline. Ainda são possíveis hiper-heurísticas mistas.

A grosso modo, pode-se dizer que as heurísticas construtivas constroem soluções incrementalmente a partir de uma solução vazia ou, pelo menos, uma solução parcial. Em contraste, uma heurística perturbativa opera através da melhoria de uma solução candidata através da alteração dos seus componentes. Além disso, uma hiper-heurística que seleciona heurísticas perturbativas pode ser dividida em dois componentes separados: método de seleção heurística e método de aceitação de movimento.

Uma hiper-heurística de seleção opera selecionando heurísticas completas, enquanto que uma hiper-heurística de geração opera sobre heurísticas construídas a partir de componentes de heurísticas. Ambas tem como saída a solução encontrada ao fim da *run*, sendo que a hiper-heurística de geração ainda retorna a nova heurística que foi utilizada para obter a solução.

Sobre as formas de aprendizagem, o aprendizado online ocorre enquanto o algoritmo estiver solucionando uma instância do problema, ou seja, a aprendizado se dá no decorrer da execução a conforme os dados vão sendo gerados. Já na aprendizagem offline, o conhecimento é reunido na forma de regras ou programas, sendo que os dados de entrada para esta forma de aprendizagem são obtidos conforme as instâncias do problema vão sendo resolvidas.

Quanto as *metodologias de seleção de heurísticas*, este trabalho as divide em construtivas ou perturbativas, de acordo com as heurísticas de baixo nível utilizadas. Sobre as metodologias de seleção de heurísticas construtivas, levando-se em conta que a performance de uma hiper-heurística é dependente do domínio/problema em que é aplicada, seguem algumas abordagens, com seus respectivos domínios de aplicação, selecionadas de acordo com a sua performance:

Calendarização	Algoritmo de Busca em Vizinhança Variável (VNS) como heurística de seleção.

Educacional	Heurísticas construtivas baseadas em uma função de decisão ponderada e heurísticas básicas do problema de coloração de grafos.
	Sistema difuso como algoritmo de seleção. Heurísticas construtivas do problema de coloração de grafos.
	Tabu-search e Algoritmo de Busca Local Iterada (ILS) como heurística de seleção. Heurísticas construtivas do problema de coloração de grafos e heurísticas de ordenação aleatória.
	Algoritmos Evolucionários como heurística de seleção. Heurísticas construtivas do problema de coloração de grafos sob um ponto de vista combinatório.
	Messy Genetic Algorithm como heurística de seleção. Heurísticas construtivas do problema de coloração de grafos.
	Raciocínio Baseado em Casos (CBR) como heurística de seleção. Heurísticas construtivas do problema de coloração de grafos e procedimento hill-climbing.
Programação da Produção	Algoritmo Genético Padrão (GA) como heurística de seleção. Regras de despacho como heurísticas construtivas e blocos de decisão (conjuntos de regras de despacho tratadas como uma unidade).
	Heurística de seleção: Busca Dispersa combinada com um conjunto de referência (matriz de frequência de combinação de regras). Regras de prioridade como heurísticas construtivas.
Problema do Empacotamento (unidimensional)	Accuracy-Based Learning Classifier System para aprender um conjunto de regras que associem o estado atual do problema com diferentes heurísticas construtivas.
	Messy Genetic Algorithm como heurística de seleção combinada com aprendizado online formando associações entre os estados do problema e as heurísticas construtivas mais adequadas.
Problema de Corte e Empacotamento (bidimensional)	Messy Genetic Algorithm como heurística de seleção. Heurísticas para selecionar figuras/objetos e heurísticas para colocar figuras em objetos.
	Heurística de seleção baseada em Algoritmo Genético combinado com aprendizado online. As heurísticas de baixo nível foram categorizadas de acordo com a sua funcionalidade: gulosa, ordenação e rotação.
Satisfação de	Messy Genetic Algorithm como heurística de seleção.

Restrições	Abordagem para geração de hiper-heurísticas em 2 etapas: No estágio de treinamento a performance das diferentes heurísticas em diferentes cenários é acumulada. No segundo estágio a informação é usada para gerar uma hiper-heurística de seleção específica para o problema.
Problema de Roteamento de Veículos	Heurística de seleção baseada em Hill-Climbing. Utilizou como heurísticas de baixo nível heurísticas construtivas e perturbativas.
	Heurística de seleção Evolucionária. Utiliza como heurísticas de baixo nível heurísticas construtivas, perturbativas e de ruído.

Sobre as metodologias de seleção de heurísticas perturbativas, seguem algumas abordagens e seus respectivos domínios/problemas de aplicação:

Unit Commitment Problem	Heurística baseada em *Random Permutation Gradient* com *Reinforcement Learning* embutido e um *método guloso de aprendizado* como heurística de seleção.
Problema de Logística	*Reinforcement Learning* (RL) como heurística de seleção. Método de aceitação de movimento: *All Moves* (AM).
Problemas de Calendarização e de Alocação de Espaço	*Heurística de seleção: Reinforcement Learning* combinada com *Tabu-search*. Abordagem competitiva em problemas mono e multiobjetivo.
	Simple Random como heurística de seleção. *Simulated Annealing* como método de aceitação de movimento.
Problema de Alocação de Pessoal	Heurísticas baseadas métodos *Simple Random, Greedy* e *Peckish* como heurísticas de seleção. Método de aceitação de movimento: *All Moves, Only Improving,* métodos baseados em *Tabu-Search*. Foram usadas 95 heurísticas de baixo nível não especificadas diretamente.
	Método *Greedy* como heurística de seleção. Método de aceitação de movimento baseado em *Tabu-Search* com redução linear do conjunto de heurísticas de baixo nível.
DTLZ	Heurística de seleção: *Reinforcement Learning* modelado como uma cadeia de Markov com uma estratégia evolucionária embutida. Método de aceitação de movimento não mencionado. Além de uma heurística de baixo nível inefetiva, foram utilizadas as heurísticas perturbativas: mutação, replicação e transposição.
Escalonamento de	Heurística de seleção: *Simple Random*. Métodos de aceitação de movimento não-

Posições de Componentes Eletrônicos em um Circuito Impresso	determinísticos baseados em *Monte Carlo,* sendo eles: *Linear* (LMC), *Exponencial* (EMC) e uma formulação baseada no tempo de computação na contagem iterações consecutivas sem melhorias (EMCQ). Obteve melhores resultados comparado a hiper-heurísticas que se baseavam em métodos de aceitação determinísticos.
Problema de Atribuição de Canal	*Simple Random* como heurística de seleção. *Great Deluge* como método de aceitação de movimento.
	Simple Random como heurística de seleção. *Record-to-Record Travel* como método de aceitação de movimento.
Alocação em Prateleira	*Simple Random* como heurística de seleção. *Simulated Annealing* com *Metropolis criterion* como método de aceitação de movimento.
Programação de Entrega de Concreto	*Simple Random* como heurística de seleção. *Adaptive Iteration Limited List-based Threshold Acceptance (AILLA)* e *Late Acceptance* como métodos de aceitação de movimento.
Problema de Empacotamento	Heurística de seleção: *Reinforcement Learning* combinado com *Tabu-Search*. Método de aceitação de movimento: *Simulated Annealing* combinado com *Reheating*. Uso de função geométrica para resfriamento, que lida com a ocorrência de diferentes tamanhos de vizinhança ocasionada por um '*pool*' de heurísticas de baixo nível.
Problemas de Roteamento de Veículos	*Busca por grande vizinhança* como heurística de seleção, combinado com técnicas de *machine learning online*. *Simulated Annealing* baseado em uma taxa de resfriamento linear como método de aceitação de movimento. Além das heurísticas de baixo nível convencionais, utiliza heurísticas de inserção e remoção.
Traveling Tournament Problem	Heurística de seleção: *Automato de aprendizado online*. Método de aceitação de movimento: *Iteration Limited Threshold Accepting* (ILTA).
Agendamento de Assistência Domiciliar	*Tabu-based learning* usando um Índice de Qualidade (QI). Método de aceitação de movimento: variante adaptativa do *Iteration Limited Threshold Accepting* (ILTA).
Predição de Sequencias de DNA	*Hiper-Heurística* de Seleção baseada em método *Roulette Wheel.* Método de aceitação de movimento: *Simulated Annealing*. Foram utilizadas 6 heurísticas de baixo nível, dentre elas as operações de inserção, deleção, swap e shift.
Otimização de Funções de	*Choice Function* como heurística de seleção. *Improving and Equal* como método de

Referência	aceitação de movimento.
Problema de Agendamento de Exames	*Choice Function* como heurística de seleção. *Simulated Annealing* como método de aceitação de movimento.
	Simple Random como heurística de seleção. *Great Deluge* como método de aceitação de movimento.
Calendarização de Cursos	Hiper-heurísticas usando memória de curto prazo produzem melhores resultados do que algoritmos sem memória ou com memória infinita.
Problema/ Domínio não Mencionado	*Choice Function* como heurística de seleção. Como método de aceitação de movimento: *All Moves* (AM).
	Quando se trata de uma combinação entre aprendizado online e o método de aceitação *Late Acceptance,* a hiper-heurística com melhor performance encontrada é o *Simple Random.*

A maioria das hiper-heurísticas mencionadas acima conduzem a busca de ponto único, processando uma única solução em cada *run.* Hiper-heurísticas que realizam busca em múltiplos pontos, processando múltiplas soluções, tem obtido resultados promissores. Dentre as hiper-heurísticas que realizam este tipo de busca são mencionadas: hiper-heurísticas baseadas no algoritmo de colônia de formigas, algoritmos genéticos, algoritmos de otimização numérica, abordagens evolucionárias, CMA-ES, método do enxame de partículas e método de busca cooperativa.

Os autores mencionam 2 frameworks interessantes: HyFlex (Hyper-heuristic Flexible) e Hyperion. O HyFlex suporta o desenvolvimento de hiper-heurísticas para diferentes problemas de otimização combinatória, provendo componentes de algoritmos específicos para cada problema. O Hyperion é um framework recursivo que suporta o desenvolvimento de meta-heurísticas, hiper-heurísticas de seleção e seus híbridos.

Sobre as *metodologias de geração de heurísticas,* é mencionado que a Programação Genética é a metodologia mais frequente na literatura para a geração automática de heurísticas de baixo nível. A principal forma de obtenção de componentes para a geração de novas heurísticas é a decomposição de heurísticas existentes.

Hiper-heurísticas de geração costumam gerar heurísticas muito específicas para a instância do problema na qual operam e com capacidade de generalização para outras instâncias diminuída em relação as heurísticas desenvolvidas por humanos, porém a literatura demonstra que heurísticas geradas automaticamente têm obtido um desempenho superior em relação as desenvolvidas por humanos. Além disso, apenas uma fração do tempo gasto por humanos no desenvolvimento de novas heurísticas é gasto por uma hiper-heurística de geração.

O uso de mecanismos de aprendizado offline tem trazido bons resultados quando se trata do aumento da capacidade de generalização e potencial de reutilização de heurísticas automaticamente geradas.

Sobre o tópico *áreas relacionadas,* os autores destacam algumas abordagens de aprendizado utilizadas para aprimorar as técnicas de busca. Sobre as abordagens offline são mencionadas: configuração dos parâmetros do algoritmo e meta-aprendizado para a seleção de algoritmos. Quanto as abordagens online: controle de parâmetro para algoritmos, algoritmos meméticos adaptativos, seleção de operador adaptativo, busca reativa, busca em vizinhança variável, portfólios de algoritmo.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …):

Hiper-Heurísticas, Computação Evolucionária, Pesquisa Operacional, Aprendizado de Máquina, Otimização Combinatória.

Contexto de validação (quais estudos de caso foram aplicados, …):

Sendo um artigo de revisão, a validação se deu a partir citação de outras obras.

3.4. FICHAMENTO 4 [Jia et al. 2015]

Referência:

[Jia et al. 2015] Y. Jia, M. B. Cohen, M. Harman, and J. Petke. Learning combinatorial interaction test generation strategies using hyperheuristic search. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, volume 1, pages 540-550, May 2015.

Classificação (nova abordagem, revisão, survey):

Nova abordagem.

Motivação do artigo:

Há mais de duas décadas vem se desenvolvendo soluções sob medida para instâncias do problema de Teste de Interação Combinatória (CIT). Escolher entre as diversas soluções levantadas na literatura para um problema em específico é uma tarefa inviável para os praticantes de Teste de Software. Com isso em mente, os autores propõem um algoritmo hiper-heurístico como solução genérica e praticável em diversas instâncias do problema de CIT.

Objetivo do artigo:

O objetivo deste trabalho foi examinar se o uso de Hiper-Heurísticas no campo de Otimização de Teste de Software é uma prática viável para gerar amostras de CIT. Para tanto foram realizados experimentos para analisar o desempenho de uma nova hiper-heurística frente a dados de referência encontrados na literatura sobre soluções meta-heurísticas feitas especificamente para instâncias específicas do problema de CIT.

Rápida descrição de metodologias/técnicas/métodos relacionados ao artigo:

A partir da formulação do CIT como um problema de busca hiper-heurístico foi introduzido um algoritmo de hiper-heurística de seleção baseado em simulated annealing, chamado HHSA, com Reinforcement Learning e com 6 operadores de navegação (NOs) como heurísticas perturbativas – mutação única, add/del, e multi-mutação, cada um com versões ‘padrão’ e ‘smart’. O HHSA foi submetido a 4 experimentos que visaram: I - mensurar o tamanho das suites de teste geradas; II - medir o tempo de execução; III - os efeitos de se utilizar NOs isolados; e IV - evidenciar se o aprendizado realmente ocorre.

No experimento I foram utilizados os seguintes dados de referência: 29 modelos sintéticos irrestritos de Garvin et al (2011), 20 modelos de mundo real irrestritos Segall et al (2011), e 6 modelos de mundo real restritos de Calvagna e Gargantini (2012).

Os dados de referência de Garvin et al (2011), que possuíam para cada instância os melhores resultados descritos na literatura e os resultados de uma

simulated annealing para problemas irrestritos, que foram comparados com os resultados da ferramenta CASA (Covering Arrays by Simulated Annealing) e o melhor resultado da HHSA em 5 runs. Dentre estes modelos haviam 14 com t=2 e 15 com t=3. O HHSA obteve resultados competitivos nas instâncias t=2, mas houve uma lacuna de desempenho nos modelos t=3.

Os modelos de Segall et al (2011) foram utilizados com t=2. Dentre os algoritmos para solucionar as instâncias encontram-se ACTS, Jenny, FoCus e PICT, que foram comparados ao HHSA. Já nos modelos de Calvagna e Gargantini (2012) os modelos foram testados em t=2 e t=3, sendo que as ferramentas utilizadas como solução no experimento foram CASA e T-tuples. Nos modelos de mundo real, quando se trata do tamanho dos suites de teste gerados, a HHSA teve um desempenho igual ou superior ao das demais soluções.

No experimento II, assim como no anterior, o HHSA foi utilizado em 3 configurações - baixa(L), média(M) e alta(H) – que implicam no consumo de recursos computacionais. Basicamente, as diferenças entre as configurações L e M são os valores atribuídos aos parâmetros da taxa de resfriamento e da função de passo de resfriamento, que são maiores em M. Já a diferença de M para H é que, em H, uma busca gulosa - com valores aumentados para os parâmetros: função de passo de resfriamento, temperatura inicial e número máximo de não melhorias permitidas - é realizada após a busca binária padrão, que é configurada da mesma forma que em M.

Pela média dos melhores resultados alcançados em cada um dos dados de referência, menos as instâncias de Calvagna e Gargantini (2012) em t=3 - em 2 destes modelos não se chegou a resultado algum em 3 dias de processamento na HHSA-H -, o tamanho das suites de teste na HHSA-L foram minimizados em cerca de 2% através da HHSA-H. No entanto, a HHSA-L obteve um tempo de processamento aproximadamente 83 vezes mais rápido que os da HHSA-H. A maioria das runs em L não ultrapassaram 5 minutos, já a maioria das runs em H levaram no máximo 1,5 horas. Os autores afirmam que, apesar da longa duração, rodar o HHSA-H pode ser viável na prática se o processamento ocorrer durante a noite, por exemplo.

Ainda no experimento II, explorou-se mais uma forma de analisar a relação do tempo de processamento e a quantidade de suites de teste geradas: o Amazon EC2 (Elastic Compute Cloud). Desta vez foram considerados apenas modelos de referência com t=2 em runs do HHSA em L e H, tendo como base comparativa os resultados da ferramenta CASA para as mesmas instâncias de referência.

O HHSA-L obteve os resultados mais otimizados, com o menor total do tamanho médio das suites de teste custando apenas 13 centavos de dollar, enquanto o CASA custou 49 centavos de dollar para chegar a resultados parecidos. O HHSA-M obteve o menor tamanho de suites de teste, no entanto custou 2 dollares e 9 cents.

Sobre experimento III, ao se examinar a desempenho individual dos NOs sem aprendizado online, verificou-se que utilizá-los aleatoriamente em conjunto produz tamanhos de suites de teste mais otimizados, porém inferiores à desempenho do HHSA padrão que utiliza aprendizado online.

Por fim, para verificar se o HHSA realmente aprende quais os melhores NOs em função de diferentes estágios de busca e diferentes instâncias de CIT, utilizou-se os mesmos dados de referência do experimento II, com o Amazon EC2, e do III. Os estados de busca foram divididos em 3 estágios – cedo, meio e tarde – nos quais foram verificadas quais as combinações de operados de navegação mais frequentes em cada uma delas. Observou-se evidências de que o HHSA está aprendendo de forma online.

Conclui-se a partir destes experimentos que o uso de hiper-heurísticas para gerar amostras de CIT é uma abordagem muito promissora, na qual ainda são possíveis muitos aperfeiçoamentos.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …):

Otimização em Engenharia de Software, Teste de Interação Combinatória, Hiper-Heurísticas.

Contexto de validação (quais estudos de caso foram aplicados, …):

A validação estatística foi obtida a partir de outros trabalhos, através do uso de modelos de referência já validados em outros experimentos.

3.5. FICHAMENTO 5 [Zamli et al. 2016]

Referência:

[Zamli et al. 2016] K. Z. Zamli, B. Y. Alkazemi, and G. Kendall. A tabu search hyper-heuristic strategy for t-way test suite generation. Applied Soft Computing, 44:57 - 74, 2016.

Classificação (nova abordagem, revisão, survey):

Nova abordagem.

Motivação do artigo:

O campo da Otimização em Engenharia de Software (SBSE) é um campo relativamente novo que tem proposto o uso de meta-heurísticas como estratégia de solução para o problema NP hard de geração de suites de teste de interação (t-way).

Muitos estudos têm demonstrado, através de experimentos comparativos relacionados com a geração de testes t-way, que estratégias baseadas em meta-heurísticas individuais aparentam gerar boas soluções. Visto que nenhuma meta-heurística é universalmente melhor que as demais, a possibilidade de utilizá-las de forma híbrida para aumentar a performance de estratégias t-way é a principal motivação deste artigo.

Objetivo do artigo:

O objetivo geral deste artigo é introduzir uma nova estratégia hibrída para a geração de conjuntos de teste de interação combinatória, chamada hiper-heurística de alto nível (HHH). Os objetivos específicos são: explicar o funcionamento do algoritmo da estratégia HHH; comparar a performance da HHH com outras estratégias existentes; e verificar os resultados através de análise estatística.

Rápida descrição de metodologias/técnicas/métodos relacionados ao artigo:

Quando se trata de estratégias para gerar suites de teste t-way existem 2 ramos: a abordagem aritmética e a abordagem computacional. A abordagem

aritmética se baseia principalmente em métodos de construção de arrays ortogonais (OT), sendo exemplos das soluções baseadas nesta abordagem o TConfig e CTS.

Sobre a abordagem computacional existem duas principais vertentes: um teste por vez (OTAT) e um parâmetro por vez (OPAT). As abordagens OTAT basicamente se constituem na estratégia de percorrer as interações desejadas para gerar a cada iteração um caso de teste, que é avaliado gulosamente para determinar se sua a cobertura sobre as interações que ainda não foram cobertas é a maior possível. Algumas soluções que empregam esta estratégia são: jenny, mAETG, TVG, ITCH, GTWay, PICT, CTE-XL e a pioneira AETG.

Já nas estratégias OPAT, os casos de teste são construídos incrementalmente através da extensão horizontal - acréscimo de parâmetros (colunas) – até o limite máximo, sendo que, ao se verificar que o conjunto de casos de teste não cobre todas as interações desejadas, realiza-se a extensão vertical – acréscimo de casos de teste (linhas). Nesta estratégia incluem-se as variantes da In-Parameter-Order (IPO), sendo exemplos: IPOG, IPOG-D e MIPOG.

Um ramo promissor de estratégias computacionais relativamente recente é o de estratégias baseadas em meta-heurísticas. Estas estratégias costumam ter como passo inicial a geração de um conjunto aleatório de soluções. A cada iteração as soluções são alteradas visando melhorias. O critério de parada, no geral, é que se obtenha um candidato que garanta a cobertura de todas as interações desejadas.

Alguns exemplos de estratégias para geração de casos de teste t-way são: algoritmo de busca harmônica (HSS), gerador de testes por enxame de partículas (PSTG), algoritmo da busca do cuco (CS), simulated annealing (SA), algoritmo genético (GA) e algoritmo da otimização da colônia de formigas (ACO).

A estratégia híbrida HHH (hiper-heurística de alto nível), emprega tabu-search (TS) como heurística de alto nível (HLH), selecionando heurísticas de baixo nível através de um mecanismo de aceitação de movimento baseado em três operadores: melhoria, diversificação e intensificação. A HHH utiliza quatro

heurísticas de baixo nível (LLH): otimização baseada em ensino-aprendizagem (TLBO), algoritmo de vizinhança global (GNA), otimização de enxame de partículas (PSO) e busca do cuco (CS).

A estratégia HHH teve parâmetros gerais e específicos calibrados. Os parâmetros gerais são: $\Theta_{max} = 20$ (quantidade máxima de iterações); S = 40 (tamanho da população); $F_{max} = 400000$ (quantidade máxima de avaliações de aptidão); e $Tabu_{max} = 4$ (tamanho da lista tabu). Os parâmetros específicos do PSO foram: $c_1 \wedge c_2 = 1.375$ (coeficientes de aceleração); e ω=0.3 (peso inercial). O parâmetro específico do CS foi o $p_a = 0.25$ (fator de elitismo).

Na análise estatística avaliou-se apenas o tamanho das suites de teste gerados. Os pesquisadores consideraram que seria injusto avaliar a performance em termos de tempo de execução do HHH com os resultados encontrados na literatura, por conta de não ser possível avaliar o todas as estratégias no mesmo hardware e, no caso das meta-heurísticas, o tempo de execução pode variar muito dependendo do número de avaliações de aptidão realizadas.

A análise estatística dos resultados obtidos foi baseada em comparações múltiplas entre pares, com um nível de confiança de 95% (α = 0,05). Foi utilizado o *teste de Friedman*, comparando o HHH com as demais estratégias analisadas. Entretanto, todas as estratégias que possuíam amostras incompletas foram ignoradas, pois o *teste de Friedman* só pode ser aplicado em amostras completas. A hipótese nula é rejeitada se o a *estatística de Friedman* (X^2) for maior que o valor crítico.

Após o *teste de Friedman*, é realizada uma *análise post-hoc de Wilcoxon-Rank sum* para identificar se há diferença estatística em relação aos pares de estratégias comparadas. A hipótese nula neste teste é que não há diferença significativa entre o tamanho do conjunto de teste gerado pela HHH e cada estratégia individual. A hipótese alternativa (H1) é que o tamanho do teste para HHH é menor que o de cada estratégia individual.

Comparações múltiplas causam a variação do valor de α. Para lidar com isso utilizou-se a *correção de Bonferroni-Holm* na qual os p-valores de cada estratégia são ordenados de forma crescente e aplica-se a fórmula:

$$\alpha_{Holm} = \frac{\alpha}{k - i + 1}$$

considerando-se que os p-valores estão ordenados conforme $P_i < \ldots < P_k$. Se $P_i < \alpha_{Holm}$ a hipótese nula em questão é rejeitada. A avaliação prossegue até P_k.

Desconsiderando-se as estratégias com amostras incompletas, nos conjuntos de instâncias das configurações de referência da literatura em CA/MCA; CA (N; t, v^7) com $2 \leq v \leq 5$ e $2 \leq t \leq 6$; e CA (N; t, 3^k) com $3 \leq k \leq 12$ e $2 \leq t \leq 6$, a hipótese nula foi rejeitada em todas as comparações, ou seja, estatisticamente o HHH obteve a melhor performance nestes casos. No conjunto de instâncias CA (N; 4, 5^k) com $5 \leq k \leq 12$, a hipótese nula foi rejeitada quando o HHH foi comparado com as estratégias TVG e HSS.

Nas demais comparações não houve diferença estatística significativa entre os conjuntos de testes gerados pelo HHH e pelas demais estratégias. Conclui-se que a estratégia HHH obteve resultados competitivos em relação as demais, inclusive obtendo melhor performance em alguns conjuntos de instâncias. Existem ainda muitas possibilidades de otimização que podem ser exploradas para gerar novas estratégias.

Neste trabalho, a função objetivo adotada retorna a quantidade de interações de tuplas não cobertas pelo conjunto de casos de teste atual Z (conjunto de variáveis de decisão Z_i), sendo descrito pelos autores:

$$f(Z) = |\{I \in VIL: ZcoversI\}|$$

$$sujeitoa: Z = \{Z_1, Z_2, \ldots, Z_i\} \in \{P_1, P_2, \ldots, P_i\}; ondei = \{1,2, \ldots, N\}$$

onde VIL é o conjunto de interações de tuplas (*I*) não cobertas, P_i é o intervalo (discreto) dos valores possíveis para cada variável de decisão, sendo:

$$\left(Z_i(1) < Z_i(2) < \ldots < Z_i(K)\right)$$

Tal que N é o número de varáveis de decisão e K é o número valores possíveis para as varáveis discretas.

Os autores destacam quatro ameaças a validade, em ordem de importância, sendo: I - falta do código fonte de algumas soluções comparadas

no experimento; II - o uso de parâmetros com valores ajustados em vez de valores ideais; III – o uso dos melhores valores ao invés da média dos valores; e IV – presença de valores omissos nos resultados de algumas das estratégias.

A ameaça I, relacionada as estratégias meta-heurísticas, ocorre por conta de muitos resultados encontrados na literatura não especificarem o código fonte e, consequentemente, houve frequente uso de dados obtidos exclusivamente de experimentos de outros estudos. Por conta disso existem parâmetros que não foram considerados durante a comparação – como, por exemplo, o número máximo de avaliações de aptidão –, que por conta disso não foi justa para todas as estratégias.

A ameaça I acabou por causar a III, pois, como muitos dos resultados que foram reutilizados diretamente da literatura não declaravam os valores médios, optou-se por utilizar os valores mais otimizados. A questão é que os resultados mais otimizados podem ter sido obtidos por sorte e as chances de isso ocorrer são ainda maiores nas estratégias meta-heurísticas, que possuem uma considerável variação.

Ao contrário das ameaças citadas anteriormente, a ameaça II ocorre pelo uso de valores paramétricos ajustados para algumas heurísticas de baixo nível (PSO e CS), ao invés do uso dos valores especificados em estudos relacionados. Isto ocorre porque, como essas meta-heurísticas estão sendo utilizadas em conjunto como LLH em um contexto hiper-heurístico em vez de serem utilizadas como algoritmo principal, assim os valores especificados na literatura não levam ao melhor desempenho do HHH, por isso os parâmetros foram ajustados.

Por último, a ameaça IV acarreta na ocorrência de amostras incompletas. Como o *teste de Friedman* e a *análise post-hoc de Wilcoxon-Rank sum* exigem amostras completas de todas as estratégias, a análise estatística completa foi comprometida e, consequentemente, não foi possível rejeitar a hipótese nula na maioria dos casos por conta da presença de valores omissos.

Domínio de aplicação (Teste de Software, Linhas de Prouto de Software, Engenharia de Software, …):

Otimização em Engenharia de Software, Teste de Software, Hiper-Heurística.

Contexto de validação (quais estudos de caso foram aplicados, …):

A validação foi obtida através de sete grandes conjuntos de instâncias do problema de geração de casos de teste, sendo eles: configurações de referência da literatura da notação de cobertura de arrays (CA)/cobertura mista de arrays (MCA); CA(N; t, 2^{10}) com $2 \leq t \leq 6$; CA (N; t, 5^{10}) com $2 \leq t \leq 6$; CA (N; 4, 5^{k}) com $2 \leq k \leq 12$; CA (N; 4, v^{10}) com $2 \leq v \leq 7$; CA (N; t, v^{7}) com $2 \leq v \leq 5$ e $2 \leq t \leq 6$; CA (N; t, 3^{k}) com $3 \leq k \leq 12$ e $2 \leq t \leq 6$; e configurações de modelos reais com $2 \leq t \leq 6$.

3.6. FICHAMENTO 6 [Zamli et al. 2017]

Referência:

[Zamli et al. 2017] K. Z. Zamli, F. Din, G. Kendall, B. S. Ahmed. An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suíte generation. Information Sciences, 399:121–153, 2017.

Classificação (nova abordagem, revisão, survey):

Nova abordagem.

Motivação do artigo:

Meta-heurísticas tem sido propostas na literatura para a geração de pequenos e eficazes conjuntos de casos de teste para lidar com falhas devido à interação em diferentes sistemas de software. Embora estratégias baseadas em meta-heurísticas sejam conhecidas por produzir um conjunto de teste de boa qualidade, frequentemente as estratégias exigem conhecimento específico do domínio para possibilitar o ajuste do algoritmo para a obtenção de soluções de ótima qualidade.

Por outro lado, as estratégias baseadas em hiper-heurísticas suportam a ideia de se alcançar ótimos resultados sem a exigência de conhecimento específico do domínio do problema, sendo uma estratégia de maior capacidade de generalização. Tendo ainda em vista que não existe uma meta-heurística que seja superior as demais na solução de todos os problemas de otimização, como sugere o teorema No Free Lunch, é torna-se propício o uso de

estratégias baseadas na hibridização de meta-heurísticas para o alcance de resultados mais otimizados.

Objetivo do artigo:

De forma geral, objetivo deste trabalho foi investigar o uso de mecanismos de seleção e aceitação de heurísticas como estratégia para a geração de conjuntos de testes de interação combinatória. Os mecanismos de seleção e aceitação estudados foram: *Exponential Monte Carlo with counter* (EMCQ); *Choice Function* (CF); *Improvement Selection Rules* (ISR); e *Fuzzy Inference Selection* (FIS).

Os experimentos visaram alcançar três objetivos específicos:

- Caracterizar a performance das hiper-heurísticas;

- Medir o padrão de distribuição dos operadores de busca selecionados por cada um dos mecanismos de seleção e aceitação;

- Comparar as hiper-heurísticas implementadas em relação a outras abordagens meta-heurísticas de referência.

Rápida descrição de metodologias/técnicas/métodos relacionados ao artigo: A fazer.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …):

Otimização em Engenharia de Software, Teste de Software, Hiper-Heurística.

Contexto de validação (quais estudos de caso foram aplicados, …): A fazer.

3.7. FICHAMENTO 7 [Carvalho et al. 2015]

Referência:

[Carvalho et al. 2015] V. R. de Carvalho, S. R. Vergilio, A. Pozo. Uma hiper-heurística de seleção de meta-heurísticas para estabelecer sequências de módulos para o teste de software. VI Workshop de Engenharia de Software Baseada em Busca, 1: 1-10, 2015.

Classificação (nova abordagem, revisão, survey):

Nova abordagem.

Motivação do artigo:

O problema de integração e teste de módulos visa determinar uma ordem em que os módulos de um sistema devem ser integrados e testados, de forma a conduzir ao menor custo possível. Sendo o custo influenciado por diversos fatores, este problema pode ser tratado de forma multi-objetiva. Dentre as abordagens propostas na literatura, destacam-se as soluções baseadas em busca.

Tais abordagens foram bem-sucedidas em determinadas instâncias do problema. Contudo, levando-se em conta que nenhum algoritmo pode ser considerado o melhor para qualquer sistema e contexto, é eminente a necessidade de escolha do algoritmo mais adequado conforme a instância do problema e, para o testador, identificar qual o melhor algoritmo para o seu sistema/contexto é, além de custoso, frequentemente inviável.

Motivados por essa lacuna na literatura e pela possibilidade reduzir o esforço de muitos testadores, os autores propõem uma hiper-heurística chamada MOCAITO-HH, que tem como objetivo selecionar em tempo de execução, durante o processo de otimização, um algoritmo de otimização multi-objetivo para determinar a sequência de módulos para o teste de integração associada ao menor custo para construção de *stubs*.

Objetivo do artigo:

Com o objetivo geral de facilitar a escolha do algoritmo multi-objetivo mais adequado ao sistema sendo testado, evitando o esforço por comparação de diferentes algoritmos, este trabalho propõe a abordagem hiper-heurística chamada MOCAITO-HH (Multi-objective Optimization and Coupling-based Approach for the Integration and Test Order problem using Hiper-Heuristics).

Assim, os autores almejaram como objetivos específicos:

- Introduzir a abordagem MOCAITO-HH a partir do seu alicerce teórico e comportamento do algoritmo;

- Avaliar o desempenho do MOCAITO-HH, verificando a existência de equivalência em relação às implementações do MOCAITO utilizando uma única

heurística de baixo nível em diferentes instâncias do problema, tendo como métricas o hypervolume e a quantidade de elementos do *pareto front*.

Rápida descrição de metodologias/técnicas/métodos relacionados ao artigo:

A abordagem MOCAITO-HH, que estende a abordagem MOCAITO, utiliza a *Choice Function*, que permite escolher durante a otimização o melhor algoritmo em um dado momento, dentre os algoritmos evolutivos: NSGA-II, SPEA-2 e IBEA. Os algoritmos evolutivos são ranqueados de acordo com os indicadores: Hypervolume; Spread; Algorithm Effort (AE); Ratio of Nondominated Individuals (RNI); e um mecanismo para troca de população que permite a alternância de execução entre os algoritmos.

Os resultados obtidos pela hiper-heurística mostram equivalência estatística com a execução do melhor algoritmo para cada caso, mas com a vantagem de não ser necessário realizar experimentos para saber qual é o melhor.

A continuar.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …):

Otimização em Engenharia de Software, Teste de Software, Hiper-Heurística.

Contexto de validação (quais estudos de caso foram aplicados, …): A fazer.

3.8. FICHAMENTO 8 [Guizzo et al. 2017]

Referência:

[Guizzo et al. 2017] G. Guizzo, S. R. Vergilio, A. T. R. Pozo, and G. M. Fritsche. A multi-objective and evolutionary hyper-heuristic applied to the integration and test order problem. Applied Soft Computing, 56:331-344, 2017.

Classificação (nova abordagem, revisão, survey):

Nova abordagem.

Motivação do artigo:

A motivação dos autores advém do fato de que os algoritmos evolutivos multi-objetivos (MOEAs) utilizados para solucionar problemas do campo de Engenharia de Software Baseada em Busca (SBSE), em especial o problema de Ordem de Teste de Integração (ITO), não são amigáveis aos praticantes de engenharia de software principalmente por conta da grande variedade de opções de parâmetros e algoritmos.

Motivados por este *gap*, estes pesquisadores introduzem o HITO (Hyper-heuristic for the Integration and Test Order Problem), para lidar especificamente com a escolha otimizada e adaptativa dos parâmetros de busca, de forma a possibilitar a aplicação de MOEAs *?(neste caso o NSGA-II)?* de maneira genérica e amigável ao testador.

Objetivo do artigo:

O objetivo geral deste trabalho é ampliar a publicação anterior da abordagem HITO. Este objetivo foi alcançado a partir dos seguintes objetivos específicos:

- Descrever as bases teóricas desta abordagem e o funcionamento do algoritmo de forma mais completa;

- Realizar análise experimental verificando a equivalência estatística das três variações do HITO, da MOEA/DD e NSGA-II na solução de diferentes instâncias do problema, frente aos indicadores médios de hypervolume e IGD, considerando duas funções objetivo diferentes;

- Analisar a quantidade média de vezes que cada Heurística de Baixo Nível (LLH) é aplicada em cada instância do problema, em cada uma das funções objetivo, por cada uma das três versões do HITO.

Rápida descrição de metodologias/técnicas/métodos relacionados ao artigo: A fazer.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …):

Otimização em Engenharia de Software, Teste de Software, Hiper-Heurística.

Contexto de validação (quais estudos de caso foram aplicados, …): A fazer.

3.9. FICHAMENTO 9 [Ferreira et al. 2017]

Referência:

[Ferreira et al. 2017] T. N. Ferreira, J. A. P. Lima, A. Strickler, J. N. Kuk, S. R. Vergilio, and A. Pozo. Hyper-heuristic based product selection for software product line testing. IEEE Computational Intelligence Magazine, 12(2):34-45, May 2017.

Classificação (nova abordagem, revisão, survey):

Nova abordagem.

Motivação do artigo:

No contexto de Linha de Produção de Software (SPL), a tarefa de definir quais os produtos – combinações praticáveis de *features* - mais interessantes a serem testados é um problema de otimização software, pois o teste exaustivo de todas as combinações de *features* é uma tarefa altamente custosa e quase sempre inviável.

A comunidade tem solucionado o problema de seleção de produto para teste de SPL com Algoritmos Evolucionários Multi-Objetivo (MOEAs). Uma situação recorrente na aplicação de MOEAs em problemas específicos é a necessidade de determinar valores adequados para parâmetros e escolher operadores apropriados.

Para lidar com esse tipo de situação a literatura tem indicado o uso de Hiper-Heurísticas (HH). Provavelmente, a principal motivação para os autores foi o fato de não encontrarem, na época, trabalhos que utilizem as HH como metodologia para solucionar o problema de testes de SPLs baseados no Modelo de *Features* (FM).

Objetivo do artigo:

O objetivo geral deste trabalho foi introduzir uma abordagem hiper-heurística para a seleção de produtos para o teste de SPLs baseado no Modelo de Features. Para alcançar o objetivo geral, os autores almejaram os seguintes objetivos específicos:

- Levantar a fundamentação teórica, abordando os conceitos de: Hiper-Heurísticas (HH), Estratégias Evolucionárias Multi-Objetivo (MOEAs), Linhas de Produção de Software (SPLs), Modelo de *Features* (FM).

- Explicar o funcionamento dos algoritmos evolvidos na abordagem;

- Realizar a avaliação experimental e descrevê-la, visando: determinar se existe algum MOEA que obtenha melhores resultados que os demais; comparar o desempenho entre os métodos de heurísticas de alto nível baseados em UCB seleção aleatória; e comparar as soluções geradas a partir de três funções objetivo e com as soluções geradas a partir de quatro funções objetivo.

Rápida descrição de metodologias/técnicas/métodos relacionados ao artigo: A fazer.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …):

Linhas de Produto de Software, Otimização em Engenharia de Software, Teste de Software, Hiper-Heurística.

Contexto de validação (quais estudos de caso foram aplicados, …): A fazer.

3.10. FICHAMENTO 10 [Deb et al. 2002]

Referência:

[Deb et al. 2002] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6: 182-197, 2002.

Classificação (nova abordagem, revisão, survey): Nova abordagem.

Motivação do artigo:

A motivação dos autores para a escrita deste artigo se deu pelas críticas que os MOEAs (Algoritmos Evolucionários Multi-Objetivo) que utilizam ordenação não-dominada e compartilhamento para a solucionar problemas multi-objetivo vinham recebendo na época. As críticas se relacionavam a: falta de elitismo; necessidade de configurar o parâmetro de compartilhamento para a preservação da diversidade de soluções; e o alto grau de complexidade. Mais uma motivação foi a necessidade de novas estratégias que fossem capazes de lidar com restrições. Motivados por essas lacunas na literatura, os autores introduzem o NSGA-II (Nondominated Sorting Genetic Algorithm II).

Objetivo do artigo: A fazer.

Rápida descrição de metodologias/técnicas/métodos relacionados ao artigo: A fazer.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …): Computação Evolucionária.

Contexto de validação (quais estudos de caso foram aplicados, …): A fazer.

3.11. FICHAMENTO 11 [Gómez e Coello 2015]

Referência:

[Gómez e Coello 2015] R. H. Gómez and C. A. C. Coello. Improved Metaheuristic Based on the R2 Indicator for Many-Objective Optimization. ACM Annual Conference on Genetic and Evolutionary Computation, 679-686, 2015.

Classificação (nova abordagem, revisão, survey): Nova abordagem.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …): Computação Evolucionária.

3.12. FICHAMENTO 12 [Li et al. 2014]

Referência:

[Li et al. 2014] K. Li, Q. Zhang, S. Kwong, M. Li and R. Wang. Stable Matching-Based Selection in Evolutionary Multiobjective Optimization. IEEE Transactions on Evolutionary Computation, 18: 909-923, 2014.

Classificação (nova abordagem, revisão, survey):

Estudo de caso, Nova abordagem.

Domínio de aplicação (Teste de Software, Linhas de Produto de Software, Engenharia de Software, …): Computação Evolucionária.

3.13. FICHAMENTO 13 [Santiago Júnior e Vijaykumar 2012]

Referência:

[Santiago Júnior e Vijaykumar 2012] SANTIAGO JÚNIOR, V. A.; VIJAYKUMAR, N. L. Case study: software embedded in satellite payload. p. 79-81. In: [Santiago Júnior e Vijaykumar 2012] SANTIAGO JÚNIOR, V. A.; VIJAYKUMAR, N. L. Generating model-based test cases from natural language requirements for space application software. Software Quality Journal, v. 20, n. 1, p. 77-143, 2012.

Classificação (nova abordagem, revisão, survey):

Estudo de caso, Nova abordagem.

Motivação do artigo:

Uma das principais motivações deste artigo é a de contribuir com o processo de Validação e Verificação através da geração de casos de teste baseado em modelos considerando entregáveis de requisitos em Linguagem Natural (NL), uma vez que a NL é simples e os stakeholders estão acostumados a utilizá-la no documento de requisitos.

Contudo, entregáveis em NL constantemente apresentam ambiguidade, vagueza, inconsistência e baixa compreensibilidade. Neste contexto, a geração automática de casos de teste é uma tarefa árdua e custosa, uma vez que se faz necessário o uso de modelos de diferentes cenários identificados em documentos de requisitos em linguagem natural. A dificuldade é especialmente maior em aplicações complexas reais, como é o caso de softwares embarcados em computadores on-board de satélites.

A motivação deste capítulo foi introduzir informações sobre o estudo de caso, *software embedded in satellite payload,* em que o SOLIMVA foi aplicado de modo a sustentar os capítulos subsequentes.

Objetivo do artigo:

O objetivo desta seção foi descrever o estudo de caso em que a metodologia SOLIMVA foi aplicada. O sistema de software em que essa metodologia foi aplicada é o SWPDC (Software for the Payload Data Handling Computer), que foi aplicado no contexto do projeto de pesquisa QSEE (Qualidade de Software Embarcado em Aplicações Espaciais), cujo um detalhe importante foi o uso dos padrões da Cooperação Europeia para Padronização Espacial (ECSS) para orientar a relação cliente-fornecedor.

Rápida descrição de metodologias/técnicas/métodos relacionados ao artigo:

A arquitetura funcional do subsistema de computação do projeto QSEE foi representada através de um diagrama, no qual foram relacionadas as unidades de computação de acordo com o um modelo de comunicação primário/secundário. As unidades computacionais são: Computador de Manipulação de Dados *On-Board* (OBDH), Computador de Manipulação de *Payload Data* (PDC), Pré-Processadores de Eventos (EPPs) e Computador de Experiências de Plasma Ionosférico (IONEX).

O OBDH é primário em relação ao PDC e ao IONEX. O OBDH é o computador responsável pelo processamento da plataforma do satélite, das informações e da formatação/geração de dados a serem transmitidos às estações terrestres.

O PDC é primário em relação aos EPPs e secundário em relação ao OBDH. O SWPDC está embarcado nesta unidade. O principal objetivo do PDC é obter os dados científicos dos EPPs e transmiti-los para o OBDH.

As principais funções do SWPDC são: (i) interação com os EPPs para a coleta de dados científicos, de diagnóstico e de teste; (ii) formatação de dados; (iii) gerenciamento de memória para armazenamento temporário de dados antes de transmiti-los ao OBDH; (iv) implementação de mecanismos de controle de fluxo; (v) geração de dados de limpeza; (vi) implementação de

mecanismos complexos de tolerância a falhas; e (vii) carregar os novos programas durante o vôo.

Domínio de aplicação:

Teste de Software, Engenharia de Software.

Contexto de validação (quais estudos de caso foram aplicados, …):

Para a validação, foi usado como estudo de caso um produto de software espacial no qual se comparou a cobertura de objetivos de teste e as características dos casos de teste executáveis obtidos pelo SOLIMVA e uma abordagem manual desenvolvida por um *expert*.

4.) ATIVIDADE 2: DEFINIÇÃO DE HIPER-HEURÍSTICAS DE SELEÇÃO

Foram escolhidos 5 métodos heurísticos de seleção para as avaliações experimentais rigorosas no contexto de geração conjuntos de teste. Estes métodos, seus respectivos mecanismos de seleção e referências encontram-se na Tabela 4.

Tabela 4 - Hiper-Heurísticas de Seleção Escolhidas

Método Heurístico de Seleção	Mecanismo de Aceitação	Referência
Simple Random	{All Moves, Great Deluge}	[Burke et al. 2013]
Choice Function	{All Moves, Great Deluge}	[Maashi et al. 2014] [Carvalho et al. 2015]
Simulated Annealing	{All Moves, Great Deluge}	[Jia et al. 2015]

Já os MOEAs escolhidos, que serão utilizados como as heurísticas de baixo nível, e suas respectivas referências encontram-se na Figura 2, que foi adaptada de Zamli e seus colaboradores [Zamli et al. 2017].

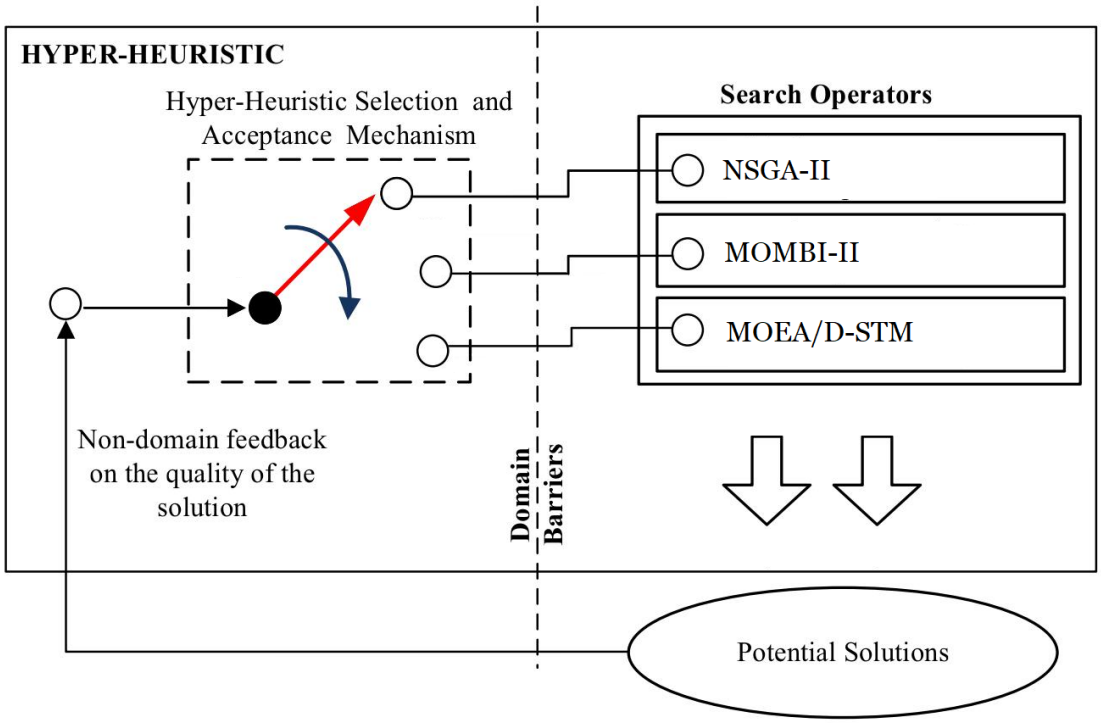

Figura 2 – Heurísticas de Baixo Nível

5.) ATIVIDADE 3: DEFINIÇÃO DOS PRODUTOS DE SOFTWARE ESPACIAIS

O produto de software espacial escolhido como estudo de caso foi o SWPDC [Santiago Júnior e Vijaykumar 2012]. Mais estudos de caso da área espacial podem ser incluídos no futuro.

Para viabilizar a comparação deste projeto com outras obras e aumentar a quantidade de estudos de caso, foram escolhidos parte dos 100 projetos do SF110 corpus benchmark, como proposto por Panichella e seus colaboradores [Panichella et al. 2018].

6.) ATIVIDADE 4: DESENVOLVIMENTO DA FERRAMENTA DE APOIO

A ferramenta computacional de apoio a avaliações experimentais deve buscar por uma fronteira de Pareto, cujos elementos são conjuntos de teste unitário para um dado programa P, que seja a mais adequada possível em relação às funções objetivo, sendo que P é um código fonte escrito em java. As métricas que serão avaliadas pelas funções se encontram na Tabela 5.

Tabela 5 – Métricas das Funções Objetivo

Métrica	Objetivo
Custo	Minimizar o tamanho das suites de teste.

Efetividade	Maximizar a cobertura de código-fonte;
	Maximizar o escore de mutação.
Eficiência	Minimizar o tempo que cada MOEA leva para executar.

Para o desenvolvimento da ferramenta em questão, que implementará as hiper-heurísticas de seleção escolhidas, foi escolhido o framework jMetal [Carvalho et al. 2015] para tratar das heurísticas de baixo nível. O jMetal 5.7 possui as implementações dos MOEAs escolhidos prontas para serem adaptadas na ferramenta.

Para tratar do teste de mutação, a ferramenta escolhida foi o MuJava [Ma et al. 2005]. Esta ferramenta automatiza as tarefas de gerar e matar mutantes, disponibilizando operadores de mutação tradicionais (nível de método) e operadores de mutação próprios à softwares orientados a objetos escritos em java (a nível de classe).

No decorrer desta atividade, o framework EvoSuite pode servir como referência durante a tarefa de desenvolvimento, pois é bem-sucedida ao utilizar uma meta-heurística simples para gerar casos de teste unitário [Panichella et al. 2018]. Outros frameworks que podem servir como modelos de referência são o HyFlex e o Hyperion, que auxiliam no desenvolvimento de hiper-heurísticas [Burke et al. 2013].

Além disso, a implementação da hiper-heurística MOCAITO-HH [Carvalho et al. 2015] tem sido consultada como uma fonte para o entendimento de como implementar adequadamente hiper-heurísticas utilizando o jMetal.

Como esta é a atividade com maior duração, tornou-se interessante dividi-la em etapas menores, conforme mostra a Tabela 6:

Tabela 6 - Etapas da Atividade 4

	Etapas	**Realização**
1	Estruturar a leitura das classes.	100%
2	Gerar mutantes do sistema sob teste com o mujava.	100%
3	Tratar da mensuração das métricas que serão utilizadas nas funções objetivo.	50%
4	Adaptar as hiper-heurísticas escolhidas na Atividade 2, em termos de métodos de seleção e mecanismos de aceitação.	10%
5	Tratar do compartilhamento da população entre os	0%

	diferentes MOEAs.	
6	Integrar o mecanismo de leitura de classes, os MOEAs, as funções objetivo, os mecanismos de aceitação e as hiper-heurísticas.	0%

Na etapa 1 foi utilizada a ferramenta ANTLR (ANother Tool for Language Recognition), que é uma ferramenta de análise sintática de domínio público [Parr e Quong 1995]. Esta ferramenta foi utilizada para gerar Árvores Sintáticas Abstratas (AST) a partir do código fonte do sistema sob teste, considerando o uso da gramática 'Java8.g4' e a versão 7.4.2 do ANTLR. Dessa forma, os dados necessários para a geração de casos de teste foram obtidos a partir da AST e estruturadas para serem utilizadas.

Considerando o código fonte de um projeto java, os dados foram modelados considerando cada um dos arquivos '.java'. Cada um destes arquivos possui como atributos: pacote, importações e declarações – deve-se frisar que o plural aqui é utilizado para indicar que o atributo é um conjunto. Cada declaração pode possuir declarações internas. Os tipos declarações foram definidos como: classe; enumerador; interface; anotação; construtor; método; e método de anotação.

Para facilitar o entendimento com uma explicação breve, os tipos de declaração serão divididos nos seguintes grupos:

A: classe, enumerador, interface, anotação;

B: método e construtor.

Os arquivos '.java' podem possuir mais de uma declaração, sendo que estas declarações se limitam aos tipos que pertencem ao grupo A. Essas declarações, por sua vez, podem possuir declarações internas de qualquer um dos tipos de declaração mencionados no parágrafo anterior, frisando que existem restrições em casos específicos.

Atributos comuns a todas as declarações do grupo A foram definidos como: modificadores comuns, modificadores de anotação, nome, e declarações internas. Já os atributos comuns ao grupo B são os mesmos do grupo A, com o acréscimo de: parâmetros e exceções. O tipo de declaração 'método de anotação' não foi agrupado em nenhum dos grupos porque seus atributos

diferem consideravelmente dos demais, sendo eles: modificadores de anotação, nome, valor padrão e tipo de retorno.

Sobre a etapa 2 dessa atividade, um detalhe interessante é que o MuJava também trabalha com o código fonte do sistema sob teste para gerar as mutações, mas é necessário fornecer o código compilado de dependências. Além disso, quando se trata de matar os mutantes e determinar o score de mutação, é necessário fornecer ao MuJava o código compilado do conjunto de testes.

Na etapa 3, mensurar o tempo de execução dos MOEAs é uma tarefa simples, assim como aferir o tamanho das suítes de teste. Já para obter o score de mutação será necessário tratar da compilação das suítes de teste durante a execução da ferramenta. Sobre a mensuração da cobertura, não há ferramenta definida, mas boas opções já são conhecidas, bastando apenas tomar a decisão.

Por fim, o que já se tem pronto da etapa 4 é determinação de valores constantes padrões a serem utilizados na codificação dos cromossomos. Foram tratados os tipos primitivos, os seus wrappers e o tipo ‘String’. Para os tipos primitivos foram determinados como valores o zero e os valores máximo e mínimo de cada tipo. Para os wrappers o valor nulo foi incluído.

Já para os tipos não primitivos, optou-se por determinar os valores manualmente, através do uso de dublês, considerando as constantes presentes no sistema sob teste sempre que possível. Valores constantes presentes sistema sob teste também serão considerados para os tipos que já possuem valor padrão definido, e, nos casos cabíveis, a operação de união será aplicada entre os entre os valores padrão e os específicos ao contexto.

7.) CONCLUSÃO

Este relatório final apresentou as atividades desenvolvidas, no período de 01 de agosto de 2018 a 15 de julho de 2019, relacionadas ao projeto “Avaliações Experimentais de Hiper-Heurísticas para Teste de Software Espacial”. Pode-se concluir que os objetivos para este período foram parcialmente alcançados.

A atividade 4, adaptação das hiper-heurísticas de seleção por meio de desenvolvimento de ferramenta computacional, deve ser completada antes do final de setembro. Já as atividades 5, 6 (atividades de avaliações experimentais rigorosas das métricas de custo e efetividade) e 7 (submissão de artigo) devem ser iniciadas e terminadas neste último semestre, conforme o cronograma.

8.) REFERÊNCIAS

[Shull et al. 2008] F. Shull, J. Singer and D. I. K. Sjoberg. Guide to Advanced Empirical Software Engineering. Springer-Verlag London, 2008.

[Harman et al. 2015] HARMAN, M.; JIA, Y.; ZHANG, Y. Achievements, open problems and challenges for search based software testing. In 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), pages 1-12, April 2015.

[Wegener e Bühler 2004] J. Wegener and O. Bühler. Evaluation of different fitness functions for the evolutionary testing of an autonomous parking system. In Kalyanmoy Deb, editor, Genetic and Evolutionary Computation - GECCO 2004, pages 1400-1412, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Everson e Fieldsend 2006] R. M. Everson and J. E. Fieldsend. Multiobjective optimization of safety related systems: an application to short-term confict alert. IEEE Transactions on Evolutionary Computation, 10(2):187-198, April 2006.

[Garvin et al. 2011] B. J. Garvin, M. B. Cohen, and M. B. Dwyer. Evaluating improvements to a meta-heuristic search for constrained interaction testing. Empirical Software Engineering, 16(1):61-102, Feb 2011.

[Petke et al. 2015] J. Petke, M. B. Cohen, M. Harman, and S. Yoo. Practical combinatorial interaction testing: Empirical findings on efficiency and early fault detection. IEEE Transactions on Software Engineering, 41(9):901-924, Sept 2015.

[Asoudeh e Labiche 2014] N. Asoudeh and Y. Labiche. Multi-objective construction of an entire adequate test suite for an efsm. In 2014 IEEE 25th International Symposium on Software Reliability Engineering, pages 288-299, Nov 2014.

[Ferrer et al. 2012] J. Ferrer, F. Chicano, and E. Alba. Evolutionary algorithms for the multi-objective test data generation problem. Softw. Pract. Exper., 42(11):1331-1362, November 2012.

[Mondal et al. 2015] D. Mondal, H. Hemmati, and S. Durocher. Exploring test suite diversification and code coverage in multi-objective test case selection. In 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), pages 1-10, April 2015.

[Shahbazi e Miller 2016] A. Shahbazi and J. Miller. Black-box string test case generation through a multi-objective optimization. IEEE Transactions on Software Engineering, 42(4):361-378, April 2016.

[Mahmoud e Ahmed 2015] T. Mahmoud and B. S. Ahmed. An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use. Expert Systems with Applications, 42(22):8753 - 8765, 2015.

[Burke et al. 2013] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society, 64(12):1695-1724, Dec 2013.

[Burke et al. 2010] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. • Özcan, and J. R. Woodward. A Classification of Hyper-heuristic Approaches, pages 449-468. Springer US, Boston, MA, 2010.

[Cohen 2017] M. B. Cohen. The evolutionary landscape of sbst: A 10 year perspective. In 2017 IEEE/ACM 10th International Workshop on Search-Based Software Testing (SBST), pages 47-48, May 2017.

[Jia et al. 2015] Y. Jia, M. B. Cohen, M. Harman, and J. Petke. Learning combinatorial interaction test generation strategies using hyperheuristic search. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, volume 1, pages 540-550, May 2015.

[Zamli et al. 2016] K. Z. Zamli, B. Y. Alkazemi, and G. Kendall. A tabu search hyper-heuristic strategy for t-way test suite generation. Applied Soft Computing, 44:57 - 74, 2016.

[Mariani et al. 2016] T. Mariani, G. Guizzo, S. R. Vergilio, and A. T. R. Pozo. Grammatical evolution for the multiobjective integration and test order problem. In Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, pages 1069-1076, New York, NY, USA, 2016. ACM.

[Guizzo et al. 2017] G. Guizzo, S. R. Vergilio, A. T. R. Pozo, and G. M. Fritsche. A multi-objective and evolutionary hyper-heuristic applied to the integration and test order problem. Applied Soft Computing, 56:331-344, 2017.

[Strickler et al. 2016] A. Strickler, J. A. P. Lima, S. R. Vergilio, and A. T. R. Pozo. Deriving products for variability test of feature models with a hyper-heuristic approach. Applied Soft Computing, 49:1232-1242, 2016.

[Ferreira et al. 2017] T. N. Ferreira, J. A. P. Lima, A. Strickler, J. N. Kuk, S. R. Vergilio, and A. Pozo. Hyper-heuristic based product selection for software product line testing. IEEE Computational Intelligence Magazine, 12(2):34-45, May 2017.

[Cowling et al. 2001] P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In Edmund Burke and Wilhelm Erben, editors, Practice and Theory of Automated Timetabling III, pages 176-190, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Fialho et al. 2009] Á. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In Thomas St• utzle, editor, Learning and Intelligent Optimization, pages 176-190, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Zannier et al. 2006] ZANNIER, C.; MELNIK, G.; MAURER, F. On the success of empirical studies in the international conference on software engineering. In Proceedings of the 28th International Conference on Software Engineering, ICSE '06, p. 341-350, New York, NY, USA, 2006. ACM.

[Dalmau e Gigou 1997] J. Dalmau and J. Gigou. Ariane-5: Learning from Flight 501 and Preparing for 502. 1997. ESA Bulletin Nr. 89. Available from: <http://www.esa.int/esapub/bulletin/bullet89/dalma89.htm>. Access in: July 3, 2018.

[Isbell e Savage 1999] D. Isbell and D. Savage. MARS CLIMATE ORBITER FAILURE BOARD RELEASES REPORT, NUMEROUS NASA ACTIONS UNDERWAY IN RESPONSE. 1999. Available from: <https://mars.nasa.gov/msp98/news/mco991110.html>. Access in: July 3, 2018.

[Santiago Júnior e Vijaykumar 2012] SANTIAGO JÚNIOR, V. A.; VIJAYKUMAR, N. L. Generating model-based test cases from natural language requirements for space application software. Software Quality Journal, v. 20, n. 1, p. 77-143, 2012. DOI: 10.1007/s11219-011-9155-6.

[Balera e Santiago Júnior 2016] BALERA, J. M.; SANTIAGO JÚNIOR, V. A. 2016. A Controlled Experiment for Combinatorial Testing. In Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing (SAST). ACM, New York, NY, USA, Article 2, 10 pages. https://doi.org/10.1145/2993288.2993289.

[Balera e Santiago Júnior 2017] BALERA, J. M.; SANTIAGO JÚNIOR, V. A. 2017. An algorithm for combinatorial interaction testing: definitions and rigorous evaluations. Journal of Software Engineering Research and Development 5, 1 (28 Dec 2017), 41. https://doi.org/10.1186/s40411-017-0043-z.

[Zamli et al. 2017] K. Z. Zamli, F. Din, G. Kendall, B. S. Ahmed. An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation. Information Sciences, 399:121–153, 2017. Available from: <dx.doi.org/10.1016/j.ins.2017.03.007>. Access in: January 25, 2019.

[Carvalho et al. 2015] V. R. de Carvalho, S. R. Vergilio, A. Pozo. Uma hiper-heurística de seleção de meta-heurísticas para estabelecer sequências de módulos para o teste de software. VI Workshop de Engenharia de Software Baseada em Busca, 1: 1-10, 2015. Available from: <http://doi.org/10.13140/RG.2.1.5092.9129>. Access in: January 25, 2019.

[Deb et al. 2002] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6: 182-197, 2002. Available from: <doi.org/10.1109/4235.996017>. Access in: January 25, 2019.

[Gómez e Coello 2015] R. H. Gómez and C. A. C. Coello. Improved Metaheuristic Based on the R2 Indicator for Many-Objective Optimization. ACM Annual Conference on Genetic and Evolutionary Computation, 679-686, 2015. Available from: <dx.doi.org/10.1145/2739480.2754776>. Access in: January 25, 2019.

[Li et al. 2014] K. Li, Q. Zhang, S. Kwong, M. Li and R. Wang. Stable Matching-Based Selection in Evolutionary Multiobjective Optimization. IEEE Transactions on Evolutionary Computation, 18: 909-923, 2014. Available from: <http://doi.org/10.1109/TEVC.2013.2293776>. Access in: January 25, 2019.

[Panichella et al. 2018] A. Panichella; F. M. Kifetew; P. Tonella. Automated Test Case Generation as a Many-Objective Optimisation Problem with Dynamic Selection of the Targets. IEEE Transactions on Software Engineering, vol. 44, n. 2, p. 122-158, 2018. Available from: <http://doi.org/10.1109/TSE.2017.2663435>. Access in: January 25, 2019.

[Maashi et al. 2014] M. MAASHI; E. ÖZCAN; G. KENDALL. A multi-objective hyper-heuristic based on choice function. Expert Systems with Applications, Elsevier, vol. 41, p. 4475-4493, 2014. Available from: <http://doi.org/10.1016/j.eswa.2013.12.050>. Access in: January 25, 2019.

[Ma et al. 2005] Y. S. MA; J. OFFUTT; Y. R. KWON. MuJava: an automated class mutation system. Software Testing, Verification and Reliability, Wiley, vol. 15, p. 97–133, 2005. Available from: <http://doi.org/10.1002/stvr.308>. Access in: July 15, 2019.

[Parr e Quong 1995] T. J. PARR; R. W. QUONG. ANTLR: A Predicated-LL(k) Parser Generator. Software - Practice and Experience, vol. 25, p. 789-810, 1995. Available from: <https://doi.org/10.1002/spe.4380250705>. Access in: July 15, 2019.