MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Otimização em Teste de Software para Análise de Desempenho de Software de Pré-Processamento de Dados do CPTEC

Arthur Genúncio da Cunha Menezes Costa

Relatório final de projeto de Iniciação Científica, orientado pelo Dr. Valdivino Alexandre de Santiago Júnior e Dr. Eduardo Batista de Moraes Barbosa.

INPE
São José dos Campos
2021

RESUMO

Diariamente, milhares de observações da Terra e do espaço sobre as condições da atmosfera e dos oceanos são coletadas e unificadas no Global Observing System (GOS) da World Meteorological Organization (WMO). Essas observações são distribuídas por meio do Global Telecommunication System (GTS) para subsidiar os sistemas operacionais de previsão numérica de tempo nos centros de meteorologia do mundo. Em linhas gerais, as observações são constituídas por medidas diretas (in situ) de direção e intensidade do vento, pressão, temperatura e umidade; observações visuais de nuvens, visibilidade e tipo de fenômeno meteorológico; e medidas indiretas (de sensores) de temperatura, umidade, nuvens e vento. O CPTEC/INPE adquire um vasto conjunto de observações por meio do GTS, cujo volume diário é bastante significativo, se aproximando de 7 GBytes. Atualmente, no CPTEC, existem limitações com a etapa de pré-processamento em tempo real, relacionadas justamente a esse grande volume de dados que precisam ser processados em tempo conveniente. Isto ocorre devido às limitações técnicas, principalmente de hardware da infraestrutura computacional disponível para a equipe de pré-processamento do CPTEC. Em Otimização em Teste de Software (OTS), o problema de testar software é formulado como um problema de Otimização, e essa é uma área que tem chamado bastante atenção da comunidade acadêmica. Usar OTS baseando-se em meta-heurísticas e hiper-heurísticas para gerar casos de teste que avaliam o desempenho do software de pré-processamento de dados do CPTEC é um caminho bastante interessante, pois poderiam ser identificados gargalos (bottlenecks) no código. Os objetivos específicos desse projeto são: a.) realizar análise de desempenho para o software da atividade de préprocessamento de dados do CPTEC via Otimização em Teste de Software; b.) propor melhorias para a codificação do software de pré-processamento de dados de acordo com a análise de desempenho realizada. Esse relatório apresenta as atividades desenvolvidas no período de 01 de setembro de 2020 a 30 de novembro de 2021.

SUMÁRIO

Pág.

1 INTRODUÇÃO

As observações meteorológicas (e ambientais e geofísicas relacionadas) são feitas por uma variedade de razões. Elas são usados para a preparação, em tempo real, de análises meteorológicas, previsões e avisos meteorológicos, para o estudo do clima, para operações locais dependentes do clima, para hidrologia e meteorologia agrícola e para pesquisas em meteorologia e climatologia [WMO 2018].

Diariamente, milhares de observações da Terra e do espaço sobre as condições da atmosfera e dos oceanos são coletadas e unificadas no Global Observing System (GOS) da World Meteorological Organization (WMO). Essas observações são distribuídas por meio do Global Telecommunication System (GTS) para subsidiar os sistemas operacionais de previsão numérica de tempo nos centros de meteorologia do mundo. Em linhas gerais, as observações são constituídas por medidas diretas (in situ) de direção e intensidade do vento, pressão, temperatura e umidade; observações visuais de nuvens, visibilidade e tipo de fenômeno meteorológico; e medidas indiretas (de sensores) de temperatura, umidade, nuvens e vento [Cintra 2005].

A assimilação de dados é uma técnica que tem sido amplamente aplicada em investigações da atmosfera, oceano e superfície terrestre. Tal abordagem combina dados de observação e os princípios dinâmicos subjacentes que governam o sistema para fornecer uma estimativa do estado do sistema que é melhor do que poderia ser obtido usando apenas os dados ou o modelo sozinho [Zhang e Moore 2015]. Portanto, uma atividade de assimilação de dados precisa das observações em formato e com qualidade adequados para que, em tempo hábil, seja possível, pela combinação de tais observações com o modelo de previsão numérico, gerar o novo estado inicial para a próxima integração do modelo de previsão.

O CPTEC/INPE adquire um vasto conjunto de observações por meio do GTS, cujo volume diário é bastante significativo, se aproximando de 7 GBytes. Existem diferentes tipos de observações do GOS (SYNOP, BUOY, PILOT, …) que precisam ser pré-processadas no CPTEC para dar apoio à atividade de assimilação de dados. Atualmente, no CPTEC, existem limitações com a etapa de pré-processamento em tempo real, relacionadas justamente a esse grande volume de dados que precisam ser processados em tempo conveniente. Isso é devido a limitações técnicas, principalmente de hardware da infraestrutura computacional disponível para a equipe de pré-processamento do CPTEC.

No entanto, metodologias e técnicas de Otimização de sistemas, tais como hiperheurísticas [Santiago Júnior et al. 2020][Balera e Santiago Júnior 2019][Drake et al. 2019] e meta-heurísticas [Deb e Jain 2014][Zitzler e Kunzli 2004] podem ser aplicadas para problemas de análise de desempenho. Particularmente, a comunidade de Teste de Software tem usado Otimização para ajudar a resolver seus problemas. Esse campo se denomina justamente Otimização em Teste de Software (OTS) onde o problema de testar software é formulado como um problema de Otimização [Harman et al. 2015][Balera e Santiago Júnior 2019]. OTS tem tido um grande interesse onde, em maior instância, metaheurísticas tais como Algoritmos Evolutivos, Particle Swarm Optimisation, Simulated Annealing têm sido usadas. Em menor instância, hiper-heurísticas de seleção e geração têm sido aplicadas para esse propósito.

Portanto, usar OTS baseando-se em meta-heurísticas e hiper-heurísticas para gerar casos de teste que avaliam o desempenho do software de pré-processamento de dados do CPTEC é um caminho bastante interessante, pois poderiam ser identificados gargalos (bottlenecks) no código. Uma vez identificados os gargalos, sugestões de melhorias no código da atividade de pré-processamento poderiam ser indicadas para, considerando a mesma plataforma computacional disponível para a equipe do CPTEC, poder realizar essa etapa de preparação de dados sem necessidade de usar um ambiente de supercomputação. Além disso, o tempo total demandado para toda a etapa de pré-

processamento de dados do CPTEC/INPE pode ser melhorado como resultado de tais esforços de pesquisa.

Desse modo, os objetivos específicos desse projeto são:

a.) Realizar análise de desempenho para o software da atividade de pré-processamento de dados do CPTEC via Otimização em Teste de Software;
b.) Propor melhorias para a codificação do software de pré-processamento de dados de acordo com a análise de desempenho realizada.

Esse relatório apresenta as atividades desenvolvidas no período de **01 de setembro de 2020 a 30 novembro de 2021.**

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. Realizar uma revisão de literatura sobre as áreas de conhecimento associadas ao projeto: Otimização em Teste de Software (OTS), requisitos não funcionais e análise de desempenho;
2. Familiarização com o software de pré-processamento de dados do CPTEC;
3. Estudar mecanismos para viabilizar análise de desempenho de software (instrumentação de código, profiles, medidas);
4. Estudar as hiper-heurísticas e meta-heurísticas que servirão para a geração de casos de teste visando a análise de desempenho do software de pré-processamento;
5. Projetar e implementar um método que permita realizar a análise de desempenho do software de pré-processamento;
6. Realizar avaliação experimental rigorosa para validar a abordagem proposta;
7. Submeter artigo para conferência e/ou workshop e/ou simpósio na área de Engenharia de Software ou Otimização ou Inteligência Artificial, e elaborar relatório final de atividades.

O cronograma de atividades pode ser visualizado na Figura 1, onde cada atividade está de acordo com os números acima, e cada uma das colunas de Ano I e II representa dois meses.

Atividade	Ano I						Ano II					
	2	4	6	8	10	12	2	4	6	8	10	12
1												
2												
3												
4												
5												
6												
7												

Figura 1 - Cronograma de Atividades

Dessa forma, esse relatório compreende o mês 1/Ano I (setembro/2020) ao mês 15/Ano II (novembro/2021) do projeto. Considerando as atividades previstas para serem desenvolvidas, conforme a Figura 1, a Tabela 1 exibe as atividades concluídas e a concluir considerando o período referente a este relatório.

Tabela 1 – Etapas Concluídas e a Concluir

	Atividades da Metodologia	Previsão	Realização
1	Realizar uma revisão de literatura sobre as áreas de conhecimento associadas ao projeto: Otimização em Teste de Software (OTS), requisitos não funcionais e análise de desempenho;	100%	100%
2	Familiarização com o software de pré-processamento de dados do CPTEC;	100%	100%
3	Estudar mecanismos para viabilizar análise de desempenho de software (instrumentação de código, profiles, medidas);	100%	90%
4	Estudar as hiper-heurísticas e meta-heurísticas que servirão para a geração de casos de teste visando a análise de desempenho do software de pré-processamento;	100%	50%
5	Projetar e implementar um método que permita realizar a análise de desempenho do software de pré-processamento;	50%	20%
6	Realizar avaliação experimental rigorosa para validar a abordagem proposta;	0%	0%
7	Submeter artigo para conferência e/ou workshop e/ou simpósio na área de Engenharia de Software ou Otimização ou Inteligência Artificial, e elaborar relatório final de atividades.	35%	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 setembro de 2020 a 30 de novembro de 2021). Desse modo, pode-se dizer que as atividades previstas para esse período da bolsa foram realizadas de forma satisfatória. As atividades 1 e 2 foram todas concluídas, assim como a atividade 3 foi quase toda concluída. A atividade 4 é a uma das mais demandantes pois demanda um profundo entendimento das meta-heurísticas e hiper-heurísticas que seriam usadas. Sobre a atividade 5, um método já foi concebido para realizar a análise de desempenho do software de pré-processamento e começou a ser implementado. A atividade 6 não estava prevista para esse período do relatório e a atividade 7 seria alcançada mais para o final do projeto.

A seguir, o detalhamento do desenvolvimento dessas atividades será apresentado.

3 ATIVIDADE 1: REVISÃO DA LITERATURA

Essa seção descreve os artigos lidos para que a atividade de revisão da literatura pudesse ser desenvolvida.

3.1 *DynaMOSA*

Referência:
Título: Automated Test Case Generation as a Many-Objective Optimisation Problem with Dynamic Selection of the Targets
Autores: Annibale Panichella, Fitsum Meshesha Kifetew, Paolo Tonella
Acesso: https://ieeexplore.ieee.org/document/7840029

Motivação do artigo:
A geração de casos de teste para otimização em teste de software é um problema multiobjetivo, no entanto, a aplicação de algoritmos de otimização multiobjetivos nunca haviam sido aplicadas neste campo.

Objetivo do artigo:
O artigo busca amplificar a aplicação de um algoritmo de geração e ordenação de casos de teste, com geração automática de casos de teste de forma dinâmica, baseado na hierarquia de controle de dependência para realizar uma abordagem mais efetiva, e eficiente para casos de recursos limitados que maximize a cobertura do teste.

Resultados de avaliações experimentais:
Como resultado, o artigo mostra que a aplicação do DynaMOSA tem um desempenho significativamente melhor do que outras alternativas de cobertura de ramificações de classes, obtendo uma aproximação mais efetiva.
O DynaMOSA também leva vantagem em comparação com o algoritmo anterior, o MOSA, tendo um desempenho significativamente melhor, em 20% dos casos de teste.
Por fim, o DynaMOSA tem um desempenho semelhante à outras aplicações de cobertura dos casos de testes de fortes mutações, mas obtém um desempenho superior no restante das aplicações.

Contexto de validação:
A metodologia foi aplicada e conduzida em 346 classes em Java extraídas de 117 projetos, pertencendo a quatro tipos diferentes de datasets.

3.2 *Evo Multi-Objective*

Referência:
Título: Evolutionary Algorithms for the Multi-Objective Test Data Generation Problem
Autores: Javier Ferrer, Francisco Chicano, Enrique Alba
Acesso: https://dl.acm.org/doi/10.1002/spe.1135

Motivação do artigo:
O custo para testar um software pode ser muito alto, contando que é necessário executar completamente o teste e checar o comportamento do sistema, sendo, portanto, fundamental a busca por uma forma de minimizar o custo e aumentar a eficiência da geração de dados para casos de teste.

Objetivo do artigo:
Maximizar a cobertura e eficácia da geração de dados para casos de teste multiobjetivo a partir da construção de um algoritmo evolutivo, de forma que se reduza os custos necessários para tal tarefa, mas mantendo a eficiência da abordagem via casos de teste.

Resultados de avaliações experimentais:

O estudo conclui que os algoritmos GA e MOCell obtiveram os melhores desempenhos nos testes, tendo o MOCell obtido melhores desempenhos em programas de menor grau de aninhamento, já o GA obteve um resultado significativamente melhor na maioria dos outros programas de maior grau de aninhamento, indicando uma média de cobertura total durante os teste.
Com isso, se conclui que GA é a melhor alternativa caso o programa necessite de uma ampla cobertura de casos de teste ou caso tenha um alto grau de aninhamento.
Outras abordagens (MM e mM) também obtiveram bons resultados, reduzindo o número de casos de teste necessários para obter uma boa cobertura, e também reduzindo bastante os custos.

Contexto de validação:
Os algoritmos do artigo foram testados em dois conjuntos de problemas: um contendo 800 programas criados por meio de um gerador de programas e o outro contendo 13 programas reais, extraídos de outras fontes.

3.3 *HH_CF*

Referência:
Título: A multi-objective hyper-heuristic based on choice function
Autores: Mashael Maashi, Ender Özcan, Graham Kendall
Acesso: https://www.sciencedirect.com/science/article/abs/pii/S095741741400013X

Motivação do artigo:
Como hiper-heurísticas são uma metodologia emergente no campo da utilização de heurísticas para resolver problemas complexos de otimização computacional, a maior parte das pesquisas na área eram relacionadas à otimização com objetivo único, tendo poucos trabalhos anteriores apresentado hiper-heurísticas para problemas multiobjetivo. O estudo prevê combinar as forças dos algoritmos evolutivos conhecidos no campo de pesquisa de meta-heurísticas (NSGAII, SPEA2 e MOGA) para construir uma hiper-heurística de seleção para resolver problemas de otimização multiobjetivo.

Objetivo do artigo:
Contribuir para o campo de otimização a partir de hiper-heurísticas apresentando um método feito especialmente para otimização multi-objetiva a partir de algoritmos evolutivos.
Um trabalho que utiliza esses componentes desenvolvidos para otimização multi-objetiva como este não havia sido realizado pela comunidade cientifica que atua na área, sendo este a primeira aproximação.

Resultados de avaliações experimentais:
Os resultados demonstraram eficiência e potencial do produto da pesquisa realizada em resolver problemas de otimização contínua multi-objetiva. A HH_CF teve melhor desempenho do que as heurísticas de baixo nível tais como NSGAII, SPEA2 e MOGA, também obtendo um melhor desempenho em relação à outras abordagens de hiperheurísticas multi-objetivas: a HH_RAND e a AMALGAM.
Apesar de demonstrar um desempenho satisfatório e promissor na maioria dos testes empíricos, a HH_CF não é capaz de evitar as fraquezas das heurísticas de baixo nível utilizadas, visto que a baixa performance do MOGA influencia diretamente na performance da HH_CF. Contudo, a abordagem apresentada na pesquisa demonstra potencial neste campo.

Contexto de validação:
São realizados os casos de teste de benchmark da Walking Fish Group e uma aplicação do produto ao problema real de pesquisa da resistência de choque de veículos.

3.4 *HH-RILA*

Referência:
Título: A Learning Automata based Multiobjective Hyper-heuristic
Autores: Wenwen Li, Ender Özcan, Robert John
Acesso: https://ieeexplore.ieee.org/document/8231198

Motivação do artigo:
A aplicação de meta-heurísticas em problemas de otimização, apesar de ser bem sucedida, acaba requisitando da intervenção de um especialista, o que compromete o processo de otimização no geral e utiliza muito recurso.

Objetivo do artigo:
Desenvolver uma abordagem que automatiza a busca de determinadas heurísticas que podem ser reutilizadas para problemas com diferentes características, para que necessite menos envolvimento de especialistas.

Resultados de avaliações experimentais:
O estudo apresentou dois algoritmos de hiper-heurísticas de seleção: HH-LA e HH-RILA, que investigaram três problemas de otimização multiobjetivo. HH-RILA teve uma performance melhor que HH-LA durante os testes, e ambas tiveram uma excelente performance na aplicação das hiper-heurísticas de escolhas aleatórias e de aprendizado. O estudo revelou que o algoritmo tem um melhor desempenho, portanto.

Contexto de validação:
A metodologia foi aplicada a três problemas de otimização multiobjetivo, sendo duas funções de benchmarking WFG e DLTZ e um problema de acidente de veículos.

3.5 *IBEA*

Referência:
Título: Indicator-Based Selection in Multiobjective Search
Autores: Eckart Zitzler, Simon Künzli
Acesso: https://link.springer.com/chapter/10.1007/978-3-540-30217-9_84

Motivação do artigo:
Existe uma dificuldade em se alcançar uma boa aproximação da Eficiência de Pareto, no que se diz respeito ao cenário de otimização multiobjetivo, visto que não há exatamente uma definição concreta do que é uma boa aproximação dentro deste quesito, junto ao fato observado que as abordagens recentes podem convergir no que se diz da seleção preferencial de informação para otimização multiobjetivo. Portanto, o artigo busca integrar uma ideia apresentada numa pesquisa anterior para solucionar este problema.

Objetivo do artigo:
Buscar uma forma de definir o objetivo da otimização nos termos dos indicadores de performance e aplicar essa medida no processo de seleção da otimização multiobjetivo,

tendo como diferencial a adaptabilidade, pois o algoritmo IBEA pode ser alterado à preferência do usuário. O produto do estudo visa impulsionar os resultados gerados por algoritmos produzidos anteriormente.

Resultados de avaliações experimentais:
Após os testes, se pôde concluir que os algoritmos propostos tem uma performance significativamente melhor que SPEA2 e NSGA-II, respeitando os indicadores de performance. Foi implementada uma versão adaptativa do SPEA2 para observar a eficiência do método, mas, no entanto, não produziu nenhum resultado significativamente diferente.Contudo, se conclui que, para os problemas de teste utilizados, as duas versões do IBEA (IBEAe e IBEAhd) obtiveram uma performance bem melhor do que o SPEA2 e o NSGA-II.

Contexto de validação:
O algoritmo proposto foi testado em problemas conhecidos de benchmark: o problema da mochila, com 100 itens, o problema da aplicação do processador de rede, com dois, três e quatro objetivos, respectivamente, do EXPO2, EXPO3 e EXPO4.Também se foi testado em outras quatro funções de teste: ZDT6 e KUR, com 2 objetivos cada, e DTLZ2 e DTLZ6, com três objetivos cada.

3.6 *MOEA-DD*

Referência:
Título: An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition
Autores: Ke Li, Kalyanmoy Deb, Qingfu Zhang, Sam Kwong
Acesso: https://ieeexplore.ieee.org/document/6964796

Motivação do artigo:
Muito esforço havia sido utilizado para o desenvolvimento de algoritmos evolutivos de otimização multiobjetivo, com foco na aplicação com dois ou três objetivos. No entanto, observa-se que muitas aplicações reais envolvem quatro ou mais objetivos, necessitando tratar de um maior número de objetivos, que, anteriormente era inviável na área.

Objetivo do artigo:
Apresentar um paradigma unificado que combina abordagens de dominância e de decomposição para encontrar soluções de problemas com múltiplos objetivos (até quinze) com desempenho similar ou superior a outras propostas similares e abordagens previamente utilizadas, para isso utilizando uma abordagem sistemática para gerar vetores que dimensionam o peso e definem os subproblemas dos objetivos.

Resultados de avaliações experimentais:
O algoritmo apresentado no artigo é capaz de obter uma performance significativamente melhor do que outros algoritmos utilizados nos mesmos testes, como o C-NSGA-III, em quase todos as instâncias de teste, exceto quando o algoritmo utiliza um valor muito alto de objetivos. Em um teste experimental, no entanto, o C-MOEA/D sofre com a perca da diversidade de sua população, se comparado com outros algoritmos, mas mesmo assim obtendo resultados melhores que algoritmos similares em casos de três e cinco objetivos.

Contexto de validação:
Para os testes empíricos, os problemas experimentais utilizados são do DTLZ1 até o DTLZ4 e o WFG1 até o WFG9.

3.7 *NSGA-III*

Referência:
Título: An Evolutionary Many-Objective Optimization Algorithm Using Reference-PointBased Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an
Adaptive Approach
Autores: Himanshu Jain, Kalyanmoy Deb
Acesso: https://ieeexplore.ieee.org/document/6595567

Motivação do artigo:
Algoritmos evolutivos de otimização para solução de problemas multiobjetivo, desde o início de suas propostas, demonstraram a eficiência de seu uso para otimização de vários problemas, pois são capazes de encontrar múltiplas soluções e têm flexibilidade para focar em qualquer parte da Fronteira de Pareto. Mas, apesar de serem eficientes, os algoritmos evolutivos ainda apresentam problemas em relação à desempenho, difíceis de serem solucionados de maneira adequada.

Objetivo do artigo:
A adaptação de um algoritmo evolutivo de otimização multiobjetivo, extensão do popular NSGA-II, para alcançar uma solução mais próxima da Fronteira de Pareto a partir de uma aproximação baseada em separar a população do algoritmo em nichos e encontrar a eficiência de Pareto para cada um desses pontos de referência, com o intuito de obter uma performance significativamente melhor que outros algoritmos de otimização multiobjetivo utilizados.

Resultados de avaliações experimentais:
Os testes demonstraram a eficácia da abordagem do A-NSGA-III, que se mostra capaz de encontrar as soluções para cada ponto de referência criado e associado a uma população, encontrando, também, melhores e mais distribuídos pontos na parte factível da Fronteira de Pareto.

Contexto de validação:
O algoritmo foi testado em diferentes tipos de testes experimentais (Problema DTLZ1 invertido e o Problema C2-DTLZ2) e em dois testes práticos, da eficiência de design de resistência ao impacto em veículos e em impactos laterais em carros.

3.8 *OTS*

Referência:
Título: Otimização em Teste e Software com Aplicação de Metaheurísticas
Autores: Fabrício Gomes de Freitas, Camila Loiola Brito Maia, Gustavo Augusto Lima de campos e Jefferson Teixeira de Souza
Acesso: http://www.fsma.edu.br/si/edicao5/FSMA_SI_2010_1_Estudantil_1.pdf

Motivação do artigo:
Realizar um estudo de diferentes técnicas de otimização em teste de software.

Objetivo do artigo:

Fomentar a pesquisa de um campo então recente da tecnologia empregada na otimização das técnicas de teste de software e engenharia de software e servir como base para futuras pesquisas na área. Foi feita uma análise de testes previamente realizados.

Resultados de avaliações experimentais:
Foram avaliados métodos e técnicas realizados por outros artigos e relatórios que abordam empiricamente o tema, obtendo resultados diferentes para diversos contextos encontrados em problemas de engenharia de software, mas sendo consensual que a aplicação de metaheurísticas neste campo de pesquisa supera a eficiência no teste de software em comparação a abordagens aleatórias.

Contexto de validação:
Foram aplicados em códigos complexos diversos da engenharia de software, tirados de relatórios e artigos realizados anteriormente que geravam casos de teste e conjuntos de dados para testes de software.

3.9 *Systematic Mapping - Hyper-Heuristics*

Referência:
Título: A systematic mapping addressing Hyper-Heuristics within Search-based Software Testing
Autores: Juliana Marino Balera, Valdivino Alexandre de Santiago Júnior
Acesso: https://www.sciencedirect.com/science/article/abs/pii/S0950584919301430

Motivação do artigo:
Otimização em Teste de Software (OTS) é uma área que vem recebendo muita atenção na comunidade acadêmica da área de Teste de Software. Formulando o teste de um produto de software como um problema de otimização, OTS pode beneficiar na busca para soluções destes problemas.

Objetivo do artigo:
O objetivo do artigo é investigar o uso de hiper-heurísticas na área de teste de software, dando destaque aos esforços atuais e procurando identificar novas direções para as pesquisas correlatas a área especificada.

Resultados de avaliações experimentais:
Dos 164 estudos relacionados a Teste de Software mapeados, apenas vinte e nove eram atribuídos a ideia de que o artigo apresenta: a utilização de hiper-heurísticas em teste de software. A pequena quantidade de trabalhos obtidos apresentou evidências da necessidade de mais esforços nessa questão.

Contexto de validação:
Foi realizada uma leitura e comparação dos artigos estudados e, com isso, se concluiu que existiram contextos de validações diferentes para estudos específicos.

3.10 *HRISE*

Referência:

Título: Hyper-Heuristics based on Reinforcement Learning, Balanced Heuristic Selection and Group Decision Acceptance

Autores: Valdivino Alexandre de Santiago Júnior, Ender Ozcan, Vinicius Renan de Carvalho

Motivação do artigo: Existem muitos estudos sobre hiper-heurísticas de seleção baseadas em Low-Level Heuristics (LLHs) de perturbação, principalmente usando uma busca de uma única solução para otimização mono-objetivo. No entanto, o uso de métodos baseados em população é uma área de pesquisa em crescimento, onde hiper-heurísticas para otimização multiobjetivo são empregadas para controlar componentes de Algoritmos Multiobjetivo Evolutivos (MOEAs), por exemplo operadores de mutação e crossover usados como LLHs, ou as LLHs sendo MOEAs completos. Apesar desses esforços e mesmo que existam várias propostas de seleção de LLHs e métodos de aceitação de população, as hiper-heurísticas geralmente consistem em um único método de seleção de LLH, sem políticas adicionais, combinado com uma abordagem de aceitação de população única.

Objetivo do artigo: O artigo apresenta uma nova hiper-heurística de seleção online baseada em LLHs de perturbação que visa resolver problemas multiobjetivos complexos, via uma abordagem de seleção de heurística baseada em Aprendizado por Reforço e um mecanismo balanceado, e métodos de aceitação de população. A *Hyper-Heuristic based on Reinforcement LearnIng, Balanced Heuristic Selection and Group Decision AccEptance* (HRISE) contempla um único método de seleção de LLH mas com políticas adicionais, e três métodos de aceitação de população em um processo de tomada de decisão em grupo. A hiper-heurística considera como LLHs MOEAs completos e, de fato, os autores propõem duas variações com base na tomada de decisão em grupo: uma baseada na regra da maioria (HRISE_M) e outra baseada na regra de responsabilidade (HRISE_R). Uma terceira hiper-heurística, chamada de *Hyper-Heuristic based on Random LLH Selection and Random Choice of Move Acceptance Methods* (HRMA) também é proposta, onde uma escolha aleatória para selecionar (meta)heurísticas é realizada, combinada com uma escolha aleatória de métodos de aceitação de população.

Rápida descrição sobre as metodologias/técnicas/métodos relacionados ao artigo: O artigo propõe três novas hiper-heurísticas de seleção (HRISE_M, HRISE_R, HRMA), dois novos métodos de aceitação (QLA, MQLA), propõe um novo indicator de qualidade para algoritmos de otimização multi e many-objective, propõe um mecanismo de controle dinâmico onde o número de iterações que uma LLH selecionada irá executar varia ao longo do processo de pesquisa. Além disso, foi realizado um experimento controlado

(rigoroso) para determinar o desempenho das três novas hiper-heurísticas comparadas com seis outros algoritmos: a hiper-heurística *Choice Function* (HH-CF), uma hiper-heurísica com seleção aleatória de (meta)heurística e com aceitação de todas as populações (HH-ALL), a hiper-heurística *Learning Automata-based Hyper-Heuristic with a Ranking Scheme Initialisation* (HH-RILA), e três MOEAs quais sejam: NSGA-II, IBEA e SPEA2.

Resultados de avaliações experimentais: As hiper-heurísticas propostas tiveram o melhor desempenho quando comparadas com HH-CF, HH-ALL e as MOEAs de baixo nível, quando executadas isoladamente, em termos de hipervolume. Ainda sobre a questão do hipervolume, ambas hiper-heurísticas HRISE assim como a HRMA apresentaram um desempenho um pouco melhor do que a HH-RILA. Considerando o indicador ε, HRISE_R foi a melhor seguida por HH-RILA. No geral, considerando os dois indicadores de qualidade, a HRISE_R apresentou o melhor desempenho de todas as hiper-heurísticas.

Contexto de validação: Foram consideradas 39 instâncias de problema, sendo 24 dessas de 4 problemas teóricos, e 15 instâncias de problemas de 4 problemas do mundo real.

3.11 *Sw Performance_1*

Referência:
Título: An efficient method for uncertainty propagation in robust software performance estimation.
Autores: Aldeida Aleti, Catia Trubiani, André van Hoornc, Pooyan Jamshidi.

Motivação do artigo: A estimativa de desempenho de software em estágios iniciais de desenvolvimento é necessária, porém difícil, logo, os engenheiros responsáveis por estimar o impacto do sistema são frequentemente forçados a tomar decisões sob incerteza sem saber o impacto exato dessas decisões no desempenho. No entanto, o estudo de métodos para melhorar a propagação de incertezas na estimativa de desempenho de softwares é de grande importância para o processo.

Objetivo do artigo: O objetivo do artigo é apresentar uma nova opção para estimativa de desempenho de software, que se baseia na expansão do caos polinomial (PCE), a qual

possui um menor custo e necessidade de amostras de teste do que o método baseado em Monte Carlo.

Rápida descrição sobre as metodologias/técnicas/métodos relacionados ao artigo: Resultados retirados de benchmarks feitos em um cluster mostraram a eficiência do método PCE quando comparado ao método Monte Carlo quando se trata de amostras para teste. Assim tornando mais acessível a utilização do mesmo para uma estimativa de desempenho de software mais barata e mais rápida.

Resultados de avaliações experimentais: O método proposto "polynomial chaos expansion" (PCE) teve maior resultado baseado no método Monte Carlo, visando o gasto e número de testes necessários para possuir uma acurácia na propagação da incerteza, mostrando ser um bom método quando não se tem muito dinheiro ou até testes a serem feitos.

Contexto de validação: Após avaliar o benchmark dos 3 métodos apresentados em várias fases, o PCE produziu erros muito baixos com 100 amostras mostrando a efetividade em relação ao Monte Carlo, economizando até 225 h no processo. Também mostrou ser pelo menos 80% acurado na propagação da incerteza, tendo acurácia acima de 97% nos casos afetados por ruído.

3.12 *Sw Performance_2*

Referência:
Título: Detecting software performance problems using source code analysis techniques.
Autores: Salma Eid, Soha Makady, Manal Ismail.

Motivação do artigo: Atualmente as empresas lançam atualizações em curtos períodos de tempo para seus sistemas, sejam elas de aprimoramentos, correções de bugs ou novos recursos. Assim o desempenho do software pode ser comprometido acidentalmente após a sua nova versão em comparação à versão anterior ocasionando uma regressão no desempenho. Os desenvolvedores muitas vezes encontram dificuldades em reconhecer alterações de código que causam esse problema especialmente quando se trata de códigos muito extensos, sendo assim de suma importância o estudo de softwares para reconhecimento de regressões no desempenho.

Objetivo do artigo: o artigo busca propor uma nova abordagem para detectar as chamadas "regressões no desempenho" resultantes de atualizações feitas no software baseada na análise do código fonte utilizando a ferramenta PerfDetect para comparar o atual código e sua versão anterior.

Rápida descrição sobre as metodologias/técnicas/métodos relacionados ao artigo: o artigo usa a ferramenta PerfDetect para realizar a comparação de códigos fontes de softwares reais, comparando o codigo da versão anterior e a atual. Assim a ferramenta citada mostra quais métodos foram alterados após a atualização e compara os seus tempos de execução para julgar assim qual seria o mais leve no processo de execução do software.

Resultados de avaliações experimentais: a ferramenta PerfDetect consegue apresentar com eficácia as partes alteradas do código que sofreram com regressões de desempenho quando comparado com sua versão anterior, apresentando assim os métodos alterados, tempo de execução dos softwares comprovando a sua eficiência em comparação à outras abordagens.

Contexto de validação: Foram feitos testes em duas versões distintas de quatro aplicativos de código aberto, sendo eles: AgileFant, Apache Commons Math, Apache Commons IO e Xalan. O processo é dividido em três etapas: 1- código-fonte alterado reconhecido, 2- o código é comparado entre a versão atual e a anterior, e por último, os códigos são executados com diferentes níveis de usuários para aumentar a complexidade sendo (5, 10, 15, 20 e 25 usuários).

3.13 *Sw Performance_3*

Referência:

Título: Lightweight embedded software performance analysis method by kernel hack and its industrial field study.

Autores: Jooyoung Seo, Byoungju Choi, Sueng-wan Yang.

Motivação do artigo: Atualmente softwares embarcados estão presentes em diversas áreas. No entanto, muitas vezes possuem limites restritos para fazerem o seu trabalho ocasionando em um grande impacto quando alguma tarefa extra é executada ao mesmo tempo, como é o caso de testes de desempenho em softwares. Assim o trabalho de otimizar um software embarcado se torna muito mais difícil quando se trata da necessidade de realizalo com ferramentas de análise otimizadas e eficazes.

Objetivo do artigo: O artigo propõe uma ferramenta para análise de software embarcado que seja leve, usando o método que permite coletar dados necessários para analisar gargalos de desempenho, identificar as causas e os locais de gargalos com pouco impacto no ambiente operacional utilizando a ferramenta Analytic Master of System v2.0 para hackear o PCB e acessar as informações de maneira precisa.

Rápida descrição sobre as metodologias/técnicas/métodos relacionados ao artigo: o AMOS v2.0 é uma ferramenta desenvolvida para hackear o PCB, visando acessar as informações ali contidas de modo que tenha o mínimo de impacto no ambiente original de operação dos softwares embarcados. Assim quando estudada a otimização, pode-se obter com mais precisão e menos influência seus dados.

Resultados de avaliações experimentais: a ferramenta AMOS v2.0 conseguiu mostrar a sua eficácia ao coletar dados do PCB com maior eficiência quando comparadas com outras ferramentas de análise, as quais necessitam de ligação por cabo de rede ou conexão via Active Sync ao contrário à ferramenta apresentada. Atendendo assim com segurança aos requisitos operacionais do teste.

Contexto de validação: o método proposto foi capaz de coletar dados para analisar gargalos de desempenho e identificar locais de sobrecarga sobre desempenho, entre 6 KB em fatores de expansão de código e 11,93% em fatores de desaceleração. Mostrando assim, ter pouco impacto no sistema de teste onde está sendo executado.

3.14 *Sw Performance_4*

Referência:

Título: Performance-driven software model refactoring.

Autores: Davide Arcelli, Vittorio Cortellessa, Daniele Di Pompeo.

Motivação do artigo: A refatoração de software é uma prática destinada a resolver problemas de desempenho ou para correção de bugs não esperados em uma aplicação. Embora existam vários estudos na área de refatoração, quase todos, são destinados aos requisitos funcionais, fazendo com que a parte dos requisitos não funcionais seja crítica e de extrema importância para correções de problemas ainda não estudados.

Objetivo do artigo: o artigo apresenta uma estrutura baseada em uma ferramenta única (EPSILON) por automação que permite a refatoração de um software por detecção e remoção de antipadrão de desempenho que quando presente resulta em um projeto funcional mas ineficiente.

Rápida descrição sobre as metodologias/técnicas/métodos relacionados ao artigo: O artigo trata de um sistema o qual por meio de automação apresenta diversos caminhos e soluções para a refatoração de softwares visados nos requisitos não funcionais como: gastos de energia e velocidade. Assim apresentando diversas tabelas com dados e codigos dos experimentos e análises feitas com relação ao EPSILON utilizando os modelos UML.

Resultados de avaliações experimentais: No sistema em que o sistema foi testado (jardim botânico) os antipadrões se mostraram eficazes quando se trata de refatoração de softwares, assim como o uso da ferramenta EPSILON apresentou grande desempenho ao apresentar vários caminhos e soluções para o tema abordado. Comprovando a sua importância quando se trata da análise de software tanto para detectar como resolver problemas geralmente ocultos.

Contexto de validação: foram implementados dentro da plataforma EPSILON regras de detecção e ações de refatoração em modelos UML o qual é um conjunto de antipadrões já conhecido e usado para testes. Explorando as linguagens EPSILON foi possível verificar as propriedades aplicando então três tipos de sessões de refatoração.

4 ATIVIDADE 2: FAMILIARIZAÇÃO COM O SOFTWARE DE PRÉ-PROCESSAMENTO DE DADOS DO CPTEC

Durante o desenvolvimento do projeto, houve uma apresentação do software de pré-processamento de dados do CPTEC, realizada pelo Dr. Eduardo Batista de Moraes Barbosa.

Nessa atividade, foram identificados os obstáculos e limitações relacionados ao desempenho de tal software, que se concentram principalmente no alto volume e na disponibilidade de diferentes tipos de dados encaminhados para a realização do pré-processamento, visto que são necessários processos muitas vezes demorados para realizar a assimilação destes dados.

Foi realizado um entendimento dos logs de saída que o software de pré-processamento realiza. A Figura 2 mostra parte de um dos logs de saída.

```
2020-12-04 21:12:01.000 (I)    Prepare Conv - Start
2020-12-04 21:12:01.010 (I)    Processing 2020-12-04 18GMT
2020-12-04 21:12:01.025 (I)    Prepare airep (TAC)
2020-12-04 21:12:02.010 (I)    Prepare airep (BUFR)
2020-12-04 21:12:03.020 (I)    Data to Assimila!
2020-12-04 21:12:03.135 (I)    Prepare ascat (TAC)
…
2020-12-04 21:12:01.400 (I)    Prepare airep (TAC)
2020-12-04 21:12:02.100 (I)    Prepare airep (BUFR)
```

Figura 2 – Parte de um log de saída

Percebe-se que existem eventos (e.g. Prepare airep (TAC)) e um tempo associado ao início do evento. Portanto, por meio dos arquivos de log de saída, é possível estimar as características de desempenho relacionadas ao software de pré-processamento do CPTEC. Com isso, obteve-se um bom entendimento de como lidar com a saída do software de pré-processamento do CPTEC.

5 ATIVIDADE 3: MECANISMOS

Os mecanismos que têm sido estudados envolvem formas de leitura eficiente dos arquivos de log, extração do tempo de processamento de cada evento (baseados nos arquivos de log). Uma ferramenta está sendo desenvolvida para, precisamente, extrair os tempos gastos para cada evento relevante processado pelo software do CPTEC, para dar apoio a criação de um modelo de estados o qual seria usado como base para formular o problema de otimização. No que se refere a tecnologia, a linguagem Java está sendo utilizada para dar início a ferramenta, pensando em facilitar a integração com o framework que será usado, jMetal. A Seção 5 apresenta mais detalhes em relação ao algoritmo geral que está sendo implementado.

6 ATIVIDADE 4: ESTUDO DAS HIPER-HEURÍSTICAS E META-HEURÍSTICAS

Conforme apresentado na Seção 3, já foi concluída a revisão da literatura de diversas hiper-heurísticas e meta-heurísticas que poderiam ser usadas na pesquisa. Os seguintes algoritmos de otimização foram selecionados para o projeto:

a) Hiper-heurísticas: HH-CF, HRISE_M, HRISE_R, HH-RILA;
b) Meta-heurísticas: IBEA, SPEA2, NSGA-III, MOEAD_STM.

No entanto, ainda é necessário se aprofundar no entendimento dos algoritmos para que seu uso seja feito de forma adequada.

7 ATIVIDADE 5: PROJETO E IMPLEMENTAÇÃO DO MÉTODO

Um algoritmo foi desenvolvido para estudar o tempo gasto para processar os eventos, de acordo com os logs de saída do software do CPTEC. Esse algoritmo é descrito nos seguintes passos:

a) Para identificar os eventos (linhas que começam com data no arquivo de log; vide Figura 2), é preciso ler os arquivos de log, e serão usadas expressões regulares para isso. Toda linha que começar com um número significa que é um evento (nem todas as linhas do arquivo de log se relacionam a um evento);
b) Usar uma estrutura de dados do tipo hash table (hash map) para armazenar os eventos. As chaves da hash table serão: nomeEventoAtual – nomeProximoEvento. O valor numérico é a diferença: nomeProximoEvento – nomeEventoAtual;
c) O evento, de fato, corresponde ao nomeEventoAtual da chave. Coloca-se nomeProximoEvento para saber, exatamente, qual é o próximo evento na sequência. Uma tabela será criada identificando cada evento, unicamente, em cada arquivo de log;
d) Cada arquivo de log terá a sua tabela (derivada da hash table). Então, será obtido um tempo médio de cada evento, de acordo com todos os arquivos de log;
e) Um modelo de estados será gerado baseado nesses tempo médios;
f) Baseando-se nesse modelo de estados e em funções objetivo (e.g. minimizar tempo de processamento, …), serão gerados casos de teste que, no fundo, são sequências que ditarão a ordem em que os eventos devem ser processados pelo software do CPTEC.

A título de informação, considere a hash table a seguir:

HashTable = {
“Prepare airep (TAC) - Prepare airep (BUFR)”, 985,
“Prepare airep (BUFR) - Data to Assimila!”, 64,
“Data to Assimila! - Prepare ascat (TAC)”, 125, ...
}

Essa hash table deve ser lida assim: O evento “Prepare airep (TAC)” demorou 985 ms, o evento “Prepare airep (BUFR)” demorou 64 ms, o evento “Data to Assimila!” demorou 125 ms.

A implementação desse método já foi iniciada, onde a leitura dos arquivos de log já está bem avançada.

8 CONCLUSÃO

Com os estudos dos artigos relacionados ao assunto e as demais atividades realizadas, foi possível ter uma boa noção do desafio proposto e o conhecimento necessário para a conclusão dessa pesquisa. O tema, apesar de complexo, é necessário para realizar uma análise profunda e rigorosa no software de pré-processamento de dados desenvolvido pelo CPTEC pois, como visto nos artigos consultados, Otimização em

Teste de Software é capaz de ajudar a melhorar o desempenho do produto do CPTEC.

Além disso, a familiarização com o software do CPTEC e os mecanismos iniciais desenvolvidos demonstram um caminho promissor para o desenvolvimento da pesquisa, cujos resultados podem ajudar na atividade operacional relacionada à previsão de tempo do CPTEC.

9 REFERÊNCIAS BIBLIOGRÁFICAS

[WMO 2018] World Meteorological Organization (WMO). Guide to Meteorological Instruments and Methods of Observation. Available from: https://www.wmo.int/pages/prog/www/IMOP/CIMO-Guide.html. Access in: July 15, 2020.

[Cintra 2005] R. S. C. Cintra. PREPARAÇÃO DE DADOS DE OBSERVAÇÕES PARA O SISTEMA DE ASSIMILAÇÃO DE DADOS PSAS DO CPTEC. 2005. Technical Report, INPE, 60 p. 2005.

[Zhang e Moore 2015] Z. Zhang e J. C. Moore, Data Assimilation, In: Mathematical and Physical Fundamentals of Climate Change, Chapter 9, p. 291-311. 2015.

[Santiago Júnior et al. 2020] V. A. Santiago Júnior, E. Ozcan, V. R. de Carvalho, Hyperheuristics based on reinforcement learning, balanced heuristic selection and group decision acceptance, Applied Soft Computing (2020) 1-48. Submitted (Pre-print, Revision 1). Available from: https://bit.ly/2CA6lPR. Access in: July 15, 2020.

[Balera e Santiago Júnior 2019] J. M. Balera, V. A. Santiago Júnior, A systematic mapping addressing hyper-heuristics within search-based software testing, Information and Software Technology, 114 (2019), 176 – 189. Doi: https://doi.org/10.1016/j.infsof.2019.06.012.

[Santiago Júnior e Ozcan] V. A. Santiago Júnior, E. Ozcan. HRMA: Hyper-Heuristic Based on the Random Choice of Move Acceptance Methods. In: Proceedings of the Annual Conference of The Operational Research Society (OR61), 2019, Canterbury, United Kingdom, p. 37-38.

[Drake et al. 2019] J. H. Drake, A. Kheiri, E. Ozcan, E. K. Burke, Recent advances in selection hyper-heuristics, European Journal of Operational Research. Doi: https://doi.org/10.1016/j.ejor.2019. 07.073.

[Deb e Jain 2014] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference point-based nondominated sorting approach, part i: Solving problems with box constraints,890 IEEE Transactions on Evolutionary Computation 18 (4) (2014) 577601. doi:10.1109/TEVC.2013.2281535.

[Zitzler e Kunzli 2004] E. Zitzler, S. Kunzli, Indicator-based selection in multiobjective search, in: X. Yao, E. K.Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tino,A. Kaban, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII, Springer 770 Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 832-842.

[Harman et al. 2015] M. Harman, Y. Jia, Y. Zhang, Achievements, open problems and challenges for search based software testing, in: 2015 IEEE 8th International Conference

on Software Testing, Verification and Validation (ICST), 2015, pp. 1-12. doi:10.1109/ICST.2015.7102580.

[Junior et al. 2020] JUNIOR, V. A.S.; OZCAN, E.; CARVALHO, V. R. *Hyper-Heuristics based on Reinforcement Learning, Balanced Heuristic Selection and Group Decision Acceptance.* Applied Soft Computing, V. 97, part A, p.106760, 2020. Disponível em: https://doi.org/10.1016/j.asoc.2020.106760. Acesso em: 14 May. 2020.

[Aleti et al. 2018] Aleti, A.; Trubiani, C.; Hoornc, A. V.; Jamshidi, P. An efficient method for uncertainty propagation in robust software performance estimation. Journal of Systems and Software, V. 138, p.222-235, 2018. Disponível em: https://doi.org/10.1016/j.jss.2018.01.010. Acesso em: April. 2018.

[Eid et al. 2020] Eid, S.; Makady, S.; Ismail, M. Detecting software performance problems using source code analysis techniques. Egyptian Informatics Journal, V. 21, Issue 4, p.219-229, 2020. Disponível em: https://doi.org/10.1016/j.eij.2020.02.002. Acesso em: Dec. 2020.

[Seo et al. 2012] Seo, J.; Choi, B.; Yang, S. Lightweight embedded software performance analysis method by kernel hack and its industrial field study. Journal of Systems and Software, V. 85, Issue 1, p.28-42, 2012. Disponível em: https://doi.org/10.1016/j.jss.2011.03.049. Acesso em: Jan. 2012.

[*Arcelli* et al. 2018] *Arcelli*, D.; *Cortellessa*, V.; *Pompeo*, D., D. *Performance-driven software model refactoring*. Information and Software Technology, V. 95, p.366-397, 2018. Disponível em: https://doi.org/10.1016/j.infsof.2017.09.006. Acesso em: March. 2018.