Configuração de Algoritmos Genéticos Multiobjetivos para Otimização de Projeto de Arquitetura de Linha de Produto

Narcizo Gabriel Freitas Palioto, Thelma Elita Colanzi[1]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – Brasil

narcizo.gabriel2@gmail.com, thelma@din.uem.br

Resumo. *Algoritmos de busca têm sido explorados com sucesso na otimização de projeto de Arquitetura de Linha de Produto de Software (PLA) na abordagem seminal chamada Multi-Objective Approach for Product-Line Architecture Design (MOA4PLA). Tal abordagem produz um conjunto de alternativas de projeto de PLA que melhora os diferentes fatores otimizados. Atualmente, o algoritmo utilizado nesta abordagem é o algoritmo NSGA-II (Non-dominated Sorting Genetic Algorithm II), um algoritmo genético multiobjetivo que otimiza várias propriedades simultaneamente. Apesar de resultados experimentais promissores, estudar a melhor combinação de configuração dos parâmetros do algoritmo genético é imprescindível para obter melhores resultados. Valores de referência para os parâmetros ainda não foram definidos para otimização de projeto de PLA porque este é um tópico de pesquisa incipiente. Nesse contexto, o objetivo deste trabalho é identificar os valores mais adequados para configurar o algoritmo NSGA-II para a otimização de projetos de PLA por meio de um estudo experimental. Uma análise quantitativa baseada no indicador de qualidade hypervolume e em testes estatísticos foi realizada para determinar o valor mais adequado para configurar cada parâmetro do algoritmo.*

1. Introdução

Linha de Produto de Software (LPS) é um conjunto de técnicas e métodos de desenvolvimento que visa a maximização da qualidade e do reuso do software, a minimização dos custos através do reuso de artefatos e a entrega mais rápida do software. Uma LPS contém um conjunto de funcionalidades comuns e que satisfazem as necessidades de um segmento de mercado particular [Linden et al. 2007]. Um dos principais artefatos de uma LPS é a Arquitetura de Linha de Produto (PLA - *Product Line Architecture*). Porém, a obtenção da PLA não é uma tarefa trivial, pois fatores relacionados a diferentes propriedades arquiteturais devem ser atendidos, os quais muitas vezes são conflitantes, por exemplo modularidade de características, reusabilidade e extensibilidade da LPS [Colanzi et al. 2014].

Problemas da Engenharia de Software similares a este têm sido eficientemente resolvidos com algoritmos de busca em um campo de pesquisa conhecido como Engenharia de Software Baseada em Busca (SBSE - *Search Based Software Engineering*) [Harman and Jones 2001]. A SBSE converte um problema de engenharia de software num problema de busca computacional que pode ser resolvido com uma metaheurística.

Algoritmos genéticos são uma metaheurística muito utilizada em SBSE. Esses algoritmos são baseados na analogia com os processos de seleção natural e genética evolucionária [Goldberg 1989]. O objetivo desses algoritmos é copiar a natureza levando

em consideração a adaptação e a sobrevivência do mais forte. Eles incluem operadores de busca de três tipos: seleção, cruzamento e mutação [Goldberg 1989]. Todas as metaheurísticas, incluindo algoritmos genéticos, são algoritmos estocásticos. Logo há um fator de aleatoriedade associado à aplicação dos operadores de busca.

A MOA4PLA (*Multi-Objective Approach for Product-Line Architecture Design*) [Colanzi et al. 2014] é uma abordagem que oferece um tratamento multiobjetivo para avaliar e otimizar projetos de PLA. Essa abordagem foi implementada na ferramenta OPLA-Tool [Freire et al. 2020] utilizando o algoritmo NSGA-II (*Non-dominated Sorting Genetic Algorithm II*) [Deb et al. 2002], um algoritmo genético multiobjetivo muito eficiente e popular para resolver problemas de otimização. NSGA-II é um algoritmo multiobjetivo, pois permite a otimização simultânea de mais de um objetivo (fator). No caso específico de otimização de projetos de PLA, os objetivos são constituídos por métricas de software relacionadas a propriedades arquiteturais que se pretende otimizar.

Para utilizar o NSGA-II é preciso configurar as probabilidades de aplicação dos operadores de mutação e de cruzamento, bem como qual o tamanho da população de indivíduos e o número de gerações do processo evolutivo. Em geral, esses valores variam de acordo com o problema em questão e com características dos indivíduos fornecidos como entrada para o processo de otimização [Goldberg 1989]. Considerando que os algoritmos de busca são estocásticos, recomenda-se a execução de várias rodadas do algoritmo e o uso de indicadores de qualidade para averiguar qual a melhor combinação de valores para os parâmetros [Arcuri and Fraser 2011].

Essa situação se aplica ao uso do NSGA-II para resolver o problema de otimização de projeto de PLA. Os parâmetros de configuração desse algoritmo geralmente são: (a) probabilidade de mutação; (b) probabilidade de cruzamento; (c) tamanho da população; (d) número de gerações (ou de avaliações de *fitness*) e (e) formação da população inicial. Existem na literatura recomendações de valores para esses parâmetros [Goldberg 1989], mas estudos anteriores mostraram que os valores de referência para um algoritmo canônico não são os mais apropriados para otimização de projeto de PLA, o que justifica investigar valores especificamente para esse problema.

Considerando o contexto exposto anteriormente, o objetivo deste trabalho é investigar os valores mais apropriados para calibrar o algoritmo NSGA-II, a fim de otimizar projeto de PLA no contexto da MOA4PLA na ferramenta OPLA-Tool. Para isso, foram realizados estudos experimentais com diferentes valores para os parâmetros de configuração de um algoritmo genético e posterior comparação dos resultados usando o indicador de qualidade de algoritmos multiobjetivo *hypervolume* e testes estatísticos, a fim de obter os valores mais apropriados para resolver o referido problema. Os experimentos são definidos na Seção 4 e os resultados são apresentados na Seção 5. Os conceitos fundamentais envolvidos neste trabalho são apresentados a seguir.

2. Referencial Teórico

2.1. Algoritmos Genéticos

Algoritmos genéticos são inspirados na evolução genética [Goldberg 1989]. Esses algoritmos codificam uma potencial solução ao problema em estruturas de dados que se assemelham a cromossomos e aplicam operadores de recombinação nessas estruturas de modo que informações cruciais não sejam perdidas.

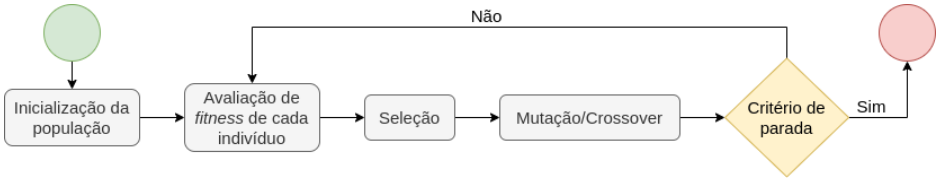

Figura 1. Passos de um algoritmo genético.

A Figura 1 explica os passos de um algoritmo genético. Uma população inicial é gerada aleatoriamente, sendo que cada indivíduo é uma possível solução. Em seguida é feito o cálculo da função de *fitness* de cada indivíduo da população. Essa função de *fitness* mede o quão adaptado um indivíduo está de acordo com o problema. A seleção escolhe os indivíduos mais aptos para continuarem para as próximas gerações e esses indivíduos passam por operadores genéticos de mutação e *crossover* com o objetivo de perpetuar seus genes. Enquanto o critério de parada não é atingido é realizada uma nova geração com os indivíduos gerados pelos operadores de mutação e de *crossover* e, caso o critério de parada seja atingido, o algoritmo se encerra. O operador de cruzamento faz uma combinação de partes de dois indivíduos selecionados para gerar novos indivíduos. Então, o operador de mutação modifica aleatoriamente um indivíduo descendente. A população descendente criada a partir dos operadores de busca substitui a população anterior.

O NSGA-II [Deb et al. 2002] é o algoritmo genético multiobjetivo mais difundido e utilizado na literatura. Ele lida com problemas multiobjetivo, aqueles que dependem de diversos fatores (objetivos) que estão em conflito. Desse modo, geralmente não há uma única solução. Assim, o algoritmo retorna diversas boas soluções que representam o *trade-off* entre os objetivos definidos. Estas soluções são chamadas de não-dominadas e formam a fronteira de Pareto.

2.2. Projeto de Arquitetura de Linha de Produto de Software Baseado em Busca

A Arquitetura de Linha de Produto (PLA) é um meio de garantir o reúso de uma LPS, pois ela envolve todos as características que são compartilhadas por todos os produtos que derivam de uma LPS [Contieri Jr et al. 2011]. Trata-se de um projeto que contém todos os componentes e características possíveis, comuns e variáveis, a todos os produtos a serem derivados da LPS [Linden et al. 2007]. Desse modo, é possível instanciar a arquitetura individual para cada produto da LPS. A importância de uma PLA é evidente, a complexidade dos softwares cresce a cada dia e o custo de desenvolvimento e manutenção deve ser contido. PLAs ajudam nesse sentido, pois funcionam como uma planta para a criação de famílias de produtos similares diminuindo drasticamente o custo e o esforço de *design* de software e o desenvolvimento de vários produtos [Linden et al. 2007].

O projeto de uma PLA modular, extensível e reusável não é uma tarefa simples. Um arquiteto precisa saber analisar os *trade-offs* necessários entre as métricas utilizadas e isso demanda um grande esforço humano.

MOA4PLA é uma abordagem baseada em SBSE e tem como foco avaliar e melhorar uma PLA otimizando propriedades arquiteturais convencionais, modularização de características e extensibilidade de LPS por meio de algoritmos multiobjetivos [Colanzi et al. 2014]. A Figura 2, extraída de [Colanzi et al. 2014], mostra as atividades realizadas pela MOA4PLA, cujas descrições são apresentadas a seguir:

- Construção da Representação de PLA (*Construction of the PLA Representation*): Recebe como entrada um arquivo XML que representa um diagrama de classes

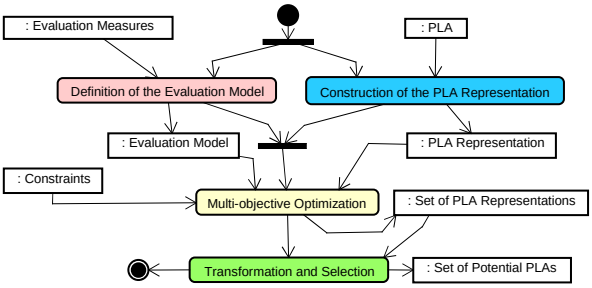

Figura 2. Atividades da MOA4PLA

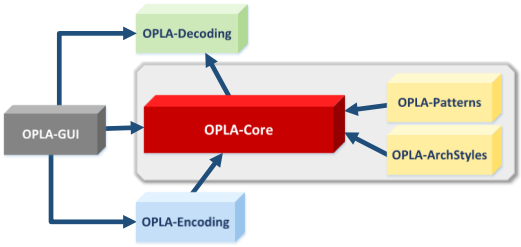

Figura 3. Módulos da OPLA-Tool.

contendo o projeto de PLA e gera uma representação computacional a partir do diagrama de classes da PLA;

- Definição do Modelo de Avaliação (*Definition of the Evaluation Model*): Escolha das métricas a serem utilizadas no processo de otimização para determinar a qualidade das soluções geradas, por exemplo, acoplamento e modularização de características;

- Otimização Multiobjetivo (*Multi-objective Optimization*): após a execução das atividades anteriores o algoritmo de busca NSGA-II gera um novo conjunto de representações de PLA que otimiza as métricas escolhidas pelo arquiteto. Este projeto está relacionado a esta atividade;

- Transformação e Seleção (*Transformation and Selection*): O conjunto de representações de PLAs geradas pela atividade anterior é convertido em diagrama de classes para ser manipulado pelo arquiteto.

A OPLA-Tool[1] (*Optimization for PLA Tool*) [Freire et al. 2020] permite a aplicação da abordagem MOA4PLA. Os módulos que compõem a OPLA-Tool estão descritos na Figura 3, extraída de [Féderle et al. 2015]. O módulo OPLA-GUI apoia o arquiteto permitindo a escolha do algoritmo de busca que deverá ser utilizado bem como seus parâmetros, funções objetivo e operadores genéticos. Esse módulo ainda permite ao arquiteto a inserção de uma PLA que é a entrada para o processo de busca. O módulo OPLA-GUI utiliza serviços dos módulos: OPLA-Encoding, OPLA-Decoding e OPLA-Core. O OPLA-Encoding é responsável por transformar uma PLA em uma representação baseada em um metamodelo predefinido. O OPLA-Core é onde está implementado o algoritmo de busca, esse módulo recebe a representação criada pelo OPLA-Encoding e realiza o processo evolutivo do algoritmo e retorna um conjunto de PLAs. Finalmente o OPLA-Decoding recebe cada representação das PLAs e decodifica para uma versão legível no Papyrus (plugin do Eclipse para modelos UML).

A Figura 4 mostra a tela de configuração de execuções de um experimento na OPLA-Tool. Vários destes parâmetros podem ser alvo dos experimentos de configuração a serem estudados no presente projeto.

3. Trabalhos Relacionados

Nesta seção apresentam-se os trabalhos identificados em uma busca em fontes eletrônicas por trabalhos sobre calibração do NSGA-II. O estudo de Arcuri e Fraser

[1]https://github.com/otimizes/OPLA-Tool

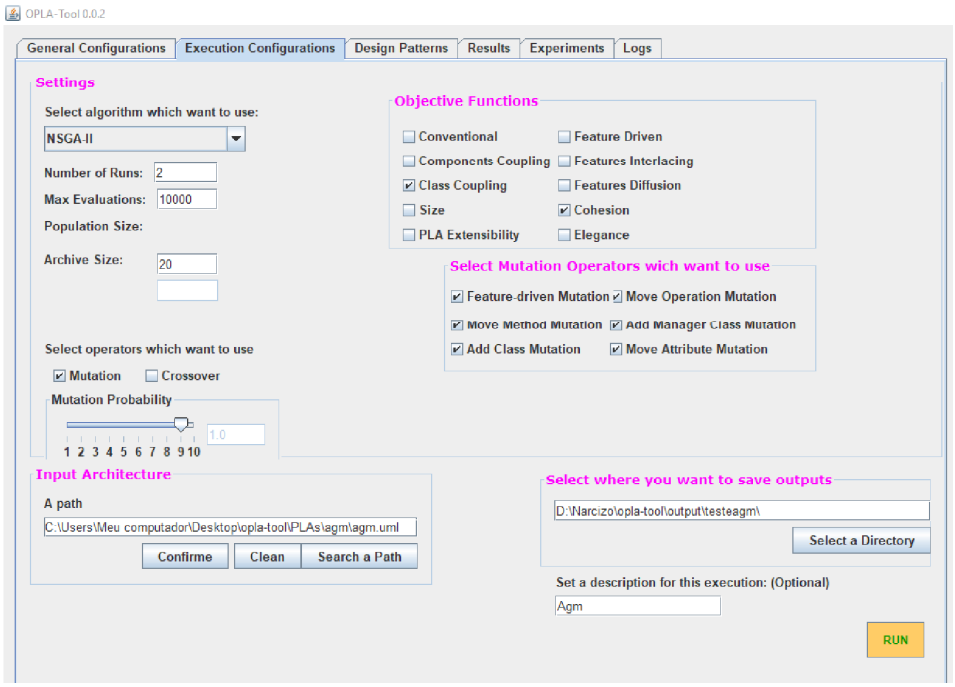

Figura 4. Interface de Configuração dos Parâmetros da OPLA-Tool.

[Arcuri and Fraser 2011], cujo foco é selecionar casos de testes para determinadas classes com o objetivo de maximizar um critério de cobertura enquanto busca-se minimizar o número de testes e sua duração, foram utilizados os parâmetros a seguir: Número de Avaliações de *Fitness* = 300.000, Tamanho da População: $\{4, 10, 50, 100, 200\}$, Probabilidade de Mutação: $\{0, 0.2, 0.5, 0.8, 1\}$.

Sayyad et al. [Sayyad et al. 2013] compararam os resultados de Arcuri e Fraser no contexto de otimização de modelos de características de LPS. Os experimentos foram realizados utilizando as seguintes configurações: o tamanho da população é formado pelos valores $\{10, 50, 100, 150, 200\}$ e a probabilidade de mutação de acordo com o conjunto: $\{0, 0.5, 1, 1.5, 2\}$. Sayyad et al. [Sayyad et al. 2013] confirmaram as afirmações de Arcuri e Fraser de que configurações de parâmetros diferentes causam uma grande variação na performance, justificando a necessidade de estudar qual a melhor configuração dos parâmetros do NSGA-II para a otimização de projeto de PLA.

No contexto de projeto de PLA, o número de avaliações de *fitness* proposto em [Colanzi et al. 2014] estipulava um valor de 30.000 com populações de 100 indivíduos e probabilidade de mutação de 0.9. Guizzo [Guizzo 2014] propôs derivar e verificar outros dois valores, um dez vezes maior (300000) e outro com um décimo do valor (3000). Os resultados mostraram que o melhor valor depende da PLA dada como entrada.

4. Descrição do Estudo Experimental

Nesta seção descreve-se os procedimentos adotados para realizar o estudo experimental a fim de identificar os melhores valores para os parâmetros do algoritmo NSGA-II. Os parâmetros a serem calibrados e os respectivos valores a serem verificados foram definidos a partir de trabalhos relacionados. Para realizar os experimentos foi utilizada a ferramenta OPLA-Tool aplicando as diferentes combinações de parâmetros com o algoritmo NSGA-II e otimizando dois projetos de PLA.

Tabela 1. Configurações dos experimentos.

ID da Configuração	Tamanho da População	Número de Avaliações de Fitness	Taxa de Mutação
config1	10	3000	0,9
config2		30000	0,9
config3		300000	0,9
config4	50	3000	0,9
config5		30000	0,9
config6		300000	0,9
config7	100	3000	0,9
config8		30000	0,9
config9		300000	0,9
config10	200	3000	0,9
config11		30000	0,9
config12		300000	0,9
config13	melhor entre os anteriores	melhor entre os anteriores	0,8

Projetos de PLA. Dois projetos de PLA foram usados no estudo experimental: *Arcade Game Maker* (AGM) [SEI 2009] e *Mobile Media* [Contieri Jr et al. 2011]. AGM é uma LPS acadêmica criada pelo *Software Engineering Institute (SEI)* que contém 3 jogos de arcade: *Brickles*, *Bowling* e *Pong*. *Mobile Media* (MM) é uma LPS para gerenciamento de mídia (música, vídeo e foto) em dispositivos móveis.

Configuração dos parâmetros. Considerando os trabalhos relacionados (Seção 3), neste trabalho decidiu-se investigar a calibração dos seguintes parâmetros: Número de Avaliações de *Fitness*; Tamanho da População e Probabilidade de Mutação. No presente trabalho foi usada a mesma estratégia proposta em [Guizzo 2014] partindo de 30000 avaliações de *fitness*. O tamanho da população variou entre: 10, 50, 100 e 200. Dois valores foram executados para probabilidade de mutação: 0,8 e 0,9.

Nesse contexto, foram realizados experimentos de configuração de parâmetros para as PLAs AGM e MM utilizando os valores para cada parâmetro conforme discriminado na Tabela 1. A fim de minimizar o número de experimentos a serem executados, optou-se por executar os doze primeiros experimentos (Config 1 a Config12) utilizando o valor 0,9. Sabendo qual o melhor valor de tamanho de população e de avaliações de *fitness*, foi então averiguado o desempenho do algoritmo com taxa de mutação de 0,8 (Config 13). Cada experimento foi executado 15 vezes, sempre com as mesmas funções objetivo: *Cohesion* (mede a coesão relacional das classes), *Feature Modularization* (mede o nível de modularização das características da PLA) e *Class Coupling* (mede o acoplamento entre classes), cujas definições podem ser obtidas em [Verdecia et al. 2017].

Critério para Avaliação dos Resultados. Foram utilizados dois critérios de avaliação sendo eles o tempo de execução e o indicador de qualidade *hypervolume*. Com relação ao critério de tempo de execução quanto menor o tempo de execução melhor o desempenho. Todos os experimentos foram executados na mesma máquina, cujo processador é um Intel Core I5 7500 com 8GB de memória RAM, para assegurar que todas configurações disponibilizavam do mesmo recurso.

O indicador de qualidade *hypervolume* foi adotado para comparar os resultados obtidos pelos experimentos. Ele consiste no volume n-dimensional entre a estimativa de Fronteira de Pareto e um ponto de referência específico [Coello et al. 2007]. Quanto maior o valor do *hypervolume*, maior a área de cobertura refletindo em uma fronteira melhor. Os dados do hipervolume foram normalizados antes de aplicar testes estatísticos

para constatar qual a melhor configuração de parâmetros. O primeiro teste estatístico executado foi o teste de Shapiro-Wilk a fim de saber se uma determinada população tem uma distribuição normal ou não de acordo com o *p-value*. Como a distribuição dos resultados foi não normal, o teste de Kruskal-Wallis foi utilizado para averiguar a diferença estatística entre os resultados obtidos com confiança de 95% (*p-value* $\leq$0.05).

Ameaças à Validade. Uma das principais ameaças à validade do nosso estudo é o uso de uma única medida para avaliar os resultados. A escolha do indicador de qualidade hypervolume foi estratégica para mitigar este risco, uma vez que este indicador avalia tanto a convergência como a diversidade de soluções geradas pelo algoritmo [Li and Yao 2019]. Outra ameaça é o tamanho da amostra utilizada no estudo. Apesar dos resultados não poderem ser generalizados, as duas PLAs utilizadas na calibração têm diferentes características, especialmente no que tange à modularização de *features*, que afeta diretamente os resultados já que a MOA4PLA tem operadores e funções objetivo sensíveis a esta propriedade arquitetural, amenizando ameaça à validade externa.

5. Resultados e Análise

Esta seção apresenta os resultados dos experimentos realizados a fim de determinar a melhor configuração de parâmetros para o NSGA-II no contexto de otimização de projeto de PLA. A Tabela 2 contém o número de soluções não dominadas encontradas pelo NSGA-II após as 15 execuções. É possível identificar que os parâmetros que mais impactam na quantidade de soluções não dominadas são o número de avaliações de *fitness* e o tamanho da população já que config11 e config12 que apresentam os maiores números em relação a esses parâmetros obtiveram o maior número dessas soluções .

Tabela 2. Número de soluções não dominadas e o tempo de execução (“d” para dias, “h” para horas e “m” para minutos) de cada configuração.

	AGM		**MM**	
Configuração	**Número de Soluções**	**Tempo de Execução**	**Número de Soluções**	**Tempo de Execução**
config1	7	1h 3m	9	1h 4m
config2	6	17h 29m	8	8h 53m
config3	4	6d 11h 45m	7	4d 9h 9m
config4	4	2h 8m	6	2h 13m
config5	4	1d 13h 18m	19	13h 10m
config6	8	6d 16h 7m	10	4d 19h 54m
config7	8	2h 44m	8	2h 6m
config8	8	17h 42m	24	14h 20m
config9	7	5d 23h 41m	21	5d 23h 32m
config10	8	5h 46m	7	22m
config11	16	16h 39m	17	15h 13m
config12	19	7d 21h 36m	27	6d 14h 57m
config13	5	2h 25m	4	47m

Configurações com o mesmo tamanho de população foram subdivididas em grupos para a aplicação do teste de Kruskal-Wallis a fim de averiguar a diferença estatística de cada grupo. A Tabela 3 apresenta a formação dos grupos das doze primeiras configurações. Os resultados de *p-value* retornados pelo teste estatístico são apresentados na Tabela 4. O boxplot de cada configuração é apresentado nas Figuras 5 e 6. Para o indicador de qualidade *hypervolume*, quanto maior a mediana melhor o resultado. Nos grupos 1 e 3 da *Mobile Media*, o *p-value* não foi significativo para atestar diferença estatística, então nesses casos foi analisado qual configuração obteve a maior mediana de *hypervolume*, sendo então considerada a melhor configuração do grupo. Logo, observando os

Tabela 3. Agrupamento para os testes estatísticos.

Grupo	Configurações
Grupo 1	{config1, config2, config3}
Grupo 2	{config4, config5, config6}
Grupo 3	{config7, config8, config9}
Grupo 4	{config10, config11, config12}
Grupo 5	Melhores configurações dos 4 primeiros grupos **AGM:** {configs 1, 4, 7 e 10} **MM:** {configs 1, 4, 9 e 10}

Tabela 4. Resultados do teste de Kruskal-Wallis

Grupo de Configurações	AGM p-value	MM p-value
Grupo 1	4.90E-04	8.02E-02
Grupo 2	2.87E-04	6.31E-02
Grupo 3	3.86E-05	6.43E-01
Grupo 4	3.45E-04	1.40E-06
Grupo 5	1.01E-06	5.24E-04

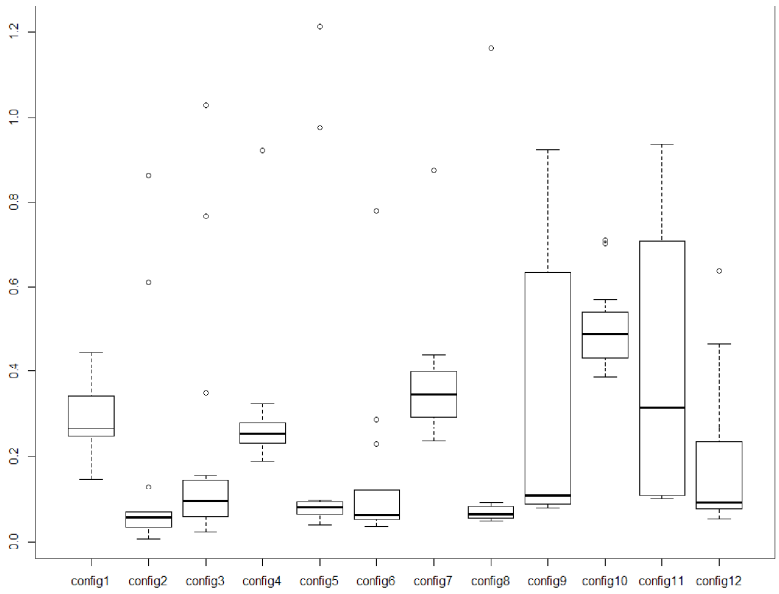

Figura 5. Boxplot referente ao teste de Kruskal-Wallis das configurações da AGM.

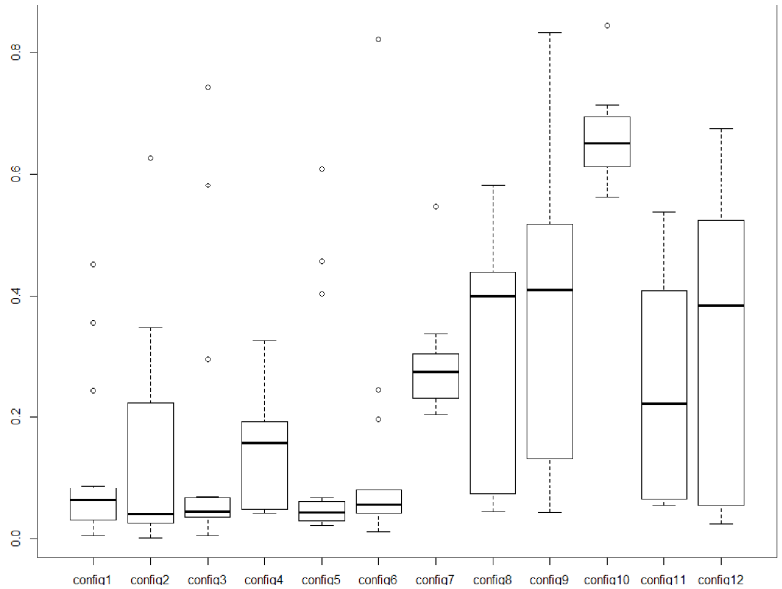

Figura 6. Boxplot referente ao teste de Kruskal-Wallis das configurações da MM.

boxplots e considerando os grupos, pode-se observar que as melhores configurações para a AGM são configs 1, 4, 7 e 10 e para a MM as configs 1, 4, 9 e 10.

Tabela 5. Resultado do teste de Kruskal-Wallis.

	Configurações	*p-value*	Melhor Configuração
AGM	config10, config13	5.20E-01	config10
MM	config10, config13	3.067E-06	config10

Tabela 6. Valores de mediana das configurações config10 e config13.

PLA	Configurações	Mediana	PLA	Configurações	Mediana
AGM	config10	0.49	**MM**	config10	0.64
	config13	0.45		config13	0.07

Após a aplicação do teste de Kruskal-Wallis nos Grupos de 1 a 4 (Tabela 4), foi possível identificar qual a melhor configuração de cada grupo e finalmente pode-se descobrir qual a melhor configuração geral aplicando o teste de Kruskal-Wallis nesse novo grupo formado (Grupo 5). Para a AGM, o Grupo 5 é formado pelas configurações 1, 4, 7 e 10 e para a *Mobile Media*, esse grupo é formado pelas configurações 1, 4, 9 e 10. Para o grupo 5 o teste estatístico atestou que há diferença estatística entre as configurações, como mostra o *p-value* (Tabela 4) e que a melhor configuração para as duas PLAs foi a config 10. Isso quer dizer que o melhor valor de tamanho de população é 200 e de número de avaliações de *fitness* é 3.000.

Como dito anteriormente após encontrada a melhor configuração (config10), ela seria testada variando o parâmetro Probabilidade de Mutação para 0.8. Deste modo, os parâmetros da config13 são: tamanho de população = 200, número de avaliações de *fitness* = 3.000 e probabilidade de mutação = 0.9. O número de soluções não dominadas e o tempo de execução da config13 pode ser visto na Tabela 2.

A Tabela 5 apresenta os resultados do teste estatístico para as configurações config10 e config13. Nesse caso, o teste de Kruskal-Wallis não apontou diferença estatística entre as configurações para a PLA AGM, porém a Tabela 6 mostra que o valor 0,9 de probabilidade de mutação (config10) é superior ao valor 0,8 (config13) em ambos os casos. No caso da AGM os valores foram equivalentes, porém config10 possui uma mediana maior e, por isso, foi considerada melhor.

A melhor configuração de parâmetros do NSGA-II para otimização de projeto de PLA é 3.000 avaliações de *fitness*, tamanho da população = 200 e probabilidade de mutação = 0,9. É interessante notar que a mesma configuração conseguiu os melhores resultados para ambas as PLAs e também foi uma das mais rápidas em termos de tempo de execução. Os parâmetros otimizados encontrados conseguem unir o melhor resultado de projeto de PLA com um tempo baixo de execução comparado com as configurações utilizadas anteriormente em [Colanzi et al. 2014] e [Guizzo 2014]; cada execução com 3.000 de avaliações de *fitness* levaram em média 1 hora para serem executadas enquanto as execuções com 30.000 levaram em média 1 dia para serem executadas.

6. Conclusão

O objetivo deste trabalho foi descobrir empiricamente as melhores configurações de parâmetros do algoritmo NSGA-II no contexto de otimização de projeto de PLA. Com base na literatura foi elaborado um conjunto de configurações de parâmetros focando em duas PLAs, a AGM e a *Mobile Media*, e dentre esse conjunto foi encontrada a configuração que encontra os melhores resultados de acordo com o indicador de qualidade *hypervolume*. Os resultados contribuem para obter melhores resultados em futuros expe-

rimentos de otimização de projetos de PLA e um menor tempo de execução. Pretende-se investigar se uma configuração com população maior pode obter melhores resultados.

Referências

[Arcuri and Fraser 2011] Arcuri, A. and Fraser, G. (2011). On parameter tuning in search based software engineering. In *Symposium on Search Based Software Engineering*, pages 33–47. Springer Berlin.

[Coello et al. 2007] Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A., et al. (2007). *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media.

[Colanzi et al. 2014] Colanzi, T. E., Vergilio, S. R., Gimenes, I., and Oizumi, W. N. (2014). A search-based approach for software product line design. In *Proceedings of the 18th International Software Product Line Conference (SPLC'14)*, volume 1, pages 237–241.

[Contieri Jr et al. 2011] Contieri Jr, A. C., Correia, G. G., Colanzi, T. E., Gimenes, I. M., Oliveira Jr, E. A., Ferrari, S., Masiero, P. C., and Garcia, A. F. (2011). Extending uml components to develop software product-line architectures: Lessons learned. In *Proceedings of the 5th European Conference on Software Architecture (ECSA)*, pages 130–138.

[Deb et al. 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[Féderle et al. 2015] Féderle, É. L., do Nascimento Ferreira, T., Colanzi, T. E., and Vergilio, S. R. (2015). OPLA-Tool: a support tool for search-based product line architecture design. In *Proc. of the 19th SPLC*, pages 370–373.

[Freire et al. 2020] Freire, W. M., Massago, M., Zavadski, A. C., Amaral, A. M. M. M., and Colanzi, T. E. (2020). OPLA-Tool v2.0: a tool for product line architecture design optimization. In *Proceedings of the 34th Brazilian Symposium on Software Engineering (SBES '20)*. ACM.

[Goldberg 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition.

[Guizzo 2014] Guizzo, G. (2014). Uso de padrões em projeto arquitetural baseado em busca de linha de produto de software. Dissertação de Mestrado, Pós-Graduação em Ciência da Computação, Universidade Federal do Paraná, Curitiba.

[Harman and Jones 2001] Harman, M. and Jones, B. F. (2001). Search-based software engineering. *Information and Software Technology*, 43(14):833 – 839.

[Li and Yao 2019] Li, M. and Yao, X. (2019). Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Comput. Surv.*, 52(2).

[Linden et al. 2007] Linden, F. J. V. d., Schmid, K., and Rommes, E. (2007). *Software product lines in action: the best industrial practice in product line engineering*. Springer Science & Business Media.

[Sayyad et al. 2013] Sayyad, A. S., Goseva-Popstojanova, K., Menzies, T., and Ammar, H. (2013). On parameter tuning in search based software engineering: A replicated empirical study. In *2013 3rd International Workshop on Replication in Empirical Software Engineering Research*, pages 84–90.

[SEI 2009] SEI (2009). Arcade game maker pedagogical product line. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=485941.

[Verdecia et al. 2017] Verdecia, Y. D., Colanzi, T. E., Vergilio, S. R., and Santos, M. C. B. (2017). An enhanced evaluation model for search-based product line architecture design. In *XX Ibero-American Conf. on Software Engineering (CIbSE2017)*, Buenos Aires.