Estabelecendo Sequências de Teste de Integração de Classes: Um Estudo Comparativo da Aplicação de Três Algoritmos Evolutivos Multiobjetivos

Wesley Klewerton Guez Assunção[1], **Thelma Elita Colanzi**[1 2], **Silvia Regina Vergilio**[1], **Aurora Trinidad Ramirez Pozo**[1]

[1]Departamento de Informática, Programa de Pós-Graduação em Informática, Universidade Federal do Paraná (UFPR)– Curitiba, PR – Brasil

[2]Departamento de Informática
Universidade Estadual de Maringá(UEM) – Maringá, PR – Brasil

{wesleyk, thelmae, silvia, aurora}@inf.ufpr.br

Resumo. *Com o intuito de minimizar os custos associados à criação de stubs, durante o teste de integração de sistemas orientados a objetos é preciso determinar a melhor ordem de integração e teste das classes. Em alguns estudos recentes, Algoritmos Evolutivos Multiobjetivos (MOEA) têm resolvido eficientemente este problema porque eles consideram vários fatores que afetam o processo de construção de stubs. Geralmente esses fatores são conflitantes e por isto, nem sempre é possível obter uma solução (ordem) única. Dentre os trabalhos publicados, os MOEAs NSGA-II e SPEA2 alcançaram bons resultados para o problema em questão. No entanto, existem outros MOEAs que não foram aplicados neste contexto e que podem melhorar o desempenho desses dois algoritmos. Com este objetivo, o presente trabalho explora o uso do algoritmo PAES neste contexto. Foi realizado um experimento com quatro sistemas reais no qual o PAES foi comparado com o NSGA-II e o SPEA2. Os resultados mostram que ele consegue superar os outros dois MOEAs no caso do sistema mais complexo.*

Abstract. *During the integration testing of object-oriented software it is necessary to determine the best class integration and test order (CITO) aiming at minimizing the costs of stubs creation. Recently, Multi-Objective Evolutionary Algorithms (MOEA) have efficiently solved this problem because they consider several factors which affect the stub creation. These factors are usually in conflict, so it is not possible to find a single solution (order). Related works showed the MOEAs NSGA-II and SPEA2 have achieved good solutions for CITO problem. Although, there are other MOEAs that were not applied in this context and they can improve the performance of NSGA-II and SPEA2. In this sense, this work explores the use of the MOEA PAES in the CITO problem. It was performed an experiment involving four real systems and PAES was compared with NSGA-II and SPEA2. The results show that PAES overcomes the other two algorithms regarding to the more complex system.*

1. Introdução

O teste de software é considerado uma atividade fundamental para o desenvolvimento de software de qualidade. Geralmente esta atividade de teste é conduzida em diferentes fases. Durante a fase de teste de integração de software orientado a objetos, é comum que

algumas classes necessitem de recursos de outras classes que ainda estão em desenvolvimento. Isso leva à necessidade de construção de *stubs* para simular recursos necessários mas não disponíveis e indispensáveis para o teste, aumentando a complexidade e o custo desta atividade.

A construção de *stubs* é um processo que aumenta o custo do teste e por isto é desejável minimizar o número de *stubs* a serem construídos. O número de *stubs* está relacionado à ordem utilizada para teste e integração das classes. Determinar a melhor ordem é um problema conhecido como CITO (*Class Integration and Test Order*) [Abdurazik and Offutt 2006], e devido à existência de ciclos de dependência entre classes, bastante comum em linguagens tais como Java [Melton and Tempero 2007], determinar essa sequência não é trivial.

A maioria dos trabalhos que visam a resolver este problema propõe estratégias baseadas em grafos [Kung et al. 1995a, Kung et al. 1995b, Tai and Daniels 1997, Traon et al. 2000, Briand and Labiche 2003], entretanto estas estratégias não consideram fatores específicos da construção de *stubs* e, em geral, encontram apenas soluções subótimas. Para contornar estas limitações, estratégias baseadas em algoritmos genéticos (AG) são mais promissoras [Briand et al. 2002a, Briand et al. 2002b]. Elas permitem o uso de diferentes medidas de acoplamento para medir a complexidade e custo dos *stubs*, tais como acoplamento de atributos e métodos. Entretanto, os melhores resultados foram alcançados com algoritmos multiobjetivo [Cabral et al. 2010], os quais apresentam uma variedade de boas soluções quando comparadas às estratégias baseadas em AG com uma agregação de funções para cálculo de *fitness*.

Um estudo recente [Pozo et al. 2011] mostrou que o algoritmo NSGA-II (*Non-dominated Sorting Genetic Algorithm*) [Deb et al. 2002], que é um algoritmo evolutivo multiobjetivo (MOEA - *Multi-Objective Evolutionary Algorithm*), se adapta melhor do que outras meta-heurísticas, tais como busca tabu e colônia de formigas para solucionar o problema CITO. Motivados por este resultado, Assunção et al [Assunção et al. 2011a, Assunção et al. 2011b] realizaram um experimento aplicando o NSGA-II juntamente com um outro algoritmo evolutivo multiobjetivo, o SPEA2 (*Strength Pareto Evolutionary Algorithm*) [Zitzler et al. 2001], obtendo resultados que atestam a eficácia da aplicação de ambos MOEAs. Dando continuidade aos trabalhos de Assunção et al e buscando melhorar ainda mais os resultados, este trabalho descreve um experimento de aplicação ao problema CITO de um outro MOEA, o algoritmo PAES (*Pareto Archived Evolution Strategy*) [Knowles and Corne 2000]. O PAES é um algoritmo multiobjetivo que usa uma estratégia de evolução um tanto diferente da estratégia adotada pelos outros dois algoritmos utilizados anteriormente. Saber qual algoritmo se comporta melhor para o problema é uma questão que só pode ser respondida através de experimentos. Por isto, este trabalho descreve os resultados de aplicação dos três MOEAs a quatro sistemas reais. O custo dos *stubs* é medido em relação a dois objetivos a serem minimizados, avaliados segundo duas medidas de acoplamento comumente usadas: número de métodos e número de atributos.

Este artigo está organizado como segue. Na Seção 2 é dado o contexto de otimização e algoritmos multiobjetivos utilizados. Na Seção 3 descreve-se como foi realizada a aplicação dos MOEA ao problema CITO. Na Seção 4 descreve-se o estudo experimental realizado cujos resultados são apresentados e analisados na Seção 5. Na Seção 6 apresenta-se um exemplo de uso das soluções geradas pelos MOEAs para o problema em

questão. Finalmente, na Seção 7 são apresentadas as conclusões do trabalho.

2. Algoritmos multiobjetivos

Problemas de otimização com duas ou mais funções objetivo são chamados de multiobjetivo. Neste tipo de problema os objetivos costumam estar em conflito e, por isso, não há uma solução única. Assim, para solucionar este tipo de problema, procura-se encontrar soluções que melhor representem o compromisso entre os objetivos. Estas soluções são chamadas de não-dominadas e formam a fronteira de Pareto [Pareto 1927]. Em muitas aplicações a busca pela fronteira de Pareto ótima é NP-difícil [Knowles et al. 2006], então o problema de otimização busca encontrar a aproximação (PFApprox) mais próxima possível da fronteira de Pareto.

Algoritmos Evolutivos Multiobjetivos têm sido usados cada vez mais para resolver problemas de otimização multiobjetivo. MOEAs são adequados para esta tarefa porque eles evoluem uma população de potenciais soluções simultaneamente, obtendo um conjunto de soluções próximas à fronteira de Pareto em uma única execução do algoritmo. Existem alguns variantes dos AGs tradicionais adaptados para problemas multiobjetivo. Cada um adota estratégias distintas para evoluir e diversificar as soluções. Como mencionado, neste trabalho os seguintes algoritmos foram avaliados: NSGA-II [Deb et al. 2002], SPEA2 [Zitzler et al. 2001] e PAES [Knowles and Corne 2000].

O NSGA-II (*Non-dominated Sorting Genetic Algorithm*) [Deb et al. 2002] ordena a população em várias fronteiras não-dominadas. A cada iteração ele utiliza as soluções não dominadas formando uma nova fronteira. Para cada solução i determina-se o número de soluções que não dominam i e o conjunto de soluções que i domina. As fronteiras representam a estratégia de elitismo adotada pelo NSGA-II. Além disso, este algoritmo usa um operador de diversidade (*crowding distance*) que ordena os indivíduos de acordo com a sua distância em relação aos vizinhos da fronteira para cada objetivo, visando garantir maior espalhamento das soluções.

O algoritmo SPEA2 (*Strength Pareto Evolutionary Algorithm*) [Zitzler et al. 2001], além de ter sua população regular, mantém um arquivo externo que armazena soluções não-dominadas encontradas a cada geração. Para cada solução que está no arquivo e na população é calculado um valor de *strength*, que é usado no cálculo do *fitness* do indivíduo e durante o processo de seleção. O valor de *strength* de uma solução i corresponde a quantidade de indivíduos j, pertencentes ao arquivo e a população, dominados por i. O tamanho do arquivo é fixo, então, caso este tamanho seja excedido, um algoritmo de agrupamento é utilizado a fim de reduzi-lo [Coello et al. 2006].

No processo evolutivo do algoritmo PAES (*Pareto Archived Evolution Strategy*) [Knowles and Corne 2000] o conceito de população é diferente das estratégias tradicionais de AGs, pois apenas uma solução é mantida em cada geração, e a criação de novos indivíduos é feita somente através da aplicação do operador de mutação. Uma vez que o algoritmo trabalha com apenas uma solução por geração, não existe motivo para a utilização do operador de cruzamento. Assim como no SPEA2, existe um arquivo externo de soluções que é populado com as soluções não dominadas encontradas durante o processo. A cada geração, o PAES cria uma nova solução filha que é comparada com a solução pai, se a solução filha domina a solução pai, a filha toma o lugar da pai e a filha

é acrescentada ao arquivo externo; se a solução filha é dominada pela pai ou por alguma solução do arquivo externo, a filha é descartada; e caso nenhuma das soluções domine a outra, a escolha da solução que vai permanecer no processo evolutivo é feita considerando a diversidade entre pai, filha e as soluções do conjunto externo. Caso o tamanho do arquivo externo exceda, é aplicada uma estratégia de diversidade (*crowded region*) sobre este conjunto de soluções, eliminando soluções similares e mantendo a exploração de um espaço de busca maior.

Portanto, apesar de se tratar de três algoritmos evolutivos a estratégia de evolução do PAES é bastante diferente das outras duas, o que pode levar a resultados diferentes na resolução do problema CITO.

3. MOEAs aplicados ao problema CITO

Para aplicar os MOEAs a fim de resolver o problema CITO é preciso encontrar uma boa representação para o problema. E esta representação influencia na implementação de todos os estágios do algoritmo. Como o problema CITO é relacionado a permutação de classes, as quais formam as ordens de teste, o cromossomo é representado por um vetor cujas posições contém inteiros que representam as classes. O tamanho do vetor é igual ao número de classes de cada sistema. Então, se cada classe é representada por um inteiro, uma solução válida para um problema de 8 classes seria $(3, 5, 4, 2, 1, 7, 6, 8)$. Neste exemplo, a primeira classe a ser testada e integrada seria a classe representada pelo número 3. Esta representação é a mesma utilizada em [Cabral et al. 2010, Assunção et al. 2011a].

Outra questão relacionada aos MOEAs é a escolha das funções objetivo. Como mencionado anteriormente, podem ser usados no problema CITO várias medidas e fatores como acoplamento, coesão e restrições de tempo. No experimento realizado no presente trabalho foram usadas duas funções baseadas nas duas medidas de acoplamento usadas na maioria dos trabalhos relacionados encontrados na literatura [Briand et al. 2002a, Briand et al. 2002b, Cabral et al. 2010]: acoplamento de atributos e de métodos.

Tais medidas foram definidas para medir as dependências entre classes acopladas em termos de número de atributos usados e número de métodos chamados que influenciam diretamente na complexidade de criação de *stubs*. Então, considerando-se que c_i e c_j são duas classes acopladas, as medidas de acoplamento são definidas como segue:

- Acoplamento de atributos (A) = Número de atributos localmente declarados em c_j quando referências ou ponteiros para uma instância de c_j aparecem na lista de argumentos de alguns métodos de c_i, como o tipo de seu valor de retorno, na lista de atributos de c_i, ou como parâmetros locais de métodos de c_i (adaptado de [Briand et al. 2002b]). Assim como no experimento realizado por [Briand et al. 2002b], no caso de herança o número de atributos herdados das classes ancestrais não foi contado.
- Acoplamento de Métodos (M) = Número de métodos (inclusive construtores) localmente declarados em c_j que são invocados por métodos de c_i (adaptado de [Briand et al. 2002b]). No caso de herança, também são contados os métodos invocados herdados das classes ancestrais.

Os dados de entrada de cada MOEA são formados por três matrizes: (i) matriz de dependência entre as classes, (ii) matriz de acoplamento de atributos (medida A) e

(iii) matriz de acoplamento de métodos (medida M). A matriz de dependência popula um vetor de duas dimensões onde linhas e colunas correspondem a identificadores das classes, na intersecção das linhas e colunas das classes dependentes é informado o tipo de dependência, sendo que a classe identificada da linha depende da classe identificada na coluna. Com base nessa matriz são definidas as restrições para cada um dos algoritmos. A restrição que deve ser considerada durante o processamento do algoritmo é não quebrar as dependências relacionadas a herança[1].

As métricas de acoplamento A e M são usadas para calcular o *fitness* de cada solução, sendo que a soma das dependências entre as classes de cada métrica corresponde a um objetivo. Durante o processo de otimização, busca-se minimizar todos os objetivos.

4. Descrição do Estudo Experimental

O estudo experimental, descrito nesta seção, envolve a aplicação dos três MOEAs descritos anteriormente, que foram implementados usando as versões disponíveis no *framework* JMetal [Durillo et al. 2010].

4.1. Sistemas Utilizados

Foram utilizados quatro sistemas reais desenvolvidos em Java na realização do estudo experimental: MyBatis, JHotDraw, BCEL e JBoss. MyBatis é um *framework* de mapeamento de classes de aplicações orientadas a objetos em base de dados relacionais. JHotDraw é um *framework* de gráficos bidimensionais para editores de desenho escritos em Java. JBoss AS é um servidor de aplicações Java. O último sistema é BCEL (*Byte Code Engineering Library*), uma biblioteca que possibilita aos seus usuários analisar, criar e manipular binários Java.

Considerando a dificuldade em obter documentação arquitetural de sistemas complexos para usar no experimento, optou-se por desenvolver um *parser* para realizar a engenharia reversa do código dos software a fim de identificar os tipos de relacionamentos existentes entre as classes. O *parser* desenvolvido teve como base o software AJATO[2] (*AspectJ and Java Assessment Tool*). Na Tabela 1 são apresentadas a versão e a quantidade de classes, dependências e linhas de código (LOC) de cada sistema.

Tabela 1. Sistemas utilizados no estudo experimental

Software	Versão	Classes	Dependências	LOC	Disponível em:
BCEL	5.0	45	289	2999	http://archive.apache.org/dist/jakarta/bcel/old/v5.0/
JBoss	6.0.0M5	150	367	8434	http://www.jboss.org/jbossas/downloads.html
JHotDraw	7.5.1	197	809	20273	http://sourceforge.net/projects/jhotdraw/
MyBatis	3.0.2	331	1271	23535	http://code.google.com/p/mybatis/downloads/list

4.2. Configuração dos parâmetros dos algoritmos

Neste experimento foram adotados os mesmos valores de parâmetros utilizados em [Assunção et al. 2011a] para o NSGA-II e o SPEA2. Como mencionado na Seção 2 o algoritmo PAES não recebe parâmetro de tamanho de população e não usa operador

[1]Diferente do trabalho de Briand et al [Briand et al. 2002b] neste trabalho dependências relacionadas a Agregação não são consideradas restrições.

[2]AJATO disponível em: http://www.teccomm.les.inf.puc-rio.br/emagno/ajato/

de cruzamento em seu processo evolutivo. Após validações empíricas, decidiu-se por adotar como taxa de mutação para o PAES o valor 1,0. No caso da taxa de mutação não era interessante adotar o mesmo valor utilizado para o NSGA-II e o SPEA2 em [Assunção et al. 2011a], já que estes dois algoritmos utilizam dois operadores evolutivos (cruzamento e mutação) e o PAES utiliza somente o operador de mutação. O número de avaliações de *fitness* foi utilizado como critério de parada e foi estabelecido em 20.000, mesmo número utilizado no experimento descrito em [Assunção et al. 2011a]. A Tabela 2 contém os valores dos parâmetros utilizados para cada um dos três algoritmos.

Tabela 2. Parâmetros para os MOEAs

Parâmetro	NSGA-II	SPEA2	PAES
Tamanho da população	300	300	-
Número de avaliações de *fitness*	20000	20000	20000
Taxa de mutação	0,02	0,02	1,00
Taxa de cruzamento	0,95	0,95	-
Tamanho do arquivo	-	250	250

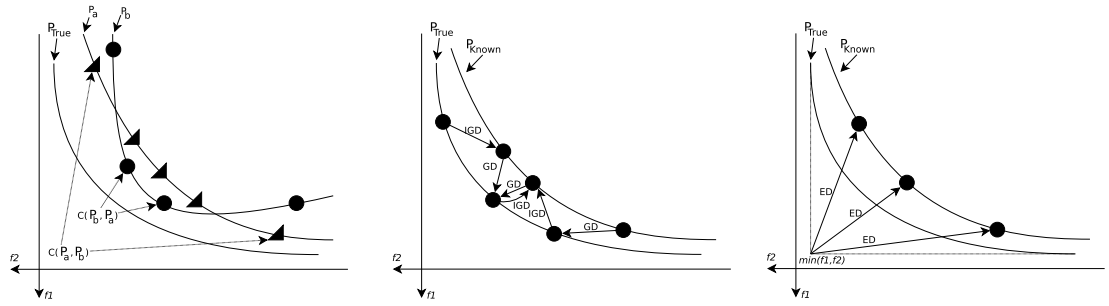
4.3. Indicadores de Qualidade

Para análise e comparação dos resultados alcançados pelos MOEAs utilizaram-se quatro indicadores de qualidade: Cobertura, Distância Geracional (GD), Distância Geracional Inversa (IGD) e Distância Euclidiana a Solução Ideal. Como os indicadores utilizam um conjunto formado pelas melhores soluções conhecidas para o problema (P_{true}), este foi obtido para cada sistema por meio da união dos conjuntos de todas as soluções encontradas por cada MOEA, eliminando-se as soluções dominadas e repetidas.

O indicador de qualidade Cobertura (C) [Knowles and Corne 2000] é usado para medir a dominância entre dois conjuntos. Comparando-se C(P_a, P_b) obtém-se um valor entre 0 e 1 referente a quanto o conjunto P_b é dominado pelo conjunto P_a, por outro lado, analisa-se C(P_b, P_a) para obter o quanto P_a é dominado por P_b. A Figura 1(a) apresenta um exemplo desse indicador, para um problema de minimização com dois objetivos. No primeiro caso analisa-se C(P_a, P_b) que retorna o valor 0.5, pois o conjunto P_b possui dois de seus quatro elementos dominados pelo conjunto P_a. No segundo caso analisa-se C(P_b, P_a) que retorna o valor 0.6 já que três dos quatro elementos do conjunto P_a são dominados pelo conjunto P_b.

O indicador GD [van Veldhuizen and Lamont 1999] é usado para calcular a distância de um conjunto de Pareto encontrado (P_{known}) em relação ao conjunto P_{true}, ou seja, é uma medida de erro na qual verifica-se o quão distante está um ponto encontrado do seu correspondente mais próximo no conjunto P_{true}. Já o indicador IGD [Radziukyniene and Zilinskas 2008] observa o inverso da GD, pois tem o objetivo de calcular a distância do conjunto P_{true} em relação ao conjunto P_{known}. Tanto para GD quanto para IGD deseja-se obter valores próximos a 0. A Figura 1(b) apresenta um exemplo das métricas de GD e IGD.

O indicador Distância Euclidiana da Solução Ideal (ED) tem como objetivo encontrar a solução que mais se aproxima dos objetivos ótimos, e consequentemente identificar o algoritmo que mais se aproximou da melhor solução possível para um determinado problema. Assim, este indicador consiste em determinar os melhores valores para cada um



(a) Cobertura (b) GD e IGD (c) Distância Euclidiana

Figura 1. Exemplos dos Indicadores de Qualidade

dos objetivos entre todas as soluções de P_{true}, encontrando um ponto considerado como uma solução ideal para o problema, então calcula-se a partir deste ponto a distância euclidiana em relação a todas as soluções de P_{known} (Figura 1(c)). Este indicador tem como base a técnica de classificação *Compromise Programming* [Cochrane and Zeleny 1973] que é usada em otimização multiobjetivo como técnica de apoio ao tomador de decisão para selecionar uma solução entre as várias encontradas.

5. Resultados e Análise dos Algoritmos

Após a execução dos três MOEAs, os resultados permitem estimar a complexidade do problema CITO para cada sistema. A Tabela 3 apresenta alguns resultados do experimento como a cardinalidade da PFApprox após todas as 30 execuções dos MOEAs (segunda coluna) e, na terceira, quarta e quinta colunas, a cardinalidade média de PFApprox e, entre parênteses, o total de diferentes soluções de PFApprox retornadas por cada MOEA. Corroborando as evidências detectadas em [Assunção et al. 2011a], (i) JBoss e BCEL se mostraram mais simples, uma vez que os três algoritmos encontraram todas as soluções em PFApprox em quase todas as execuções, e (ii) JHotDraw e Mybatis são mais complexos pois os algoritmos encontraram algumas das soluções em PFApprox em cada execução. MyBatis é ainda mais complexo do que o JHotDraw pois tem uma PFApprox com maior cardinalidade. Neste caso, o NSGA-II encontrou um maior número de diferentes soluções (53) seguido pelo PAES (49).

Tabela 3. Número de soluções não dominadas

Sistema	# PFApprox	NSGA-II	SPEA2	PAES
BCEL	29	28,70 (29)	28,57 (29)	25,13 (29)
JBoss	1	1,00 (1)	1,00 (1)	1,00 (1)
JHotDraw	3	1,80 (3)	1,73 (3)	1,97 (3)
MyBatis	49	42,53 (53)	39,97 (44)	38,47 (49)

Os três algoritmos encontraram uma única e igual solução para o sistema JBoss com os seguintes valores para cada objetivo: A = 10, M = 6. Então, esta solução envolve criar *stubs* para simular dependências de 10 atributos e 6 métodos. Apesar de alcançar uma solução final ótima, as soluções iniciais geradas pelos algoritmos multiobjetivos não são ótimas. Inicialmente os valores de *fitness* são altos e ao longo das gerações eles conseguem baixar o *fitness* até alcançar a solução supracitada.

O número de classes, dependências e ciclos influencia a cardinalidade da PFApprox obtida pelos algoritmos. Por exemplo, apesar de o JBoss ter mais classes e dependências do que o BCEL, os três MOEAs alcançaram uma única solução para ele, enquanto para o BCEL foram alcançadas 29 soluções. Portanto, o número de ciclos tem a maior influência sobre o número de soluções encontradas, já que o JBoss tem 8 ciclos e o BCEL tem 416.091 ciclos. Mas é possível notar que os dois sistemas que têm mais dependências, JHotDraw e Mybatis, são mais complexos para serem resolvidos. Entretanto, a respeito do JHotDraw, parece que as métricas não são altamente interdependentes e conflitantes já que o número de soluções alcançadas pelos MOEAs é menor, talvez porque os métodos do JHotDraw sejam menos acoplados ou mais simples que os do MyBatis.

Quando os MOEAs são usados, o testador pode usufruir da variedade de soluções encontradas de acordo com suas preferências e necessidades. O testador pode conduzir o teste de integração priorizando alguma das métricas de acoplamento e a decisão sobre a priorização de objetivos pode ser influenciada por diferentes fatores, tais como restrições do negócio, econômicas ou contratuais.

Nas próximas subseções são apresentados resultados obtidos para cada um dos quatro indicadores de qualidade analisados, além da discussão desses resultados. Para os indicadores GD, IGD e ED, não se analisam os resultados para o sistema JBoss, uma vez que os MOEAs obtiveram uma única solução para este sistema. Após obter os resultados dos indicadores GD e IGD, foi executado o teste de Friedman [Gárcia et al. 2009], através do software R, de modo a verificar se os MOEAs são considerados estatisticamente iguais ($p-value$ maior que o nível de significância α = 0,05). O teste de Friedman pode ser usado para comparar os resultados de três ou mais amostras relacionadas. Para o indicador C foi executado o teste pareado de Wilcoxon [Gárcia et al. 2009], também através do software R, de modo a verificar se os algoritmos são considerados estatisticamente diferentes ($p - value <= 0,05$).

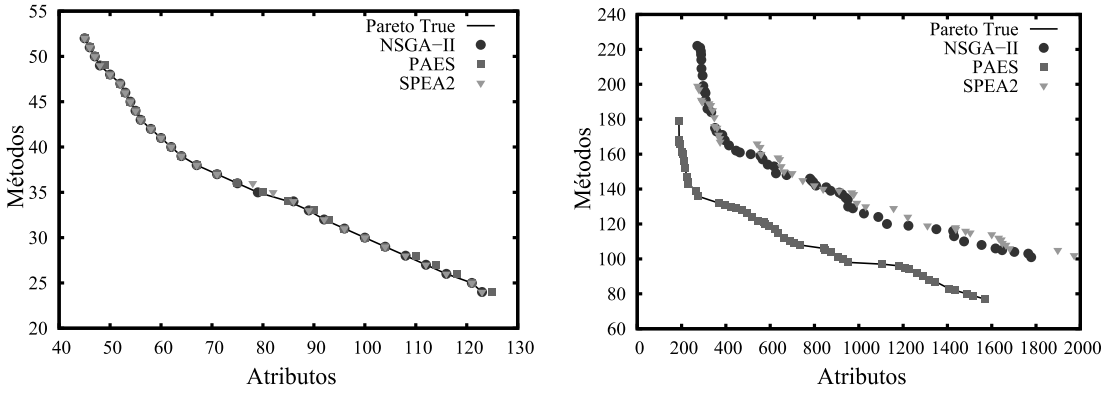
5.1. Cobertura

A Tabela 4 apresenta os valores do indicador C do conjunto final de soluções encontradas por cada algoritmo. A leitura é feita por linha-coluna, portanto o MOEA de determinada linha exerce uma determinada dominância sobre o MOEA da coluna. Para o sistema BCEL os três algoritmos são equivalentes, já para os sistemas JHotDraw e MyBatis o PAES domina em 100% tanto o NSGA-II como o SPEA2. Além disso, nestes dois sistemas o NSGA-II domina o SPEA2 com mais de 65% de dominância. Esses resultados são confirmados ao analisar a distribuição das soluções alcançadas pelos algoritmos no espaço de busca. Na Figura 2(a) fica evidente a equivalência entre os MOEAs no caso do BCEL; e, na Figura 2(b) é perceptível que o PAES alcança soluções que dominam as soluções dos outros dois MOEAs, e que o NSGA-II têm várias soluções que dominam soluções do SPEA2.

A Tabela 5 apresenta os valores de $p - values$ da comparação do indicador de C entre as 30 rodadas de cada MOEA utilizando o teste de Wilcoxon. Aparecem destacados em negrito os valores significativos, menores que 0,05, da comparação par a par entre os algoritmos. Portanto, com base neste teste estatístico, pode-se concluir que para o sistema JHotDraw os três algoritmos são equivalentes já que os $p-values$ não apontam diferença estatística entre eles. Por outro lado, no caso dos sistemas BCEL e MyBatis há diferença

Tabela 4. Análise de cobertura dos algoritmos

Sistema	MOEA	NSGA-II	SPEA2	PAES
BCEL	NSGA-II	-	0,0689655	0,275862
	SPEA2	0	-	0,241379
	PAES	0,0344828	0,103448	-
JHotDraw	NSGA-II	-	0,666667	0
	SPEA2	0	-	0
	PAES	1	1	-
MyBatis	NSGA-II	-	0,727273	0
	SPEA2	0,301887	-	0
	PAES	1	1	-



(a) BCEL (b) MyBatis

Figura 2. Soluções no espaço de busca

significativa entre os algoritmos. A análise dos *boxplots* (Figura 3) permite identificar qual deles é o melhor. Como o indicador C analisa a cobertura de um conjunto sobre o outro, então o algoritmo que tiver um maior valor de cobertura pode ser considerado melhor. No caso do BCEL, o NSGA-II e o SPEA2 superam o PAES neste indicador (Figuras 3(a) e 3(b)). Em relação ao MyBatis, nas Figuras 3(c) e 3(d) é perceptível a dominância absoluta do PAES sobre os outros dois algoritmos.

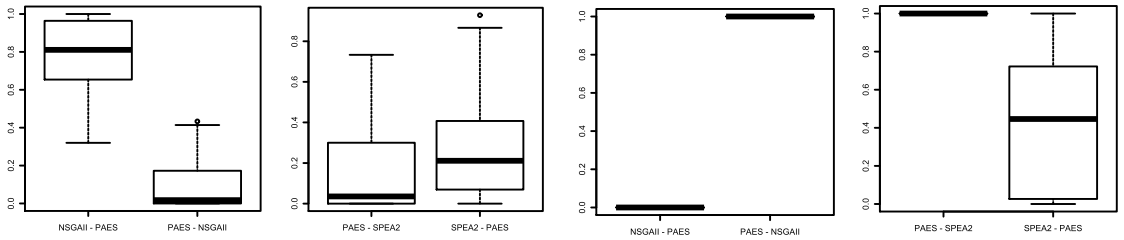
Tabela 5. $P - values$ **retornados para o indicador C**

	NSGA-II x SPEA2	NSGA-II x PAES	SPEA2 x PAES
BCEL	0,09879	**4,562e-11**	**0,04647**
JHotDraw	0,07083	0,3737	0,5279
MyBatis	0,3097	**1,685e-14**	**1,887e-10**

5.2. Distância Geracional (GD) e Distância Geracional Inversa (IGD)

Para os cálculos dos indicadores GD e IGD são utilizados: um conjunto de Pareto Conhecido (P_{known}) para cada algoritmo, formado pelas soluções encontradas pelo MOEA nas 30 rodadas, eliminando-se repetidas e dominadas; e um conjunto único chamado Pareto Real (P_{true}) que é formado por todas as soluções encontradas pelos 3 algoritmos, nas 30 rodadas, eliminando-se repetidas e dominadas.

Pela Tabela 6 pode-se observar os resultados dos algoritmos para o indicador GD. O NSGA-II obteve melhor média de GD para os sistemas BCEL e JHotDraw, enquanto



(a) BCEL (b) BCEL (c) MyBatis (d) MyBatis

Figura 3. *Boxplots* para o indicador C

o PAES obteve melhor média de GD para o sistema MyBatis. O teste de Friedman retornou os $p - values$: 2.41615449499253e-06 para o BCEL, 0.0564904201243948 para o JHotDraw, e 7.35295806145384e-11 para o MyBatis, indicando que existe diferença significativa entre os algoritmos somente para os sistemas BCEL e MyBatis.

No caso do indicador IGD, os resultados foram semelhantes ao do GD: O NSGA-II obteve o melhor GD para os sistemas BCEL e JHotDraw e o PAES para o MyBatis, conforme Tabela 6. Mas, assim como no indicador GD, o teste de Friedman só atestou a diferença significativa entre os algoritmos para os sistemas BCEL e MyBatis. Os $p - values$ retornados são: 3.26990952923229e-07 para o BCEL, 0.0564904201243948 para o JHotDraw e 1.13411309337498e-10 para o MyBatis.

Tabela 6. Média e Desvio Padrão dos indicadores GD e IGD

Indicador	Sistema	NSGA-II		SPEA2		PAES	
		Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
GD	BCEL	0,00456987	0,00210061	0,00562624	0,00238220	0,01033035	0,00412745
	JHotDraw	1,05060163	2,06375473	1,06093700	2,16080866	3,04802903	4,01798222
	MyBatis	0,04827702	0,01292301	0,05158728	0,01275482	0,01656948	0,00826484
IGD	BCEL	0,00457060	0,00220206	0,00569144	0,00259717	0,01187230	0,00526975
	JHotDraw	1,04954454	2,25400178	1,08284691	2,38655655	3,39360395	4,58535582
	MyBatis	0,05124519	0,01045320	0,05537378	0,01089798	0,01923484	0,00955634

A análise dos *boxplots* gerados pelo R e os resultados retornados pelo teste de Friedman indicam que, no caso do BCEL, não há diferença estatística entre o NSGA-II e o SPEA2 e que os dois são melhores que o PAES tanto nos indicadores GD como IGD. Já em relação ao MyBatis, a situação é inversa, o PAES alcança o melhor desempenho nestes dois indicadores e não há diferença entre os outros dois MOEAs.

5.3. Distância Euclidiana a Solução Ideal (ED)

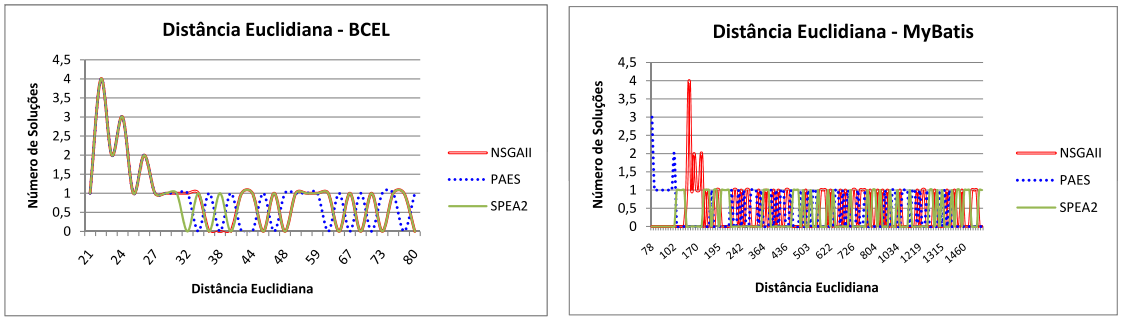
Para o cálculo da ED o custo da solução ideal para cada sistema é apresentado na segunda coluna da Tabela 7. Esta tabela também apresenta a menor distância encontrada e seu respectivo *fitness* para os sistemas usados no experimento. A menor distância encontrada está destacada em negrito. O PAES alcançou as soluções que tem a menor ED a partir das soluções ideais para o JHotDraw e o MyBatis. Para os outros dois sistemas os três MOEAs alcançaram a mesma ED.

Os gráficos da Figura 4 mostram o número de soluções encontradas por cada um dos MOEAs em relação a ED. No caso do BCEL (Figura 4(a)) é possível notar que não

Tabela 7. Custo da Solução Ideal e menor ED

Sistema	Custo da Solução Ideal	MOEA	Menor ED	*Fitness* da solução de menor ED
BCEL	45, 24	NSGA-II	**21,954498**	**56, 43**
		SPEA2	**21,954498**	**56, 43**
		PAES	**21,954498**	**56, 43**
JBoss	10, 6	NSGA-II	**0**	**10, 6**
		SPEA2	**0**	**10, 6**
		PAES	**0**	**10, 6**
JHotDraw	27, 10	NSGA-II	3,162278	28, 13
		SPEA2	3,162278	28, 13
		PAES	**1,414214**	**28, 11**
MyBatis	188, 77	NSGA-II	164,268074	298, 199
		SPEA2	148,121572	272, 199
		PAES	**78,230429**	**230, 143**

há diferença entre os algoritmos em termos de distribuição das soluções. Já no caso do MyBatis (Figura 4(b)), o PAES tem soluções mais perto da solução ideal do que os outros dois, seguido pelo SPEA2 que alcança duas soluções de ED baixa. No caso do JBoss e do JHotDraw foram encontradas 1 e 3 soluções, respectivamente, inviabilizando a análise de distribuição das soluções de acordo com a ED. Sendo assim, os três algoritmos têm um desempenho bem parecido para o sistema BCEL e o PAES alcança os melhores resultados para o sistema MyBatis neste indicador de qualidade (Tabela 7).



(a) BCEL (b) MyBatis

Figura 4. Distância Euclidiana das Soluções Encontradas pelos MOEAs

5.4. Discussão dos resultados

Após a apresentação dos resultados dos quatro indicadores de qualidade é possível analisá-los. Os três MOEAs alcançaram uma única solução para o sistema JBoss, indicando que neste caso os objetivos não estão em conflito. Apesar de ser um sistema complexo os três algoritmos também se mostraram equivalentes para o sistema JHotDraw considerando os resultados dos quatro indicadores.

Em relação ao BCEL, o NSGA-II alcançou os melhores valores de GD e IGD e teve resultados de cobertura equivalentes aos do SPEA2. Em termos de ED, os três algoritmos tiveram a mesma distribuição de soluções e alcançaram a mesma distância. Ainda assim pode-se apontar o NSGA-II como o melhor para este caso já que teve melhor desempenho em dois indicadores e dominou as soluções do PAES em relação à cobertura.

No sistema MyBatis, o mais complexo, o PAES alcançou os melhores resultados em todos os indicadores de qualidade e sua superioridade foi atestada estatisticamente. Sendo assim, há evidências de que o PAES se ajusta melhor para tratar problemas multi-objetivos bastante complexos.

6. Exemplo de Uso

O uso de MOEAs para resolver o problema CITO é factível e eficaz como relatado acima; e boas ordens de teste puderam ser obtidas automaticamente por meio deles. O testador pode escolher entre elas de acordo com algum objetivo que ele precise priorizar ou outros fatores associados ao desenvolvimento de software. Para ilustrar isso, na Tabela 8 são apresentados os custos de algumas soluções encontradas pelos MOEAs para o MyBatis.

Nessa tabela, a solução a tem a menor ED e a solução b é uma alternativa da solução a uma vez que tem a segunda menor ED. Mas, como este resultado pode ser utilizado pelo testador a fim de escolher entre a e b? Se a solução a do PAES fosse escolhida para teste do sistema MyBatis (Tabela 8), seria preciso criar *stubs* para simular dependências de 230 atributos e 143 métodos. Por outro lado, se a solução b do mesmo algoritmo fosse escolhida seria preciso criar *stubs* para simular dependências de 223 atributos e 147 métodos. Desta forma, no primeiro caso, o objetivo priorizado seria a medida M e, no segundo caso, a medida A. As soluções a e b têm o melhor *trade-off* entre os objetivos e, por isso, possuem menores EDs do que as soluções que contêm valores menores (porém extremos) para as medidas A ou M.

Tabela 8. Custo de algumas soluções próximas a solução ideal para o MyBatis

	NSGA-II			SPEA2			PAES			Solução
	A	**M**	**ED**	**A**	**M**	**ED**	**A**	**M**	**ED**	**Ideal**
a	298	199	164,268	272	199	148,122	230	143	78,230	188, 77
b	304	196	166,184	279	197	150,602	223	147	78,262	

Já que os MOEAs encontraram uma única solução para o sistema JBoss, o testador não precisa estabelecer qualquer critério para escolher a ordem de integração e teste de classes porque a melhor sequência é conhecida (solução gerada pelos algoritmos).

Apesar de neste estudo a análise e exemplos considerarem três MOEAs, na vida real o testador provavelmente terá resultados de um único algoritmo. A partir de todas as soluções obtidas, o testador poderá decidir qual objetivo priorizar e escolher a ordem de teste de acordo com esta prioridade. A decisão de priorização de objetivos pode ser influenciada por diversos fatores, tais como restrições econômicas e de negócios.

7. Conclusões

Neste trabalho foi analisado o emprego de três MOEAs (NSGA-II, SPEA2 e PAES) para resolver o problema CITO visando a minimização de dois objetivos que representam fatores que influenciam na criação de *stubs* durante o teste de integração de classes.

Foi realizado um experimento envolvendo quatro sistemas reais: BCEL, JBoss, JHotDraw e MyBatis. Três destes sistemas são exemplos característicos do problema CITO, pois contêm objetivos conflitantes e, portanto, os algoritmos encontraram um conjunto de diferentes alternativas que representam uma solução de compromisso entre os

objetivos. Apesar disso, o sistema JBoss é um caso particular onde isso não ocorre, e por esta razão os três algoritmos encontraram uma única solução.

Os resultados do experimento mostraram que os MOEAs são eficazes para resolver o problema em questão e se mostraram equivalentes quando da análise dos indicadores de qualidade para alguns sistemas. Entretanto, o algoritmo PAES apresentou o melhor desempenho para o sistema mais complexo utilizado no experimento. Isso fornece evidências de que talvez a sua estratégia de evolução seja mais eficaz do que as estratégias adotadas pelos outros dois MOEAs quando existem objetivos muito conflitantes no sistema a ser testado. Apesar disso, como vários fatores podem influenciar o custo de construção de *stubs* durante a fase de teste de integração, é importante analisar o comportamento deste algoritmo quando se utilizam mais de dois objetivos.

Neste contexto, pretende-se realizar novos experimentos utilizando esses três MOEAs com um número maior de objetivos, e, possivelmente utilizando outros sistemas, a fim de confirmar as evidências identificadas neste trabalho.

8. Agradecimentos

Os autores agradecem a Edison Klafke Fillus por sua contribuição no experimento. Este trabalho foi apoiado pela Fundação Araucária, CAPES/Reuni e CNPq.

Referências

Abdurazik, A. and Offutt, J. (2006). Coupling-based Class Integration and Test Order. In *International Workshop on Automation of Software Test*, pages 50–56, Shanghai, China. ACM.

Assunção, W. K. G., Colanzi, T. E., Pozo, A. T. R., and Vergilio, S. R. (2011a). Establishing integration test orders of classes with several coupling measures. In *Genetic and Evolutionary Computation Conference*. To appear.

Assunção, W. K. G., Colanzi, T. E., Pozo, A. T. R., and Vergilio, S. R. (2011b). Reduzindo o custo do teste de integração com algoritmos evolutivos multiobjetivos e diferentes medidas de acoplamento. In *VIII Encontro Nacional de Inteligência Artificial*. Submitted.

Briand, L. and Labiche, Y. (2003). An investigation of graph-based class integration test order strategies. *IEEE Transactions on Software Engineering*, 29(7):594–607.

Briand, L. C., Feng, J., and Labiche, Y. (2002a). *Experimenting with Genetic Algorithms and Coupling Measures to Devise Optimal Integration Test Orders*. Carleton University, Technical Report SCE-02-03.

Briand, L. C., Feng, J., and Labiche, Y. (2002b). Using genetic algorithms and coupling measures to devise optimal integration test orders. In *14th International Conference on Software Engineering and Knowledge Engineering*, Ischia, Italy.

Cabral, R., Pozo, A., and Vergilio, S. (2010). A Pareto Ant Colony Algorithm Applied to the Class Integration and Test Order Problem. In *22nd IFIP International Conference on Testing Software and Systems (ICTSS'10)*. Springer.

Cochrane, J. and Zeleny, M. (1973). *Multiple Criteria Decision Making*. University of South Carolina Press, Columbia.

Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6(2):182 –197.

Durillo, J., Nebro, A., and Alba, E. (2010). The jMetal framework for multi-objective optimization: Design and architecture. In *2010 IEEE Congress on Evolutionary Computation(CEC)*, pages 4138–4325, Barcelona, Spain.

Gárcia, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics*, 15(6):617–644.

Knowles, J., Thiele, L., and Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland. Revised version.

Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8:149–172.

Kung, D., Gao, J., Hsia, P., Toyoshima, Y., and Chen, C. (1995a). A test strategy for object-oriented programs. In *19th International Computer Software and Applications Conference*. IEEE Computer Society.

Kung, D. C., Gao, J., Hsia, P., Lin, J., and Toyoshima, Y. (1995b). Class firewal, test order and regression testing of object-oriented programs. *Journal of Object-Oriented Program*, 8(2).

Melton, H. and Tempero, E. (2007). An empirical study of cycles among classes in Java. *Empirical Software Engineering*, 12:389–415.

Pareto, V. (1927). *Manuel D'Economie Politique*. Ams Press, Paris.

Pozo, A., Bertoldi, G., Árias, J., Cabral, R., and Vergilio, S. (2011). Multi-objective optimization algorithms applied to the class integration and test order problem. *Software Tools for Technology Transfer*. Submitted.

Radziukyniene, I. and Zilinskas, A. (2008). Evolutionary Methods for Multi-Objective Portfolio Optimization. In *Proceedings of the World Congress on Engineering 2008 Vol II*.

Tai, K.-C. and Daniels, F. J. (1997). Test order for inter-class integration testing of object-oriented software. In *21st International Computer Software and Applications Conference*, pages 602–607. IEEE Computer Society.

Traon, Y. L., Jéron, T., Jézéquel, J.-M., and Morel, P. (2000). Efficient object-oriented integration and regression testing. *IEEE Transactions on Reliability*, pages 12–25.

van Veldhuizen, D. A. and Lamont, G. B. (1999). Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, SAC'99, pages 351–357, New York, NY, USA. ACM.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland.