UMA PROPOSTA DE OTIMIZAÇÃO PARA SELEÇÃO DE CASOS DE TESTES PARA AUTOMAÇÃO

Pedro Henrique Pereira
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 - 60.740-903 – Fortaleza - CE
pedrohenripereira@gmail.com

Camila Loiola Brito Maia
Serviço Federal de Processamento de Dados (SERPRO)
Av. Pontes Vieira, 832 - 60.130-240 – Fortaleza - CE
camila.maia@serpro.gov.br

Jerffeson Teixeira. de Souza
Universidade Estadual do Ceará (UECE)
Av. Paranjana, 1700 - 60.740-903 – Fortaleza - CE
jeff@larces.uece.br

RESUMO

A automação de teste de software é influenciada por fatores como: manutenção dos scripts de teste, aprendizagem da ferramenta pela equipe, quantidade de pontos de verificação, entre outros. É preciso encontrar um meio termo na relação custo-benefício ao se escolher quais casos de teste serão automatizados. O trabalho apresenta critérios a serem avaliados no momento da seleção dos casos de teste para a automação, a fim de se maximizar a qualidade da suíte de teste selecionada, fazendo uso de uma implementação de Algoritmo Genético. Ao avaliar as soluções segundo o tempo de execução da suíte de teste, percebeu-se uma tendência das suítes a se aproximarem à soma de todos os casos de testes que compõem o cenário. Já ao se avaliar as soluções segundo a função objetivo definida ao longo do trabalho, percebeu-se que as suítes selecionadas apresentaram melhor desempenho em comparação com suítes selecionadas sem qualquer critério de avaliação.

PALAVRAS-CHAVE. Seleção de casos de teste. Automação de casos de teste. Algoritmo Genético.

Área principal: Metaheurísticas.

ABSTRACT

The automation of software testing is influenced by factors such as: test scripts maintenance, learning tool for team test, number of checkpoints, and others. So, it's necessary looking for mean values when choosing which test cases will be automated. The paper presents some criteria to be evaluated at the time of selection of test cases for automation, in order to maximize the quality of the test suite selected by making use of a Genetic Algorithm implementation. When evaluating the solutions according to the execution time, it's noticed a trend of suites to approach the sum of all the software test cases. Since when evaluating the solutions according to objective function defined in this paper, it was noticed that the select suites have better performance compared with suites selected without any evaluation criterion.

KEYWORDS. Selection test cases. Automation test cases. Genetic Algorithm.

Main area: Metaheuristics.

1. Introdução

Diante da crescente complexidade dos softwares produzidos, surgiu a necessidade de implementar processos capazes de aumentar a qualidade do que está sendo desenvolvido. Para Dias e Menna (2008), uma ferramenta utilizada na qualificação do software é a implementação de testes, uma maneira de verificar erros ou situações onde o software possa se comportar de maneira indesejada.

Há duas maneiras de executar a atividade de testar um software: de forma manual e de forma automatizada. Na forma manual, há a figura do testador, executando os passos definidos pelo caso de teste. Já na forma automatizada, há a utilização de ferramentas de teste que simulam usuários ou processos, de forma a não empregar procedimentos manuais.

Segundo Bartié (2002), a automação é desejada por diversos fatores, inclusive em termos de custos finais. À medida que os testes vão sendo reexecutados, os ganhos de tempo, de controle e de confiabilidade destacam a vantagem inerente a esse processo.

Mas a decisão de quais casos de testes devem ser automatizados ainda é algo meramente humano, podendo sofrer influências de impressões pessoais e fatores que não são analisados no momento da tomada de tal decisão. Entre os fatores que podem influenciar na automação dos casos de testes, podem-se citar o tempo disponível para os testes, a complexidade dos mesmos, a quantidade de pontos de verificação, a volatilidade do requisito, entre outros. Caso a decisão do que automatizar não leve em consideração os diversos aspectos inerentes à atividade, a automação dos casos de teste escolhidos pode acabar tornando-se algo negativo para o processo de desenvolvimento como um todo.

Assim, o problema de selecionar casos de testes para automação se mostra de difícil solução, apresentando um espaço de busca grande e crescente, à medida que mais dados são utilizados nessa decisão. Por isso, o uso de técnicas de otimização metaheurísticas nesse contexto se mostra adequado. As metaheurísticas são algoritmos genéricos que usam de métodos heurísticos para problemas de otimização combinatória. Poucas modificações são necessárias para adaptar uma metaheurística para um problema.

Segundo Glover (1986), apesar do uso de tais algoritmos não garantir a determinação da solução ótima do problema, as metaheurísticas retornam soluções de boa qualidade em um tempo aceitável.

O uso de técnicas de otimização, especialmente metaheurísticas, já foi aplicado em outras áreas da Engenharia de Software. A conciliação entre técnicas de otimização e Engenharia de Software ficou conhecida como Otimização em Engenharia de Software, adaptada do inglês *Search-Based Software Engineering*, termo criado por Harman (2001), apesar de já existirem registros de estudos na área desde 1992.

Neste trabalho, será empregada a metaheurística conhecida como Algoritmo Genético para apresentar soluções para o problema de seleção de casos de testes para automação. A metaheurística usada será apresentada de forma introdutória mais a frente. Os trabalhos relacionados já existentes na literatura apresentam seleção de casos de testes para execução manual, priorização de casos de testes, seleção de casos de testes de regressão baseados em riscos, entre outros. Até então, não se identificou na literatura uma proposta de seleção de casos de testes para automação.

O trabalho está descrito da seguinte forma. Alguns trabalhos relacionados são mostrados na Seção 2. A Seção 3 descreve a metaheurística utilizada no experimento, Algoritmo Genético. A Seção 4 apresenta o problema e seus aspectos. A formulação matemática é apresentada na Seção 5. A Seção 6 descreve os testes realizados, e a Seção 7 descreve as conclusões obtidas a partir dos testes. Na Seção 8, estão as considerações finais do presente trabalho e sugestões de trabalhos futuros.

2. Trabalhos relacionados

No trabalho produzido por Mansour, Bahsson e Baradhi (2001), foram utilizados 8 (oito) critérios no processo de seleção dos casos de teste: número de casos de teste, tempo de

execução, precisão, inclusividade (o quanto a técnica seleciona casos que cobrem falhas na nova versão), processamento de requisitos, tipo de manutenção, nível de teste e tipo de abordagem. Neste trabalho, os autores empregaram cinco algoritmos de seleção: têmpera simulada (Kirkpatrick, Gelatt e Vecchi, 1983), *reduction* (Harrold, Gupta e Soffa, 1993), *slicing* (Agrawal, Horgan, Krauser e London, 1993), *dataflow* (Gupta, Harrold e Soffa, 1996 *firewall*), (Mansour, Bahsson e Baradhi, 2001). Os resultados mostraram que as técnicas apresentam resultados melhores dependendo do critério.

No trabalho de Kim e Porter (2002), são apresentados resultados iniciais de um estudo empírico sobre o uso de dados históricos de execução de teste para priorizar a seleção de casos de teste em um processo restrito de teste de regressão. Os resultados experimentais relatados pelos autores apoiam fortemente o princípio de que o teste de regressão pode ter que ser feito de forma diferente em ambientes restritos e não restritos.

Apesar dos avanços, os problemas já estudados podem ser reformulados, aplicando-se outras metaheurísticas, além de haver ainda problemas não atacados, como a seleção de casos de teste para automação, foco deste trabalho.

3. Algoritmo Genético

Algoritmos Genéticos são métodos de otimização e busca inspirados nos mecanismos de evolução de populações de seres vivos, que foram introduzidos por John Holland (Holland, 1975). Esses algoritmos seguem o princípio da seleção natural e sobrevivência do mais apto, declarado em 1859 pelo naturalista e fisiologista inglês Charles Darwin em seu livro "A Origem das Espécies".

Os Algoritmos Genéticos usam um vocabulário emprestado da genética natural. Fala-se sobre indivíduos (genótipos) de uma população, estes indivíduos também são chamados de cromossomos.

Nessa técnica, uma população inicial de soluções é gerada e a partir desta são aplicados operadores bioinspirados chamados de crossover (combinação de duas soluções) e mutação (mudança aleatória em soluções) por um passo finito de vezes até chegarmos a uma população final. Nesta última população é feita uma busca pela solução de melhor função de avaliação (Figura 1)

```
Algoritmo AG()
1  inicio
2      t ← 0;
3      Gera_Populacao_InicialP(t);
4      avalia P(t);
5      enquanto Critério de parada não satisfeito
6          t ← t + 1;
7          P' ← Selecione pais de P(t − 1);
8          P'' ← Cruzamento(P');
9          P'' ← Mutação(P'');
10         avalia P'';
11         Atualiza P(t) a partir de P(t − 1) e P'';
12     fim enquanto;
13 fim.
```

Figura 1 – Pseudo-código de Algoritmo Genético

Para Silva (2008), a ideia dos algoritmos genéticos é que novos pontos de busca sejam definidos, através da combinação de soluções bem sucedidas das populações anteriores, como ocorre na natureza. A taxa de mutação visa explorar novas regiões do espaço de busca e evitar convergência prematura.

4. Definição do problema

O problema consiste em selecionar entre os elementos de um determinado conjunto de casos de testes aqueles que trarão maior benefício ao serem automatizados, levando em consideração alguns fatores. A abordagem proposta leva em consideração a quantidade de

execuções de cada caso de teste e a importância do caso de teste para o cliente.

Para realizar a seleção dos casos de teste para automação, os objetivos a serem alcançados pela técnica de otimização são:

a) Selecionar um subconjunto com os casos de teste de maior recorrência, ou seja, os casos de testes que são executados mais vezes;

b) Selecionar um subconjunto com os casos de teste que cubram os casos de uso de maior importância para o negócio do cliente.

Para a abordagem proposta, esses dois objetivos terão importâncias, ou seja, pesos diferentes.

Além disso, a soma do tempo de execução manual dos casos de testes selecionados não deve exceder um determinado percentual do somatório do tempo de execução de todos os casos de testes existentes. Caso esta restrição não fosse definida, seria mais prático para o analista de teste selecionar sempre os casos de teste de maior tempo de execução, o que, a princípio, não se mostra uma estratégia eficiente.

Outro ponto a ser levado em consideração é a precedência entre os casos de testes, que deve ser respeitada, ou seja, se existir um caso de teste que possui um precedente, este deve ser automatizado, a fim de garantir um processo integralmente automatizado.

5. Formulação do problema

No problema procuramos uma suíte de testes onde todos os casos de testes e suas precedências possam ser automatizadas. Esta suíte deve apresentar os casos de testes com maior número de previsão de execuções, assim como os casos de testes que cobrem os casos de uso que apresentam as maiores importâncias para o cliente. O tempo de execução dos casos de testes também deve obedecer a uma restrição de tempo pré-determinada.

5.1. Aspectos considerados

a) tempo de execução: tempo que um determinado caso de teste leva para ser executado manualmente.

b) recorrência de execução: é a característica que um caso de teste apresenta de ser executado mais de uma vez durante o ciclo de desenvolvimento de um software.

c) precedência entre casos de teste: para executar um determinado caso de teste, outro caso de teste deve ser executado antes. Automatizar um caso de teste e não automatizar as suas dependências acaba por descaracterizar o processo de automação, já que este terá casos automatizados e manuais, com intervenções humanas. Para esta abordagem, assume-se que cada caso de teste possui, no máximo, uma precedência.

d) importância do caso de uso: cada funcionalidade apresenta um grau de relevância ou importância diferente para o cliente. De acordo com a importância que se dá, é interessante que esta funcionalidade receba um cuidado maior no momento dos testes, reduzindo a sua margem de falhas.

5.2. Notação

A formulação matemática é a seguinte:

(OS) = **(OS1)** x α + **(OS2)** x β. Onde:

(OS1) *Max* $|previsaoExecucaoSuite_{ST}|$, onde
$previsaoExecucaoSuite_{ST} = \sum(quantidadeExecucoes_j)$, onde $0 <= j <= M$.

(OS2) *Max* $importanciaSuite_{ST}$, onde
$importanciaSuite_{ST} = \sum(importanciaCaso_j)$/M, onde $0 <= j <= M$.

Sujeito a:

(RS1) $tempoExecucaoST < \sum(tempoExecucao_k)\ x\ percentualLimiteTempo,$ onde $1 <= k <= N$

(RS2) $\forall\ t_{j1}$ e $t_{j2} \in C$, $((precedencia_{tj2} = t_{j1})$ e $(ST_{j2} = 1)) \rightarrow (ST_{j1} = 1)$.

Onde temos a notação:

a) α e β: pesos atribuídos a cada um dos objetivos, a fim de compor um único valor para o objetivo, atendendo à natureza mono-objetiva da metaheurística.

b) M: tamanho da suíte selecionada como solução

c) N: o tamanho do conjunto de todos os casos de testes do sistema

d) previsaoExecucaoSuite$_{ST}$: somatório da quantidade prevista de execuções de todos os casos de testes que compõem a suíte.

e) importanciaSuite$_{ST:}$ a importância da suíte *ST* corresponde à média aritmética das importâncias de todos os casos de testes que compõem a suíte.

f) importanciaCaso$_j$: importância do requisito coberto pelo caso de teste *j*. Quão maior for a importância do requisito coberto pelo caso de teste *j* para o cliente, maior será a probabilidade do caso de teste ser automatizado.

g) tempoExecucaoST: o somatório dos tempos de execução manual de cada caso de teste que compõe a suíte.

h) percentualLimiteTempo: corresponde a um percentual da soma dos tempos de execução manual de todos os casos de teste. Este valor servirá para determinar a tolerância máxima de tempo de execução de uma suíte de teste válida.

i) precedencia$_{tj}$: caso de teste precedente do caso de teste *j*.

6. Testes

Para o processo de seleção dos casos de testes para automação aplicou-se a metaheurística Algoritmo Genético. A implementação da metaheurística ficou a cargo do *framework* EasyMeta, desenvolvido por Carmo (2008).

Para a parametrização do algoritmo, definiram-se as taxas de mutação, *crossover* e os pontos de corte. A taxa de mutação é o percentual dos indivíduos a serem modificados na população e foi definida como 0.2%. A taxa de *crossover* é de 0.5, tendo como único ponto de corte (ou *crossover*) o ponto central da maior suíte de testes, entre as duas suítes a serem combinadas, ou seja, o número de casos de testes da suíte gerada como solução, dividido por dois.

Para a execução do algoritmo, faz-se necessário o uso de um arquivo de texto [1]com as configurações dos elementos que comporão possíveis soluções e algumas restrições. O arquivo de entrada para o algoritmo contém um conjunto de 100 casos de testes que irão compor as suítes de teste da população de entrada para a metaheurística.

Cada caso de teste apresenta um identificador sequencial, para fins de controle; o tempo de execução do caso de teste; um caso de teste precedente, caso haja; a quantidade prevista de execuções do caso de teste e a importância do mesmo para o cliente.

A modelagem levou em consideração dois objetivos: tempo de execução da suíte de teste e a importância dos casos de testes para o cliente. A cada objetivo foi atribuído um peso, baseado em conversas com profissionais da área de teste de software. Para o tempo de execução da suíte, ou seja, associado ao coeficiente alfa da formulação, atribuiu-se peso 3 (três) e à importância da suíte para o cliente, associada ao coeficiente beta, peso 1 (um). Somando-se esses dois produtos, determina-se a função objetivo, segundo a qual a metaheurística avaliará as

[1] URL para arquivo de configuração:
https://docs.google.com/document/d/1h47JZgwSSrb6N3qV0LFZwrv5gDTVvfYsq_tPLXfnV30/edit

soluções geradas.

A restrição de tempo máximo de execução da solução, para que esta seja considerada uma solução válida, é informada no arquivo de configuração como um percentual sobre o somatório de todos os casos de testes informados. A importância da suíte é baseada na soma das importâncias dos casos de teste que a compõem.

6.1. Instâncias

Para a execução dos experimentos foram geradas inicialmente 100 soluções válidas, de modo aleatório, que compuseram a população inicial. Estas soluções foram submetidas ao Algoritmo Genético, a fim de que este gerasse melhores soluções a partir da população inicial.

As instâncias, que modelavam suítes de teste, apresentavam-se como vetores binários, onde cada posição estava associada a um determinado caso de teste. Cada caso de testes apresentava um identificador e um precedente, além das informações: tempo de execução, quantidade de execuções e importância. Estas três últimas grandezas poderiam variar de 0 a 100.

Neste trabalho, considera-se como uma execução o ciclo de 10 iterações do Algoritmo Genético, valor este também parametrizado no arquivo de configuração da metaheurística. Ao final de cada execução, uma solução que esteja entre as melhores soluções geradas, segundo a função objetivo, é selecionada (Figura 2).

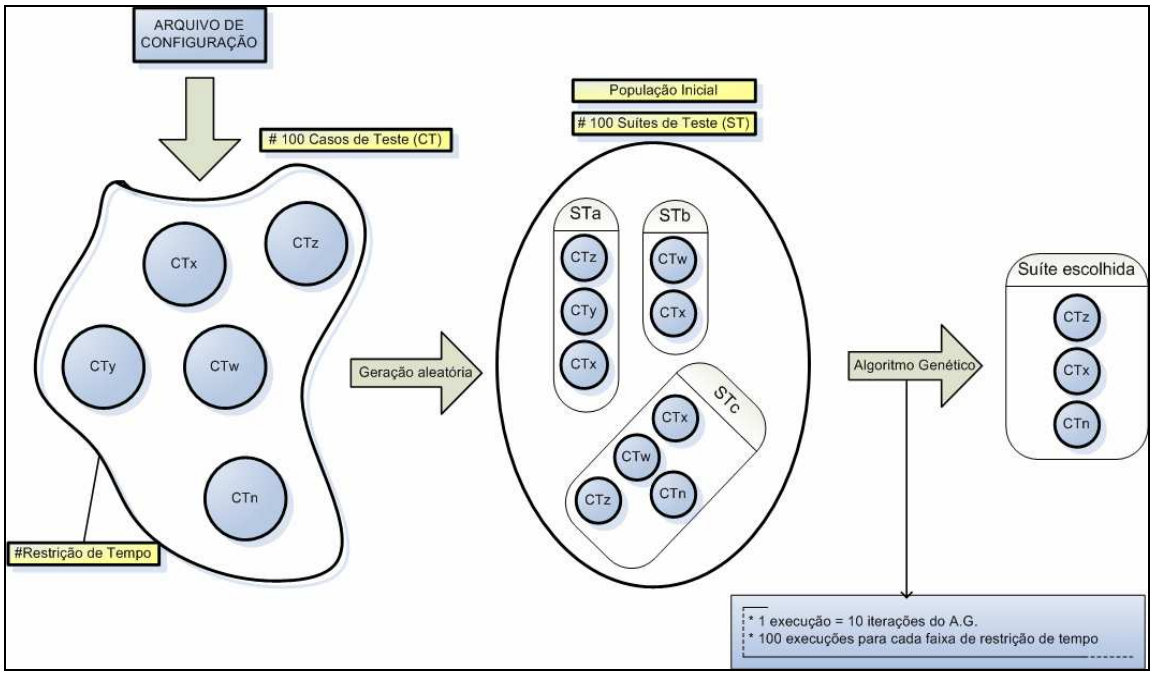

Figura 2 – Esquema da abordagem proposta

Antes de iniciar as iterações do Algoritmo Genético, uma solução é gerada e selecionada aleatoriamente para cada faixa de restrição de tempo e suas informações são armazenadas, para que, posteriormente possam ser comparadas com a solução fornecida pela metaheurística, ao final de todas as execuções.

6.2. Indicadores utilizados

No experimento, foram utilizados como indicadores a média aritmética e o desvio padrão, aplicados sobre os valores da função de avaliação da solução e o tempo de execução da suíte de teste gerada como solução.

6.3. Experimentos

A cada execução do algoritmo, coletaram-se os valores da função objetivo e do tempo de execução manual da suíte de teste selecionada como solução pela metaheurística. Este procedimento foi executado 100 vezes para cada faixa de restrição de tempo máximo de execução da solução. A faixa de restrição de tempo máximo de execução da solução variou de 10% a 90%, com variações de 10% em 10%.

Concluídas as 100 execuções, calculou-se a média dos tempos de execução manual das suítes e das avaliações da função objetivo das soluções selecionadas pela metaheurística. Estes valores foram consolidados em tabela, recuperando-se as informações das soluções aleatórias, geradas e selecionadas antes das execuções do Algoritmo Genético.

O resultado gerado pelo experimento pode ser visualizado nas Tabelas 1 e 2.

Tabela 1 – Tempo médio e desvio padrão do tempo das suítes selecionadas

Restrição de tempo	Tempo médio - Metaheurística	Tempo médio - Randômica	Desvio padrão – Metaheurística	Desvio padrão – Randômica
10%	472,44	237,00	17,92	104,47
20%	965,75	835,00	19,63	166,59
30%	1461,31	927,00	19,17	115,11
40%	1950,58	1724,00	23,28	93,43
50%	2442,42	2324,00	27,84	97,59
60%	2932,71	2932,00	31,36	56,04
70%	3417,19	3450,00	38,97	45,91
80%	3917,64	3940,00	34,04	33,73
90%	4414,34	4433,00	24,97	28,50

Tabela 2 - Função objetivo e desvio padrão das soluções

Restrição de tempo	Função objetivo - Metaheurística	Função objetivo - Randômica	Desvio padrão - Metaheurística	Desvio padrão - Randômica
10%	2124,88	384,00	134,10	402,53
20%	3758,47	1819,00	228,95	565,06
30%	5270,62	2515,00	229,53	569,37
40%	6733,03	3884,00	272,63	511,37
50%	8182,51	5721,00	245,64	618,52
60%	9636,78	7503,00	213,03	503,88
70%	11052,83	9449,00	213,11	450,51
80%	12399,69	11258,00	152,89	460,24
90%	13636,30	13126,00	98,57	308,76

Neste trabalho também foram coletadas as informações ‘Tamanho da suíte selecionada’, ou seja, a quantidade de casos de testes cobertos pela suíte e ‘Tempo de execução

do algoritmo', ou seja, o tempo que se levou para gerar a suíte e selecioná-la como solução, em milissegundos.

Estas informações adicionais foram coletadas tanto para a seleção via Algoritmo Genético, como pela seleção aleatória e tiveram os seus valores consolidados nas Tabelas 3 e 4, respectivamente.

Tabela 3 - Tamanho Médio da Suíte e desvio padrão das soluções

Restrição de tempo	Tamanho da suíte - Metaheurística	Tamanho da suíte - Randômica	Desvio padrão - Metaheurística	Desvio padrão - Randômica
10%	13,07	4,00	1,33	2,42
20%	22,77	15,00	1,56	3,61
30%	32,50	21,00	1,55	2,99
40%	41,89	33,00	1,88	2,84
50%	51,78	44,00	2,02	3,23
60%	60,85	52,00	1,95	2,64
70%	70,51	68,00	1,79	1,97
80%	80,57	76,00	1,64	2,06
90%	90,32	90,00	1,07	1,37

Tabela 4 - Tempo médio de execução do algoritmo e desvio padrão das soluções

Restrição de tempo	Tempo algoritmo - Metaheurística	Tempo algoritmo - Randômica	Desvio padrão - Metaheurística	Desvio padrão - Randômica
10%	876,82	16,00	202,69	2,69
20%	2986,79	16,00	572,22	5,17
30%	5477,40	16,00	1145,05	6,51
40%	7752,19	16,00	1529,93	7,81
50%	10012,22	31,00	1979,01	7,31
60%	11500,41	32,00	1950,97	7,79
70%	12095,76	63,00	1912,39	9,67
80%	12725,02	125,00	1027,81	17,23
90%	18822,36	282,00	802,72	34,17

Para facilitar a visualização do comportamento das soluções geradas ao longo das execuções do algoritmo, de acordo com a variação da restrição de tempo máximo de execução da solução, geraram-se gráficos baseados nos Tempos de execução da suíte de testes (Figura 3) e nas Avaliações da função objetivo (Figura 4).

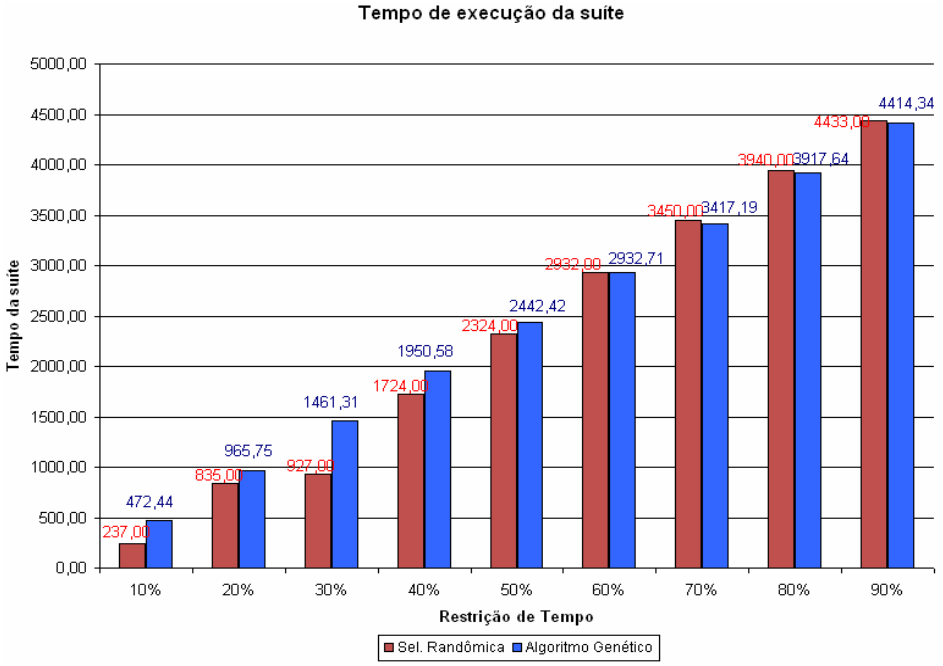

Figura 3 – Tempo de execução das suítes de teste

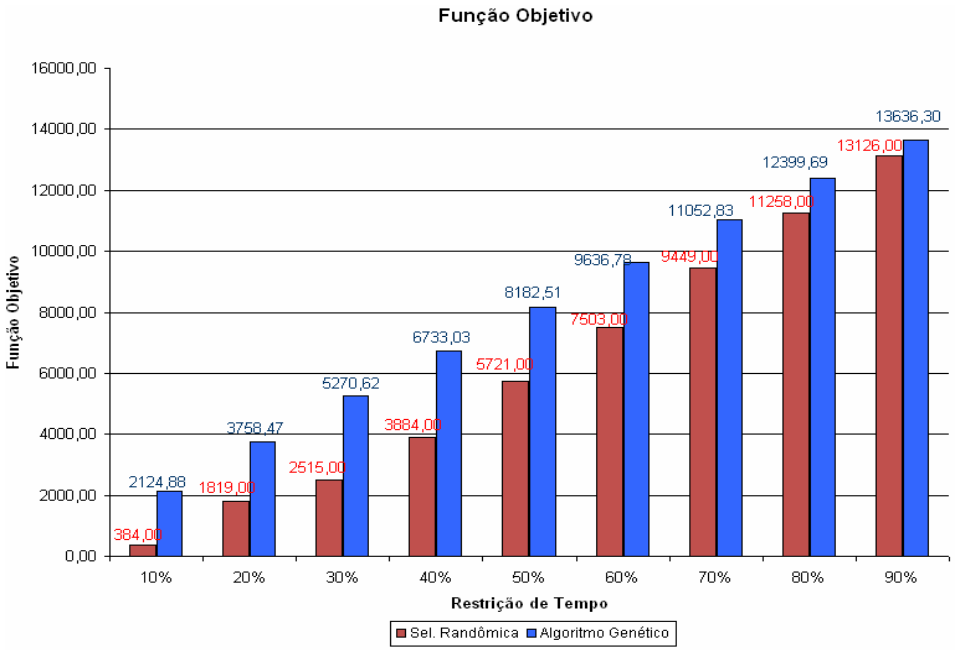

Figura 4 – Avaliação da função objetivo das soluções selecionadas

O Tamanho médio das suítes selecionadas e o Tempo médio de execução dos algoritmos podem ser visualizados nas Figuras 5 e 6, respectivamente.

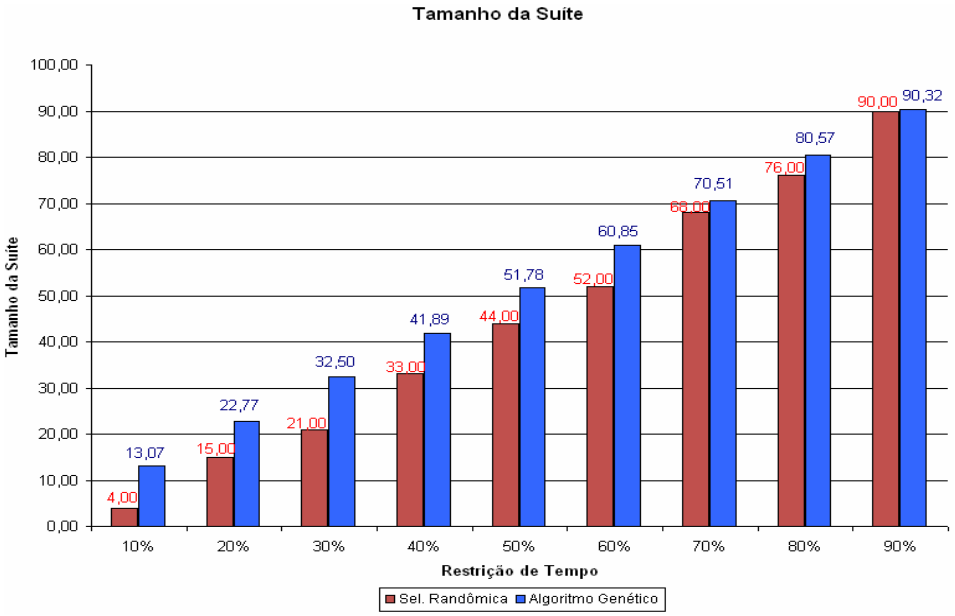

Figura 5 - Tamanho médio das suítes selecionadas

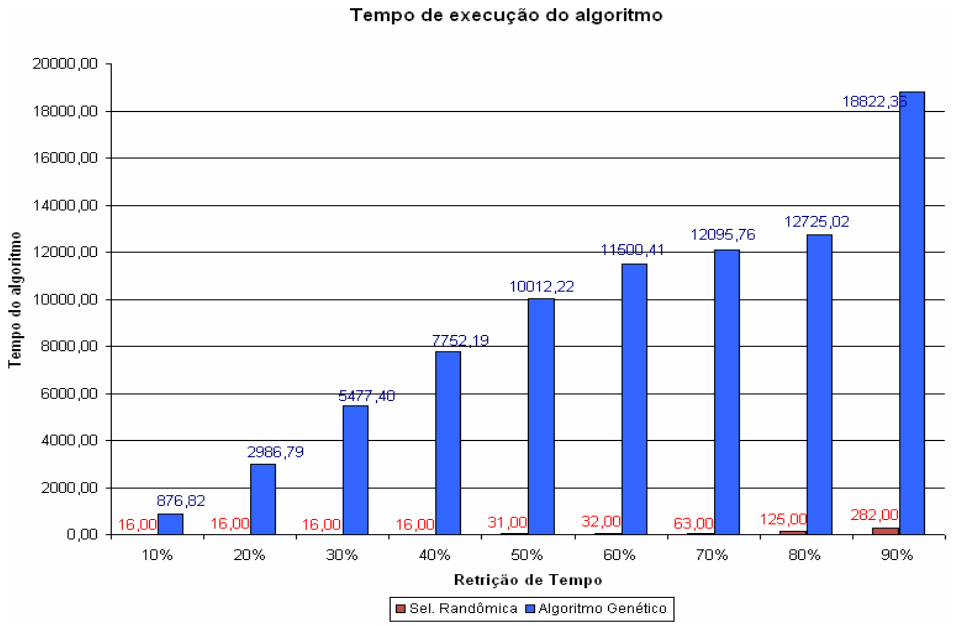

Figura 6 - Tempo médio de execução do algoritmo para seleção

7. Conclusões sobre os testes

A partir do experimento realizado, analisaram-se dois aspectos: o tempo de execução dos casos de testes que compõem a suíte, e a avaliação da mesma, segundo a função objetivo da modelagem. Além disso, informações referentes à cobertura da suíte e ao tempo de execução do algoritmo também foram coletadas.

Em relação ao tempo de execução das suítes de testes, percebeu-se que as suítes selecionadas pela metaheurística eram, em pouco, superiores às suítes selecionadas aleatoriamente. Notou-se também a tendência dos dois tipos de soluções a se aproximarem, conforme a restrição de tempo era ampliada. Isto se deve ao fato de, conforme a restrição de

tempo era ampliada, mais casos de testes passaram a compor a solução, convergindo para que a restrição se torne o somatório de todos os casos de teste, tanto para a metaheurística como para o algoritmo randômico Porém, os dois algoritmos foram implementados de forma a considerar o máximo de tempo de execução disponível, então os dois tentaram "ocupar" todo o tempo disponível com casos de teste.

As soluções selecionadas pela abordagem proposta foram melhor avaliadas em relação às soluções selecionadas aleatoriamente, segundo a função objetivo da modelagem. Em média, as soluções da metaheurística foram 1.94 vezes superiores às soluções aleatórias.

Quanto à cobertura da suíte de teste, ou seja, a quantidade de casos de teste selecionados em cada uma das abordagens, nota-se que as soluções selecionadas de acordo com as restrições propostas neste trabalho apresentam um maior número de casos de teste selecionados para automação do que as soluções selecionadas aleatoriamente, segundo as mesmas restrições de tempo, em média 1.45 vezes. Ou seja, por mais que os dois algoritmos "ocupassem" o tempo máximo disponível para testes, o Algoritmo Genético conseguiu selecionar uma quantidade maior de casos de teste que o algoritmo randômico, além de conseguir uma função objetivo com valor melhor.

O Algoritmo Genético, porém, teve uma desvantagem em relação ao randômico. No que diz respeito ao desempenho da abordagem proposta, a execução do algoritmo consome um tempo maior quando comparado a uma seleção aleatória. Em média, o tempo gasto pela metaheurística foi 234 vezes maior que o tempo de uma seleção aleatória. Isto se justifica pelo fato de o algoritmo fazer uso de procedimentos e análises não realizados pelo algoritmo randômico.

Esta diferença de desempenho pode ser absorvida pela equipe de testes, já que esta seleção seria feita poucas vezes durante o projeto de desenvolvimento do software. Uma vez fechado o escopo de quais casos de teste seriam automatizados, dificilmente este mudaria; passar-se-ia, então, para o custo de implementar os *scripts* de automação e dá-los manutenção, atividades estas bem mais frequentes.

A suíte gerada pelo Algoritmo Genético apresenta, então, cobertura e qualidade maiores do que casos de testes selecionados randomicamente.

Para uma equipe de testes, estes pontos são de grande relevância, já que, como dito anteriormente, o processo de automação apresenta um custo e um impacto no processo de testes, ou seja, automatizar sem critérios pode acarretar em mais prejuízo, em vez de benefícios.

Assim, de acordo com os números gerados, nota-se que a decisão de levar em consideração a quantidade de execuções do caso de teste, assim como a importância do mesmo para o usuário, influencia diretamente na qualidade da suíte que será gerada para automação.

8. Considerações finais

No processo de automação de casos de teste, na maioria dos casos, automatizar todos os casos pode não ser viável ou pode ter um alto custo. Portanto, faz-se interessante uma maneira de selecionar os casos de teste que passarão pelo processo de automação, de modo a diminuir o tempo de execução dos testes.

Este trabalho propõe uma forma de selecionar esses casos de teste para automação, baseados em informações mais precisas que o julgamento do testador, e isento de vícios de teste.

Uma primeira sugestão de trabalho futuro é analisar a seleção de casos de teste para automação segundo outros aspectos que não sejam o a quantidade de execuções dos casos de teste da suíte e a importância para o cliente. Entres estes aspectos, podem-se citar: custo de manutenção dos *scripts* de teste, complexidade da funcionalidade a ser testada, casos de testes com mais de uma precedência, entre outros.

Um segundo trabalho futuro seria rever a implementação do algoritmo aleatório, no qual foram localizados alguns pontos de melhoria. Outra sugestão seria tornar a abordagem multi-objetiva, fazendo uso assim de outras metaheurísticas, como, por exemplo, um Algoritmo de colônia de formigas multi-objetivo.

Referências

AGRAWAL, H.; HORGAN, J. R.; KRAUSER, E. W.; LONDON, S. **Incremental regression testing**, Conference on Software Maintenance, 1993.

BARTIÉ, Alexandre. **Garantia da qualidade de software: adquirindo maturidade organizacional**. Rio de Janeiro. Editora Campus, 2002.

CARMO, Rafael Augusto Ferreira. **EasyMeta: um framework para problemas de otimização mono-objetivo**. Monografia –Centro de Ciências e Tecnologia, Universidade Estadual do Ceará. Fortaleza – CE, 2008.

DIAS, Leonardo Cardoso. MENNA, Romulo da Silva. **Teste de desempenho a partir de modelos UML para componentes de software**. Porto Alegre, 2008. Monografia - Departamento de Informática, Pontifícia Universidade Católica do Rio Grande do Sul.

GLOVER, F. **Future paths for integer programming and links to artificial intelligence**, Computer Operational Research 13, pp. 533-549., 1986.

GUPTA, R.; HARROLD, M. J.; SOFFA, M. L. **Program Slicing-Based regression testing techniques, Software Testing, Verification and Reliability**, Vol 6, Number 2, 1996.

HARMAN, M.; JONES, B. F. **Search based software engineering**. Information and Software Technology: 833–839, Dezembro. 2001.

HARROLD, M. J.; GUPTA, R.; SOFFA, M. L. **A methodology for controlling the size of a test suite**. ACM Transactions on Software Engineering and Methodology, Vol. 2, Issue 3, 1993.

HOLLAND, J.H. **Adaptation in natural and artificial systems**. The University of Michigan Press, AnnArbor, 1975.

KIRKPATRICK, S., GELATT, C. e VECCHI, M. **Optimization by simulated annealing**, vol. 220, pp. 671-680, 1983.

KIM, J.M, PORTER, A. **A history-based test prioritization technique for regression testing in resource constrained environments**. In ICSE '02: Proceedings of the 24th International Conference on Software Engineering, pages 119–129, New York, NY, USA, 2002. ACM.

MANSOUR, N.; BAHSSON, R.; BARADHI, G. **Empirical comparison of regression test selection algorithms**, The Journal of Systems and Software 57, 2001.