UMA ABORDAGEM MULTI-OBJETIVA PARA O PROBLEMA DE SELEÇÃO DE DEFEITOS

Tarciane de Castro Andrade, Fabrício Gomes de Freitas, Márcia Maria Albuquerque Brasil, Thiago Gomes Nepomuceno da Silva, Daniel Pinto Coutinho, Jerffeson Teixeira de Souza

Grupo de Otimização em Engenharia de Software (GOES.UECE)
Universidade Estadual do Ceará (UECE)
Fortaleza, Ceará, Brasil
tarcianeandrade@gmail.com, fabriciogf.uece@gmail.com, marcia.abrasil@gmail.com, thi.nepo@gmail.com, daniel.pintocoutinho@gmail.com, jeff@larces.uece.br

RESUMO

Várias falhas humanas ocorrem durante o desenvolvimento de sistemas. Os testes são essenciais para melhorar a qualidade do software e possuem como resultado uma lista de defeitos encontrados devido aos erros inseridos durante o desenvolvimento. A atividade de corrigir os defeitos é custosa e necessita de uma abordagem para selecionar aqueles defeitos que utilize a melhor forma de distribuir os recursos, o esforço e o planejamento das futuras versões. Este artigo tem por objetivo formular matematicamente o problema de selecionar um conjunto de defeitos para serem corrigidos. A formulação do problema foi realizada através de uma abordagem multi-objetiva de otimização matemática. A abordagem é resolvida em instâncias de tamanho real e mostra que as metaheurísticas utilizadas são boas formas de resolver o problema.

PALAVRAS-CHAVE. Defeitos de Software. Meta-Heurística. NSGA-II. MOCell.

MH – Metaheuristicas, OA - Outras aplicações em PO, OC - Otimização Combinatória.

ABSTRACT

Várias falhas humanas ocorrem durante o desenvolvimento de sistemas. Os testes são essenciais para melhorar a qualidade do software e possuem como resultado uma lista de defeitos encontrados devido aos erros inseridos durante o desenvolvimento. A atividade de corrigir os defeitos é custosa e necessita de uma abordagem para selecionar aqueles defeitos que utilize a melhor forma de distribuir os recursos, o esforço e o planejamento das futuras versões. Este artigo tem por objetivo formular matematicamente o problema de selecionar um conjunto de defeitos para serem corrigidos. A formulação do problema foi realizada através de uma abordagem multi-objetiva de otimização matemática. A abordagem é resolvida em instâncias de tamanho real e mostra que as metaheurísticas utilizadas são boas formas de resolver o problema.

KEYWORDS: Software defects, Metaheuristic. NSGA-II. MOCell.

1. Introdução

Durante o desenvolvimento de sistemas, como em qualquer atividade realizada por humanos, é provável que falhas possam ocorrer. Ao final do processo de desenvolvimento, as falhas ocorridas fazem com que o sistema produzido não seja aquele desejado pelo cliente. Nesse contexto, as falhas podem ter acontecido em diferentes fases do desenvolvimento, como levantamento de requisitos, análise de sistemas, projeto e implementação, por exemplo. Nesse sentido, é importante que o desenvolvimento do software seja acompanhado de atividades de testes de forma que as falhas humanas possam ser identificadas e corrigidas e, como conseqüência, o software produzido funcione conforme o especificado pelos clientes.

O padrão IEEE 610.12 (IEEE, 1990) define defeito como sendo um passo, processo ou definição de dados incorretos; como, por exemplo, uma instrução ou comando incorreto. Nesse escopo, o teste de software pode ser definido como o processo de executar um programa de uma forma controlada com o objetivo de encontrar defeitos, revelando que o funcionamento do software em uma determinada situação não está de acordo com o esperado (Koscianski, 2007). Assim, um teste bem-sucedido identifica defeitos que ainda não foram descobertos.

Segundo (Padua, 2009), um processo de testes deve contemplar, de forma geral, as atividades relativas ao planejamento, preparação, realização e avaliação dos testes. Entre as atividades realizadas em testes está a identificação de defeitos, seguida da solicitação de correções. Dado o conjunto de defeitos encontrados, é importante planejar a atividade de correção, pois restrições de tempo, recursos e orçamento podem impedir que todos os defeitos encontrados possam ser corrigidos. Dessa forma, é necessário selecionar determinados defeitos de forma a garantir maior abrangência, maior importância, satisfação dos clientes, e ao mesmo tempo, otimizar o tempo gasto na atividade de correção de defeitos.

A importância da tarefa de seleção de defeitos para correção reside no fato de que uma seleção incorreta pode contribuir para maior insatisfação dos clientes. Outro aspecto relevante na atividade de seleção de defeitos trata do uso eficiente dos recursos, como o esforço necessário para corrigi-los. O efeito colateral direto desse aspecto é a má utilização do esforço necessário em função do enfoque na correção de defeitos menos prioritários para o cliente (Caetano, 2007).

Do exposto, o problema da seleção de defeitos para correção se mostra um problema que deve ser resolvido com a aplicação de técnicas de otimização matemática. Em geral, os problemas dessa forma são caracterizados por apresentarem resolução não trivial por humanos, pois fatores matemáticos estão envolvidos. Assim, a forma adequada para resolução desses problemas é a utilização de um método automatizado e não exaustivo. Através disso, as pessoas envolvidas com o processo de desenvolvimento podem se preocupar com atividades onde a capacidade da inteligência humana é mais conveniente (Freitas, 2009).

Nesse contexto, nos últimos anos, foi apresentada a utilização de métodos de otimização matemática para a resolução de problemas da Engenharia de Software em várias fases do desenvolvimento, como planejamento de projeto (Colares, 2009) (Alvim, 2009), teste de software (Yoo, 2007) (Freitas, 2010), estimativa de software (Kirsopp, 2002), entre outros. Tal utilização permite a solução de problemas complexos de forma computacionalmente eficiente e precisa. A área resultante do contexto é denominada *Search-based Software Engineering* (SBSE).

Neste artigo, formulamos um novo problema nessa área: seleção de defeitos para correção. Para isso propomos uma formulação multiobjetiva, pois mais de um aspecto está presente no processo de seleção. Os aspectos utilizados na modelagem são: severidade, importância e esforço (tempo). A severidade de um defeito é um fator técnico que define cada defeito (aspectos técnicos do sistema, como a arquitetura ou componente), a importância trata da participação dos clientes na decisão de quais defeitos devem ser corrigidos (o quão importante é a correção do defeito para o cliente), e o aspecto de esforço é utilizado de forma que o resultado da otimização realizada seja o mais eficiente possível (utilize o menor esforço entre a equipe). Um aspecto presente na formulação que faz com que a mesma seja mais real trata da existência de precedência entre defeitos, ou seja, alguns defeitos dependem da correção prévia de outros.

As contribuições desse trabalho são:

- Uma formulação multiobjetiva para o problema da seleção de defeitos para correção, considerando os objetivos severidade, importância, e tempo;
- Resolução do problema com o uso de metaheurísticas multiobjetivas e realização de avaliação empírica para demonstrar a aplicabilidade da formulação apresentada e a eficiência da técnica proposta na resolução do problema com uso de métricas.

O presente trabalho está organizado da seguinte forma. A seção 2 descreve trabalhos relacionados com a área de defeitos de software. A seção 3 apresenta a formulação o problema de seleção de para correção e apresenta breve análise da complexidade do problema. A seção 4 explica de forma resumida as metaheurísticas utilizadas nos experimento, NSGA-II (Deb, 2000) e MOCell (Nebro, 2009). A seção 5 apresenta a avaliação empírica avaliada de forma a comparar a sua solução com as abordagens NSGA-II e MOCell. Finalmente, a seção 6 indica as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Nessa seção apresentamos algumas abordagens de priorização da correção de defeitos utilizadas na literatura e alguns trabalhos relacionados na área.

A atividade de gerenciamento de defeitos de software possui um papel importante no planejamento das versões a serem disponibilizadas para os clientes. Em (Alvim, Barros, Neto, 2009), os autores propõem uma abordagem com heurística híbrida para planejar as tarefas de correção de erros do sistema para os desenvolvedores de forma que minimize o custo para corrigi-lo. Os autores consideram custo como sendo três fatores: minimizar o esforço requerido para a próxima versão do sistema, garantir que os erros mais urgentes sejam corrigidos na próxima versão e maximizar a taxa de ocupação da equipe de testes. A abordagem proposta constrói um cronograma viável para um determinado conjunto de tarefas de correção de erros com prioridades diferentes para cada desenvolvedor.

Algumas ferramentas de registro e rastreamento de defeitos existentes (Bugzilla, 2010) (Mantis, 2010) (Scarab, 2008) possibilitam a classificação dos defeitos de acordo com aspectos, como severidade e prioridade. Essas ferramentas permitem que os desenvolvedores associem a cada defeito um valor representativo de sua importância. A maioria das abordagens para priorização de defeitos considera esses dois tipos de classificação (prioridade e severidade). Prioridade indica a ordem de correção do defeito (defeitos com alta prioridade são corrigidos imediatamente ou num curto prazo de tempo) e severidade é o impacto do defeito no desenvolvimento da aplicação (Bugzilla, 2010) (Whalen, 2009)(Zeller, 2009) (Caetano, 2007). Outros trabalhos indicam que prioridade é a urgência de correção de defeito na visão do cliente e severidade é o impacto do defeito na funcionalidade requerida pelo cliente (Kelsey, 2006)(Tsiu, 2004). Para caracterizar os defeitos, o presente trabalho usa os conceitos de severidade como fator técnico e de importância como fator dos clientes.

Outras abordagens apresentam uma classificação mais completa: severidade (impacto no desenvolvimento), frequência (percentual de ocorrência do defeito) e importância (visão do cliente) (Horch, 2003). Cada uma das classificações possuem valores para os defeitos que podem variar entre: crítico, urgente, médio e baixo, por exemplo. Uma diferença do presente trabalho está no fato que permitimos a participação mais ativa de todos os cliente envolvidos no sistema, pois cada cliente define um nível de importância para cada defeito.

Do exposto, o presente trabalho diferencia-se dos demais devido à melhor formulação do problema. Um ponto existente em nossa formulação trata da possibilidade de dependência entre os defeitos. Além disso, a abordagem de otimização matemática não foi utilizada para o problema em sua formulação.

3. Problema da Seleção de Defeitos para Correção

A partir da descrição inicial do problema da seleção de defeitos para correção apresentada anteriormente, pretende-se nessa seção desenvolver uma descrição formal desse problema, a qual irá permitir a formulação do mesmo como um problema de otimização multiobjetiva.

3.1 Descrição Formal do Problema

Considere inicialmente um conjunto, D, de defeitos, descobertos durante a fase de testes de um dado sistema. Seja N o número de defeitos contidos no conjunto D. Denota-se como D_i, o defeito i do conjunto D, com i variando de 1 até N.

Cada defeito $D_i \in D$, possui uma severidade, S_i, que representa um fator técnico da complexidade de correção do defeito. Outro aspecto dos defeitos trata do tempo necessário para sua correção, $tempo_i$. Esse aspecto resulta em uma função conflitante com as outras já que para atingir maior severidade ou importância (descrita abaixo) deve-se no geral necessitar de mais tempo para a correção. Essa característica faz com que o problema não seja trivial para resolução, pois a resolução deve ser feita de acordo com esse comportamento conflituoso entre as funções.

O sistema é utilizado por um conjunto C de m clientes. Cada cliente possui um valor de estima para a empresa, $estima_i$, que indica o quão importante o cliente é para a empresa. Além disso, cada cliente define um valor de preferência para cada defeito, indicando sua prioridade na atividade de correção. A variável $preferência_{i,j}$ indica a preferência que o cliente i tem pelo defeito j. Tal valor é considerado em conjunto com o valor de estima do cliente, tendo em vista a seleção dos requisitos de maior preferência e dos clientes com maior estima.

Além disso, o problema é composto por duas restrições: tempo máximo, e dependência. A restrição de tempo máximo representa o valor de orçamento existente na empresa. A restrição de dependência trata da necessidade de correção de determinado defeito para que outro possa ser corrigido, o que é uma situação real do contexto do problema. Assim, caso o defeito dependente seja selecionado é necessário que o defeito o qual o mesmo depende seja também selecionado. Esses fatores adicionam maior complexidade ao problema, dificultando a sua resolução por humanos ou estratégias exaustivas, pois a restrição de dependência, por exemplo, insere maior dificuldade para a resolução.

Nesse contexto, o objetivo da resolução do problema é encontrar um conjunto de defeitos de forma que os aspectos de severidade e importância sejam maximizados, enquanto o tempo seja minimizado. Além disso, as restrições de tempo máximo e dependência devem ser respeitadas.

3.2 Formulação Multiobjetiva

A seguir apresentamos a formulação multiobjetiva do problema da seleção de defeitos para correção. A formulação é composta por três funções de otimização (maximizar severidade, maximizar *score*, minimizar tempo e duas restrições - tempo máximo e dependência entre os defeitos):

$$maximizar \sum_{i=1}^{m}\sum_{j=1}^{n} estima_i * preferência_{i,j} * X_j$$

$$minimizar \sum_{j=1}^{n} tempo_j * X_j$$

$$maximizar \sum_{j=1}^{n} severidade_j * X_j$$

$$Sujeito\ a:$$

$$\sum_{j=1}^{n} tempo_j * X_j \leq T$$

$$X_i \leq X_j, \qquad se\ D_i\ depende\ de\ D_j$$

- $estima_i$ é o valor do cliente i para a empresa
- $preferência_{i,j}$ é a prioridade que o cliente i tem para o defeito j
- $tempo_j$ é o tempo necessário para corrigir o defeito j
- $severidade_j$ é o fator técnico de severidade do defeito j
- X_j é a variável booleana de seleção do defeito j. Se i depende de j, a seleção de i (X_i=1) faz com que j seja selecionado ($X_j \geq 1 \rightarrow X_j = 1$), pois trata de um valor booleano.
- T é o esforço máximo para a atividade de correção

---

4. Otimização Multiobjetiva

Nessa seção são apresentados os conceitos referentes básicos à otimização multiobjetiva. Inicialmente, a teoria essencial da área é apresentada e em seguida os algoritmos utilizados no presente artigo são descritos.

4.1 Abordagens Multiobjetivas

Como apresentado na seção anterior, o problema da seleção defeitos para correção é composto por mais de uma função de otimização. Essa abordagem permite a definição de soluções que sejam boas simultaneamente em relação aos aspectos considerados.

Alternativamente, um problema de otimização pode ser formulado com apenas uma função objetivo. Tal tipo de abordagem, chamada mono-objetivismo, tem a característica da existência de uma solução que seja a melhor para o problema considerado. Esta melhor solução, denominada o ótimo global, é aquela que apresenta o melhor valor para a função objetivo.

Contudo, a maioria dos problemas reais não possui apenas um aspecto a ser otimizado. No problema tratado neste artigo, por exemplo, são definidas três características que devem ser consideradas na otimização. Além disso, os aspectos existentes em problemas geralmente são conflitantes, ou seja, a melhoria em um aspecto pode representar piora em outros. Nesse contexto, o conceito de ótimo global presente no mono-objetivimo perde seu sentido essencial, tendo em vista que em problemas multiobjetivos não existe apenas uma função sob otimização e que, dado o conflito entre as diferentes funções, não é possível definir uma única solução que seja a melhor em todos os aspectos.

Para resolver a questão da não existência de ótimo global quando há mais de uma função, as abordagens de otimização multiobjetivas apresentam o conceito chamado de Frente de Pareto. Este conceito representa um conjunto com as melhores soluções possíveis para o problema considerando todos os objetivos ao mesmo tempo. Este conjunto é formado por soluções que não são piores que nenhuma outra considerando todos os objetivos. O fato de ser pior, ou melhor, em relação a outras soluções, conhecido como dominância, pode ser usado para uma definição formal da Frente de Pareto: as soluções pertencentes à Frente de Pareto são aquelas que não são dominadas por nenhuma outra solução, ou seja, não existem soluções melhores simultaneamente em todos os aspectos.

Outro ponto fundamental em relação à otimização multiobjetiva trata da incapacidade de comparação entre as soluções da Frente de Pareto. Não é possível realizar uma comparação entre, por exemplo, duas soluções da Frente para definir qual é a melhor, pois cada uma é melhor em um determinado objetivo. Dessa forma, percebe-se que em problemas com mais de uma função objetivo, não existe apenas uma solução ótima, mas um conjunto de soluções consideradas boas.

4.2 O NSGA-II

A metaheurística NSGA-II (*Non-dominated Sorting Genetic Algorithms II*) (Deb, 2000) é uma técnica de otimização multiobjetiva desenvolvida em 2000 e baseada nos Algoritmos Genéticos.

O NSGA-II funciona da seguinte forma: a partir de uma população inicial, operadores genéticos de mutação e recombinação são aplicados para a geração de uma população *offspring*. As duas populações são reunidas para a escolha de quais indivíduos continuaram no sistema. Tal seleção é feita inicialmente pela seleção das soluções que não são dominadas por nenhuma outra, e depois das soluções eu são dominadas por apenas uma solução do conjunto. O método age assim sucessivamente até que não seja possível incluir todas as soluções não dominadas por n soluções para a próxima população. Nesse caso, um algoritmo de distância é utilizado para selecionar as soluções que implicam em maior diversidade com a sua inclusão. Ao final, o processo é repetido de forma evolutiva. Assim, o algoritmo usa aspectos de intensificação e diversificação.

4.3 MOCell

O MOCell (Nebro, 2009) é um algoritmo genético para resolver problemas de otimização multiobjetivo. Os indivíduos são alinhados em uma grade bidimensional e os operadores genéticos são excessivamente aplicados a eles até encontrar uma condição de parada. Por isso, para cada indivíduo, o algoritmo consiste em selecionar dois pais da sua vizinhança, recombinando-os em ordem a obter um filho, aplicando o operador de mutação a ela, avaliando o indivíduo resultante e inserindo-o na população auxiliar, caso ele não seja dominado pelo indivíduo atual, e na frente de Pareto.

Depois de cada geração, a população anterior é substituída pela população auxiliar e um procedimento de realimentação é chamado para trocar um número fixo de indivíduos aleatoriamente escolhidos da população por soluções no arquivo.

5. Avaliação Empírica

Nessa seção apresentamos a metodologia dos testes realizados assim como os resultados encontrados e a análise dos mesmos.

5.1 Descrição dos Dados

Para os experimentos realizados, foram utilizadas três instâncias de diferentes tamanhos para que os experimentos representem a aplicabilidade da abordagem em diferentes contextos. A quantidade de defeitos é entre 50 e 300, representando contextos gerais. A Tabela 1 a seguir indica as configurações de cada instância.

Tabela 1: Significados dos valores em cada fator considerado.

Instância	Quantidade de Clientes	Quantidade de Defeitos
A	10	50
B	15	100
C	20	300

Cada defeito possui um valor de severidade associado ao mesmo. Este valor é um número entre 1 e 5, com significado mostrado na Tabela 2.

Tabela 2: Significados dos valores em cada fator considerado.

Valor de Severidade	Significado
1	Baixo
2	Regular
3	Média
4	Alta
5	Crítico

Como apresentado na formulação, cada cliente possui um valor de estima para a empresa. Consideramos esse valor um número entre 1 e 10, em que 1 significa baixa estima e 10 representa a maior estima possível. Além disso, cada cliente define um valor de importância para cada defeito. Esse valor é um número entre 1 e 5, com significado apresentado na Tabela 3.

Tabela 3: Significados dos valores em cada fator considerado.

Valor de Importância	Significado
1	Insignificante
2	Inferior
3	Moderada
4	Importante
5	Urgente

Cada defeito também possui um valor estimado de tempo para sua correção. No presente estudo, consideramos a escala entre 1 e 10, em que 1 representa a menor unidade de estimativa e 10 o valor máximo.

Os aspectos restantes tratam das restrições. O valor máximo de tempo considerado foi

de 70% do tempo necessário para corrigir todos os defeitos. A dependência entre os defeitos foi tratada considerando que 10% dos defeitos do conjunto podem apresentar dependência, valor encontrado aproximadamente no contexto prático.

5.2 Metodologia

Inicialmente foram gerados valores aleatórios para cada fator para efeito de simulação. Tais valores servem para o presente estudo empírico. Eles, em essência, representam qualquer contexto no qual o problema pode ocorrer.

Para a resolução das soluções com a abordagem proposta, foi usado o framework JMetal (Durillo, 2006). Assim, cada instância foi resolvida utilizando as duas metaheurísticas: NSGA-II e MOCell. Elas foram selecionadas tendo em vista o vasto uso do NSGA-II, e os recentes bons resultados do MOCell na literatura (Durillo, 2006). Ambas foram executadas de forma independente, 100 vezes cada, e posteriormente comparadas a partir dos resultados de *hypervolume*, *spread* e tempo de execução.

O *hypervolume*, para problemas multiobjetivos onde todos os objetivos devem ser minimizados, calcula o volume coberto pelos membros do conjunto de soluções não-dominadas. Assim, um maior valor de *hypervolume* é desejável, visto que ele indica que as soluções geradas estão próximas da Frente de Pareto Ótima. O *spread*, por sua vez, mede a diversidade das soluções geradas. Assim, um conjunto de soluções com *spread* próximo de zero é preferível, já que isso indica que as solução estão uniformemente distribuídas.

Os parâmetros utilizados para a metaheurística NSGA-II em cada teste foram: tamanho da população de 100, quantidade de gerações de 1000, quantidade de avaliações de 100.000, e taxa de cruzamento de 90% e taxa de mutação de 5%. As configurações para a metaheurística MOCell foram semelhantes: matriz populacional de 100 por 100, arquivo externo de tamanho 100, e número de gerações de 1000, quantidade de avaliações de 100.000 e mesmas configurações de cruzamento de mutação do NSGA-II. O tipo de crossover realizado é o Single Point Crossover e procedimento de mutação Bit Flip Mutation.

A abordagem aleatória, como o nome indica, resolve o problema indicando uma ordem gerada de forma randômica. Esta forma de resolução está presente no estudo para apresentação de um resultado base de comparação (Harman, 2007). Consideramos uma solução aleatória a cada solução gerada pelo NSGA-II e MOCell.

5.3 Apresentação e Análise dos Resultados

As tabelas a seguir mostram os resultados das métricas de desempenho para as metaheurísticas aplicadas. Os valores indicam a média entre as 100 execuções, com a variação do desvio-padrão. A Tabela 4 apresenta os resultados das metaheurísticas utilizadas nas três instâncias do problema. Os dados são mostrados com a média e desvio padrão de 100 execuções das metaheurísticas em cada instância.

Tabela 4: Resultados do tempo (em milisegundos) para NSGA-II e MOCell.

Instância	NSGA-II	MOCell
A	6226.34 ± 1410.90	8905.14 ± 1973.52
B	9478.56 ± 1956.29	11580.50 ± 2867.04
C	45374.60 ± 7844.93	50366.60 ± 5418.39

As Tabelas 5 e 6 apresentam, respectivamente, o *hypervolume* e o *spread* dos métodos. Novamente, os dados são referentes a 100 execuções das metaheurísticas nas três instâncias.

Tabela 5: Resultados do *hypervolume* para NSGA-II e MOCell.

Instância	NSGA-II	MOCell
A	0.4222 ± 0.0032	0.4247 ± 0.0037
B	0.4497 ± 0.0031	0.4481 ± 0.0035
C	0.4063 ± 0.0056	0.3884 ± 0.0052

Tabela 6: Resultados do *spread* para NSGA-II e MOCell.

Instância	NSGA-II	MOCell
A	0.5036 ± 0.0408	0.4662 ± 0.0373
B	0.3999 ± 0.0330	0.3332 ± 0.0377
C	0.3552 ± 0.0279	0.2986 ± 0.0291

Para indicar os resultados de forma visual, uma frente gerada por cada metaheurística para cada instância foi utilizada na construção de gráficos apresentados na Figuras 1, 2 e 3. Para que esses gráficos sejam interpretados, é necessário que se façam algumas considerações iniciais. Como discutido anteriormente, o problema tratado neste trabalho objetiva a maximização de três funções. Entretanto, a visualização de um gráfico tridimensional no plano não se mostra adequada dada a dificuldade de leitura do mesmo. Nesse sentido, apresentamos os resultados, para cada instância, na forma de três gráficos, sendo que cada um tem os resultados referentes a dois objetivos. Os eixos são dispostos de forma a permitir a visualização global dos gráficos. O significado é que eles representam o eixo para o qual estão paralelos. Além disso, vale ressaltar que para as funções de maximização os gráficos são apresentados de forma adaptada, como se fossem gráficos de minimização, no sentido de permitir uma melhor visualização da Frente de Pareto. Assim, os valores em cada eixo nas funções de maximização representam o oposto do resultado da solução das funções matemáticas apresentadas na formulação original do problema.

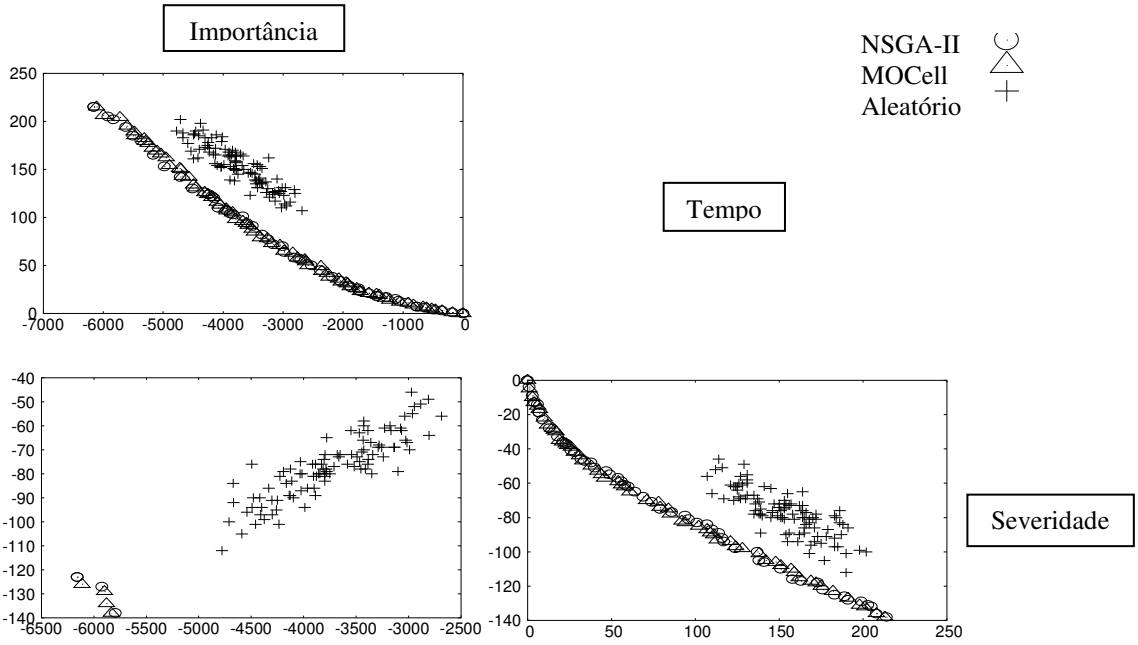

Figura 1: Resultados de NSGA-II, MOCell e Aleatório na instância A.

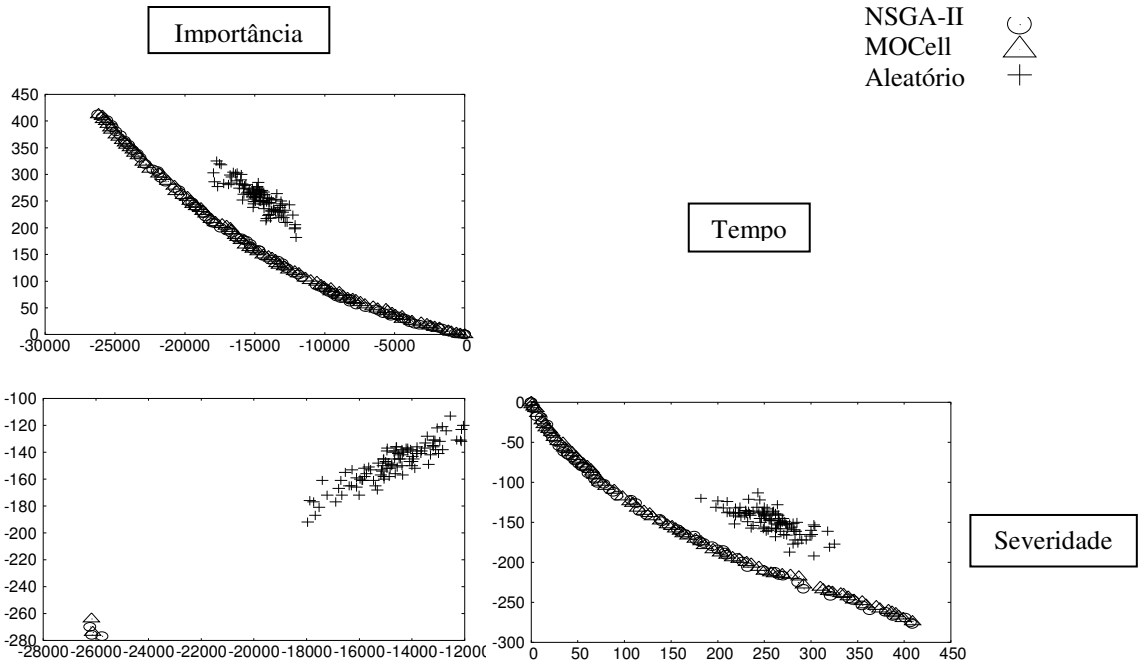

Figura 2: Resultados de NSGA-II, MOCell e Aleatório na instância B.

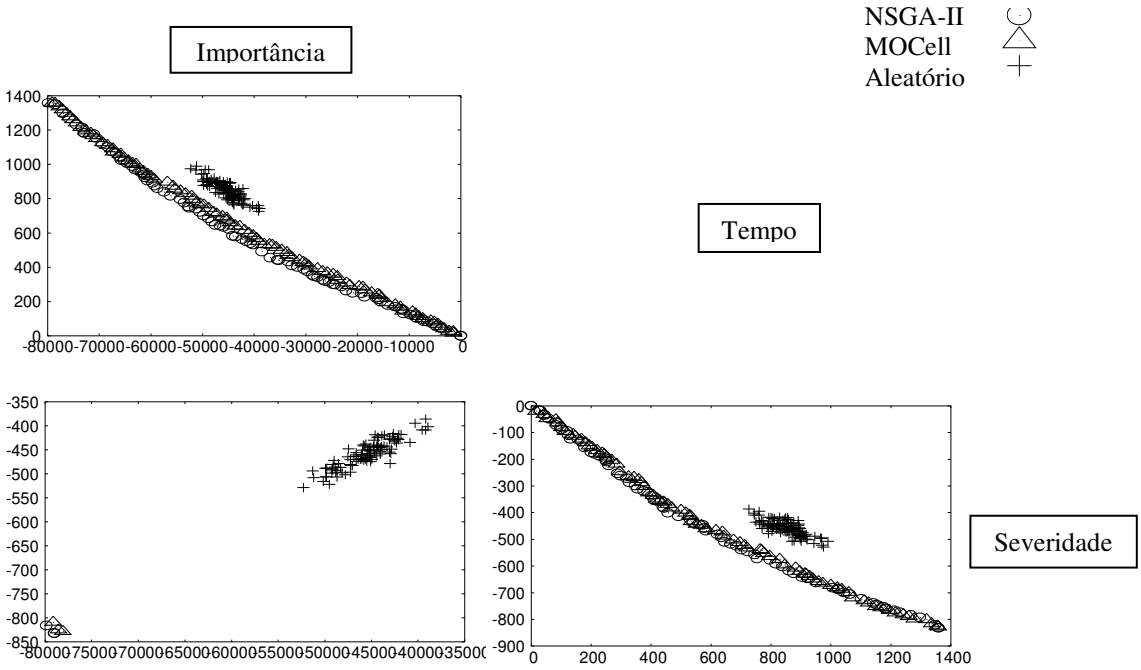

Figura 3: Resultados de NSGA-II, MOCell e Aleatório na instância C.

Em relação ao desempenho das metaheurísticas, a análise da Tabela 4 indica que a metaheurística MOCell apresenta maior tempo para a resolução das instâncias do problema. Na instância A, por exemplo, o valor médio de execução da metaheurística MOCell foi 43,02% maior que o da metaheurística NSGA-II. Nas instâncias B e C os percentuais de acréscimo são 22,17% e 11,00%, respectivamente. No caso da métrica *hypervolume*, MOCell apresentou melhor resultado apenas na instância A, com *hypervolume* maior em 0,59% que o da NSGA-II. Nas instâncias B e C a metaheurística NSGA-II apresentou melhores resultados de *hypervlume* e superou a MOCell em 0,35% e 4,60%. Os resultados de *spread* apresentados na Tabela 6 mostram que a metaheurística MOCell teve melhor desempenho nessa métrica, pois seus valores foram inferiores em 7,42%, 16,67% e 15,93% em relação aos apresentados pelo NSGA-II nas instância A, B e C, respectivamente.

Os resultados obtidos nos experimentos mostram a capacidade e viabilidade da aplicação das metaheurísticas. A busca aleatória foi superada em todas as instância e em todas as configurações de pares de funções, como mostraram as Figuras 1, 2, e 3. A análise das figuras mostra que não são encontradas diferenças fortes em relação às soluções encontradas pelas metaheurísticas.

Do exposto, destacamos que em relação ao tempo de execução a MOCell apresentou piores resultados, mas a análise realizada indica que a diferença de desempenho diminui com o aumento da instância, o que mostra que os métodos se mostram similares em instâncias cada vez maiores. Além disso, no caso da instância C, por exemplo, a diferença de tempo em relação ao NSGA-II é de apenas 11,00%. Em relação à métrica de *hypervolume*, a diferença entre as metaheurísticas foi relativamente pequena. Finalmente, os resultados da métrica spread indicam que MOCell apresenta melhores resultados nas três instâncias. Assim, os métodos se mostraram adequados para resolver o problema tratado nesse artigo. Apesar de similaridades, a metaheurística MOCell se mostra relativamente melhor pois, apesar de ter tempo de execução maior nesse problema, apresenta em geral melhores resultados nas métricas analisadas.

6. Conclusões e Trabalhos Futuros

A fase de teste de software é uma das mais custosas durante o desenvolvimento de software e apresenta como um de seus artefatos a lista de defeitos encontrados no sistema. A partir dessa lista, é necessário indicar a ordem de defeitos para correção de acordo com vários aspectos. Nesse contexto, a correção dos defeitos mais importantes se mostra uma atividade relevante dada à necessidade de alocação de recursos e distribuição de esforço.

Diante da complexidade de resolução do problema, percebe-se que o uso de técnicas automáticas ajuda a resolver o problema de forma eficiente. Nesse artigo, mostramos inicialmente que a metaheurística MOCell se mostra bastante adequada para resolver o problema de seleção de correção de defeitos. Além da melhor qualidade das soluções geradas, a resolução do problema por uma metaheurística multiobjetiva resulta em um conjunto de soluções igualmente boas, o que permite ao gerente de projeto a escolha de uma solução (conjunto de defeitos) mais adequada ao seu contexto.

Como trabalhos futuros, indicamos a utilização de outras técnicas de otimização com o intuito de determinar a existência de outras abordagens adequadas. Apesar de os dados utilizados no presente artigo simularem qualquer contexto, outro aspecto a ser analisado trata da aplicação da abordagem proposta com dados reais.

Referências

Alvim, A.; Barros, M.; Netto, F. (2009) A Hybrid Heuristic Approach for Scheduling Bug Fix Tasks to Software. *1st International Symposium on Search Based Software Engineering*. Windsor: [s.n.]

Bugzilla. (2009) Bugzilla. Disponivel em: <http://www.bugzilla.org/>. Acesso em: 12 jan. 2010.

Caetano, C. (2007) Gestão de Defeitos. Engenharia de Software Magazine, v. 1, n. 1, p. 60-67.
Chandler, L. (2007) Software Defects: Severity, Priority and Impact, Disponivel em: http://scaffadaffa.com/2007/06/software-defects-severity-priority-and.html. Acesso em: 05/01/10
Colares, F., Souza, J., Carmo, R., Pádua, C., Mateus, G. (2009) A New Approach to the Software Release Planning, pp.207-215, *XXIII Brazilian Symposium on Software Engineering*
Deb, K. et al. (2000) A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, V. 6, pp. 182-197.
Durillo, J. J., et al. (2006) JMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics, *Technical Report* ITI-2006-10, University of Málaga.
Freitas, F. G., Maia, C. L. B., Coutinho, D. P., Campos, G. A. L., Souza, J. T., Aplicação de metaheurísticas em problemas de engenharia de software: Revisão de literatura. II Congresso Tecnológico Infobrasil, 2009.
Freitas, F. G., Maia, C. L. B., Campos, G. A. L., Souza, J. T., Otimização em Teste de Software com Aplicação de Metaheurísticas, Revista Sistemas de Informação, Edição 5, 2010.
Harman, M. (2007) The Current State and Future of Search-based Software Engineering. *Proceedings of International Conference on Software Engineering / Future of Software Engineering 2007*, Minneapolis Minnesota USA, pp. 342-357.
IEEE. (1990) IEEE Standard glossary of software engineering terminology, Stantard 610.12. IEEE Press.
Kelsey, R. B. (2006) Software Project Management: Measures for Improving Performance. [S.l.]: Management Concepts, 92-93 p. ISBN 978-1567261738.
Kirsopp, C., et al. (2002) Search heuristics, case-based reasoning and software project effort prediction, *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, pp. 1367-1374.
Horch, J. W. (Ed.). (2003) Practical Guide to Software Quality Management. 2ª. ed. [S.l.]: Artech House, 2003. ISBN 9781580535274.
Koscianski, A., Soares, M. (2007) Qualidade de Software. Ed. 2. Editora Novatec, ISBN 8575221124.
Mantis. (2009) MantisBT Group, Disponivel em: <http://www.mantisbt.org/bugs/main_page.php>. Acesso em: 3 jan. 2010.
Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B. & Alba, E.. (2009) MOCell: A Cellular Genetic Algorithm for Multiobjective Optimization. *International Journal of Intelligent Systems* , 24, 7, Julho.
Padua, **W.** (2009) Engenharia de Software: Fundamentos, Métodos e Padrões. 3. ed. Rio de Janeiro: LCT, ISBN 9788521616504.
Pressman, R. (2006) Engenharia de Software. [S.l.]: Makron Books.
Scarab. (2008) Collabnet Enterprise. Disponivel em: <http://scarab.tigris.org/>. Acesso em: 3 janeiro 2010.
SIX SIGMA. (2008) Failure Modes and Effects Analysis (FMEA). Disponivel em: <http://www.isixsigma.com/tt/fmea/>. Acesso em: 05 jan. 2010.
Tsui, F. (2004) Managing Software Projects. [S.l.]: Jones & Bartlett Publishers, pp. 59-64 ISBN 9780763725464.
Whalen, **D**. (2009) Software Testing. Severity vs Priority, 13 nov. Disponivel em: <http://www.softwaretestingclub.com/profiles/blogs/severity-vs-priority>. Acesso em: 05 jan. 2010.
Yoo, S., Harman, M. (2007) Pareto Efficient Multi-Objective Test Case Selection, *Proceedings of the International Symposium on Software Testing and Analysis*, pp. 140-150.
Zeller, A. (2009) Why Programs Fail: A Guide to Systematic Debugging. 2ª. ed. [S.l.]: Elsevier Science & Technology Books, 32-34 p. ISBN 978-1558608665.