BRANCH-AND-BOUND APLICADO NA SELEÇÃO MULTIOBJETIVA DE REQUISITOS DE SOFTWARE COM DEPENDÊNCIA

Fabrício Gomes de Freitas
Grupo de Otimização em Engenharia de Software (GOES.UECE)
Universidade Estadual do Ceará (UECE) - Fortaleza, Ceará, Brasil
fabriciogf.uece@gmail.com
Thiago Gomes Nepomuceno da Silva
Grupo de Otimização em Engenharia de Software (GOES.UECE)
Universidade Estadual do Ceará (UECE) - Fortaleza, Ceará, Brasil
thi.nepo@gmail.com
Rafael Augusto Ferreira do Carmo
Grupo de Otimização em Engenharia de Software (GOES.UECE)
Universidade Federal do Ceará (UFC) - Fortaleza, Ceará, Brasil
carmorafael@gmail.com
Márcia Maria Albuquerque Brasil
Grupo de Otimização em Engenharia de Software (GOES.UECE)
Universidade Estadual do Ceará (UECE) - Fortaleza, Ceará, Brasil
marcia.abrasil@gmail.com
Jerffeson Teixeira de Souza
Grupo de Otimização em Engenharia de Software (GOES.UECE)
Universidade Estadual do Ceará (UECE) - Fortaleza, Ceará, Brasil
jeff@larces.uece.br

RESUMO

Neste trabalho é discutida a aplicabilidade de técnicas exatas de otimização em problema da engenharia de requisitos. Restrições podem impedir que todas as funcionalidades desejadas pelos clientes possam ser implementadas para a próxima versão. Assim, a seleção de quais requisitos devem ser implementados se faz necessária. Na literatura, metaheurísticas já foram empregadas para a resolução desse problema. Trabalhos anteriores não consideraram a existência de dependência entre requisitos. O presente trabalho tem o intuito de investigar o desempenho de técnicas exatas de otimização matemática como Branch-and-Bound na resolução do problema com dependência, com a vantagem de que tal resolução garante a definição das melhores soluções e a restrição de dependência torna a formulação mais real. A partir dos resultados encontrados, indica-se a aplicabilidade de técnicas exatas em tem prático, além da eficácia em relação a outras formas de resolução, incluindo soluções de especialistas de software.
PALAVRAS-CHAVE. Requisitos. Branch-and-Bound. Otimização Multiobjetiva.
MH – OA -Outras aplicações em PO, PM – Programação Matemática, TEL&SI - PO em Telecomunicações e Sistemas de Informações.

ABSTRACT

In this paper we discuss the applicability of exact optimization techniques in a requirements engineering problem. Restrictions may prevent all features desired by customers to be implemented for the next version. Thus, the selection of which requirements should be implemented is necessary. In literature, metaheuristics have been employed to solve this problem. Previous work have not consider the existence of dependency among requirements. This study aims to investigate the performance of exact optimization techniques such as Branch-and-Bound in solving the problem with the dependency, with the advantage that this resolution ensures the best solutions and the constraint of dependence makes the formulation more realistic. From results, we indicate the applicability of exact techniques in practical time, and as expected the superior efficacy compared to other resolution ways, including software experts.
KEYWORDS: Software Requirements. Branch-and-Bound. Multiobjective Optimization.

1. Introdução

Os avanços trazidos pela área de Engenharia de Software são notáveis para o avançado atual estágio de desenvolvimento de sistemas. Através de décadas de pesquisa, foram definidos modelos, normas e metodologias de suporte ao processo desenvolvimento de software (DYBA, 2005), tendo em vista que a qualidade do produto final é fortemente relacionada à qualidade do processo de produção (FUGGETTA, 2000).

Infelizmente, em alguns casos, tais metodologias não são capazes de resolver alguns problemas do contexto, ou o fazem de modo insatisfatório. Isso acontece em problemas intrinsecamente complexos. Em problemas desse tipo formas automatizadas de resolução devem ser usadas para que o problema possa ser resolvido de maneira eficiente (FREITAS et al. 2009).

Uma importante área presente no desenvolvimento de software é Engenharia de Requisitos. Tal fase contém problemas de alta complexidade, como o problema da definição de quais requisitos devem ser implementados para a próxima versão do sistema, considerando a satisfação do cliente, custos de implementação e restrições de orçamento (BRASIL et al., 2010).

Tal problema foi atacado na literatura (ZHANG; HARMAN; MANSOURI, 2007) com uso de metaheurísticas, que não garantem a descoberta das melhores soluções. O presente trabalho propõe a resolução do problema da seleção de requisitos através da técnica Branch-and-Bound, com a vantagem de que as melhores soluções possam ser encontradas. Isso condiz com a necessidade de soluções de qualidade para uso no processo de desenvolvimento de software (YOO; HARMAN, 2007).

Com a realização do trabalho pretende-se apresentar o uso da otimização exata na engenharia de requisitos. Para isso, comparações de eficiência e eficácia são realizadas entre a técnica exata e metaheurísticas multiobjetivas.

2. Trabalhos Relacionados

O problema da seleção de requisitos foi apresentado em 2001 (BAGNALL; RAYWARD-SMITH; WHITTLEY, 2001), com maximização da satisfação dos clientes, e respeitando o limite de orçamento. Os autores aplicaram técnicas de otimização em cinco instâncias do problema. As metaheurísticas utilizadas pelos autores foram Têmpera Simulada, Hill-Climbing, além de uma técnica gulosa. Nos experimentos foi constatado que no geral a técnica Têmpera Simulada obteve melhores resultados em comparação com as outras técnicas.

O problema da seleção de requisitos foi formulado de forma multiobjetiva em 2007 (ZHANG; HARMAN; MANSOURI, 2007), com a maximização da satisfação dos clientes e a minimização do custo de implementação dos requisitos. A função de importância considera o nível de importância que cada cliente tem por cada requisito. Os autores realizaram testes com metaheurísticas multiobjetivas como NSGA-II e mostraram que as mesmas eram capazes de resolver o problema, ainda que sem a garantia de encontrar o melhor conjunto de soluções.

Em relação ao trabalho relacionado indicado, a presente pesquisa estende seu uso ao inserir a existência de dependências entre requisitos. Tal fator não foi considerado na modelagem de 2007, apesar de ser um aspecto recorrente durante o desenvolvimento de software. Além disso, uma importante contribuição do presente trabalho trata da modelagem e resolução do problema com o uso de otimização exata. Tal abordagem instiga o uso de técnicas exatas em outros problemas de otimização da Engenharia de Software, como seleção de casos de teste, gerenciamento de projetos, entre outros.

Outra contribuição do trabalho reside na indicação da validade da utilização de técnicas de otimização em comparação aos resultados encontrados por profissionais da área. O trabalho mostra que, de fato, as soluções encontradas pelas técnicas automáticas de otimização são melhores que as encontradas por especialistas em engenharia d software para os problemas complexos tratados nessa área. Outro objetivo do trabalho trata da apresentação da aplicabilidade de otimização exata em problemas de otimização matemática com mais de um objetivo (otimização multiobjetiva) na área. Pela pesquisa bibliográfica realizada na área de Otimização em Engenharia de Software (*Search based Software Engineering*), tal abordagem de otimização exata em cenário multiobjetivo ainda não foi realizada na área.

3. Fundamentação Teórica

Nessa seção são descritos conceitos e definições utilizados no trabalho. Inicialmente, na seção 3.1 apresentamos um breve resumo sobre a área na qual a presente pesquisa se encaixa (Otimização em Engenharia de Software). A seção 3.2 indica conceitos sobre otimização matemática, incluindo a otimização multiobjetiva. O problema atacado no trabalho é formalizado na seção 3.3.

3.1 Otimização em Engenharia de Software

A Engenharia de Software, como uma disciplina de engenharia, é uma área em que problemas com características matemáticas podem ser encontrados (HARMAN, 2006). Os aspectos matemáticos encontrados geralmente podem ser usados na definição de problemas de otimização que ocorrem durante o desenvolvimento de software. Tais problemas em geral possuem aspectos como elevada quantidade de soluções no espaço de busca, alta complexidade estrutural do problema, e existência de restrições, que dificultam o processo de resolução de forma manual. Então, a resolução para esses problemas desse tipo deve ser feita com métodos automatizados (CLARKE et al., 2003) (HARMAN; JONES, 2001).

Na engenharia de software, o primeiro passo nessa direção aconteceu em 1976, quando o problema de geração de dados de teste foi atacado através de maximização numérica (MILLER; SPOONER, 1976). Contudo, foi em um segundo momento, em 2001, que um trabalho apresentou definições para esta abordagem de resolução para problemas da Engenharia de Software e denominou o campo de pesquisa como *Search-based Software Engineering* (SBSE) (HARMAN et al., 2001). Desde então, pesquisadores e engenheiros de software intensificaram a modelagem de problemas complexos da Engenharia de Software como problemas de otimização matemática, principalmente em Teste de Software (FREITAS et al., 2010).

A engenharia de requisitos apresenta destacada importância visto que é nessa fase em que o software em si é definido, e, por isso, as atividades realizadas neste ponto têm impacto em todas as fases seguintes do desenvolvimento. Por motivos de restrição orçamentária, em geral não é possível implementar todas as funcionalidades do sistema desejadas pelo cliente. Assim, a seleção de quais requisitos devem ser implementados para a próxima versão consiste de uma atividade relevante nessa fase.

3.2 Otimização Matemática

Os problemas de otimização matemática são aqueles com funções a serem maximizadas ou minimizadas e definidas a partir de coeficientes e variáveis. As variáveis que definem a função podem estar sujeitas a restrições, ou seja, as variáveis devem satisfazer um conjunto de equações definidas de acordo com cada instância do problema. De maneira formal, o problema de otimização é definido como:

$$Minimizar\ \{f_1(x), f_2(x), \dots, f_k(x)\}$$

$$Sujeito\ a:$$
$$g_i(x) \leq 0, \qquad i = [i \dots p]$$
$$h_i(x) \leq 0, \qquad i = [i \dots q]$$

Quando apenas uma função deve ser otimizada, denominamos o problema como mono-objetivo. Nesse caso, então, a busca de soluções é realizada de acordo com os valores de somente uma função matemática. Assim, em um problema de minimização, por exemplo, se tomarmos uma solução A com valor da função menor que o valor de uma solução B, a solução A é melhor que B.

Contudo, muitos problemas são caracterizados pela existência de mais uma função que devem ser otimizadas simultaneamente. Nesse caso, otimização multiobjetiva, pode não ser possível definir uma única solução que seja a melhor em todos os aspectos, possivelmente conflitantes (SARAWAGI; NAKAYAMA; TANINO, 1985). Como não há apenas uma solução melhor, utiliza-se um conjunto de boas soluções, denominado Frente de Pareto. Tal conjunto possui as melhores soluções considerando todos os objetivos ao mesmo tempo. Este conceito é chamado de dominância entre soluções, isto é, uma solução x é dominada por y se x é estritamente pior em relação a y em pelo menos uma função. Uma definição formal da Frente de Pareto usando esse conceito é: 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. Por exemplo, considere um problema multiobjetivo com duas funções A e B para minimização e as soluções S1 (20, 10), S2 (10, 10), e S3 (20, 5), como mostra a figura abaixo.

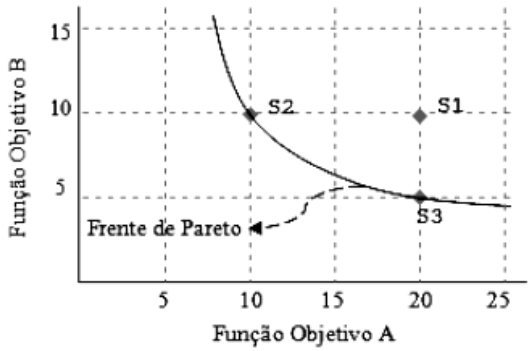

Figura 1: Exemplo ilustrativo de Frente de Pareto.

Entre as três as soluções, S2 e S3 podem pertencer à Frente de Pareto pois não são piores ao mesmo tempo nas duas funções tratadas em relação as outras (são não-dominadas). A solução S1, por sua vez, é pior no objetivo A devido a S2 e pior no objetivo B devido a S3, ou seja, é uma solução dominada e assim não pertence à frente de Pareto.

3.3 Seleção de Requisitos Multiobjetiva

A formulação do problema é multiobjetiva, dado que o problema possui mais de um aspecto a ser otimizado simultaneamente. A multiobjetividade acontece com a otimização tanto da importância dos requisitos quanto do tempo de implementação (ZHANG; HARMAN; MANSOURI, 2007). Na formulação multiobjetiva, a importância considerada no processo tem tanto o fator do cliente como também a importância que cada cliente associa a cada requisito.

Os aspectos que definem o problema são:

- **Clientes**: Seja o conjunto $C = \{c_1, c_2, ..., c_m\}$ formado por m clientes. Cada cliente c_j possui um valor de importância para a empresa. Tal valor de importância representa o nível de estima que aquele cliente representa para a empresa, sendo que valores maiores significam maior importância.
- **Requisitos**: O conjunto $R = \{r_1, r_2, ..., r_n\}$ é formado por n requisitos. Cada requisito r_i tem um custo de implementação $custo_1$, que pode ser estimado em homens-hora.
- **Clientes e Requisitos**: Esse aspecto trata da relação que contém a visão de cada cliente para cada requisito desejado. Nesse sentido, o cliente não apenas indica quais funcionalidades são desejadas, mas também um nível de importância para cada uma. O valor é definido como $value_{i,j}$.
- **Relação entre Requisitos:** Os requisitos têm uma relação de precedência entre si, pois a implementação de alguns requisitos pode depender da implementação prévia de outros.

A seguinte formulação matemática representa o problema:

$$\max \sum_{i=1}^{n} score_i * X_i$$

$$\min \sum_{i=1}^{n} custo_i * X_i$$

$$Sujeito\ a:$$
$$X_i \leq X_j,\ se\ r_i \rightarrow r_j$$

Onde:

$$score_i = \sum_{j=1}^{m} importance_j * value_{j,i}$$

A variável X_i é um valor booleano que indica a seleção ou não do requisito i. Assim, a restrição presente na formulação representa a relação de dependência entre os requisitos, pois, dado que a variável booleana X_i é 1 caso o requisito i é selecionado e 0 caso contrário, tal desigualdade na restrição indica que, caso o requisito i dependa de requisito j ($r_i \rightarrow r_j$), para i ser implementado ($X_i = 1$), X_j deve ser maior ou igual a 1, que no contexto binário indica que X_j deve ser 1, isto é, o requisito j está selecionado pela solução.

3.4 Metaheurísticas

O termo metaheurísticas (GLOVER, 1986) representa uma classe de algoritmos genéricos de busca. Estes métodos utilizam ideias de diversos domínios como inspiração para realizar o processo de busca da solução para problemas de otimização. As abordagens metaheurísticas tentam resolver o problema visitando de forma inteligente algumas soluções, mas não existe garantia de a solução retornada ser a melhor. A seguir apresentamos em resumo o funcionamento das metaheurísticas mono-objetivas utilizadas nesse trabalho.

- **NSGA-II**: A metaheurística NSGA-II (*Non-dominated Sorting Genetic Algorithms II*) (DEB et al., 2000) é uma técnica de otimização multiobjetiva que utiliza os conceitos de permutação e mutação. O NSGA-II inicia com a geração de forma aleatória de um conjunto P_0 de N soluções. Em seguida, um segundo conjunto Q_0, também de tamanho N, é gerado com o uso dos operadores de permutação e mutação. As soluções dos dois conjuntos formam a população R_0 de tamanho 2N. Então, R_0é ordenado usando o conceito de dominância da seguinte maneira: as soluções que não são dominadas por nenhuma outra são colocadas na primeira Frente de Pareto, F1; as soluções que são dominadas por uma solução são colocadas na segunda Frente de Pareto, F2; e assim sucessivamente até que todas as soluções de R_0 estejam em alguma Frente. Então, uma população P_1 é formada considerando as Frentes de Pareto iniciais até o limite de N soluções. Quando este limite impede que todas as soluções de uma Frente sejam tomadas, um operador de distância de soluções (*crowding distance sorting*) é utilizado na seleção de quais serão tomadas para P_1, para selecionar soluções mais distantes entre si e permitir maior nível de diversificação na população. A população P_1 é utilizada como ponto inicial na iteração seguinte, ou seja, uma população será formada, e as soluções das duas populações serão ordenadas de acordo com a dominância para a seleção de quais participarão da próxima geração.

- **MOCell**: O MOCell (NEBRO et al., 2009) é um algoritmo genético multiobjetivo para resolver problemas de otimização. 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.

3.5 Otimização Exata

As técnicas exatas são métodos que utilizam operações matemáticas sobre os dados do problema. O método mais conhecido no caso de problemas com funções e restrições lineares é o método Simplex (DANTZIG, 1960). Tal método utiliza a formulação do problema de forma matricial, e a partir de operações definidas e baseadas em lemas consegue atingir o ótimo global, caso exista. O método é baseado na representação geométrica do problema de otimização em que as equações lineares formam um "politopo" no espaço de busca. De acordo com teoremas da área de programação linear, a solução ótima do problema de otimização é encontrada em um dos vértices do politopo considerado como o sistema de equações lineares da formulação. Assim, a partir de uma solução inicial válida, geralmente no vértice situado na origem do espaço, o método Simplex visita os vértices adjacentes na busca de valores melhores para a função objetivo. Tal abordagem também é baseada no lema que indica que a quantidade de vértices do politopo para um problema de otimização linear é um número finito e, assim, a busca terá fim.

O processo de visita aos vértices adjacentes até que seja encontrado o ótimo global é realizado por operações matriciais incluindo a troca entre as variáveis básicas do sistema de equações. Isso é realizado com manipulações entre as equações do sistema linear que representa o modelo. No presente trabalho, a versão utilizada é a revisada do Simplex na forma do produto da inversa. Para a resolução de problema de Programação Inteira é utilizado o método *Branch-and-Bound* (LAND, DOIG, 1960). Tal abordagem inicialmente resolve o problema utilizando métodos lineares ou não-lineares (dependendo do problema tratado) sem a restrição de solução inteira com o intuito de encontrar os limites (*bounds*) da solução. Além disso, o problema é dividido em sub-problemas em estrutura de árvore (*branch*) de acordo com divisões dos domínios das variáveis, e as divisões que se apresentam piores que o limite são descartadas. O processo é realizado até que os sub-problemas tenha sido considerados.

No caso de técnicas exatas em contexto multiobjetivo, consideramos as funções, com exceção de uma, como restrições do problema. Assim, são efetuadas diversas resoluções do problema, uma para cada valor do domínio da restrição. Como o problema é otimizado em relação a outra função objetivo, a solução encontrada a cada execução é a melhor para aquela configuração. Ao final, o conjunto de soluções formado a partir de todas as execuções com variação do valor de limite da restrição consiste da frente de Pareto do problema. Este processo faz com que não seja necessária a ponderação entre funções. Contudo, essa abordagem faz com que a quantidade de execuções possa limitar, dependendo do contexto, o uso de tal abordagem para problemas com poucas funções objetivo. Outro aspecto desejado nessa abordagem é que o domínio das funções a serem tratadas como restrições deve ser inteiro, para permitir a descoberta de todas as soluções e assim a formação completa da Frente de Pareto.

4. Metodologia

Foram gerados dados aleatórios para a representação de diversos contextos. Apesar de serem dados artificiais, isso não prejudica o presente estudo tendo em vista que os dados aleatórios representam uma simulação de um contexto qualquer onde os problemas podem acontecer e, assim, servem como base para a simulação.

No caso do problema atacado, (MONRP, sigla para *Multiobjective Next Release Problem*), as instâncias analisadas estão indicadas na Tabela 1.

Tabela 1: Instâncias utilizadas nos experimentos.

Instância	Quantidade de Requisitos	Quantidade de Clientes
MONRP-A	20	10
MONRP-B	40	20
MONRP-C	80	50
MONRP-D	100	100
MONRP-E	200	100
MONRP-F	400	100

As instâncias MONRP-A, MONRP-B e MONRP-C foram usadas para definir o comportamento dos métodos em contextos pequenos a médios. A partir da instância MONRP-D, os contextos representam o aumento na quantidade de requisitos, quando analisaremos o comportamento das metaheurísticas.

Os valores de domínio para as instâncias seguem as indicações do artigo original correspondente ao problema (ZHANG; HARMAN; MANSOURI, 2007). Assim, os intervalos considerados na geração dos dados são:

- Peso dos clientes: entre 1 e 10
- Custo dos requisitos: entre 1 e 9
- Nível de importância de requisitos por cada cliente: entre 0 e 5 (nível 0 indica que o cliente não deseja tal requisito)

Para a relação de dependência, consideramos que até 10% dos requisitos podem estar em alguma dependência. Tais relações foram definidas aleatoriamente.

A resolução por metaheurísticas é realizada com o *framework* jMetal (DURILLO et al., 2006). Cada metaheurística foi executada 100 vezes. O resultado de uma das 100 execuções é apresentado no sentido de indicar o comportamento (resultados similares nas outras execuções). As metaheurísticas NSGA-II e MOCell foram escolhidas nesse estudo dado o amplo uso da primeira e aos recentes resultados promissores apresentados pela segunda abordagem. Com o aumento no tamanho da instância, o tempo de execução da técnica exata aumenta consideravelmente (como mostrado nos experimentos). Assim, também é importante realizar um estudo com as metaheurísticas para seu potencial uso em instâncias grandes. A comparação entre esses métodos pode ser feita utilizando-se duas métricas de técnicas multiobjetivas: hypervolume, e spread. O hypervolume calcula o nível que a frente gerada pela metaheurística cobre do espaço de busca, e seu nível de proximidade à frente exata. Assim, valores maiores de hypervolume são desejados. O spread representa a distância média entre as soluções da frente (DURILLO et al., 2006). Valores menores de spread são melhores, visto que é desejado fornecer ao tomador de decisão diversas opções que cubram de forma equivalente o espaço de busca, evitando o acúmulo de soluções em apenas uma região do espaço.

A modelagem e resolução com otimização exata é realizada com o software LINGO (LINGO, 2010). Como indicado na seção anterior, a resolução multiobjetiva com otimização exata foi baseada na múltipla execução do problema, cada vez com uma configuração diferente no limite da restrição (no caso, utilizamos a função de custo como restrição). Tal método é baseado no €-constraint.

Na resolução por humanos, as soluções dos profissionais foram coletadas em formulários dos problemas. No total, 21 pessoas resolveram a instâncias A (MONRP-A). Nas instâncias B, MONRP-B, o número de participantes atingiu 13 especialistas. Para as outras instâncias dos problemas não foi realizada a resolução por especialistas, tendo em vista a alta complexidade de tais instâncias. Para a versão multiobjetiva, as soluções dos especialistas foram consideradas e estão representadas nos gráficos referentes aos resultados correspondentes apresentados na seção seguinte.

5. Resultados

A seguir apresentamos os resultados para a versão multiobjetiva do problema da seleção de requisitos. Como cada solução é composta por dois valores de funções objetivo, uma maneira apropriada para apresentação dos resultados é por meio de gráficos. Nesse caso, as funções objetivo são representadas no sentido de minimização, pois tal abordagem facilita a visualização do gráfico e da frente de Pareto. Assim, ao invés de usarmos os valores da função de score (que é de maximização), utilizamos o oposto (simétrico) da mesma.

Inicialmente, a figura a seguir mostra o resultado para as instâncias MONRP-A e B. Como pode ser observado, a frente de Pareto é na maior parte compartilhada entre os métodos. Isso acontece pois essa instância em particular tem espaço de busca relativamente pequeno, e assim as metaheurísticas conseguem sem dificuldade encontrar as soluções ótimas. A parte inferior da frente de Pareto, contudo, só tem soluções da otimização exata (OE), indicando sua característica de fornecer todas as soluções ótimas para o problema. Algumas soluções por especialistas também compartilham a frente de Pareto, mas outras soluções estão sendo dominadas.

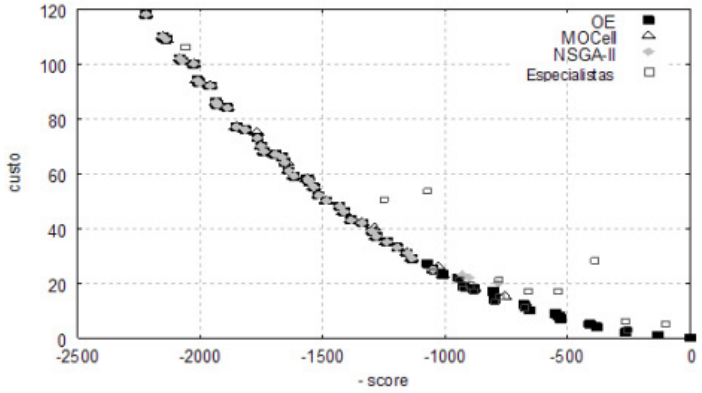

Figura 2: Resultado para instância MONRP-A.

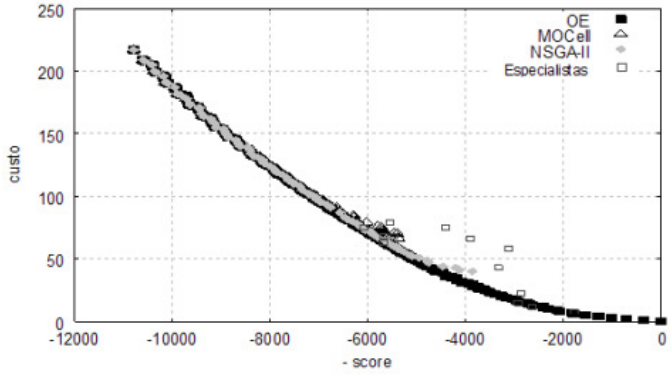

Figura 3: Resultado para instância MONRP-B.

No caso da B, percebe-se que as soluções das metaheurísticas estão mais afastadas, apesar de sutilmente, da frente de Pareto ótima encontrada pela otimização exata. Algumas soluções por especialistas também continuam sendo dominadas por soluções da frente de Pareto, isto é, apresentam valores piores. Também pode ser observado que as soluções de especialistas são concentradas na região inferior do espaço de busca, com custo menor, o que mostra a necessidade do uso de otimização para encontrar todas as possibilidades de solução.

A seguir apresentamos os resultados para a instância MONRP-C na Figura 4. Os resultados e gráficos são estruturalmente semelhantes aos da instância MONRP-B.

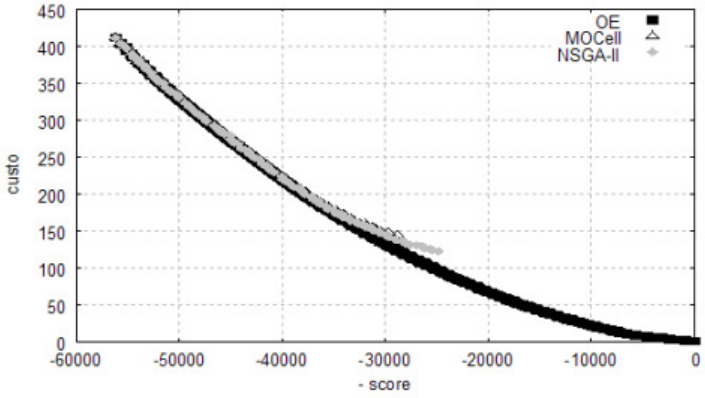

Figura 4: Resultado para instância MONRP-C.

Assim, a acurácia da técnica exata é validada no contexto. Para a melhor visualização dos resultados, apresentamos três gráficos correspondentes a aproximações visuais do resultado mostrado acima: a parte que representa as regiões da frente onde o afastamento das soluções das metaheurísticas é perceptível (1), a parte final das frentes das metaheurísticas, onde são observadas soluções dominadas pela frente da otimização exata (2), e o destaque para a região onde as metaheurísticas não conseguiram encontrar soluções para o problema (3). Tais aproximações visuais são indicadas na Figura 5 abaixo.

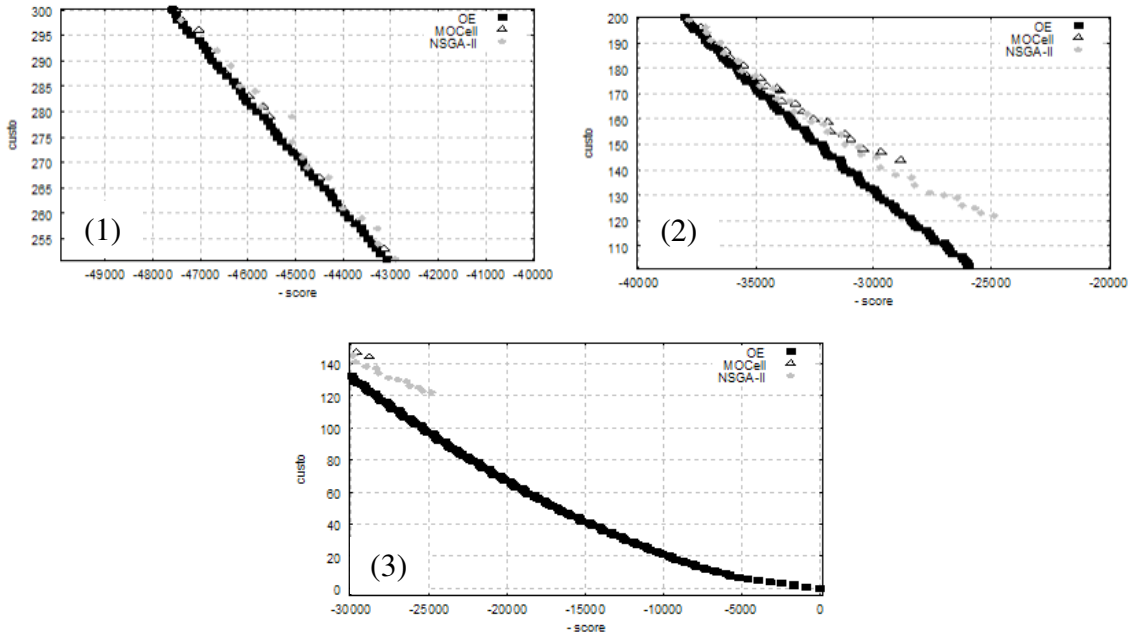

Figura 5: Aproximações do resultado de MONRP-C para melhor visualização.

Em relação ao tempo de execução, a análise para a maior instância usada (MONRP-C) indica que a Otimização Exata demorou 39 segundos para gerar toda a frente de Pareto, enquanto a metaheurística NSGA-II, por exemplo, tomou em média 3,41 segundos para a geração de sua frente aproximada. Um fato que deve ser analisado é o tempo necessário para a geração de cada

solução. Para o NSGA-II, foi gasto em média 0,034 segundos por solução, enquanto na Otimização Exata, que gerou toda a frente de Pareto, encontrando mais soluções que a metaheurística, precisou em média de 0,112 segundos por solução. Assim, pode-se dizer que o tempo da Otimização Exata foi até 3,39 vezes mais lento. Contudo, se observarmos em termos absolutos, a Otimização Exata conseguiu em menos de um minuto gerar todas as soluções ótimas da frente de Pareto, o que se mostra um tempo aceitável para uma solução que define o processo de planejamento de requisitos, atividade que pode durar semanas ou mesmo meses.

Para as três instâncias maiores (D, E, F) apresentaremos os resultados das métricas hypervolume e spread para as metaheurísticas. Tais valores não são apresentados para a otimização exata, visto que os mesmos são utilizados para a comparação entre as metaheurísticas. De fato, a frente exata é utilizada como base no cálculo de tais métricas de desempenho. As tabelas 2 e 3 a seguir trazem esses valores para as três maiores instâncias.

Tabela 2: Resultados de hypervolume para NSGA-II e MOCell.

Instância	NSGA-II	MOCell
MONRP-D	0.63174206	**0.63187358**
MONRP-E	0.60380369	**0.60479774**
MONRP-F	0.54763855	**0.57274789**

Tabela 3: Resultados de spread para NSGA-II e MOCell.

Instância	NSGA-II	MOCell
MONRP-D	0.99812831	**0.61412380**
MONRP-E	0.82486297	**0.65851800**
MONRP-F	0.80460948	**0.71077400**

Percebemos que MOCell apresentou melhores valores para hypervolume (maior que NSGA-II) e spread (menor que NSGA-II). Assim, tal metaheurística se mostra uma opção adequada para a resolução de instâncias muito grandes nas quais o uso do método exato não possa ser feito devido ao tempo de execução. Para encontrar tal tamanho máximo, a seguir apresentamos os resultados de tempo de execução do Branch-and-Bound nas três instâncias.

Tabela 4: Resultados de tempo (segundos) dos métodos.

Instância	Exata	NSGA-II	MOCell
MONRP-D	42,30	9,26	**3,63**
MONRP-E	187,07	19,58	**14,10**
MONRP-F	926,96	139,47	**74,85**

Assim, verificamos o comportamento do método exato. Entre as instâncias MONRP-D, E, e F, a quantidade de requisitos dobra. Tal aspecto é o principal do problema, já que o mesmo trata da seleção de requisitos. O tempo de execução observado chega a mais que quadruplicar entre as instâncias. Entre MONRP-D e E, o tempo fica 4,45 vezes maior. Entre MONRP-E, e F, observamos que o tempo de execução necessário pelo Branch-and-Bound é 4,95 maior. Assim, o uso da técnica exata pode não ser apropriado em contextos com uma quantidade muito maior de requisitos, dado o tempo de execução. Contudo, na maior parte dos casos estudados a técnica é apropriada.

6. Conclusão e Trabalhos Futuros

Dada a importância dos sistemas de software na sociedade atual, é importante que o desenvolvimento dos mesmos seja feito da melhor forma possível. Durante tal desenvolvimento, alguns problemas complexos podem acontecer e a resolução dos mesmos pelos envolvidos se mostra dificultada. Na fase de Engenharia de Requisitos, por exemplo, um problema complexo trata da seleção de quais requisitos devem ser implementados com base em valores de importâncias dos clientes e restrição de orçamento.

Na literatura, tal problema já foi tratado por metaheurísticas, e nesse trabalho apresentamos a proposta da resolução por técnicas exatas, que possuem a vantagem de encontrar as melhores soluções para o problema. A partir dos testes realizados em diversas instâncias de cada problema, percebe-se a validade de tal abordagem pois, além de, como esperado, os resultados da otimização exata terem sido melhores que as metaheurísticas, o tempo de execução da mesma foi razoável. O tempo de execução, contudo, foi verificado como exponencial no tamanho da instância (quantidade de variáveis de decisão; quantidade de requisitos). Assim, foram realizados testes com instâncias grandes para determinar o comportamento das metaheurísticas em tais ocasiões. A partir dos resultados, a metaheurística MOCell se mostrou mais apropriada, pois encontrou uma frente mais próxima a real (maior hypervolume), além de prover soluções mais igualmente espaçadas (menor spread).

O presente trabalho também realizou comparação com soluções de especialistas envolvidos na área. Pelos resultados, percebe-se a necessidade da aplicação das técnicas automáticas e baseadas em otimização matemática na resolução dos problemas, pois em geral o resultado encontrado pelos especialistas se mostrou pior. Percebeu-se que as soluções dos especialistas foram em geral piores que as encontradas pela abordagem proposta nas duas instâncias testadas. Particularmente, apesar de diferentes especialistas, as suas soluções se concentraram em uma região do espaço de soluções.

Como trabalhos futuros indicamos a realização de experimentos em instâncias maiores dos problemas, assim como testes com diferentes estruturas entre os dados com o intuito de demonstrar a validade da abordagem em diferentes cenários. Outra pesquisa adicional trata da realização de testes com dados reais de projetos de software.

A aplicação da otimização exata em outras áreas e problemas da Engenharia de Software onde apenas metaheurísticas já foram aplicadas também é indicada como um desenvolvimento da presente pesquisa.

Referências

Bagnall, A., Rayward-Smith, V. E Whittley, L., The next release problem, Information and Software Technology, 2001, pp. 883–890.

Brasil, M. M. A., Freitas, F. G., Silva, T. G. N., Souza, J. T., Cortés, M. I., Uma Nova Abordagem de Otimização Multiobjetiva para o Planejamento de Releases em Desenvolvimento Iterativo e Incremental de Software, Anais do I Workshop Brasileiro de Otimização em Engenharia de Software, WOES 2010, 2010.

Carmo, R. A. F, Campos. G. A. L., Souza, J. T., Easymeta: a framework of metaheuristics for mono-Objective optimization problems, Anais do XL Simpósio Brasileiro de Pesquisa Operacional (SBPO´2008), 2008.

Clarke, J. et al. Reformulating software engineering as a search problem, IEE Proceedings Software, Vol. 150, No. 3, June 2003, PP. 161-175.

Dantizig, G. B., Inductive proof of the simplex method. IBM J. Res. Development, 505-506.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, V. 6, pp. 182-197, 2000.

Durillo, J. J., Nebro, A. J., Luna, F., Dorronsoro, B. E Alba, E. (2006), jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics, Technical report, ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos.

Dyba, T., An empirical investigation of the key factors for success in software process improvement, IEEE Transactions on Software Engineering, IEEE, May 2005, pp. 410-424.
Fuggetta, A., Software process: a roadmap, The Future of Software Engineering, A. Finkelstein (ed), 2000.
Freitas, F. G.; Maia, C. L. B.; Coutinho, D. P.; Campos, G. A. L.; Souza, J. T.; Aplicação de Metaheurísticas em Problemas da Engenharia de Software: Revisão de Literatura, Anais do II Congresso Tecnológico InfoBrasil (Infobrasil 2009), Fortaleza, 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.
Glover, F., Future paths for integer programming and links to artificial intelligence, Computer Operational Research 13, 1986, pp. 533-549.
Greer, D. E Ruhe, G., Software release planning: an evolutionary and iterative approach," Information & Technology, vol. 46, no. 4, pp. 243–253, 2004.
Harman, M. Search Based Software Engineering, Workshop on Computational Science in Software Engineering, 2006.
Harman, M., Jones, B.F., Search-based software engineering, Information and Software Technology, 2001, pp. 833-839.
Harman, M. E Jones, B.F., The SEMINAL workshop: reformulating software engineering as a metaheuristic search problem, ACM SIGSOFT Software Engineering Notes, Volume 26, Issue 6, Novembro de 2001, PP. 62-66.
Land, A. H., Doig, A. G.; An automatic method of solving discrete programming problems. Econometrica 28(3): 497-520, July 1960.
Lingo, Lingo Systems, http://www.lindo.com/, Acesso em Outubro de 2010.
Miller, W., Spooner, D.L., Automatic generation of floating-point test data, IEEE Transactions on Software Engineering, IEEE, 1976, pp. 223-226.
Nebro, A. J.; Durillo, J. J.; Luna, F.; Dorronsoro, B.; E Alba, E., MOCell: A Cellular Genetic Algorithm for Multiobjective Optimization. International Journal of Intelligent Systems, 24, 7, Julho 2009.
PROJECT MANAGEMENT INSTITUTE, A Guide To The Project Management Body of Knowledge – PMBOK Guide, 2000.
Sarawagi, Y.; Nakayama, H; Tanino, H; Theory of multiobjective optimization, Mathematical optimization, Volume 176, Academic Press (Orlando), 1985.
SEI (SOFTWARE ENGINEERING INSTITUTE), Capability Maturity Model Integration – CMMI, 2005.
SOFTEX, Melhoria do Processo de Software Brasileiro – Guia Geral – MPS.BR, 2006.
Yoo, S. E Harman, M., Pareto Efficient Multi-Objective Test Case Selection, Proceedings of the International Symposium on Software Testing and Analysis, 2007, pp. 140-150
Zhang, Y., Harman, M. E Mansouri, A., The multi-objective next release problem. In Proceedings of the 9th annual conference on Genetic and evolutionary computation (GECCO '07). ACM, New York, NY, USA, 1129-1137, 2007.